

SICO70

SISTEMAS INTELIGENTES 2

Aula 07 - Redes Neurais Convolucionais
(Convolutional Neural Networks - CNNs)

Prof. Rafael G. Mantovani



Apucarana - PR, Brasil

Universidade Tecnológica Federal do Paraná (UTFPR)
Engenharia de Computação

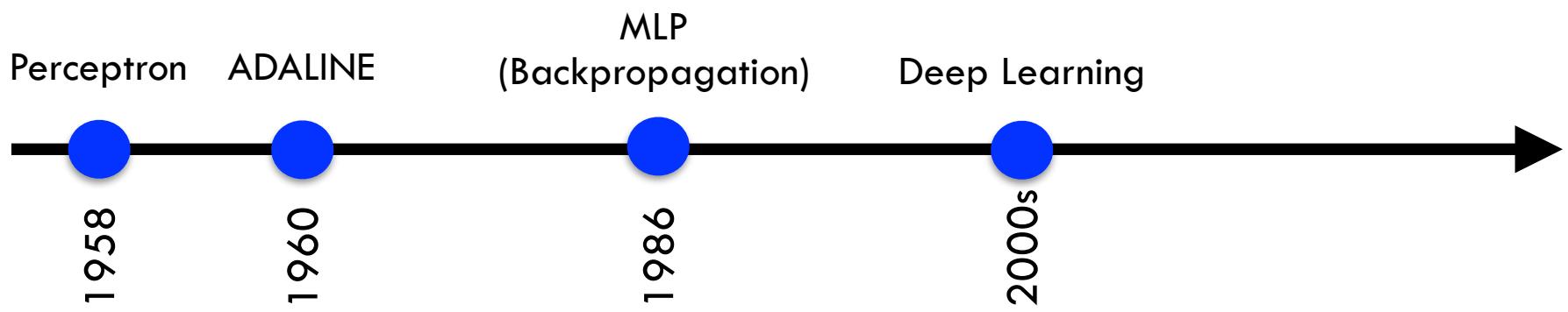
Roteiro

- 1 Introdução
- 2 *Convolutional Neural Networks (CNNs)*
- 3 Modelagem de CNNs
- 4 Hands on / Exemplos
- 5 Referências

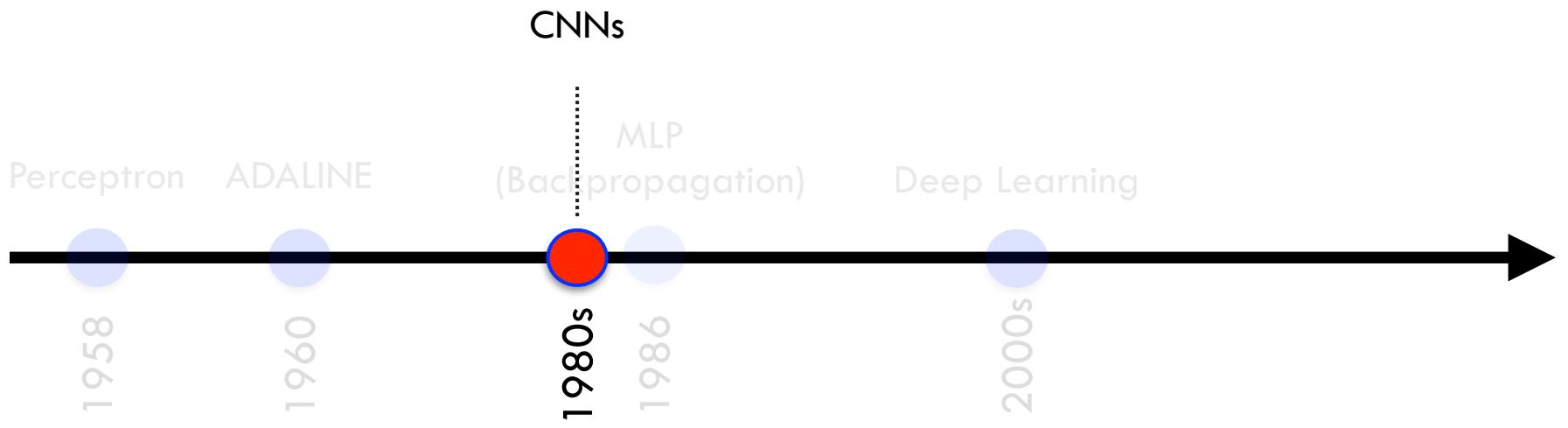
Roteiro

- 1 Introdução**
- 2 *Convolutional Neural Networks (CNNs)***
- 3 Modelagem de CNNs**
- 4 Hands on / Exemplos**
- 5 Referências**

Introdução



Introdução



Introdução

- **Redes Neurais Convolucionais (CNNs)**
 - surgiram a partir de estudos realizados com o cérebro, na década de 1980 → córtex visual
 - Porém só nos últimos anos tivemos avanços “sobre-humanos” em tarefas visuais complexas → poder/avanço computacional

Introdução

■ Redes Neurais Convolucionais (CNNs)

- surgiram a partir de estudos realizados com o cérebro, na década de 1980 → córtex visual
- Porém só nos últimos anos tivemos avanços “sobre-humanos” em tarefas visuais complexas → poder/avanço computacional

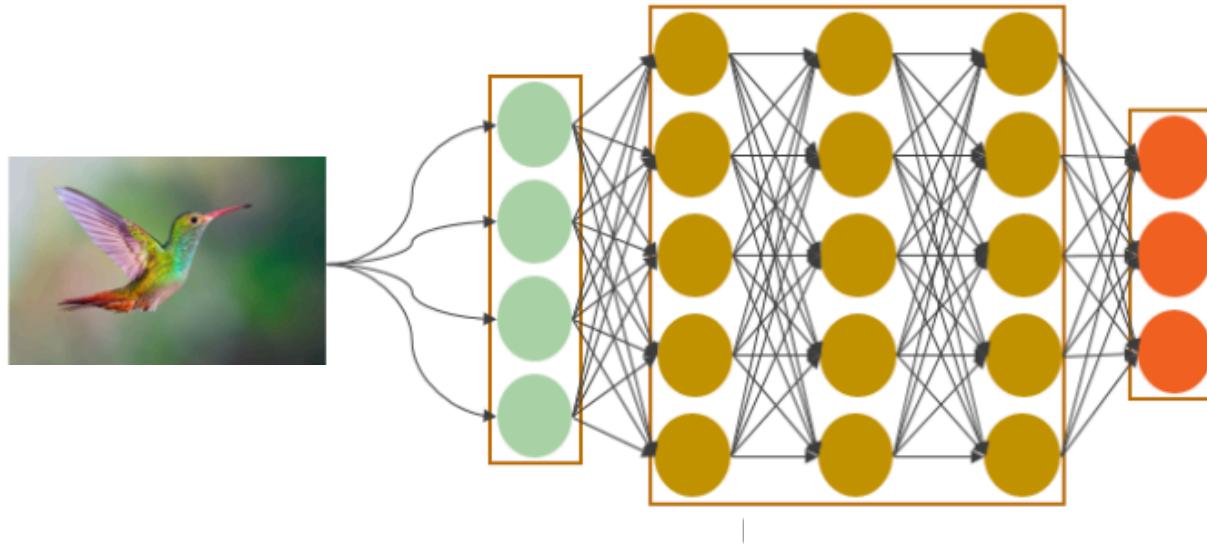
□ **Tarefas:**

- reconhecimento/classificação de imagens
- processamento de linguagem natural
- reconhecimento de voz

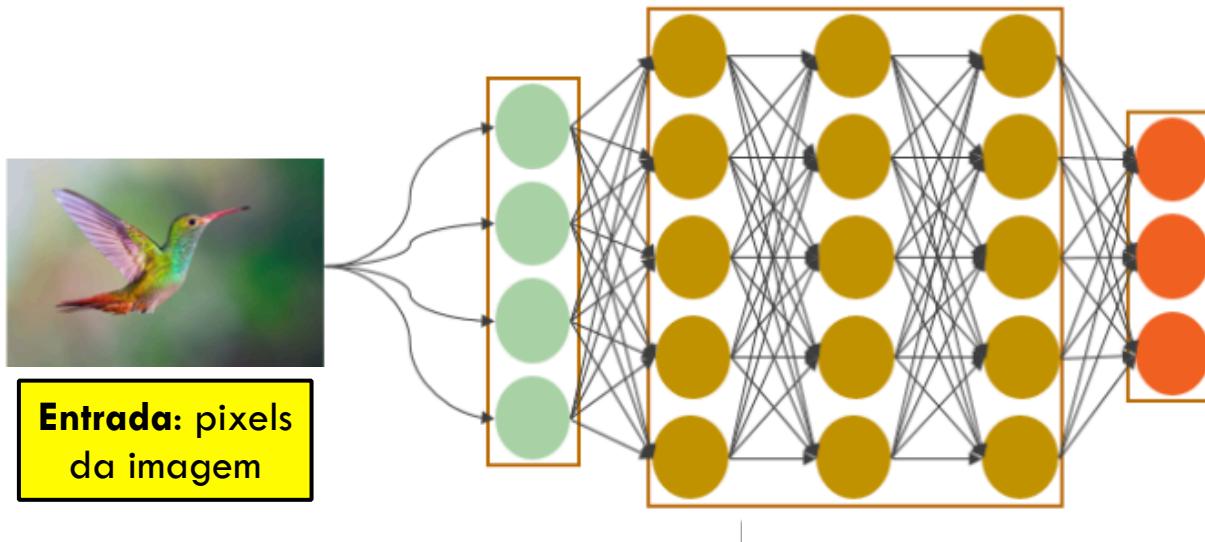
Introdução



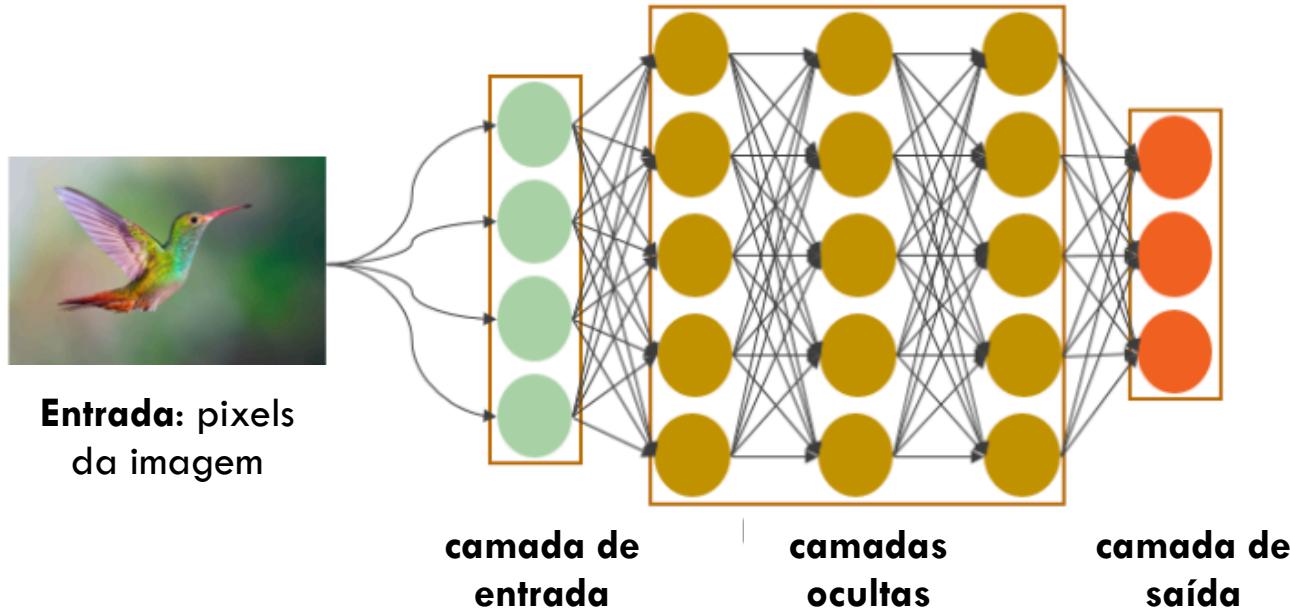
Introdução



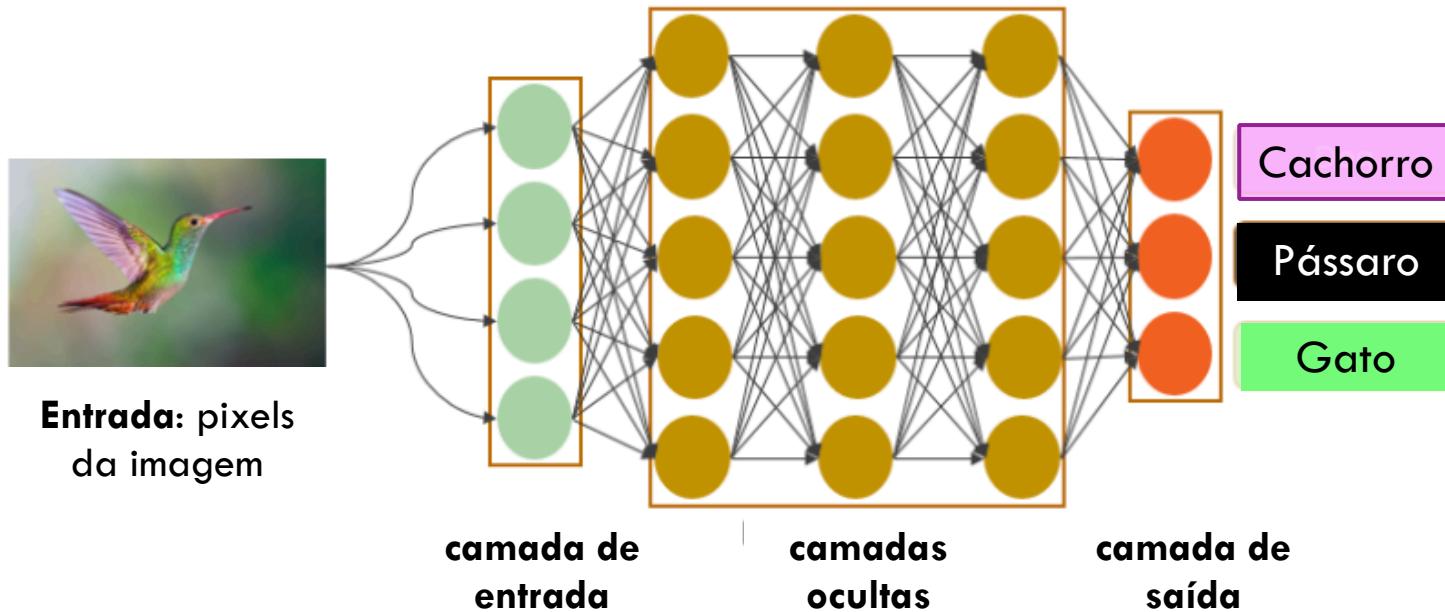
Introdução



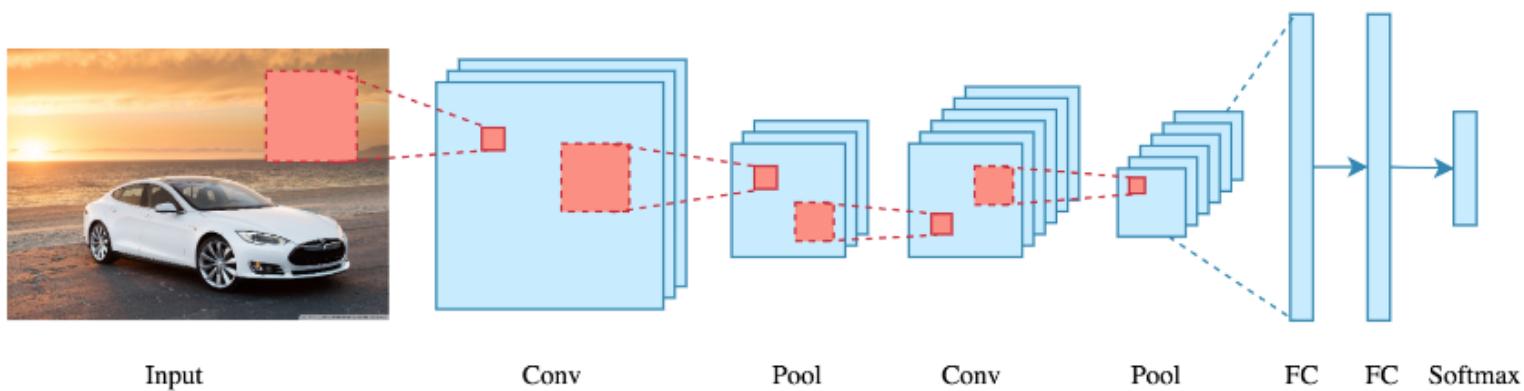
Introdução



Introdução

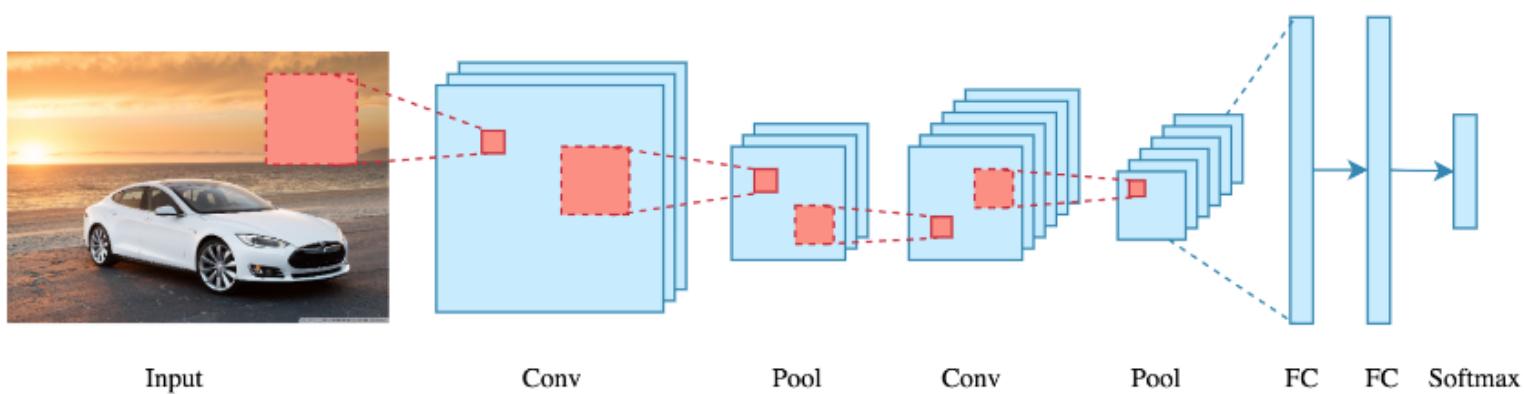


Introdução



Introdução

Primeiro Vislumbre. Mas porque tem essa estrutura? Como chegamos até esse tipo de RNAs?



Roteiro

- 1 Introdução
- 2 *Convolutional Neural Networks (CNNs)*
- 3 Modelagem de CNNs
- 4 Hands on / Exemplos
- 5 Referências

CNNs

- **Inspiração:** córtex visual

CNNs

- **Inscrição: córtex visual**



CNNs

□ Inspiração: córtex visual

- trabalhos de *David H Hubel & Torsten Wiesel* (1958-59)
- experimentos com gatos e macacos
 - neurônios possuem um **pequeno campo receptivo local**
 - reagem a **estímulos visuais** localizados em uma **região limitada** do campo visual



CNNs

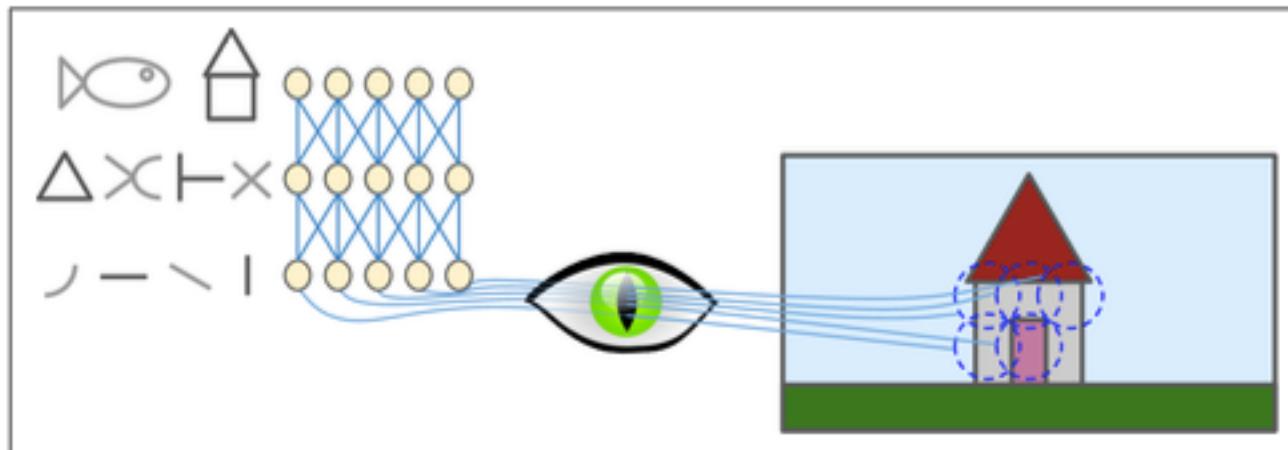


Figura de: Aurélien Gerón (2019)

CNNs

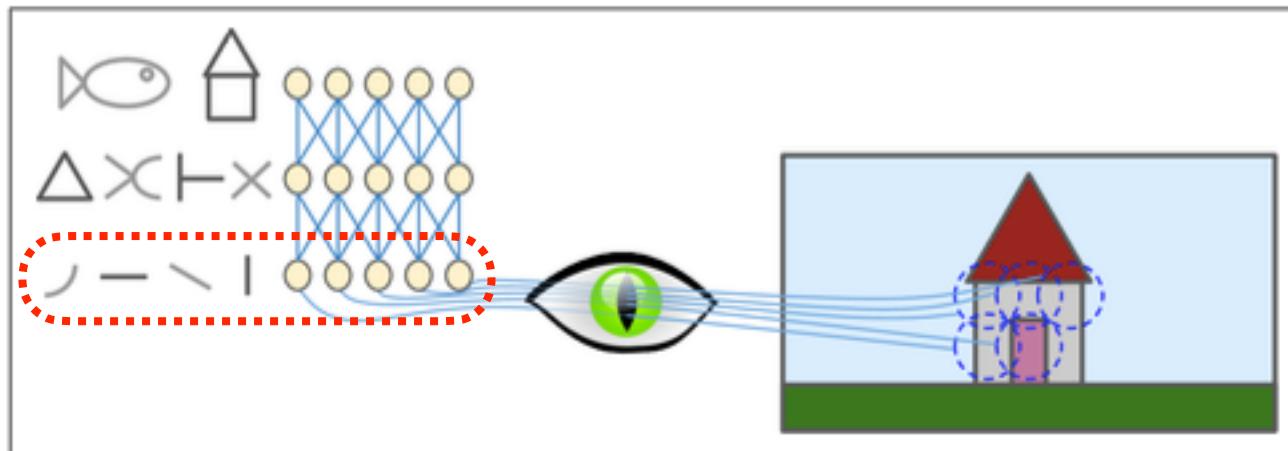


Figura de: Aurélien Gerón (2019)

CNNs

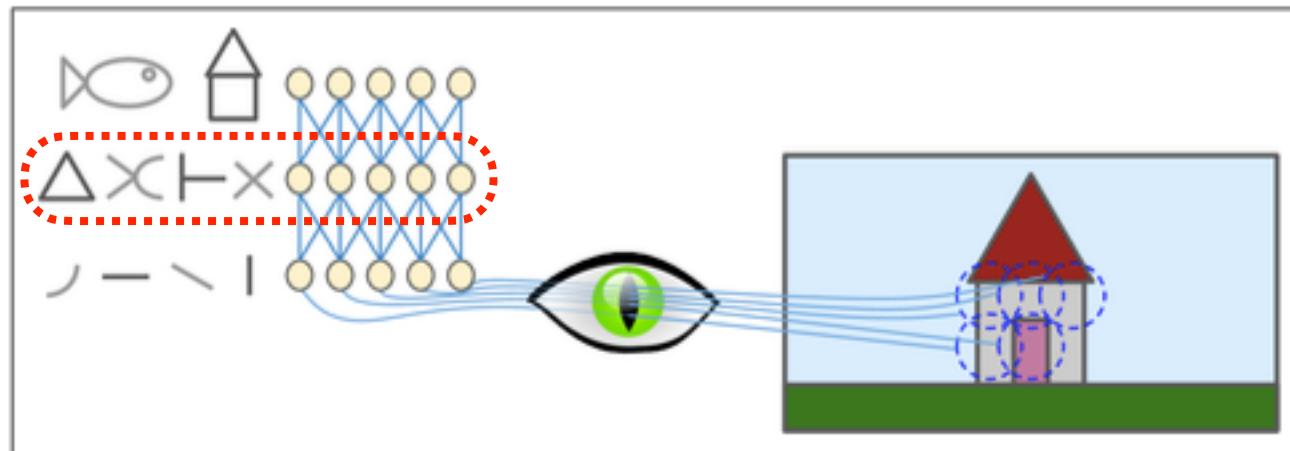


Figura de: Aurélien Gerón (2019)

CNNs

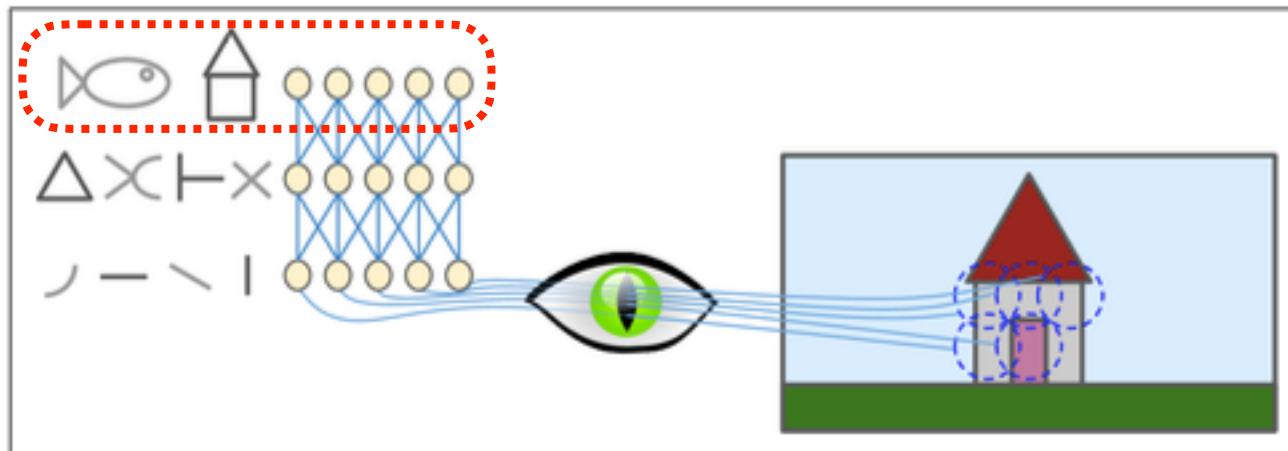


Figura de: Aurélien Gerón (2019)

CNNs

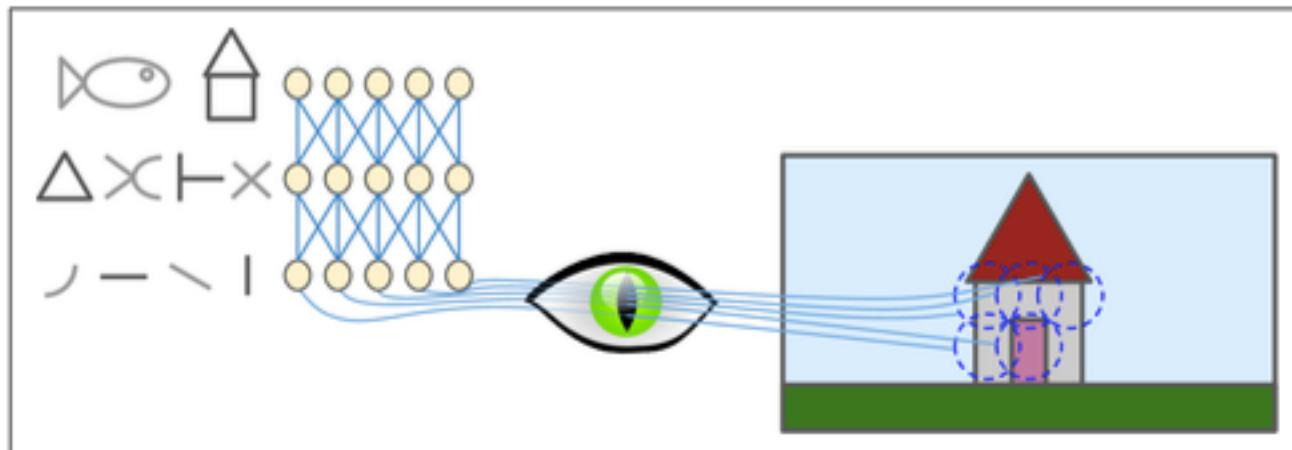
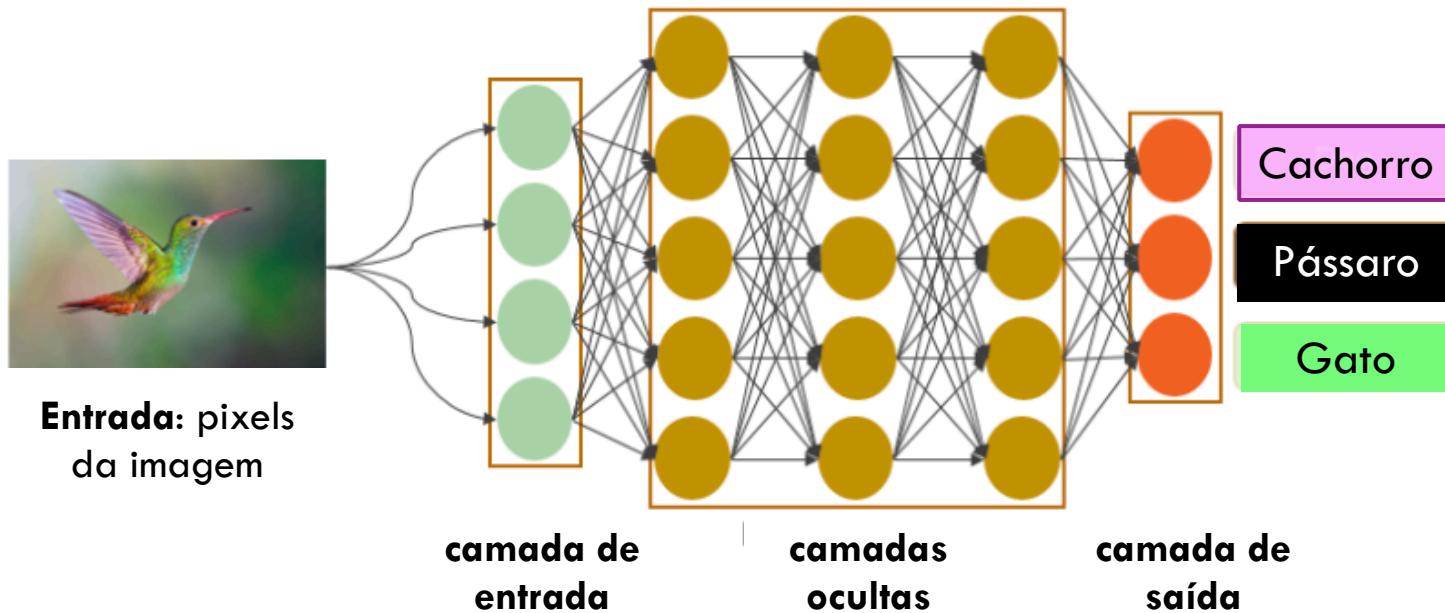


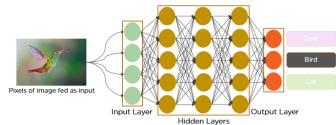
Figura de: Aurélien Gerón (2019)

- alguns neurônios reagem à linhas horizontais/verticais/diagonais, etc
- alguns neurônios tem campo de percepção maior que os outros, e reagem a padrões mais complexos (**combinação de padrões simples**)
- há uma região de inserção entre o campo de percepção de diferentes neurônios

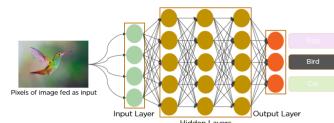
CNNs



CNNs



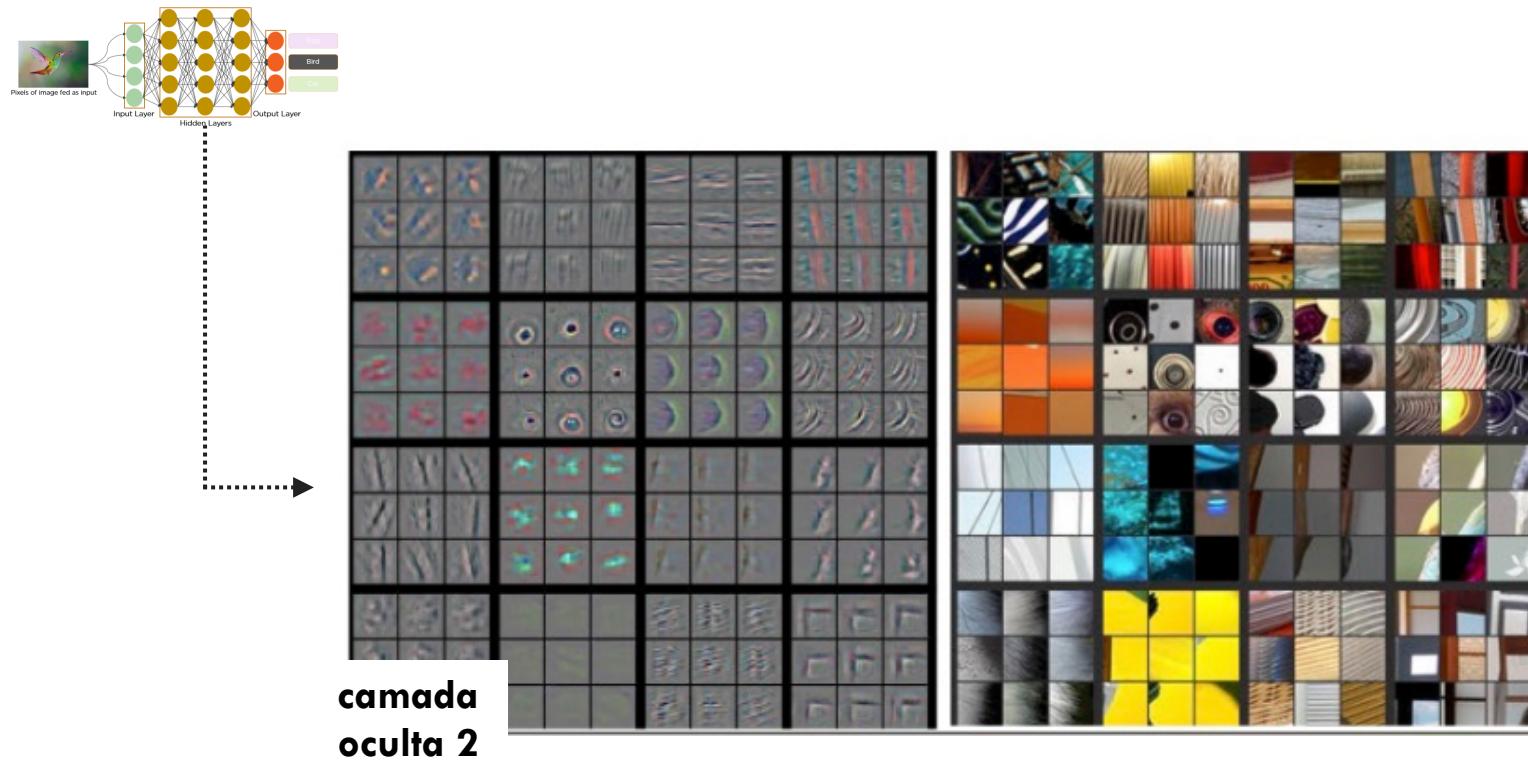
CNNs



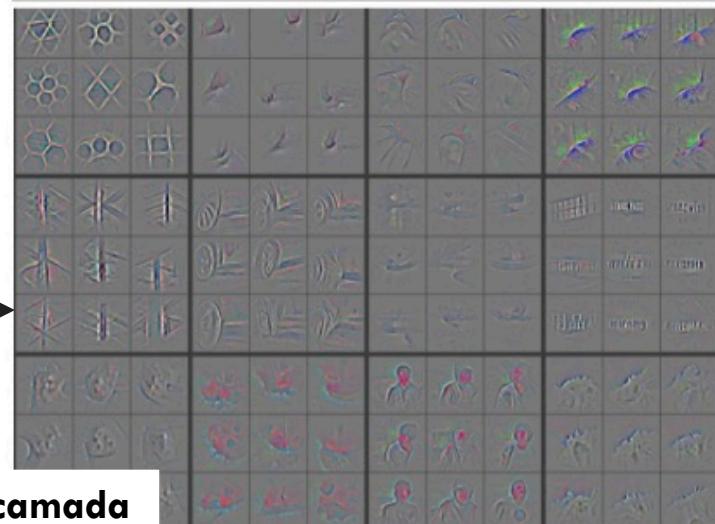
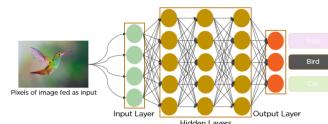
**camada
oculta 1**



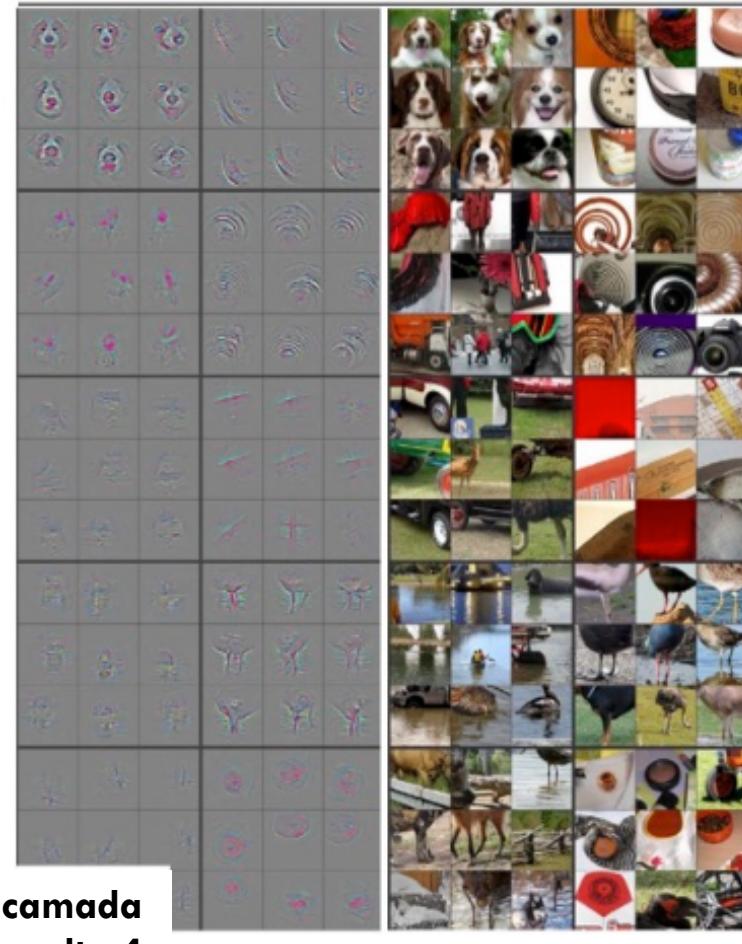
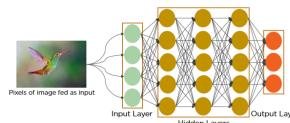
CNNs



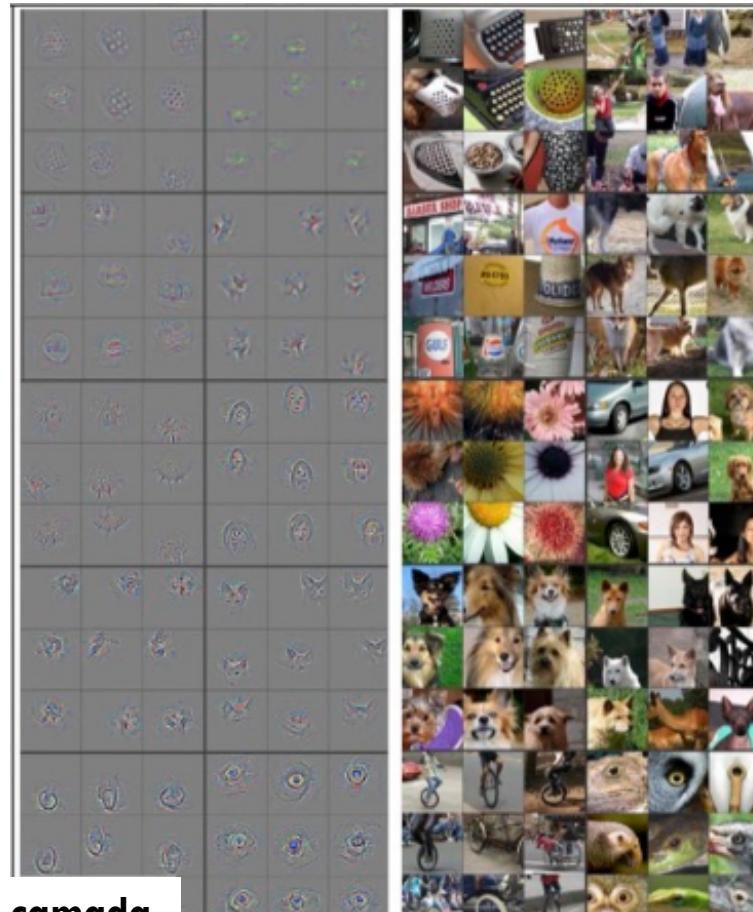
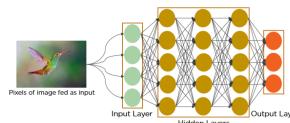
CNNs



CNNs



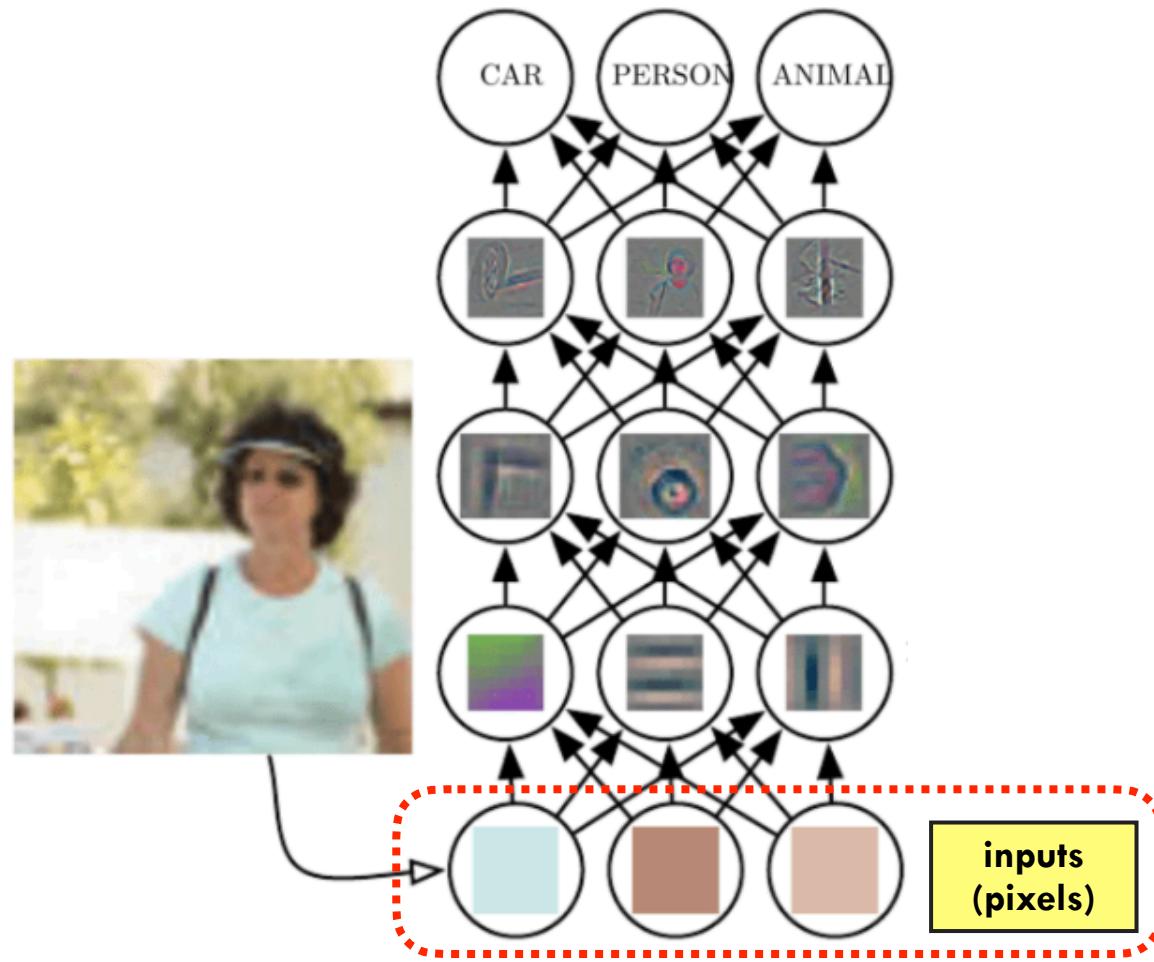
CNNs



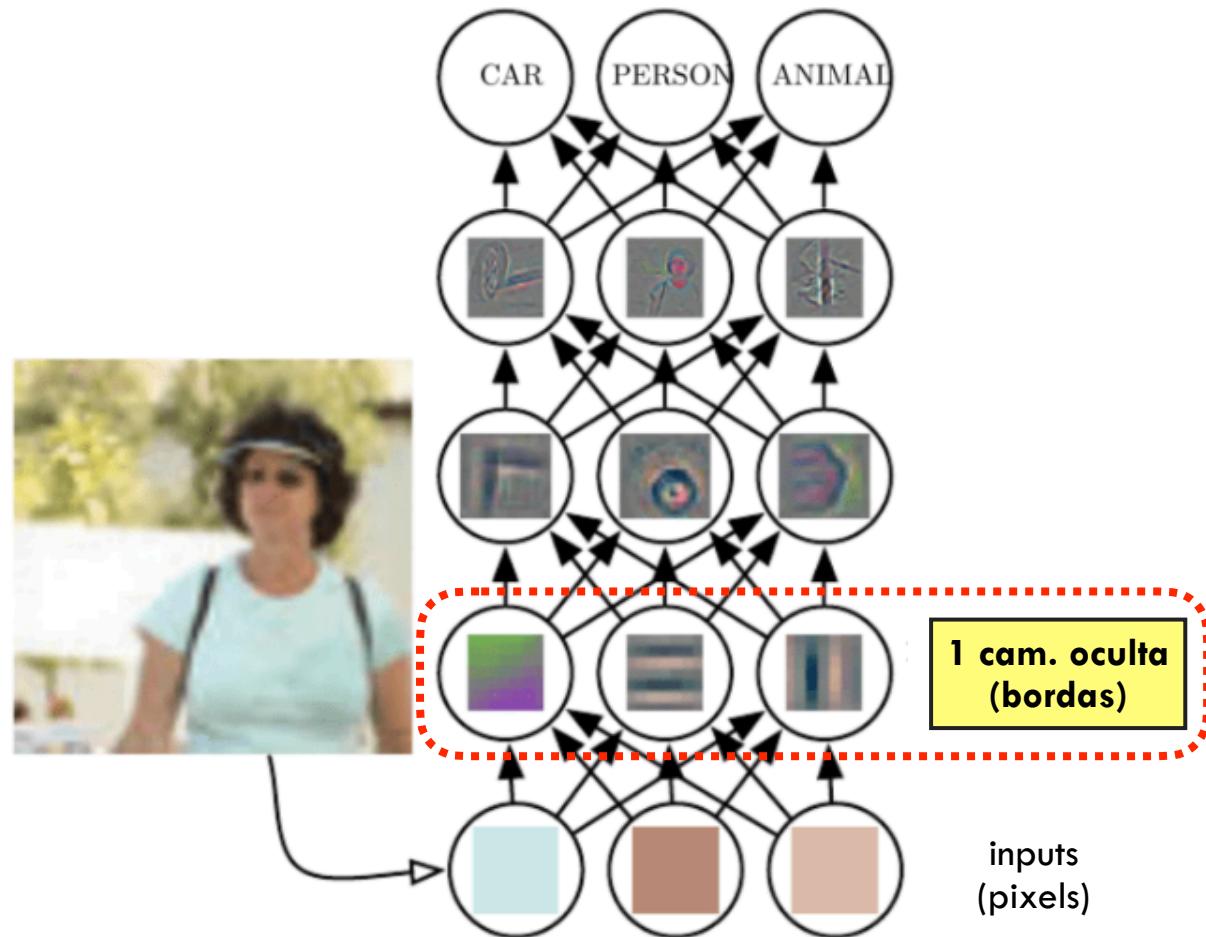
CNNs



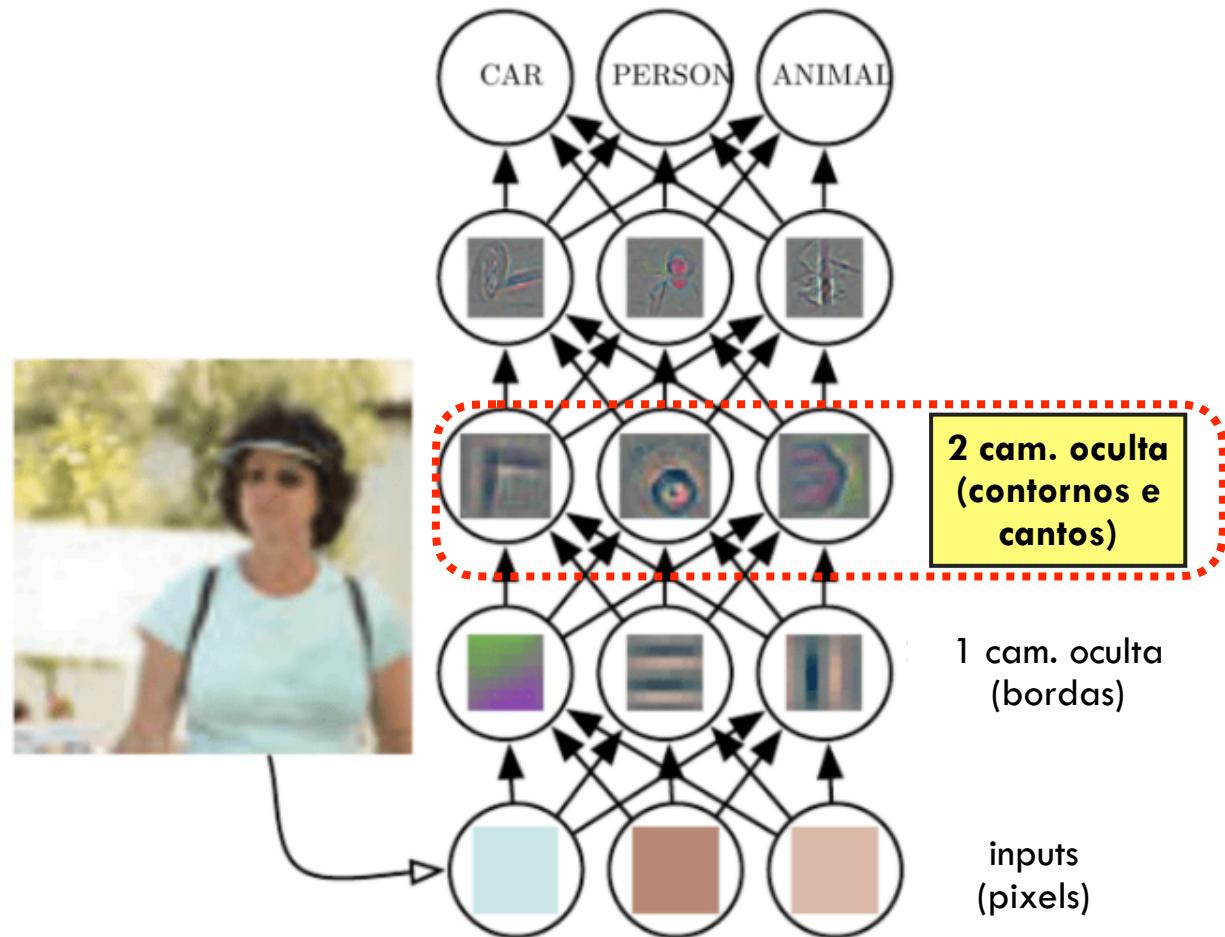
CNNs



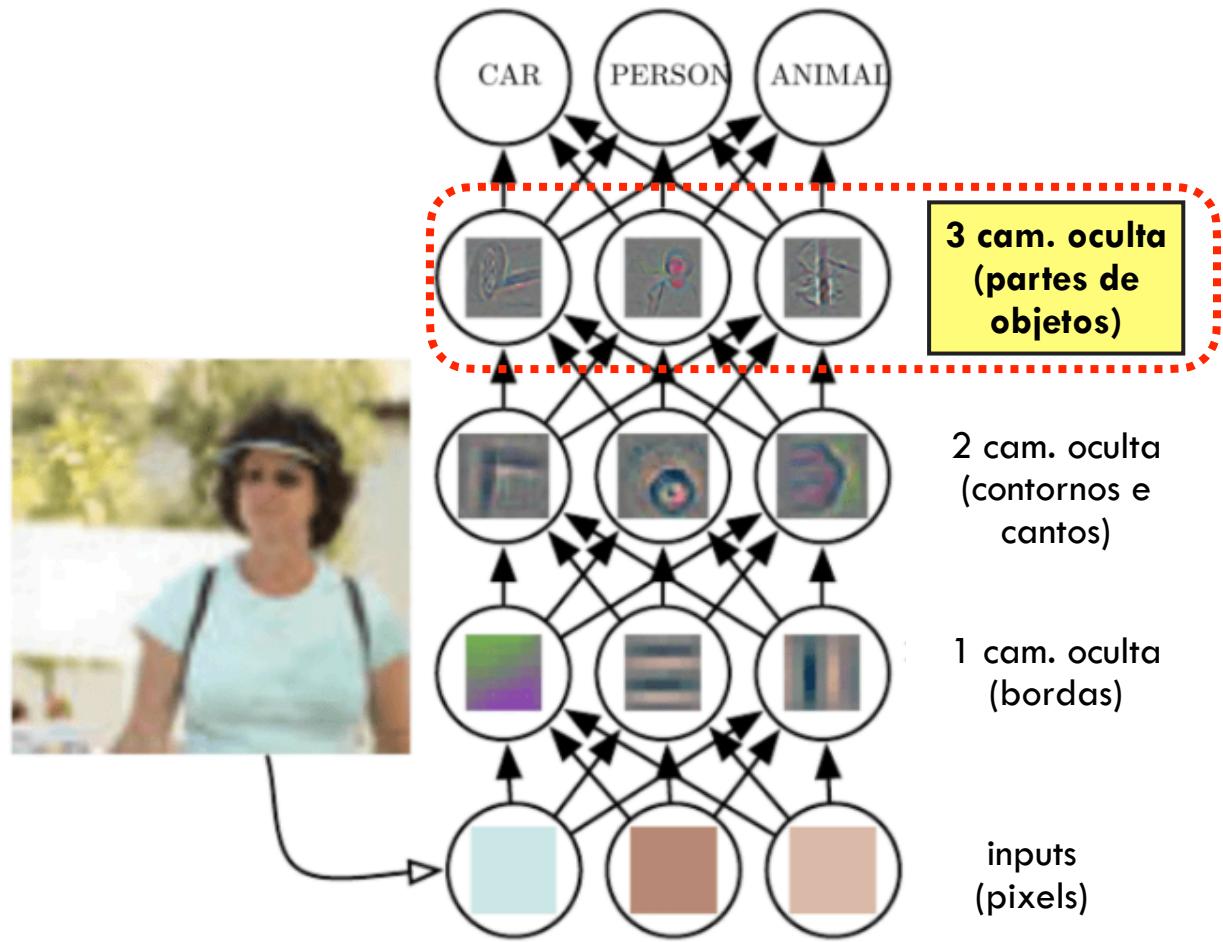
CNNs



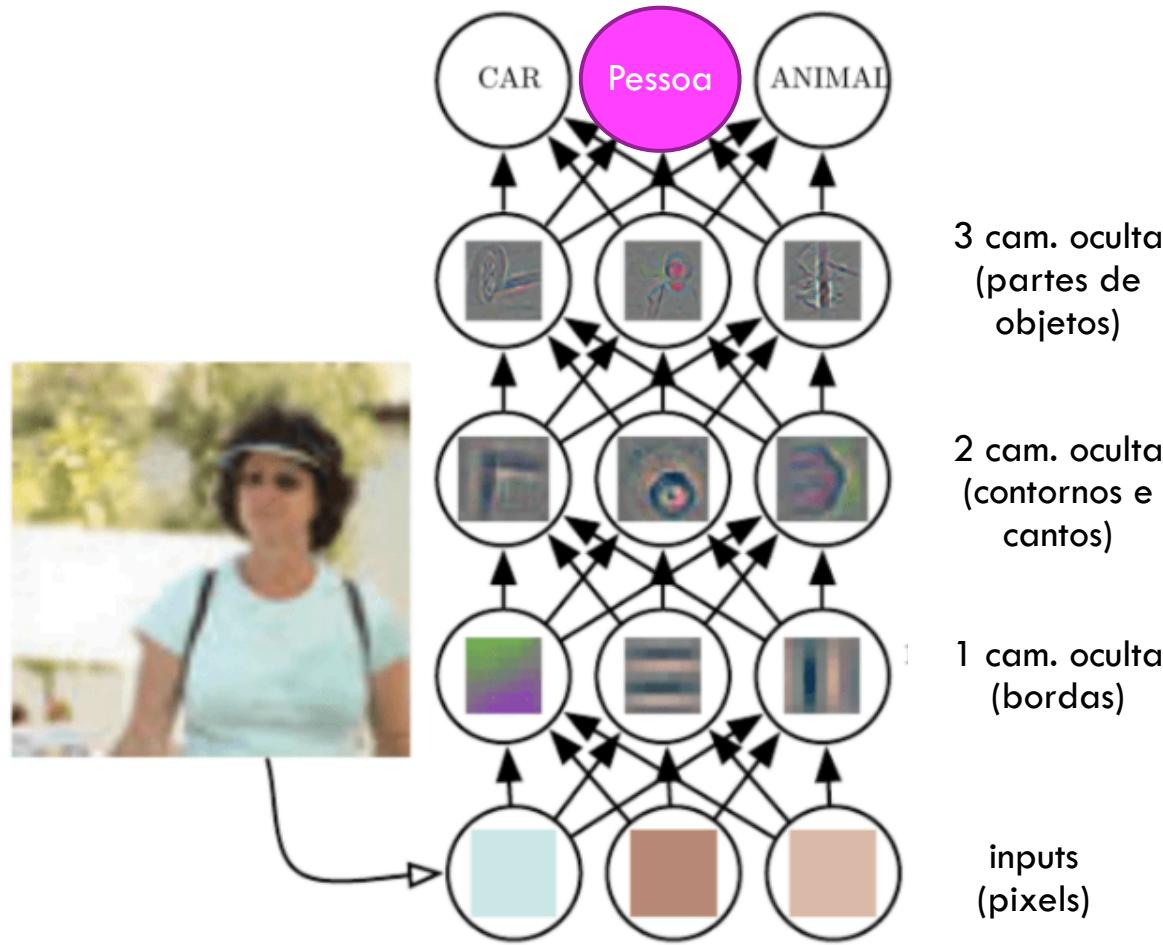
CNNs



CNNs



CNNs



Roteiro

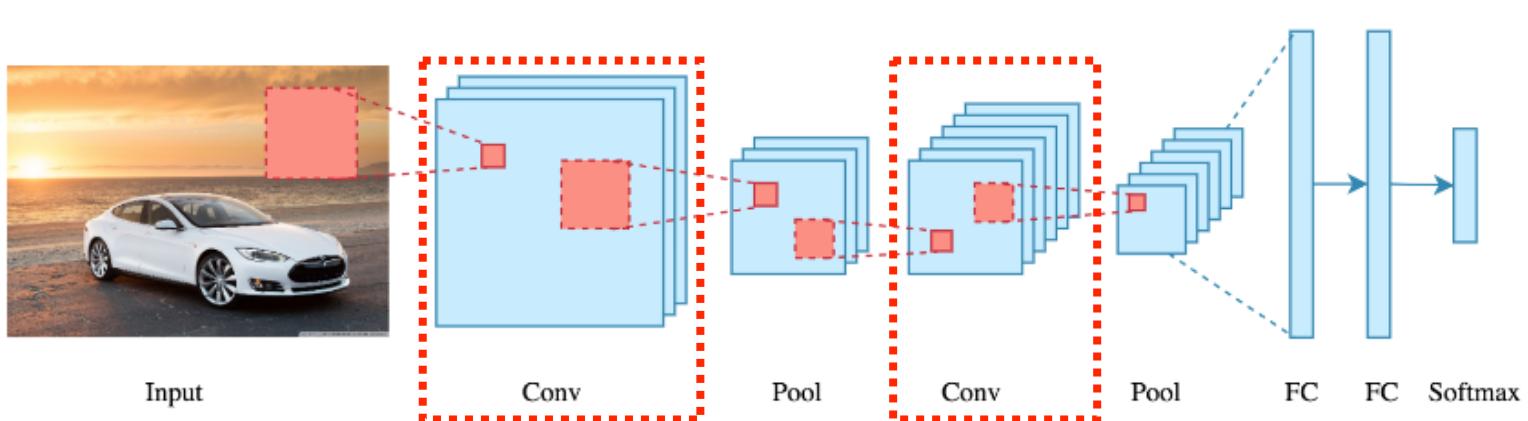
- 1 Introdução
- 2 *Convolutional Neural Networks (CNNs)*
- 3 Modelagem de CNNs
- 4 Hands on / Exemplos
- 5 Referências

CNNs

□ Redes Neurais Convolucionais (CNNs)

- 1º modelo é de 1980 (*neocognitron*)
- 1988 → duas novas estruturas
 - Camadas Convolucionais
 - Poolings

Camadas convolucionais



Camadas convolucionais

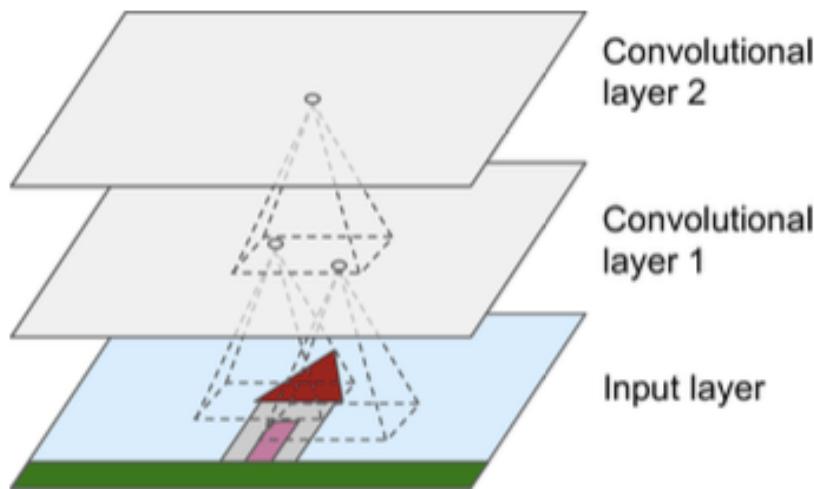
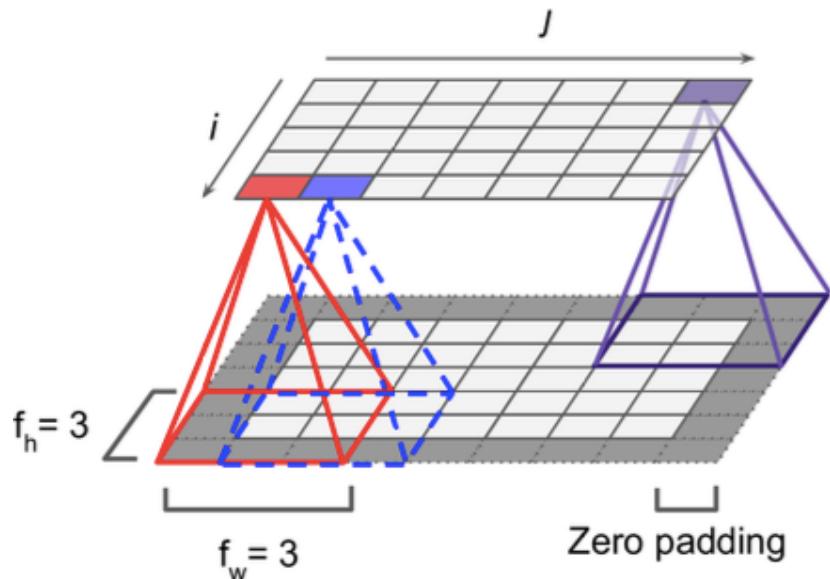


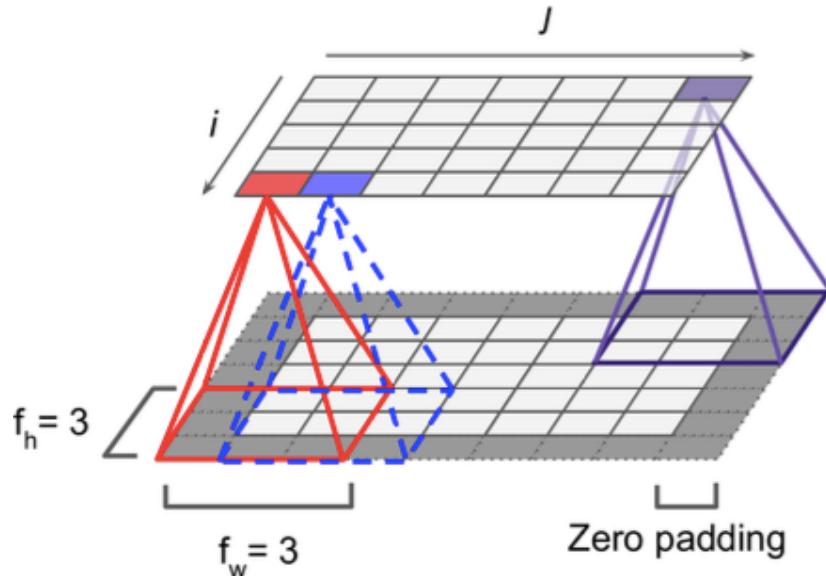
Figura de: Aurélien Gerón (2019)

- neurônios da 1a camada são conectados a alguns pixels de seus campos receptivos
 - não são totalmente conectados (como na MLP)
- neurônios da 2a camada são conectados a poucos neurônios localizados em um pequena região da camada 1
- kernel ($f_w \times f_h$)

Camadas convolucionais

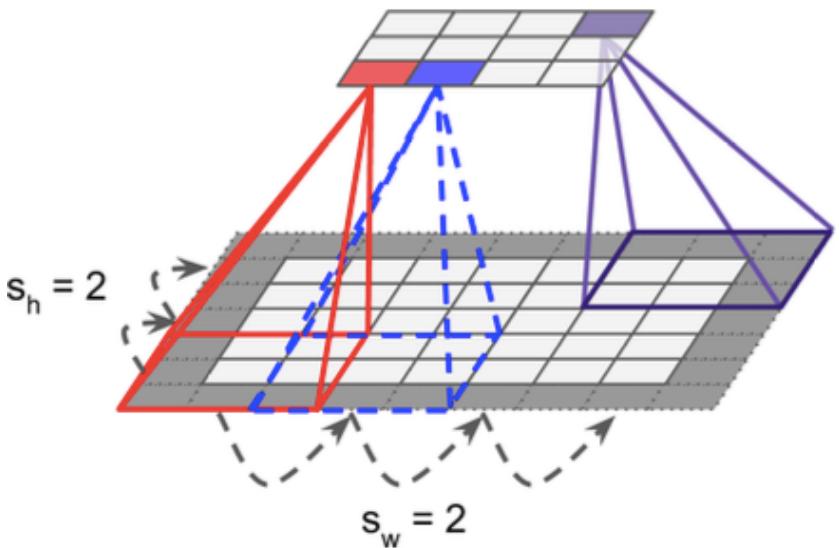


Camadas convolucionais

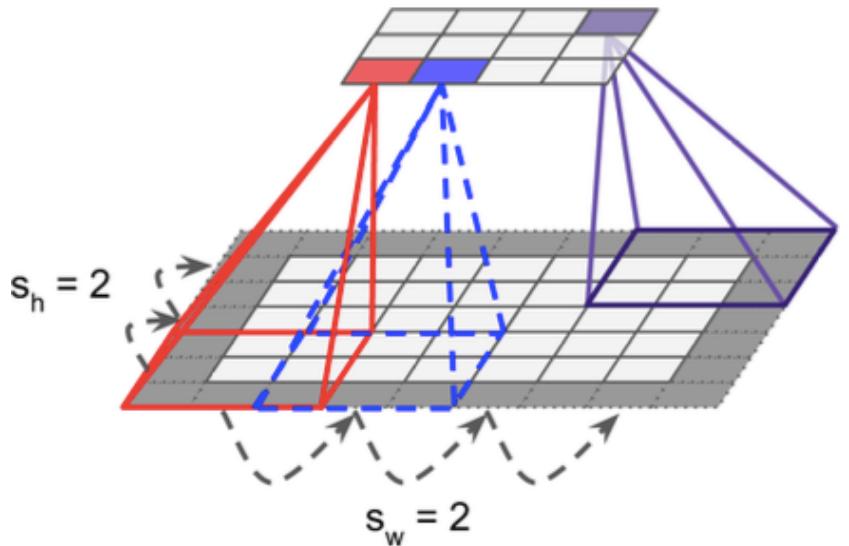


1. neurônio i,j está conectado nas saídas da camada prévia (*kernel* - 3×3)

Camadas convolucionais

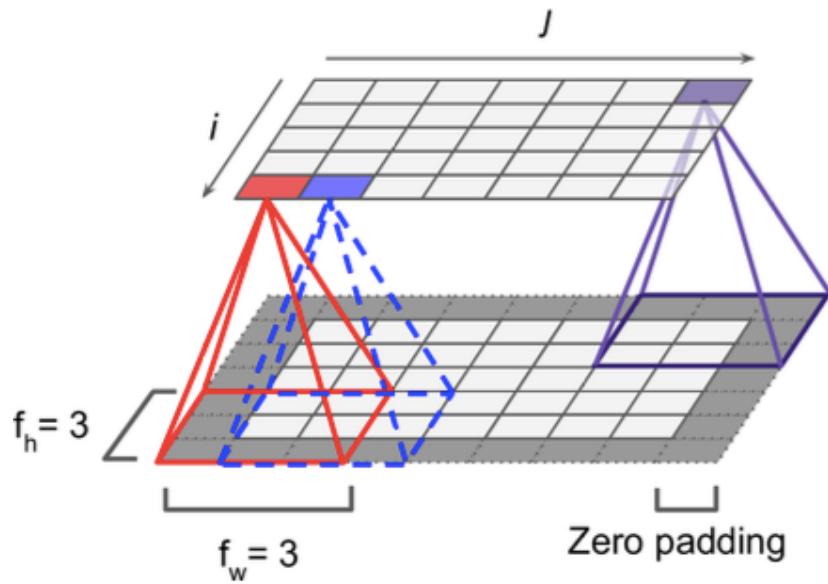


Camadas convolucionais

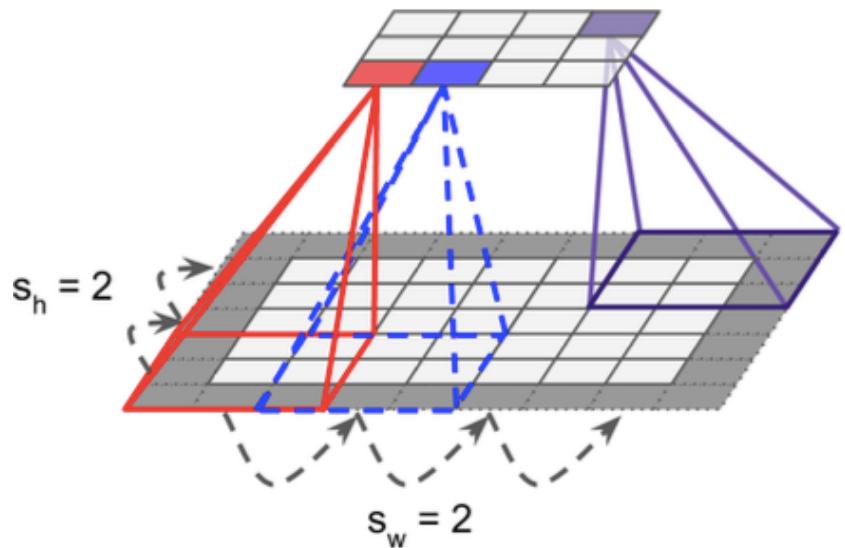


2. stride (deslocamento)
Neurônios na borda aplicam
padding

Camadas convolucionais



1. neurônio i,j está conectado nas saídas da camada prévia (*kernel* - 3×3)



2. *stride* (deslocamento)
Neurônios na borda aplicam **padding**

Filtros

- Pesos dos neurônios formam pequenas imagens do tamanho do campo de percepção
- estas imagens funcionam como **filtros**, detectando padrões simples
- durante o treinamento a camada aprende quais filtros são os mais úteis, e as demais camadas vão combinando estes filtros para encontrar padrões mais complexos

Filtros

imagem com padrões verticais ressaltados

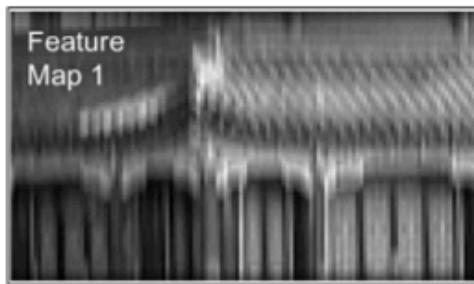
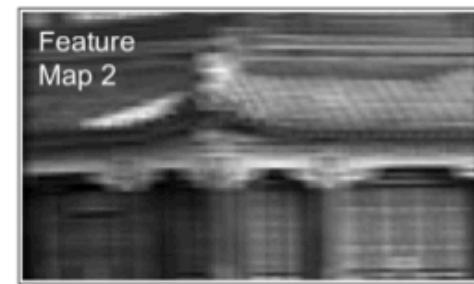


imagem com padrões horizontais ressaltados



Vertical filter



Horizontal filter



filtros
(imagens)



imagem original

Múltiplos filtros

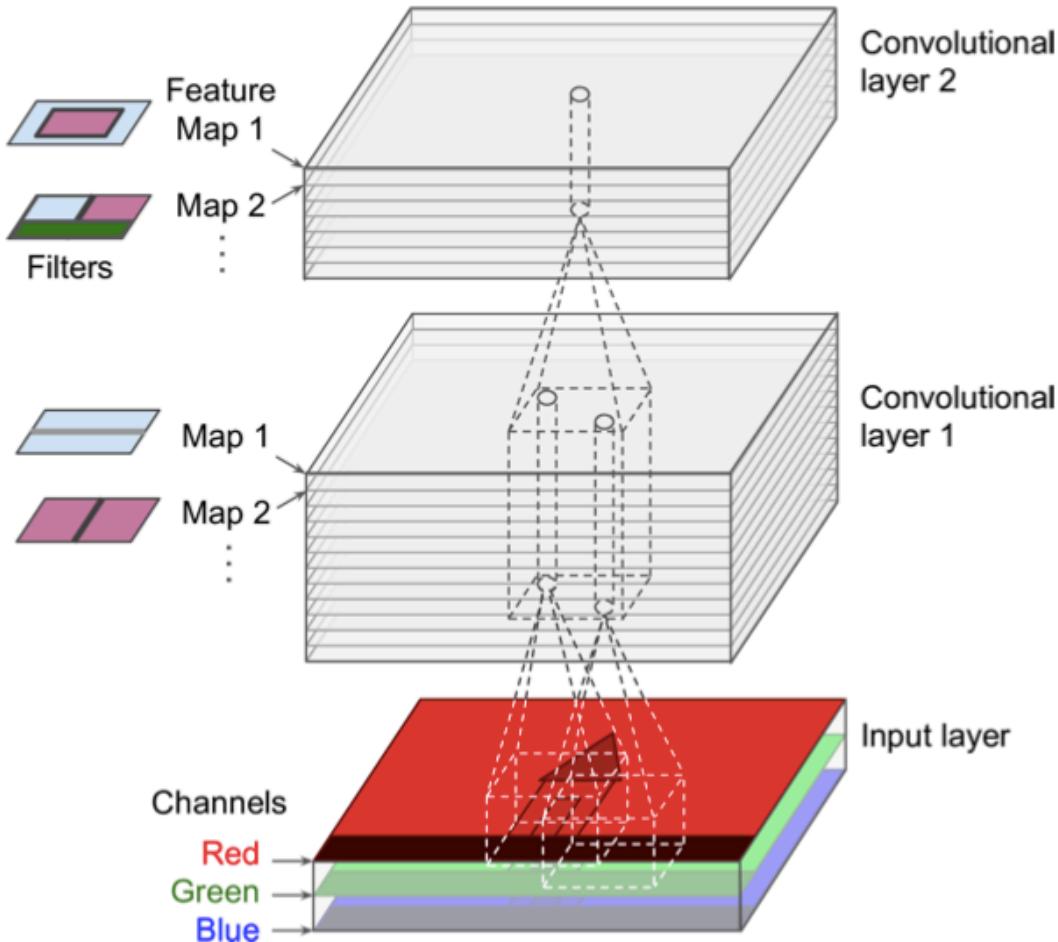


Figura de: Aurélien Gerón (2019)

Múltiplos filtros

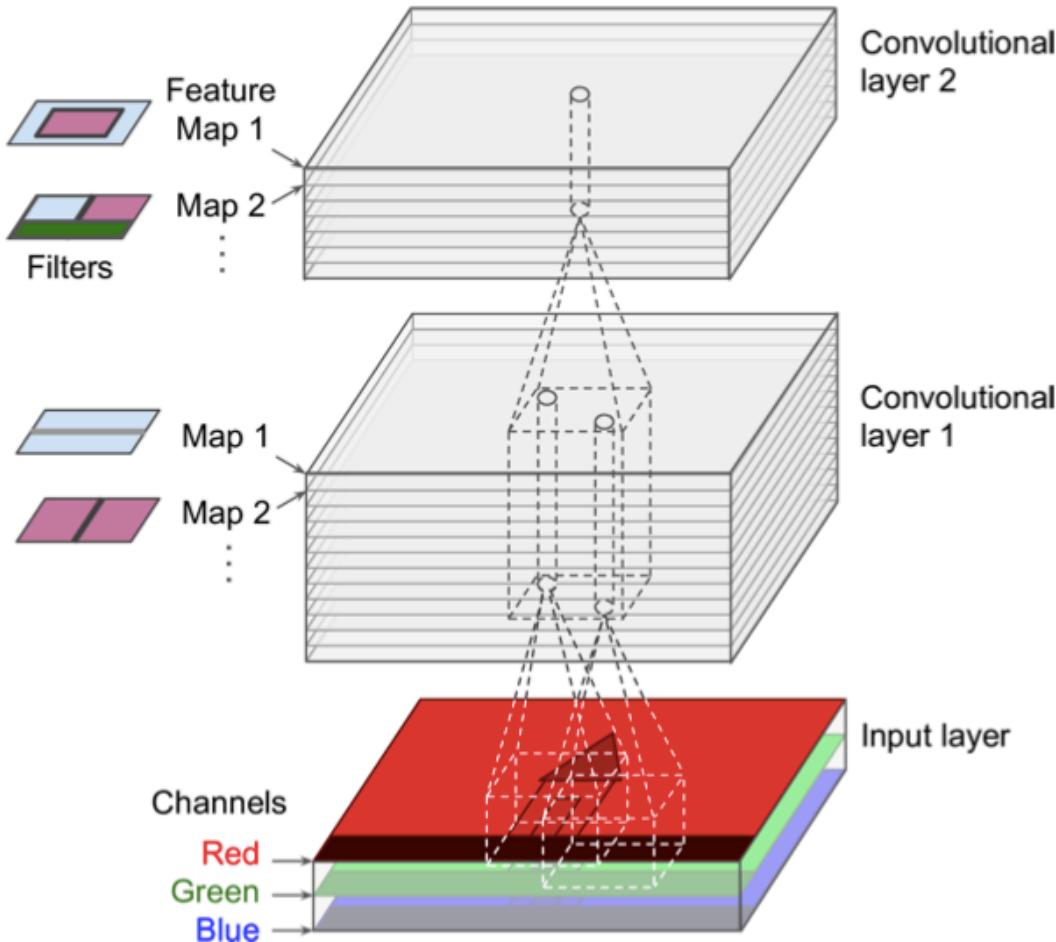


Figura de: Aurélien Gerón (2019)

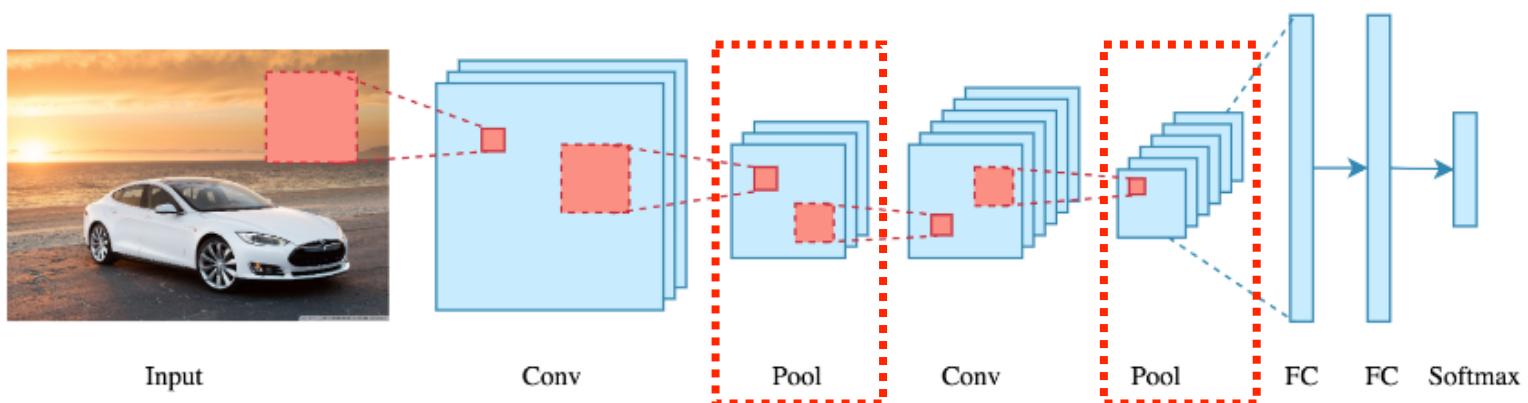
- Mas ... saída de cada camada convolucional tem **múltiplos filtros** (matriz 3D)
- 1 neurônios para cada filtro (*feature map*)
- todos os neurônios de um mesmo mapa compartilham os mesmos parâmetros (pesos, bias)
- campo de percepção do neurônio se estende a todos filtros

Múltiplos filtros

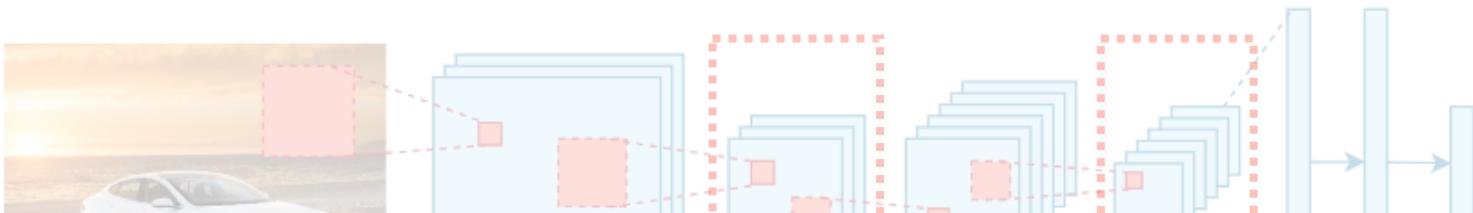
□ Resumindo:

- cada camada convolucional aplica simultaneamente múltiplos **filtros** treináveis às suas entradas (*inputs*), sendo capaz de identificar múltiplas características em qualquer lugar
- Imagens → RGB (3 canais)
 - cinza = média desses valores
 - cinza = $(R + G + B)/3$

Camadas de Pooling



Camadas de Pooling



- **Objetivo:** reduzir a imagem de entrada na camada (sub-amostragem/*subsample*)
 - Reduz o custo computacional, memória, e parâmetros no modelo (pesos)
 - Cada neurônio é conectado a um campo de visão limitado (*kernel*)
 - Camada de *pooling* **não tem pesos** !
 - Agregação dos valores (máximo ou média)

Camadas de Pooling

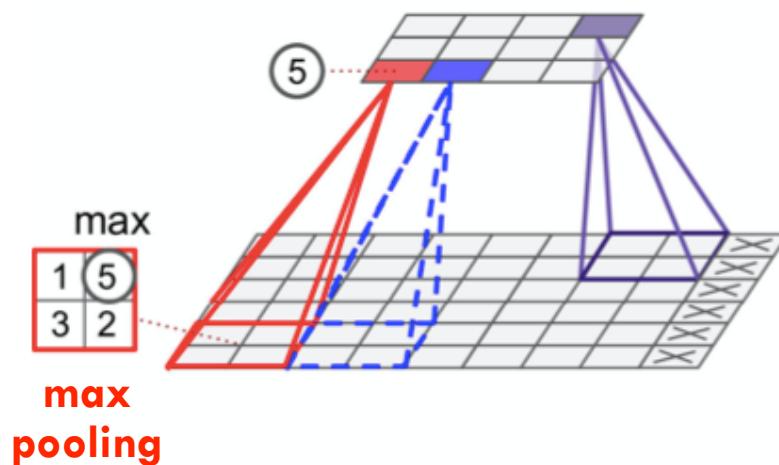


imagem saída

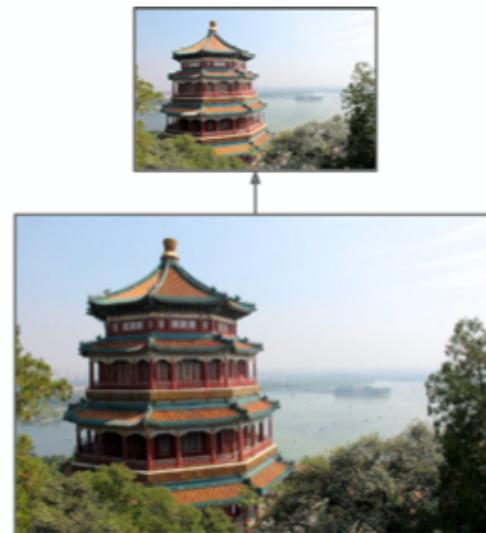
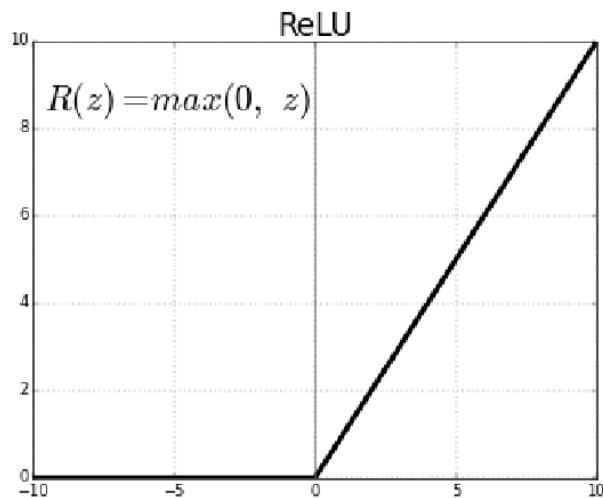


imagem entrada

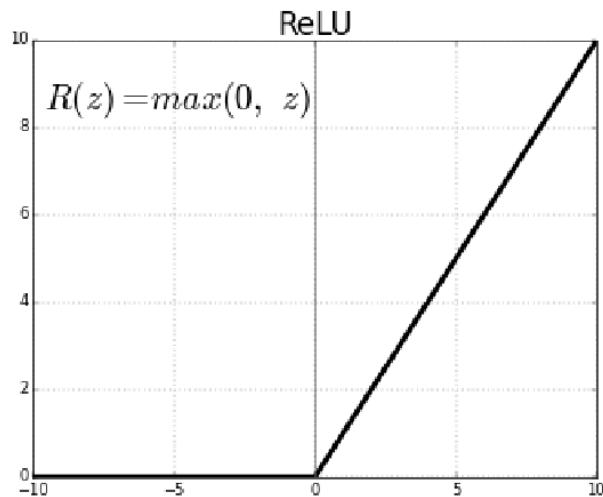
Funções de Ativação

- Rectifier Linear Units (ReLU) → camadas intermediárias



Funções de Ativação

- Rectifier Linear Units (ReLU) → camadas intermediárias



- Acelera a convergência do gradiente
- previne o desaparecimento do gradiente
- adiciona **não-linearidade** no processo de treinamento

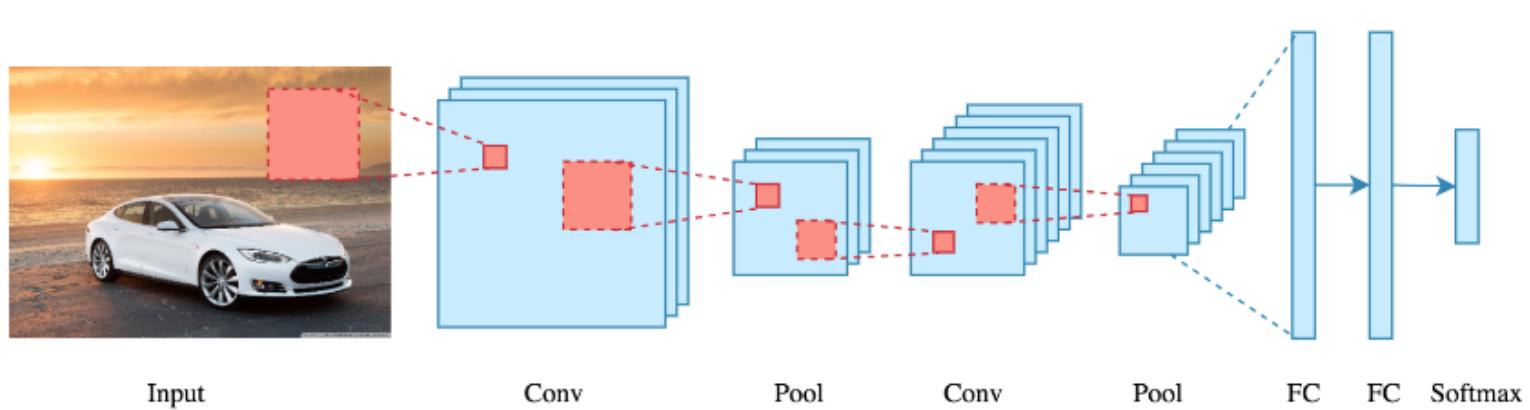
$$R(z) = \max(0, z)$$

Arquitetura Geral

Arquitetura Geral

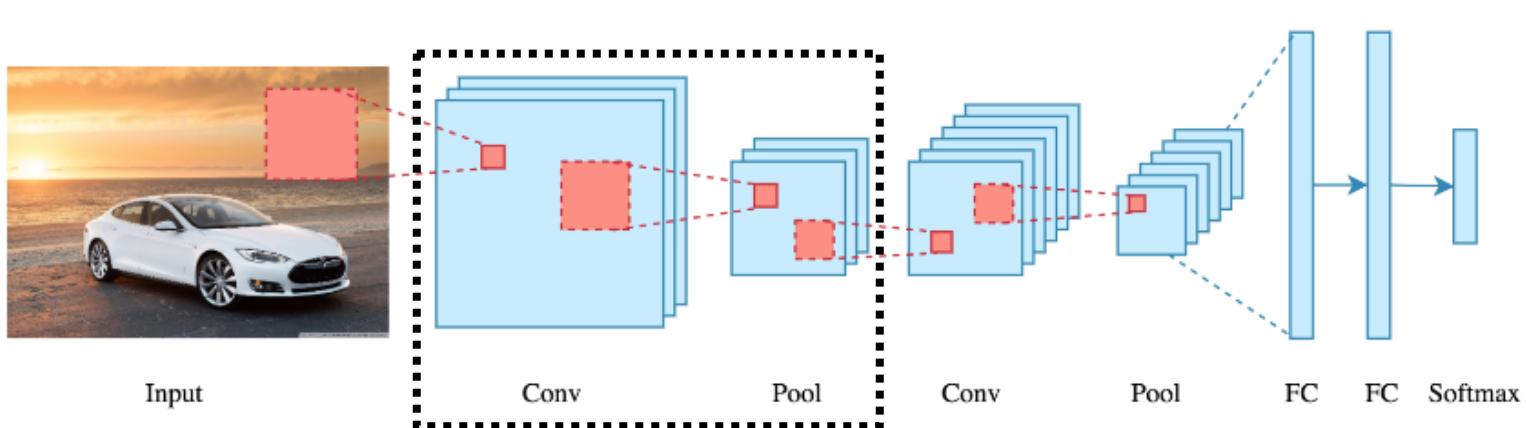
- **Arquitetura geral**
 - Poucas camadas convolucionais geralmente seguidas de *poolings*
 - mais camadas convolucionais, seguidas *poolings*, etc
 - no final há uma rede *feedforward* totalmente conectada, com 2 camadas densas (MLP)
 - camadas **intermediárias** → ativação do tipo **ReLU**
 - camada **final** → ativação do tipo **softmax**/sigmoidal (probabilidades das classes)
- Conforme o sinal é propagado a imagem fica menor e menor

Arquitetura Geral



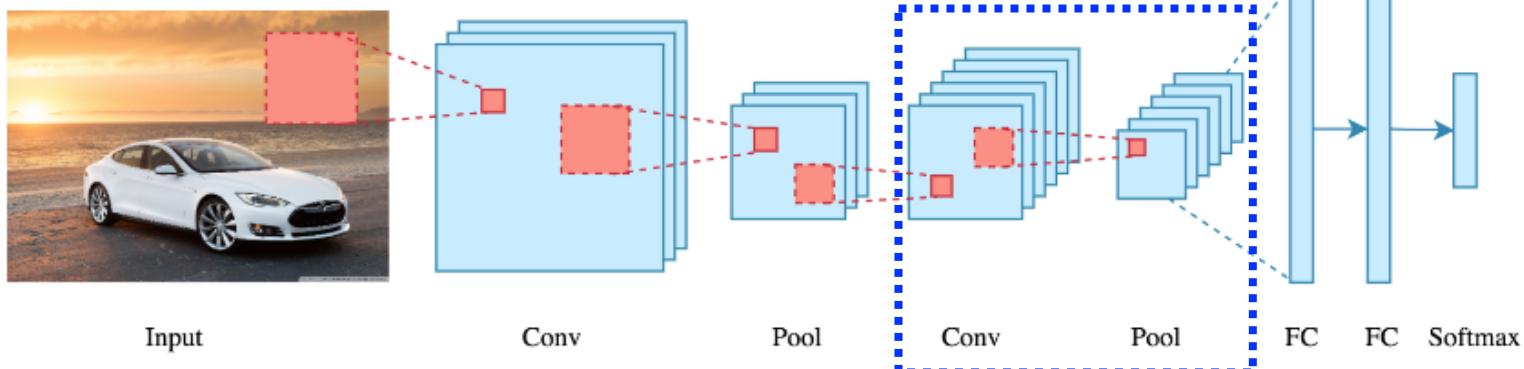
Arquitetura Geral

**camada convolucional
seguida de pooling**

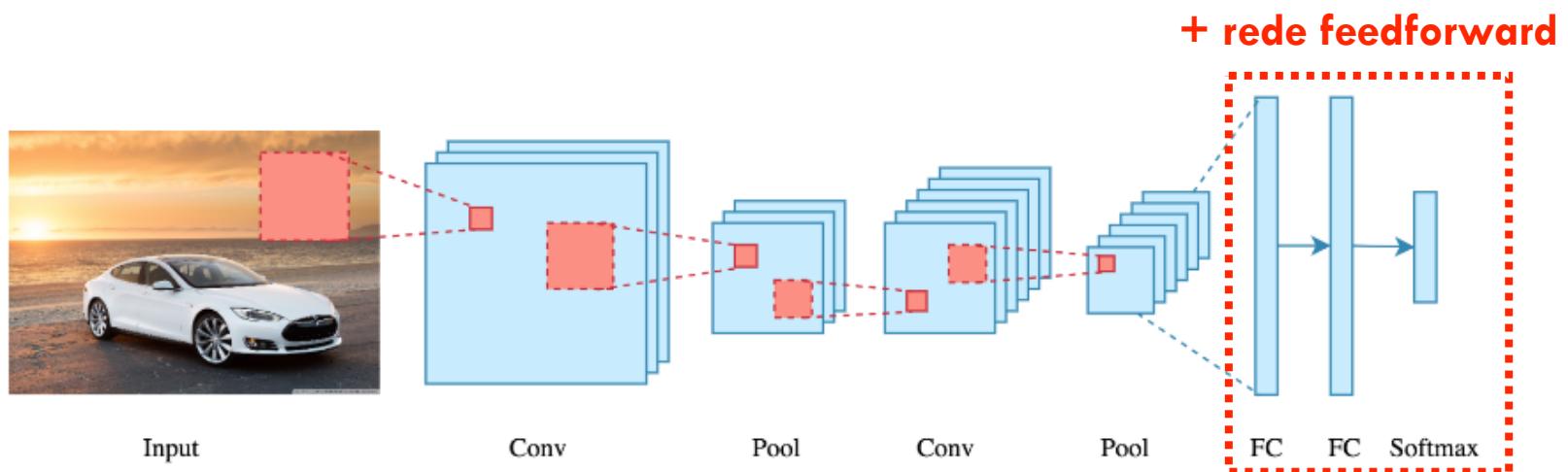


Arquitetura Geral

+ camada convolucional
seguida de pooling

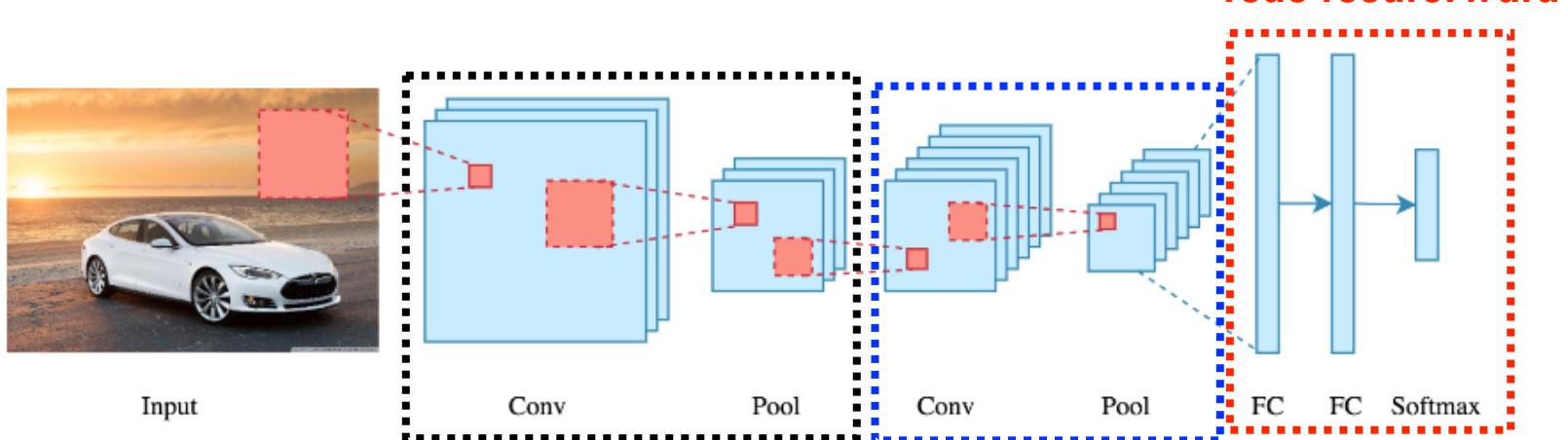


Arquitetura Geral



Arquitetura Geral

**camada convolucional
seguida de pooling + camada convolucional
seguida de pooling + rede feedforward**



Arquiteturas

- Ao longo das camadas o número de filtros aumenta
 - poucas formas básicas (usamos menos filtros)
 - mais formas de combinar as formas básicas (usamos mais filtros)
 - **Prática comum:** dobrar o número de filtros depois de *pooling*

Arquiteturas

- Ao longo das camadas o número de filtros aumenta
 - poucas formas básicas (usamos menos filtros)
 - mais formas de combinar as formas básicas (usamos mais filtros)
 - **Prática comum:** dobrar o número de filtros depois de *pooling*

- **Flatten:** achatar/condensar o sinal
 - Camadas densas necessitam de sinais 1D (vetores)
 - Convertemos imagens em (2 ou 3 dimensões) em arrays que alimentam a rede *feedforward*

LeNet-5

- Uma das arquiteturas de CNN mais conhecidas na [literatura](#)
 - criada por Yann LeCun (1988)
 - usada para classificação de dígitos escritos a mão (MNIST)

LeNet-5

- Uma das arquiteturas de CNN mais conhecidas na [literatura](#)
 - criada por Yann LeCun (1988)
 - usada para classificação de dígitos escritos a mão (MNIST)

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg Pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg Pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–

LeNet-5

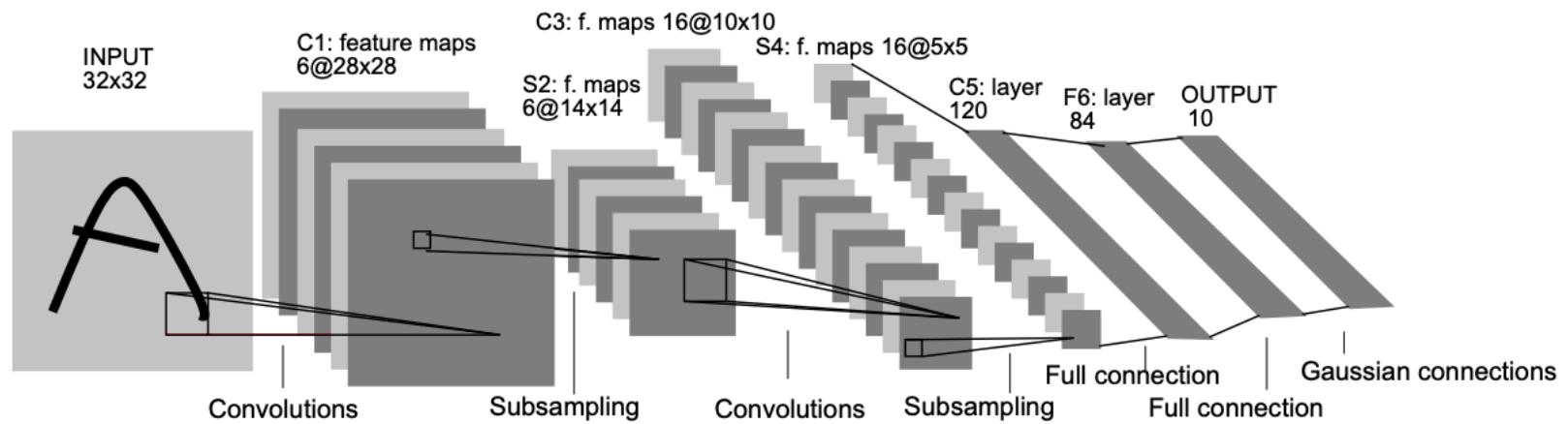


Figura de: [Y. LeCun et al \(1988\)](#)

LeNet-5

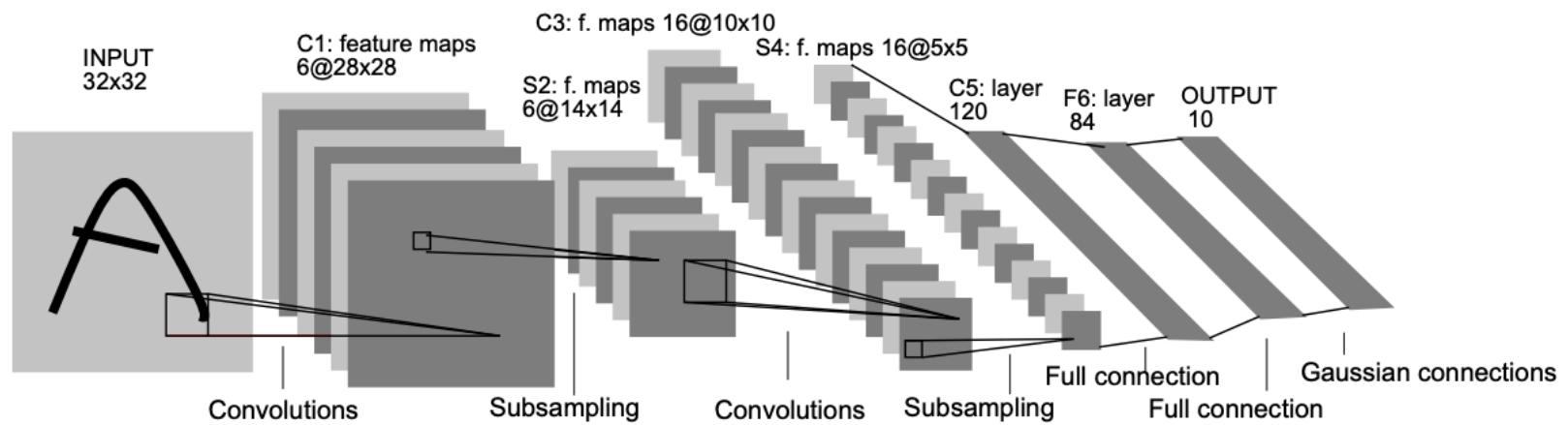


Figura de: [Y. LeCun et al \(1988\)](#)

MNIST: imagens 28x28, adicionado zero-padded criando imagens 32x32,
posteriormente normalizadas antes de alimentar a rede

Arquiteturas

- Outras arquiteturas famosas

- AlexNet
- GoogLeNet
- VGGNet
- ResNet
- Xception
- ...

Roteiro

- 1 Introdução
- 2 *Convolutional Neural Networks (CNNs)*
- 3 Modelagem de CNNs
- 4 Hands on / Exemplos
- 5 Referências

Hands on

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

Vamos exercitar :)

Hands on

CO CNN_Keras_example1_CIFAR10.ipynb ☆

File Edit View Insert Runtime Tools Help Last saved at 1:44 PM

+ Code + Text

```
# e as mostraremos em um grid 5 x 6 - 5 linhas, 6 colunas
par(mfcol = c(5,6), mar = rep(1, 4), oma = rep(0.2, 4))
cifar$train$x[index,,,] %>%
  purrr::array_tree(1) %>%
  purrr::set_names(class_names[cifar$train$y[index] + 1]) %>%
  purrr::map(as.raster, max = 255) %>%
  purrr::iwalk(~{plot(.x); title(.y)})
```

The image shows a 5x6 grid of 30 small square images, each labeled with its corresponding class name below it. The classes represented are: frog, automobile, deer, truck, deer, frog; truck, bird, horse, truck, cat, cat; truck, horse, horse, cat, frog, dog; deer, ship, bird, bird, frog, deer; automobile, cat, truck, frog, bird, airplane. The images are arranged in five rows and six columns.

Seria interessante analisar também qual é a distribuição das classes do problema. Podemos então gerar um histograma para verificar a frequência de cada uma das classes, tanto no conjunto de treinamento (`q1`) como no conjunto de teste (`q2`).

Hands on

CNN_Keras_example1_CIFAR10.ipynb

File Edit View Insert Runtime Tools Help Last saved at 1:44 PM

+ Code + Text

```
# e as mostraremos em um grid 5 x 6 - 5 linhas, 6 colunas
par(mfcol = c(5,6), mar = rep(1, 4), oma = rep(0.2, 4))
cifar$train$x[index,,,] %>%
  purrr::array_tree(1) %>%
  purrr::set_names(class_names[cifar$train$y[index] + 1]) %>%
  purrr::map(as.raster, max = 255) %>%
  purrr::iwalk(~{plot(.x); title(.y)})
```

frog automobile deer truck deer frog

cat dog deer airplane

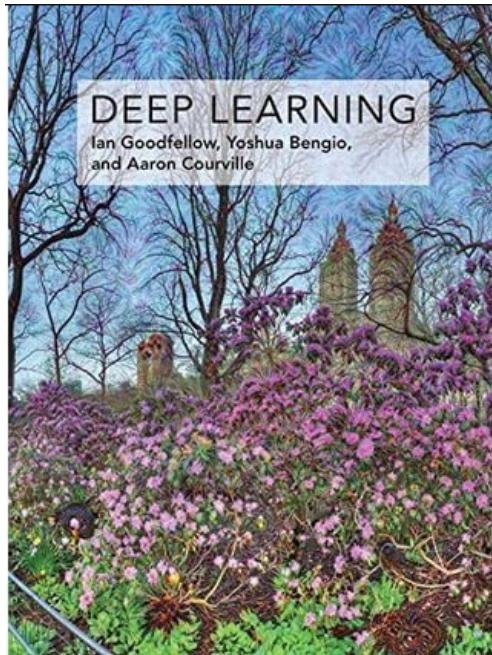
Vamos exercitar :)

Seria interessante analisar também qual é a distribuição das classes do problema. Podemos então gerar um histograma para verificar a frequência de cada uma das classes, tanto no conjunto de treinamento (`q1`) como no conjunto de teste (`q2`).

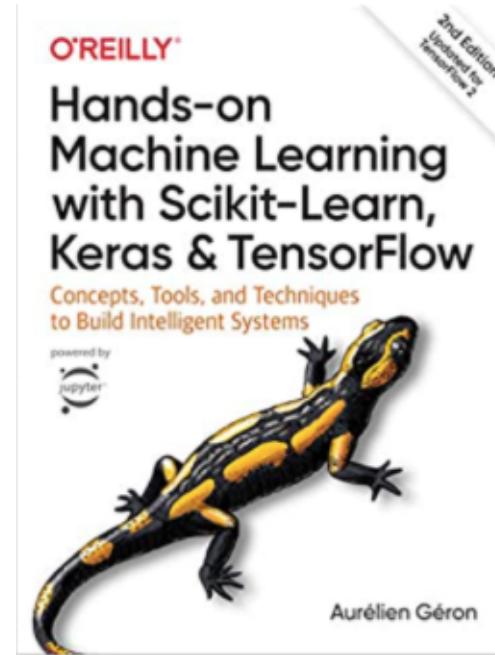
Roteiro

- 1 Introdução
- 2 *Convolutional Neural Networks (CNNs)*
- 3 Modelagem de CNNs
- 4 Hands on / Exemplos
- 5 Referências

Literatura Sugerida



(Goodfellow, Bengio, Courville; 2015)



(Géron, 2019)

Literatura Sugerida

- MIT book: <http://www.deeplearningbook.org>
- Deep Learning: <http://deeplearning.net>
- Andrew Ng: <https://www.deeplearning.ai>

Literatura Complementar

- Coursera: <https://www.coursera.org/specializations/deep-learning>
- Google AI: <https://ai.google/education/>
- Keras: <https://keras.io>
- Auto-Keras: <https://autokeras.com>
- h2o: <http://docs.h2o.ai/h2o/latest-stable/h2o-docs/index.html#>

Literatura Complementar

- Aulas de Hugo Larrochelle: <https://www.youtube.com/watch?v=vXMpKYRhpml>
- DL + MK: <https://blog.mgechev.com/2018/10/20/transfer-learning-tensorflow-js-data-augmentation-mobile-net/>
- Hide Screen: <https://github.com/Hironsan/BossSensor>

Perguntas?

Prof. Rafael G. Mantovani

rgmantovani@gmail.com