

Міністерство освіти і науки України

Національний технічний університет України „КПІ імені Ігоря Сікорського”

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Звіт до лабораторної роботи №4

З дисципліни «Штучний інтелект в задачах обробки зображень»

|  |  |
| --- | --- |
| Перевірив:  доц. Нікітін В.А. | Виконала:  Студентка 3 курсу, гр. ІС-13  Росновська О.О. |

Київ 2024 р.

**Лабораторна робота №3.**

**Тема:** Розпізнавання людини на фото з використанням бібліотеки Dlib.

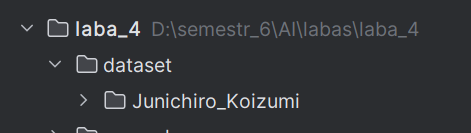
**Мета:** Навчитися розпізнавати обличчя на фото використовуючи навчені нейронні мережі

**Завдання:**

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

# Хід роботи

Перед початком проекту необхідно створити всі необхідні бібліотеки. А особливо dlib. Потім обрати датасет для навчання. Для першого завдання я оберу Коїдзумі Дзюнітіро, колишньго прем’єр-міністра Японії.





Першим кроком у конвеєрі розпізнавання облич є виявлення всіх облич на зображенні. Це можна зробити за допомогою детектора обличчя на основі глибокого навчання.

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.

def get\_image\_paths(root\_dir, class\_names):  
 *""" grab the paths to the images in our dataset"""* image\_paths = []  
  
 # loop over the class names  
 for class\_name in class\_names:  
 # grab the paths to the files in the current class directory  
 class\_dir = os.path.sep.join([root\_dir, class\_name])  
 class\_file\_paths = glob(os.path.sep.join([class\_dir, '\*.\*']))  
  
 # loop over the file paths in the current class directory  
 for file\_path in class\_file\_paths:  
 # extract the file extension of the current file  
 ext = os.path.splitext(file\_path)[1]  
  
 # if the file extension is not in the valid extensions list, ignore the file  
 if ext.lower() not in VALID\_EXTENSIONS:  
 print("Skipping file: {}".format(file\_path))  
 continue  
  
 # add the path to the current image to the list of image paths  
 image\_paths.append(file\_path)  
  
 return image\_paths

Далі ми створимо допоміжну функцію для вилучення областей обличчя із зображень. Для цього ми будемо використовувати детектор обличчя Dlib.

def face\_rects(image):  
 # convert the image to grayscale  
 gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 # detect faces in the grayscale image  
 rects = face\_detector(gray, 1)  
 # return the bounding boxes  
 return rects

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

def face\_landmarks(image):  
 return [shape\_predictor(image, face\_rect)  
 for face\_rect in face\_rects(image)  
 ]

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

def face\_encodings(image):  
 # compute the facial embeddings for each face  
 # in the input image. the `compute\_face\_descriptor`  
 # function returns a 128-d vector that describes the face in an image  
 return [  
 np.array(face\_encoder.compute\_face\_descriptor(image, face\_landmark))  
 for face\_landmark in face\_landmarks(image)  
 ]

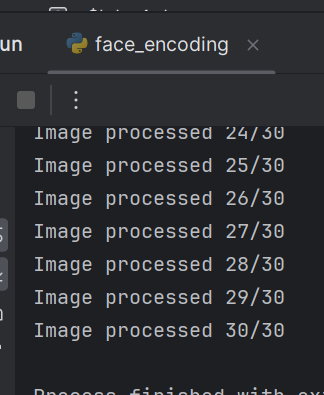
Починаємо будувати систему розпізнавання обличчя

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.

import os  
import pickle  
  
import cv2  
from utils import get\_image\_paths, face\_encodings  
  
root\_dir = "dataset"  
class\_names = os.listdir(root\_dir)  
  
# get the paths to the images  
image\_paths = get\_image\_paths(root\_dir, class\_names)  
  
# initialize a dictionary to store the name of each person and the corresponding encodings  
name\_encodings\_dict = {}  
  
  
# initialize the number of images processed  
nb\_current\_image = 1  
# now we can loop over the image paths, locate the faces, and encode them  
for image\_path in image\_paths:  
 print(f"Image processed {nb\_current\_image}/{len(image\_paths)}")  
 # load the image  
 image = cv2.imread(image\_path)  
 # get the face embeddings  
 encodings = face\_encodings(image)  
 # get the name from the image path  
 name = image\_path.split(os.path.sep)[-2]  
 # get the encodings for the current name  
 e = name\_encodings\_dict.get(name, [])  
 # update the list of encodings for the current name  
 e.extend(encodings)  
 # update the list of encodings for the current name  
 name\_encodings\_dict[name] = e  
 nb\_current\_image += 1  
  
  
# save the name encodings dictionary to disk  
with open("encodings.pickle", "wb") as f:  
 pickle.dump(name\_encodings\_dict, f)

Запустимо програму.



Після завершення роботи сценарію можна 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.

Тепер, коли у нас є вкладення обличчя, ми можемо почати розпізнавати обличчя на зображеннях

import os  
import glob  
import pickle  
import cv2  
from matplotlib import pyplot as plt  
  
from utils import face\_encodings, nb\_of\_matches, face\_rects  
  
# load the encodings + names dictionary  
with open("encodings.pickle", "rb") as f:  
 name\_encodings\_dict = pickle.load(f)  
  
# get the list of image files in the examples directory  
image\_files = glob.glob("examples/\*.jpg")  
  
for image\_file in image\_files:  
 # load the input image  
 image = cv2.imread(image\_file)  
  
 # get the 128-d face embeddings for each face in the input image  
 encodings = face\_encodings(image)  
 # this list will contain the names of each face detected in the image  
 names = []  
  
 # loop over the encodings  
 for encoding in encodings:  
 # initialize a dictionary to store the name of the  
 # person and the number of times it was matched  
 counts = {}  
 # loop over the known encodings  
 for (name, encodings) in name\_encodings\_dict.items():  
 # compute the number of matches between the current  
 # encoding and the encodings  
 # of the known faces and store the number of matches  
 # in the dictionary  
 counts[name] = nb\_of\_matches(encodings, encoding)  
 # check if all the number of matches are equal to 0  
 # if there is no match for any name, then we set the name  
 # to "Unknown"  
 if all(count == 0 for count in counts.values()):  
 name = "Unknown"  
 # otherwise, we get the name with the highest number of matches  
 else:  
 name = max(counts, key=counts.get)  
  
 # add the name to the list of names  
 names.append(name)  
  
 # loop over the `rectangles` of the faces in the  
 # input image using the `face\_rects` function  
 for rect, name in zip(face\_rects(image), names):  
 # get the bounding box for each face using the `rect` variable  
 x1, y1, x2, y2 = rect.left(), rect.top(), rect.right(), rect.bottom()  
 # draw the bounding box of the face along with the name of the person  
 cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)  
 cv2.putText(image, name, (x1, y1 - 10),  
 cv2.FONT\_HERSHEY\_SIMPLEX, 0.75, (0, 255, 0), 2)  
  
 # add the filename as a title at the top of the image  
 filename = os.path.basename(image\_file)  
 cv2.putText(image, filename, (10, 30),  
 cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 2)  
  
 # show the output image  
 cv2.imshow("image", image)  
 cv2.waitKey(0)  
  
 # save the output image  
 output\_image\_path = f"output\_{filename}"  
 cv2.imwrite(output\_image\_path, image)

Проведемо розпізнавання обличчя Коїдзумі Дзюнітіро.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Як бачимо, із задачею програма справляється добре, проте в останньому випадку була помилка. Я вважаю, програма помилилась, адже було замало даних для тренування (30) та на фото, на якому була здійснена помилка присутня людина зі схожими рисами обличчя.

Тепер навчимо програму на датасеті з моїми фото та спробуємо розпізнати мене.

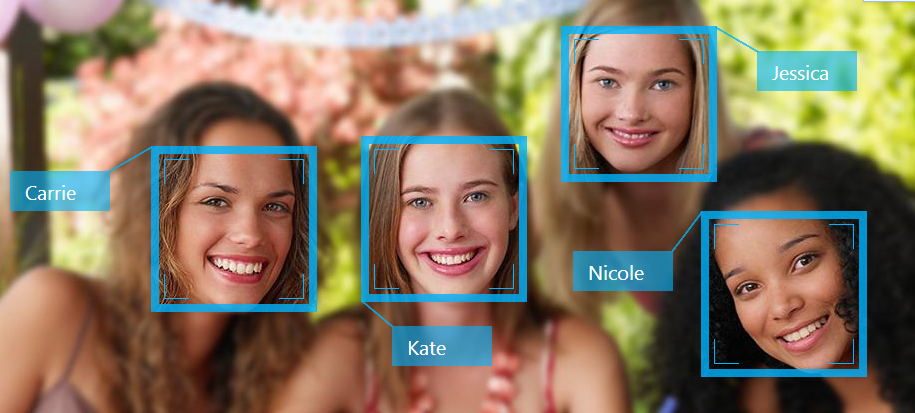
|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Як бачимо, із задачею програма справляється добре, проте в одному випадку була помилка. Я вважаю, програма помилилась, адже було замало даних для тренування (15) та на фото. Також фото, на якому було зроблено помилку належить жінці з активною мімікою. Це мого вплинути на якість розпізнавання. Добре що не чоловіка розпізнала як мене.

**Контрольні запитання:**

1. **Що таке 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.



1. **Які кроки для розпізнавання обличчя?**

1) Пошук обличчя

2) Вирівнювання обличчя за допомогою орієнтирів обличчя

3) Кодування обличчя

4) Розпізнавання обличчя

1. **Що таке згорткова нейронна мережа?**

Згорткова нейронна мережа (Convolutional Neural Network, CNN) — це тип глибинної нейронної мережі, який широко використовується для аналізу зображень і відео. CNN має здатність автоматично виявляти та витягувати важливі особливості (фічі) з вхідних даних за допомогою спеціалізованих шарів, таких як згорткові шари, шари підвибірки (пулінгу) та повнозв'язні шари.

1. **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.

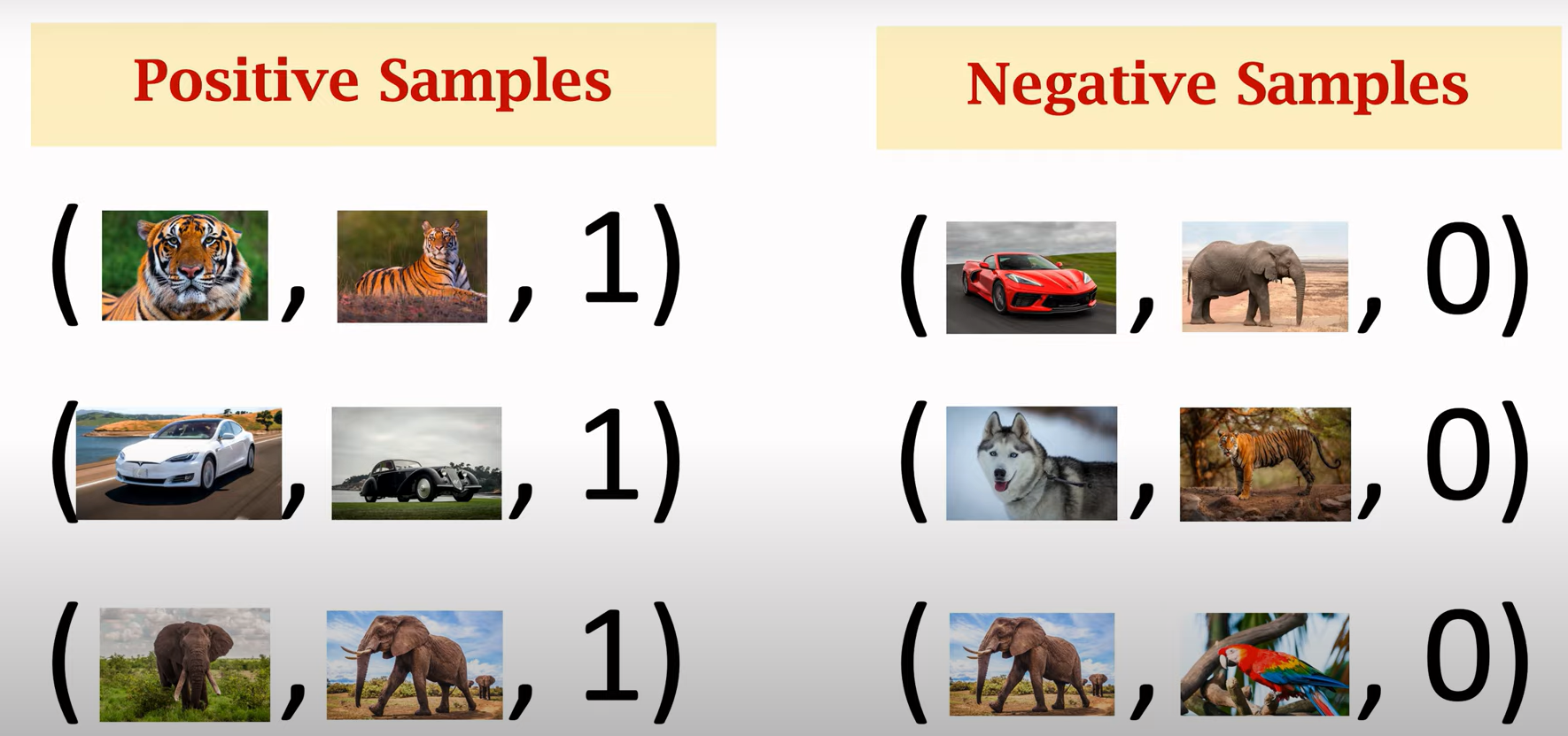


Рисунок - Приклад процесу навчання сіамської мережі

**5. Що таке dlib?**

Це багатофункціональна програмна бібліотека, яка містить різні інструменти для машинного навчання та комп'ютерного зору. Вона широко використовується для задач обробки зображень, таких як розпізнавання облич, відстеження облич, виявлення об'єктів і багато іншого.