



Processamento de Imagens - OCR e Aplicações

Novembro, 2022

Sobre o OCR

- O que é exatamente o OCR?

Todo e qualquer algoritmo onde temos o intuito de extrair o conteúdo escrito dentro de imagens em texto.

Exemplos de OCR

an input object image into a sequence usually essential. For example, Graves set of geometrical or image features for while Su and Lu [33] convert word images into HOG features. The preprocessing is the subsequent components in the pipeline systems based on RNN can not be trained



Exemplos de OCR

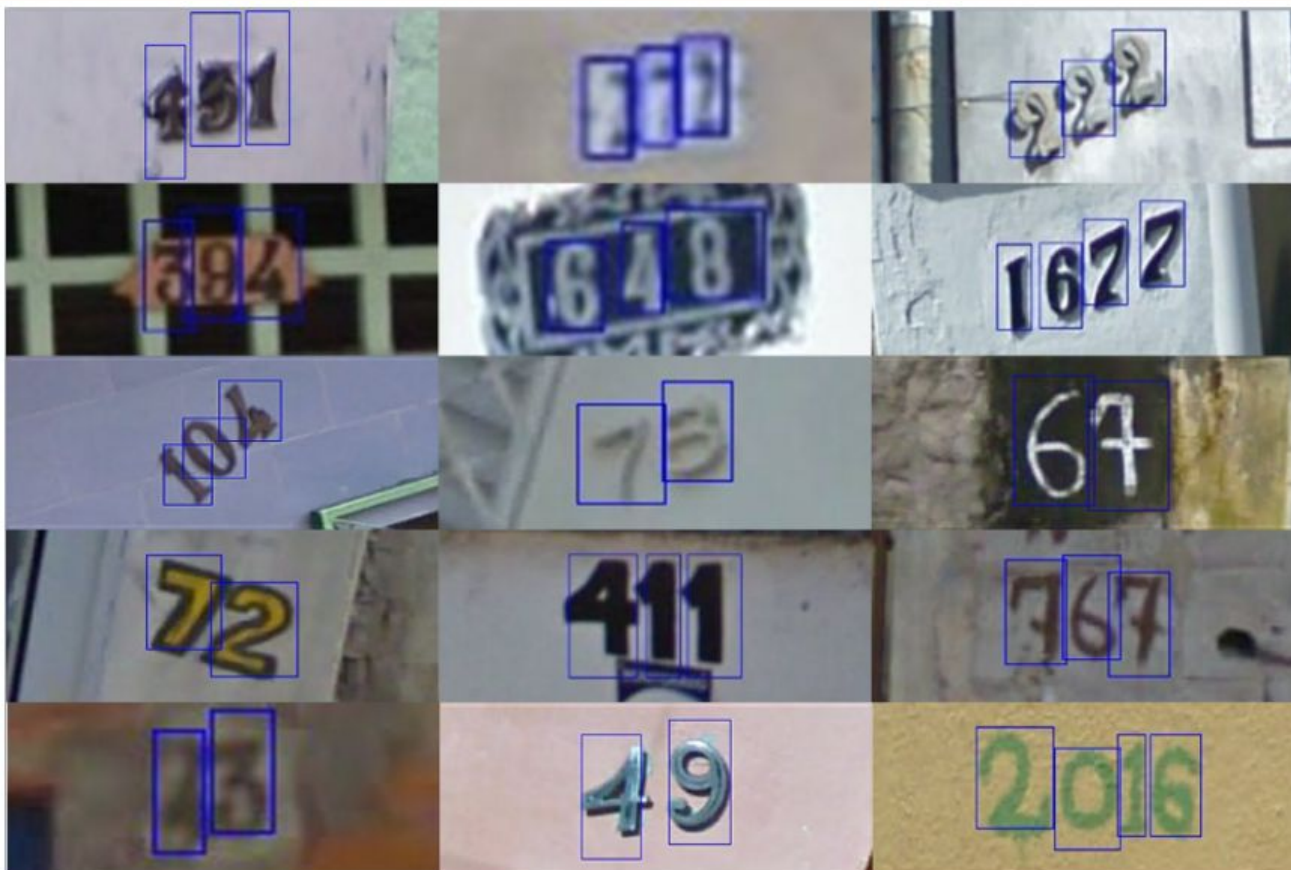


Figura. Conjunto de imagens do dataset SVHN.

OCR combinado com bancos de dados



Figura. Identificação de modelos de carros usando OCR e bancos de dados externos.

Abordagens mais usadas em OCR

- **Detecção por Regiões (Faster R-CNN, R-FCN)**

Achamos as partes da imagem que contém o conteúdo de interesse para o OCR, e depois usamos um classificador para o conteúdo.

- **Single Shot Detectors (SSD, YOLO)**

Tanto as regiões (boundary box) e a classe do conteúdo são detectados ao mesmo tempo, em um único processo.

Vantagens e Desvantagens

- **Detecção por Regiões (Faster R-CNN, R-FCN)**

Modelos mais precisos, que demandam maior tempo de processamento e portanto não são indicados para inferência real-time.

- **Single Shot Detectors (SSD, YOLO)**

Modelos menos precisos e mais leves, podendo portanto gerar inferências em menos tempo (até mesmo real-time response, dependendo da aplicação específica).

Vantagens e Desvantagens

- **Detecção por Regiões (Faster R-CNN, R-FCN)**

Modelos mais precisos, que demandam maior tempo de processamento e portanto não são indicados para inferência real-time.

- **Single Shot Detectors (SSD, YOLO)**

Modelos menos precisos e mais leves, podendo portanto gerar inferências em menos tempo (até mesmo real-time response, dependendo da aplicação específica).

Muitas dessas abordagens usam CNNs!

Principais técnicas de ML em OCR

Como funciona a 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	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image

0	0	1
1	0	0
0	1	1

Feature Detector

Como funciona a 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	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image



0	0	1
1	0	0
0	1	1

Feature Detector



0				

Feature Map

Como funciona a 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	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image



0	0	1
1	0	0
0	1	1

Feature Detector



0	1			

Feature Map

Como funciona a 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	0	0
0	1	0	0	0	1	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0

Input Image



0	0	1
1	0	0
0	1	1

Feature Detector



0	1	0	0	0
0	1	1	1	0
1	0	1	2	1
1	4	2	1	0
0	0	1	2	1

Feature Map

Como funciona a Convolução

```
data = [[0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0]]  
detector = [[[0]], [[1]], [[0]]],  
            [[[0]], [[1]], [[0]]],  
            [[[0]], [[1]], [[0]]]]  
weights = [asarray(detector), asarray([0.0])]
```

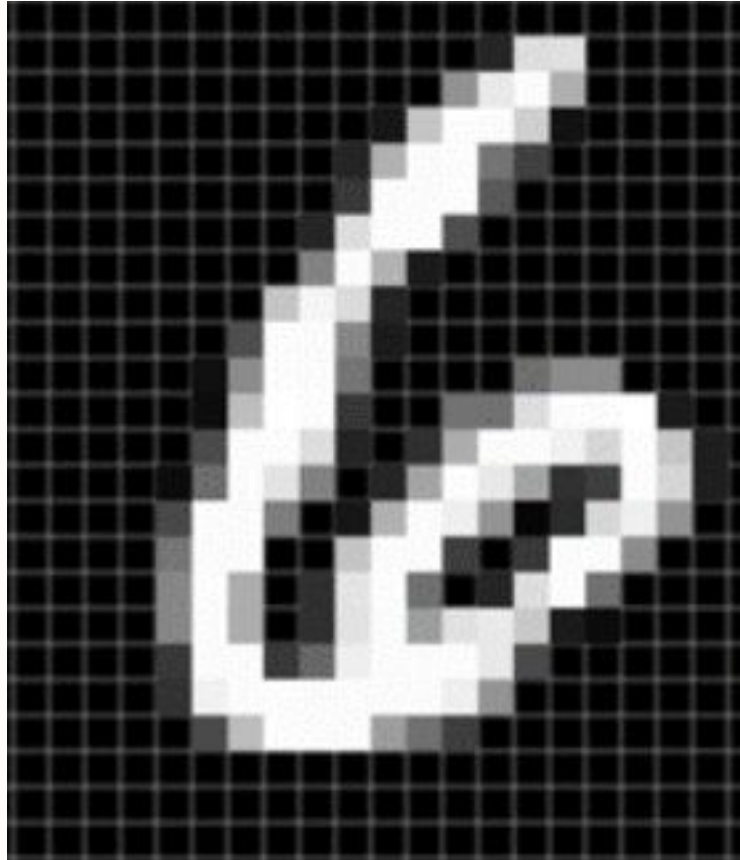
Como funciona a Convolução

```
data = [[0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0],  
        [0, 0, 0, 1, 1, 0, 0, 0]]  
detector = [[[[0]], [[1]], [[0]]],  
            [[[[0]], [[1]], [[0]]],  
            [[[[0]], [[1]], [[0]]]]  
weights = [asarray(detector), asarray([0.0])]
```



```
[0.0, 0.0, 3.0, 3.0, 0.0, 0.0]  
[0.0, 0.0, 3.0, 3.0, 0.0, 0.0]  
[0.0, 0.0, 3.0, 3.0, 0.0, 0.0]  
[0.0, 0.0, 3.0, 3.0, 0.0, 0.0]  
[0.0, 0.0, 3.0, 3.0, 0.0, 0.0]  
[0.0, 0.0, 3.0, 3.0, 0.0, 0.0]
```

Como funciona a Convolução



Como funciona a Convolução



*

1	0	-1
2	0	-2
1	0	-1



Técnicas de Downsampling - Pooling

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

max pool with 2x2
window and stride 2

6	8
3	4

Resumo das principais operações em CNN

- **Convolução**

Gerar Feature Maps e reconhecer padrões específicos de grupos de pixels (geralmente com padding e com stride = 1)

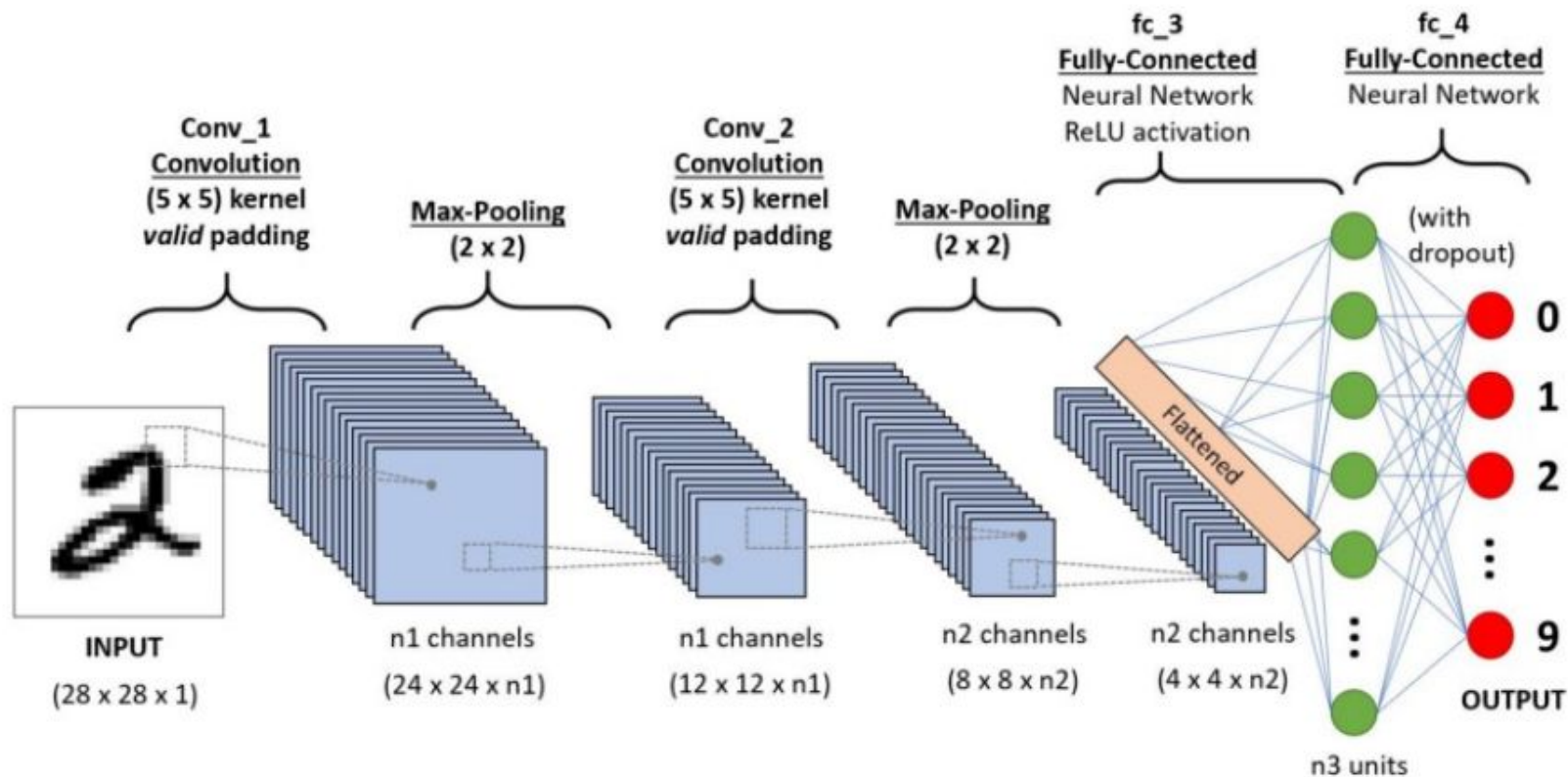
- **Downsampling e Pooling**

Redução de dimensionalidade dos Feature Maps e preservação de informação (geralmente sem padding e com stride > 1)

- **Padding**

Basicamente, significa colocar zeros ao redor dos inputs, para manter a dimensão do input e output de uma operação de convolução

Arquiteturas típicas de CNNs



Datasets mais utilizados

- VOC

O acrônimo de **Visual Object Classes** é um conjunto de dados (imagens) de diferentes categorias.

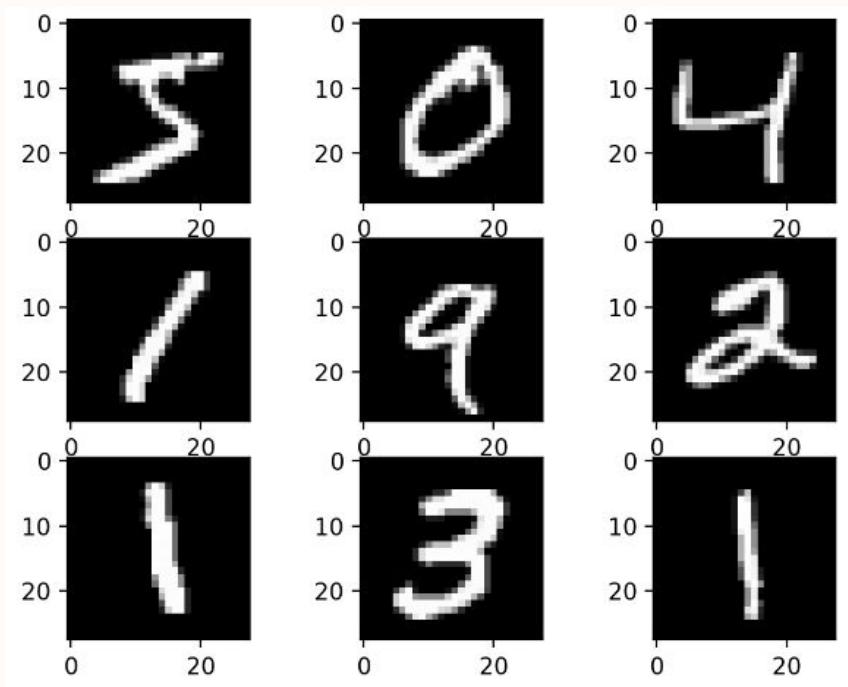
- MNIST Database

Banco de dados de dígitos manuscritos para treinos e testes de modelos de Machine Learning

- SVHN (Street View House Number)

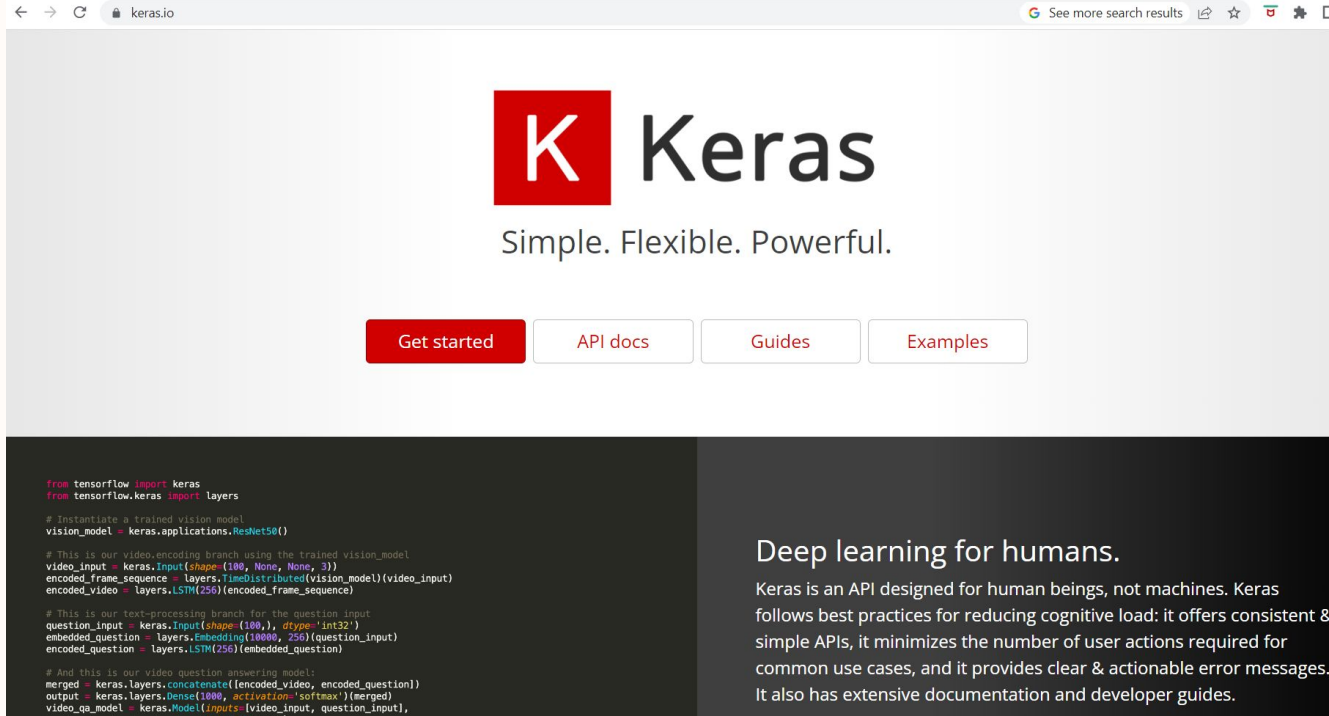
Em termos do tipo de imagens, é similar ao MNIST, mas com imagens mais “realistas”

Comparativo do MNIST e SVHN



Começando a trabalhar com OCR em Python

O framework Keras



The screenshot shows the Keras website homepage in a web browser. The browser's address bar displays 'keras.io'. The page features the Keras logo, which consists of a red square with a white 'K' and the word 'Keras' in a large, black, sans-serif font. Below the logo, the tagline 'Simple. Flexible. Powerful.' is centered. Underneath the tagline, there are four buttons: 'Get started' (highlighted in red), 'API docs', 'Guides', and 'Examples'. The bottom section of the page has a dark background and is split into two parts. The left part contains Python code for a video question-answering model, and the right part contains the text 'Deep learning for humans.' followed by a paragraph describing Keras as an API designed for human beings.

← → ↻ 🔒 keras.io 🔍 See more search results 📄 ☆ 🚩

K Keras

Simple. Flexible. Powerful.

[Get started](#) [API docs](#) [Guides](#) [Examples](#)

```
from tensorflow import keras
from tensorflow.keras import layers

# Instantiate a trained vision model
vision_model = keras.applications.ResNet50()

# This is our video encoding branch using the trained vision_model
video_input = keras.Input(shape=(100, None, None, 3))
encoded_frame_sequence = layers.TimeDistributed(vision_model)(video_input)
encoded_video = layers.LSTM(256)(encoded_frame_sequence)

# This is our text-processing branch for the question input
question_input = keras.Input(shape=(100,), dtype='int32')
embedded_question = layers.Embedding(10000, 256)(question_input)
encoded_question = layers.LSTM(256)(embedded_question)

# And this is our video question answering model
merged = keras.layers.concatenate([encoded_video, encoded_question])
output = keras.layers.Dense(1000, activation='softmax')(merged)
video_qa_model = keras.Model([video_input, question_input],
                              output)
```

Deep learning for humans.

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.

Comentários Finais

- **Muitas bibliotecas de processamento de imagens e OCR são baseadas em C++/CUDA, exigindo um esforço adicional de OS, Hardware e requerimentos para uso (como por exemplo, GPUs)**
- **Conteúdos desse seminário estão disponíveis em repositório do GitHub (<https://github.com/gabogomes/ocr-fgv>) e no Google Drive**