



FIAP

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

DISRUPTIVE ARCHITECTURES: IOT, IOB & IA

08 – Redes Neurais Convulacionais – CNN



Prof. Airton Y. C. Toyofuku



profairton.toyofuku@fiap.com.br

O que temos para hoje?

FIAP



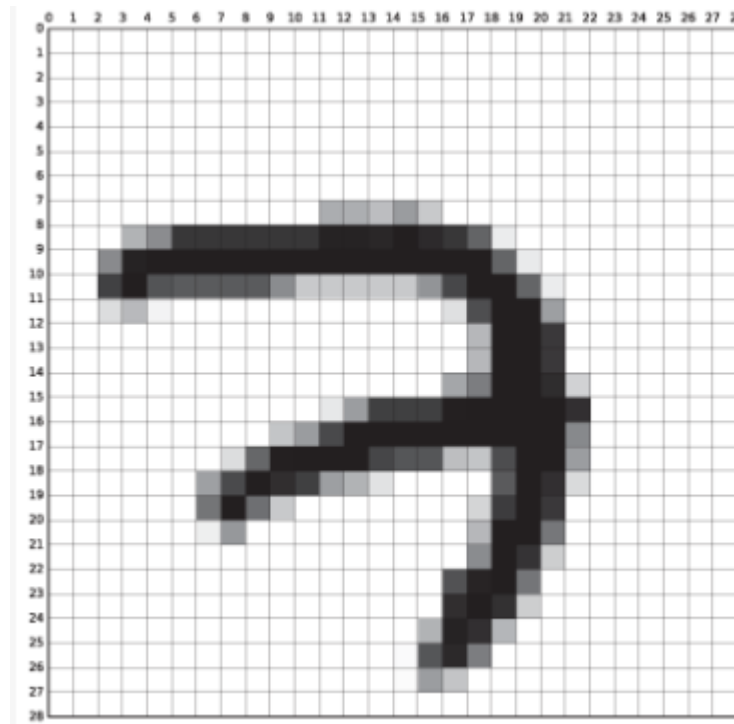
E como trabalhar com Imagens?



MNIST Dataset
Fonte: Wikipedia

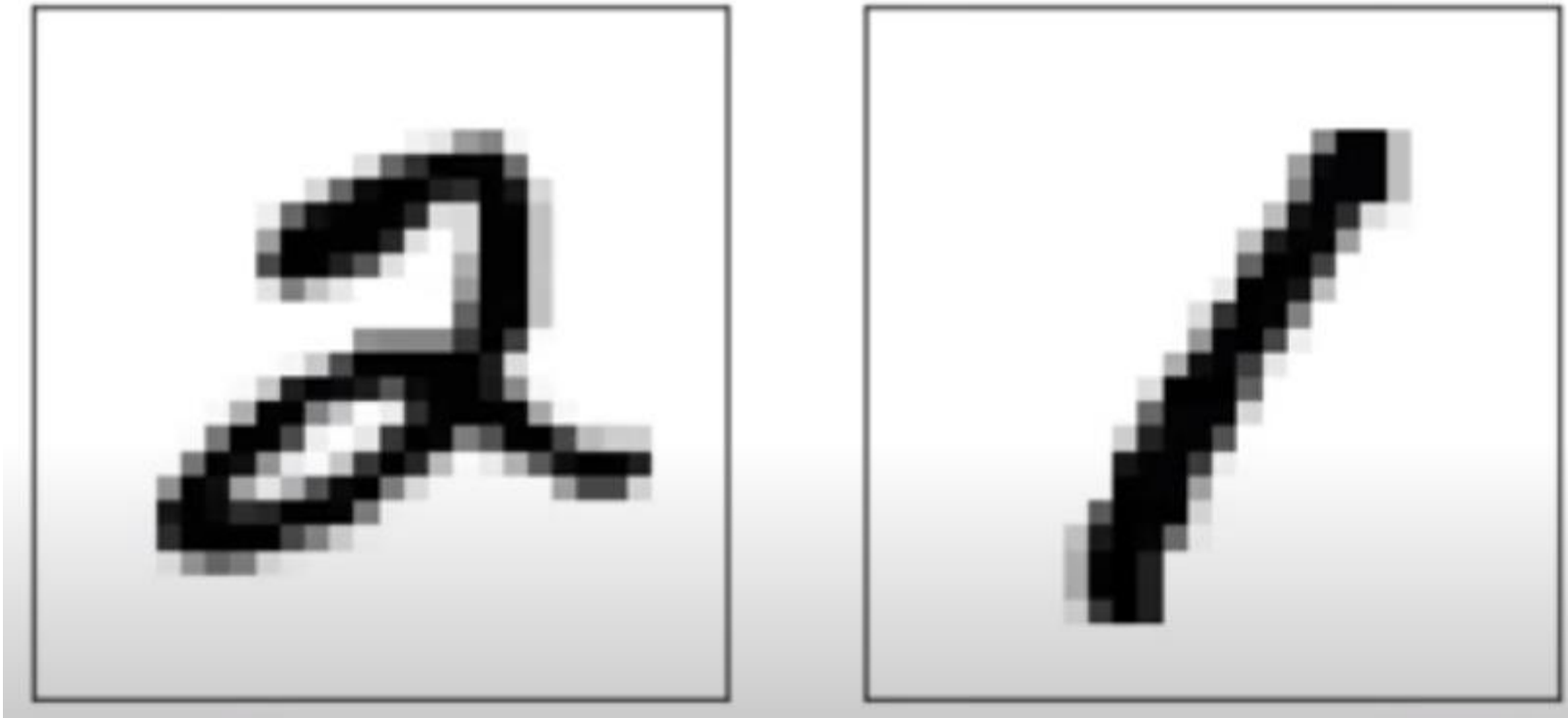
Características do MNIST

- ❖ Possui 60.000 dados para treino;
- ❖ Possui 10.000 dados para teste;
- ❖ Foi escrito a mão por funcionários do United States Census Bureau e por alunos do High School;
- ❖ Os dígitos são centralizados num Quadro de 28 x28 pixels;
- ❖ Estão todos em escala de cinza;



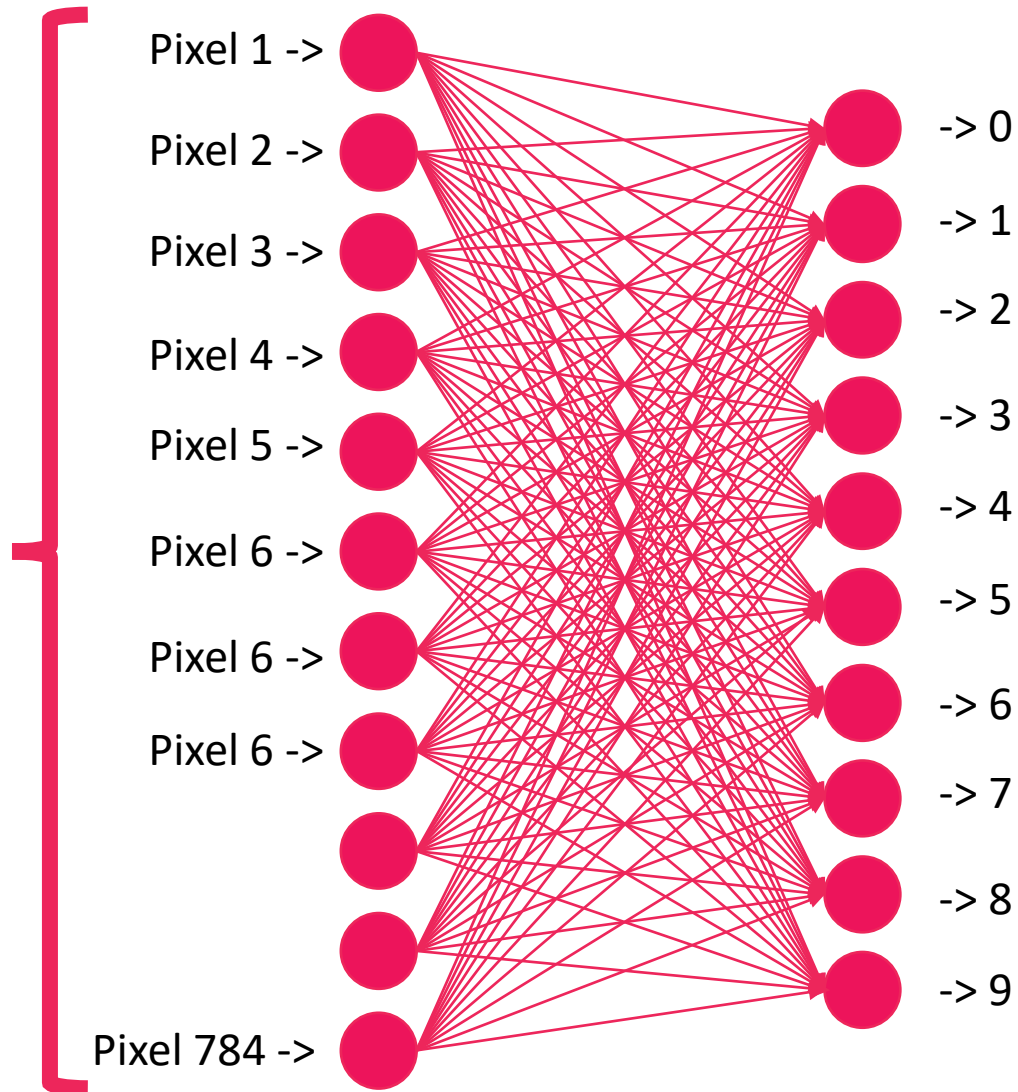
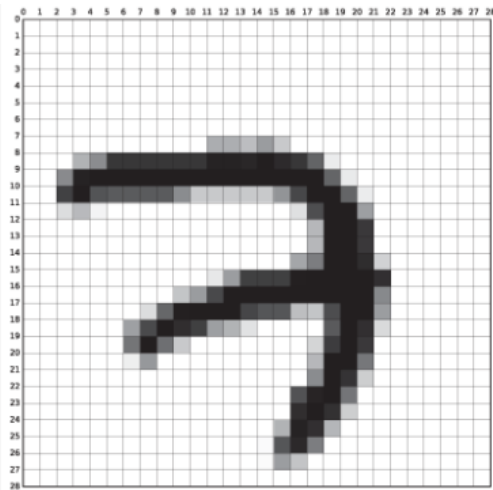
Como diferenciar os números?

FIA/P



E como aplicar numa RNA?

FIA/P



Vamos para Prática!

```
# Importar as bibliotecas
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
```

```
# Carrega o dataset
mnist = keras.datasets.mnist
# Carrega os dados de treino e teste
(train_images, train_labels), (test_images, tes
t_labels) = mnist.load_data()
```

```
# Mostra as características do dataset
print(train_images.shape)
print(test_images.shape)
print(np.unique(train_labels))
```

```
# Exibe uma amostra do dataset
plt.figure()
plt.imshow(train_images[1], cmap = 'binary')
plt.colorbar()
print(train_labels[1])
```

```
# Normalizando a escala de cinza para ficar ent
re 0 e 1
train_images = train_images/255
test_images = test_images/255
plt.figure()
plt.imshow(train_images[1], cmap = 'binary')
plt.colorbar()
print(train_labels[1])
```


Vamos para Prática!

```
# Mostrando o dataset normalizado
```

```
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.imshow(train_images[i], cmap = 'binary')
    plt.xlabel(train_labels[i])
```

```
# Criando o modelo com keras
```

```
model = keras.Sequential([
    # Camada de entrada Flatten para transformar a matriz de 28x28 para um vetor de 784
    keras.layers.Flatten(input_shape=(28,28)),
    # Camada Oculta Densa (totalmente conectada) de 128 neuronios, com função de ativação Relu
    keras.layers.Dense(128, activation = tf.nn.relu),
    # Camada de saída Densa de 10 neuronios, com função de ativação softmax
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
# Compila o modelo com função Loss e métrica de acurácia
model.compile(loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
# Mostra o modelo compilado
```

```
model.summary()
```

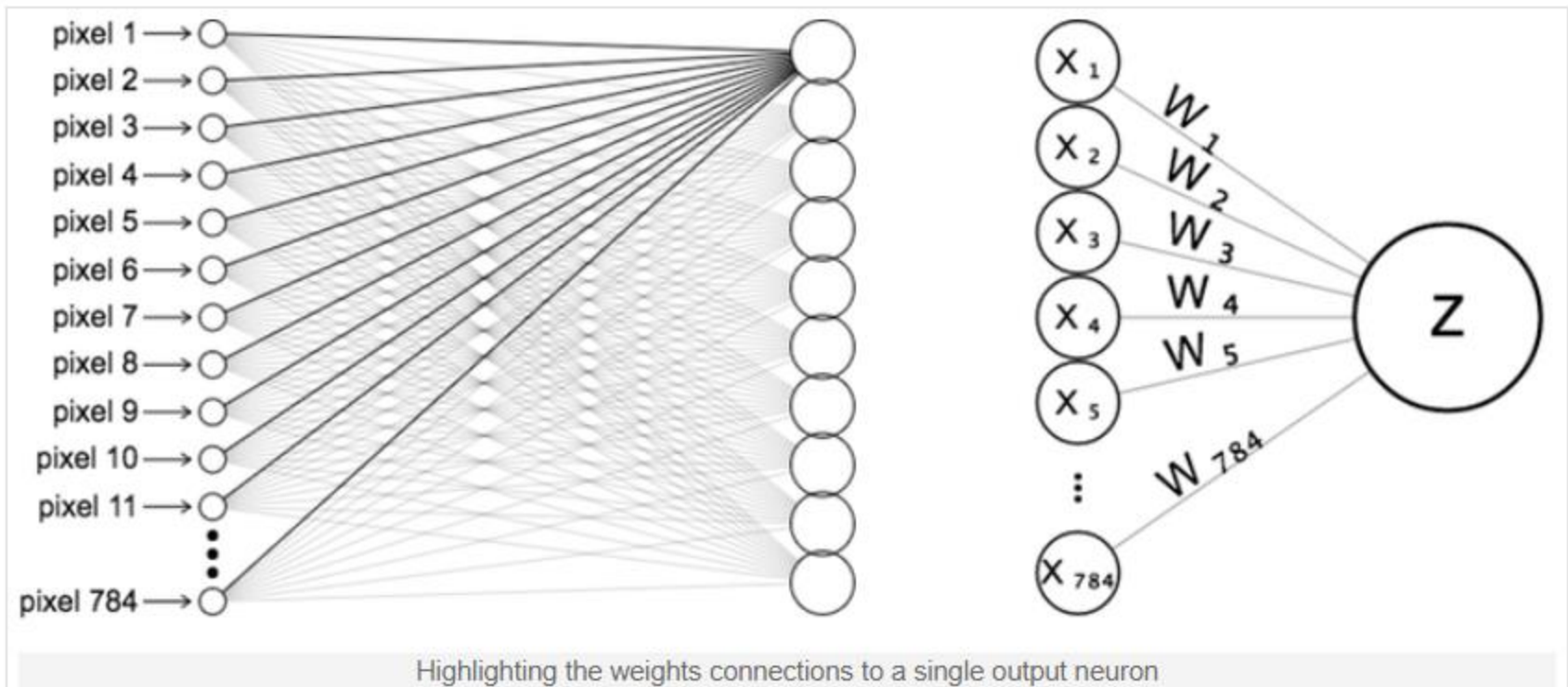
```
# Treina o modelo, com 5 épocas
```

```
model.fit(train_images, train_labels, epochs = 5)
```

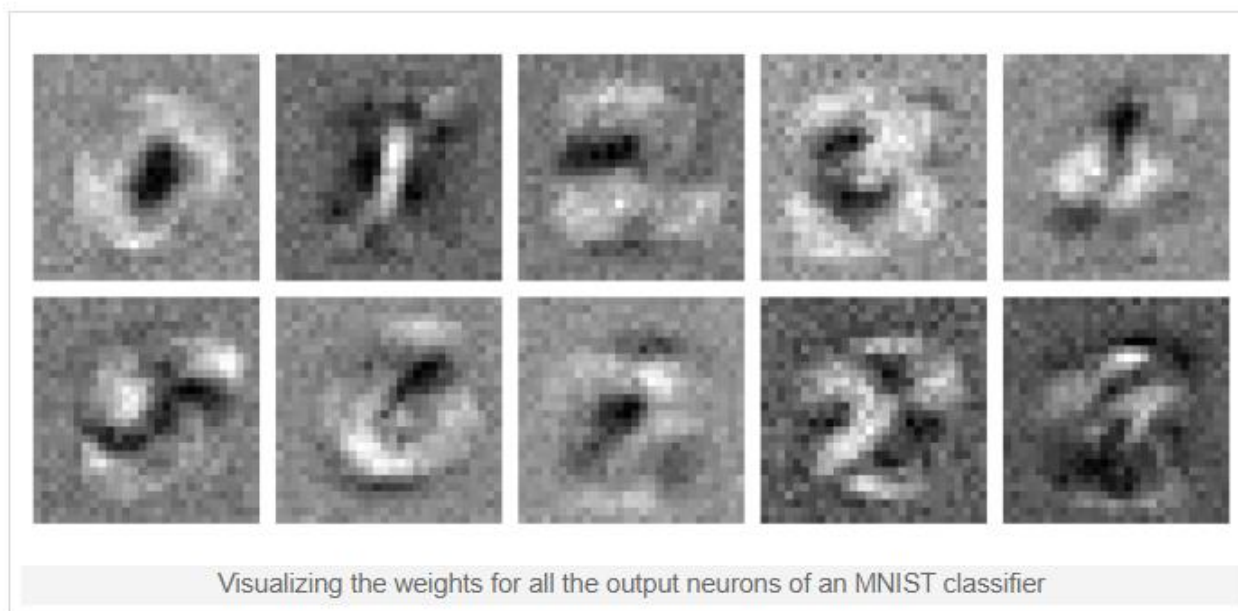
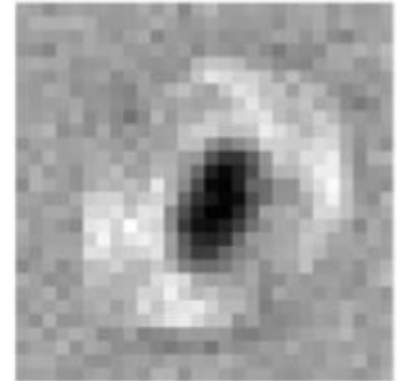
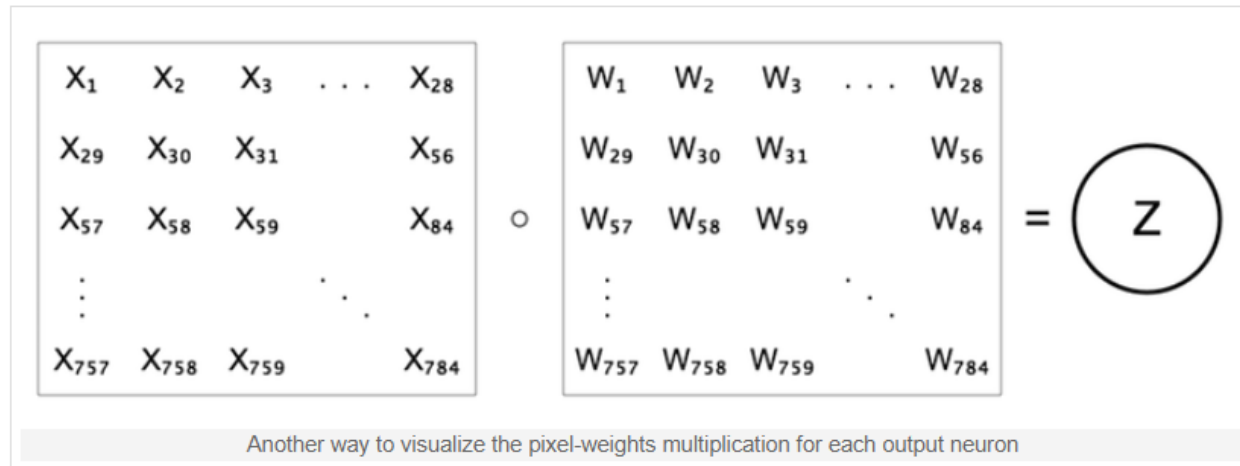
```
# Testa e mostra as métricas
```

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print(test_loss)
print(test_acc)
```

E o que aconteceu?



E o que aconteceu?



E se colocarmos cor nisso?

airplane



automobile



bird



cat



deer



dog



frog



horse



ship



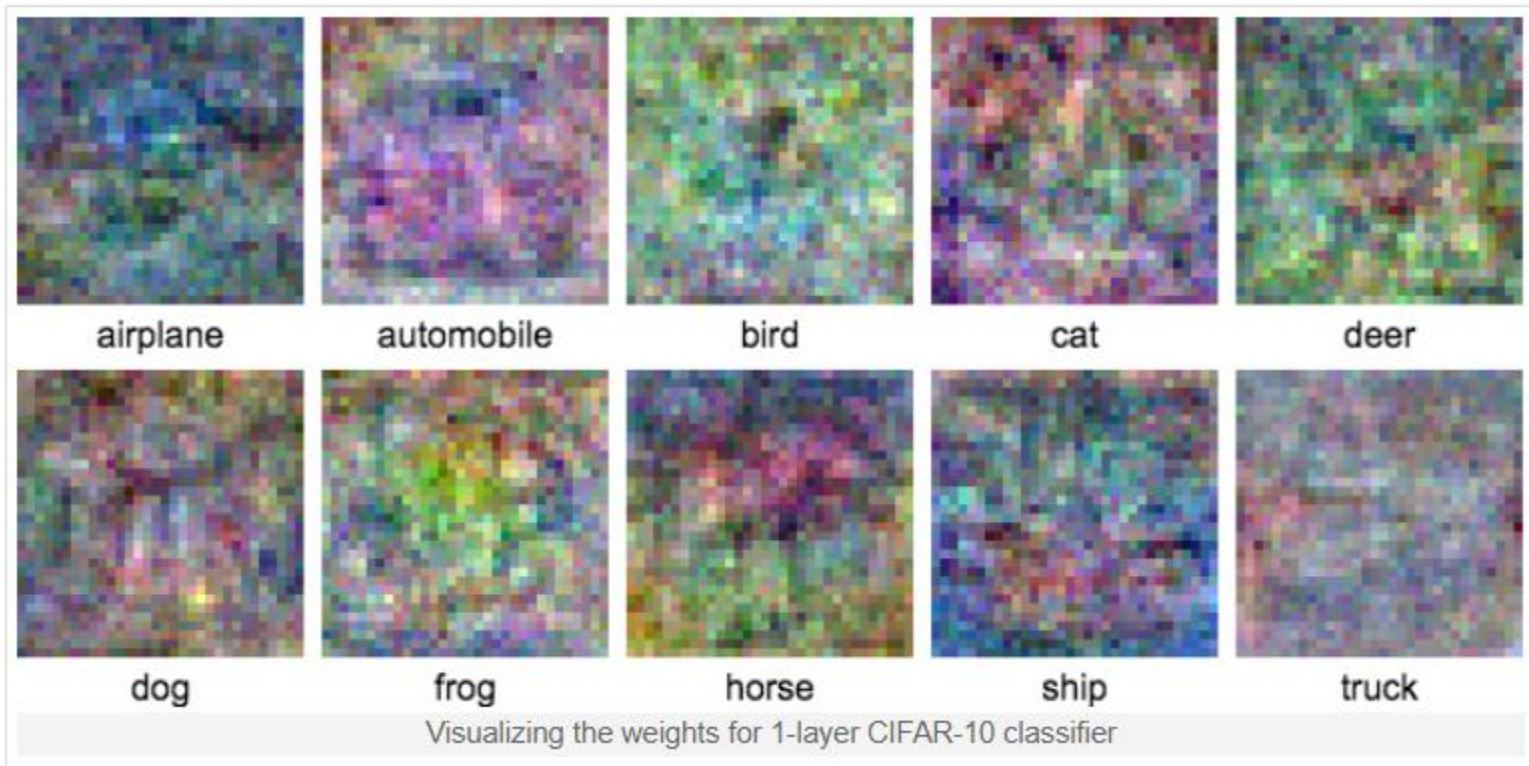
truck



❖ EXEMPLO: CIFAR-10

- 6.000 Imagens;
- 32 x 32 pixels;
- 10 classes independentes
- Classes balanceadas

Acontece isso...



Por que?

- ❖ Fazer com que cada pixel da imagem seja um **atributo** ignora a dependência entre pixels vizinhos!
- ❖ As características de contorno, profundidade e textura são percebidas por um **conjunto de pixels próximos** um do outro.
- ❖ A inclusão de cores força a pensarmos em 3 Dimensões – RGB!
- ❖ Existem duas alternativas para lidar com isso:
 1. Extrair as características de contorno, profundidade e textura através de técnicas de processamento de sinais;
 2. Aprender a extrair essas características através de parâmetros;

CIFAR-10 - Object Recognition in Images

Identify the subject of 60,000 labeled images



Kaggle · 231 teams · 8 years ago

[Overview](#) [Data](#) [Code](#) [Discussion](#) [Leaderboard](#) [Rules](#)

[Join Competition](#)



Leaderboard

[Raw Data](#)

[Refresh](#)

[Public](#) [Private](#)

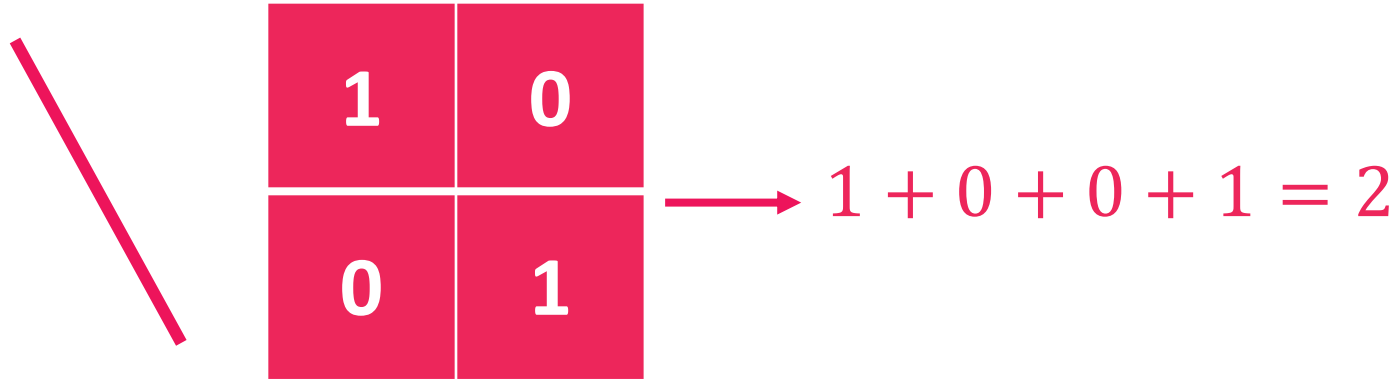
The private leaderboard is calculated with approximately 97% of the test data. This competition has completed. This leaderboard reflects the final standings.

#	△	Team	Members	Score	Entries	Last	Solution
1	—	DeepCNet		0.95530	18	8y	
2	—	jiki		0.94740	42	8y	
3	—	Anil Thomas		0.94300	3	8y	
4	—	Frank Sharp		0.94190	13	8y	
5	—	nagadomi		0.94150	16	8y	
6	—	Phil & Triskelion & Kazanova		0.94120	107	8y	

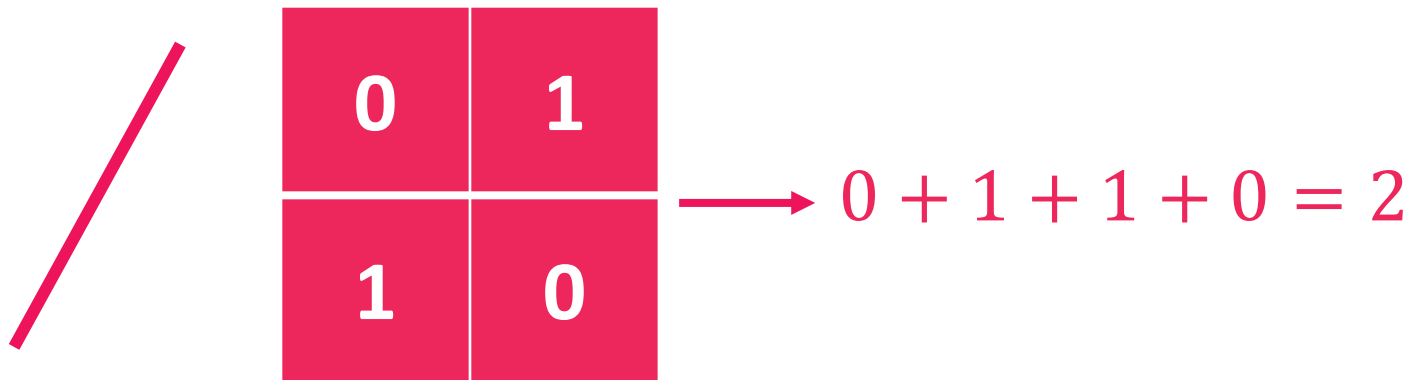
Redes Neurais Convolucionais (CNN) FIA/P

- ❖ As CNN (Convolutional Neural Networks) são usada para **reconhecimento visual**;
- ❖ Baseiam-se nos conceitos conhecidos sobre a **visão humana**;
- ❖ **Considera a relação entre os pixels vizinhos** para determinar contornos, texturas e profundidades;
- ❖ A relação é estipulada através de **filtros**, que são aplicados através de um processo chamado **convolução**.


E o que são filtros?



O que acham desse filtro?




E o que são filtros?



1	0
0	1

$$\begin{matrix} + & - & - & + \\ \rightarrow & 1 & + & 0 & + & 0 & + & 1 & = & 2 \end{matrix}$$

O que acham desse filtro?

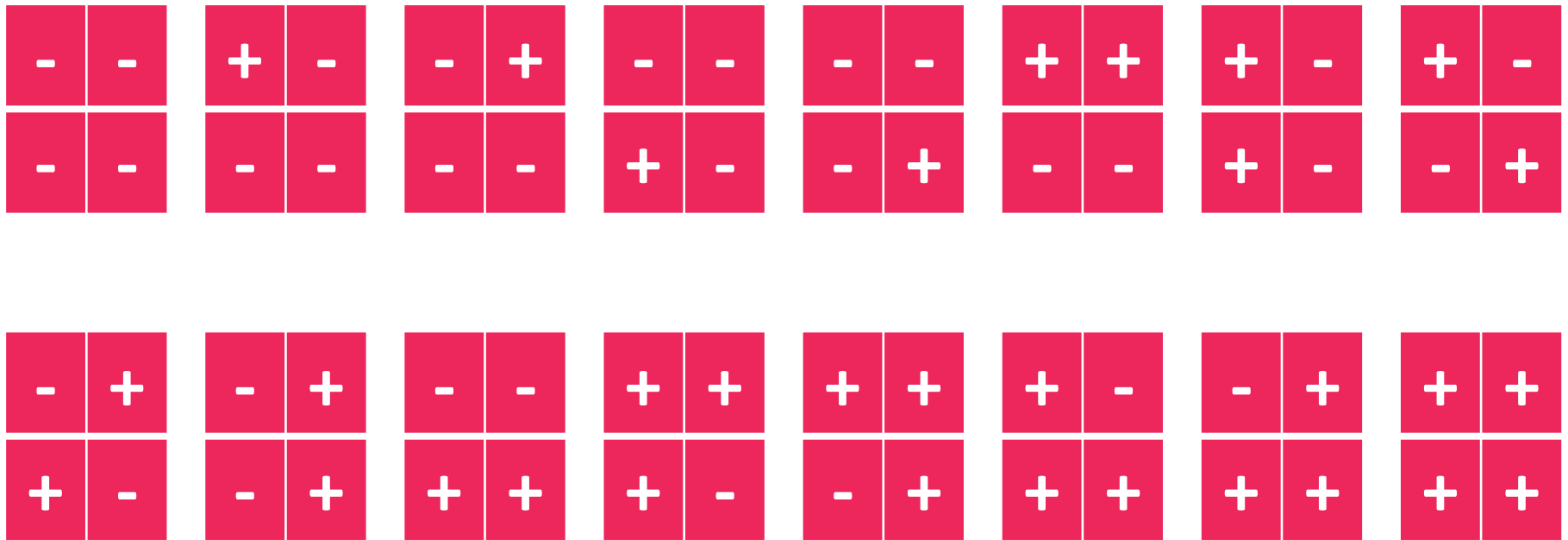


0	1
1	0

$$\begin{matrix} + & - & - & + \\ \rightarrow & 0 & + & 1 & + & 1 & + & 0 & = & -2 \end{matrix}$$

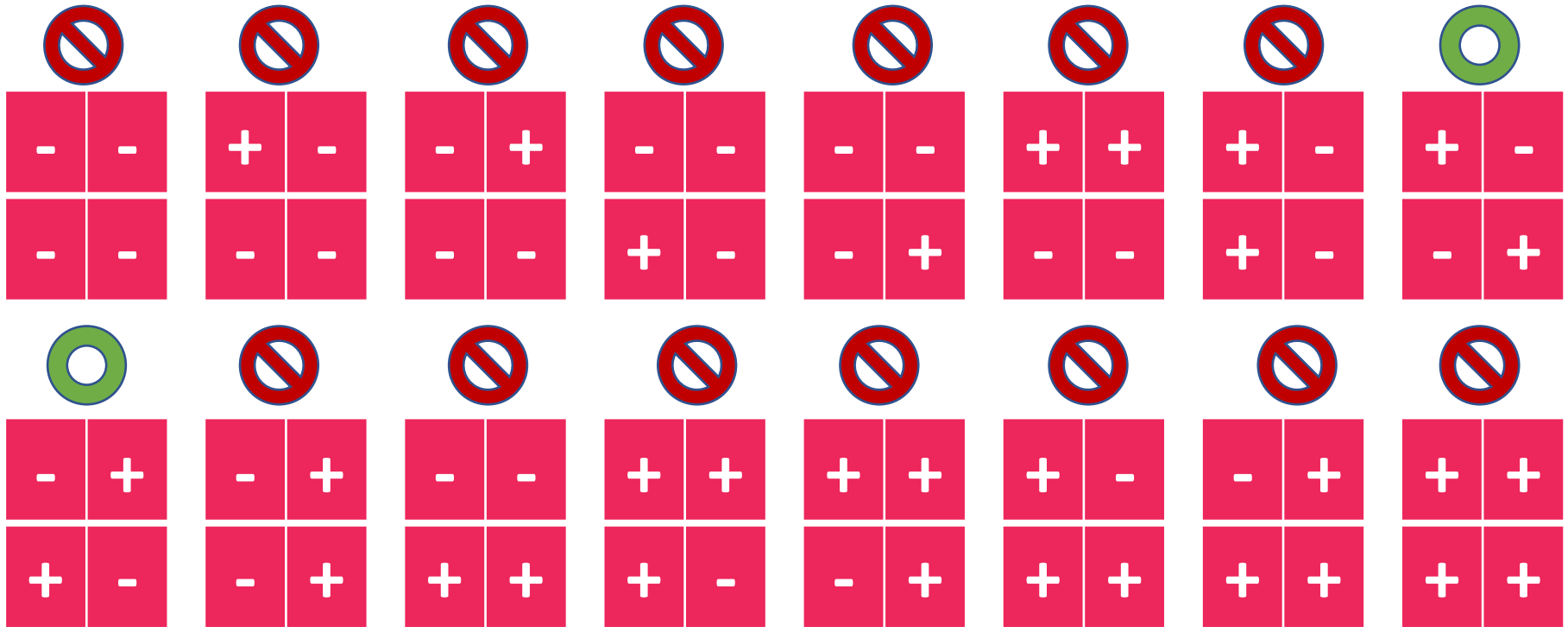
E o que são filtros?

❖ Para uma image 2x2, apenas com “-” e “+” = $2^4 = 16$ possibilidades!



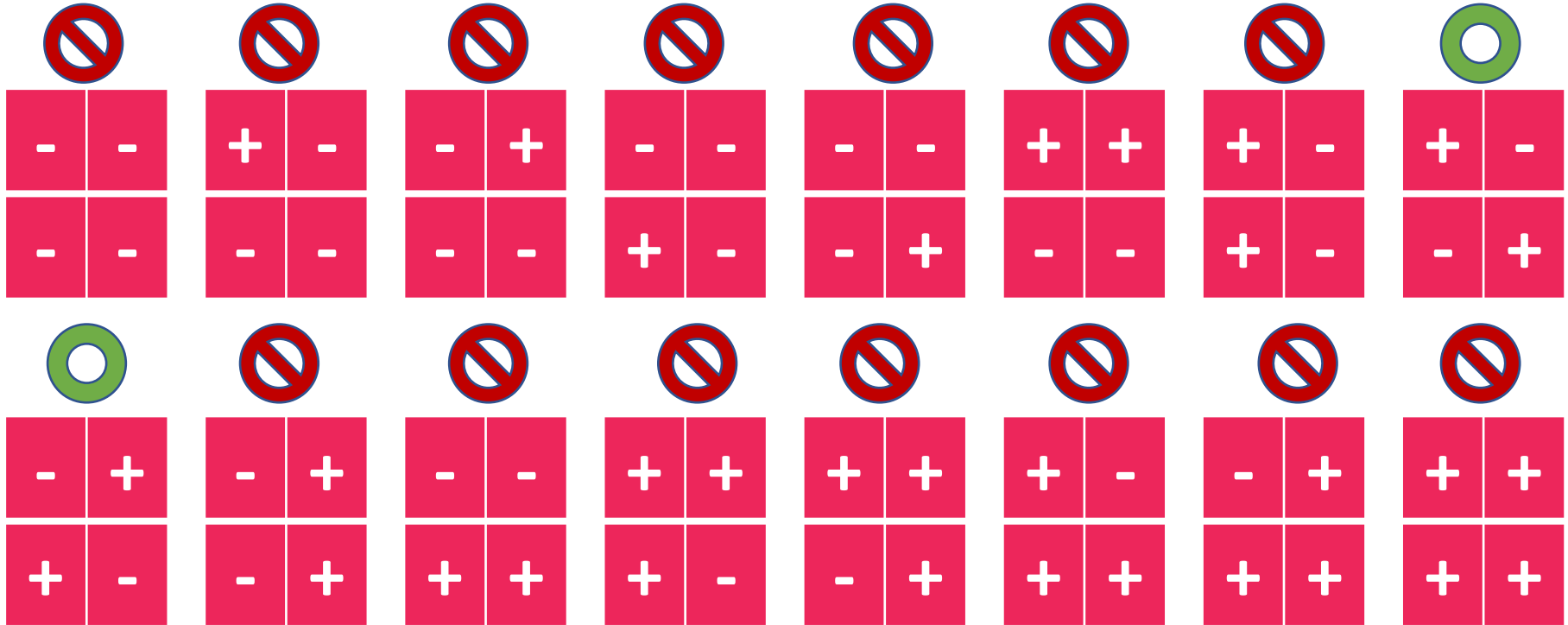
E o que são filtros?

❖ Para uma image 2x2, apenas com “-” e “+” = $2^4 = 16$ possibilidades!



E o que são filtros?

❖ Para uma image 2x2, apenas com “-” e “+” = $2^4 = 16$ possibilidades!



❖ E se aumentar a quantidade de pixels?

❖ E se usarmos valores reais randomicos, por exemplo 3.14, 2.75, 0.33...?

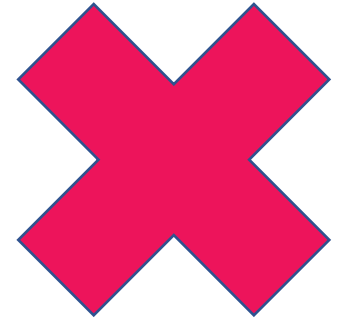
Voltando para CNN

FILAP



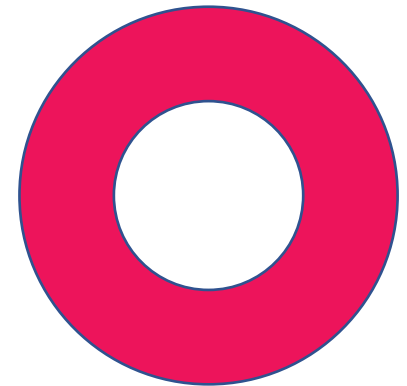
1	-1	-1
-1	1	-1
-1	-1	1

1	-1	1
-1	1	-1
1	-1	1

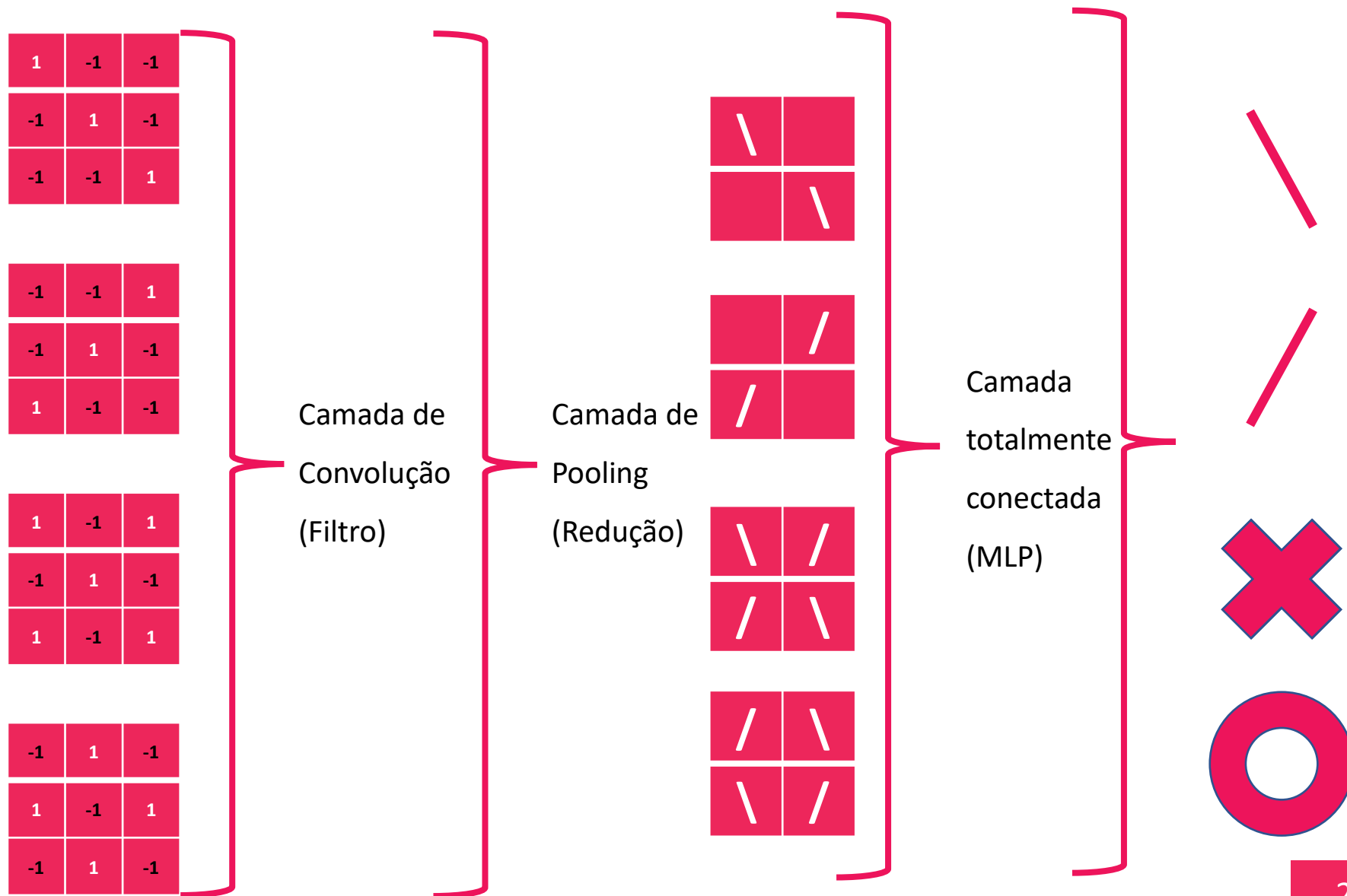


-1	-1	1
-1	1	-1
1	-1	-1

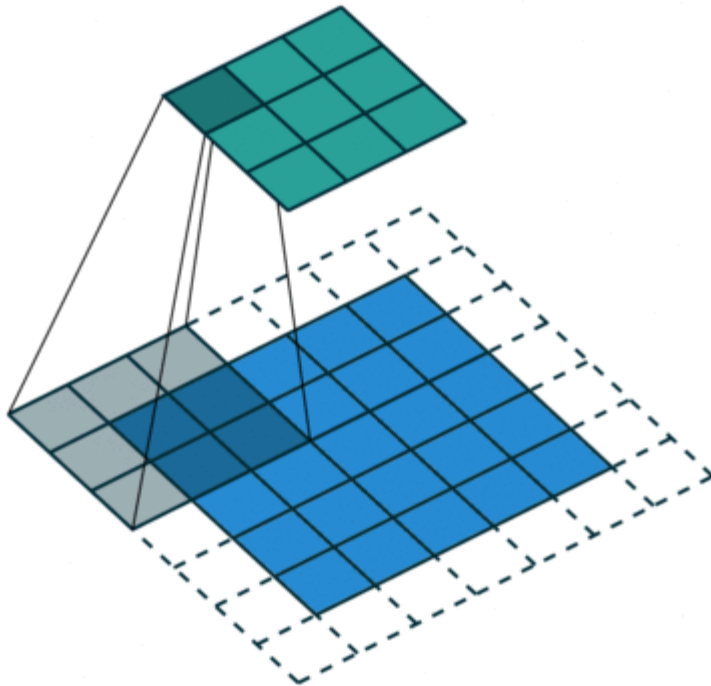
-1	1	-1
1	-1	1
-1	1	-1



Voltando para CNN



Camada de Convolução



Fonte: <https://giphy.com/gifs/blog-daniel-keypoints-i4NjAwytglRDW>

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

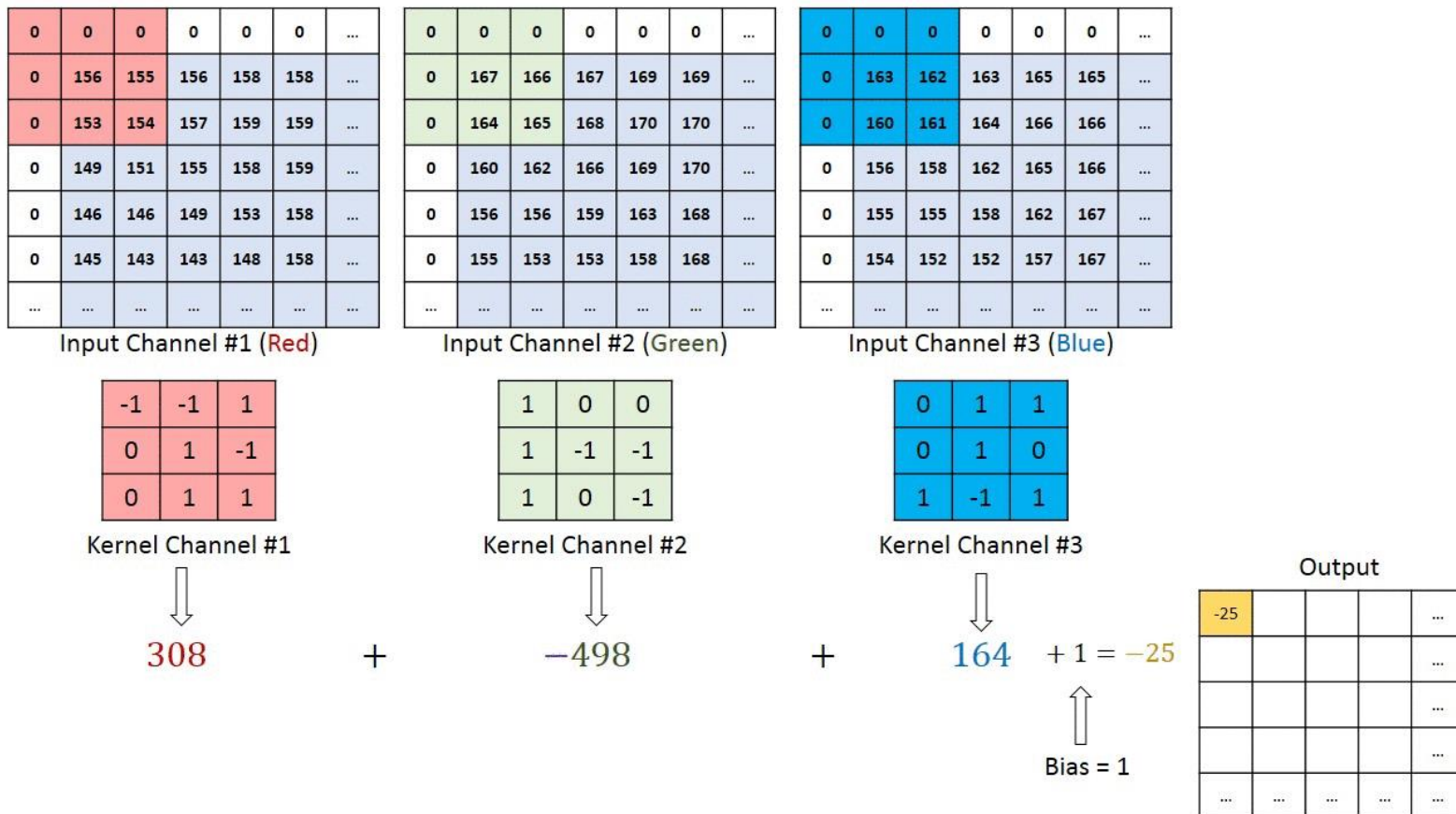
Image

4		

Convolved
Feature








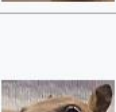
Fonte: https://commons.wikimedia.org/wiki/File:Convolution_arithmetic_-_Padding_strides.gif

Camada de Convolução em imagens $FILP$ coloridas



Fonte: https://miro.medium.com/v2/resize:fit:1280/1*ciDgQEjViWLnCbmX-EeSrA.gif

Mas para que serve isso?

Operation	Kernel ω	Image result $g(x,y)$
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Ridge or edge detection	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3×3 (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	
Gaussian blur 5×5 (approximation)	$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	
Unsharp masking 5×5 Based on Gaussian blur with amount as 1 and threshold as 0 (with no image mask)	$\frac{-1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & -476 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	

- ❖ Serve para achar bordas nas imagens;
- ❖ Serve para evidenciar alguma característica como o sharpen e o blur;

E o que isso tem a ver com CNN?

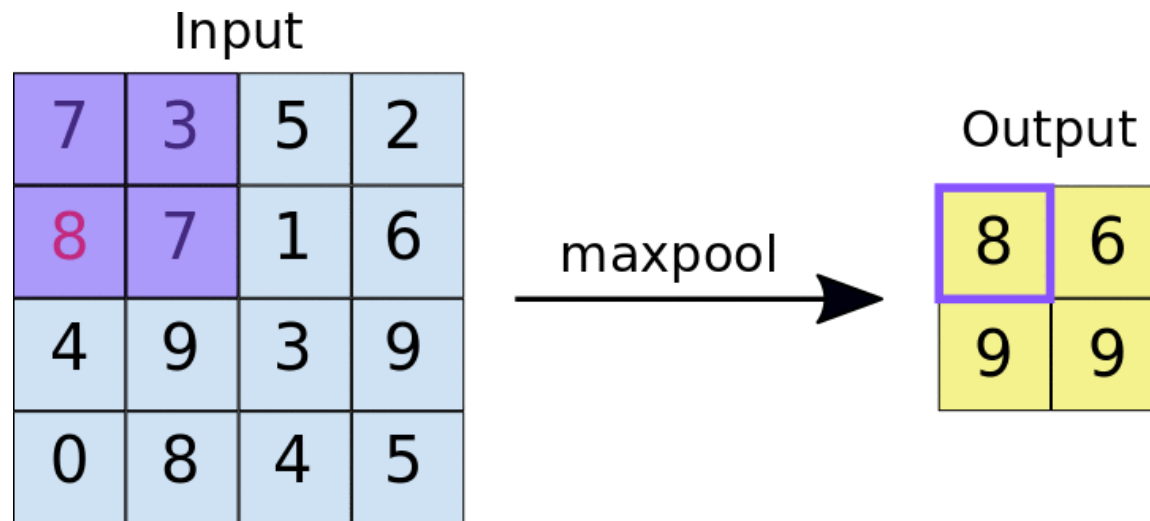
- A CNN aplica esses filtros de convolução várias vezes;
- O treinamento da CNN consiste em descobrir quais os valores que mais se adequam aos filtros;

Camada de Pooling

- ❖ Usada para reduzir a dimensão da imagem e conseguir discriminar a imagem.

Exemplos:

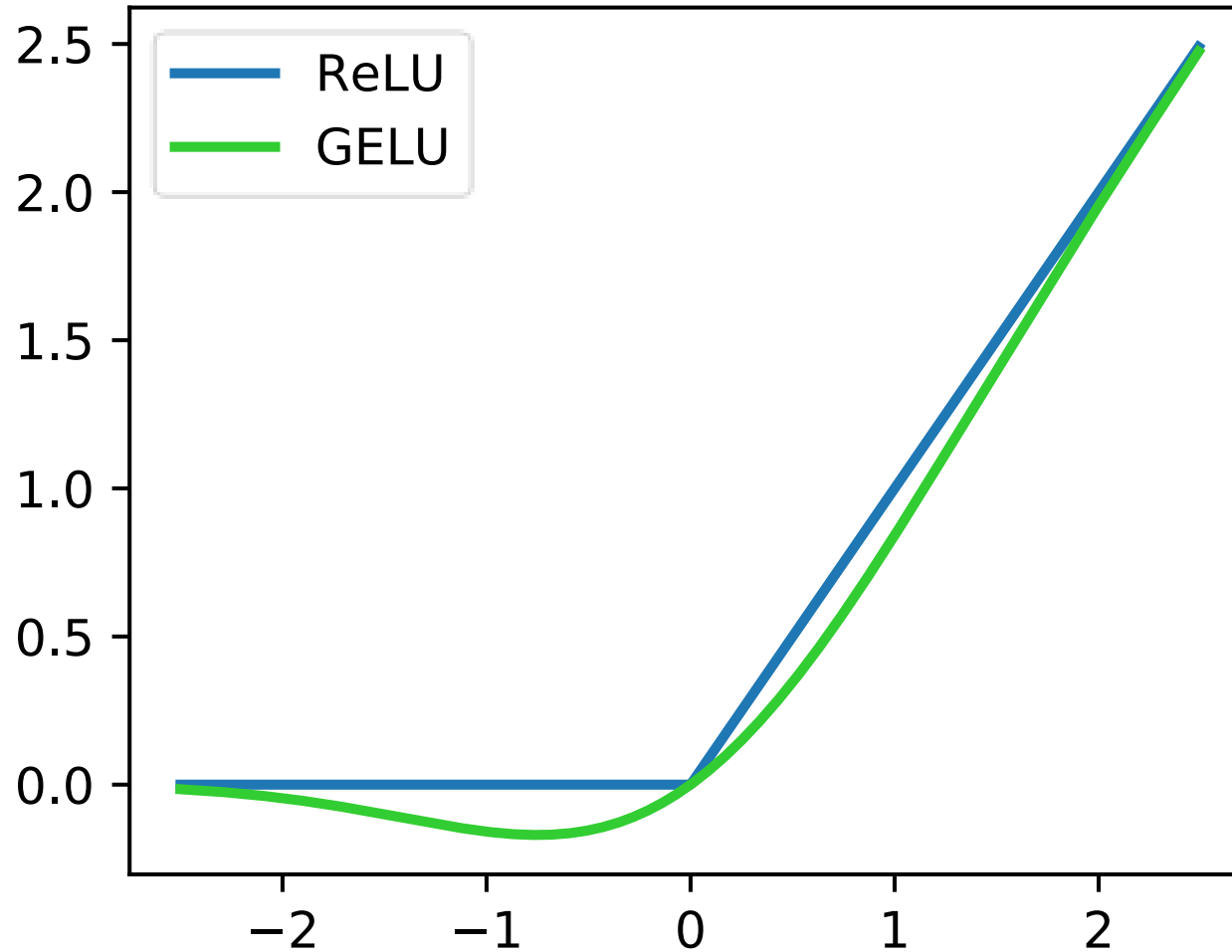
- ❖ Max Pooling



Fonte: <https://nico-curti.github.io/NumPyNet/NumPyNet/images/maxpool.gif>

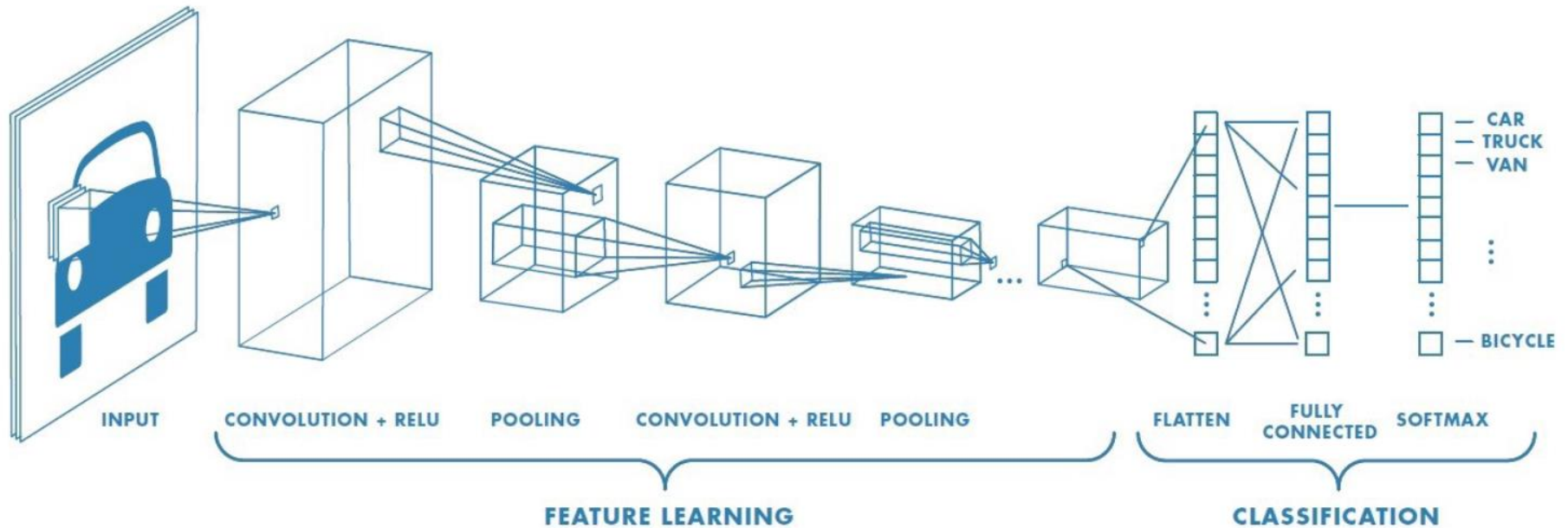
- ❖ Outros exemplos: Min Pooling e Average Pooling

Nonlinearities



Fonte: https://upload.wikimedia.org/wikipedia/commons/4/42/ReLU_and_GELU.svg

Processo Completo da CNN



Fonte: <https://rstudio-conf-2020.github.io/dl-keras-tf/04-computer-vision-cnns.html#13>

Vamos para Prática!

```
# Criando um modelo de rede neural sequencial.
model2 = keras.Sequential()

# Adiciona uma camada de convolução 2D com 32 filtros de tamanho 5x5, que usa a função de ativação ReLU
# e tem uma entrada de imagem 2D com altura e largura de 28 pixels e um único canal de cor (escala de cinza).
model2.add( keras.layers.Conv2D(filters=32, kernel_size=5, padding='same',
                                activation='relu', input_shape=(28,28,1)) )

# Adiciona uma camada de max pooling 2D que reduz a dimensão da imagem de entrada pela metade
# (em ambas as direções - altura e largura) usando uma janela de 2x2 pixels e um passo de 2 pixels.
# A camada é adicionada à rede neural após a camada de convolução, o que ajuda a reduzir o número
# de parâmetros e a extrair as características mais importantes da imagem.
model2.add( keras.layers.MaxPooling2D(pool_size=[2,2], strides=2))

# Adiciona uma camada de convolução 2D com 64 filtros de tamanho 5x5, que usa a função de ativação ReLU
# e tem uma entrada de imagem 2D com altura e largura de 28 pixels e um único canal de cor (escala de cinza).
model2.add( keras.layers.Conv2D(filters=64, kernel_size=5, padding='same',
                                activation='relu', input_shape=(28,28,1)) )

# Adiciona uma camada de max pooling 2D que reduz a dimensão da imagem de entrada pela metade
# (em ambas as direções - altura e largura) usando uma janela de 2x2 pixels e um passo de 2 pixels.
# A camada é adicionada à rede neural após a camada de convolução, o que ajuda a reduzir o número
# de parâmetros e a extrair as características mais importantes da imagem
model2.add(keras.layers.MaxPooling2D(pool_size=[2,2], strides=2))

# Transforma a imagem no formato de "matriz" em um vetor
model2.add(keras.layers.Flatten())

# Adiciona uma camada de 1024 neurônios totalmente conectados (Densa) com função de ativação ReLU
model2.add(keras.layers.Dense(1024, activation='relu'))

# Adiciona uma camada de 10 neurônios totalmente conectados (Densa), que calcula
# as probabilidades de cada classe usando a função softmax.
model2.add(keras.layers.Dense(10, activation='softmax'))

# Compila o modelo model2 com o otimizador Adam, a função de perda de entropia cruzada categórica esparsa
# e a métrica de acurácia. O modelo agora está pronto para ser treinado.
model2.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Vamos mostrar como ficou a rede
model2.summary()
```

Vamos para Prática!

```
# Treinando nosso modelo!
model2.fit(train_images, train_labels, epochs = 5)
```

```
# Vamos expandir as dimensões da imagem para verificação de acerto depois
# de colocar o modelo a prova
test_images = (np.expand_dims(test_images,3))
test_images.shape
```

```
# Vamos testar o modelo!
test_loss, test_acc = model.evaluate(test_images, test_labels)

print(test_loss)
print(test_acc)
```

```
# Agora vamos por o modelo a prova! Vamos passar imagens e ele vai nos dizer o que é
predictions = model.predict(test_images)
print("Score para o objeto #0 no teste: ")
print(predictions[0])
print("Classe predita = " +str(np.argmax(predictions[0])))
```

Algumas verificações:

```
# Funções para facilitar a visualização
def plot_image(i, predictions_array, true_label, img):
    predictions_array, true_label, img = predictions_array[i], true_label[i], img[i]
    plt.grid(False)
    plt.xticks([])
    plt.yticks([])

    plt.imshow(img, cmap = plt.cm.binary)

    predicted_label = np.argmax(predictions_array)
    if(predicted_label == true_label):
        color = 'blue'
    else:
        color = 'red'

    plt.xlabel("{} {:2.0f}% ({})".format(predicted_label,
                                         100*np.max(predictions_array),
                                         true_label,
                                         color = color))

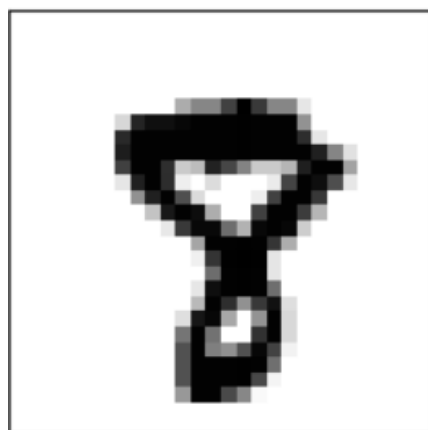
def plot_value_array(i, predictions_array, true_label):
    predictions_array, true_label = predictions_array[i], true_label[i]
    plt.grid(False)
    plt.xticks(range(10), range(10))
    plt.yticks([])
    thisplot = plt.bar(range(10), predictions_array, color = "#77777777")
    plt.ylim([0,1])
    predicted_label = np.argmax(predictions_array)

    thisplot[predicted_label].set_color('red')
    thisplot[true_label].set_color('blue')
```

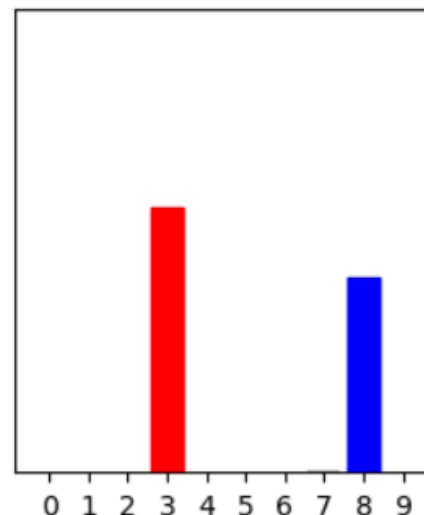

Algumas verificações:

```
# Vamos verificar quais as imagens que o nosso modelo errou
predictions = model.predict(test_images)
for i in range(500):
    if(np.argmax(predictions[i]) != test_labels[i]):
        print(i)
```

```
# Agora vamos avaliar os numeros que ele errou
i = 233 # Substitua por um dos indices apontados na celula anterior!
plt.figure(figsize=(6,3))
plt.subplot(1,2,1)
plot_image(i, predictions, test_labels, test_images[:, :, :, 0])
plt.subplot(1,2,2)
plot_value_array(i, predictions, test_labels)
```



3 57% (8)



Copyright © 2023 Prof. Airton Y. C. Toyofuku

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).