Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота № 4

з дисципліни
«Штучний інтелект в задачах обробки зображень»

Виконав:
студент групи ІМ-11
Царик Микола Миколайович

Перевірив:
Нікітін Валерій Андрійович

Київ 2024

Мета: Навчитися розпізнавати обличчя на фото використовуючи навчені нейронні мережі

Завдання:

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

Хід виконання:

Файл utils.py:

1. Спочатку ми імпортуємо необхідні бібліотеки: **dlib** для роботи з обличчями, **glob** для пошуку файлів, **cv2** (OpenCV) для обробки зображень, **numpy** для математичних обчислень, і **os** для роботи з файловою системою.

```
import dlib
from glob import glob
import cv2
import numpy as np
import os
```

2. Ініціалізуємо детектор облич, модель для визначення особливостей обличчя (landmarks), і модель для кодування облич.

```
face_detector = dlib.get_frontal_face_detector()
shape_predictor =
dlib.shape_predictor("models/shape_predictor_68_face_landmarks.dat")
face_encoder =
dlib.face_recognition_model_v1("models/dlib_face_recognition_resnet_model_v1.dat"
)
```

3. Задаємо список валідних розширень зображень, які будемо обробляти.

```
VALID_EXTENSIONS = ['.png', '.jpg', '.jpeg']
```

4. Створимо функцію, яка проходить по директоріям з заданими іменами класів, перевіряє розширення файлів і збирає шляхи до зображень, що відповідають валідним розширенням.

```
def get_image_paths(root_dir, class_names):
    image_paths = []

    for class_name in class_names:
        class_dir = os.path.sep.join([root_dir, class_name])
        class_file_paths = glob(os.path.sep.join([class_dir, '*.*']))

        for file_path in class_file_paths:
            ext = os.path.splitext(file_path)[1]

            if ext.lower() not in VALID_EXTENSIONS:
                print("Файл пропущено: {}".format(file_path))
                continue

            image_paths.append(file_path)

    return image_paths
```

5. Напишемо функції для виявлення облич, отримання landmarks та визначення кодувань облич на зображенні.

```
def face_rects(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    rects = face_detector(gray, 1)
    return rects

def face_landmarks(image):
    return [shape_predictor(image, face_rect) for face_rect in face_rects(image)]

def face_encodings(image):
    return [np.array(face_encoder.compute_face_descriptor(image, face_landmark))
            for face_landmark in face_landmarks(image)]
```

6. Реалізуємо функцію, що порівнює кодування невідомого обличчя з відомими і повертає кількість схожих облич.

```
def nb_of_matches(known_encodings, unknown_encoding):
    distances = np.linalg.norm(known_encodings - unknown_encoding, axis=1)
    small_distances = distances <= 0.6
    return sum(small_distances)
```

Файл face_encoding.py:

1. Завантажуємо модулі **pickle** для серіалізації даних, **cv2** для роботи з зображеннями, та **os** для доступу до файлової системи. Також імпортуємо функції **get_image_paths** і **face_encodings** з модуля **utils**.

```
import pickle
import cv2
import os

from utils import get_image_paths
from utils import face_encodings
```

2. Визначаємо директорію, де зберігаються зображення (**root_dir**), і отримуємо список назв папок у цій директорії, які вважаємо за назви класів.

```
root_dir = "dataset"
class_names = os.listdir(root_dir)
```

3. Використовуємо функцію get_image_paths для збору шляхів до зображень у вказаній директорії. Ініціалізація словника для збереження кодувань Створюємо словник, де ключем буде назва класу (ім'я особи), а значенням — список кодувань облич цієї особи.

```
image_paths = get_image_paths(root_dir, class_names)
name_encondings_dict = {}
```

4. Проходимо по всіх зображеннях, зчитуємо кожне з них, обраховуємо кодування облич, додаємо їх у словник. Додатково виводимо повідомлення про оброблене зображення.

```
nb_current_image = 1
for image_path in image_paths:
    print(f"Зображення оброблене {nb_current_image}/{len(image_paths)}")
    image = cv2.imread(image_path)
    encodings = face_encodings(image)
    name = image_path.split(os.path.sep)[-2]
    e = name_encondings_dict.get(name, [])
    e.extend(encodings)
    name_encondings_dict[name] = e
    nb_current_image += 1
```

6. Використовуємо модуль **pickle** для серіалізації словника з кодуваннями у файл, що дозволяє зберегти обчислені дані для подальшого використання.

```
with open("encodings.pickle", "wb") as f:
    pickle.dump(name_encondings_dict, f)
```

Файл face_recognition_images.py:

1. Імпортуємо модулі **pickle** для роботи з серіалізованими даними, **cv2** для обробки зображень, і **glob** для знаходження файлів за шаблоном. Також імпортуємо функції для роботи з обличчями з модуля **utils**.

```
import pickle
import cv2
import glob
from utils import face_rects, face_encodings, nb_of_matches
```

2. Відкриваємо файл, де збережено серіалізовані кодування облич, і завантажуємо їх у словник **name_encodings_dict**.

```
with open("encodings.pickle", "rb") as f:
    name_encodings_dict = pickle.load(f)
```

3. Використовуємо **glob** для отримання списку зображень з певної директорії.

```
image_directory = "examples/"
image_paths = glob.glob(image_directory + "*.jpg")
```

4. По черзі читаємо кожне зображення, обраховуємо кодування облич на зображенні та визначаємо імена осіб на них.

```
for image_path in image_paths:
    image = cv2.imread(image_path)
    encodings = face_encodings(image)
    names = []

    for encoding in encodings:
        counts = {}
        for (name, known_encodings) in name_encodings_dict.items():
            counts[name] = nb_of_matches(known_encodings, encoding)
        if all(count == 0 for count in counts.values()):
            name = "Unknown"
        else:
            name = max(counts, key=counts.get)
        names.append(name)
```

5. Для кожного обличчя визначаємо координати прямокутника та виводимо ім'я над обличчям на зображенні.

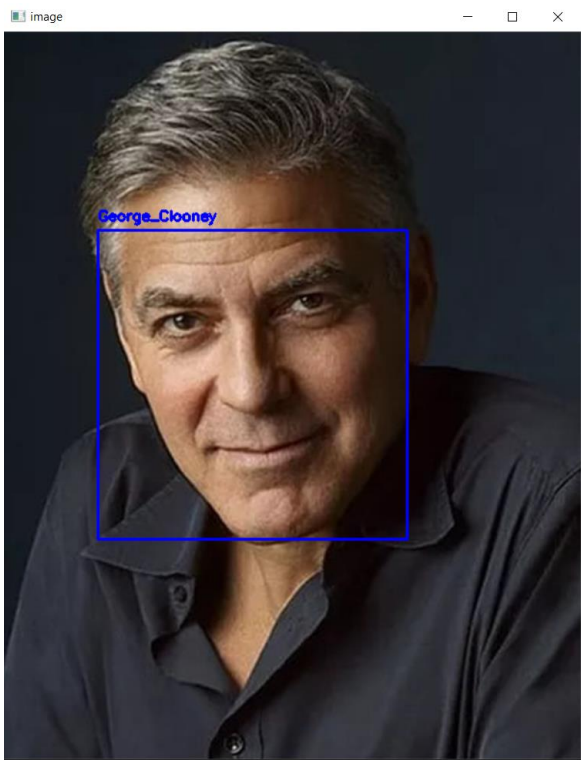
```
for rect, name in zip(face_rects(image), names):
        x1, y1, x2, y2 = rect.left(), rect.top(), rect.right(), rect.bottom()
        cv2.rectangle(image, (x1, y1), (x2, y2), (255, 0, 0), 2)
        cv2.putText(image, name, (x1, y1 - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (255, 0, 0), 2)
```

6. Показуємо оброблені зображення і чекаємо на закриття вікна користувачем. Завершуємо роботу з ресурсами, закриваючи усі вікна OpenCV.

```
    cv2.imshow("image", image)
    cv2.waitKey(0)

cv2.destroyAllWindows()
```

Результат



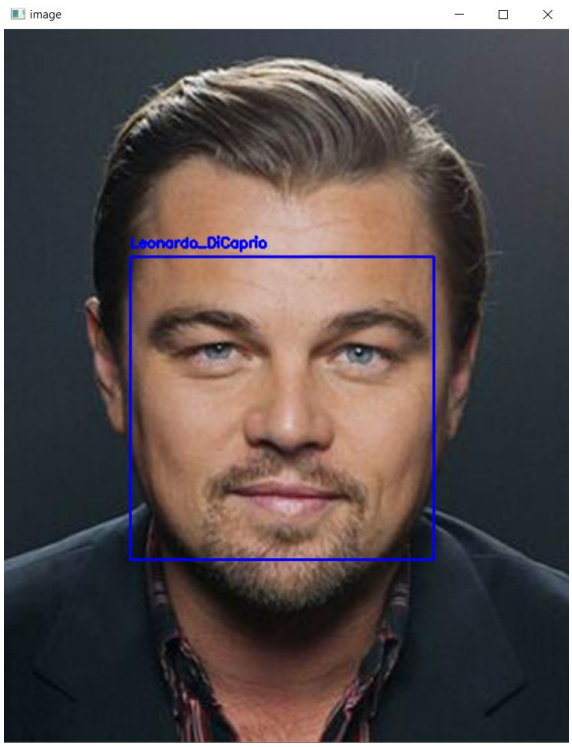


image
Leonardo_DiCaprio

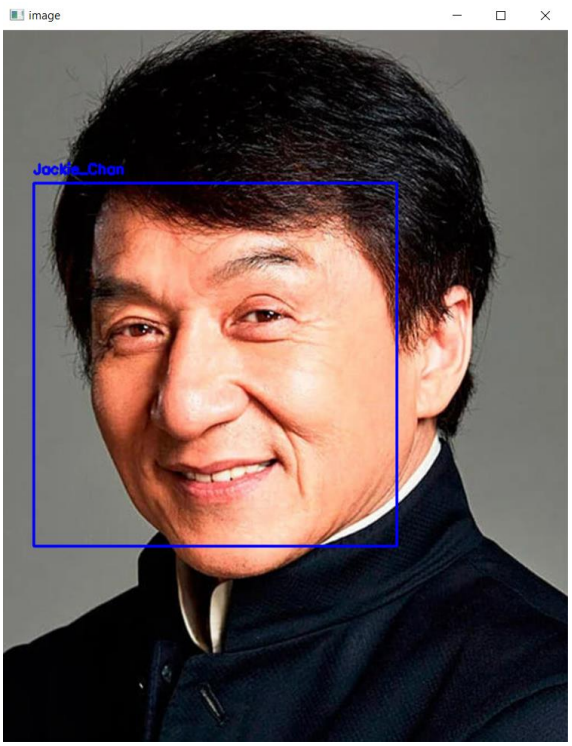


image
Jackie_Chan

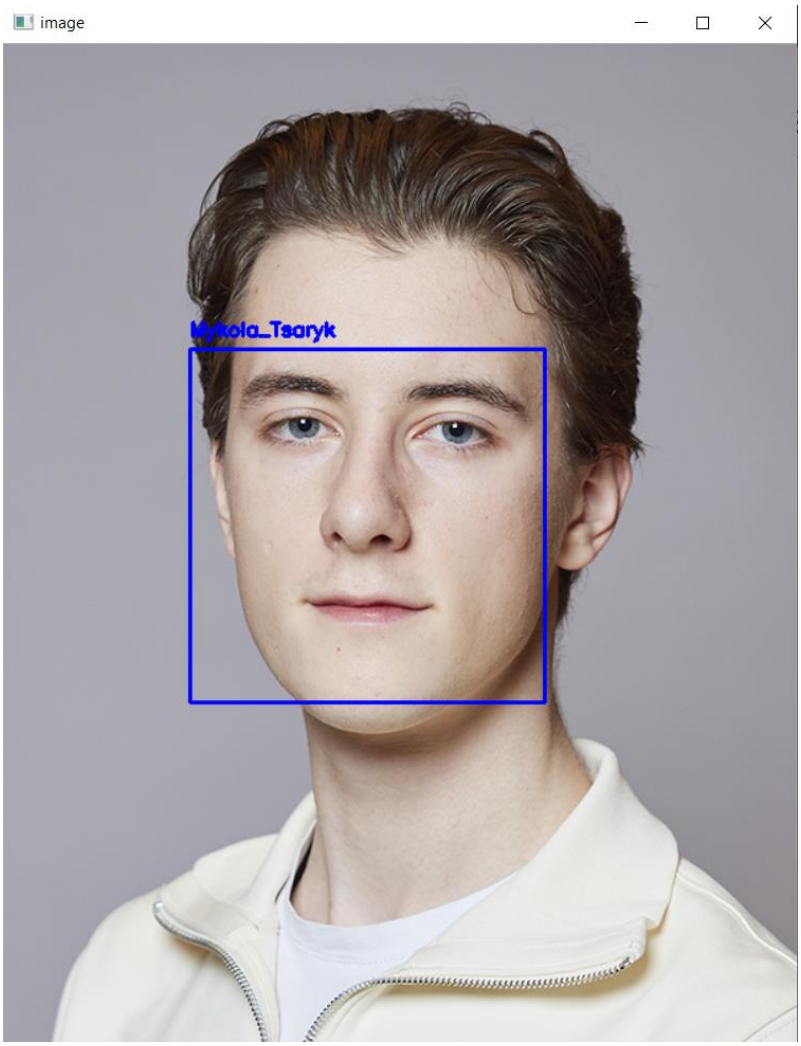


image
Mykola_Tsaryk

Висновки:

У ході виконання цієї лабораторної роботи було зроблено декілька висновків:

Успішно реалізовано завдання з розпізнавання облич "зірки" та себе, що підтвердило здатність технологій глибокого навчання до ідентифікації осіб.

Використання бібліотеки dlib разом з OpenCV дозволило ефективно знаходити обличчя на фотографіях за допомогою переднавчених моделей.

Навчання нейронних мереж виявилося ключовим для точності розпізнавання, особливо в умовах відмінностей освітлення та ракурсів на зображеннях.

Було встановлено, що точність ідентифікації можна покращити, збільшивши набір даних для тренування, що підтверджує важливість об'єму та різноманітності навчальних даних.

Загалом, робота надала не лише технічні навички у сфері обробки зображень, а й глибше розуміння механізмів машинного навчання та комп'ютерного зору, що можуть бути застосовані для вирішення реальних задач ідентифікації людей в різних умовах.

Контрольні запитання:

Що таке розпізнавання обличчя?

Розпізнавання обличчя — це технологічний процес ідентифікації або верифікації особи на основі її обличчя. Системи розпізнавання обличчя аналізують відстань між очима, форма щелепи, розташування носа, тощо, для створення математичної моделі обличчя, яка використовується для порівняння з обличчями у базі даних.

Які кроки для розпізнвання обличчя?

1) Виявлення обличчя: визначаємо чи присутнє обличчя на зображенні або відео.

2) Нормалізація обличчя: вирівнюємо та масштабуємо обличчя для забезпечення консистентності перед аналізом.

3) Витягнення ознак: визначаємо ключові точки або особливості на обличчі, які використовуються для створення унікального профілю обличчя.

4) Порівняння або верифікація: порівнюємо витягнуті ознаки із базою даних для ідентифікації особи або її верифікації.

Що таке згорткова нейронна мережа?

Згорткова нейронна мережа (CNN) — це тип глибинної нейронної мережі, яка часто використовується в задачах комп'ютерного зору. CNN ефективно обробляють візуальні дані завдяки своїй архітектурі, яка містить згорткові шари. Ці типи нейронних мереж відомі як сіамські мережі. По суті, сіамська мережа — це тип нейронної мережі, яка містить дві або більше ідентичних підмереж. Кожна підмережа має однакові ваги та параметри.

Як відбувається навчання мережі?

Навчання нейронної мережі для розпізнавання облич часто виконується за допомогою методу, відомого як триплетний вибір (triplet loss). Ось як це працює:

1. Вхідні дані: Модель отримує три типи зображень:
    - Якірне зображення: Фотографія конкретної особи.
    - Позитивне зображення: Інше зображення тієї ж особи, що і на якірному зображенні.
    - Негативне зображення: Зображення іншої особи.
2. Генерація вкладень: Мережа обробляє кожне зображення, результатом чого є 128-вимірні вкладення для кожного образу.
3. Функція втрат (Triplet Loss): Функція втрат використовується для оцінки якості вкладень. Ця функція штрафує модель, якщо вкладення позитивного зображення занадто далеко від вкладення якірного зображення, або якщо вкладення негативного зображення занадто близько до якірного. Це спонукає модель генерувати близькі вкладення для зображень однієї особи і віддалені вкладення для зображень різних осіб.
4. Оптимізація: За допомогою оптимізаційних алгоритмів, таких як градієнтний спуск, параметри мережі коригуються на основі розрахунків функції втрат.
5. Використання моделі: Після навчання модель може генерувати вкладення для нових зображень. Ці вкладення потім можна використовувати для розпізнавання облич за допомогою класифікаційних алгоритмів, таких як K-Nearest Neighbors (KNN) або опорна векторна машина (SVM).

Що таке dlib?

Dlib — це сучасна бібліотека машинного навчання, написана на мові C++. Вона містить широкий спектр інструментів для розробки складних програм машинного навчання і комп'ютерного зору в областях розпізнавання обличчя, детекції об'єктів і інших. Однією з важливих особливостей dlib є набір готових до використання засобів для розпізнавання облич, включаючи алгоритми для виявлення облич, визначення ключових точок на обличчі, а також для визначення біометричних ознак.