

# 08 REDES CONVOLUCIONAIS

## APRENDIZAGEM PROFUNDA

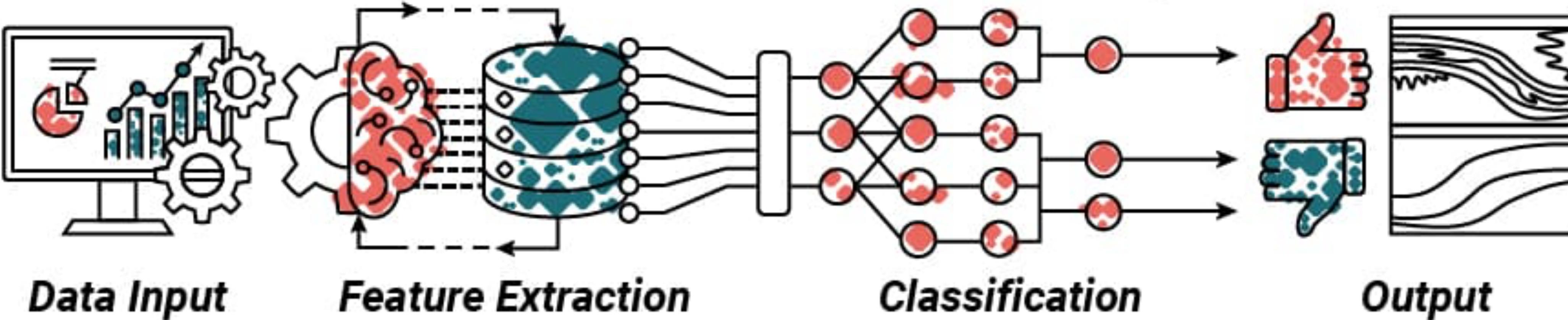
PPGCC – 2023.1

Prof. Saulo Oliveira <[saulo.oliveira@ifce.edu.br](mailto:saulo.oliveira@ifce.edu.br)>

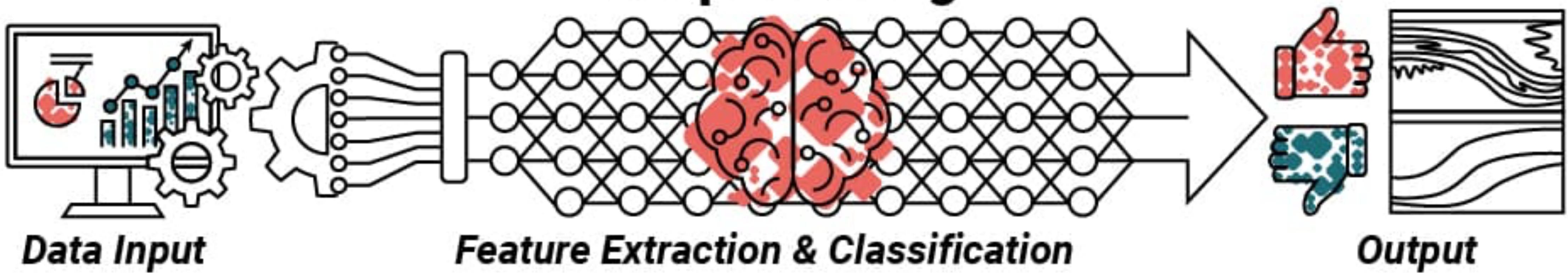


# REVISÃO DE APRENDIZAGEM DE MÁQUINA

## Traditional Machine Learning



## Deep Learning

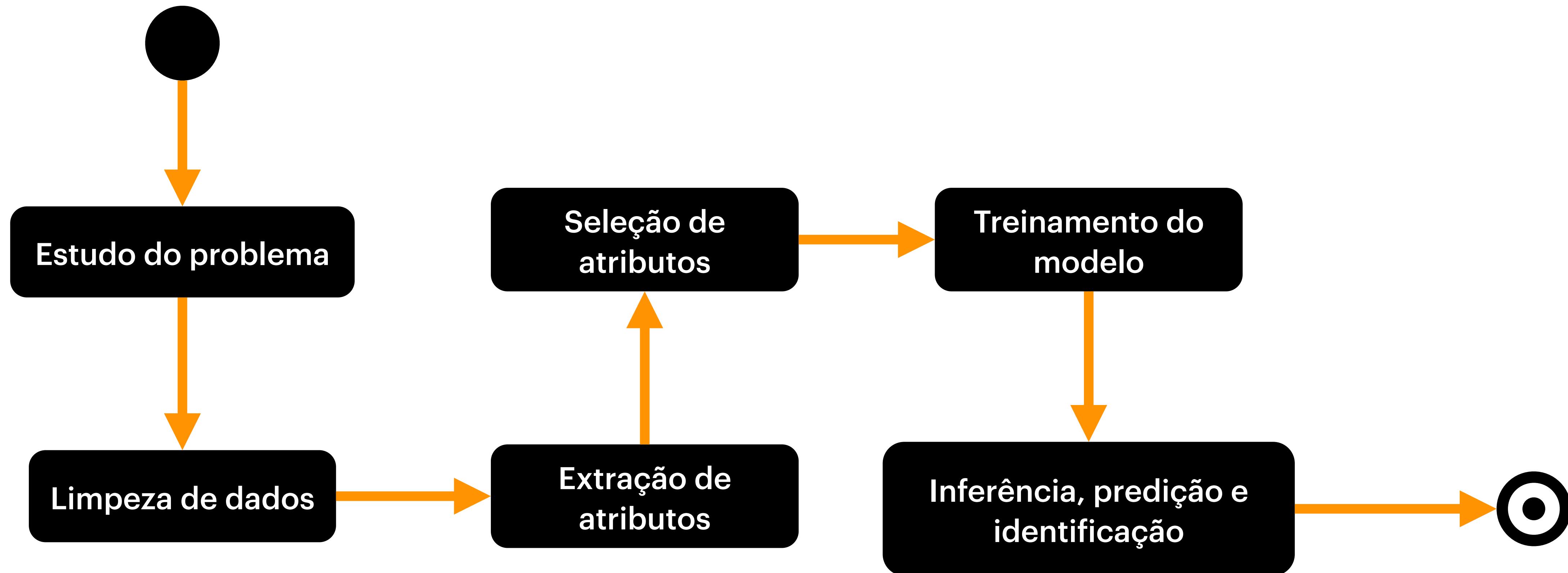


**Fonte:** <https://mosaicdatascience.com/2022/02/02/how-deep-learning-facilitates-automation-innovation-and-when-to-use-it/>

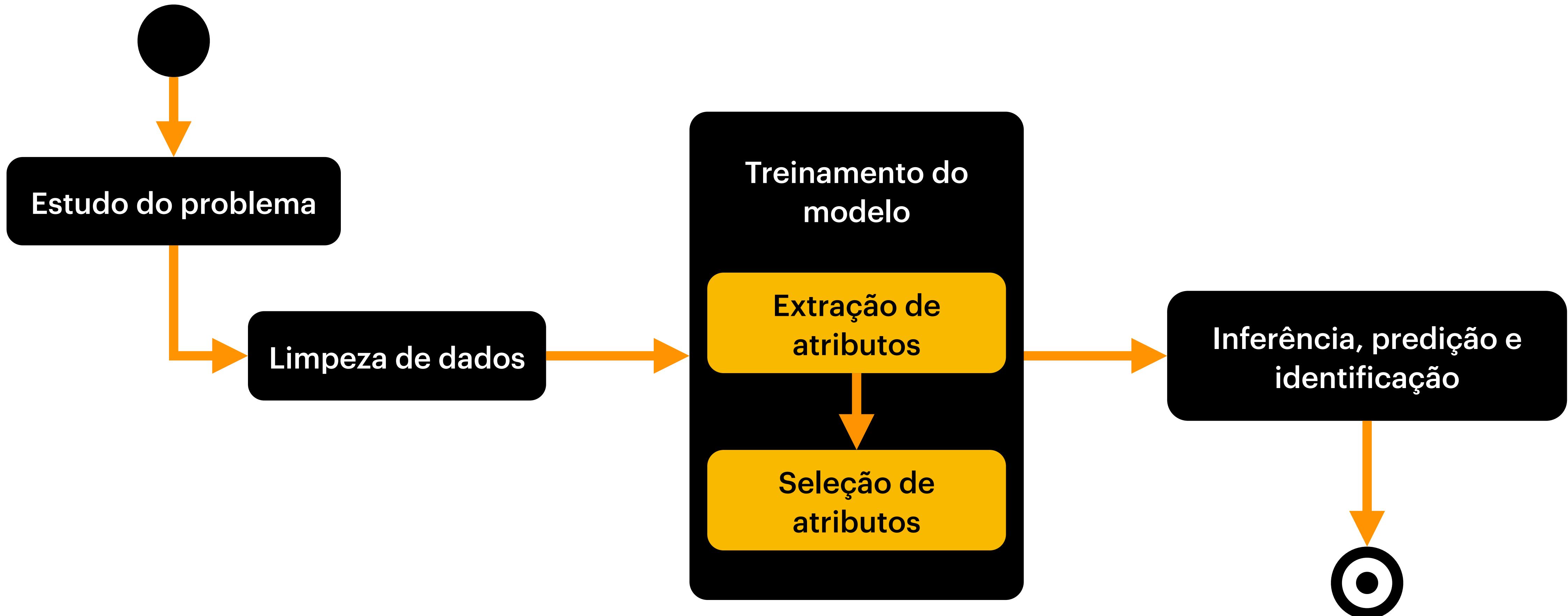
# Aprendizagem de máquina vs Aprendizagem Profunda

| Tradicional  | Profunda   |
|--|--|
| Baixos requisitos de hardware no computador: Dada a quantidade limitada de computação, o computador, <b>geralmente, não precisa de uma GPU</b> para computação paralela.       | Requisitos de hardware mais altos no computador: Para executar operações de matriz em <b>dados massivos</b> , o <b>computador precisa de uma GPU</b> (ou TPU) para executar a computação paralela. |
| Aplicável ao treinamento com uma <b>pequena quantidade de dados</b> e cujo desempenho <b>não pode ser melhorado continuamente à medida que a quantidade de dados aumenta</b> . | Cujo desempenho pode ser alto quando parâmetros de <b>peso de alta dimensão</b> e <b>dados de treinamento massivos</b> são fornecidos.   |
| Detalhamento do problema nível por nível.  | Aprendizagem ponto-a-ponto (E2E learning).   |
| Seleção manual de atributos.   | Extração automática de atributos.  |
| Atributos fáceis de explicar.  | Atributos (ainda) difíceis de explicar.  |

# Framework tradicional



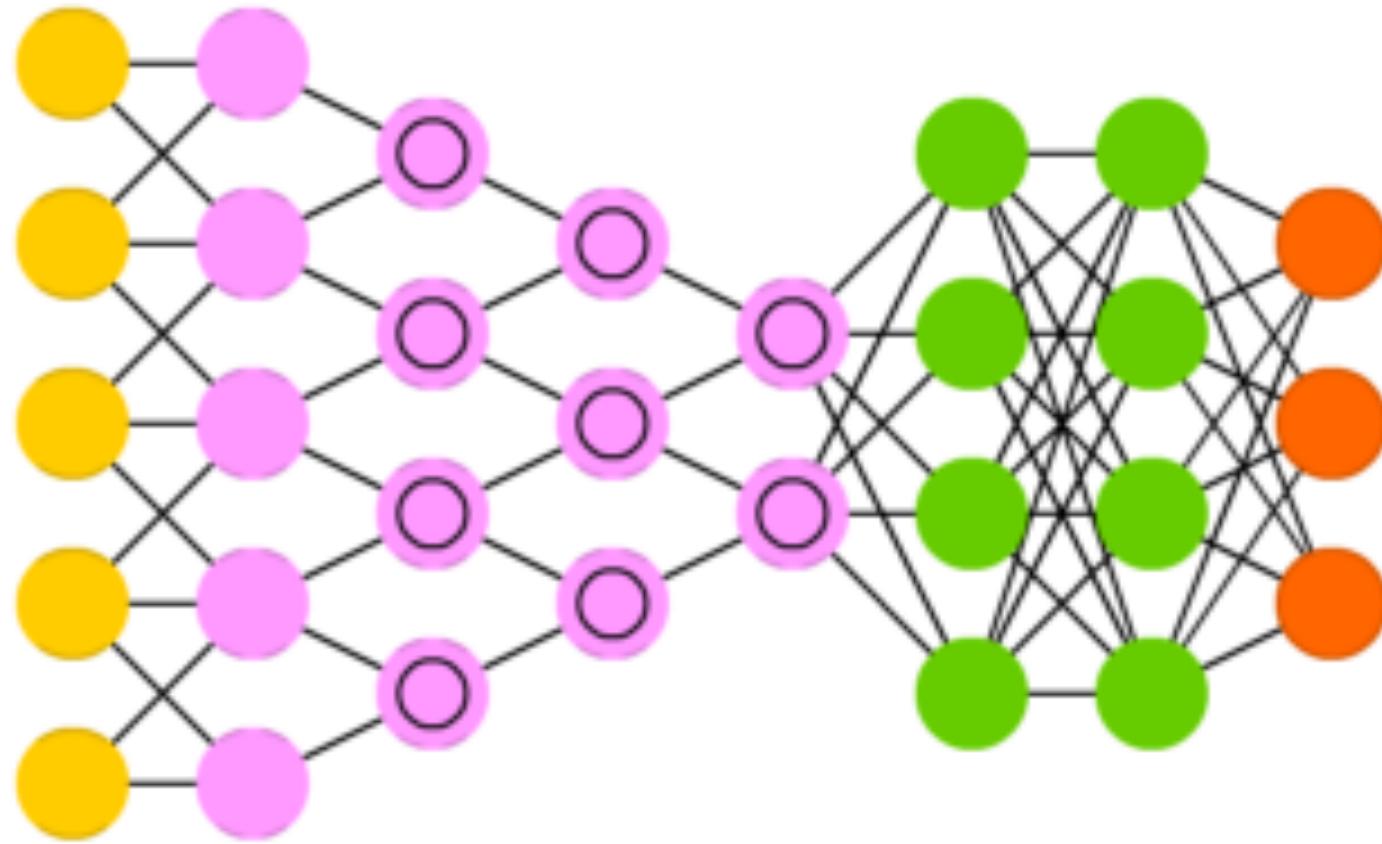
# Framework aprendizagem profunda



# Algumas redes neurais

Normalmente, a arquitetura é uma rede neural profunda e o "Deep" em "deep learning" refere-se ao número de camadas da rede neural.

Deep Convolutional Network (DCN)

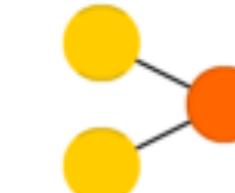


- Backfed Input Cell
- Input Cell
- △ Noisy Input Cell
- Hidden Cell
- Probabilistic Hidden Cell
- △ Spiking Hidden Cell
- Output Cell
- Match Input Output Cell
- Recurrent Cell
- Memory Cell
- △ Different Memory Cell
- Kernel
- Convolution or Pool

## A mostly complete chart of Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

Perceptron (P)



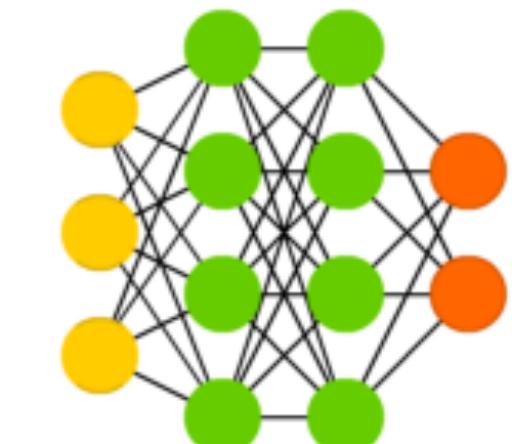
Feed Forward (FF)



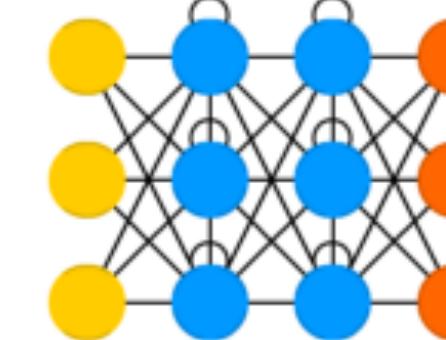
Radial Basis Network (RBF)



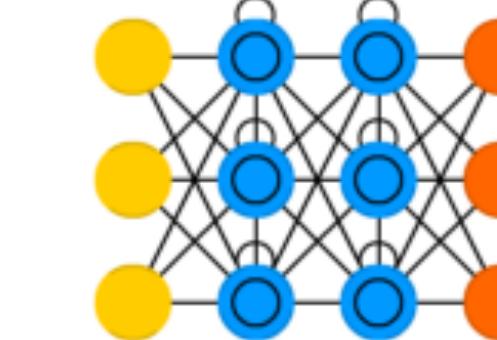
Deep Feed Forward (DFF)



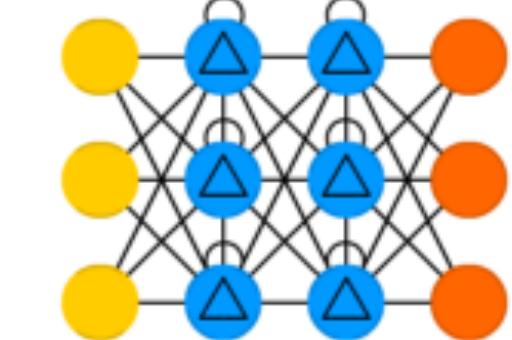
Recurrent Neural Network (RNN)



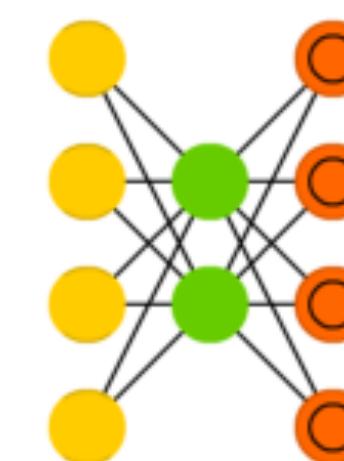
Long / Short Term Memory (LSTM)



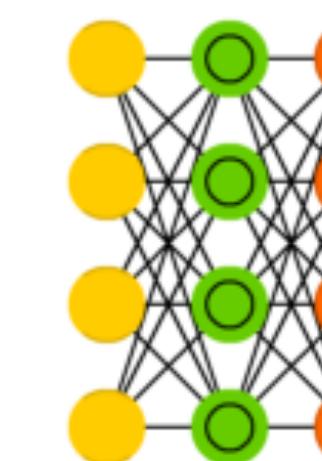
Gated Recurrent Unit (GRU)



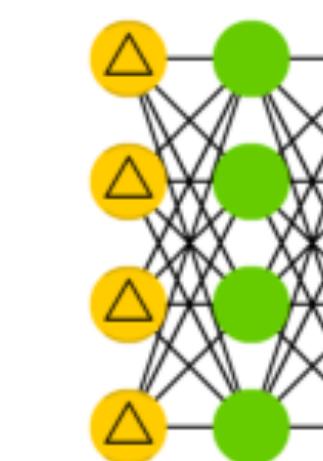
Auto Encoder (AE)



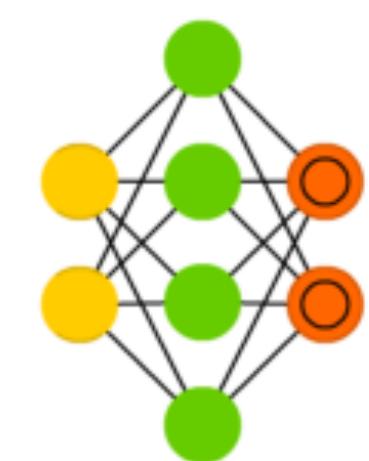
Variational AE (VAE)



Denoising AE (DAE)



Sparse AE (SAE)

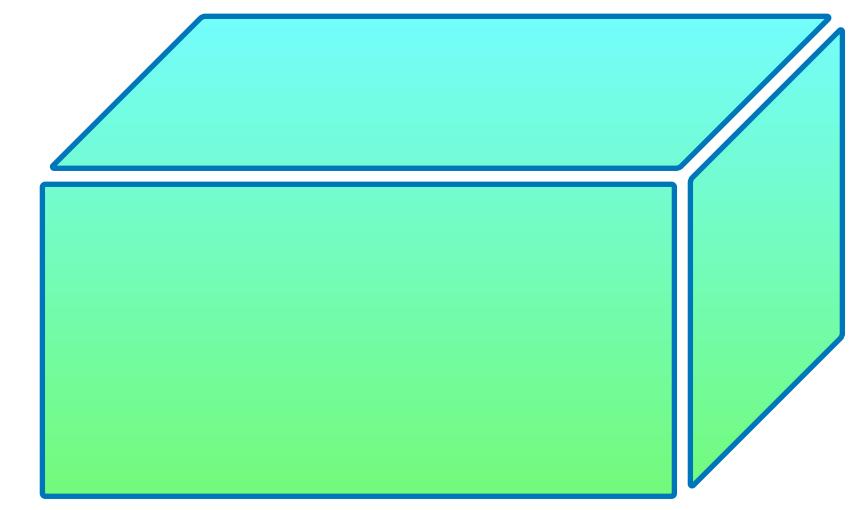
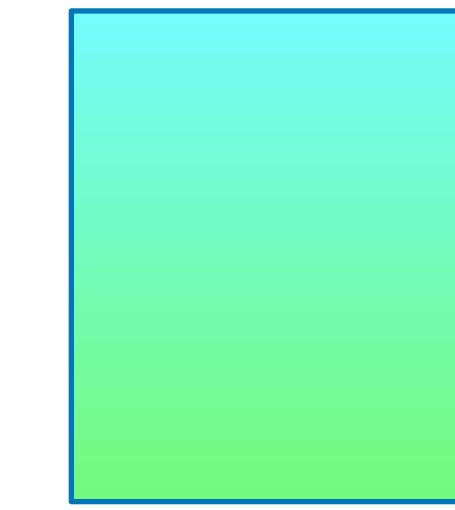


Fonte: <https://medium.com/predict/the-complete-list-to-make-you-an-ai-pro-be83448720b8>

**TENSORES SÃO TUDO**

# Tensores

Tensores são as estruturas de dados mais básicas nos Frameworks de Deep Learning. Todos os dados são encapsulados em tensores.



**Tensor: uma matriz multidimensional.**

Rank 0  
Tensor  
(Escalar)

Rank 1  
Tensor  
(Vetor)

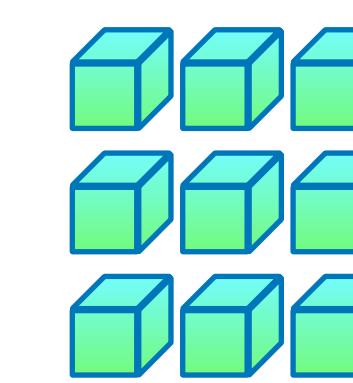
Rank 2  
Tensor  
(Matriz)

Rank 3  
Tensor

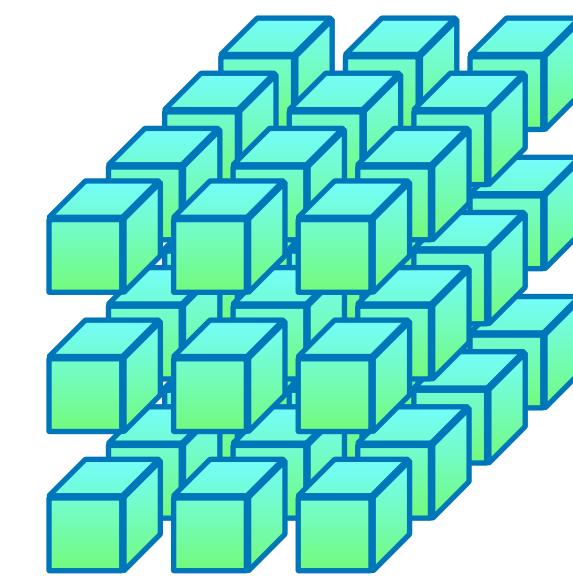
- Um escalar é um tensor de posto 0.
- Um vetor é um tensor de posto 1.
- Uma matriz é um tensor de posto 2.



Rank 4  
Tensor



Rank 5  
Tensor



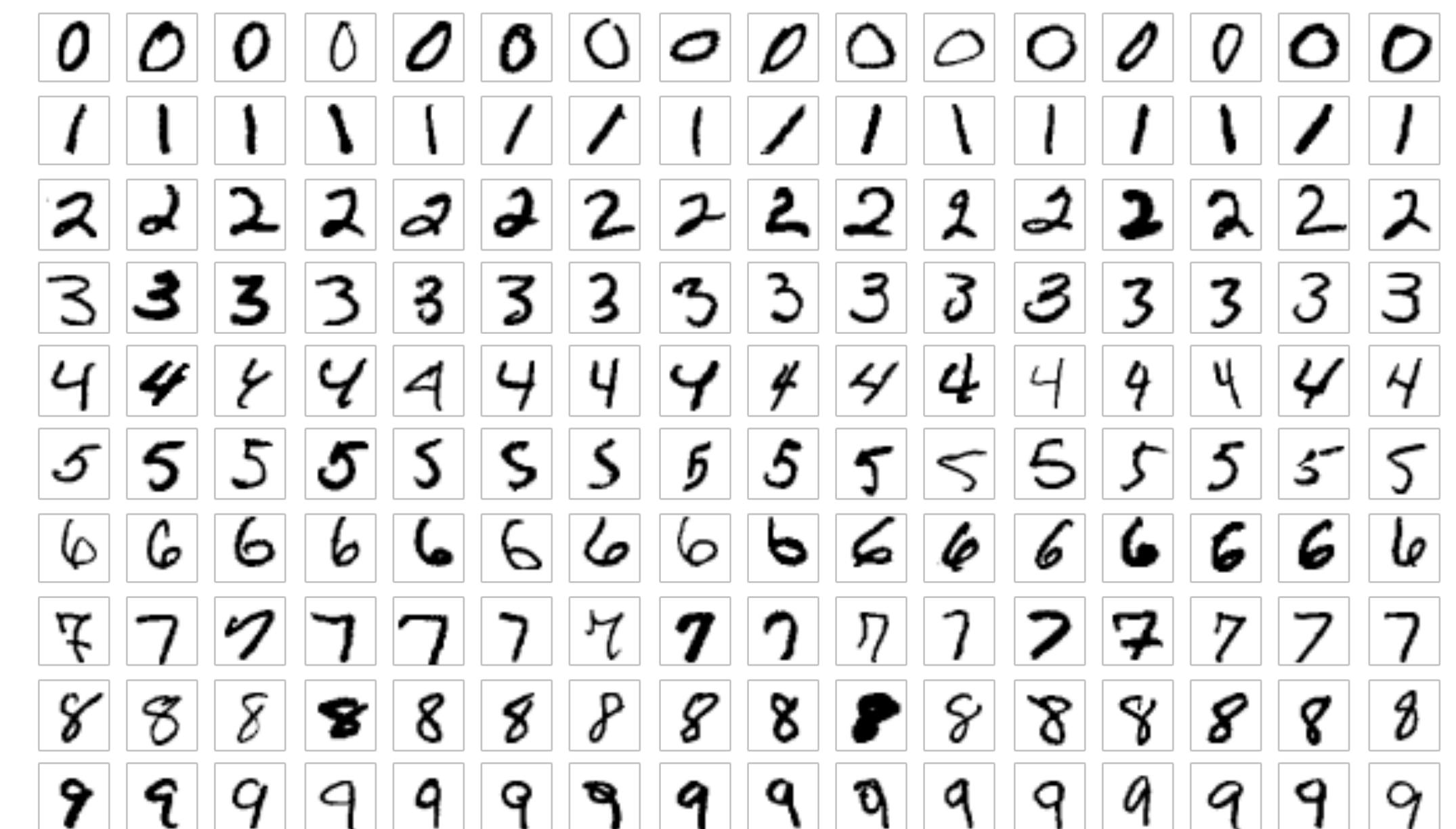
Rank 6  
Tensor

# MNIST

O banco de dados **MNIST** (banco de dados modificado do Instituto Nacional de Padrões e Tecnologia) é um grande banco de dados de dígitos manuscritos que é comumente usado para treinar vários sistemas de processamento de imagem.

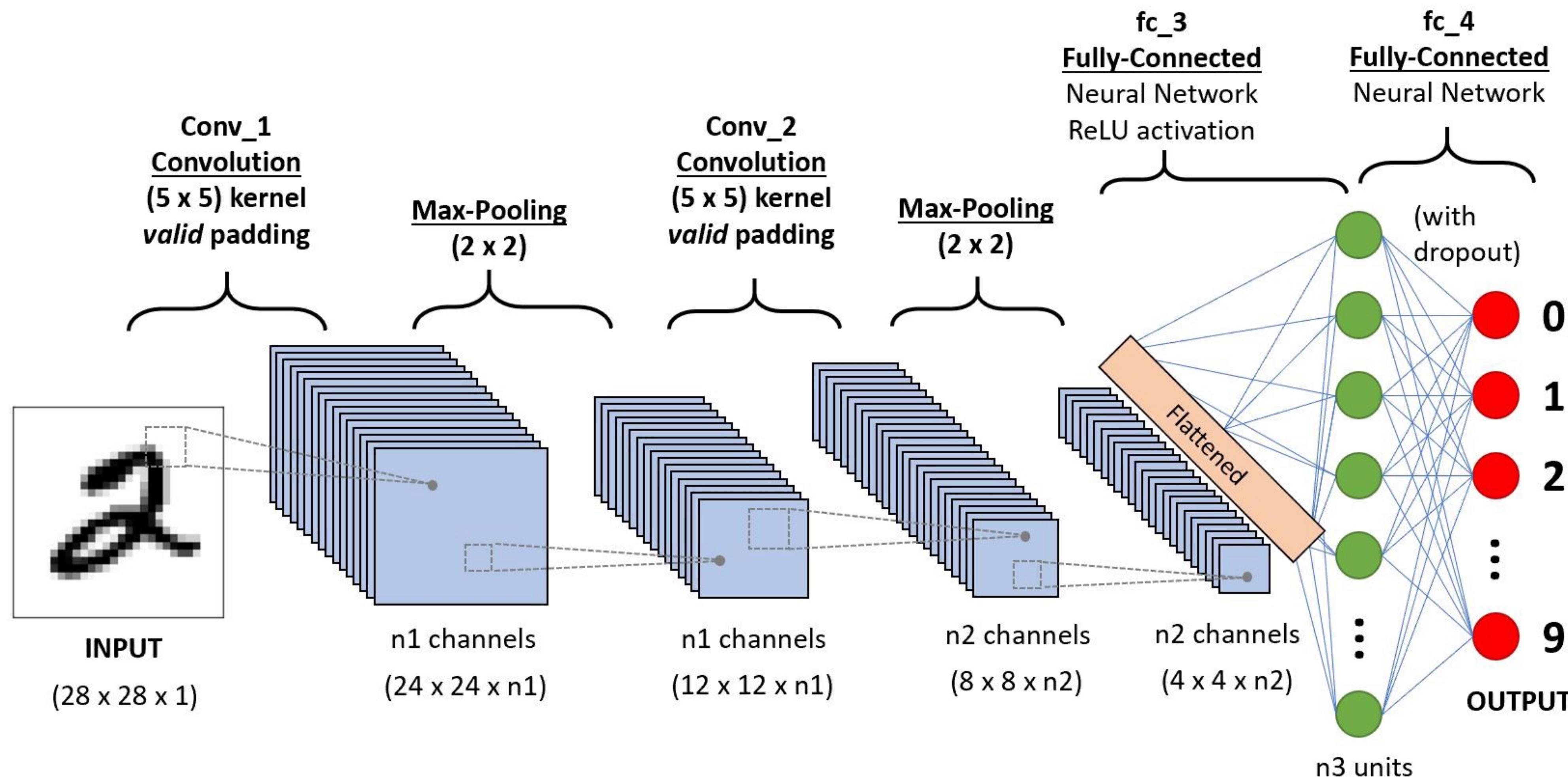
O banco de dados MNIST contém 60.000 imagens de treinamento e 10.000 imagens de teste.

Além disso, as imagens em preto e branco do NIST foram normalizadas para caber em uma caixa delimitadora de pixel e anti-aliasing, o que introduziu níveis de tons de cinza.



Fonte: [https://en.wikipedia.org/wiki/MNIST\\_database](https://en.wikipedia.org/wiki/MNIST_database)

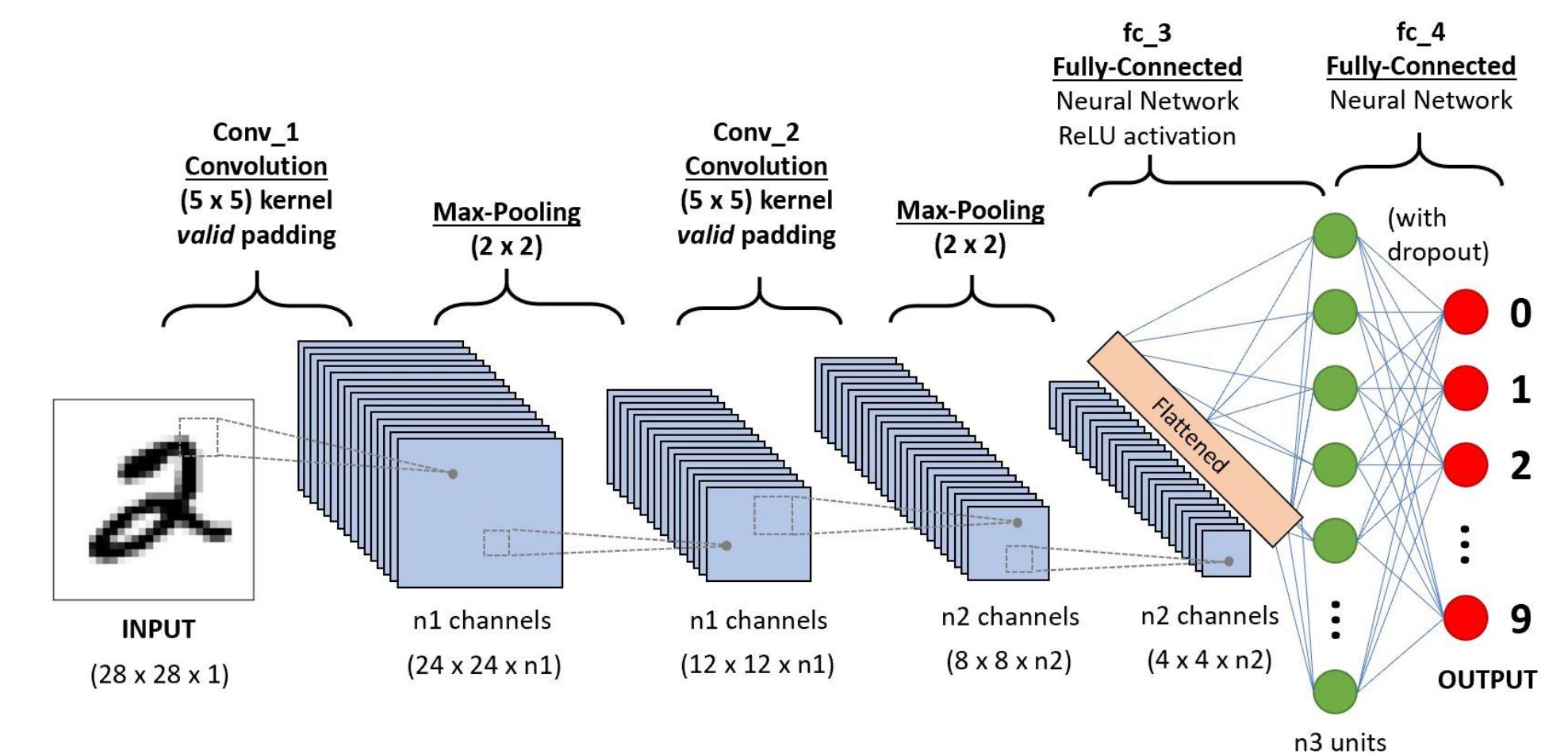
# Arquitetura geral de uma CNN



**Fonte:** <https://agents.co/blog/publication/introduction-to-convolutional-neural-networks-cnns/>

# Arquitetura geral de uma CNN

**Campo receptivo local:** considera-se que a percepção humana do mundo exterior é do local ao global. As correlações espaciais entre os pixels locais de uma imagem são mais próximas do que entre os pixels distantes. Portanto, cada neurônio não precisa conhecer a imagem global, basta saber a imagem de forma local. As informações locais são combinadas em um nível superior (hierarquia) para gerar informações globais.



**Compartilhamento de parâmetros:** um ou mais filtros de convolução podem ser usados para processar imagens de entrada. Os parâmetros carregados pelos núcleos de convolução são pesos. Em uma camada formada por núcleos de convolução, cada núcleo usa os mesmos parâmetros durante a computação ponderada. O compartilhamento de pesos significa que, quando uma convolução varre uma imagem inteira, os parâmetros da convolução são fixos.

Error signal: 0.007

Variation weights: 0.163

Total bias: -0.537

# ADVANCED NEURAL NETWORK

9  
8  
7  
6  
5  
4  
3  
2  
1  
0

Epoch: 1

Iteration: 33

Error: 9.500524

Fonte: <https://www.youtube.com/watch?v=2GYLpzZzqeg>

**NOVAS CAMADAS...**  
**NOVOS SUPERPODERES...**

# CNN em ação



CS231n: Convolutional Neural Networks for Visual Recognition



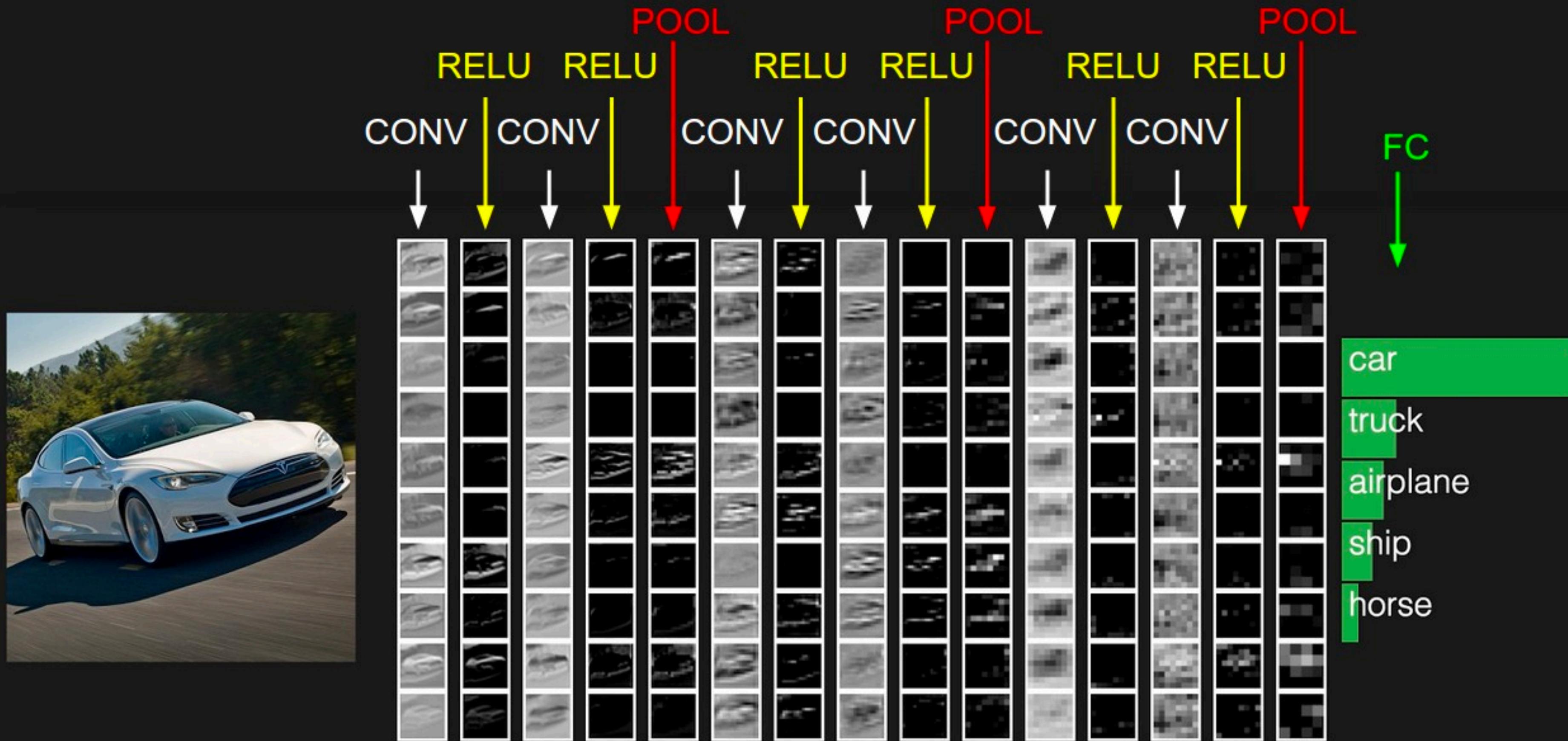
Spring 2020

Previous Years: [\[Winter 2015\]](#) [\[Winter 2016\]](#) [\[Spring 2017\]](#) [\[Spring 2018\]](#) [\[Spring 2019\]](#)



Fonte: <http://cs231n.stanford.edu/>

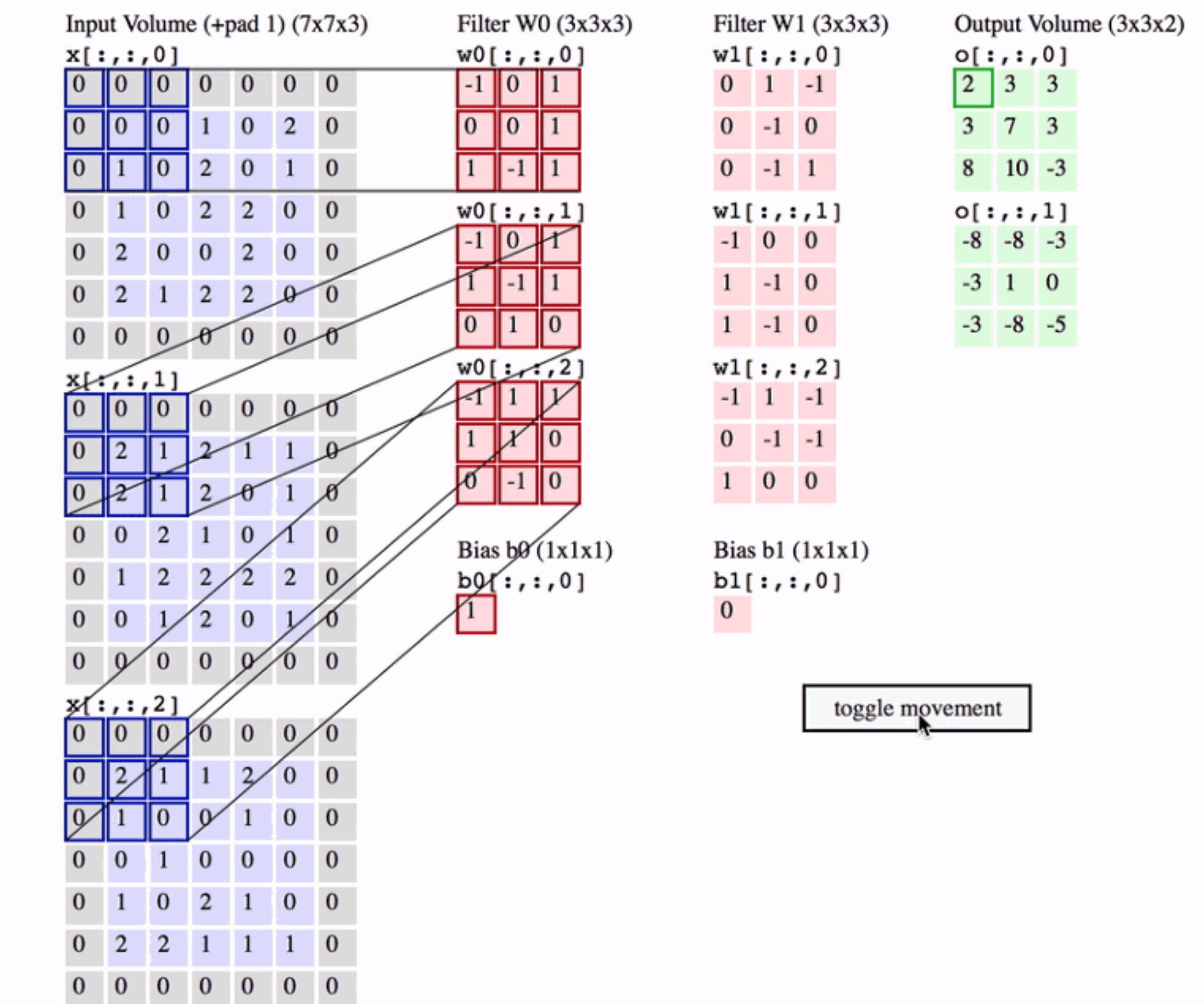
# CNN em ação



Fonte: <https://cs231n.github.io/convolutional-networks/>

# Camada de convolução

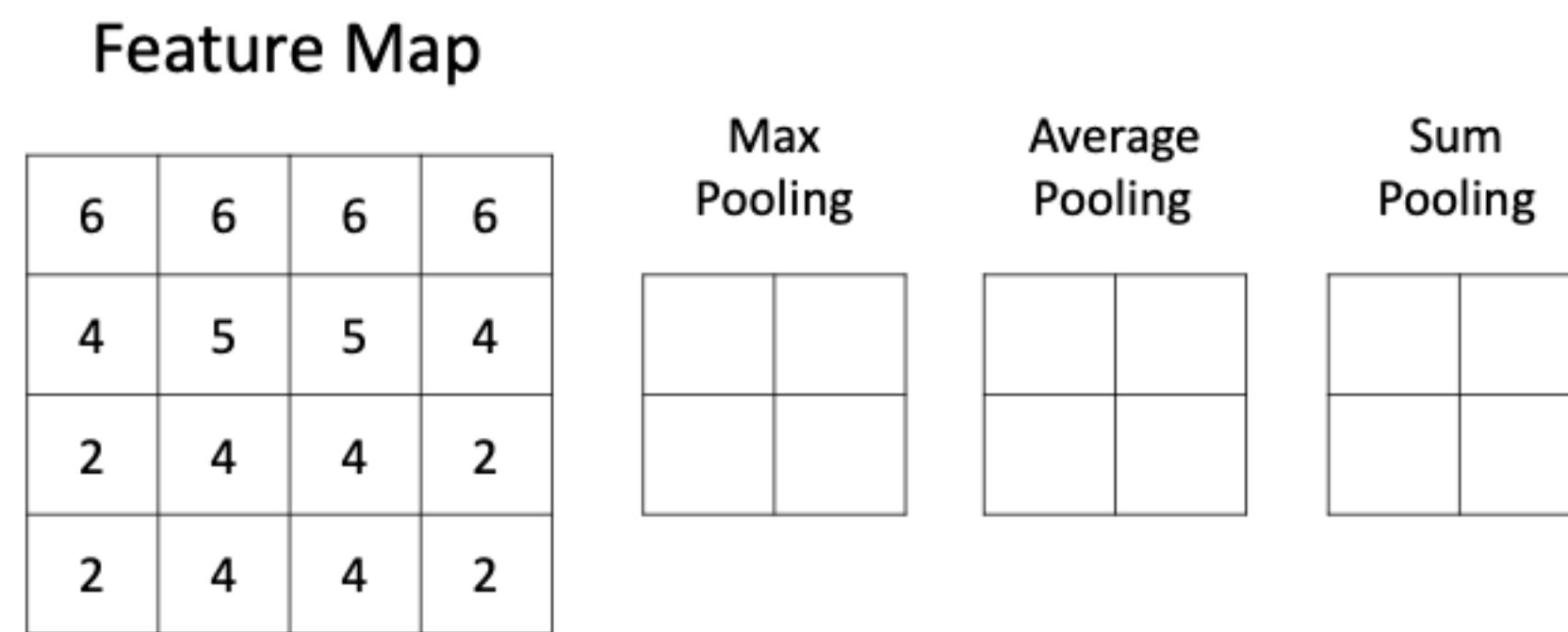
- Os filtros são tensores ( $W_1 \times H_1 \times D_1$ );
- Cada camada possui quatro hiper-parâmetros:
  - ❖ Número de filtros  $K$ ;
  - ❖ Tamanho de filtro  $F$ ;
  - ❖ Tamanho do passo (stride)  $S$ ;
  - ❖ Quantidade de preenchimento (padding)  $P$ ;
- Geralmente, usa-se  $F = 3, S = 1$  e  $P = 1$ .
- Após a convolução, o Tensor resultante passa a ter um novo tamanho de acordo com os parâmetros definidos na camada, a saber,
  - ❖  $W_2 = (W_1 - F + 2P)/S + 1$
  - ❖  $H_2 = (H_1 - F + 2P)/S + 1$
  - ❖  $D_2 = K$



Fonte: <https://cs231n.github.io/convolutional-networks/>

# Camada de pooling (agrupamento)

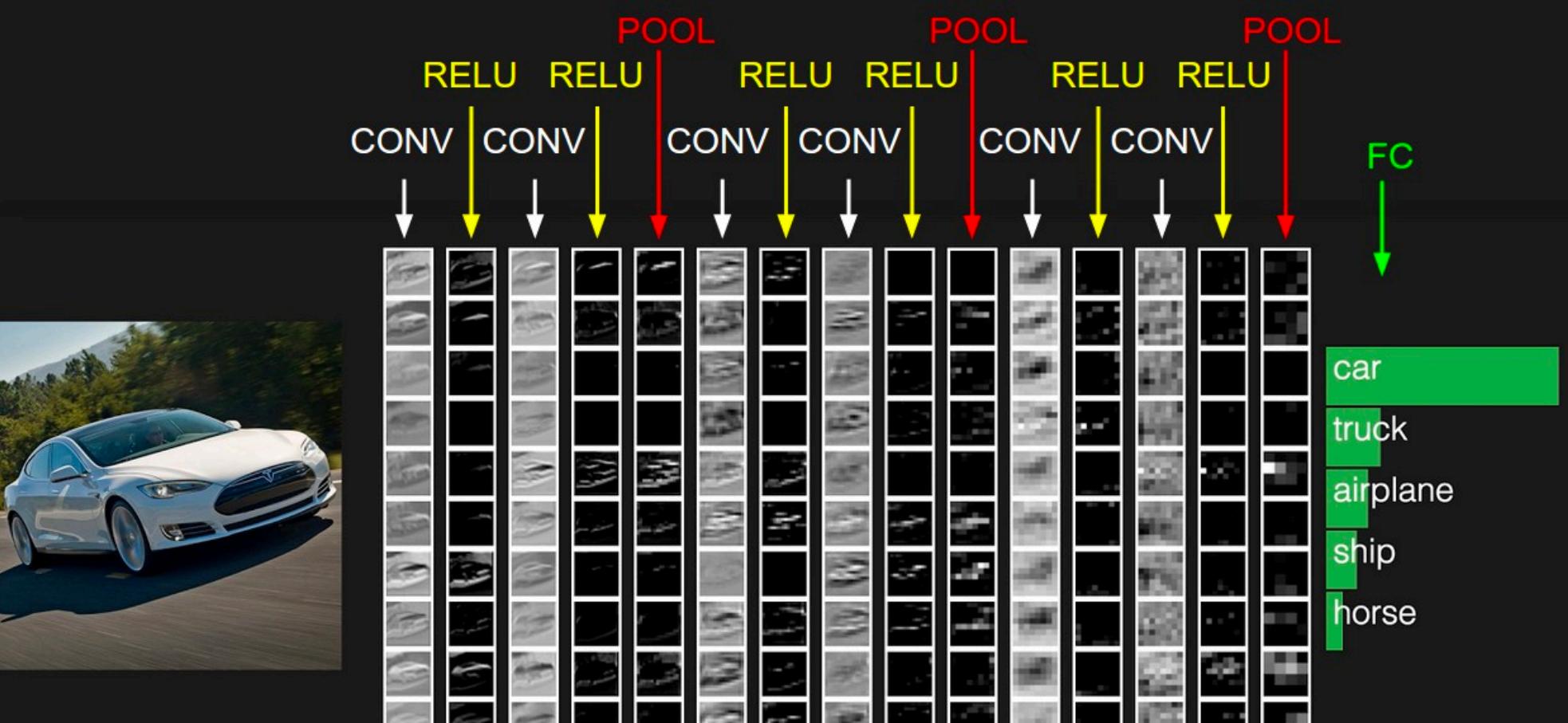
- O resultado do uso de uma camada de *pooling* e da criação de mapas de atributos amostrados ou agrupados é uma versão resumida dos atributos detectadas na entrada;
- São úteis porque pequenas mudanças na localização desses atributos detectadas pela camada convolucional resultarão em um mapa agrupado com os atributos no mesmo local;
- Essa capacidade adicionada pelo *pooling* é chamada de invariância do modelo para a translação local;



- A camada de *pooling* combina unidades próximas para reduzir o tamanho da entrada na próxima camada, reduzindo as dimensões. Operações muito comuns incluem o agrupamento máximo e o agrupamento médio.

# Camada totalmente conectada

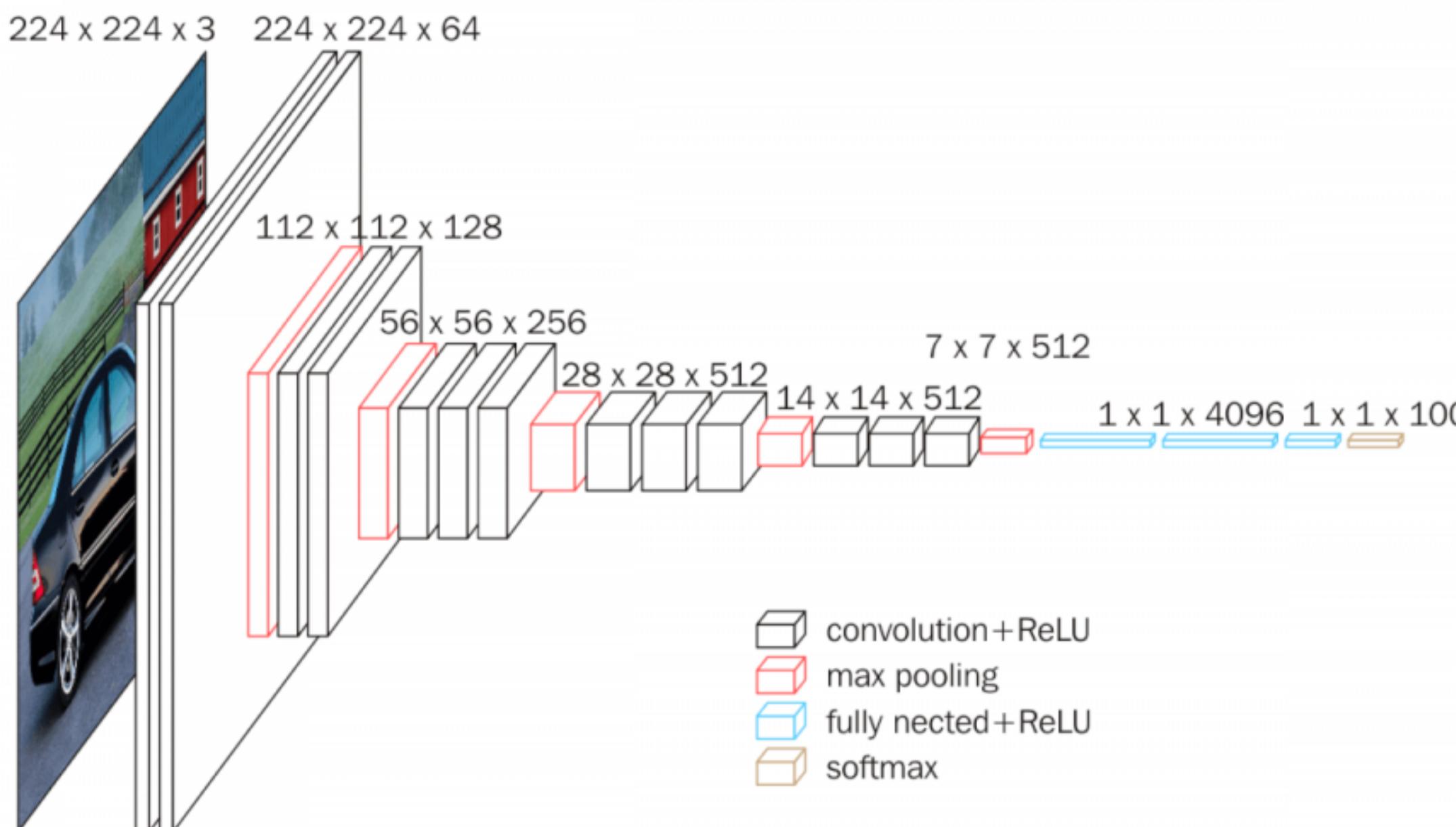
- A camada totalmente conectada é essencialmente um classificador/regressor. Os atributos extraídos na camada convolucional e na camada de *pooling* são achados (*flattened*) e colocados na camada totalmente conectada para gerar e classificar/regredir os resultados;



- Geralmente, a função Softmax( · ) é usada como a função de ativação da camada de saída, em problemas de classificação, totalmente conectada final para combinar todos os atributos locais em recursos globais e calcular a pontuação de cada tipo.

$$\text{Softmax}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^N \exp(x_j)}$$

# Camadas vs número de parâmetros



| VGG16 - Structural Details |             |     |       |        |     |      |           |        |        |    |             |       |
|----------------------------|-------------|-----|-------|--------|-----|------|-----------|--------|--------|----|-------------|-------|
| #                          | Input Image |     |       | output |     |      | Layer     | Stride | Kernel | in | out         | Param |
| 1                          | 224         | 224 | 3     | 224    | 224 | 64   | conv3-64  | 1      | 3      | 3  | 64          | 1792  |
| 2                          | 224         | 224 | 64    | 224    | 224 | 64   | conv3064  | 1      | 3      | 3  | 64          | 36928 |
|                            | 224         | 224 | 64    | 112    | 112 | 64   | maxpool   | 2      | 2      | 2  | 64          | 64    |
| 3                          | 112         | 112 | 64    | 112    | 112 | 128  | conv3-128 | 1      | 3      | 3  | 64          | 128   |
| 4                          | 112         | 112 | 128   | 112    | 112 | 128  | conv3-128 | 1      | 3      | 3  | 128         | 128   |
|                            | 112         | 112 | 128   | 56     | 56  | 128  | maxpool   | 2      | 2      | 2  | 128         | 128   |
| 5                          | 56          | 56  | 128   | 56     | 56  | 256  | conv3-256 | 1      | 3      | 3  | 128         | 256   |
| 6                          | 56          | 56  | 256   | 56     | 56  | 256  | conv3-256 | 1      | 3      | 3  | 256         | 256   |
| 7                          | 56          | 56  | 256   | 56     | 56  | 256  | conv3-256 | 1      | 3      | 3  | 256         | 256   |
|                            | 56          | 56  | 256   | 28     | 28  | 256  | maxpool   | 2      | 2      | 2  | 256         | 256   |
| 8                          | 28          | 28  | 256   | 28     | 28  | 512  | conv3-512 | 1      | 3      | 3  | 256         | 512   |
| 9                          | 28          | 28  | 512   | 28     | 28  | 512  | conv3-512 | 1      | 3      | 3  | 512         | 512   |
| 10                         | 28          | 28  | 512   | 28     | 28  | 512  | conv3-512 | 1      | 3      | 3  | 512         | 512   |
|                            | 28          | 28  | 512   | 14     | 14  | 512  | maxpool   | 2      | 2      | 2  | 512         | 512   |
| 11                         | 14          | 14  | 512   | 14     | 14  | 512  | conv3-512 | 1      | 3      | 3  | 512         | 512   |
| 12                         | 14          | 14  | 512   | 14     | 14  | 512  | conv3-512 | 1      | 3      | 3  | 512         | 512   |
| 13                         | 14          | 14  | 512   | 14     | 14  | 512  | conv3-512 | 1      | 3      | 3  | 512         | 512   |
|                            | 14          | 14  | 512   | 7      | 7   | 512  | maxpool   | 2      | 2      | 2  | 512         | 512   |
| 14                         | 1           | 1   | 25088 | 1      | 1   | 4096 | fc        |        |        | 1  | 1           | 25088 |
| 15                         | 1           | 1   | 4096  | 1      | 1   | 4096 | fc        |        |        | 1  | 1           | 4096  |
| 16                         | 1           | 1   | 4096  | 1      | 1   | 1000 | fc        |        |        | 1  | 1           | 4096  |
| Total                      |             |     |       |        |     |      |           |        |        |    | 138,423,208 |       |

**Fonte:** <https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/#:~:text=A%20pooling%20layer%20is%20a,Convolutional%20Layer>

**COMO SE OTIMIZA  
ESSA COISA?**

# Otimizadores

Existem várias versões otimizadas de algoritmos de gradiente descendente. No paradigma OO, por se encapsular o gradiente descendente em outros algoritmos, chamamos de **otimizadores**.

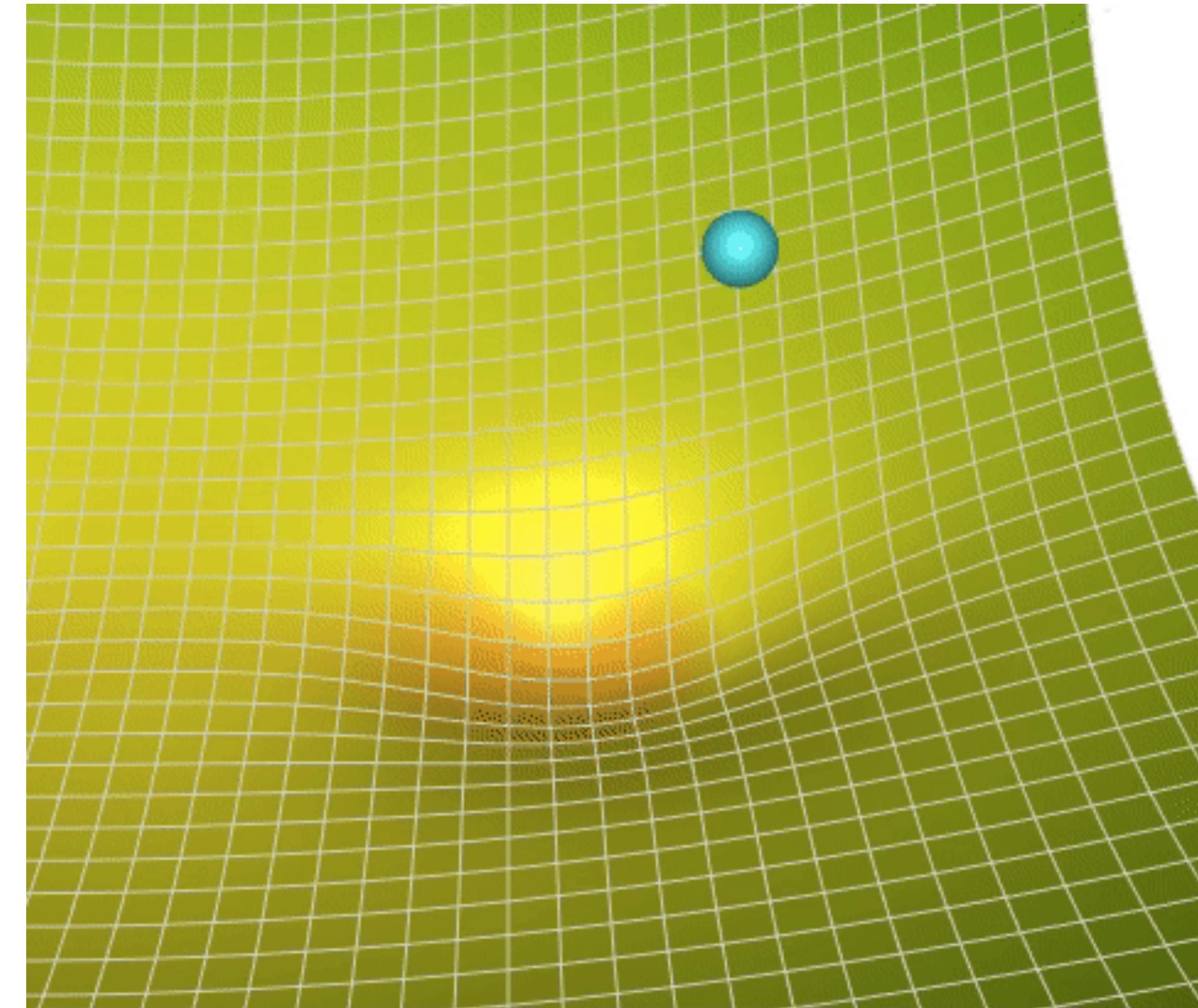
Os propósitos dos otimizados incluem, mas não estão limitados a:

- Acelerar a convergência de algoritmos;
- Prevenir ou pular fora de valores extremos locais.
- Simplificar a configuração manual de parâmetros, especialmente a taxa de aprendizado (*learning rate*).

Otimizadores comuns: otimizador **GD comum**, otimizador de **momento**, Nesterov, **AdaGrad**, AdaDelta, **RMSProp**, **Adam**, AdaMax e Nadam.

# Gradiente descendente

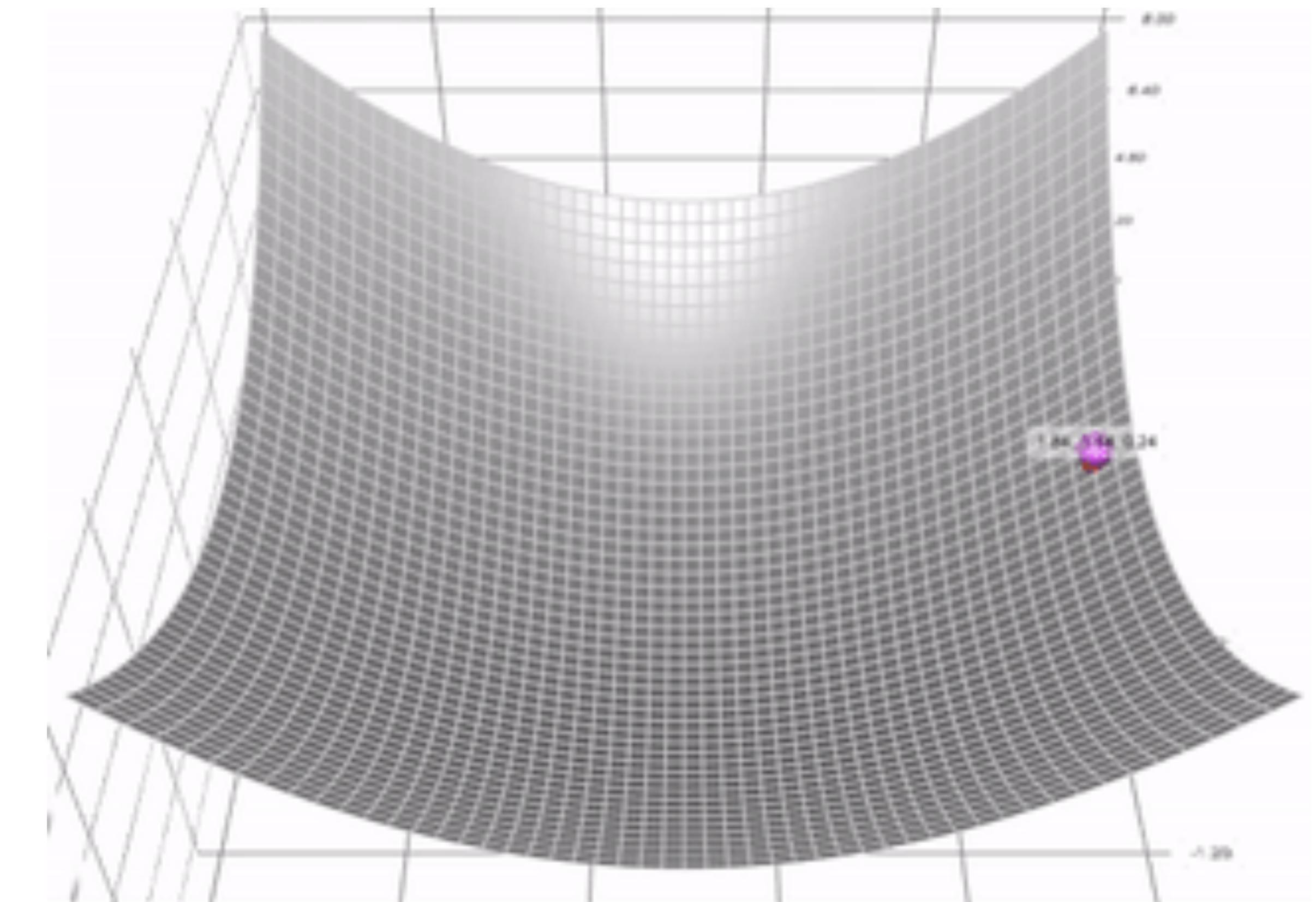
- No contexto do aprendizado de máquina, o objetivo gradiente descendente geralmente é minimizar a função de perda. Um bom algoritmo deve encontrar o mínimo de forma rápida e confiável, ou seja, não fica preso em mínimos locais, pontos de sela ou regiões de platô, logo, indo para o mínimo global;
- O algoritmo básico do gradiente descendente segue a ideia de que a direção oposta do gradiente aponta para onde está a área de menor perda. Assim, iterativamente, dá-se passos nas direções opostas aos gradientes.
- Para cada parâmetro  $\theta$ , ele faz o seguinte:



$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla J(\theta^{(t)})$$

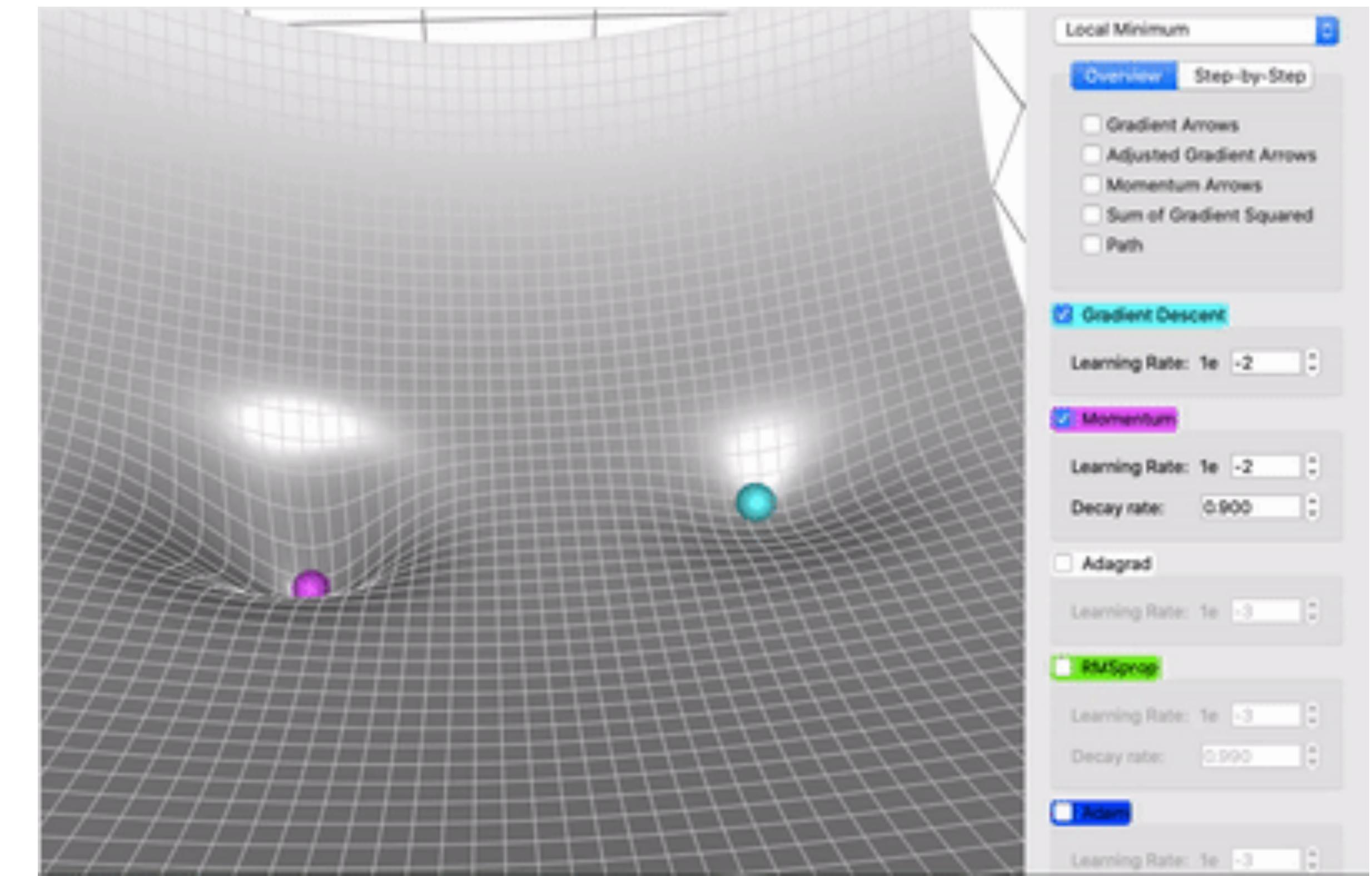
# Momentum (1)

- O gradiente descendente com o algoritmo de momento (ou Momentum para abbreviar) pega emprestado a ideia do **Momento de uma força** da Física.
- Imagine rolar uma bola dentro de uma tigela sem atrito. Em vez de parar no fundo, o impulso acumulado a empurra para frente e a bola continua rolando para frente e para trás.
- Ao aplicar o conceito de momento ao gradiente descendente, **em cada etapa**, além do gradiente regular, também adiciona-se o movimento da etapa anterior. Matematicamente, é comumente expresso como:



# Momentum (2)

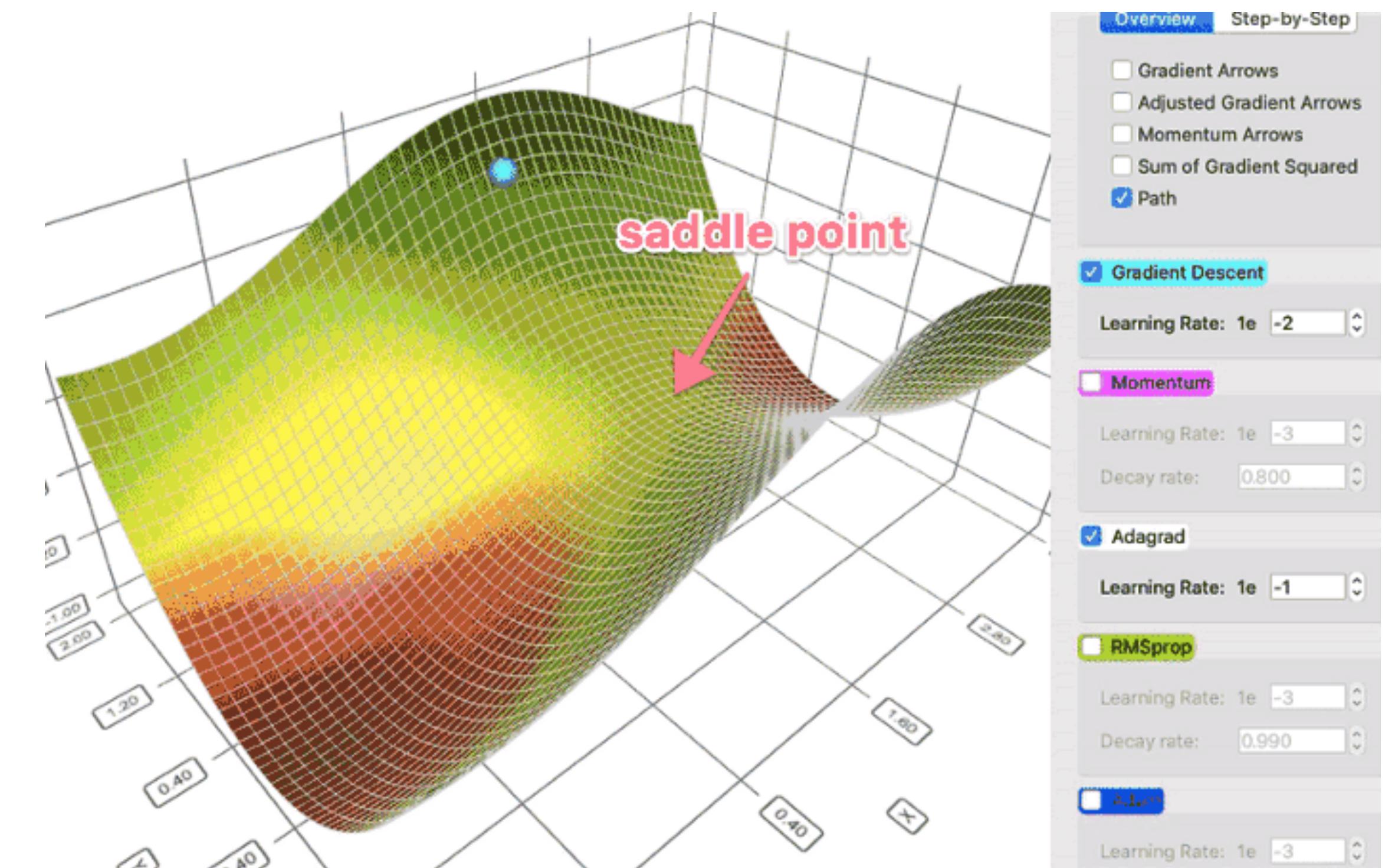
- O gradiente descendente com o algoritmo de momento (ou Momentum para abreviar) pega emprestado a ideia do **Momento de uma força** da Física.
- Imagine rolar uma bola dentro de uma tigela sem atrito. Em vez de parar no fundo, o impulso acumulado a empurra para frente e a bola continua rolando para frente e para trás.
- Ao aplicar o conceito de momento ao gradiente descendente, **em cada etapa**, além do gradiente regular, também adiciona-se o movimento da etapa anterior. Matematicamente, é comumente expresso como:



$$\theta^{(t+1)} = \underbrace{\theta^{(t)} - \eta \nabla J(\theta^{(t)})}_{\text{SGD}} + \underbrace{\alpha \nabla J(\theta^{(t-1)})}_{\text{momento}}$$

# AdaGrad (1)

- Na otimização, alguns parâmetros são muito esparsos. O gradiente médio para tais parâmetros, geralmente, é pequeno, portanto, são treinados em uma taxa muito mais lenta. Uma maneira de resolver isso é definir diferentes taxas de aprendizado para cada parâmetro, mas isso pode ficar confuso rapidamente.
- Em vez de acompanhar a soma do gradiente como momento, o algoritmo Adaptive Gradient, ou AdaGrad para abreviar, acompanha a soma do gradiente ao quadrado e usa isso para adaptar o gradiente em diferentes direções:



$$r^{(t)} = r^{(t-1)} + \nabla J(\theta^{(t)})^2,$$

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\eta}{\varepsilon + \sqrt{r^{(t)}}} \nabla J(\theta^{(t)}).$$

# AdaGrad (2)

O algoritmo de otimização AdaGrad mostra que o  $r^{(t)}$  continua aumentando enquanto a taxa de aprendizado geral continua diminuindo à medida que o algoritmo itera. Isso ocorre porque esperamos que o LR diminua à medida que o número de atualizações aumenta. Na fase inicial de aprendizado, estamos longe da solução ótima para a função perda. À medida que o número de atualizações aumenta, estamos mais próximos da solução ótima e, portanto, o LR pode diminuir.

- ❖ Ponto positivo: a taxa de aprendizado é atualizada automaticamente. À medida que o número de atualizações aumenta, a taxa de aprendizado diminui;
- ❖ Ponto negativo: o denominador continua acumulando de forma que a taxa de aprendizado acabará se tornando muito pequena e o algoritmo se tornará ineficaz.

$$r^{(t)} = r^{(t-1)} + \nabla J(\theta^{(t)})^2,$$

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\eta}{\varepsilon + \sqrt{r^{(t)}}} \nabla J(\theta^{(t)}).$$

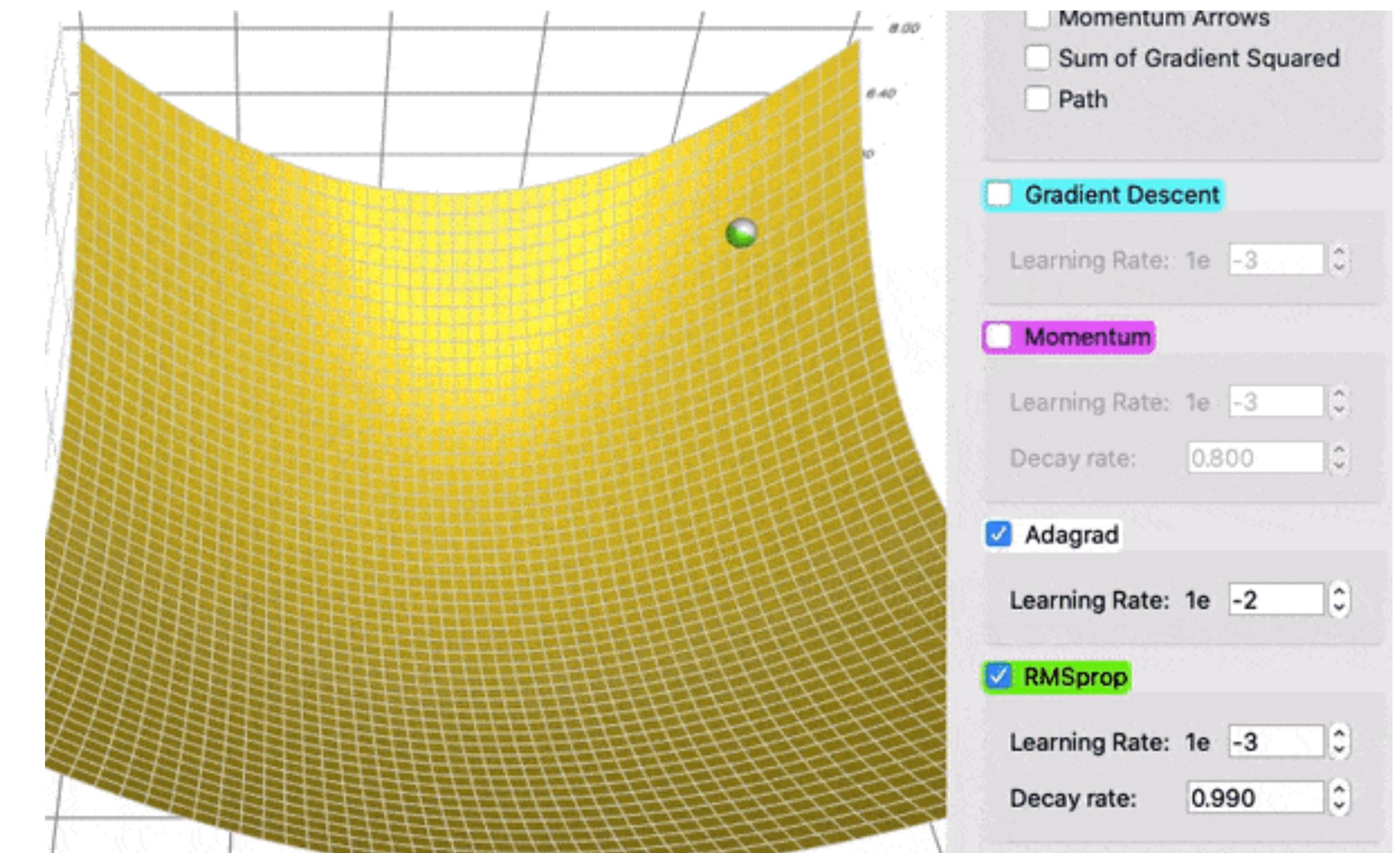
# RMSProp

- O problema do AdaGrad, no entanto, é que ele é incrivelmente lento. Isso ocorre porque a soma do gradiente ao quadrado só cresce e nunca diminui. RMSProp (para Root Mean Square Propagation) corrige esse problema adicionando um fator de decaimento  $\beta$ ;
- Mais precisamente, a soma do gradiente ao quadrado é, na verdade, atualizada para a soma decaída do gradiente ao quadrado;

$$r^{(t)} = \beta r^{(t-1)} + (1 - \beta) \nabla J(\theta^{(t)})^2.$$

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\eta}{\varepsilon + \sqrt{r^{(t)}}} \nabla J(\theta^{(t)}).$$

RMSProp



$$r^{(t)} = r^{(t-1)} + \nabla J(\theta^{(t)})^2.$$

$$\theta^{(t+1)} = \theta^{(t)} - \frac{\eta}{\varepsilon + \sqrt{r^{(t)}}} \nabla J(\theta^{(t)}).$$

AdaGrad

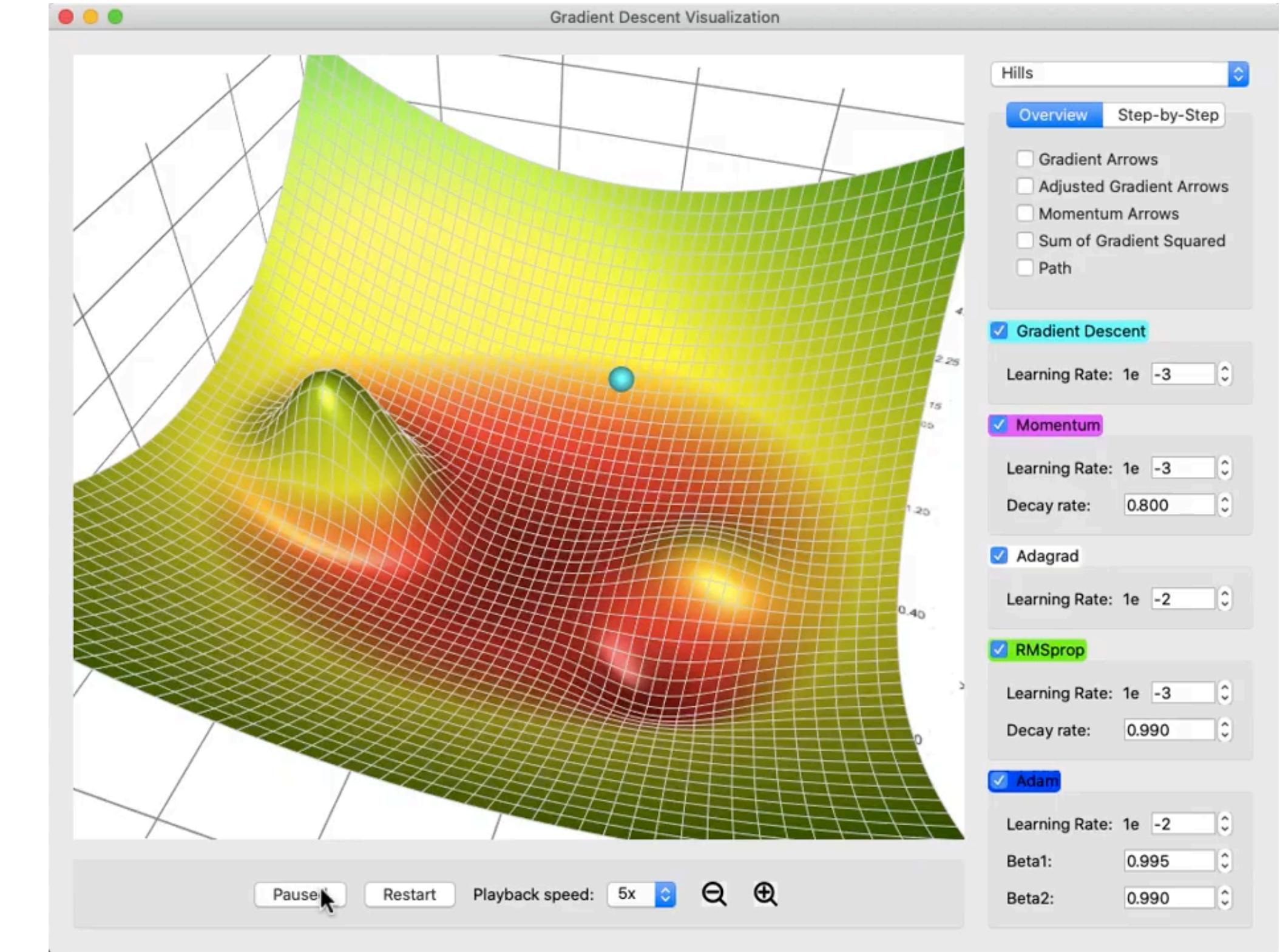
# Adam

Por último, mas não menos importante, Adam (abreviação de Adaptive Moment Estimation) obtém o melhor dos dois mundos de Momentum e RMSProp. Adam funciona empiricamente bem e, nos últimos anos, é comumente a escolha preferida dos problemas de aprendizagem profunda.

$$r^{(t)} = \beta_1 r^{(t-1)} + (1 - \beta_1) \nabla J(\theta^{(t)}), \quad \text{#Momentum}$$

$$m^{(t)} = \beta_2 m^{(t-1)} + (1 - \beta_2) \nabla J(\theta^{(t)})^2, \quad \text{#RMSProp}$$

Se  $r^{(t)}$  e  $m^{(t)}$  forem inicializados usando zero, e estarão próximos de 0 durante as iterações iniciais, especialmente quando  $\beta_1$  e  $\beta_2$  estiverem próximos de 1. Para resolver este problema, emprega-se  $\bar{r}^{(t)}$  e  $\bar{m}^{(t)}$ .



$$\bar{r}^{(t)} = \frac{r^{(t)}}{1 - \beta_1}, \quad \bar{m}^{(t)} = \frac{m^{(t)}}{1 - \beta_2} \quad \text{e} \quad \theta^{(t+1)} = \theta^{(t)} - \frac{\eta}{\varepsilon + \sqrt{r^{(t)}}} m^{(t)}$$

# Referências

- Fei-Fei Li et al. **CS231n**: Convolutional Neural Networks for Visual Recognition. <http://cs231n.stanford.edu/>. 2020, Acessado em fev, 2021.
- Juan Cruz Martinez. **Introduction to Convolutional Neural Networks CNNs**. <https://aigents.co/blog/publication/introduction-to-convolutional-neural-networks-cnns>. 2020, Acessado em fev, 2021.
- Lili Jiang. **A Visual Explanation of Gradient Descent Methods (Momentum, AdaGrad, RMSProp, Adam)**. <https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c>. 2020, Acessado em fev, 2021.
- HUAWEI. **Deep Learning Overview**. 2020, Acessado em fev, 2021.
- Mohamed Alimoussa. **Is Deep Learning always overcome Machine Learning?** <https://medium.com/machine-learning-vs-deep-learning-who-wins/machine-learning-vs-deep-learning-who-wins-in-classification-problems-1fe97126e816>. 2018, Acessado em abr, 2021.