Recunoasterea fetelor

Recunoasterea fetelor implica identificarea persoanei. In esenta, este un proces de detectie - clasificare in care imaginile fiecarei persoane sunt grupate in propria clasa.

Insightface (insightface.ai este un instrument software care ofera (printre altele) instrumentele necesare pentru constructia unui sistem de recunoastere a fetelor.

Constructia unui sistem de recunoasterea a fetelor implica urmatoarele etape

1. Acumularea imaginilor
2. Extragerea embedding-urilor (amprenta numerica a fiecarei imagini)
3. Stocarea grupata a acestor amprente (fiecare persoana este reprezentata de mai multe fete iar fiecare fata va a vea o amprenta)

Alegerea imaginilor depinde de scop - un sistem de "face unlocking" trebuie sa aiba doar imagini curente in timp de un sistem de tip arhiva trebuie sa aiba imagini din toate varstele unei persoane.

Pregatiri

Instalarea modulelor onnxruntime si insightface. ONNX Runtime este un motor de inferență rapid și eficient care rulează modele AI salvate în formatul standard ONNX, optimizând execuția pe diverse platforme și hardware. Insightface stocheaza modelele in format onnx.

```
!pip install onnxruntime
```

```
Requirement already satisfied: onnxruntime in
/usr/local/lib/python3.11/dist-packages (1.22.0)
Requirement already satisfied: coloredlogs in
/usr/local/lib/python3.11/dist-packages (from onnxruntime) (15.0.1)
Requirement already satisfied: flatbuffers in
/usr/local/lib/python3.11/dist-packages (from onnxruntime) (25.2.10)
Requirement already satisfied: numpy>=1.21.6 in
/usr/local/lib/python3.11/dist-packages (from onnxruntime) (2.0.2)
Requirement already satisfied: packaging in
/usr/local/lib/python3.11/dist-packages (from onnxruntime) (24.2)
Requirement already satisfied: protobuf in
/usr/local/lib/python3.11/dist-packages (from onnxruntime) (5.29.5)
Requirement already satisfied: sympy in
/usr/local/lib/python3.11/dist-packages (from onnxruntime) (1.13.1)
Requirement already satisfied: humanfriendly>=9.1 in
/usr/local/lib/python3.11/dist-packages (from coloredlogs-
>onnxruntime) (10.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.11/dist-packages (from sympy->onnxruntime)
(1.3.0)
```

```
!pip install insightface

Collecting insightface
  Downloading insightface-0.7.3.tar.gz (439 kB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.0/439.5 kB ? eta -:--:--
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 174.1/439.5 kB 4.8 MB/s eta
0:00:01 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 439.5/439.5 kB 6.2
MB/s eta 0:00:00
ents to build wheel ... etadata (pyproject.toml) ... ent already
satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from
insightface) (2.0.2)
Collecting onnx (from insightface)
  Downloading onnx-1.18.0-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl.metadata (6.9 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-
packages (from insightface) (4.67.1)
Requirement already satisfied: requests in
/usr/local/lib/python3.11/dist-packages (from insightface) (2.32.3)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.11/dist-packages (from insightface) (3.10.0)
Requirement already satisfied: Pillow in
/usr/local/lib/python3.11/dist-packages (from insightface) (11.2.1)
Requirement already satisfied: scipy in
/usr/local/lib/python3.11/dist-packages (from insightface) (1.15.3)
Requirement already satisfied: scikit-learn in
/usr/local/lib/python3.11/dist-packages (from insightface) (1.6.1)
Requirement already satisfied: scikit-image in
/usr/local/lib/python3.11/dist-packages (from insightface) (0.25.2)
Requirement already satisfied: easydict in
/usr/local/lib/python3.11/dist-packages (from insightface) (1.13)
Requirement already satisfied: cython in
/usr/local/lib/python3.11/dist-packages (from insightface) (3.0.12)
Requirement already satisfied: albumentations in
/usr/local/lib/python3.11/dist-packages (from insightface) (2.0.8)
Requirement already satisfied: prettytable in
/usr/local/lib/python3.11/dist-packages (from insightface) (3.16.0)
Requirement already satisfied: PyYAML in
/usr/local/lib/python3.11/dist-packages (from albumentations-
>insightface) (6.0.2)
Requirement already satisfied: pydantic>=2.9.2 in
/usr/local/lib/python3.11/dist-packages (from albumentations-
>insightface) (2.11.7)
Requirement already satisfied: albucore==0.0.24 in
/usr/local/lib/python3.11/dist-packages (from albumentations-
>insightface) (0.0.24)
Requirement already satisfied: opencv-python-headless>=4.9.0.80 in
/usr/local/lib/python3.11/dist-packages (from albumentations-
>insightface) (4.11.0.86)
Requirement already satisfied: stringzilla>=3.10.4 in
/usr/local/lib/python3.11/dist-packages (from albucore==0.0.24-
```

```
>albumentations->insightface) (3.12.5)
Requirement already satisfied: simsimd>=5.9.2 in
/usr/local/lib/python3.11/dist-packages (from albucore==0.0.24-
>albumentations->insightface) (6.4.9)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->insightface)
(1.3.2)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->insightface)
(0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->insightface)
(4.58.4)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->insightface)
(1.4.8)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->insightface)
(24.2)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->insightface)
(3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.11/dist-packages (from matplotlib->insightface)
(2.9.0.post0)
Requirement already satisfied: protobuf>=4.25.1 in
/usr/local/lib/python3.11/dist-packages (from onnx->insightface)
(5.29.5)
Requirement already satisfied: typing_extensions>=4.7.1 in
/usr/local/lib/python3.11/dist-packages (from onnx->insightface)
(4.14.0)
Requirement already satisfied: wcwidth in
/usr/local/lib/python3.11/dist-packages (from prettytable-
>insightface) (0.2.13)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.11/dist-packages (from requests->insightface)
(3.4.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.11/dist-packages (from requests->insightface)
(3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.11/dist-packages (from requests->insightface)
(2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.11/dist-packages (from requests->insightface)
(2025.6.15)
Requirement already satisfied: networkx>=3.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-image-
>insightface) (3.5)
```

```
Requirement already satisfied: imageio!=2.35.0,>=2.33 in
/usr/local/lib/python3.11/dist-packages (from scikit-image-
>insightface) (2.37.0)
Requirement already satisfied: tifffile>=2022.8.12 in
/usr/local/lib/python3.11/dist-packages (from scikit-image-
>insightface) (2025.6.11)
Requirement already satisfied: lazy-loader>=0.4 in
/usr/local/lib/python3.11/dist-packages (from scikit-image-
>insightface) (0.4)
Requirement already satisfied: joblib>=1.2.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn-
>insightface) (1.5.1)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/usr/local/lib/python3.11/dist-packages (from scikit-learn-
>insightface) (3.6.0)
Requirement already satisfied: annotated-types>=0.6.0 in
/usr/local/lib/python3.11/dist-packages (from pydantic>=2.9.2-
>albumentations->insightface) (0.7.0)
Requirement already satisfied: pydantic-core==2.33.2 in
/usr/local/lib/python3.11/dist-packages (from pydantic>=2.9.2-
>albumentations->insightface) (2.33.2)
Requirement already satisfied: typing-inspection>=0.4.0 in
/usr/local/lib/python3.11/dist-packages (from pydantic>=2.9.2-
>albumentations->insightface) (0.4.1)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.7-
>matplotlib->insightface) (1.17.0)
Downloading onnx-1.18.0-cp311-cp311-
manylinux_2_17_x86_64.manylinux2014_x86_64.whl (17.6 MB)
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.6/17.6 MB 88.7 MB/s eta
0:00:00
l) ... e=insightface-0.7.3-cp311-cp311-linux_x86_64.whl size=1060442
sha256=8717c500d1425159f2f8bc4642d42f9ee8f73149e7b06f0bfd129a14802ecb1
3
  Stored in directory:
/root/.cache/pip/wheels/27/d8/22/f52d858d16cd06e7b2e6aad34a1777dcfaf00
0be833bbf8146
Successfully built insightface
Installing collected packages: onnx, insightface
Successfully installed insightface-0.7.3 onnx-1.18.0
```

Montarea google drive

Datele folosite in acest program sunt citite / scrise in Google Drive. Acest lucru este optional.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

Setarea cailor principale

Acest notebook foloseste [Celebrity Face Image Dataset]
(https://www.kaggle.com/datasets/vishesh1412/celebrity-face-image-dataset). Acest set de date contine 18 subdirectoare, fiecare continand in jur de 100 de fotografii ale aceleiasi persoane.

```
import os
PROJECT_PATH = os.path.join('/content/drive/MyDrive/02_Fete')
data_dir = os.path.join(PROJECT_PATH, 'Celebrity Faces Dataset')
model_dir = os.path.join(PROJECT_PATH, 'model')
```

Structura de date este urmatoarea

```
variabila PROJECT_PATH
├── Celebrity Faces Dataset (variabila data_dir)
│   ├── Angelina Jolie
│   ├── Brad Pitt
│   ├── Denzel Washington
│   ├── Hugh Jackman
│   ├── Jennifer Lawrence
│   ├── Johnny Depp
│   ├── Kate Winslet
│   ├── Leonardo DiCaprio
│   ├── Megan Fox
│   ├── Natalie Portman
│   ├── Nicole Kidman
│   ├── Robert Downey Jr
│   ├── Sandra Bullock
│   ├── Scarlett Johansson
│   ├── Tom Cruise
│   ├── Tom Hanks
│   └── Will Smith
├── model (variabila model_dir)
│   └── celebrity_faces.pkl
```

Instalarea modulelor

```
import zipfile
from google.colab import files
from IPython.display import display
from google.colab.patches import cv2_imshow
import numpy as np
import insightface
from insightface.app import FaceAnalysis
```

```
import cv2
from insightface.utils import face_align
from tqdm import tqdm
from sklearn.manifold import TSNE
import umap
import pickle
import matplotlib.pyplot as plt
from PIL import Image
from sklearn.metrics.pairwise import cosine_similarity
```

Initializarea modelului Insightface

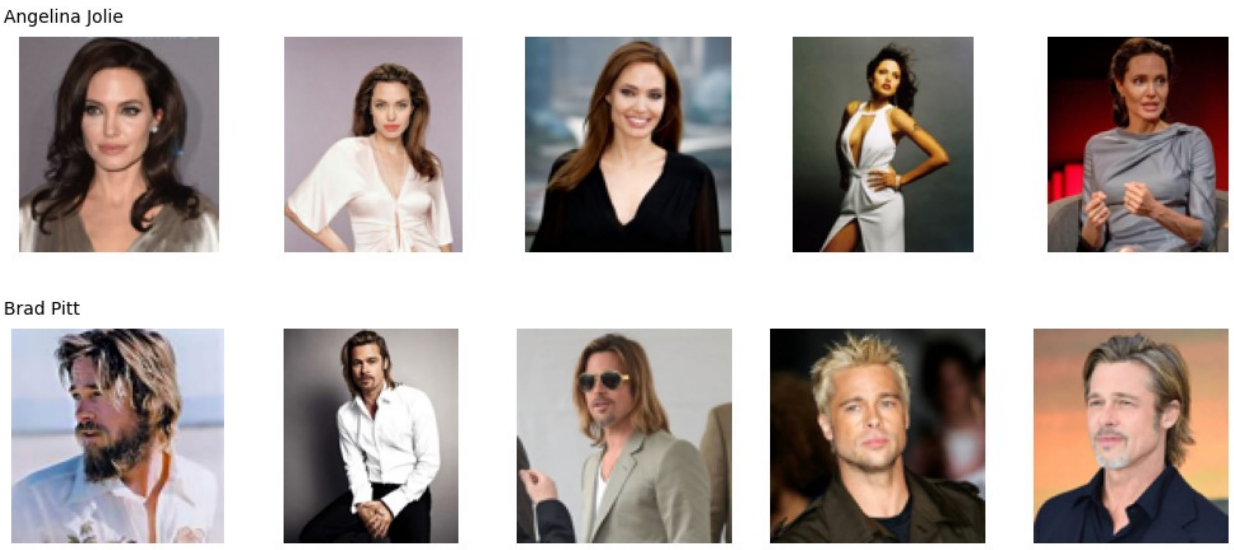
```
model = insightface.app.FaceAnalysis(name='buffalo_l',
providers=['CPUExecutionProvider'])
model.prepare(ctx_id=0)
```

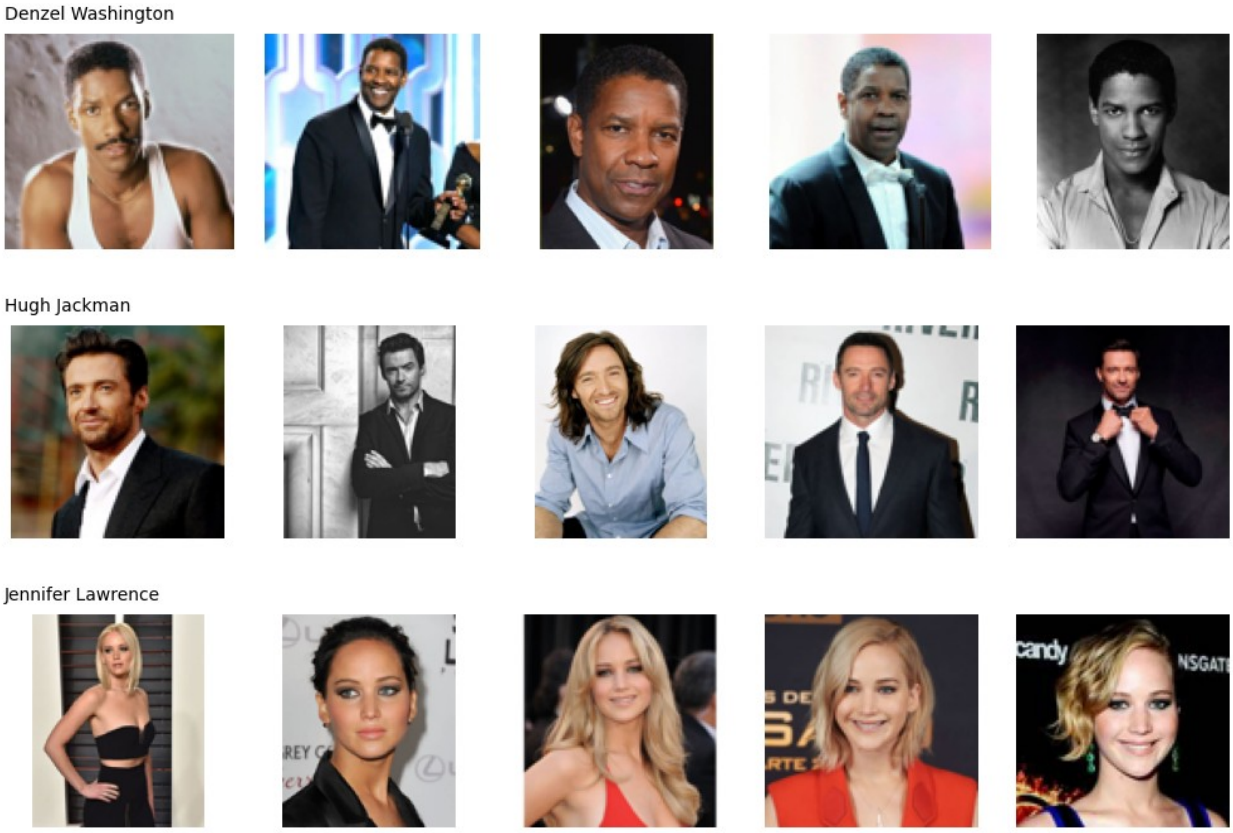
```
download_path: /root/.insightface/models/buffalo_l
Downloading /root/.insightface/models/buffalo_l.zip from
https://github.com/deepinsight/insightface/releases/download/v0.7/
buffalo_l.zip...

100%|██████████| 281857/281857 [00:04<00:00, 65498.99KB/s]

Applied providers: ['CPUExecutionProvider'], with options:
{'CPUExecutionProvider': {}}
find model: /root/.insightface/models/buffalo_l/1k3d68.onnx
landmark_3d_68 ['None', 3, 192, 192] 0.0 1.0
Applied providers: ['CPUExecutionProvider'], with options:
{'CPUExecutionProvider': {}}
find model: /root/.insightface/models/buffalo_l/2d106det.onnx
landmark_2d_106 ['None', 3, 192, 192] 0.0 1.0
Applied providers: ['CPUExecutionProvider'], with options:
{'CPUExecutionProvider': {}}
find model: /root/.insightface/models/buffalo_l/det_10g.onnx detection
[1, 3, '?', '?'] 127.5 128.0
Applied providers: ['CPUExecutionProvider'], with options:
{'CPUExecutionProvider': {}}
find model: /root/.insightface/models/buffalo_l/genderage.onnx
genderage ['None', 3, 96, 96] 0.0 1.0
Applied providers: ['CPUExecutionProvider'], with options:
{'CPUExecutionProvider': {}}
find model: /root/.insightface/models/buffalo_l/w600k_r50.onnx
recognition ['None', 3, 112, 112] 127.5 127.5
set det-size: (640, 640)
```

Afisarea datelor

Putem afisa o mica parte din imagini - primele 5 subdirectoare cate 5 imagini

```
subdirs = [d for d in os.listdir(data_dir) if
os.path.isdir(os.path.join(data_dir, d))]

for subdir in sorted(subdirs)[:5]:
    subdir_path = os.path.join(data_dir, subdir)
    image_files = [f for f in os.listdir(subdir_path) if
f.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif'))]

    if not image_files:
        continue  # Skip empty directories

    n_imgs = min(5, len(image_files))
    fig, axes = plt.subplots(1, n_imgs, figsize=(n_imgs * 2, 2),
constrained_layout=True)

    fig.suptitle(subdir, fontsize=10, x=0.01, ha='left')

    if n_imgs == 1:
        axes = [axes]  # Ensure it's iterable if only one subplot

    for ax, img_name in zip(axes, image_files[:n_imgs]):
        img_path = os.path.join(subdir_path, img_name)
        img = Image.open(img_path)

        width = 100
        aspect_ratio = img.height / img.width
        img_resized = img.resize((width, int(width * aspect_ratio)))

        ax.imshow(img_resized)
        ax.axis('off')

    plt.show()
```





Constructia bazei de date

Urmarim sa obtinem un dictionar (`embedding_dict`) care contine pentru fiecare subdirector (echivalent cu o persoana) o lista de embeddings.

Instructiunea `%%time` masoara durata executiei celulei. Dureaza in jur de 40 - 45 min.

```
%%time

embeddings_dict = {}

for subdir in os.listdir(data_dir):
    subdir_path = os.path.join(data_dir, subdir)
    if os.path.isdir(subdir_path):
        embeddings_dict[subdir] = []
        files = sorted(os.listdir(subdir_path))  # Limit to 10 files,
sorted for consistency
        for file in tqdm(files, desc=f'Processing {subdir}'):
            img_path = os.path.join(subdir_path, file)
            img = cv2.imread(img_path)
            if img is None:
                continue
            faces = model.get(img)
            if len(faces) == 0:
```

```
                continue
            embedding = faces[0].embedding
            embeddings_dict[subdir].append(embedding)
```

```
Processing Tom Hanks: 100%|██████████| 100/100 [01:55<00:00,
1.15s/it]
Processing Will Smith: 100%|██████████| 100/100 [02:00<00:00,
1.21s/it]
Processing Leonardo DiCaprio: 100%|██████████| 100/100 [02:04<00:00,
1.25s/it]
Processing Nicole Kidman: 100%|██████████| 100/100 [02:03<00:00,
1.24s/it]
Processing Robert Downey Jr: 100%|██████████| 100/100 [02:07<00:00,
1.27s/it]
Processing Natalie Portman: 100%|██████████| 100/100 [02:00<00:00,
1.21s/it]
Processing Tom Cruise: 100%|██████████| 100/100 [02:08<00:00,
1.29s/it]
Processing Scarlett Johansson: 100%|██████████| 200/200 [04:00<00:00,
1.20s/it]
Processing Megan Fox: 100%|██████████| 100/100 [01:54<00:00,
1.15s/it]
Processing Sandra Bullock:  79%|███████   | 79/100 [01:40<00:26,
1.27s/it]

---------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
last)
<timed exec> in <module>

KeyboardInterrupt:
```

Salvarea bazei de date

Pickle permite salvarea și încărcarea obiectelor în format binar, pentru a fi stocate sau transmise ușor. Nu este un format specific proiectelor AI, este doar un mod rapid si destul de eficient de a stoca o structura complexa de date intr-un fisier.

```
model_name=os.path.join(model_dir, 'celebrity_faces_sec.pkl')
with open(model_name, 'wb') as f:
    pickle.dump(embeddings_dict, f)
```

Incarcarea bazei de date

Urmatoarele rulari ale programului pot incarca baza de date astfel incat sa nu mai fie necesara extragerea amprentelor din toate imaginile.

```
model_name=os.path.join(model_dir, 'celebrity_faces.pkl')
with open(model_name, 'rb') as f:
    embeddings_dict = pickle.load(f)
```

Vizualizarea clusterizarii embedding-urilor

Daca modelul functioneaza bine, amprentele imaginilor aceluiasi personaj se strang aproape una de alta si se distanteaza de cele ale altor imagini. In spatiul de 512 dimensiuni, daca modelul este corect, vom avea un numar de clustere izolate, cate unul pentru fiecare persoana.

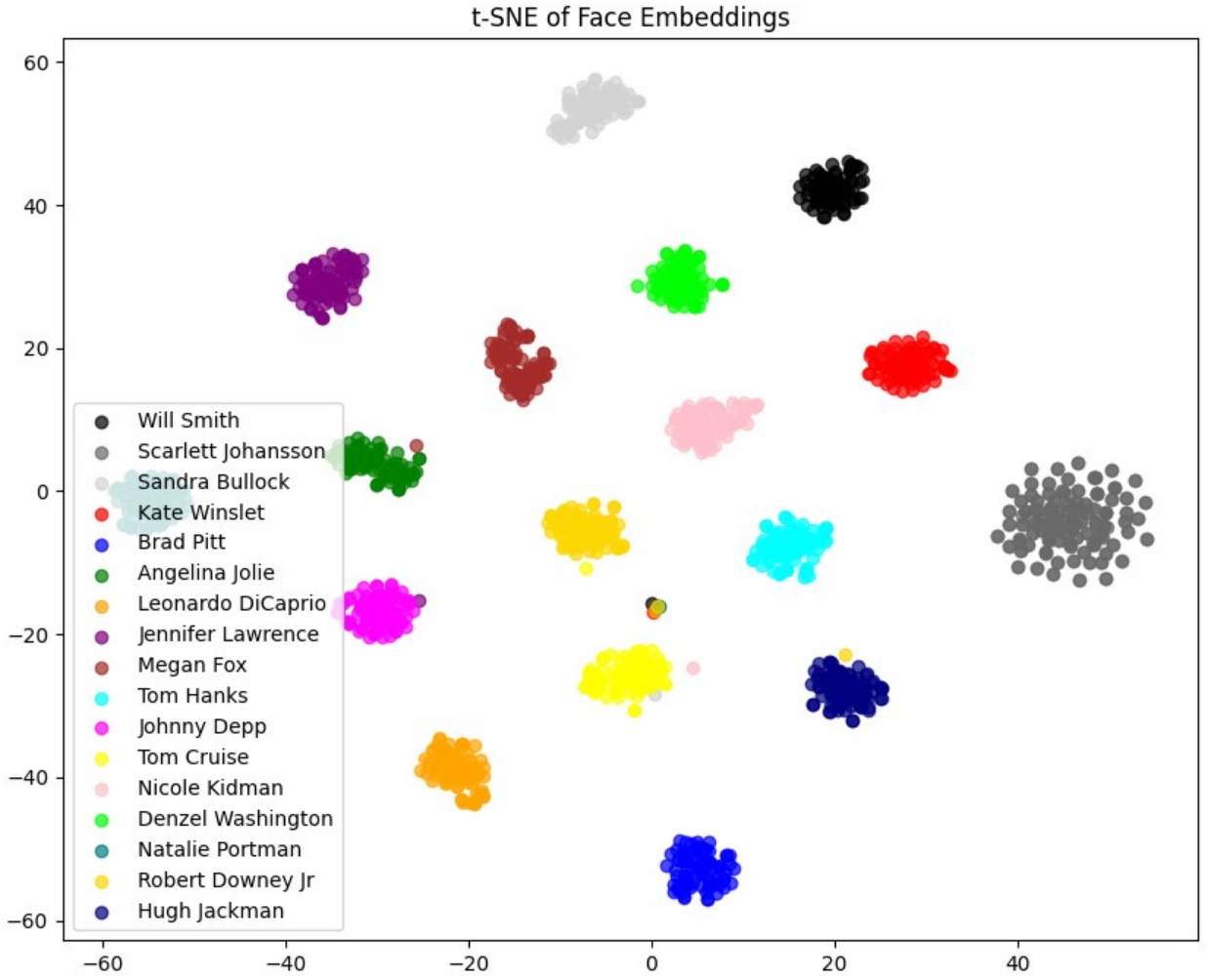
Nu putem vizualiza spatii de 512 dimensiuni - folosim tehnici de reducere a dimensiunalitatii. Acestea reduc dimensiunile de la 512 la 2, cu incercarea de pastrare a structurii. Folosim TSNE si UMAP (doua metode de reducere a dimensionalitatii).

Folosim o paleta cromatica specifica pentru a ne asigura ca atat TSME cat si UMAP mentin aceleasi clase.

```
custom_colors = [
    'black',
    'dimgray',
    'lightgray',
    'red',
    'blue',
    'green',
    'orange',
    'purple',
    'brown',
    'cyan',
    'magenta',
    'yellow',
    'pink',
    'lime',
    'teal',
    'gold',
    'navy'
]

all_embeddings = []
labels = []

for label, vectors in embeddings_dict.items():
    for vec in vectors:
        all_embeddings.append(vec)
        labels.append(label)

all_embeddings = np.array(all_embeddings)
```

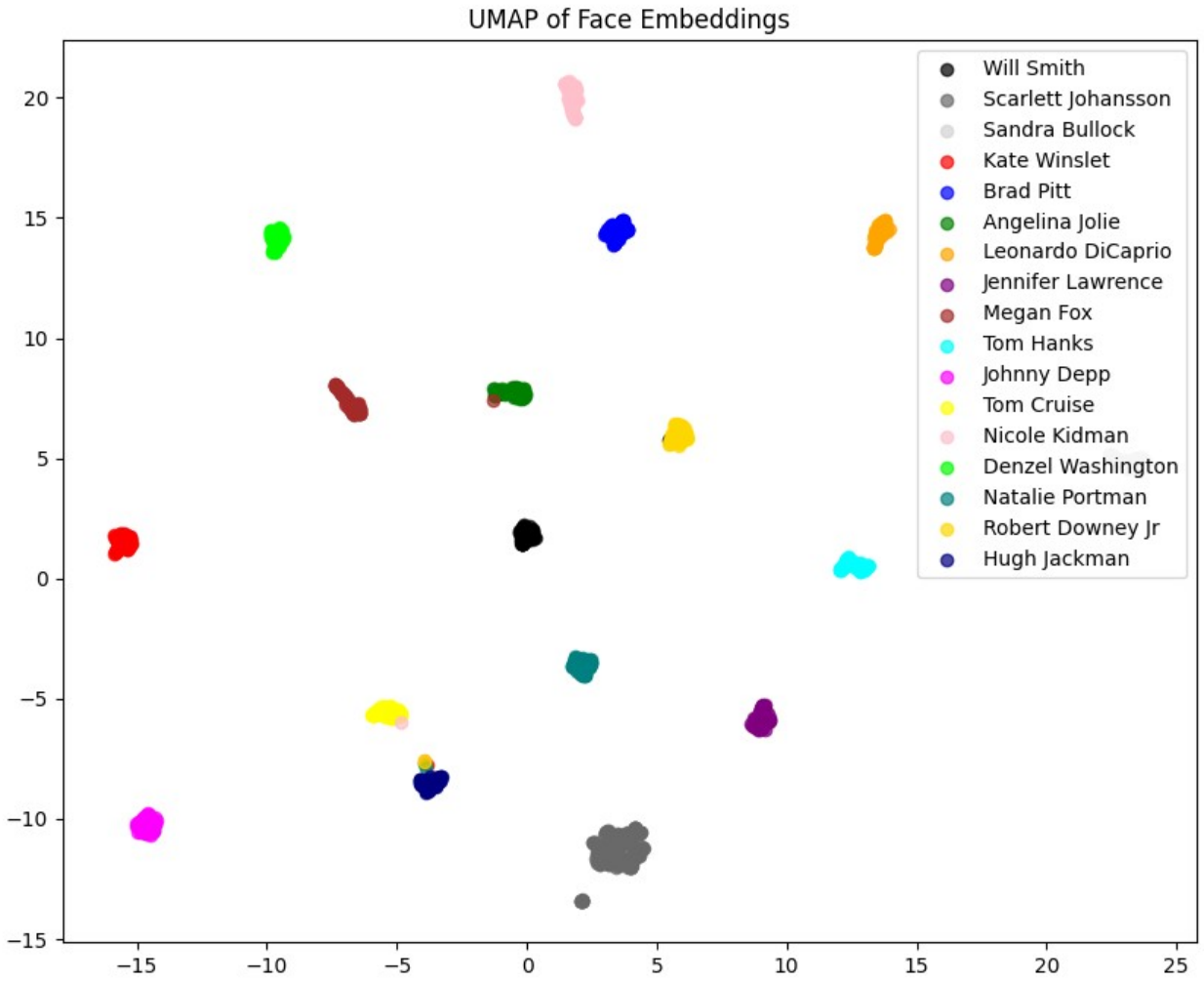
TSNE

```
colors = custom_colors
plt.figure(figsize=(10, 8))
unique_labels = list(set(labels))

tsne = TSNE(n_components=2, random_state=42, perplexity=30)
tsne_result = tsne.fit_transform(all_embeddings)

for idx, label in enumerate(unique_labels):
    points = tsne_result[np.array(labels) == label]
    plt.scatter(points[:, 0], points[:, 1], label=label, alpha=0.7,
color=colors[idx % len(colors)])

plt.title("t-SNE")
plt.legend()
plt.show()
```



UMAP

```
umap_model = umap.UMAP(n_components=2)

umap_result = umap_model.fit_transform(all_embeddings)

colors = custom_colors
plt.figure(figsize=(10, 8))
unique_labels = list(set(labels))

for idx, label in enumerate(unique_labels):
    points = umap_result[np.array(labels) == label]
    plt.scatter(points[:, 0], points[:, 1], label=label, alpha=0.7,
color=colors[idx % len(colors)])

plt.title("UMAP of Face Embeddings")
plt.legend()
plt.show()
```



Utilizarea bazei de date

Recunoasterea unei imagini noi implica extragerea vectorul de embedding din aceasta si compararea sa cu toti vectorii aflati in baza de date.

Similaritatea se calculeaza folosind distanta cosinus (cosinusul unghiului dintre cei doi hipervectori). Daca valoarea este 1, unghiul este 0, cei doi vectori sunt identici, fetele sunt identice. Daca valoarea este 0, unghiul dintre cei doi vectori este 90 de grade - nu exista similaritate.

```
print("Incarcati o imagine de test")
uploaded_test_image = files.upload()  # Permite incarcarea interactiva
a unei imagini in notebook

uploaded_test_image_path = list(uploaded_test_image.keys())[0]

img = cv2.imread(uploaded_test_image_path)

new_width = 450
height, width = img.shape[:2]

aspect_ratio = height / width
new_height = int(new_width * aspect_ratio)

img_small = cv2.resize(img, (new_width, new_height))

cv2_imshow(img_small)
faces = model.get(img)
if len(faces) > 0:
   embedding_to_search = faces[0].embedding

similarities = cosine_similarity([embedding_to_search],
all_embeddings)[0]

# Get the best match
best_idx = np.argmax(similarities)
best_label = labels[best_idx]
best_score = similarities[best_idx]

print(f"\n\nBest match: {best_label} (score: {best_score:.4f})")
```

```
Incarcati o imagine de test
```

```
<IPython.core.display.HTML object>
```

```
Saving Angelina_3.jpg to Angelina_3 (1).jpg
```

```
Best match: Angelina Jolie (score: 0.6811)
```