Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

ЛАБОРАТОРНА РОБОТА № 4

з дисципліни «Штучний інтелект»

на тему «Розпізнавання людини на фото з використанням бібліотеки Dlib»

ВИКОНАЛА:

студентка ІІI курсу ФІОТ

групи ІО-26

Гриценко Ксенія Олексіївна

Залікова № 2604

ПЕРЕВІРИВ:

Нікітін Валерій Андрійович

ДАТА ЗДАЧІ:

21.05.2025

Київ – 2025

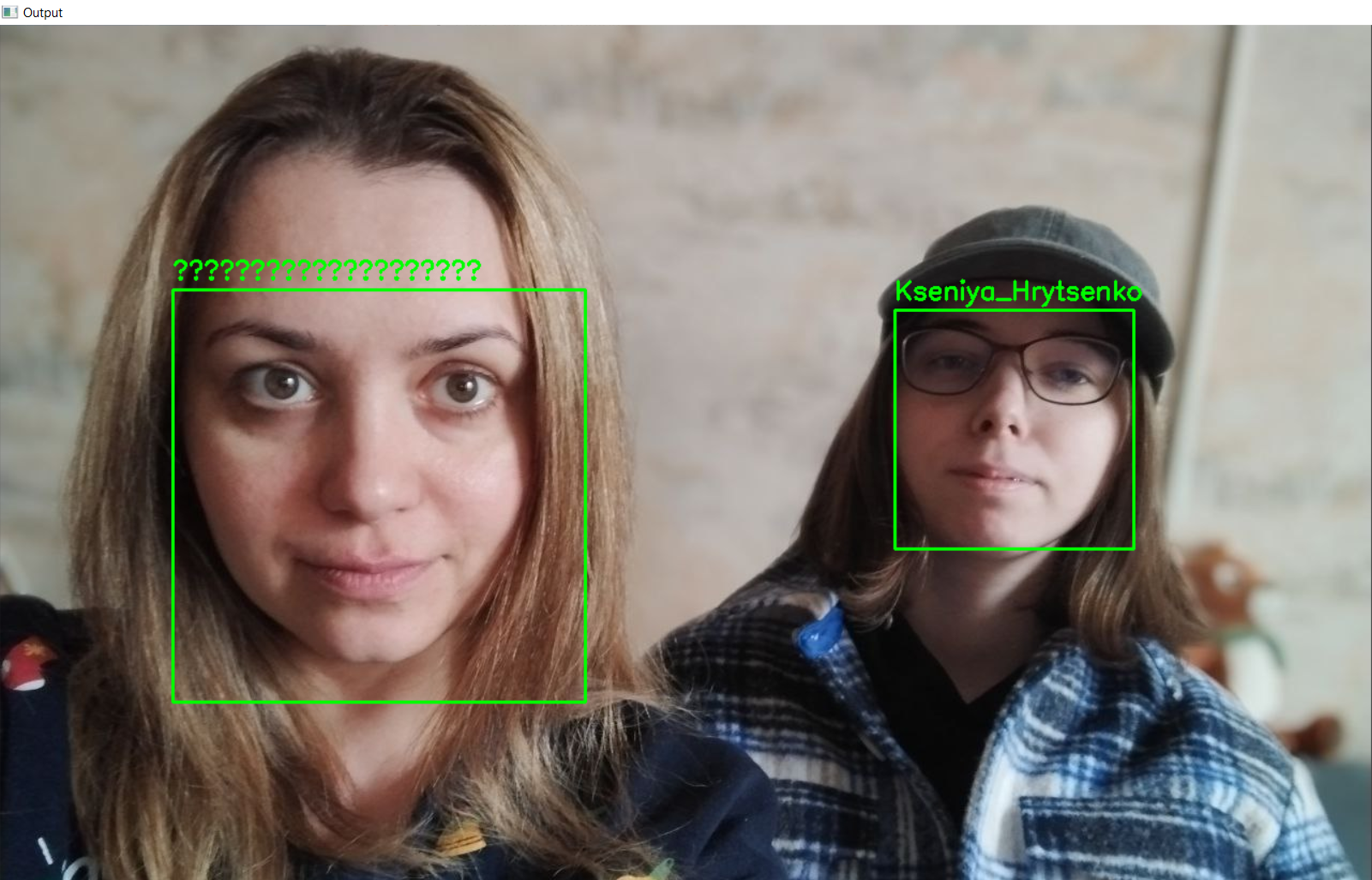
**Лабораторна робота № 4**

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

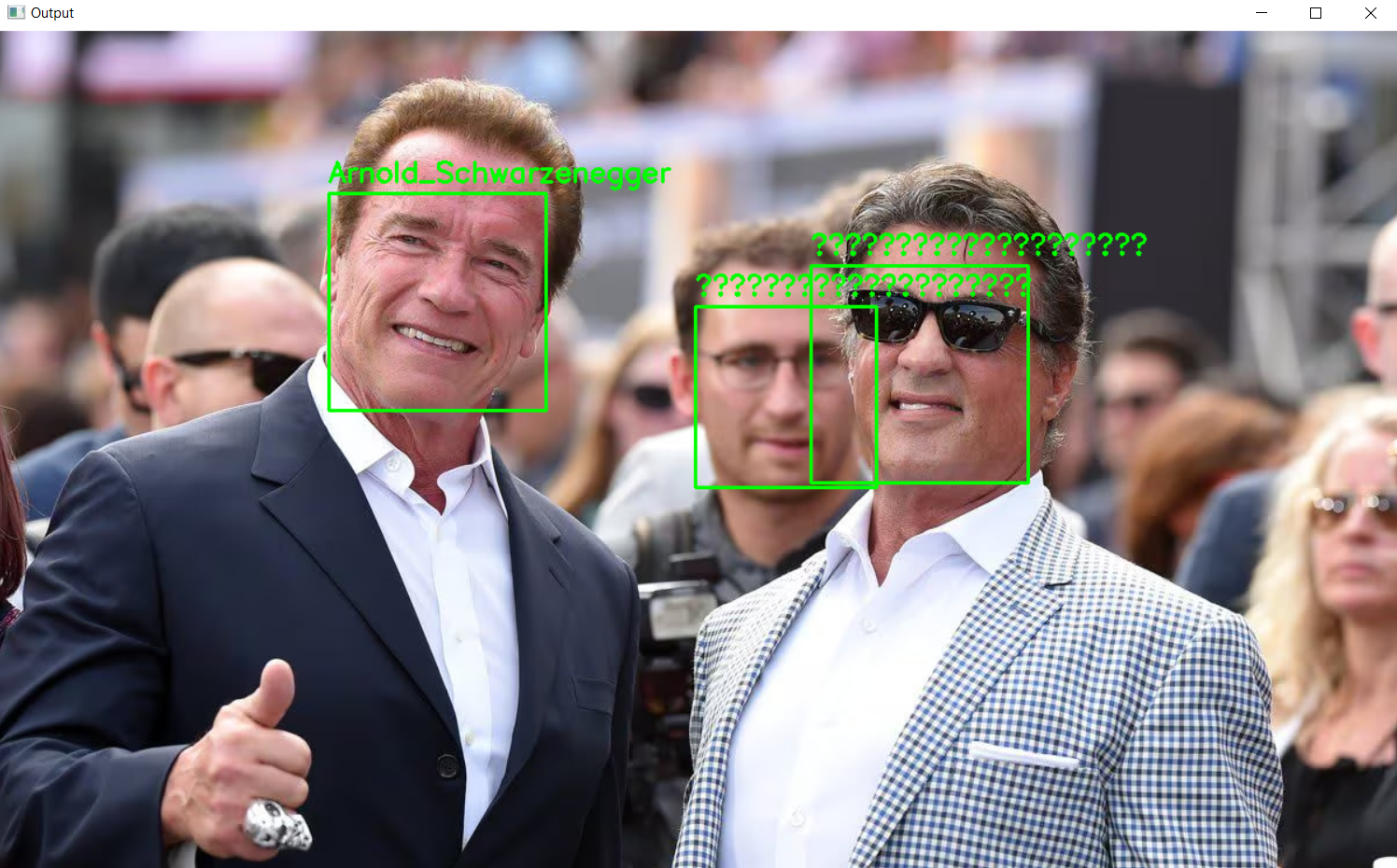
**Виконання роботи:**

Для виконання роботи я використовувала зображення себе, а також зірки – Арнольда Шварцнегера.

Я провела навчання моделі на 22 своїх фото та отримала результат.



Я провела навчання моделі на 42 фото Арнольда та отримала результат.



**Висновок:** Навчилася розпізнавати обличчя на фото, використовуючи навчені нейронні мережі.

**Контрольні запитання:**

1.Що таке розпізнавання обличчя?

Распознавание лиц — это технология, которая определяет или идентифицирует человека по его чертам лица на изображении или в видео.

2.Які кроки для розпізнвання обличчя?

Основные шаги: обнаружение лица, извлечение признаков, сравнение с базой данных и принятие решения о совпадении.

3.Що таке згорткова нейронна мережа?

Сверточная нейронная сеть (CNN) — это тип нейронной сети, которая эффективно обрабатывает изображения, автоматически выделяя важные признаки.

Основная идея CNN — использовать свёртки (convolutions), то есть специальные фильтры, которые "сканируют" изображение и извлекают ключевые признаки, такие как границы, углы, формы и текстуры.

4.Як відбувається навчання мережі?

Сеть обучается на большом наборе изображений, сравнивая предсказания с правильными ответами и корректируя внутренние параметры с помощью обратного распространения ошибки.

5.Що таке dlib?

Dlib — это мощная библиотека машинного обучения на C++ с интерфейсом для Python, используемая для распознавания лиц, трекинга объектов и других задач компьютерного зрения.

**Додаток А – Код файлу face\_encoding.py**

**import** os

**import** cv2

**import** pickle

**import** utils

# Отримуємо список усіх зображень у папці dataset

image\_paths **=** utils**.**get\_image\_paths**(**"dataset"**)**

encodings\_dict **=** **{}** # словник: ім’я особи -> список ембеддінгів її обличчя

**for** img\_path **in** image\_paths**:**

# Завантажуємо зображення та отримуємо ім’я папки (ім’я особи)

image **=** cv2**.**imread**(**img\_path**)**

name **=** os**.**path**.**basename**(**os**.**path**.**dirname**(**img\_path**))**

# Знаходимо обличчя та обчислюємо його ембеддінг

rects **=** utils**.**face\_rects**(**image**)**

**if** **len(**rects**)** **==** 0**:**

**continue** # обличчя не виявлено, пропускаємо файл

landmarks **=** utils**.**face\_landmarks**(**image**,** rects**)**

encodings **=** utils**.**face\_encodings**(**image**,** landmarks**)**

**if** **len(**encodings**)** **==** 0**:**

**continue** # не вдалося отримати ембеддінг

encoding **=** encodings**[**0**]** # беремо перший ембеддінг (для одного обличчя на фото)

# Зберігаємо у словник

**if** name **not** **in** encodings\_dict**:**

encodings\_dict**[**name**]** **=** **[]**

encodings\_dict**[**name**].**append**(**encoding**)**

# Зберігаємо словник з ембеддінгами у файл

**with** **open(**"encodings.pickle"**,** "wb"**)** **as** f**:**

pickle**.**dump**(**encodings\_dict**,** f**)**

**print(**f"[INFO] Збережено ембеддінгів облич: {**len(**encodings\_dict**)**} особ(и)"**)**

**Додаток Б – Код файлу face\_recognition\_images.py**

**import** os

**import** cv2

**import** pickle

**import** utils

# Завантажуємо базу відомих облич з файлу

**with** **open(**"encodings.pickle"**,** "rb"**)** **as** f**:**

encodings\_dict **=** pickle**.**load**(**f**)**

# Отримуємо список тестових зображень для розпізнавання

test\_image\_paths **=** utils**.**get\_image\_paths**(**"examples"**)**

**for** img\_path **in** test\_image\_paths**:**

image **=** cv2**.**imread**(**img\_path**)**

# Виявляємо обличчя на зображенні

rects **=** utils**.**face\_rects**(**image**)**

**if** **len(**rects**)** **==** 0**:**

**print(**f"[WARN] Облич не знайдено на зображенні {img\_path}"**)**

**continue**

landmarks **=** utils**.**face\_landmarks**(**image**,** rects**)**

encodings **=** utils**.**face\_encodings**(**image**,** landmarks**)**

# Проходимо по кожному обличчю і намагаємося його впізнати

**for** rect**,** face\_enc **in** **zip(**rects**,** encodings**):**

# Порівнюємо з кожною відомою особою

matches\_count **=** **{}**

**for** name**,** known\_enc\_list **in** encodings\_dict**.**items**():**

count **=** utils**.**nb\_of\_matches**(**known\_enc\_list**,** face\_enc**)**

matches\_count**[**name**]** **=** count

# Визначаємо найкращого кандидата

best\_match **=** **max(**matches\_count**,** key**=**matches\_count**.**get**)**

label **=** best\_match **if** matches\_count**[**best\_match**]** **>** 0 **else** "Невідомо"

# Малюємо результат на зображенні

x1**,** y1 **=** rect**.**left**(),** rect**.**top**()**

x2**,** y2 **=** rect**.**right**(),** rect**.**bottom**()**

cv2**.**rectangle**(**image**,** **(**x1**,** y1**),** **(**x2**,** y2**),** **(**0**,** 255**,** 0**),** 2**)**

cv2**.**putText**(**image**,** label**,** **(**x1**,** y1 **-** 10**),** cv2**.**FONT\_HERSHEY\_SIMPLEX**,** 0.8**,** **(**0**,** 255**,** 0**),** 2**)**

# Відображаємо зображення з підписами

cv2**.**imshow**(**"Output"**,** image**)**

cv2**.**waitKey**(**0**)**

cv2**.**destroyAllWindows**()**

**Додаток В – Код файлу utils.py**

**import** os

**import** cv2

**import** dlib

**import** numpy **as** np

# Ініціалізація детектора облич та моделей Dlib

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"**)**

**def** get\_image\_paths**(**root\_dir**):**

"""Отримує список шляхів до всіх зображень (jpg/png) у директорії root\_dir (включаючи підкаталоги)."""

image\_paths **=** **[]**

valid\_exts **=** **(**'.jpg'**,** '.jpeg'**,** '.png'**)**

**for** dirpath**,** \_**,** filenames **in** os**.**walk**(**root\_dir**):**

**for** fname **in** filenames**:**

**if** fname**.**lower**().**endswith**(**valid\_exts**):**

image\_paths**.**append**(**os**.**path**.**join**(**dirpath**,** fname**))**

**return** image\_paths

**def** face\_rects**(**image**):**

"""Повертає список прямокутників (областей облич), знайдених на зображенні."""

# Конвертуємо в сірий колір для детектора

gray **=** cv2**.**cvtColor**(**image**,** cv2**.**COLOR\_BGR2GRAY**)**

# Знаходимо обличчя (upsample=1 для кращого виявлення дрібних облич)

rects **=** face\_detector**(**gray**,** 1**)**

**return** rects

**def** face\_landmarks**(**image**,** rects**):**

"""Для кожного обличчя (rect) повертає знайдені ключові точки обличчя."""

gray **=** cv2**.**cvtColor**(**image**,** cv2**.**COLOR\_BGR2GRAY**)**

landmarks **=** **[]**

**for** rect **in** rects**:**

shape **=** shape\_predictor**(**gray**,** rect**)** # 68 ключових точок обличчя

landmarks**.**append**(**shape**)**

**return** landmarks

**def** face\_encodings**(**image**,** landmarks\_list**):**

"""Генерує 128-вимірні кодування (embedding) для кожного обличчя на зображенні."""

encodings **=** **[]**

**for** shape **in** landmarks\_list**:**

face\_descriptor **=** face\_encoder**.**compute\_face\_descriptor**(**image**,** shape**)**

encodings**.**append**(**np**.**array**(**face\_descriptor**))**

**return** encodings

**def** nb\_of\_matches**(**known\_encodings**,** unknown\_encoding**,** threshold**=**0.6**):**

"""Рахує, скільки ембеддінгів з known\_encodings знаходяться на відстані менше threshold від unknown\_encoding."""

count **=** 0

**for** enc **in** known\_encodings**:**

dist **=** np**.**linalg**.**norm**(**enc **-** unknown\_encoding**)**

**if** dist **<** threshold**:**

count **+=** 1

**return** count