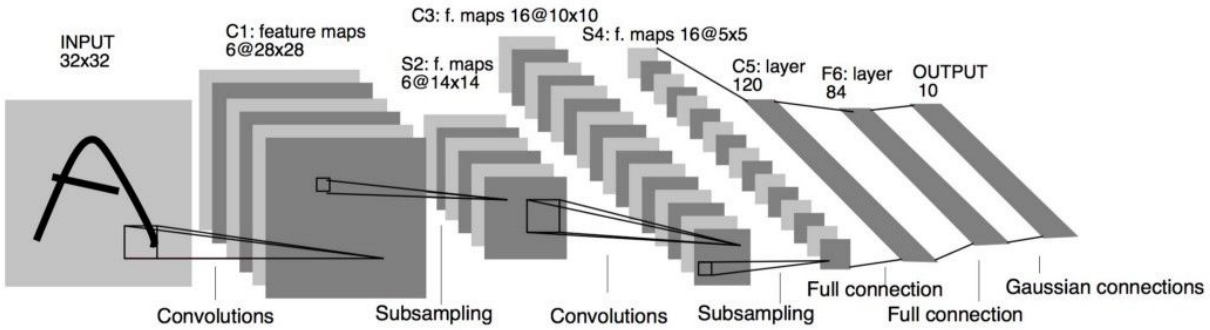
Redes Neurais Convolucionais

Deep Learning

Redes Neurais Convolucionais

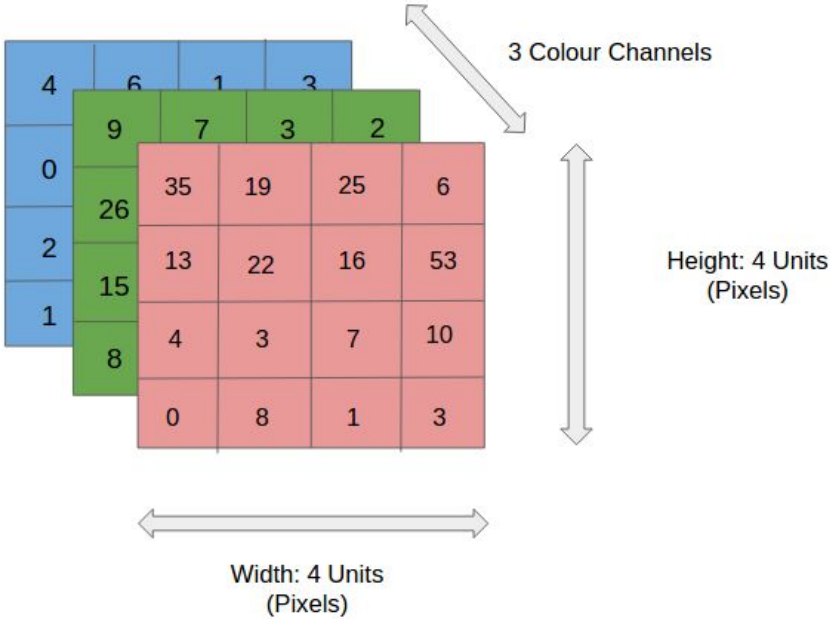
O reconhecimento de imagem é um clássico problema de classificação, e as Redes Neurais Convolucionais possuem um histórico de alta acurácia para esse problema. A primeira aplicação com sucesso de uma CNN foi desenvolvida por Yann LeCun em 1998, com sete camadas entre convoluções e fully connected. Desde então as CNNs ficaram cada vez mais profundas e complexas, como AlexNet em 2012, que, apesar de ter apenas oito camadas (cinco convoluções e três fully connected), apresenta sessenta milhões de parâmetros, e a GoogleNet com vinte e duas camadas e quatro milhões de parâmetros.



Arquitetura da rede LeNet5

Entradas

Quando falamos em reconhecimento/classificação de imagens, as entradas são usualmente matrizes tridimensionais com altura e largura (de acordo com as dimensões da imagem) e profundidade, determinada pela quantidade de canais de cores. Em geral as imagens utilizam três canais, RGB, com os valores de cada pixel.



Etapas de uma CNN

Etapa 1 – Operador de convolução

Etapa 2 – Pooling

Etapa 3 – Flattening

Etapa 4 – Rede neural densa

Etapa 1 – Operador de convolução

- Convolução é o processo de adicionar cada elemento da imagem para seus vizinhos, ponderado por um kernel

- A imagem é uma matriz e o kernel é outra matriz

$$
\begin{aligned}
(f * g)[n] &= \sum_{m=-\infty}^{\infty} f[m]g[n-m] \\
&= \sum_{m=-\infty}^{\infty} f[n-m]g[m].
\end{aligned}
$$

Etapa 1 – Operador de convolução

Explicações sobre os kernels

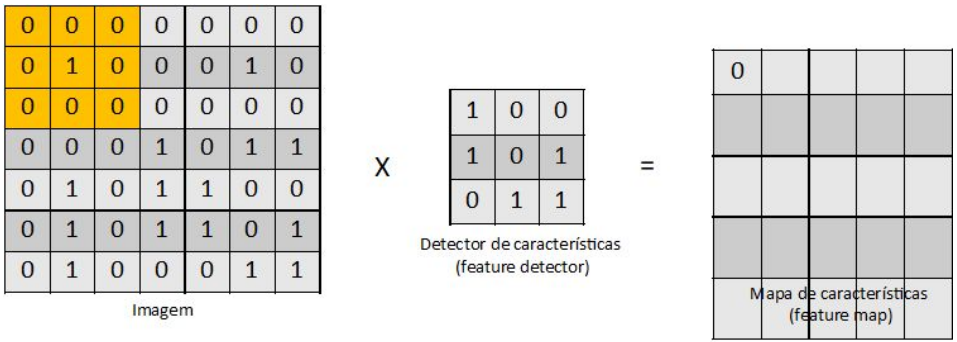
https://en.wikipedia.org/wiki/Kernel_(image_processing)

Exemplo on-line

http://setosa.io/ev/image-kernels/

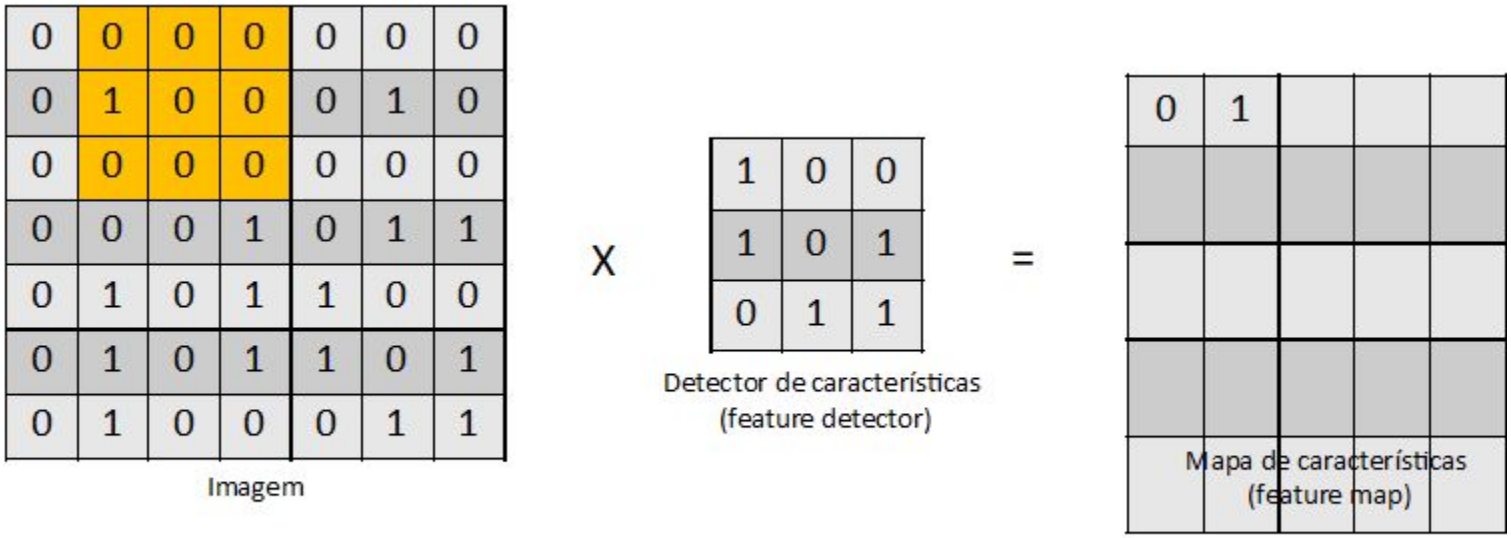
Etapa 1 – Operador de convolução

As convoluções funcionam como filtros que enxergam pequenos quadrados e vão “escorregando” por toda a imagem captando os traços mais marcantes. Explicando melhor, com uma imagem 7x7x3 e um filtro que cobre uma área de 3x3 da imagem com movimento em saltos (chamado de stride), o filtro passará pela imagem inteira, por cada um dos canais, formando no final um feature map ou activation map de 5x5x1.



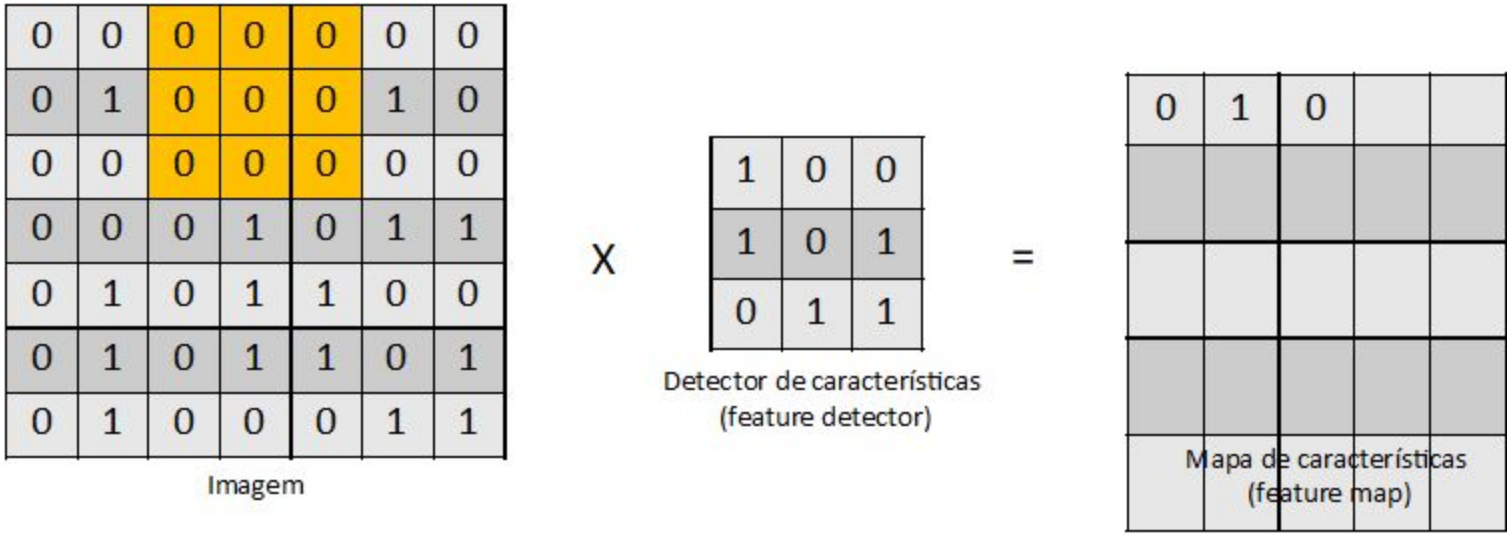
0 * 1 + 0 * 0 + 0 * 0 + 0 * 1 + 1 * 0 + 0 * 1 + 0 * 0 + 0 * 1 + 0 * 1 = 0

Etapa 1 – Operador de convolução



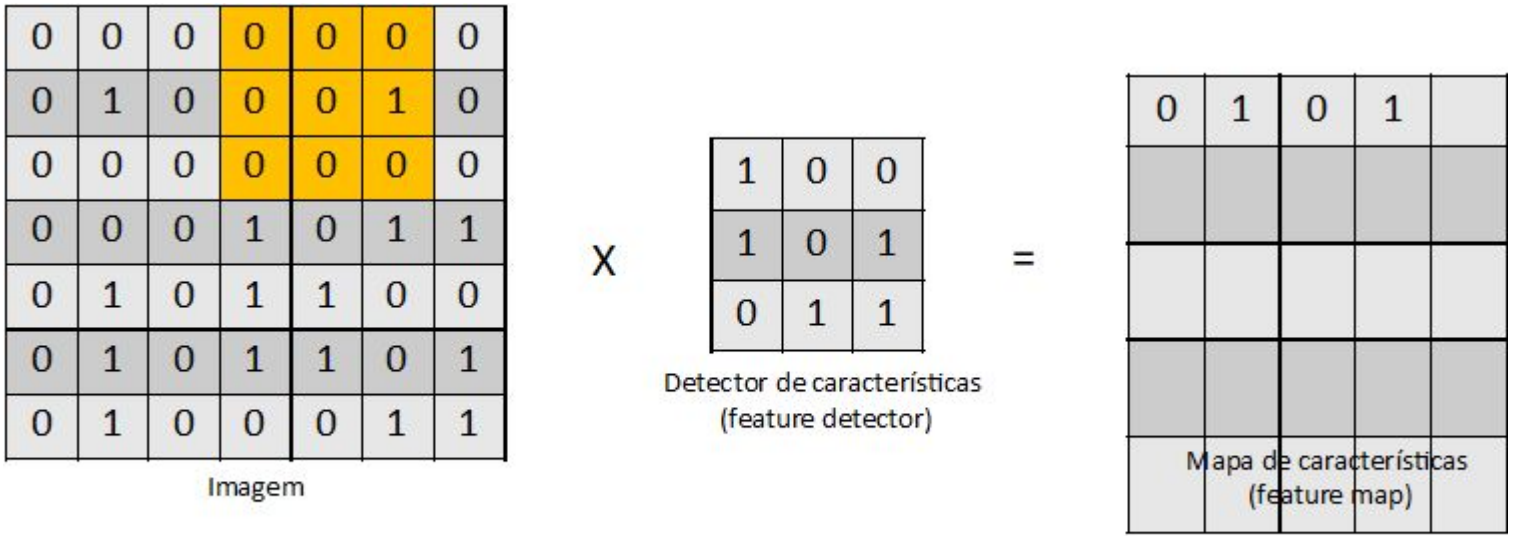
$$0 * 1 + 0 * 0 + 0 * 0 + 1 * 1 + 0 * 0 + 0 * 1 + 0 * 0 + 0 * 1 + 0 * 1 = 1$$

Etapa 1 – Operador de convolução



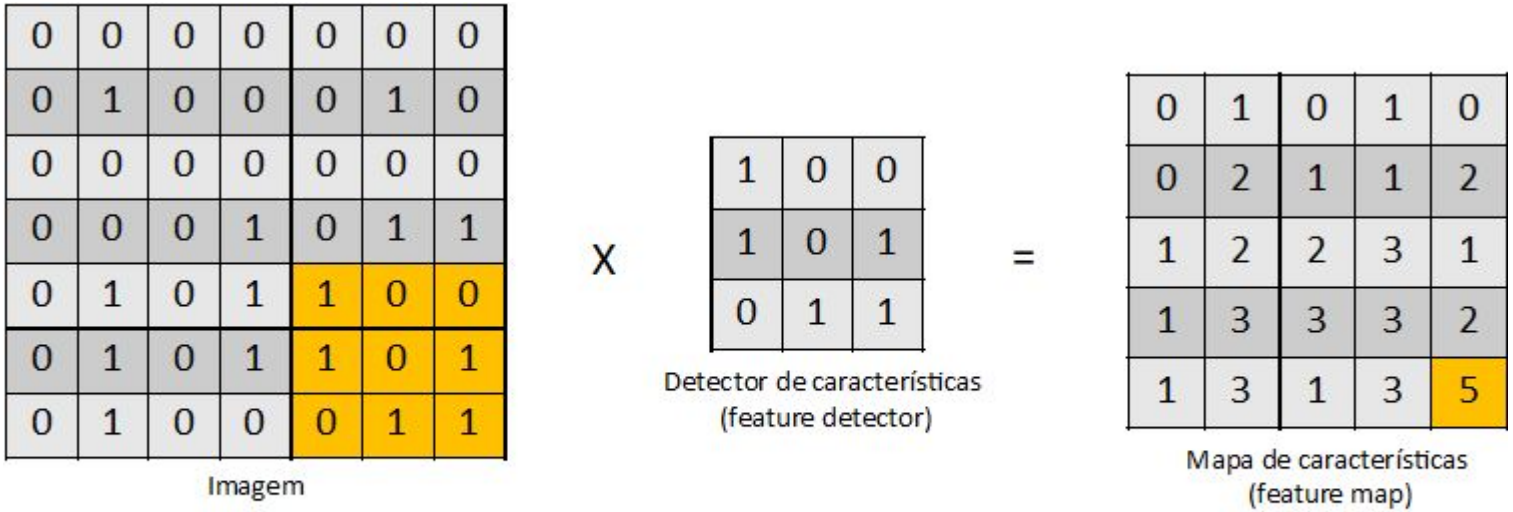
$$0 * 1 + 0 * 0 + 0 * 0 + 0 * 1 + 0 * 0 + 0 * 1 + 0 * 0 + 0 * 1 + 0 * 1 = 0$$

Etapa 1 – Operador de convolução



$$0 * 1 + 0 * 0 + 0 * 0 + 0 * 1 + 0 * 0 + 1 * 1 + 0 * 0 + 0 * 1 + 0 * 1 = 1$$

Etapa 1 – Operador de convolução



$$1 * 1 + 0 * 0 + 0 * 0 + 1 * 1 + 0 * 0 + 1 * 1 + 0 * 0 + 1 * 1 + 1 * 1 = 5$$

Etapa 1 – Operador de convolução

- Com o mapa de características (filter map) a imagem fica menor para facilitar o processamento.

- Alguma informação sobre a imagem pode ser perdida, porém o propósito é detectar as partes principais (quanto maior os números melhor).

- O mapa de características preserva as características principais da imagem (olho, boca, nariz, por exemplo).

Etapa 1 – Operador de convolução

- A profundidade da saída de uma convolução é igual a quantidade de filtros aplicados. Quanto mais profundas são as camadas das convoluções, mais detalhados são os traços identificados com o activation map.

- O filtro, que também é conhecido por kernel, é formado por pesos inicializados aleatoriamente, atualizando-os a cada nova entrada durante o processo de backpropagation. A pequena região da entrada onde o filtro é aplicado é chamada de receptive field.

Etapa 1 – Operador de convolução

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	1	0	1	1	0	0
0	1	0	1	1	0	1
0	1	0	0	0	1	1

Imagem

X

1	0	0
1	0	1
0	1	1

Detector de características
(feature detector)

=

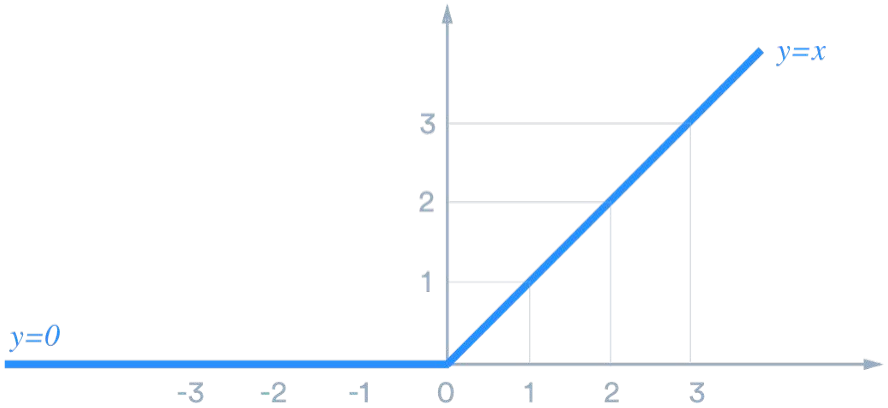
0	1	0	1	0
0	2	1	1	2
1	2	2	3	1
1	3	3	3	2
1	3	1	3	5

Mapa de características
(feature map)

Etapa 1 – Operador de convolução

ReLU

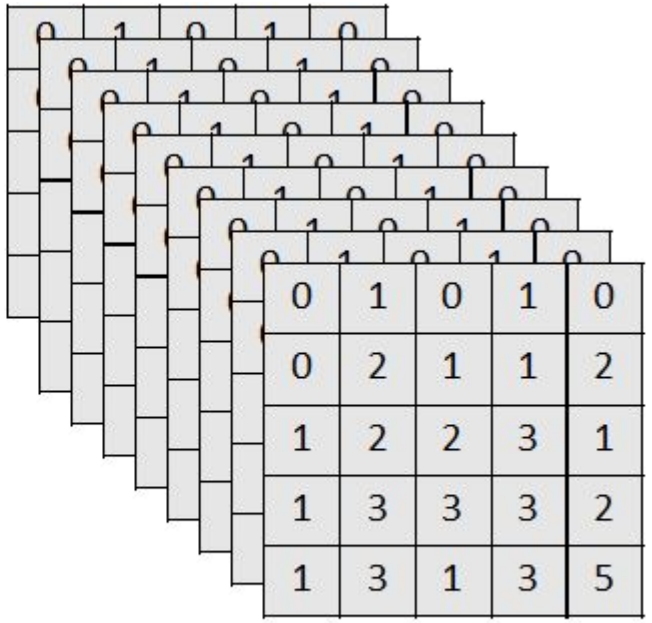
Uma rede neural sem função de ativação torna-se um modelo linear. Se o seu problema é linear, existem outros modelos mais simples que te atenderão tão bem quanto uma rede neural. Infelizmente a maioria dos problemas complexos não são lineares. Portanto, para adicionar a não linearidade a rede, utilizamos as funções de ativação. Nos dias de hoje, e principalmente no contexto de imagens, a mais utilizada é a função ReLU.
Matematicamente a função ReLU é definida como y = max(0, x). O gráfico a seguir é a ilustração desta função.



Etapa 1 – Operador de convolução

0	0	0	0	0	0	0
0	1	0	0	0	1	0
0	0	0	0	0	0	0
0	0	0	1	0	1	1
0	1	0	1	1	0	0
0	1	0	1	1	0	1
0	1	0	0	0	1	1

Imagem



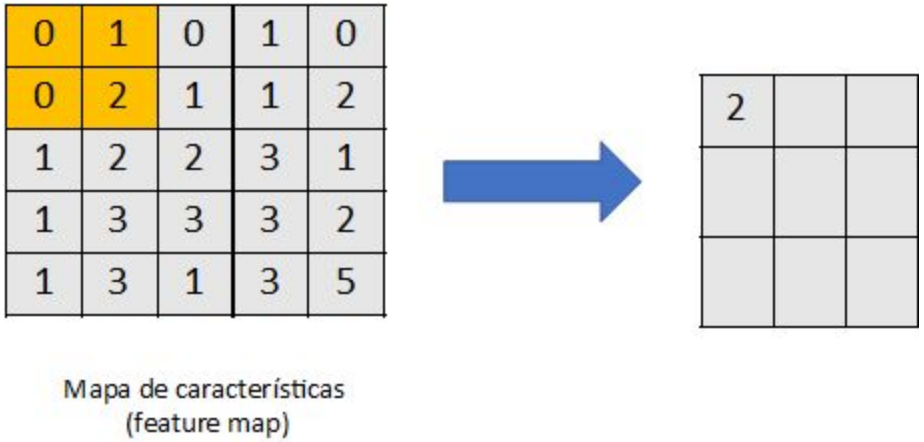
Mapas de características
(feature maps)

A rede decidirá qual detector de características que será utilizado
Camada de convolução é o conjunto de mapa de características

Etapa 2 - Pooling

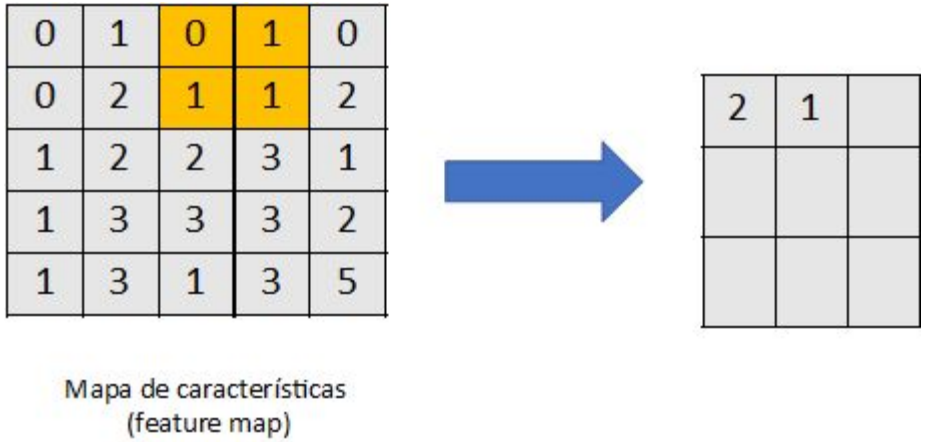
Uma camada de pooling serve para simplificar a informação da camada anterior. Assim como na convolução, é escolhida uma unidade de área, por exemplo 2x2, para transitar por toda a saída da camada anterior. A unidade é responsável por resumir a informação daquela área em um único valor. Se a saída da camada anterior for 5x5, a saída do pooling será 3x3. Além disso, é preciso escolher como será feita a sumarização. O método mais utilizado é o maxpooling, no qual apenas o maior número da unidade é passado para a saída. Essa sumarização de dados serve para diminuir a quantidade de pesos a serem aprendidos e também para evitar overfitting

Etapa 2 - Pooling

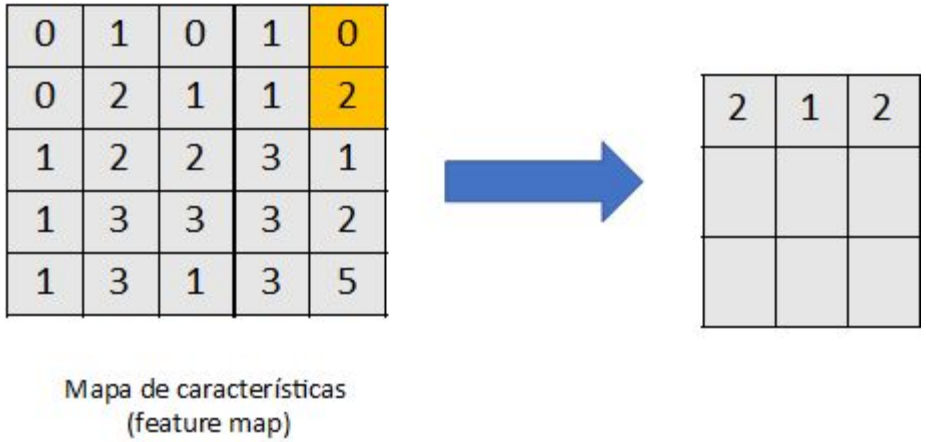


Mapa de características
(feature map)

Etapa 2 - Pooling

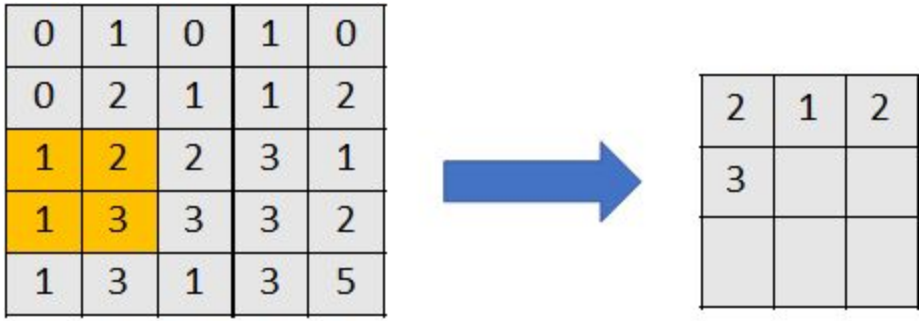


Etapa 2 - Pooling



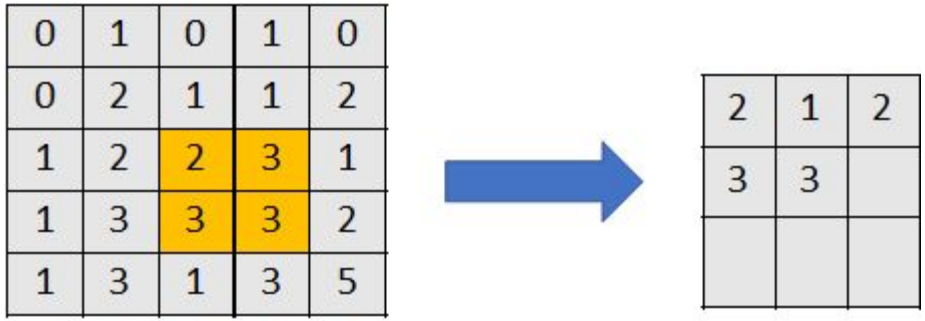
Mapa de características
(feature map)

Etapa 2 - Pooling



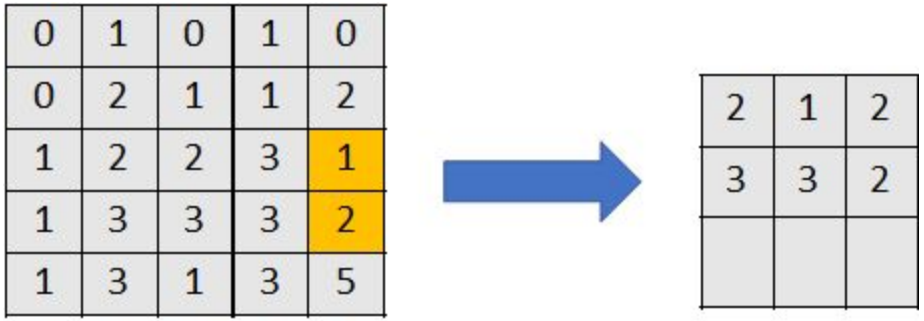
Mapa de características
(feature map)

Etapa 2 - Pooling



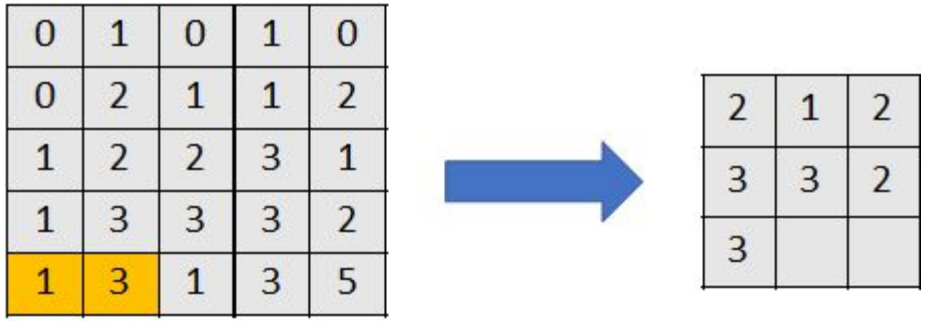
Mapa de características
(feature map)

Etapa 2 - Pooling



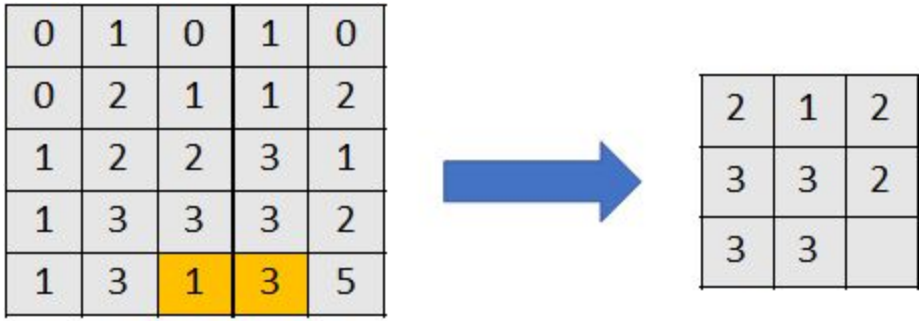
Mapa de características
(feature map)

Etapa 2 - Pooling



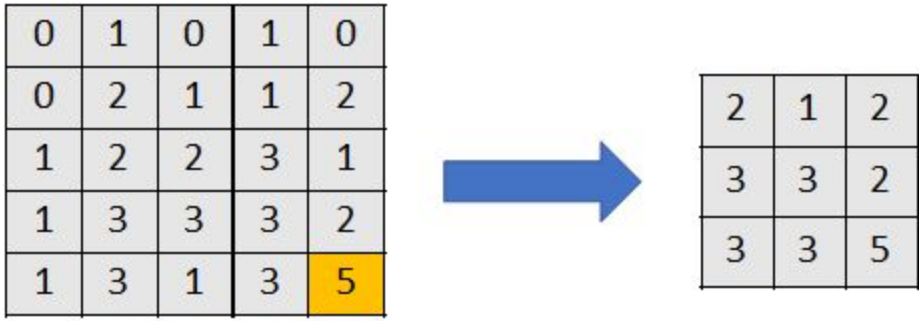
Mapa de características
(feature map)

Etapa 2 - Pooling



Mapa de características
(feature map)

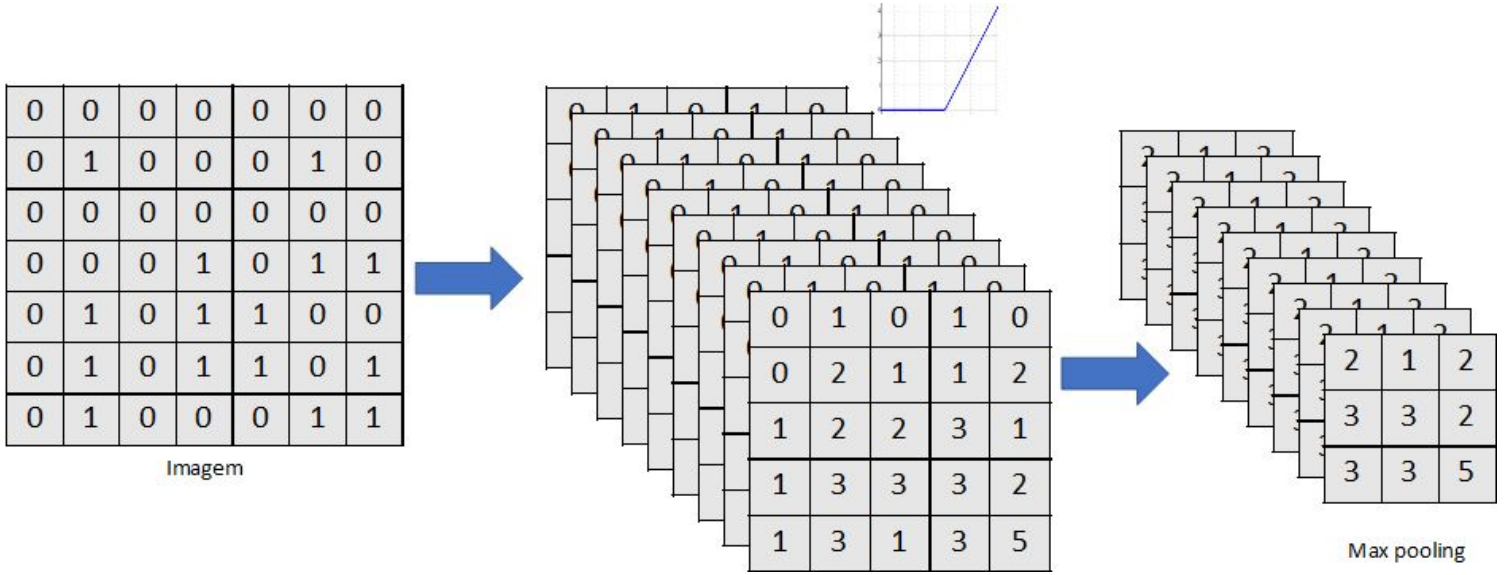
Etapa 2 - Pooling



Mapa de características
(feature map)

Etapa 2 - Pooling

- Seleciona as características mais relevantes (reduz overfitting e ruídos desnecessários)
- Max polling (mínimo, média): max foca nas características mais relevantes



Etapa 2 - Pooling

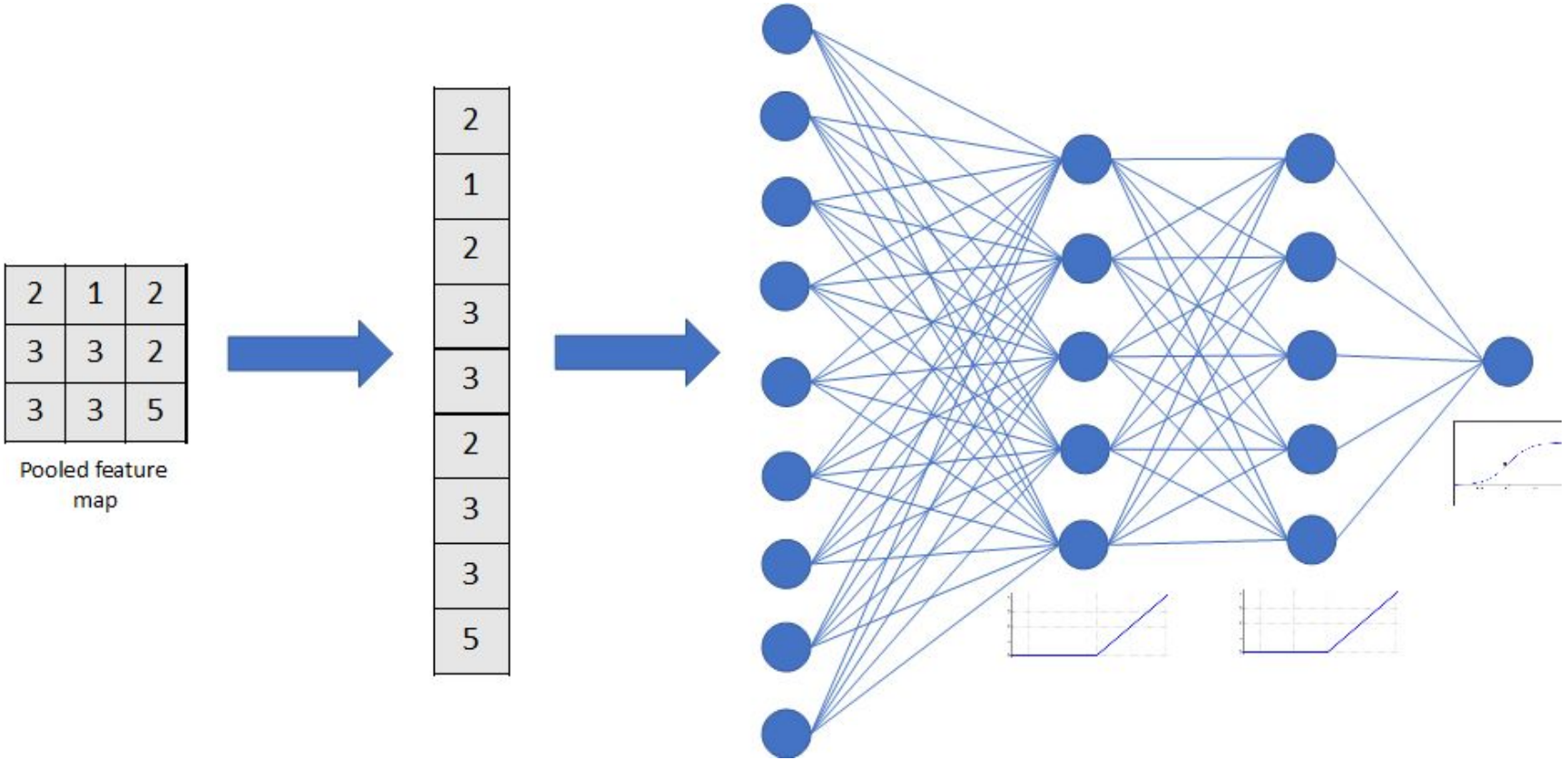
Dropout

Dropout não é uma especificidade de uma CNN, porém a utilizaremos em nossa implementação técnica, portanto abordaremos seu funcionamento.

Em resumo, a camada de Dropout é utilizada para evitar que determinadas partes da rede neural tenham muita responsabilidade e consequentemente, possam ficar muito sensíveis a pequenas alterações.

Essa camada recebe um hyper-parâmetro que define uma probabilidade de “desligar” determinada área da rede neural durante o processo de treinamento.

Etapa 3 – Flattening



Etapa 4 - Rede Neural Densa

Ao final da rede é colocada uma camada Fully connected, onde sua entrada é a saída da camada anterior e sua saída são N neurônios, com N sendo a quantidade de classes do seu modelo para finalizar a classificação.

As camadas densas (totalmente conectadas) são conectados por pesos. Por fim, a camada de saída da rede será a previsão ou classificação. Se for uma classificação binária, poderá ser um número indicando a probabilidade de presença do objeto em questão (ex.: probabilidade de na imagem conter um pedestre). Se o objetivo for classificar entre diversas classes, a saída será um vetor com a distribuição de probabilidades para cada classe. O processo funciona como um MLP.

Etapa 4 - Rede Neural Densa