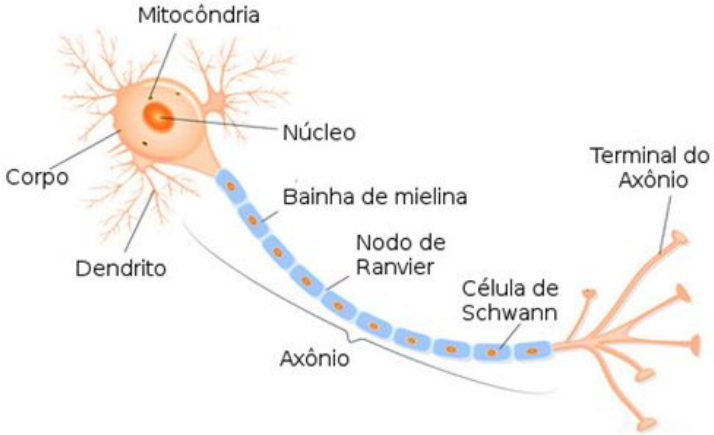
# EA991 - Laboratório de Aprendizado de Máquina

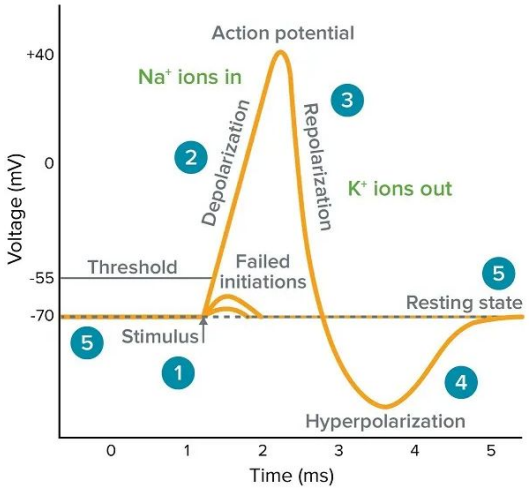
## Redes Neurais e Aprendizado Profundo

Prof. Denis G. Fantinato
Prof. Levy Boccato

# Inspiração

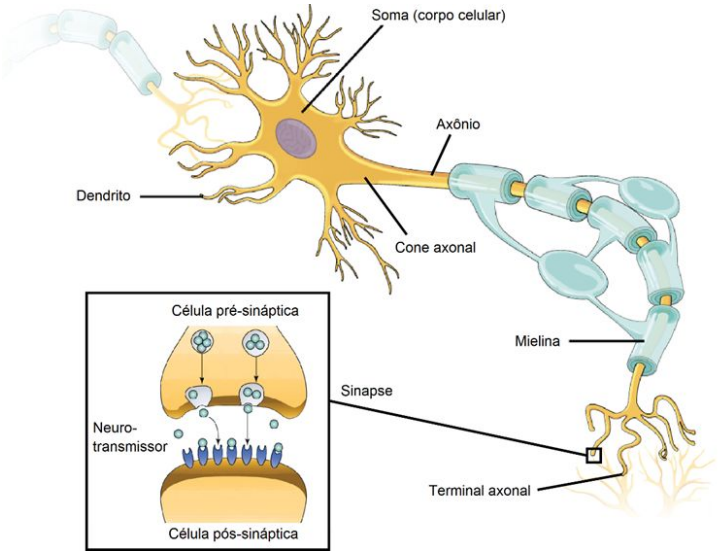
- **Neurônio:** célula especializada na recepção e transmissão de pulsos elétricos.
  - O neurônio recebe estímulos elétricos fundamentalmente a partir dos dendritos; caso a estimulação integrada (conjunta) exceda certo limiar, o neurônio gera um potencial de ação.



# Inspiração

- **Plasticidade sináptica:** é a capacidade de as sinapses sofrerem modificações ao longo do tempo.
  - É apontada como um dos fatores decisivos para vários processos cognitivos.

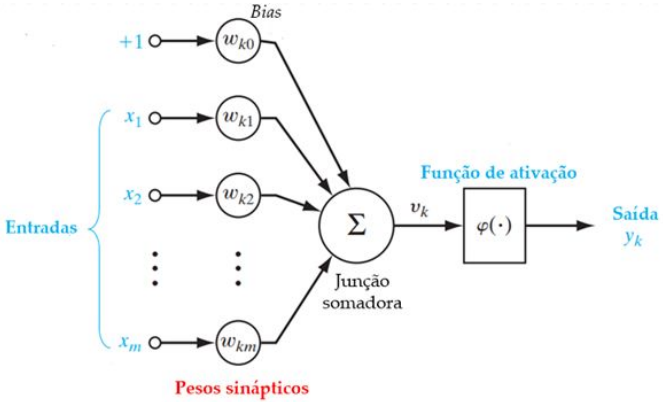


# Conceitos básicos

- **Neurônio artificial:** unidade elementar de processamento de informação.

- **Modelo clássico de neurônio:** *perceptron*.

$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj} x_j\right)$$




**Rede neural artificial** (ANN, *artificial neural network*): é uma estrutura composta de múltiplas unidades de processamento (neurônios) tipicamente organizadas em camadas e interligadas conforme um padrão de conexões.

# Conceitos básicos

**Função de ativação:** função não-linear que confere flexibilidade para o modelo aproximar mapeamentos entrada-saída complexos, além de afetar a eficiência de seu treinamento.

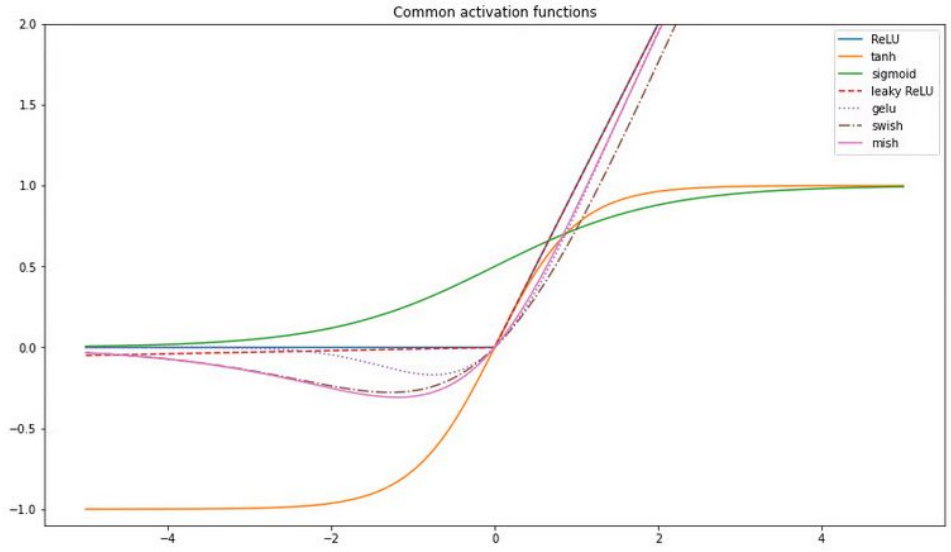
- **Exemplos:**

$$\mathrm{ReLU}(x) = \max(0, x)$$

$$\mathrm{GELU}(x) = \frac{1}{2}x\left(1 + \mathrm{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$$

$$\mathrm{swish}(x) = \frac{x}{1 + e^{-x}}$$

$$\mathrm{mish}(x) = x\tanh(x)\ln(1 + e^{x})$$



# Conceitos básicos

O tipo e a quantidade de neurônios artificiais presentes em uma rede, junto com a organização destes elementos em camadas e o respectivo padrão de conectividade definem a ***arquitetura*** da rede neural.

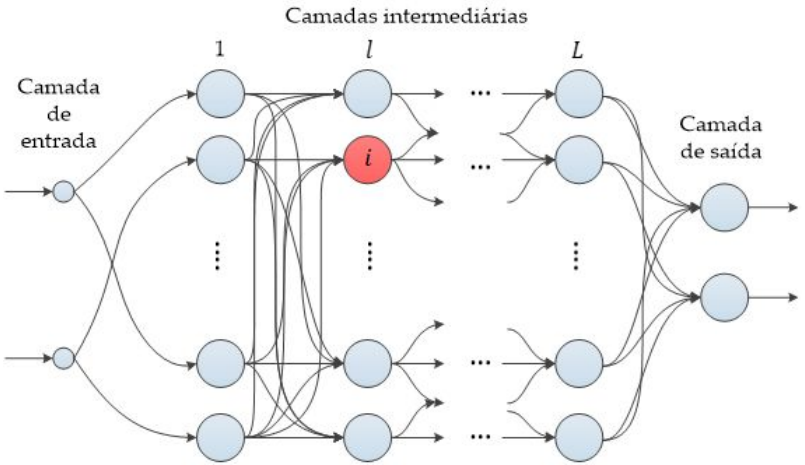
- Redes *feedforward*: os sinais recebidos pela rede são propagados em um único sentido através de todas as camadas até que as saídas sejam determinadas.
  - Mapeamento estático: $\mathbf{y}(n) = \Phi(\mathbf{x}(n); \boldsymbol{\theta})$

- Redes recorrentes: existem laços de realimentação que transmitem as saídas de neurônios de uma determinada camada para a entrada de neurônios pertencentes à mesma camada ou a camadas anteriores.
  - O modelo passa a ter uma memória interna que evolui dinamicamente e que afeta a resposta da rede.

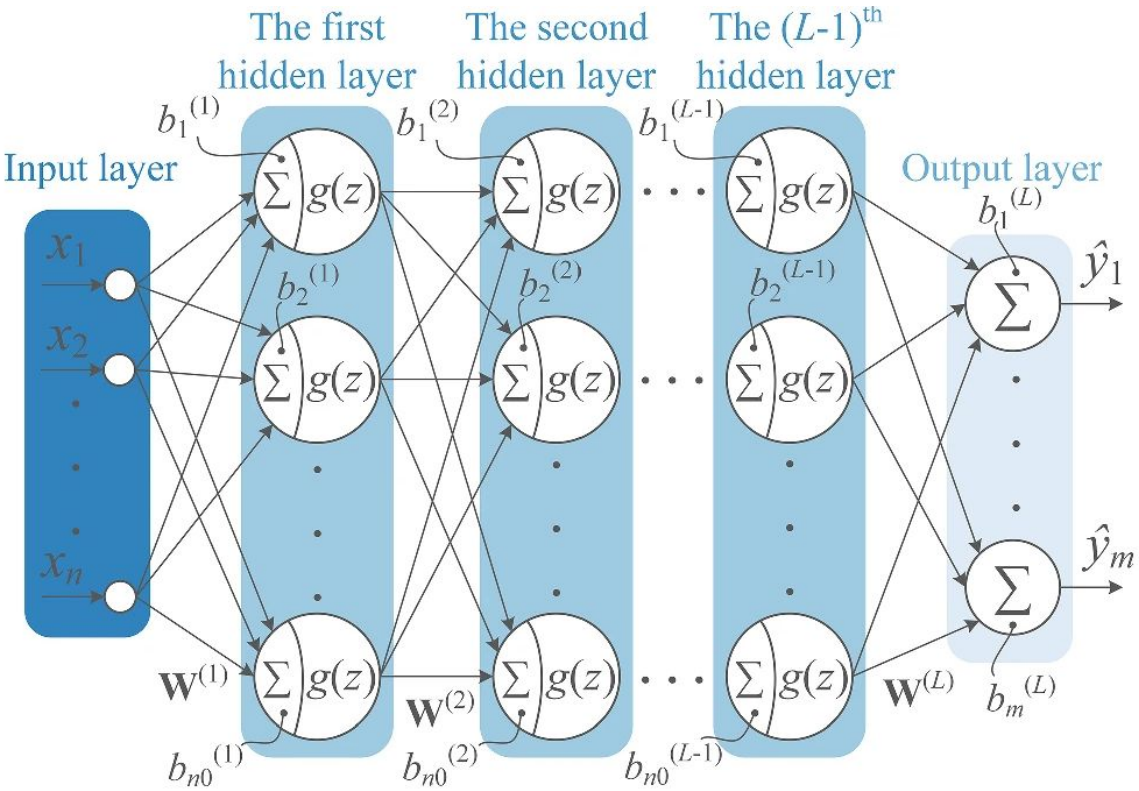
# MLP

- Um dos principais expoentes dentre as ANNs é a rede conhecida como *perceptron* de múltiplas camadas (MLP, do inglês *multilayer perceptron*).

- **Características:**
  - Existe um número arbitrário de camadas ocultas, também chamadas de camadas intermediárias, entre a entrada e a saída da rede.
  - Os neurônios de uma camada *l* estão conectados a todos os neurônios da camada seguinte (*l*+1). Por isso, essa estrutura é também chamada de totalmente conectada (ou densa).



# MLP


The first hidden layer
The second hidden layer
The $(L\text{-}1)^{th}$ hidden layer
Input layer
Output layer
$x_1$
$x_2$
$x_n$
$b_1^{(1)}$
$b_2^{(1)}$
$b_{n0}^{(1)}$
$b_1^{(2)}$
$b_2^{(2)}$
$b_{n0}^{(2)}$
$b_1^{(L-1)}$
$b_2^{(L-1)}$
$b_{n0}^{(L-1)}$
$b_1^{(L)}$
$b_m^{(L)}$
$\Sigma$ $g(z)$
$\mathbf{W}^{(1)}$
$\mathbf{W}^{(2)}$
$\mathbf{W}^{(L)}$
$\hat{y}_1$
$\hat{y}_m$

# Treinamento

- O treinamento de uma rede neural consiste no processo de ajuste dos parâmetros livres (e.g., os pesos sinápticos) de todas as camadas na busca pelo melhor mapeamento entrada-saída possível.

- Isto dá origem a um problema de otimização não-linear irrestrito, no qual, sem perda de generalidade, desejamos minimizar uma função de perda $J(\mathbf{w})$ que expressa uma medida de erro entre as saídas fornecidas pela rede e as saídas desejadas, onde $\mathbf{w}$ representa o vetor com todos os parâmetros da rede.

- Para isto, diversos **algoritmos de otimização não-linear** podem ser utilizados, tais como gradiente estocástico descendente, RMSProp e Adam.

# Treinamento

- **Obtenção do vetor gradiente:** requer o cálculo das derivadas da função de perda em relação a todos os parâmetros da rede, isto é, aos pesos dos neurônios de todas as camadas.
    - **Retropropagação do erro (*error backpropagation*):** tendo por base a regra da cadeia, este algoritmo realiza a propagação das derivadas desde a saída da rede - onde se computa diretamente a *loss* - até a primeira camada.

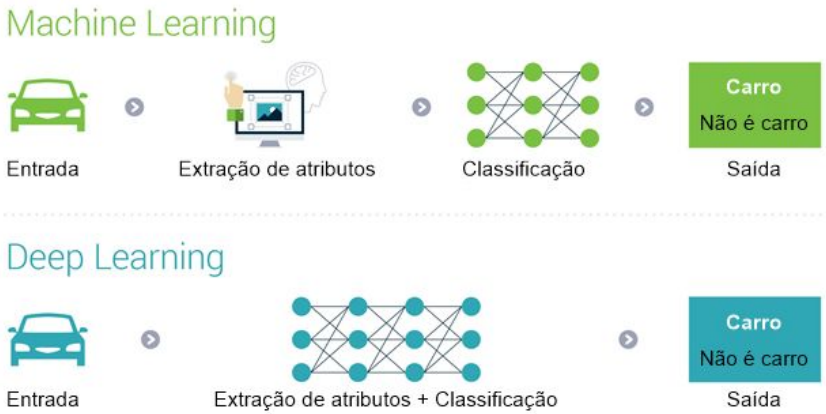
- **Processo típico:**

- Defina uma condição inicial para o vetor de pesos $\mathbf{w}$ e um passo $\eta$ pequeno;
- Faça $k = 0$ e calcule $J(\mathbf{w}(k))$;
- Enquanto o critério de parada não for atendido, faça:
    - Para $l$ variando de 1 até $N_B$:
        - Apresente o *batch* $l$ à rede;
        - Calcule $J_l(\mathbf{w}(k))$ e $\nabla J_l(\mathbf{w}(k))$;
        - Atualize os pesos: $\mathbf{w}(k+1) = \mathbf{w}(k) - \eta \nabla J_l(\mathbf{w}(k))$;
        - $k = k + 1$;
    - Incremente o contador de épocas.

# Redes convolucionais

# Aprendizado profundo

- **Marcas:**
  - Modelos compostos por um número relativamente elevado de camadas (e, consequentemente, de parâmetros ajustáveis).
  - Redução da ênfase no pré-processamento dos dados e na extração de atributos: os modelos conseguem lidar com dados brutos e não-estruturados.



# Redes neurais convolucionais

- As redes convolucionais (CNNs, do inglês *convolutional neural networks*) empregam a operação de **convolução** no lugar da transformação afim inerente a uma camada do tipo *perceptron*.

1D: $y(n) = \sum_{k=-\infty}^{\infty} x(k)w(n-k)$ 2D: $S(i,j) = \sum_{m}\sum_{n} X(m,n)K(i-m,j-n)$

- Essa operação linear permite explorar informações em estruturas organizadas no tempo, como séries temporais, ou no espaço, como imagens.

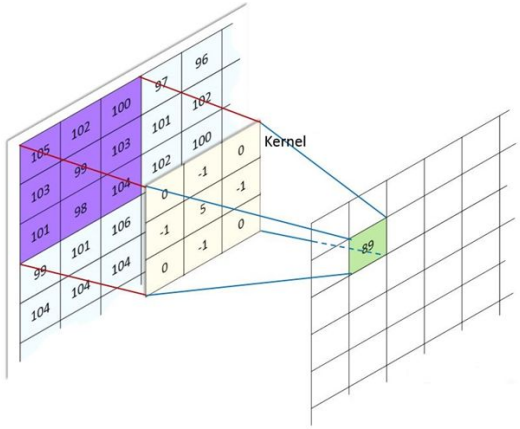
- Outra operação tipicamente presente em CNNs é chamada de agrupamento (ou *pooling*), a qual realiza uma sub-amostragem da entrada, resumindo a informação.

# Camada convolucional

- É composta por um ou mais *kernels*, que são filtros espaciais retangulares responsáveis por processar o dado recebido através da operação de convolução.

$$S(i,j) = \sum_m \sum_n X(m,n)K(i-m, j-n)$$

- O *kernel* percorre toda a entrada, sendo que, em cada posição, a resposta é obtida através da soma dos valores (pixels) da entrada dentro da vizinhança definida pelo *kernel*, multiplicados pelos coeficientes do *kernel*.

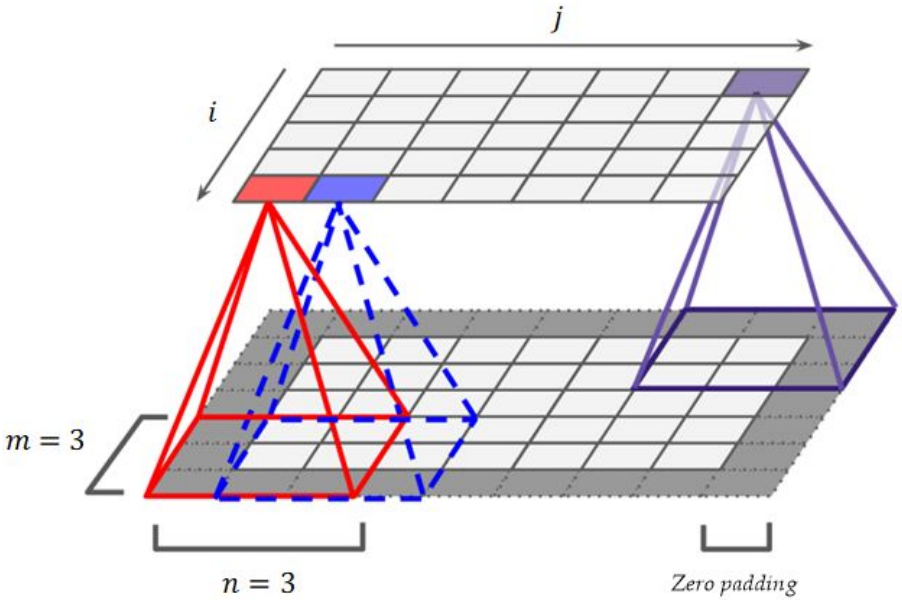


# Camada convolucional

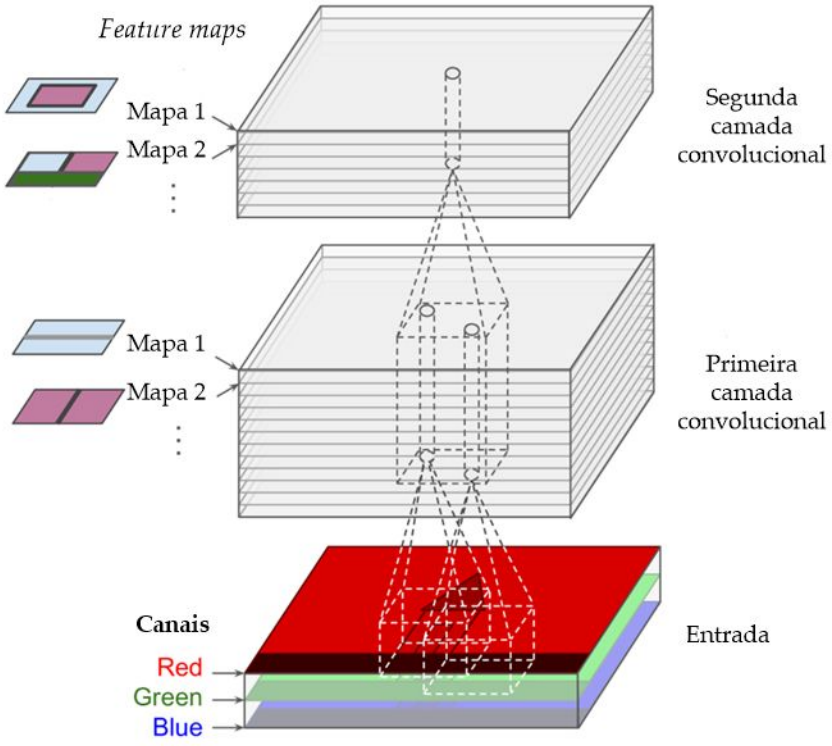
- Cada filtro (*kernel*) gera uma nova imagem como resposta à entrada, a qual é chamada de ***feature map*** (ou apenas canal). Uma camada convolucional contendo *N kernels* gera, portanto, um volume de saída com *N* canais.

- Cada "pixel" em um *feature map* pode ser considerado como um neurônio, e é influenciado por um subconjunto dos *pixels* da imagem de entrada, os quais definem o chamado **campo receptivo** daquele neurônio.

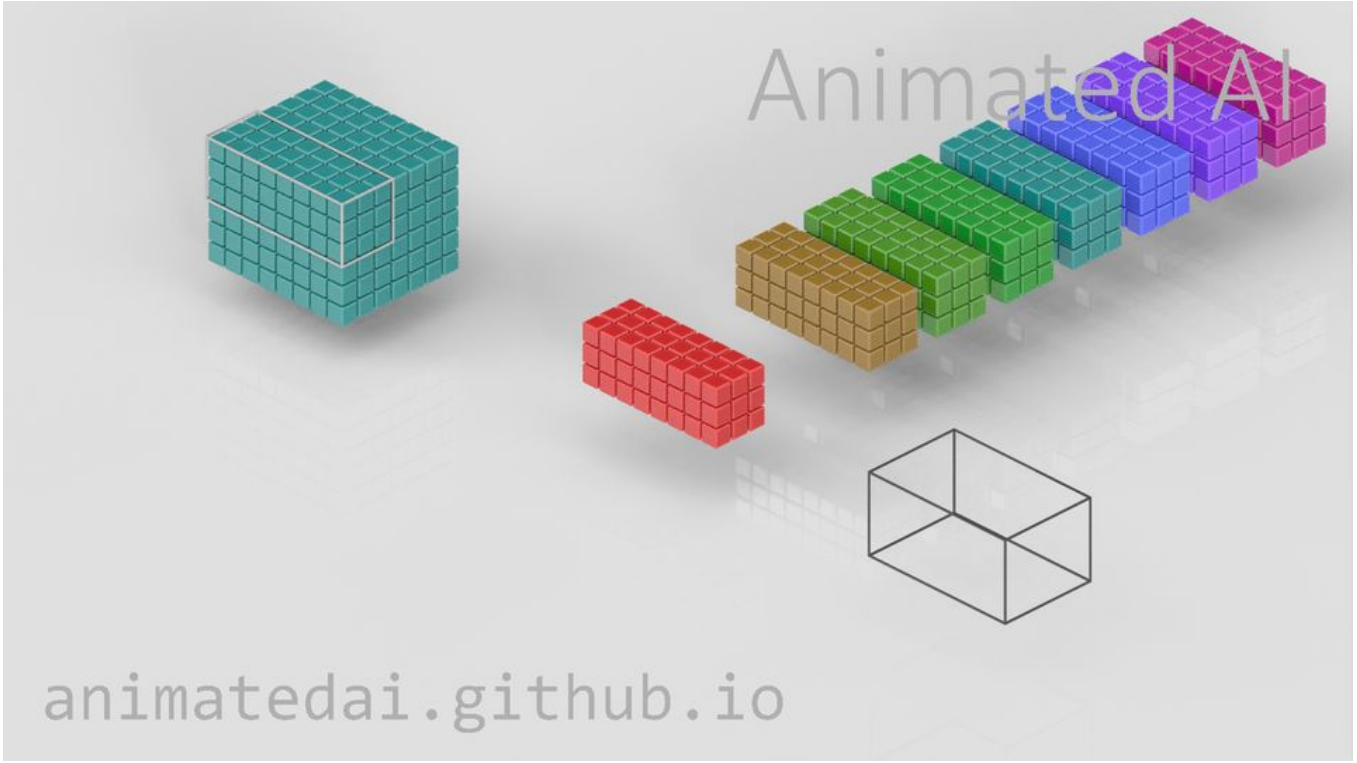
- ***Stride*****:** a quantidade de *pixels* com que o *kernel* é deslocado horizontal e verticalmente durante a convolução.



# Camada convolucional



# Camada convolucional




https://animatedai.github.io/

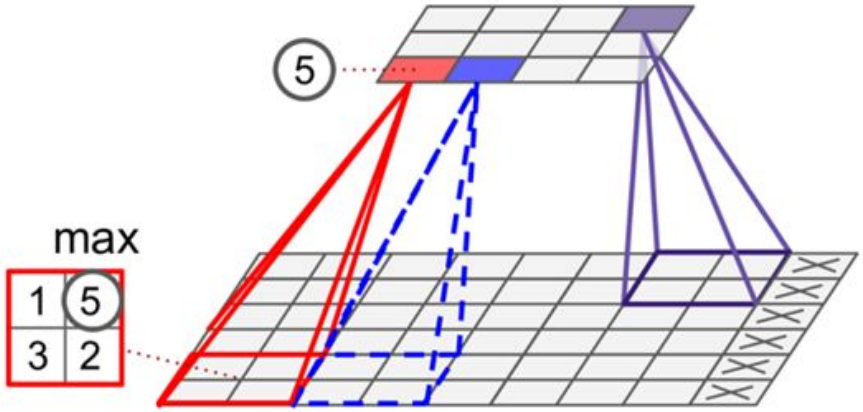
# *Pooling*

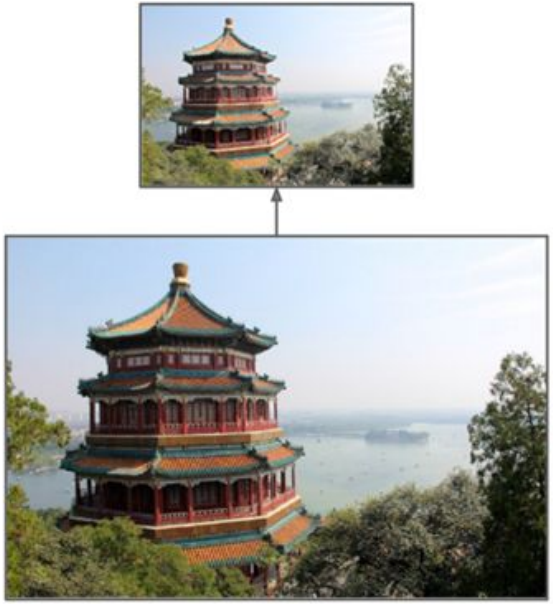
- O objetivo da camada de agrupamento (*pooling*) é sub-amostrar a imagem de entrada, fornecendo um "sumário" da informação em elementos vizinhos, a fim de reduzir a carga computacional, o uso de memória e o número de parâmetros da rede.

- **Hiperparâmetros:**
  - Tamanho do campo receptivo retangular;
  - *Stride*;
  - Tipo de *padding*;
  - Função de agregação, a qual será aplicada sobre os elementos da entrada contidos dentro do campo receptivo.

- Dois tipos de *pooling* bastante usuais em CNNs são o *average pooling* e o *max-pooling*.

# Pooling



# CNNs

- Tipicamente, as arquiteturas de CNNs empilham algumas camadas convolucionais em sequência para, então, aplicar uma camada de *pooling*. Este padrão - camadas convolucionais e *pooling* - é repetido algumas vezes.
    - Com isso, a imagem de entrada se torna cada vez menor à medida que atravessa a rede, mas, ao mesmo tempo, ela também fica mais e mais profunda (i.e., com mais canais), por conta das camadas convolucionais.

- Então, uma rede *feedforward* convencional é acrescentada, composta por algumas camadas totalmente conectadas, e, finalmente, a camada de saída gera a resposta da rede ao padrão de entrada.