Desenvolvimento de uma API de reconhecimento facial em Python

Prof. João Robson

Agenda

01

O que é reconhecimento facial?

02

O que é uma API?

03

Como criar uma API em Python?

04

Como realizar reconhecimento facial em Python?

05

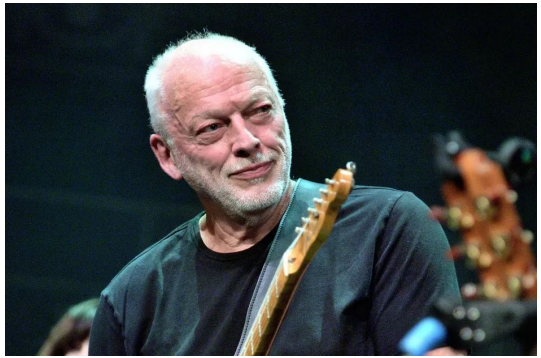
Referências +
Materiais de estudo +
Comentários finais

01

O QUE É RECONHECIMENTO FACIAL?

O QUE É RECONHECIMENTO FACIAL?

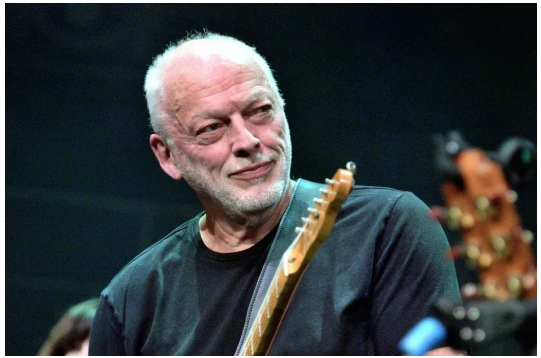
- Técnica para **identificar a identidade** de uma pessoa com base em suas características faciais;
- Esse processo envolve a **captura de uma imagem ou vídeo do rosto** de uma pessoa e a **comparação com uma base de dados** de faces previamente armazenadas;



Exemplo: reconhecimento facial de músicos

COMO O RECONHECIMENTO FACIAL FUNCIONA?

- Geralmente, o processo de reconhecimento facial consiste em 5 etapas: detecção, alinhamento, normalização, representação e verificação.
- Após a aplicação das etapas, caso haja correspondência entre a face analisada e as faces armazenadas, o sistema pode retornar a identidade da pessoa, como mostrado abaixo:

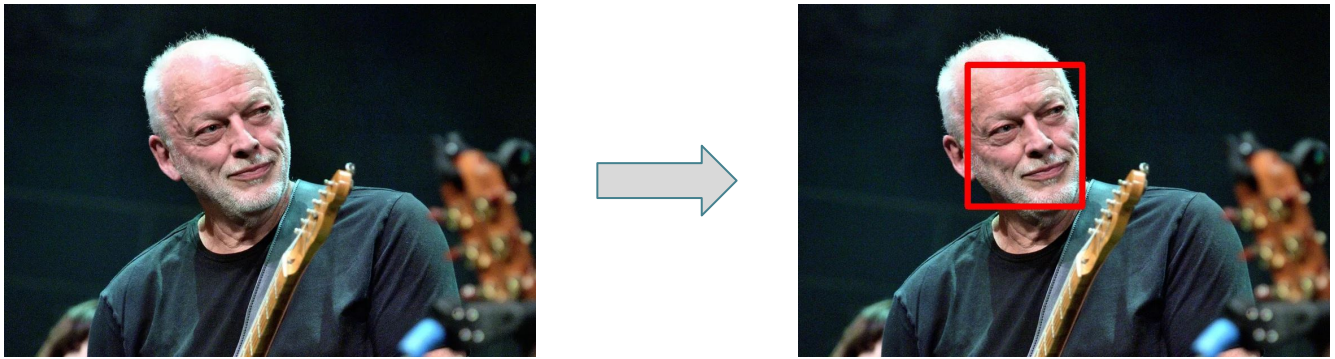


David Gilmour

Exemplo: reconhecimento facial de músicos

ETAPAS DO RECONHECIMENTO FACIAL - DETECÇÃO

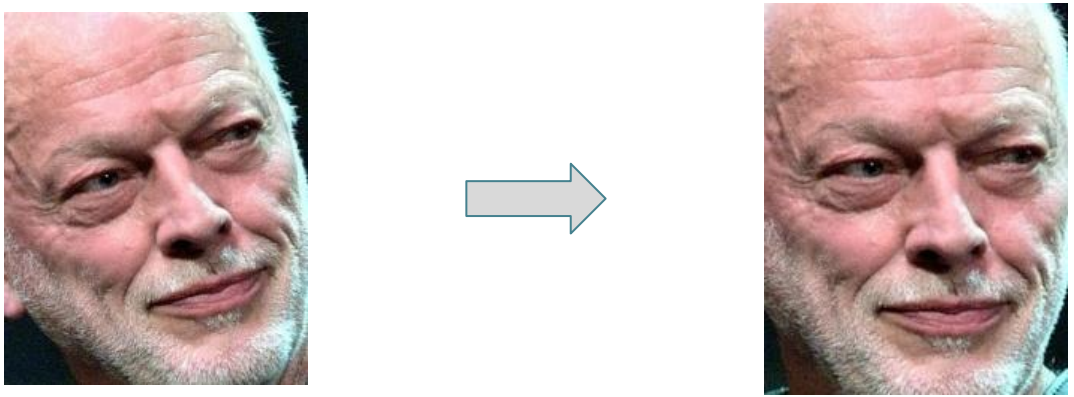
- A etapa de detecção consiste em identificar a posição das faces em uma imagem;
- Isso geralmente é feito por um **modelo treinado para predizer as regiões de uma imagem em que existem (possíveis) faces**;
- Exemplos de modelos de detecção facial: YuNet, SSD (Single-Shot Multibox Detector), Haar Cascade.



Exemplo de detecção facial

ETAPAS DO RECONHECIMENTO FACIAL - ALINHAMENTO

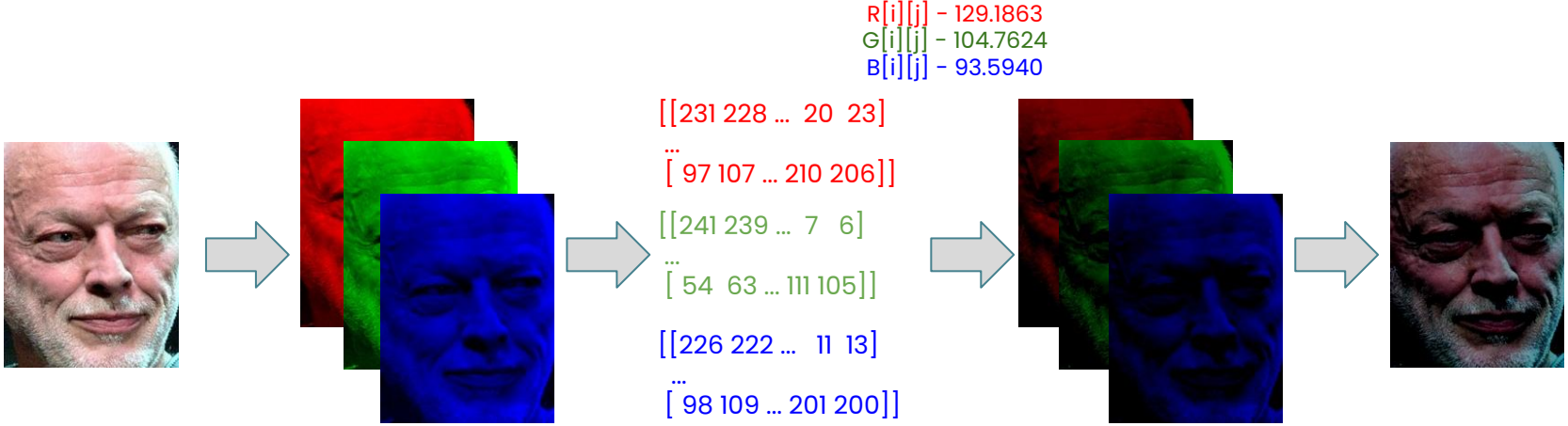
- A etapa de alinhamento consiste em **ajustar e posicionar uma imagem do rosto** para garantir que características faciais importantes (como olhos, nariz e boca) estejam sempre na mesma posição relativa;
- Isso ajuda a melhorar a precisão dos modelos de reconhecimento facial, garantindo que as variações de **rotação, escala e translação** (deslocamento) do rosto **não afetem o desempenho do sistema**.



Exemplo de alinhamento facial

ETAPAS DO RECONHECIMENTO FACIAL - NORMALIZAÇÃO

- O processo de normalização consiste em **ajustar os valores dos pixels das imagens** para garantir que todas as imagens tenham características semelhantes;
- Por meio deste pré-processamento, **que é opcional**, há a possibilidade de melhora na **precisão e a eficácia dos modelos de reconhecimento facial,** pois as imagens ficam com uma distribuição de valores mais uniforme;
- Exemplo de normalização: subtração dos valores médios dos pixels de cada canal (vermelho, verde, azul) baseados no conjunto de dados de treinamento do modelo;



Exemplo de normalização

ETAPAS DO RECONHECIMENTO FACIAL - REPRESENTAÇÃO

- A etapa de representação consiste no processo de **transformar a imagem de um rosto** em uma **representação numérica** (também chamada de vetor ou *embedding*) que captura as características únicas da face;
- Esse passo é importante pois garante uma maior eficiência na comparação das faces na etapa de verificação por meio **da redução de dimensionalidade do dado armazenado** (256 ou 512 dimensões, por exemplo, ao invés de milhões de pixels);
- Exemplos de modelos que realizam representação: FaceNet, VGG-Face, ArcFace

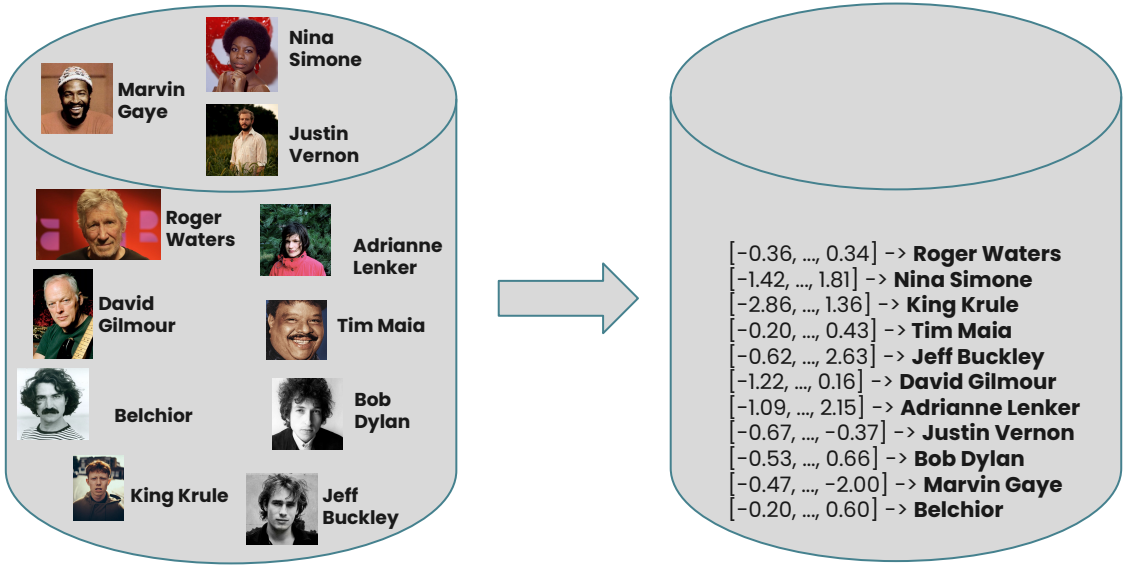
Facenet
(128 dim.)

[-1.3476386070251465, -0.5846326351165771, 0.027262166142463684, 0.6717081665992737, 1.2571663856506348, -0.16742382943630219, 0.9175747632980347, -0.28482910990715027, 1.4241315126419067, 1.3159728050231934, 0.7263108491897583, -1.567671775817871, -0.9662340879440308, -0.12014228105545044, 0.3984776437282562, -0.285436749458313, -2.457336902618408, -0.1456911414861679, 0.10687001794576645, -0.44676926732063293, -0.07382138073444366, 0.20167051255702972, -0.3917120695114136, 1.780278205871582, 0.0030873091891407967, 0.7608511447906494, 1.0888515710830688, -2.1178975105285645, 0.8003355860710144, -1.5119686126708984, -0.7157609462738037, -0.21858416497707367, -1.4923112392425537, 1.0008957386016846, 0.0030315304175019264, 0.7668362855911255, -0.7617815732955933, 0.5258893370628357, 1.293231129043579, -1.22344970703125, 1.72514009475708, -0.3174593150615692, -1.5061317682266235, -1.3124375343322754, -1.1244232654571533, 0.15241891145706177, -0.6436970233917236, -0.23637038469314575, 0.03004347532975946, -0.3954750895500183, -1.5334522724151611, 1.6850513219833374, -0.2036539614200592, 0.36906981468200684, -0.6699639558792114, -1.0092374086380005, 1.8128081560134888, -0.0832395926117897, -0.1366286426782608, -1.6233313083648682, -0.1886400580406189, -0.7690274119377136, -0.10093852877616882, 0.5512117743492126, 0.1662171483039856, 1.3166465759277344, 1.3015754222869873, -0.11710339784622192, 0.28954070806503296, -0.5312575101852417, -0.6495854258537292, -2.0727763175964355, -0.4357745051383972, 0.3417588472366333, -0.13255411386489868, -0.6127800345420837, -1.2754700183868408, 1.1783939599990845, -1.8674168586730957, 0.7170730233192444, -0.2668403685092926, -1.1272724866867065, -0.3314240574836731, -0.6099909543991089, -0.5256004929542542, 1.5979743003845215, -0.506541907787323, 2.86336588595581, -1.0662106275558472, 0.7316594123840332, 1.4648940563201904, -0.012669753283262253, 1.9784188270568848, 0.9789175391197205, 0.5424352884292603, 0.21212640404701233, -0.416063517332077, 0.21363820135593414, 0.7134866714477539, 0.6055770516395569, -3.2417998504638672, 0.08981551975011826, 0.0085846185684241, 0.19696812331676483, 1.510533332824707, 0.6542445421218872, 0.16822265088558197, 1.682959794998169, -0.39710742235183716, -0.1071283146739006, 0.8109453320503235, -0.7549762725830078, 0.0491647943854332, -0.23308506608009338, -0.21244534850120544, 1.055838942527771, -0.42175209522247314, -0.6758638024330139, 0.4206183850765228, 1.1604241132736206, -2.328225612640381, 0.25608932971954346, -1.0041340589523315, -1.3608835935592651, 0.2871673107147217, 0.4474157989025116, 0.6472117900848389, 0.4510827362537384]

Exemplo de representação facial

ETAPAS DO RECONHECIMENTO FACIAL - VERIFICAÇÃO

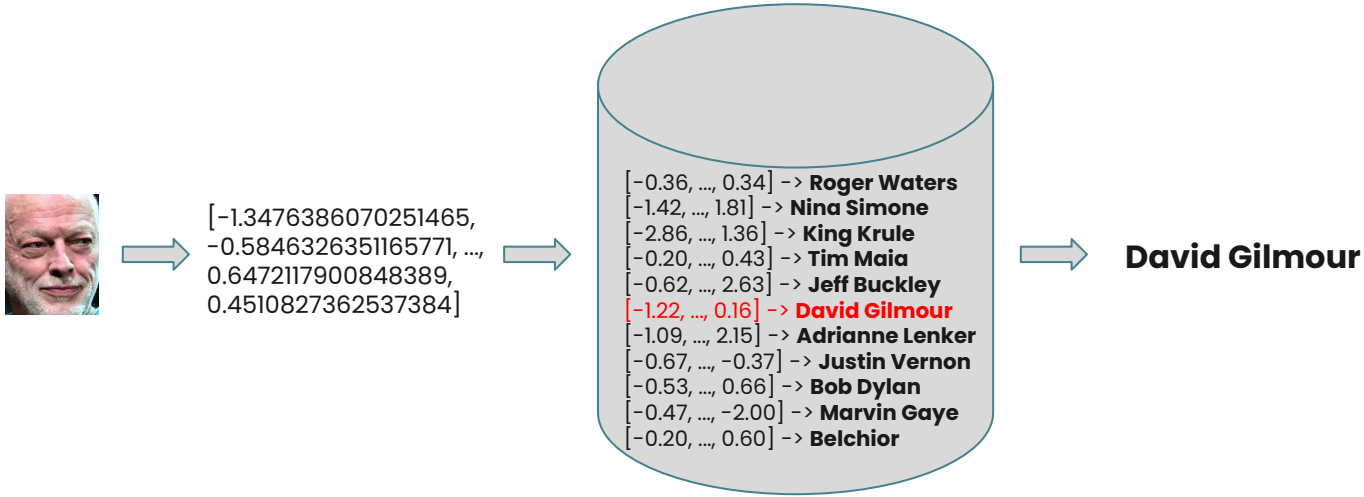
- A etapa de verificação consiste em **checar se a imagem fornecida** de uma face **corresponde a alguma face** já armazenada;
- Esse processo geralmente é feito por meio da **comparação entre o *embedding*** de uma imagem de entrada **com vetores já armazenados em um banco de dados**. **Assim, antes da verificação acontecer de fato, é necessário um banco de dados de vetores de faces já criado**.



Geração dos embeddings de um banco de dados de imagens

ETAPAS DO RECONHECIMENTO FACIAL - VERIFICAÇÃO

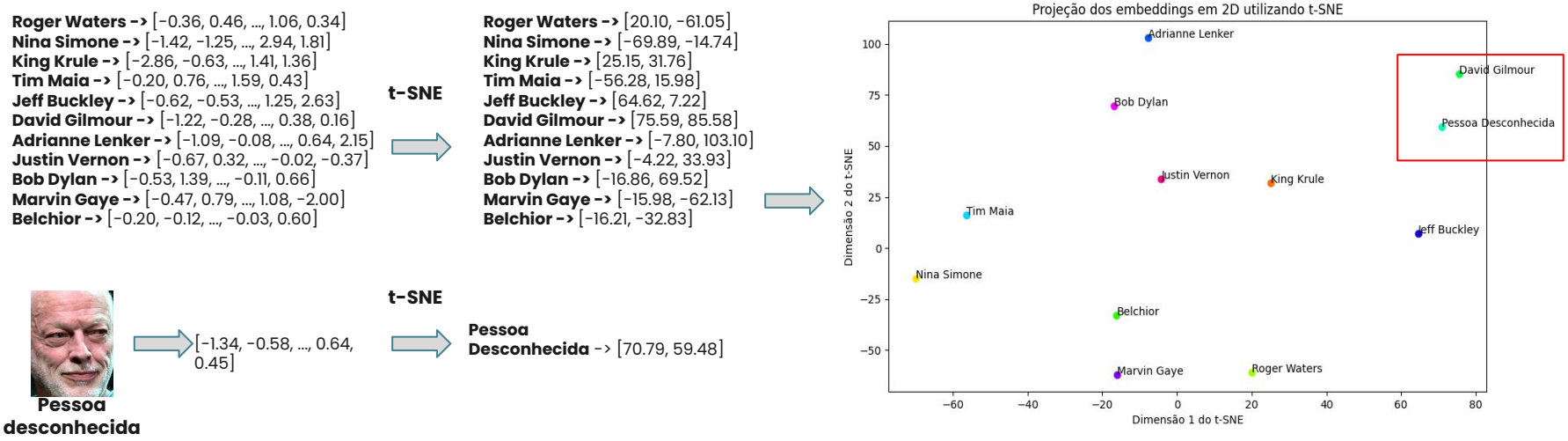
- Com isso, a verificação em si consiste na checagem da similaridade ou distância entre o vetor da face fornecida e os vetores armazenados;
- A **proximidade entre os vetores** é um indicativo entre a **similaridade entre as faces**, ou seja, quanto mais próximos no espaço, maior a similaridade entre as faces que os originaram;



Exemplo de verificação facial

ETAPAS DO RECONHECIMENTO FACIAL - VERIFICAÇÃO

- Como os *embeddings* geralmente possuem dezenas ou centenas de dimensões, é impossível visualizar diretamente suas distâncias;
- Para isso, pode-se utilizar algoritmos de redução de dimensionalidade a fim de checar visualmente a distribuição aproximada dos *embeddings* originais no espaço;
- Exemplos de algoritmos para redução de dimensionalidade: PCA, t-SNE



Uso do t-SNE para redução de dimensionalidade dos *embeddings*

ETAPAS DO RECONHECIMENTO FACIAL - VERIFICAÇÃO

- Na prática, o que se faz é usar **algum algoritmo de cálculo de distância entre vetores**;
- Assim, o vetor de entrada é comparado com todos os vetores armazenados. O vetor ou o conjunto de vetores mais próximos são utilizados para predizer a identidade da face de entrada;
- Algoritmos geralmente utilizados para cálculo da distância:
  - Similaridade de cosseno:

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

  - Distância euclidiana (l2):

$$d(x, y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

ETAPAS DO RECONHECIMENTO FACIAL - VERIFICAÇÃO

- Nos *embeddings* de exemplo, a comparação por distância euclidiana resultaria no seguinte:

Pessoa desconhecida

[-1.34, -0.58, ..., 0.64, 0.45]

[-0.36, ..., 0.34] -> **Roger Waters**
[-1.42, ..., 1.81] -> **Nina Simone**
[-2.86, ..., 1.36] -> **King Krule**
[-0.20, ..., 0.43] -> **Tim Maia**
[-0.62, ..., 2.63] -> **Jeff Buckley**
[-1.22, ..., 0.16] -> **David Gilmour**
[-1.09, ..., 2.15] -> **Adrianne Lenker**
[-0.67, ..., -0.37] -> **Justin Vernon**
[-0.53, ..., 0.66] -> **Bob Dylan**
[-0.47, ..., -2.00] -> **Marvin Gaye**
[-0.20, ..., 0.60] -> **Belchior**

d("Pessoa desconhecida", "Roger Waters") = 16.37
d("Pessoa desconhecida", "Nina Simone") = 18.51
d("Pessoa desconhecida", "King Krule") = 13.53
d("Pessoa desconhecida", "Tim Maia") = 17.36
d("Pessoa desconhecida", "Jeff Buckley") = 16.87
d("Pessoa desconhecida", "David Gilmour") = 8.55
d("Pessoa desconhecida", "Adrianne Lenker") = 16.44
d("Pessoa desconhecida", "Justin Vernon") = 14.56
d("Pessoa desconhecida", "Bob Dylan") = 15.82
d("Pessoa desconhecida", "Marvin Gaye") = 17.93
d("Pessoa desconhecida", "Belchior") = 17.74

TAREFAS

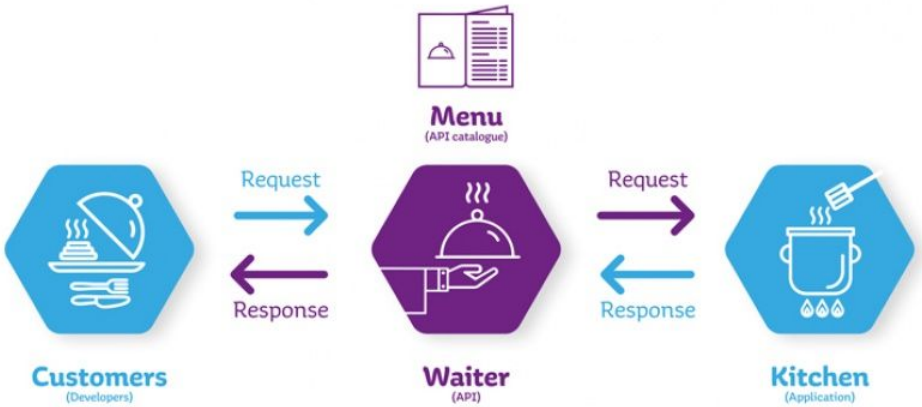
- Clonar repositório da oficina;
- Instalar bibliotecas;
- Executar *scripts* com etapas do reconhecimento facial:
  - detect.py
  - align.py
  - normalize.py
  - represent.py
  - verify.py

02

O QUE É UMA API?

O QUE É UMA API?

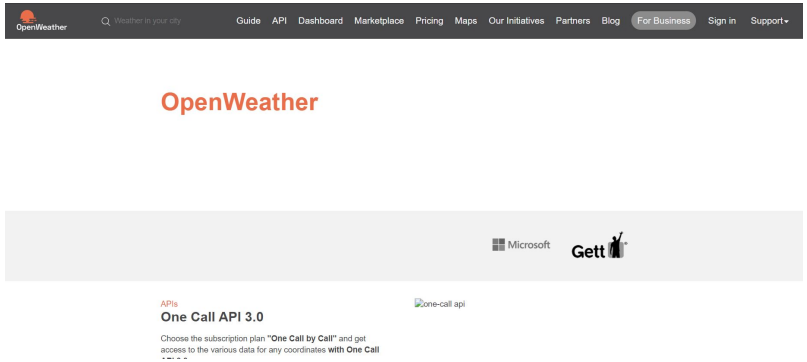
- Uma API (Application Programming Interface) é um **conjunto de regras e protocolos** que permite que **diferentes sistemas ou serviços** se **comuniquem entre si**.
- Por meio de uma API, é definido como os componentes de software devem interagir, fornecendo funções e comandos para acessar recursos ou executar tarefas em outro sistema;
- Analogia: restaurante

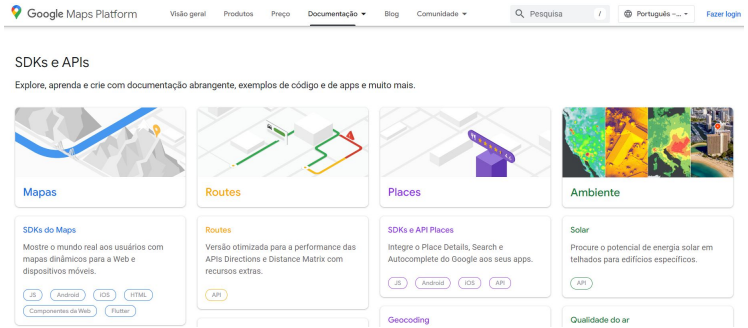


Fonte: API: Definition and application in procurement

COMO USAR UMA API NA PRÁTICA?

- APIs são fundamentais na programação moderna, pois **possibilitam a integração de diferentes sistemas e serviços**;
- Por exemplo, uma API de clima permite que você obtenha informações meteorológicas para seu aplicativo sem precisar construir todo o sistema de coleta de dados climáticos;
- Exemplos de APIs: OpenWeather, Google Maps.





Páginas das documentações do OpenWeather e Google Maps Platform

03

COMO CRIAR UMA API EM PYTHON?

COMO CRIAR UMA API EM PYTHON?

- Para criar APIs em Python, geralmente se utiliza algum *framework* ou biblioteca;
- Em geral, essas ferramentas permitem a criação de uma API REST, que **utiliza o protocolo HTTP** para envio e recebimento de dados;
- Há várias opções *open-source* para isso:
  - Flask: Um micro-framework leve e fácil de usar.
  - FastAPI: Ideal para APIs assíncronas e com suporte a validação automática de dados.
  - Django REST Framework: Para aplicações maiores que já usam Django, que é um *framework full-stack*, capaz de realizar facilmente integração com banco de dados e autenticação, por exemplo.
- Geralmente
- Na oficina de hoje, iremos utilizar **FastAPI:**
  - Algumas vantagens:
    - Desempenho;
    - Validação automática de dados;
    - Documentação interativa da API.
- **O código inicial pode ser encontrado aqui:**
  - https://github.com/joaorobson/facial_recognition_workshop

TAREFAS

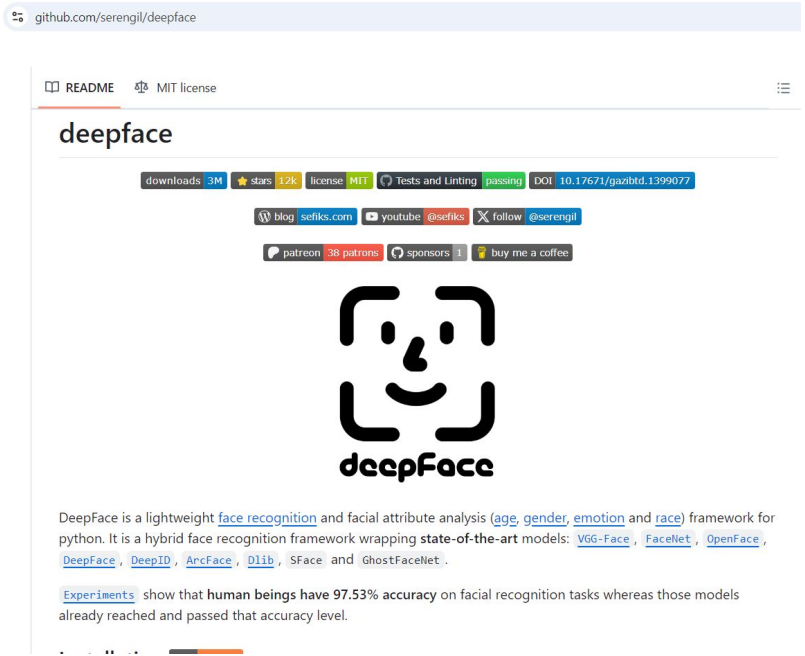
- Executar API;
- Testar *endpoints* já criados;
- Criar um novo *endpoint* que retorna a soma de 2 números passados na requisição.

04

COMO REALIZAR RECONHECIMENTO FACIAL EM PYTHON?

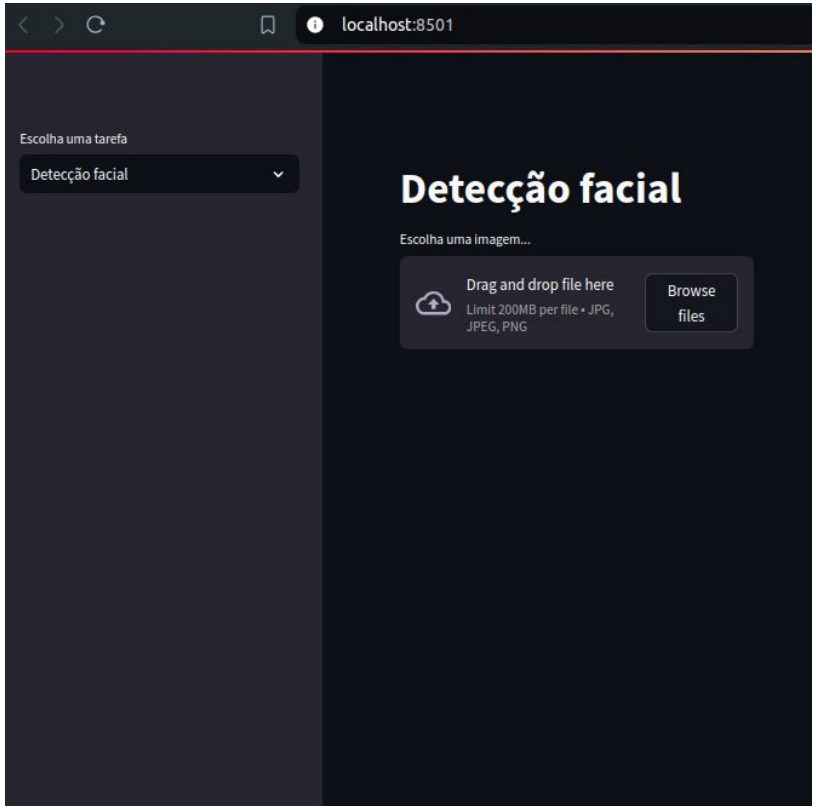
COMO REALIZAR RECONHECIMENTO FACIAL EM PYTHON?

- Para realizar reconhecimento facial em Python, geralmente se utiliza alguma biblioteca com modelos **pré-treinados**;
- Por meio de tais modelos, é possível realizar inferências sobre faces, sem nos preocuparmos com detalhes internos do seu funcionamento;
- Uma biblioteca bastante usada atualmente é a **deepface;**
- Ela é *open-source*, encapsulando e facilitando o uso dos modelos mais recentes de detecção e reconhecimento facial;
- Exemplos de outras bibliotecas: OpenCV, Dlib.



COMO REALIZAR RECONHECIMENTO FACIAL EM PYTHON?

- Para facilitar a visualização dos resultados das detecções e do reconhecimento facial, usaremos uma *framework* chamado **Streamlit** para gerar um app interativo em Python;
- Por meio dele, podemos gerar uma interface gráfica simples que nos permitirá chamar os *endpoints* criados.



TAREFAS

- Executar *frontend* da aplicação;
- Testar *endpoint* de **detecção facial** por meio da interface gráfica;
- Executar *script* para criação da base vetorial;
- Testar reconhecimento facial com *script* local;
- Terminar implementação do *endpoint* de reconhecimento facial;
- Testar *endpoint* de **reconhecimento facial** por meio da interface gráfica.
- Ajustar *threshold*;
- Incluir faces de um novo artista;
- Atualizar base vetorial;
- Testar *endpoint* de **reconhecimento facial** por meio da interface gráfica;
- Implementar função *lifespan* no código da API.

05

REFERÊNCIAS + MATERIAL DE ESTUDO + COMENTÁRIOS FINAIS

REFERÊNCIAS

- DeepFace. Disponível em: https://github.com/serengil/deepface
- Rosebrock, Adrian. Face recognition with OpenCV, Python, and deep learning. Disponível em: https://pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/
- Serengil, Sefik. A Gentle Introduction to Face Recognition in Deep Learning. Disponível em: https://sefiks.com/2020/05/01/a-gentle-introduction-to-face-recognition-in-deep-learning/

MATERIAIS DE ESTUDO

- Cursos:
  - Álgebra Linear
    - Gilbert Strang lectures on Linear Algebra (MIT)
  - Visão computacional
    - 3blue1brown
      - But what is a convolution?
    - StatQuest
      - Neural Networks Part 8: Image Classification with Convolutional Neural Networks (CNNs)
    - Udemy
      - The Ultimate Beginners Guide to Face Detection & Recognition.
- Livros:
  - James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). **An introduction to statistical learning: With applications in python** (1st ed.). Springer International Publishing. Disponível em: https://www.statlearning.com/
- Sites:
  - Documentação do OpenCV: https://docs.opencv.org/4.x/

COMENTÁRIOS FINAIS

- Dúvidas?
- Contatos:
  - LinkedIn: https://br.linkedin.com/in/joaorobson
  - GitHub: https://github.com/joaorobson

OBRIGADO!