Лабораторна робота №4

Розпізнавання людини на фото з використанням бібліотеки Dlib

**Мета:** Навчитися розпізнавати обличчя на фото використовуючи навчені нейронні мережі

**ТЕОРЕТИЧНІ ВІДОМОСТІ:**

**Що означає розпізнавання обличчя?**

Розпізнавання обличчя – це процес ідентифікації або перевірки особи людини на основі її рис обличчя. Використовується для перевірки особи, пошуку зниклих безвісти, ідентифікації злочинців тощо. Розпізнавання облич використовує базу даних відомих облич і порівнює їх із невідомим обличчям, щоб знайти відповідність і передбачити особу людини. Існують різні алгоритми, які використовуються для розпізнавання обличчя, включаючи власні обличчя, локальні двійкові шаблони та глибоке навчання.

Розпізнавання обличчя зазвичай поділяється на 4 кроки, які є такими:

1) Пошук обличчя

Першим кроком у конвеєрі розпізнавання облич є виявлення всіх облич на зображенні. Це можна зробити за допомогою детектора обличчя, наприклад каскадів Хаара, гістограми орієнтованих градієнтів (HOG) або детектора обличчя на основі глибокого навчання;

2) Вирівнювання обличчя за допомогою орієнтирів обличчя (необов’язково)

Другим кроком у конвеєрі є вирівнювання або нормалізація обличчя за допомогою орієнтирів обличчя. Цей крок необов’язковий, але він може підвищити точність системи розпізнавання обличчя;

3) Кодування обличчя

На цьому кроці ми передаємо зображення обличчя моделі та витягуємо риси обличчя;

4) Розпізнавання обличчя

Це останній крок у процесі, на якому ми порівнюємо витягнуті риси обличчя з базою даних відомих рис обличчя та намагаємося знайти відповідність. Це можна зробити за допомогою різноманітних алгоритмів, у тому числі K-найближчих сусідів (KNN), опорних векторних машин (SVM), Random Forest тощо.

**Алгоритми розпізнавання облич на основі глибокого навчання та як вони працюють?**

Алгоритми глибокого навчання для розпізнавання облич базуються на згорткових нейронних мережах (CNN). Ці типи нейронних мереж відомі як сіамські мережі (рис. 1).

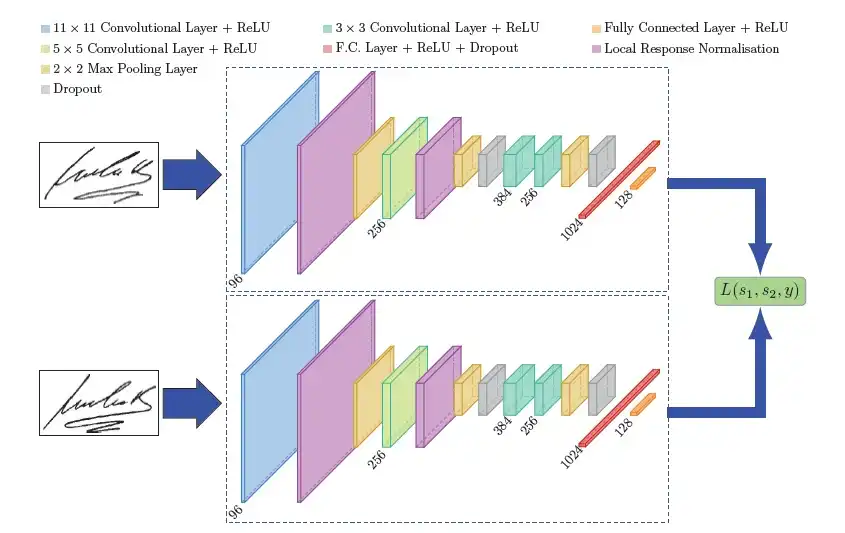


Рисунок 1 - Приклад архітектури сіамської мережі

По суті, сіамська мережа — це тип нейронної мережі, яка містить дві або більше ідентичних підмереж. Кожна підмережа має однакові ваги та параметри.

Деякі найсучасніші архітектури сіамської мережі, які використовуються для розпізнавання облич, включають:

1) VGG-Face   
2) Dlib’s ResNet-based face recognition model   
3) FaceNet   
4) OpenFace   
5) Facebook DeepFace   
6) DeepID   
7) ArcFace   
8) SFace

**Як відбувається навчання мережі?**

Модель навчається, надаючи їй вхідні зображення дерева:

1) “Якірне” (англ. **anchor**) зображення, яке є образом даної людини;

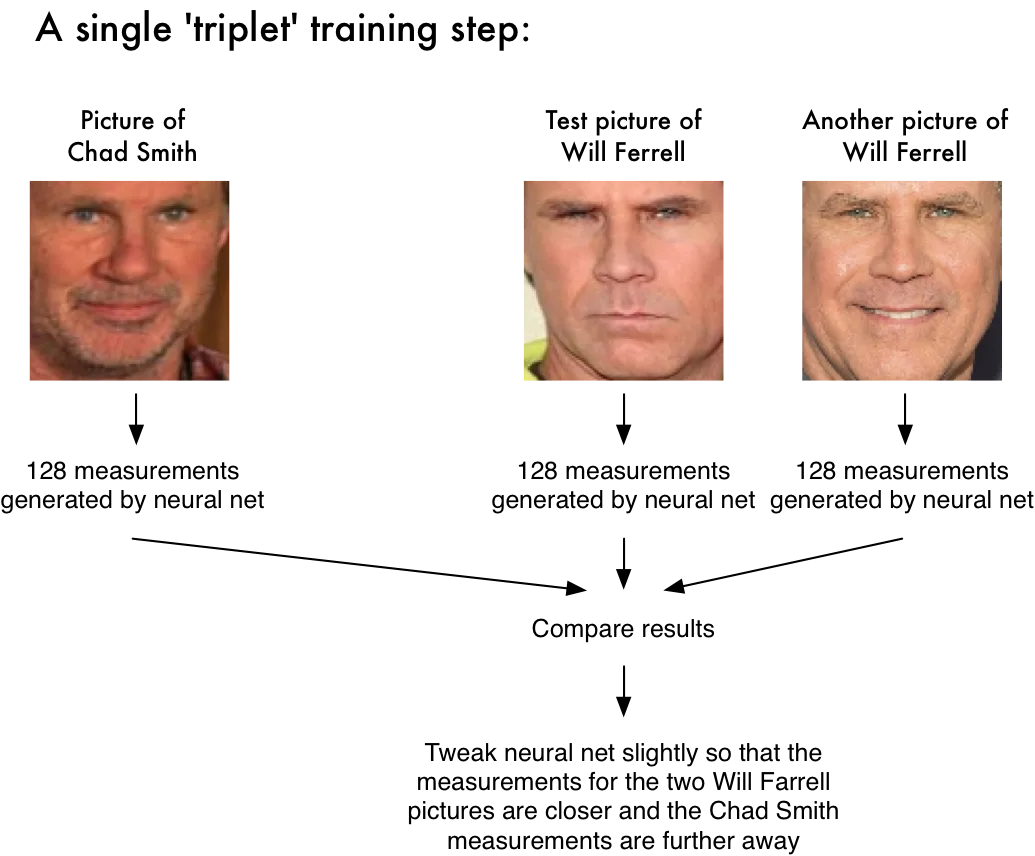
2) «Позитивне» (англ. positive) зображення, яке є зображенням тієї ж особи, що й основне зображення;

3) «Негативний» (англ. negative) образ, який є образом іншої людини.

Потім мережа видає 128-d вкладення для кожного зображення.

Нейронна мережа навчається за допомогою функції втрати, яка штрафує її, якщо відстань між двома вбудовуваннями подібної людини («якорні» та «позитивні» зображення) велика або якщо відстань між двома вкладеннями двох різних людей («якорні» та «негативні» зображення) невелика.

Роблячи це, мережа навчиться генерувати вкладення, які розташовані ближче одне до одного для зображень однієї людини та далі одне від одного для зображень різних людей (рис. 2).

Рисунок 2 - Приклад процесу навчання сіамської мережі

Після того, як мережу буде навчено, її можна буде використовувати для створення вставок для нових зображень. Потім ці вкладення можна використовувати для навчання класифікатора розпізнаванню облич. Класифікатором може бути будь-який алгоритм машинного навчання, наприклад K-Nearest Neighbors (KNN), опорна векторна машина (SVM), Random Forest тощо.

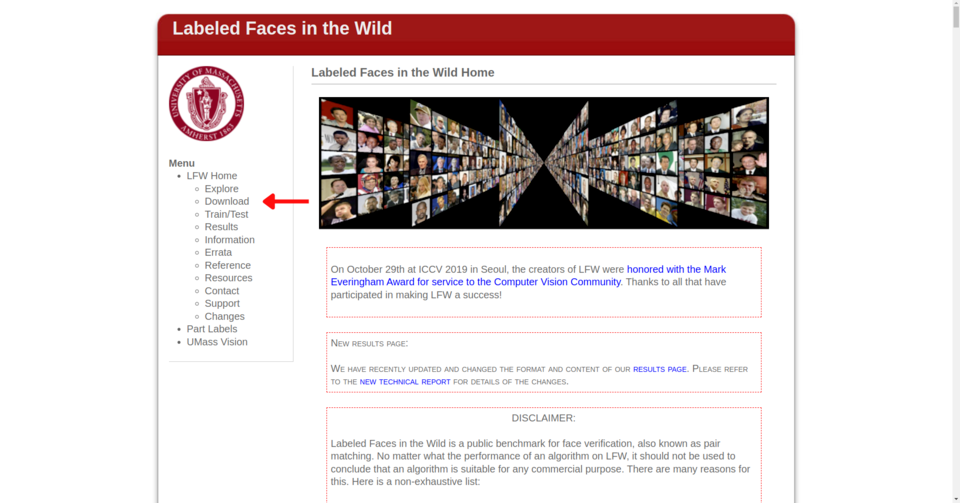
**Розпізнавання обличчя за допомогою OpenCV, Dlib, Numpy**

1) встановлення необхідних бібліотек

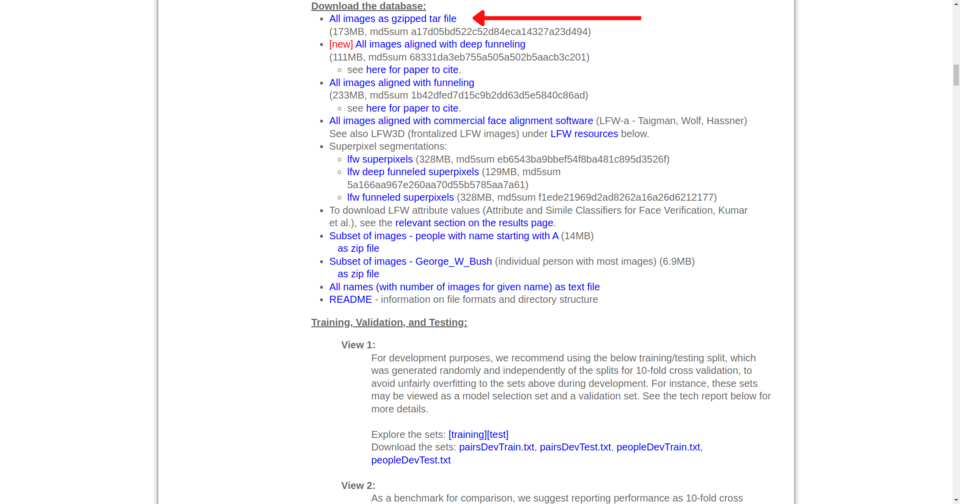
2) завантаження датасету

- перейти за посиланням <http://vis-www.cs.umass.edu/lfw/>

- натиснути download (рис. 3)

Рисунок 3 — Завантажування датасету

- обрати “**All images as gzipped tar file**” (рис. 4)

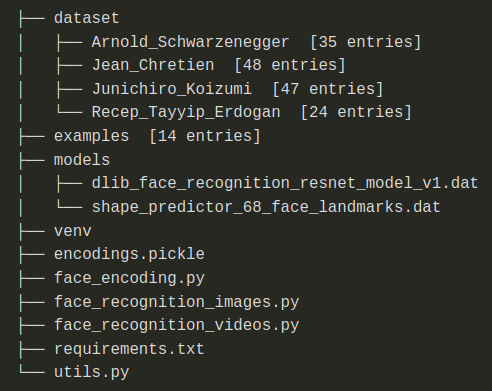
Рисунок 4 — завантажування датасету в gzip

- розархівувати архів

3) обираємо людину, обличчя якої будемо розпізнавати. Надалі будемо використовувати датасет із “залізним Арні” (рис. 5)

Рисунок 5 — Арнольд **Шварценеггер**

4) приклад структури проекту зображена на рис. 6

Рисунок 6 — структура проекту

- dataset/: містить “якорні” зображення людей, яких ми хочемо впізнати;

- examples/: містить приклади зображень, які ми будемо використовувати для тестування нашої системи розпізнавання обличчя;

- models/: містить попередньо підготовлені моделі, які ми будемо використовувати для генерації вбудовування обличчя;

- encodings.pickle: файл із вкладеннями облич людей у нашому наборі даних. Ці вкладення будуть згенеровано сценарієм face\_encoding.py;

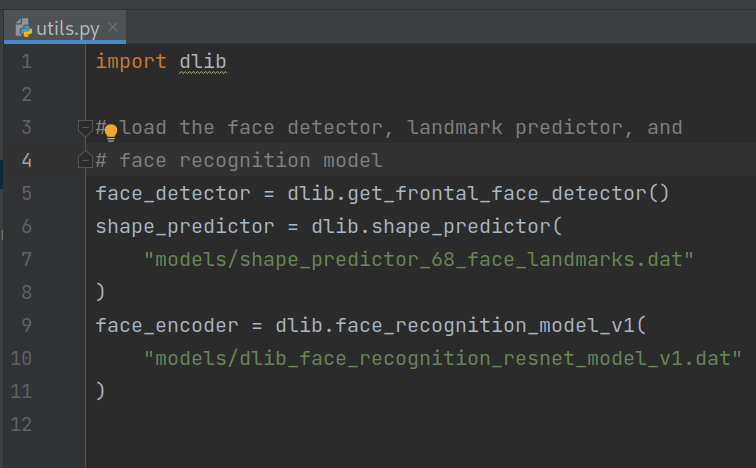
- face\_encoding.py: цей файл містить код для генерації вкладень облич людей із нашого набору даних;

- face\_recognition\_images.py: цей файл містить код для виконання розпізнавання обличчя на зображеннях;

- utils.py: Щоб зробити наш код більш організованим, ми розмістимо всі службові функції в цьому файлі.

5) Кодування обличчя за допомогою Dlib і Deep Learning

Перше, що нам потрібно зробити, це завантажити зображення людей, яких ми хочемо впізнати. Потім ми витягнемо області обличчя із зображень і згенеруємо вкладення для кожної області обличчя (рис. 7).

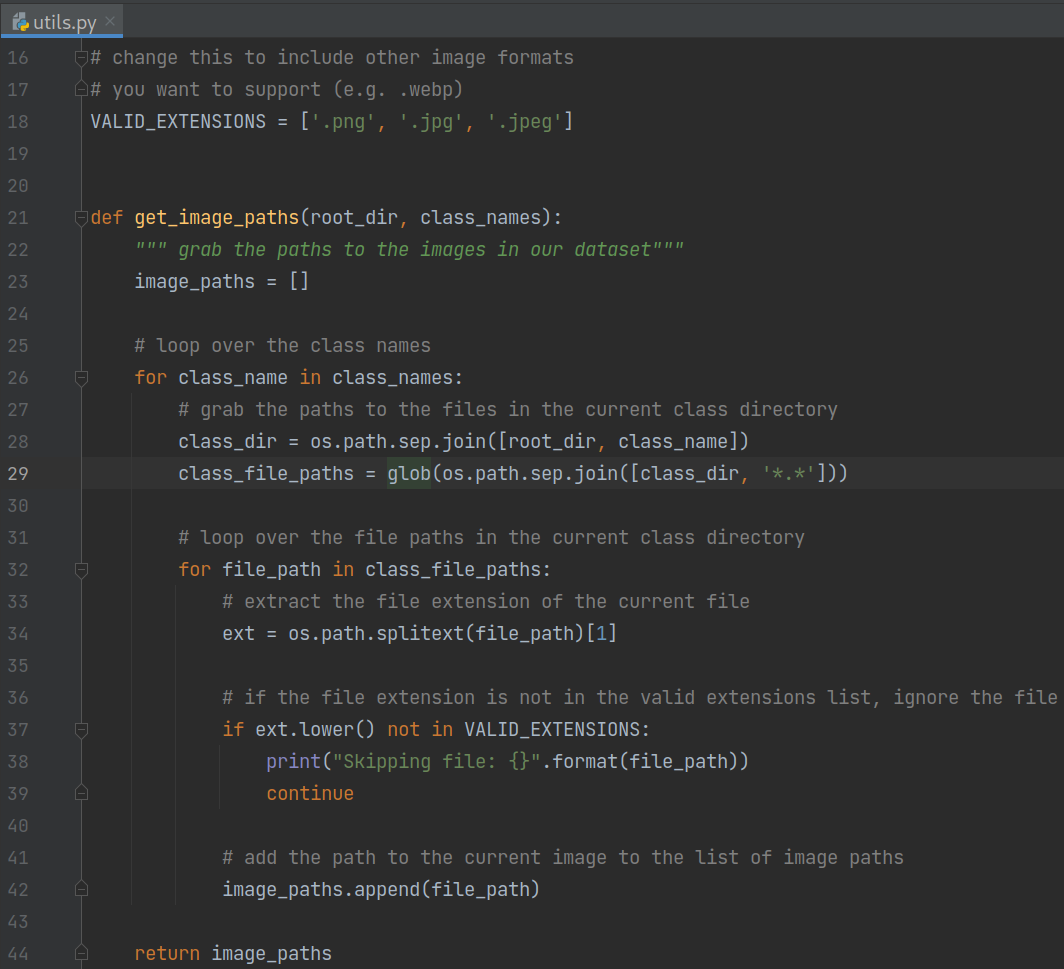
Рисунок 7 — Використання dlib в utils.py

Детектор обличчя (face\_detector) використовується для визначення областей обличчя на вхідному зображенні.

Прогноз орієнтирів (shape\_predictor) використовується для локалізації орієнтирів обличчя в області обличчя. Нам знадобляться орієнтири обличчя, щоб створити вбудовування обличчя.

Модель кодера обличчя (face\_encoder) використовується для генерації вбудовування обличчя.

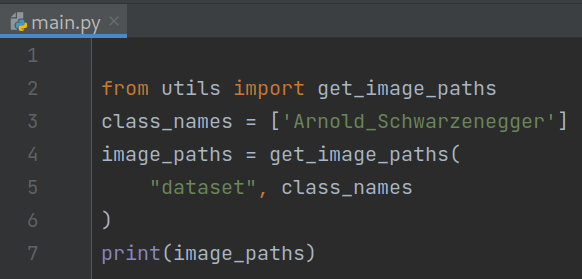
6) Далі ми отримаємо шляхи до зображень у нашому наборі даних. Cтворимо для цього допоміжну функцію (рис. 8)

Рисунок 8 — Створення допоміжної функції

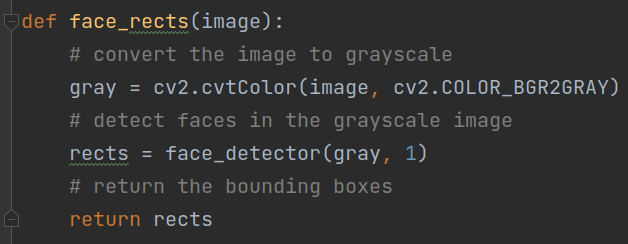
Кореневий\_каталог — це шлях до набору даних. Class\_names — це список імен кожної особи, яку ми хочемо розпізнати.

Функція glob() повертає список шляхів до файлів у поточному каталозі класу. Ми проходимо шляхи до файлів і витягуємо розширення поточного файлу. Якщо розширення файлу недійсне (не зображення), ми пропустимо його та перейдемо до наступного.

Якщо розширення файлу дійсне (зображення), ми додаємо шлях до поточного зображення до списку image\_paths. Приклад використання зображено на рис. 9.

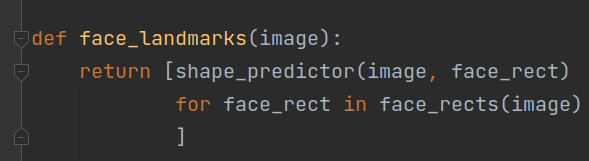
Рисунок 9 — Приклад використання get\_image\_paths

7) Далі ми створимо допоміжну функцію для вилучення областей обличчя із зображень. Для цього ми будемо використовувати детектор обличчя Dlib (рис. 10).

Рисунок 10 — Вилучення облич з зображень

Функція face\_rects() приймає зображення як вхідні дані та застосовує до нього наш детектор обличчя. Він повертає обмежувальні рамки областей обличчя на зображенні.

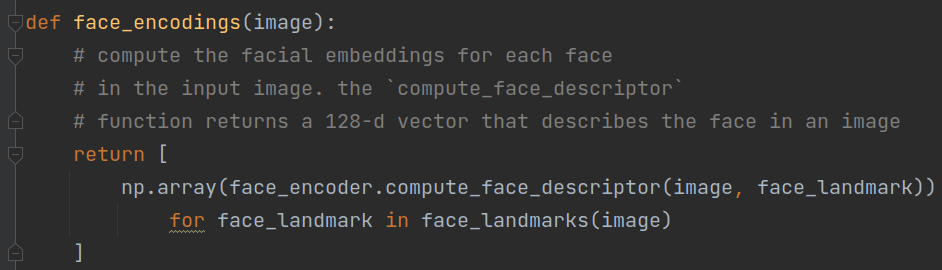
8) Тепер, коли у нас є обмежувальні рамки, ми можемо використовувати їх, щоб застосувати предиктор орієнтирів і отримати орієнтири обличчя (рис. 11).

Рисунок 11 — Використання предиктору орієнтирів

Ця функція виконує цикл по обмежувальним рамкам кожної області обличчя. Для кожної області обличчя він застосовує предиктор орієнтирів.

Він повертає список, що містить орієнтири обличчя для кожної області обличчя.

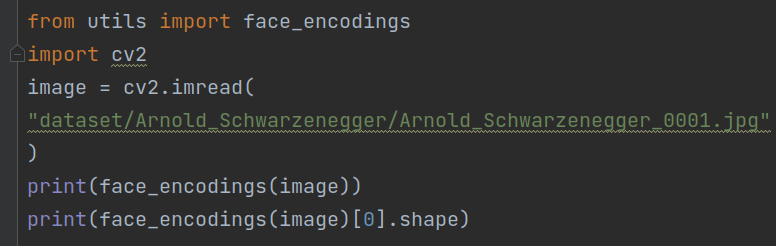
9) А тепер останній крок — створити вкладення обличчя для кожної області обличчя. Знову ж таки, ми створимо для цього допоміжну функцію (рис. 12).

Рисунок 12 — Кодування облич

Тож тут функція face\_encodings() приймає зображення як вхідні дані та циклічно переглядає орієнтири обличчя кожної області обличчя.

Для кожної області обличчя він застосовує кодер обличчя та генерує вкладення обличчя. Він повертає список, що містить вкладення обличчя кожної області обличчя.

Давайте перевіримо нашу функцію face\_encodings() (рис. 13).

Рисунок 13 — Перевірка роботи функції face\_encodings

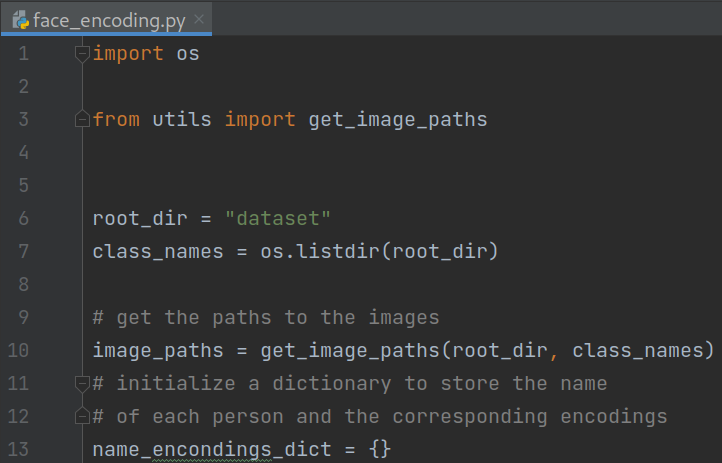
Як ви бачите, функція face\_encodings() повертає список із вкладеннями облич кожної області обличчя на зображенні. Кожне вкладення грані є 128-d масивом Numpy.

10) Починаємо будувати систему розпізнавання обличчя

Ми почнемо з генерації вкладення обличчя для кожного зображення в нашому наборі даних. Потім ми збережемо вкладення облич у словнику. Ключами словника будуть імена кожної людини в нашому наборі даних, а значеннями буде список вкладень облич для кожного зображення людини.

В кінці, ми збережемо це на диск.

Відкрийте файл face\_encoding.py і додайте такий код:

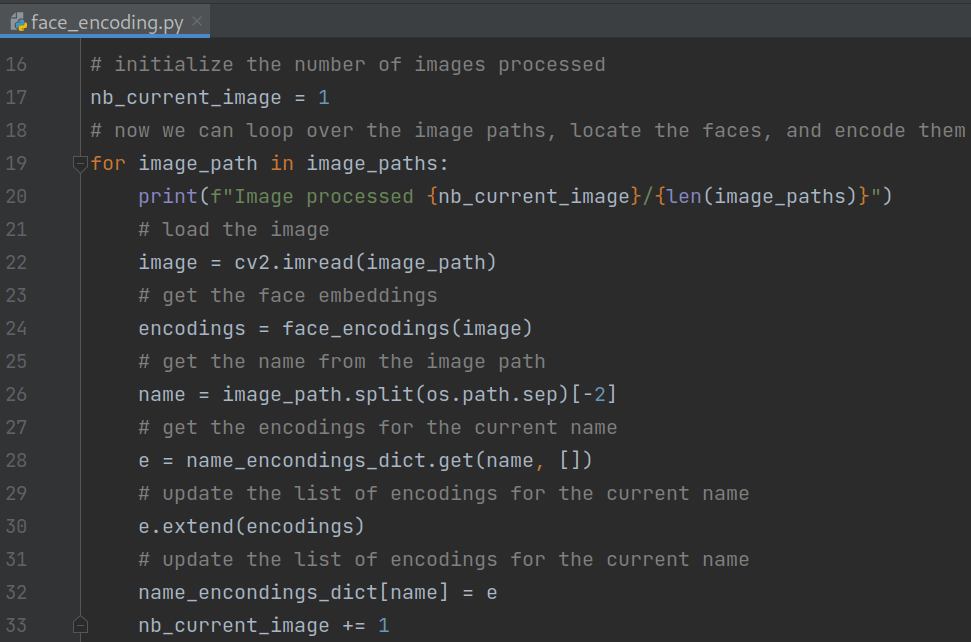
Рисунок 14 — Початок реалізації face\_encoding.py

Ми будемо використовувати модуль pickle, щоб зберегти наш словник у файлі, щоб ми могли використовувати його пізніше, коли будемо тестувати нашу систему розпізнавання обличчя.

Тут ми використовуємо функцію get\_image\_paths(), щоб отримати шляхи до зображень у нашому наборі даних.

Ми також ініціалізуємо словник для зберігання міток (імен кожної особи) і відповідних вставок обличчя.

Тепер ми пройдемося по контурах зображень і згенеруємо вкладення обличчя для кожного зображення.

Рисунок 15 — Генерація вкладень для кожного обличчя

Отже, ми переглядаємо шляхи зображень. Для кожного шляху ми завантажуємо зображення за допомогою OpenCV і генеруємо вкладення обличчя за допомогою функції face\_encodings().

Пам’ятайте, що функція face\_encodings() застосовує детектор обличчя, щоб отримати обмежувальні рамки областей обличчя на зображенні, а потім застосовує предиктор орієнтирів, щоб отримати орієнтири обличчя. Нарешті, він застосовує кодер обличчя, використовуючи орієнтири обличчя для створення вставок обличчя.

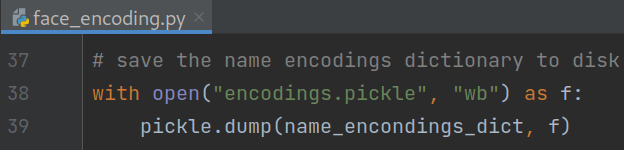
Ми також отримуємо ім'я людини зі шляху зображення. Шлях до зображення має вигляд dataset/Arnold\_Schwarzenegger/Arnold\_Schwarzenegger\_0001.jpg. Отже, ми можемо отримати ім’я особи, розділивши шлях за допомогою роздільника каталогу (os.path.sep) і отримавши передостанній елемент списку.

Далі ми намагаємося отримати зі словника список вкладень облич для поточної людини. Якщо імені людини немає в словнику, ми ініціалізуємо порожній список для вкладення.

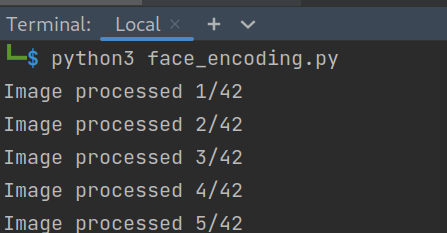
Потім ми розширюємо список вбудовувань за допомогою особливостей обличчя, згенерованого для поточного зображення. Нарешті, ми оновлюємо словник новим списком особливостей облич для поточної особи.

Після циклу ми матимемо словник з іменами кожної людини та відповідним списком вкладень облич для кожного зображення людини.

Останнім кроком є збереження словника у файл за допомогою модуля pickle.

Рисунок 16 — Збереження вкладень на диск

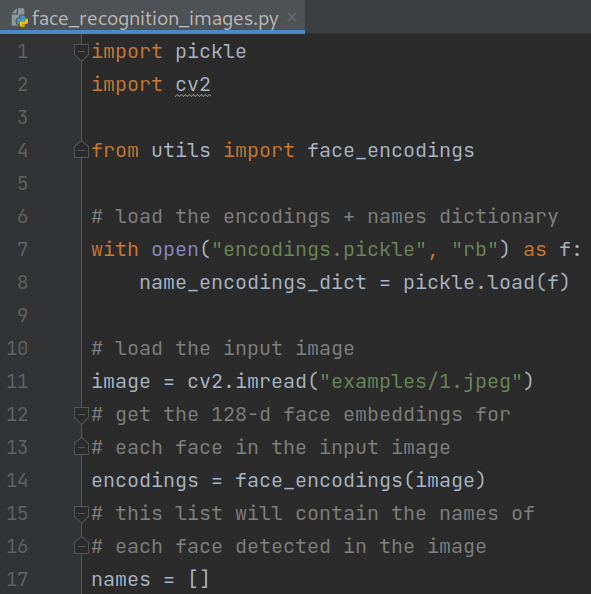
Тепер ми можемо запустити сценарій face\_encoding.py, щоб створити вкладення обличчя для кожного зображення в нашому наборі даних (рис. 17).

Рисунок 17 — Приклад запуску

Після завершення роботи сценарію ви повинні побачити файл під назвою encodings.pickle у каталозі проекту. Цей файл містить словник з іменами кожної людини та відповідний список вкладень облич для кожного зображення людини.

11) Розпізнавання обличчя в зображеннях

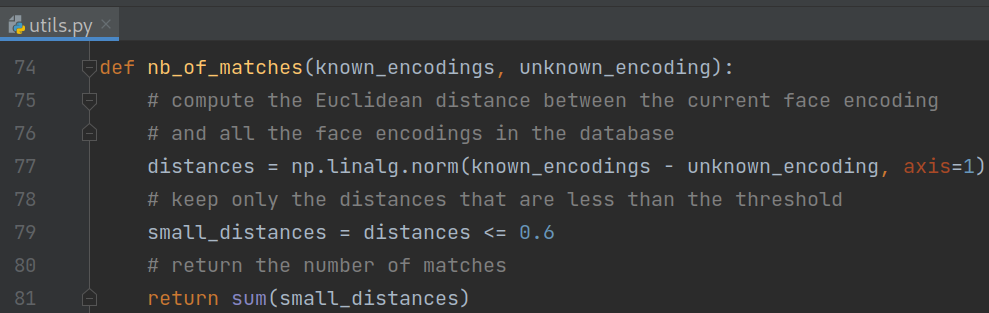
Тепер, коли у нас є вкладення обличчя, ми можемо почати розпізнавати обличчя на зображеннях (рис. 18).

Рисунок 18 — Завантаження вкладок облич

Спочатку ми завантажуємо словник, що містить імена кожної особи та відповідний список вкладок облич. Далі ми завантажуємо вхідне зображення за допомогою OpenCV і генеруємо вбудовування обличчя для кожного обличчя на зображенні за допомогою функції face\_encodings().

Наступним кроком є перегляд вкладених облич і порівняння їх із вкладеннями облич кожної людини в нашому наборі даних. Нам знадобиться інша допоміжна функція, яка обчислює відстань між поточним вкладенням обличчя та вкладенням обличчя кожної людини в нашому наборі даних.

Якщо відстань між поточним вкладенням обличчям і вкладенням обличчя у базі даних менша за порогове значення, обличчя вважається збігом, тому ми збільшуємо «кількість збігів» для поточної особи на 1. Простіше зрозуміти цю логіку, подивившись на код. Відкрийте файл utils.py і додайте код, що наведений на рис. 15.

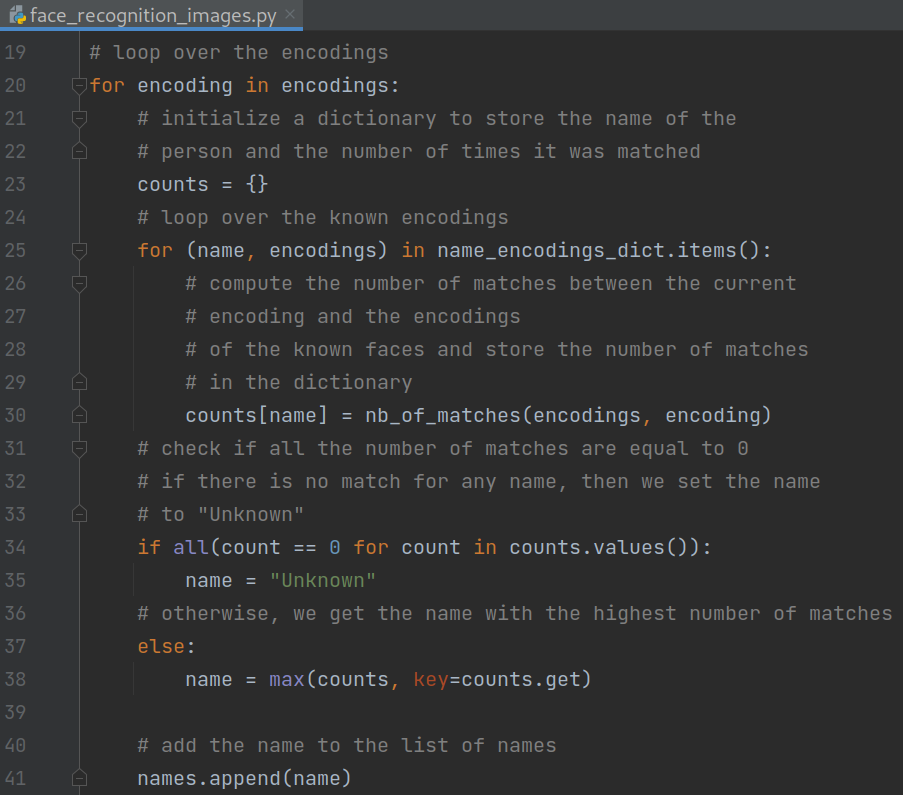
Рисунок 15 — Розрахунок кількость збігів для поточного кодування обличчя

Функція nb\_of\_matches() приймає два аргументи: список вкладених облич кожної людини в нашому наборі даних (known\_encodings) і поточне обличчя, вкладене у вхідне зображення (unknown\_encoding).

Він обчислює евклідову відстань між поточним кодуванням обличчя та всіма кодуваннями обличчя в базі даних. Потім він зберігає лише відстані, менші за порогове значення (особа вважається збігом, якщо відстань менша за порогове значення).

Нарешті, він підсумовує, скільки разів відстань менша за порогове значення, щоб отримати кількість збігів для поточного обличчя.

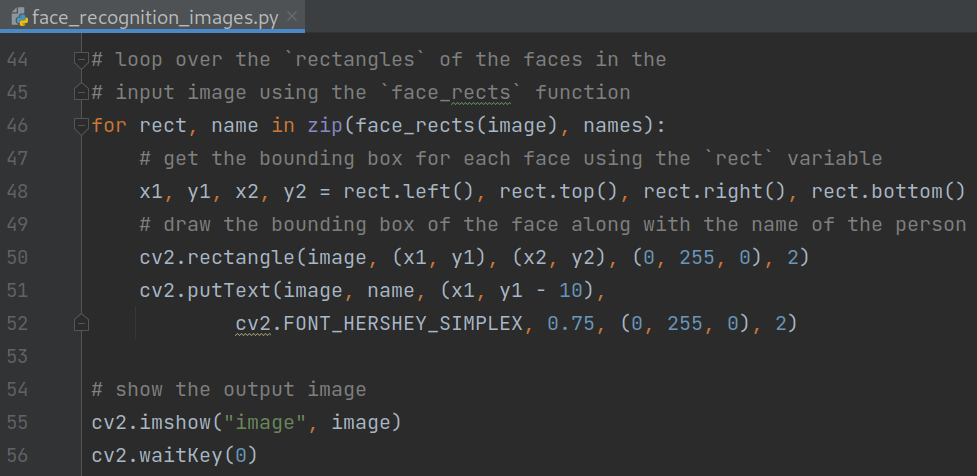
Тепер ми можемо прокрутити вкладення кожного обличчя у вхідне зображення та використати функцію nb\_of\_matches(), щоб отримати кількість збігів для кожного обличчя (рис. 16).

Рисунок 16 — Пошук обличчя у базі

Отже, спочатку ми ініціалізуємо словник, щоб відстежувати кількість збігів для кожної особи. Потім, використовуючи наш словник відомих кодувань, ми використовуємо функцію nb\_of\_matches(), щоб отримати кількість збігів для поточного кодування обличчя та зберегти його в словнику counts.

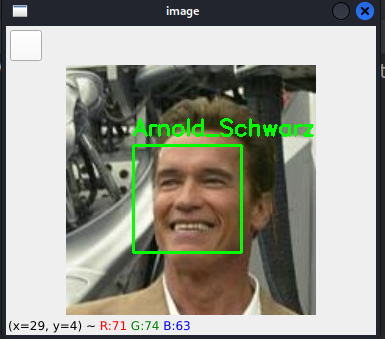
Нарешті, ми використовуємо словник підрахунків, щоб отримати ім’я людини з найбільшою кількістю збігів. Якщо для жодної людини немає збігів (усі збіги дорівнюють 0), ми встановлюємо ім’я на «Невідомо». Потім ми додаємо ім’я до списку імен.

Останній крок — намалювати прямокутник навколо кожного обличчя на вхідному зображенні та написати ім’я людини поверх прямокутника (рис. 17).

Рисунок 17 — Додавання імен до зображень

Допоміжна функція face\_rects() повертає список прямокутників для кожного обличчя у вхідному зображенні. Ці прямокутники надаються бібліотекою Dlib. Ми проходимо петлю по прямокутникам і отримуємо обмежувальну рамку для кожної грані. Потім ми малюємо прямокутник навколо кожного обличчя та пишемо ім’я людини поверх прямокутника.

Нарешті, ми показуємо вихідне зображення за допомогою OpenCV. Тепер настав час перевірити наш алгоритм розпізнавання обличчя (рис. 18).

Рисунок 18 — Приклад розпізнавання

**ПІДГОТОВКА ДО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ №4**

Перед виконанням лабораторної роботи рекомендується ознайомитись з відповідним розділом лекційного матеріалу курсу та засвоїти теоретичний матеріал.

**ВИМОГИ ДО ОФОРМЛЕННЯ ЗВІТУ**

Звіт з лабораторної роботи обов’язково повинен містити наступну інформацію:

* назва комп’ютерного практикуму;
* мета роботи;
* відповіді на завдання у текстовому форматі та графічними зображеннями за необхідності.

**Завдання на лабораторну роботу:**

1. Зробити розпізнавання будь-якої “зірки”;
2. Зробити розпізнавання себе.

**Контрольні запитання:**

1. Що таке розпізнавання обличчя?

2. Які кроки для розпізнвання обличчя?

3. Що таке згорткова нейронна мережа?

4. Як відбувається навчання мережі?

5. Що таке dlib?

**Навчальні матеріали та ресурси:**

1. <https://uk.wikipedia.org/wiki/Dlib>

2. [https://dontrepeatyourself.org/post/face-recognition-with-python-dlib-and-deep-learning/#project-structure](https://dontrepeatyourself.org/post/face-recognition-with-python-dlib-and-deep-learning/" \l "project-structure)