МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Телеграм-бот для розпізнавання емоцій людей по фото.»

Завідувач

випускаючої

кафедри

Довбиш А. С

Керівник роботи **Петров С.О.**

Студент гр. ІН-73 **Цілуйко В.О.**

СУМИ 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2021 р.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи Ін-73 спеціальності "Комп'ютерні науки" денної форми навчання Цілуйка Володимира Олександровича.

Тема: «Телеграм бот для розпізнавання емоцій людей по фото.»

Затверджена наказом по СумДУ

№_____ від _____2021 р.

Зміст пояснювальної записки: 1) аналітичний огляд предметної області веб-форумів; 2) постановка завдання й формування завдань реалізації; 3) інструменти для вирішення поставлених задач; 5) реалізація клієнтської частини web-форуму для автомобілістів; 6) аналіз результатів.

Дата видачі завдання "_____"_____20__ р.

Керівник випускної роботи _____ Петров С.А.

Завдання прийняв до виконання _____ Цілуйко В.О.

РЕФЕРАТ

Записка: 48 стор., 20 рис., 4 табл., 1 додаток, 10 джерел.

Об’єкт дослідження — телеграм бот для розпізнавання емоцій людей по фото

Мета роботи — розробка телеграм боту з визначення емоцій по фото. Продукт дозволяє задовольнити потреби цільової аудиторії, яка потребує позначити емоції людини.

Методи дослідження — алгоритми роботи з фото

Результати — Ми вивчили сучасну бібліотеку OpenCv, яка використовується в Machine Learning і яка має понад дві тисячі різноманітних алгоритмів. Також ми навчилися робити найпростіший комп'ютерний зір, що дозволяє нам розробити багато функціоналу

TELEGRAM, BOT, PYTHON, OpenCV, ПРОГРАМУВАННЯ

Зміст

Вступ

Ми живемо в 21 столітті, яке дарує нам високі інформаційні технології. Зараз період, який характеризується небувалим зростанням обсягу інформаційних потоків. Так само ми могли помітити збільшення спілкування в соціальних мережах і власне зменшення живого спілкування так званого лицем до лиця. Також нині дуже популярний вид спілкування - через відео повідомлення, але навіть враховуючи цей фактор, деякі люди мають проблеми в розпізнаванні емоцій. Особливо часто в нашому суспільстві почали використовувати сарказм та інші емоції, які складно розпізнати, наприклад: щастя, злість, смуту і тд. Особливо під час карантину це стає все більш явною проблемою, тому я вирішив зробити телеграм-бота, який допоможе їм з цим.

Мій телеграм бот вміє розпізнавати безліч емоцій, що може допомогти цій невеликій кількості людей. Основна задача полягала в тому, щоб навчитися працювати з моделями, які згенеровані з допомогою data science і поєднати це зі звичайним телеграм ботом. Адже Machine Learning це нова та популярна тема, яка може вирішувати велику кількість проблем.

Розглянемо базові поняття, що лежать в основі методів автоматичного розпізнавання емоцій. Емоції відображають ставлення людини до різних явищ. Вони характеризуються суб'єктивними переживаннями людини, не пов'язані на пряму з порушенням певних рецепторів, можуть виникати спонтанно. При однаковій дії одних і тих же факторів на різних людей, емоції можуть викликати у них різні переживання.

Відповідно до теорії емоцій Роберта Плутчика, основними емоціями є радість, смуток, страх, довіра, очікування, подив, злість, невдоволення.

Емоції виявляються по-різному і в комбінації один з одним можуть вийти зовсім нові емоційні стани. Повноцінне спілкування між людьми неможливо без прояву і аналізу емоцій. Тому при створенні сучасних людино-машинних систем актуально застосування методів автоматичного розпізнавання емоцій.

Одним з основних способів розпізнавання емоцій людини іншою людиною є аналіз візуальної інформації. Тому автоматизація цього процесу очевидно повинна бути основана на використанні методів і засобів комп'ютерного зору. Комп'ютерне зір є науковою областю, в рамках якої ведуться дослідження по вивченню теорії і фундаментальних алгоритмів аналізу зображень об'єктів і сцен. Часто також замість поняття «Комп'ютерне зір» використовують «Машинний зір» або «Технічний зір». Однак останні поняття відносяться до більш загальної науково-практичної області, що охоплює всі етапи розробки систем, базуючих на обробці і аналізі відеоінформації. Приблизно так це буде виглядати -

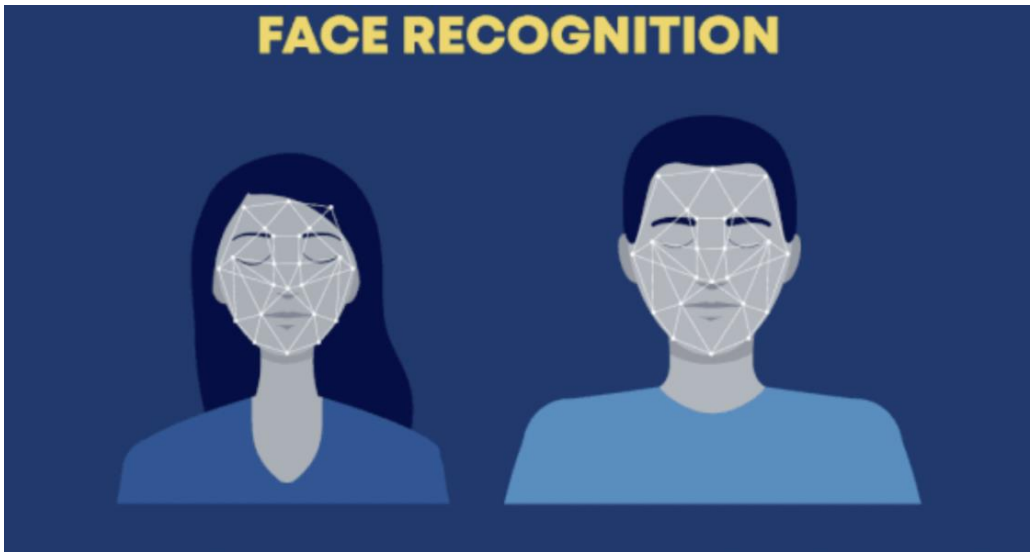

Рис.1.1 “Приклад роботи”

ПОСТАНОВКА ЗАДАЧІ

Метою випускної роботи є розробка телеграм боту з визначення емоцій по фото. Продукт дозволяє задовольнити потреби цільової аудиторії, яка потребує позначити емоції людини.

Методи обробки та аналізу зображень в інтелектуальних детекторах позаштатних ситуацій розвиваються в рамках відеоаналітики. До даного напрямку відносяться також методи, використовувані в маркетингових системах.

У маркетинговій сфері відповідні системи можуть бути використані з метою оперативного відстеження та реагування на різні проблеми в торгових центрах, супермаркетах та інших місцях продажів товарів і послуг, наприклад:

- визначення черг і їх оптимізація;
- підрахунок відвідувачів з класифікацією за віком, статтю, расою;
- оцінка якості обслуговування;
- аналіз поведінки персоналу;
- аналіз ефективності промоакцій;
- аналіз ефективності методики продажу товару в магазині (мерчендайзингу);
- визначення часу доби і днів тижня з найбільш щільним потоком відвідувачів;
- визначення «гарячих» зон в магазинах;
- показ реклами в залежності від статі і віку;
- визначення оптимального положення рекламних місць.

Ефективність вирішення ряду зазначених маркетингових завдань може бути значно підвищена за рахунок автоматичного розпізнавання емоцій клієнтів. Розпізнавання емоцій застосовується також в цілому ряді інших областей, таких як телекомунікації, відеоігри, анімація, психіатрія, автоматизоване навчання і т.д.

В роботі розглядається задача створення програми для трекінгу емоцій через фото, при цьому необхідно виконати такі завдання :

- знайти навчену модель даних
- отримати фото через телеграм бота
- обвести обличчя на фото
- відправити фото назад

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження предметної області

Для початку визначимо що саме означає це нове визначення “Machine learning”. Клас методів штучного інтелекту, характерною рисою яких є не пряме рішення задачі, а навчання за рахунок застосування рішень безлічі подібних завдань. Для побудови таких методів використовуються засоби математичної статистики, чисельних методів, математичного аналізу, методів оптимізації, теорії ймовірностей, теорії графів, різні техніки роботи з даними в цифровій формі.

Що таке боти в Telegram?

Боти - це спеціальні програми, що виконують різні функції і спрощують життя їх користувачів. Написані для платформи Telegram, вони призначені для виконання самих різних функцій: від отримання новин до пошуку інформації і навіть торгівлі акціями. Головне завданням бота є автоматичний відповідь після введеної йому користувачем команди. При цьому, працюючи безпосередньо через інтерфейсу Telegram, програма імітує дії живого користувача, за рахунок чого користування таким ботом набагато зручно і зрозуміло.

Саме тому, багато компаній, що розвивають бізнес через інтернет, використовують можливості ботів

1.2 Аналіз існуючих аналогів

В рамках програми Project Oxford корпорація Microsoft продовжує розробляти різні сервіси, що використовує методи машинного навчання. Цієї весни ми вже розповідали про сайт How-Old.net, який за фотографією визначає вік і стать зображених на ній людей, а тепер софтверний гігант в тестовому режимі запустив онлайн-аналізатор емоцій людини по виразу його обличчя. У своїй роботі сервіс оперує вісьмома параметрами - гнів, презирство, відраза, страх, щастя, байдужість, смуток і здивування, кожному з яких присвоюється значення від 0 до 1.

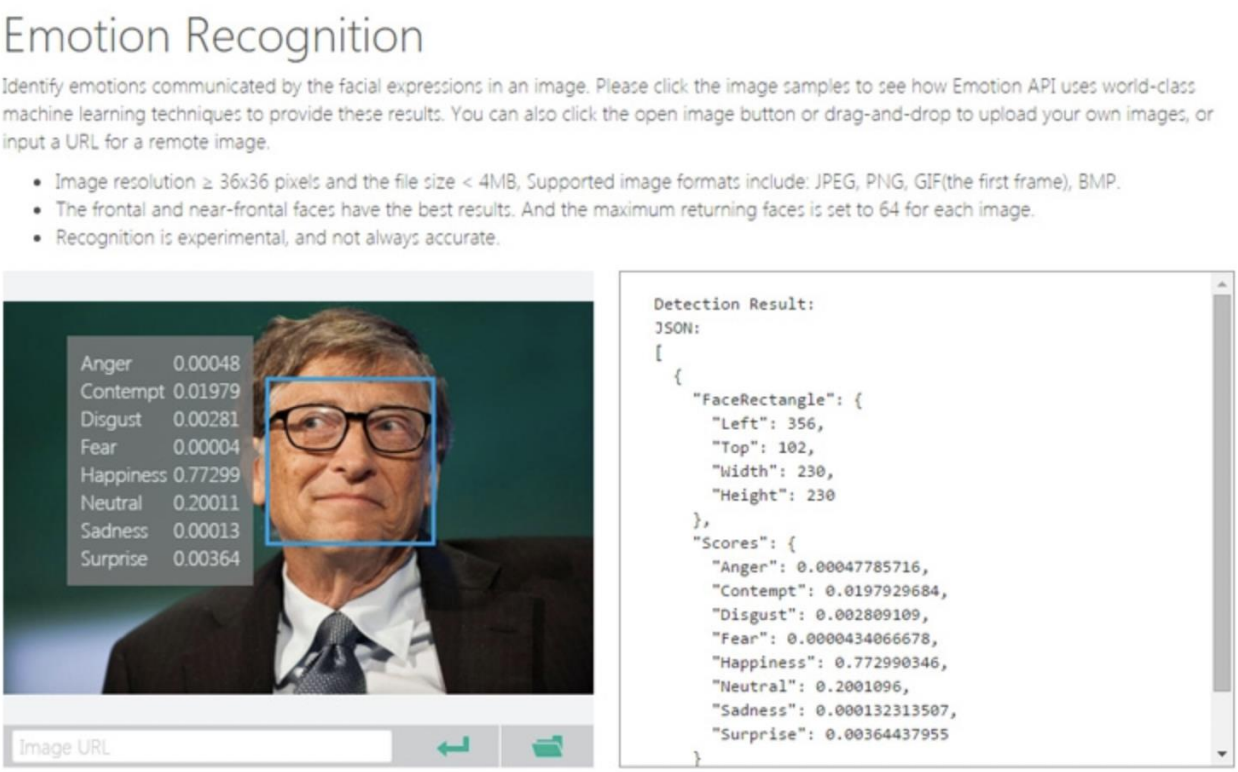

Рис.2.1 “Перший конкурент”

Даний варіант є на належному рівні дизайну та розробки, але у нас є неймовірна перевага. Люди дуже часто використовують соцмережі на своїх мобільних пристроях і для вирішення нашої задачі користувачеві не потрібно закривати його додаток (в нашому випадку телеграм), щоб визначити емоції по фото.

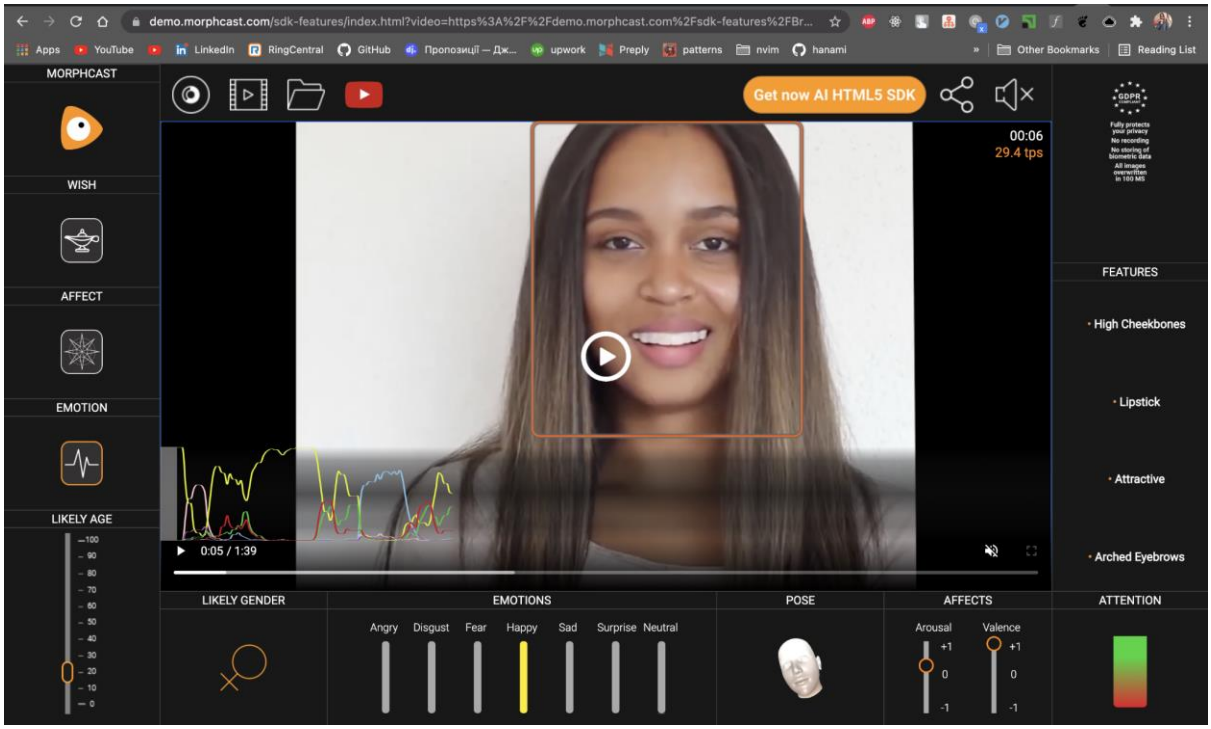

Рис.2.2 “Другий конкурент”

Наш другий конкурент це https://www.morphcast.com/ .
Цей продукт має чудовий сайт і багато додаткових функцій, такі як визначення емоцій по відео, а так само позначення кольору волосся і приблизний вік людини. Але їх основний мінус, що пропонується веб додаток або додаток на ваш комп'ютер.

1.3 ІНФОРМАЦІЙНИЙ ОГЛЯД

Набір даних, що використовується для навчання, - це набір даних розпізнавання емоцій Kaggle FER2013: https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data . Дані складаються із 48x48 піксельних відтінків сірого у гранях облич. Обличчя були автоматично зареєстровані, так що обличчя більш-менш відцентровано і займає приблизно однакову кількість місця на кожному зображенні. Завдання полягає в класифікації кожного обличчя на основі емоцій, що відображаються у виразі обличчя, до однієї із семи категорій (0 = злий, 1 = огида, 2 = страх, 3 = щасливий, 4 = сумний, 5 = сюрприз, 6 = нейтральний).

Ця реалізація за замовчуванням виявляє емоції на всіх обличчях у каналі веб-камери. За допомогою простого 4-шарового CNN точність випробування досягла 63,2% за 50 епох.

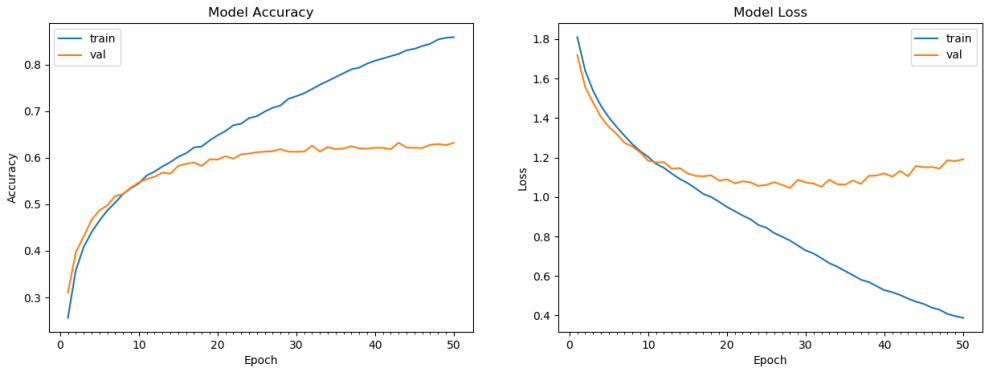

Рис.3.1 “Графік”

Ознаки Хаара - ознаки цифрового зображення, використовувані в розпізнаванні образів. Своєю назвою вони зобов'язані інтуїтивним схожістю з вейвлетами Хаара. Ознаки Хаара використовувалися в першому детекторі осіб, що працює в реальному часі.

Історично склалося так, що алгоритми, що працюють тільки з інтенсивністю зображення (наприклад значення RGB в кожному пікселі),

мають велику обчислювальну складність. В роботі Папагеоргіу , була розглянута робота з безліччю ознак, заснованих на вейвлет Хаара. Віола і Джонс адаптували ідею використання вейвлетів Хаара та розробили те, що було названо ознаками Хаара. Ознака Хаара складається з суміжних прямокутних областей. Вони позиціонуються на зображенні, далі сумуються інтенсивності пікселів в областях, після чого обчислюється різниця між сумами. Ця різниця і буде значенням певної ознаки, визначеного розміру, певним чином позиційований на зображенні.

Для прикладу розглянемо базу даних з людськими обличчями. Загальним для всіх зображень є те, що область в районі очей темніше, ніж область в районі щік. Отже загальним ознакою Хаара для осіб є 2 суміжних прямокутних регіону, що лежать на очах і щоках.

На етапі виявлення в методі Віоли - Джонса вікно встановленого розміру рухається по зображенню, і для кожної області зображення, над якою проходить вікно, розраховується ознака Хаара. Наявність або відсутність предмета в вікні визначається різницею між значенням ознаки і учнем порогом. Оскільки ознаки Хаара мало підходять для навчання або класифікації (якість трохи вище ніж у випадкової нормально розподіленої величини), для опису об'єкта з достатньою точністю необхідна більша кількість ознак. Тому в методі Віоли - Джонса ознаки Хаара організовані в каскадний класифікатор.

Наступним кроком є знаходження ключових точок виділених елементів особи. Визначити емоції можна на основі аналізу декількох ключових точок. Наприклад, на рис. 2 показані комбінації точок брів і рота (зображення брів і рота; комбінація ключових точок брів і рота, відповідна їх зображенням; комбінація ключових точок, відповідна іншим положенням розглянутих елементів на зображеннях).

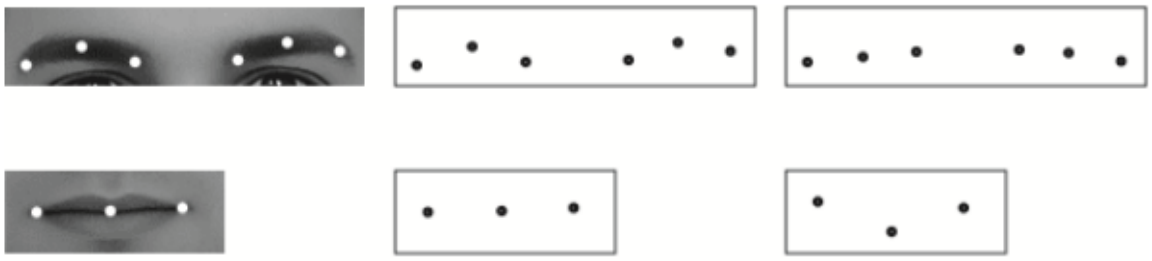

Рис.3.2 “Приклад пошуку ключових точок на обличчі”

Також дуже часто використовується алгоритм AdaBoost. AdaBoost найкраще використовувати для підвищення продуктивності дерев рішень щодо двійкових проблем класифікації. Автори методики Фрейнд і Шапір спочатку називали AdaBoost.M1 AdaBoost.M1. Зовсім недавно його можна було називати дискретним AdaBoost, оскільки він використовується для класифікації, а не для регресії. AdaBoost можна використовувати для підвищення продуктивності будь-якого алгоритму машинного навчання. Найкраще використовувати його для слабких учнів. Це моделі, які досягають точності трохи вище випадкового шансу на проблему класифікації.

Найбільш відповідний і тому найпоширенішим алгоритмом, що використовується з AdaBoost, є дерева рішень з одним рівнем. Оскільки ці дерева такі короткі і містять лише одне рішення для класифікації, їх часто називають пнями рішення. Кожен екземпляр у навчальному наборі даних зважується. Початкова вага встановлюється так:

weight(xi) = 1/n

Де xi - і-й навчальний екземпляр, а n - кількість навчальних екземплярів.

2. ОПИС РІШЕННЯ ЗАДАЧІ

Для вирішення задач я вибрав мову програмування Python. Ця мова програмування має багато бібліотек за допомогою яких ми будемо реалізовувати нашу ціль. Основні бібліотеки з нашого проекту:

NumPy - це бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих (і дуже швидких) математичних функцій для операцій з цими масивами.

Pandas - це високорівнева Python бібліотека для аналізу даних. Чому я її називаю високорівневою, тому що побудована вона поверх більш низкоуровневой бібліотеки NumPy (написана на C), що є великим плюсом в продуктивності. В екосистемі Python, pandas є найбільш потужною, що швидко розвивалася бібліотекою для обробки і аналізу даних.

opencv-python - бібліотека комп'ютерного зору і машинного навчання з відкритим вихідним кодом. У неї входять понад 2500 алгоритмів, в яких є як класичні, так і сучасні алгоритми для комп'ютерного зору і машинного навчання. Ця бібліотека має інтерфейси на різних мовах, серед яких є Python. Як зрозуміло із опису це лише обертка над OpenCv

OpenCV (Open Source Computer Vision Library: http://opencv.org) - це бібліотека з відкритим кодом, що включає кілька сотень алгоритмів комп'ютерного зору. Документ описує так званий API OpenCV 2.x, який, по суті, є API C ++, на відміну від API OpenCV 1.x на основі C (C API застарілий і не тестується за допомогою компілятора "C" з моменту випуску OpenCV 2.4) OpenCV має модульну структуру.

PyTelegramBotApi - це бібліотека яка є оберткою для Bot Telegram API в неї я дуже багато методів які спрощують роботу запитів до API .

Telegram bot API - цей API дозволяє підключати ботів до нашої системи. Боти Telegram - це спеціальні облікові записи, для налаштування яких не потрібен додатковий номер телефону. Ці облікові записи служать інтерфейсом для коду, який працює десь на вашому сервері.

Щоб використовувати це, вам не потрібно нічого знати про те, як працює наш протокол шифрування MTProto - наш посередницький сервер буде обробляти для вас все шифрування та зв’язок з API Telegram. Ви спілкуєтесь із цим сервером через простий HTTPS-інтерфейс, який пропонує спрощену версію API Telegram.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ

Перед початком програмування завжди повинен бути план. Тому я зробив невелику схему:

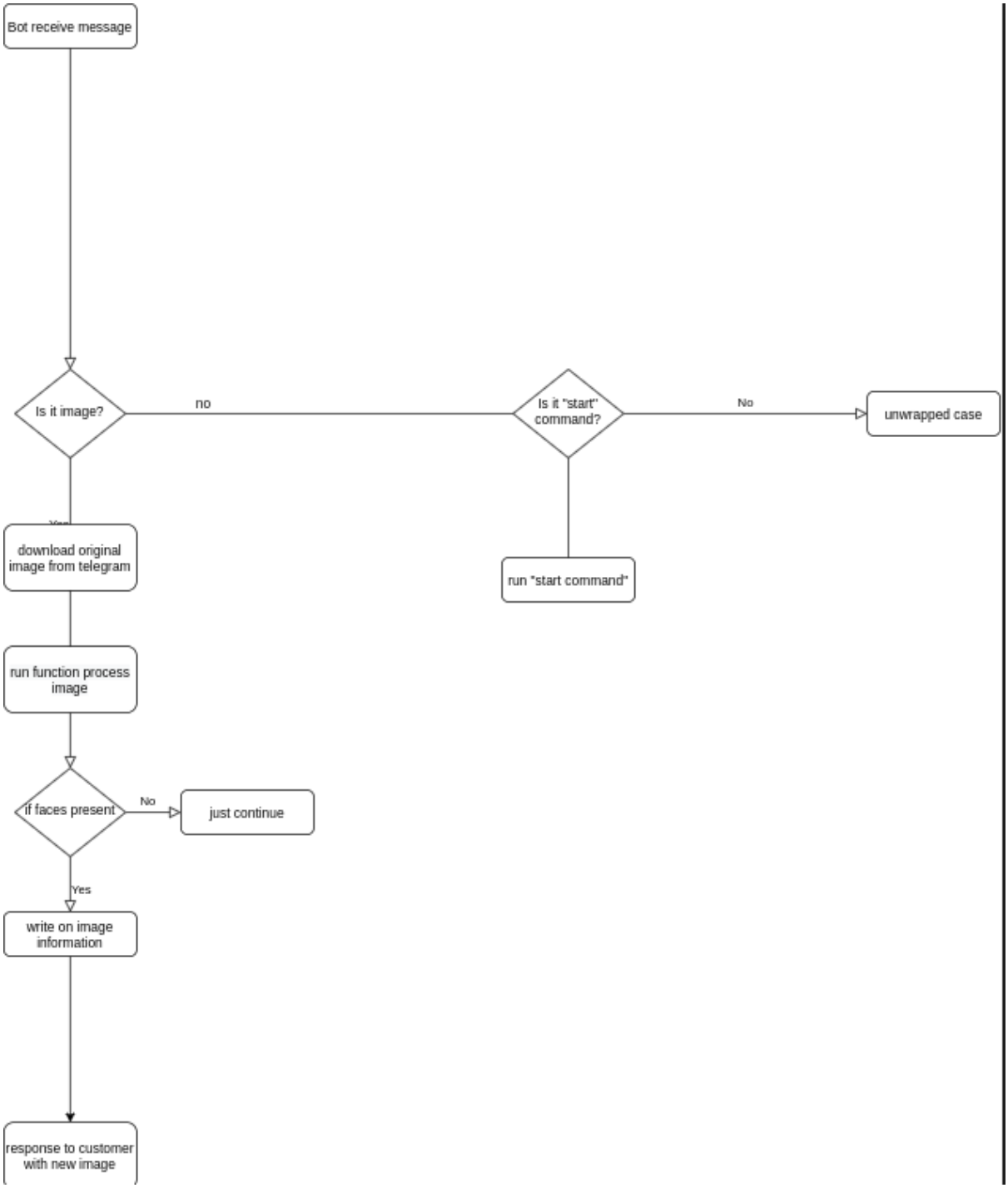

Рис.4.1 “Блок-схема проекту”

Розпізнавання облич за допомогою каскадів Хаара - це підхід, заснований на машинному навчанні, де функція каскаду навчається за допомогою набору вхідних даних. OpenCV вже містить багато попередньо навчених класифікаторів для обличчя, очей, посмішок тощо. Сьогодні ми будемо використовувати класифікатор обличчя.

Нам потрібно завантажити XML-файл навченого класифікатора (haarcascade_frontalface_default.xml), який доступний у сховищі GitHub OpenCv. Збережіть його на своєму робочому місці. Тепер коли ми розуміємо структуру проекту можемо приступати до коду. Розпочнемо з Models - папка із моделями, які ми скачали. Далі main.py - код телеграм боту, який є оберткою для нашої логіки пошуку обличь. Наступний файл live_face.py - логіка пошуку обличь на фото. І нарешті requirements.txt - наші бібліотеки.

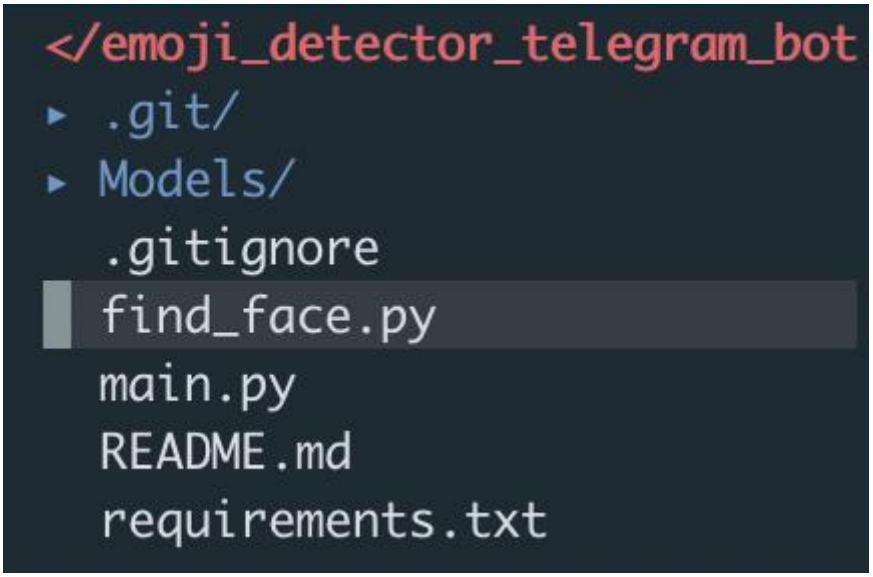

Рис.4.2 “Файли проекту”

Виклик нашої програми за домогою команди```python3 main.py``` запускає нашого телеграм бота.

```
3 @server.route("/")
2 def webhook():
1     bot.remove_webhook()
37    bot.set_webhook(url = SERVER_URL + TOKEN)
1     return "!", 200
2
3 bot.polling()
4
```

Рис.4.3 “код вебхука”

def webhook - запускається, якщо ця програма запускається на сервері. Вона робить запит до телеграму та надає посилання на наш webhook. Це зроблено для того, щоб телеграм робив до нас запити, коли до боту приходить повідомлення.

Представимо, що ми відправили до нашого боту фото та подивимося як воно пройде весь наш програмний код від початку до кінця.

Рис.4.4 “фото для тесту”

Бот отримує це фото в функцію:

```
@bot.message_handler(content_types=['photo'])
def handle_docs_audio(message):
    file_id = message.json["photo"][-1]["file_id"]
    file_info = bot.get_file(file_id)

    url = 'https://api.telegram.org/file/bot{0}/{1}'.format(API_TOKEN, file_info.file_path)
    file = requests.get('https://api.telegram.org/file/bot{0}/{1}'.format(API_TOKEN, file_info.file_path))
    response = requests.get(url, stream=True)

    with open("file_from_tg.jpg", "wb") as handle:
        for data in response.iter_content():
            handle.write(data)
    result = find_face.process_image("file_from_tg.jpg")
    time.sleep(1)
    photo = open('released_faces.jpg', 'rb')
    bot.reply_to(message, result)
    bot.send_photo(message.chat.id, photo)
```

Рис.4.5 “код”

Для початку ми беремо file_id , після цього робимо запит до телеграму та отримуємо повну інформацію про це фото. Коли у нас є повна інформація ми беремо з неї file_path - це путь за яким можливо отримати посилання для завантаження. За допомогою request.get ми робимо запит на завантаження та завантажуємо наш файл. Далі ми викликаємо функцію process_image .

```
def process_image(image) :

    shape_x = 48
    shape_y = 48
    input_shape = (shape_x, shape_y, 1)
    nClasses = 7

    thresh = 0.25
    frame_check = 20

    def eye_aspect_ratio(eye):
        A = distance.euclidean(eye[1], eye[5])
        B = distance.euclidean(eye[2], eye[4])
        C = distance.euclidean(eye[0], eye[3])
        ear = (A + B) / (2.0 * C)
        return ear

    def detect_face(frame):

        #Cascade classifier pre-trained model
        cascPath = 'Models/Landmarks/face_landmarks.dat'
        faceCascade = cv2.CascadeClassifier(cascPath)

        #BGR -> Gray conversion

        frame = cv.imread("file_froN.jpg")
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        #Cascade MultiScale classifier
        detected_faces = faceCascade.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=6,
                                                      minSize=(shape_x, shape_y),
                                                      flags=cv2.CASCADE_SCALE_IMAGE)
        coord = []

        for x, y, w, h in detected_faces :
            if w > 100 :
                sub_img=frame[y:y+h,x:x+w]
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0, 255,255),1)
                coord.append([x,y,w,h])

        return gray, detected_faces, coord

    def extract_face_features(faces, offset_coefficients=(0.075, 0.05)):
        gray = faces[0]
        detected_face = faces[1]

        new_face = []
```

Рис.4.6 “код”

Вилучення функцій: коли ми обрізали обличчя із зображення, ми витягуємо з нього функції. Тут ми будемо використовувати вбудовування обличчя, щоб витягнути риси обличчя. Нейронна мережа приймає зображення обличчя людини як вхідний сигнал і виводить вектор, який представляє найважливіші риси обличчя. У машинному навчанні цей вектор називається вбудовуванням, і, отже, ми називаємо цей вектор як вбудовування обличчя. Тепер, як це допомагає розпізнавати обличчя різних людей?

Навчаючи нейронну мережу, мережа вчиться виводити подібні вектори для облич, які схожі. Наприклад, якщо у мене є кілька зображень

облич протягом різного проміжку часу, звичайно, деякі особливості мого обличчя можуть змінитися, але не в значній мірі. Отже, у цьому випадку вектори, пов’язані з гранями, подібні або коротше кажучи, вони дуже близькі у векторному просторі. Погляньте на діаграму нижче, щоб отримати приблизну ідею:

Рис.4.7 “діаграма”

Зображення - це не що інше, як стандартний массив Numpy, що містить пікселі точок даних. Чим більше кількість пікселів у зображенні, тим краща його роздільна здатність. Ви можете думати, що пікселі - це крихітні блоки інформації, розташовані у формі двовимірної сітки, а глибина пікселя стосується наявної в ньому інформації про колір. Для обробки комп’ютером зображення потрібно перетворити у двійкову форму. Колір зображення можна розрахувати наступним чином:

Number of colors/ shades = 2^bpp where bpp represents bits per pixel.

Природно, що чим більше біт / пікселів, тим більше можливих кольорів на зображеннях. Наступна таблиця чіткіше показує взаємозв'язок.

Bits/Pixel	Possible colours
1	2^1 = 2
2	2^2 = 4
3	2^3 = 8
4	2^4 = 16
8	2^8 = 256
16	2^16 = 65000

Рис.4.8 “bit/pixel”

Кольорові зображення представлені у вигляді комбінації червоного, синього та зеленого, а всіх інших кольорів можна досягти, змішавши ці

основні кольори у правильних пропорціях.

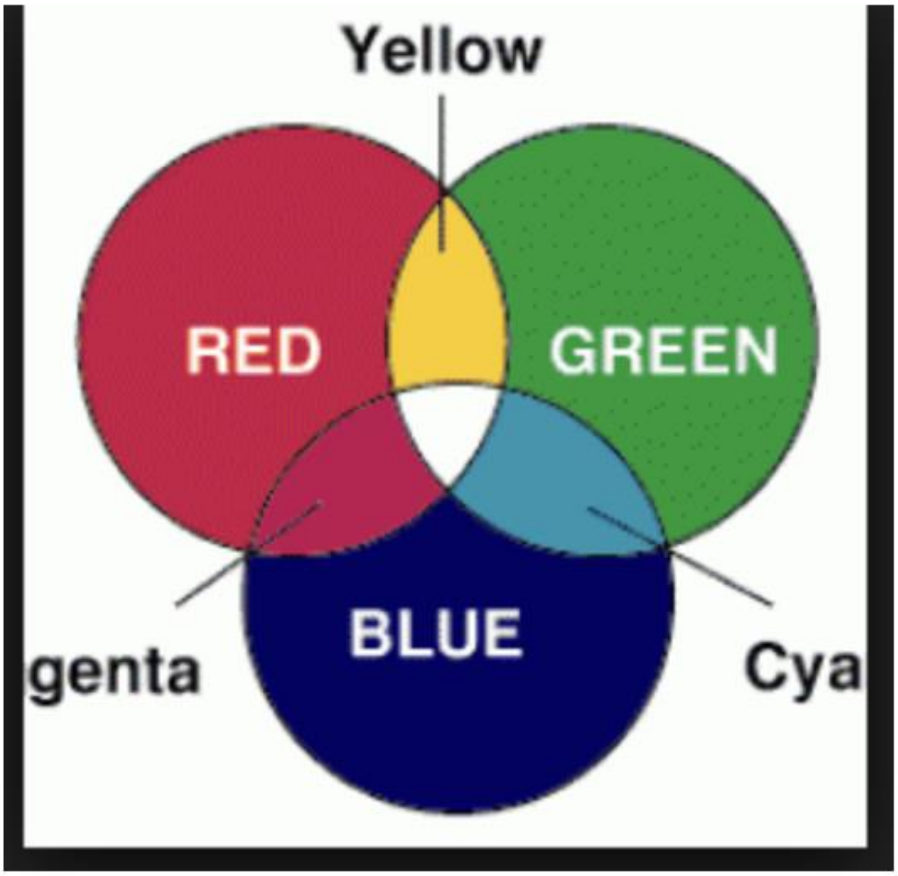

Рис.4.8 “Кольорова пропорция”

Кольорове зображення також складається з 8 біт на піксель. Як результат, 256 різних відтінків кольорів можуть бути представлені 0, що позначає чорний і 255 білий. Давайте розглянемо відоме кольорове зображення мандрили, яке цитувалось у багатьох прикладах обробки зображень.

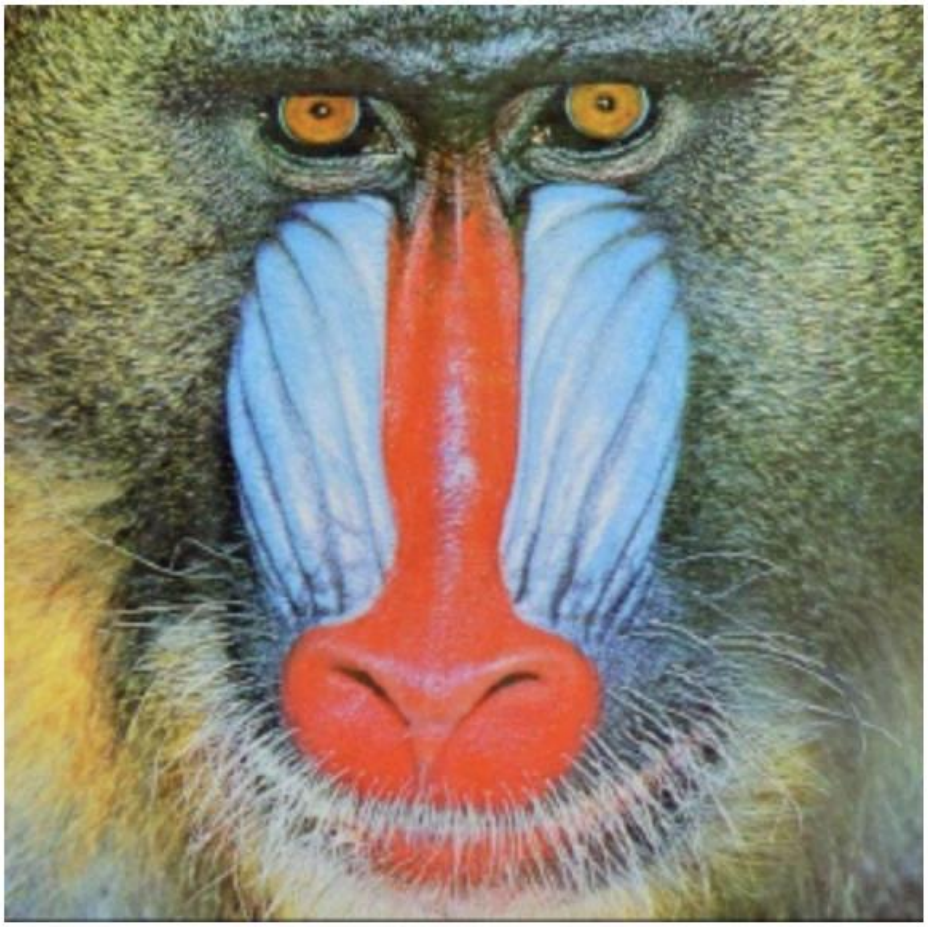

Рис.4.9 “Зображення мандрили”

Якби ми перевірили форму зображення вище, ми отримали б:

```
Shape
(288, 288, 3)
288: Pixel width
288: Pixel height
3: color channel
```

Рис.4.9 “формула зображення”

Це означає, що ми можемо представити вищезазначене зображення у вигляді тривимірного масиву.

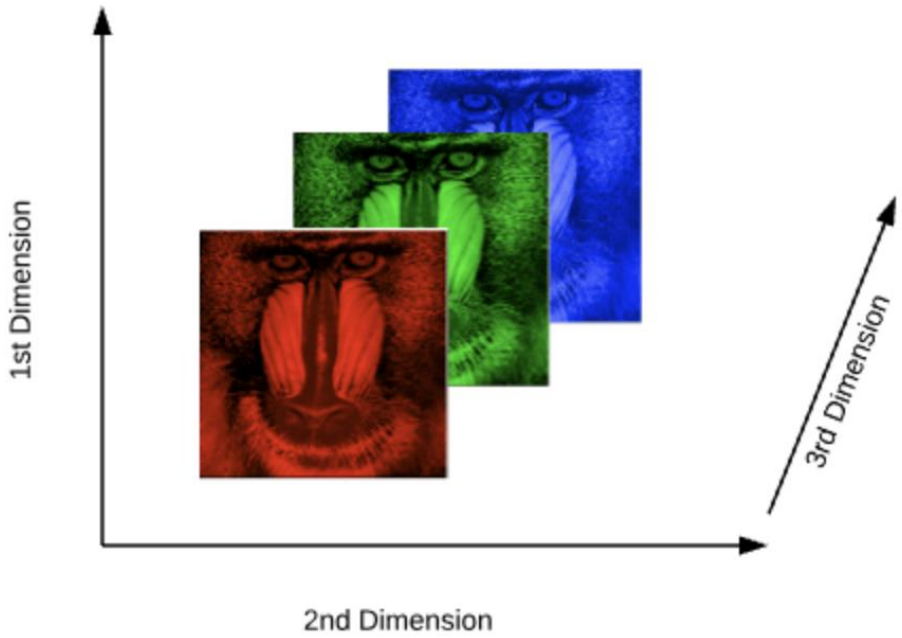

Рис.4.12 “Приклад працювання кольорових фото”

Саме так ми і визначаємо кольорові фотографії в нашому додатку (як і в будь-якому іншому).

В даній ділянці коду ми робимо зум на кожну особу, яка є на нашому фото читаємо його і отримуємо результат у вигляді тривимірного масиву даних (як було описано раніше).

```python
#Launch capture video
video_capture = cv2.VideoCapture(0)

while True:
    # Capture frame-by-frame
    ret, frame = video_capture.read()

    face_index = 0

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rects = face_detect(gray, 1)
    #gray, detected_faces, coord = detect_face(frame)

    for (i, rect) in enumerate(rects):

        shape = predictor_landmarks(gray, rect)
        shape = face_utils.shape_to_np(shape)

        # Identify face coordinates
        (x, y, w, h) = face_utils.rect_to_bb(rect)
        face = gray[y:y+h,x:x+w]

        #Zoom on extracted face
        face = zoom(face, (shape_x / face.shape[0],shape_y / face.shape[1]))

        #Cast type float
        face = face.astype(np.float32)

        #Scale
        face /= float(face.max())
        face = np.reshape(face.flatten(), (1, 48, 48, 1))

        #Make Prediction
        prediction = model.predict(face)
        prediction_result = np.argmax(prediction)
        prediction_result

        # Rectangle around the face
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        cv2.putText(frame, "Face #{}".format(i + 1), (x - 10, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

        for (j, k) in shape:
            cv2.circle(frame, (j, k), 1, (0, 0, 255), -1)
```

Рис.4.13 “код”

Далі ми шукаємо наших осіб на фото:

```python
def extract_face_features(faces, offset_coefficients=(0.075, 0.05)):
    gray = faces[0]
    detected_face = faces[1]

    new_face = []

    for det in detected_face :
        #Region dans laquelle la face est détectée
        x, y, w, h = det
        #X et y correspondent à la conversion en gris par gray, et w, h correspondent à la hauteur/largeur

        #Offset coefficient, np.floor takes the lowest integer (delete border of the image)
        horizontal_offset = np.int(np.floor(offset_coefficients[0] * w))
        vertical_offset = np.int(np.floor(offset_coefficients[1] * h))

        #gray transforme l'image
        extracted_face = gray[y+vertical_offset:y+h, x+horizontal_offset:x-horizontal_offset+w]

        #Zoom sur la face extraite
        new_extracted_face = zoom(extracted_face, (shape_x / extracted_face.shape[0],shape_y / extracted_face.shape[1]))
        #cast type float
        new_extracted_face = new_extracted_face.astype(np.float32)
        #scale
        new_extracted_face /= float(new_extracted_face.max())

        new_face.append(new_extracted_face)

    return new_face
```

Рис.4.13 “пошук особ”

Наша програма дивиться на ніс та очі (як і було описано раніше в наших методах):

```
(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]

(nStart, nEnd) = face_utils.FACIAL_LANDMARKS_IDXS["nose"]
(mStart, mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]
(jStart, jEnd) = face_utils.FACIAL_LANDMARKS_IDXS["jaw"]

(eblStart, eblEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eyebrow"]
(ebrStart, ebrEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eyebrow"]

model = tf.keras.models.load_model('Models/EmotionXCeption/video.h5')
face_detect = dlib.get_frontal_face_detector()
predictor_landmarks  = dlib.shape_predictor("Models/Landmarks/face_landmarks.dat")

#Launch la capture video
video_capture = cv2.VideoCapture(0)

# Capture frame-by-frame
frame = cv2.imread("file_from_tg.jpg")
#ret, frame = video_capture.read()

face_index = 0

gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
rects = face_detect(gray, 1)
#gray, detected_faces, coord = detect_face(frame)

face_reports = ""
for (i, rect) in enumerate(rects):

    shape = predictor_landmarks(gray, rect)
    shape = face_utils.shape_to_np(shape)

    # Identify face coordinates
    (x, y, w, h) = face_utils.rect_to_bb(rect)
    face = gray[y:y+h,x:x+w]

    #Zoom on extracted face
    face = zoom(face, (shape_x / face.shape[0],shape_y / face.shape[1]))

    #Cast type float
    face = face.astype(np.float32)

    #Scale
    face /= float(face.max())
```

Рис.4.13 “обробка очей”

Коли ми визначити дані емоцій ми їх запам'ятовуємо, щоб відправити ці дані нашому користувачеві.

```
+  43
+  42          # 3. Eye Detection and Blink Count
+  41          leftEye = shape[lStart:lEnd]
+  40          rightEye = shape[rStart:rEnd]
+  39
+  38          # Compute Eye Aspect Ratio
+  37          leftEAR = eye_aspect_ratio(leftEye)
+  36          rightEAR = eye_aspect_ratio(rightEye)
+  35          ear = (leftEAR + rightEAR) / 2.0
+  34
+  33          # And plot its contours
+  32          leftEyeHull = cv2.convexHull(leftEye)
+  31          rightEyeHull = cv2.convexHull(rightEye)
+  30          cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
+  29          cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
+  28
+  27          # 4. Detect Nose
+  26          nose = shape[nStart:nEnd]
+  25          noseHull = cv2.convexHull(nose)
+  24          cv2.drawContours(frame, [noseHull], -1, (0, 255, 0), 1)
+  23
+  22          # 5. Detect Mouth
+  21          mouth = shape[mStart:mEnd]
+  20          mouthHull = cv2.convexHull(mouth)
+  19          cv2.drawContours(frame, [mouthHull], -1, (0, 255, 0), 1)
+  18
+  17          # 6. Detect Jaw
+  16          jaw = shape[jStart:jEnd]
+  15          jawHull = cv2.convexHull(jaw)
+  14          cv2.drawContours(frame, [jawHull], -1, (0, 255, 0), 1)
+  13
+  12          # 7. Detect Eyebrows
+  11          ebr = shape[ebrStart:ebrEnd]
+  10          ebrHull = cv2.convexHull(ebr)
+   9          cv2.drawContours(frame, [ebrHull], -1, (0, 255, 0), 1)
+   8          ebl = shape[eblStart:eblEnd]
+   7          eblHull = cv2.convexHull(ebl)
+   6          cv2.drawContours(frame, [eblHull], -1, (0, 255, 0), 1)
+   5
+   4      cv2.putText(frame,'Number of Faces : ' + str(len(rects)),(40, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, 155, 1)
+   3
+   2      # When everything is done, release the capture
+   1      cv2.imwrite("released_faces.jpg", frame)
+ 236      return face_reports
```

Рис.4.13 “визначення эмоцій”

Далі ми складаємо звіти по обличчю та його емоційний стан. За допомогою нашої моделі даних ми отримуємо такий відсоток емоцій агресивності, щастя, радості, здивування, смутку.

```
    prediction_result = np.argmax(prediction)
    prediction_result

    # Rectangle around the face
    cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

    cv2.putText(frame, "Face #{}".format(i + 1), (x - 10, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    for (j, k) in shape:
        cv2.circle(frame, (j, k), 1, (0, 0, 255), -1)

    # 1. Add prediction probabilities
    #cv2.putText(frame, "----------------",(40,100 + 180*i), cv2.FONT_HERSHEY_SIMPLEX, 0.5, 155, 0)
    emotional_report = "Emotional report : Face #" + str(i+1),(40,120 + 180*i)
    angry = "Angry : " + str(round(prediction[0][0],3))
    disgust = "Disgust : " + str(round(prediction[0][1],3))
    fear = "Fear : " + str(round(prediction[0][2],3))
    happy = "Happy : " + str(round(prediction[0][3],3))
    sad = "Sad : " + str(round(prediction[0][4],3))
    surprise = "Surprise : " + str(round(prediction[0][5],3))
    neutral = "Neutral : " + str(round(prediction[0][6],3))

    face_reports += f"{emotional_report}, {angry} {disgust}, {happy}, {sad}, {surprise}, {neutral}"
    # 2. Annotate main image with a label
    if prediction_result == 0 :
        cv2.putText(frame, "Angry",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    elif prediction_result == 1 :
        cv2.putText(frame, "Disgust",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    elif prediction_result == 2 :
        cv2.putText(frame, "Fear",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    elif prediction_result == 3 :
        cv2.putText(frame, "Happy",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    elif prediction_result == 4 :
        cv2.putText(frame, "Sad",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    elif prediction_result == 5 :
        cv2.putText(frame, "Surprise",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
    else :
        cv2.putText(frame, "Neutral",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

    # 3. Eye Detection and Blink Count
    leftEye = shape[lStart:lEnd]
    rightEye = shape[rStart:rEnd]

    # Compute Eye Aspect Ratio
    leftEAR = eye_aspect_ratio(leftEye)
    rightEAR = eye_aspect_ratio(rightEye)
    ear = (leftEAR + rightEAR) / 2.0
```

Рис.4.14 “створення репорту”

Далі добавляємо різноманітну інформацію, а також наш результат.

Наприклад, обводимо наші очі:

```
        # 1. Add prediction probabilities
        cv2.putText(frame, "----------------",(40,100 + 180*i), cv2.FONT_HERSHEY_SIMPLEX, 0.5, 155, 0)
        cv2.putText(frame, "Emotional report : Face #" + str(i+1),(40,120 + 180*i), cv2.FONT_HERSHEY_SIMPLEX, 0.5, 155, 0)
        cv2.putText(frame, "Angry : " + str(round(prediction[0][0],3)),(40,140 + 180*i), cv2.FONT_HERSHEY_SIMPLEX, 0.5, 155, 0)
        cv2.putText(frame, "Disgust : " + str(round(prediction[0][1],3)),(40,160 + 180*i), cv2.FONT_HERSHEY_SIMPLEX, 0.5, 155, 0)
        cv2.putText(frame, "Fear : " + str(round(prediction[0][2],3)),(40,180 + 180*i), cv2.FONT_HERSHEY_SIMPLEX, 0.5, 155, 1)
        cv2.putText(frame, "Happy : " + str(round(prediction[0][3],3)),(40,200 + 180*i), cv2.FONT_HERSHEY_SIMPLEX, 0.5, 155, 1)
        cv2.putText(frame, "Sad : " + str(round(prediction[0][4],3)),(40,220 + 180*i), cv2.FONT_HERSHEY_SIMPLEX, 0.5, 155, 1)
        cv2.putText(frame, "Surprise : " + str(round(prediction[0][5],3)),(40,240 + 180*i), cv2.FONT_HERSHEY_SIMPLEX, 0.5, 155, 1)
        cv2.putText(frame, "Neutral : " + str(round(prediction[0][6],3)),(40,260 + 180*i), cv2.FONT_HERSHEY_SIMPLEX, 0.5, 155, 1)

        # 2. Annotate main image with a label
        if prediction_result == 0 :
            cv2.putText(frame, "Angry",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        elif prediction_result == 1 :
            cv2.putText(frame, "Disgust",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        elif prediction_result == 2 :
            cv2.putText(frame, "Fear",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        elif prediction_result == 3 :
            cv2.putText(frame, "Happy",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        elif prediction_result == 4 :
            cv2.putText(frame, "Sad",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        elif prediction_result == 5 :
            cv2.putText(frame, "Surprise",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        else :
            cv2.putText(frame, "Neutral",(x+w-10,y-10), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

        # 3. Eye Detection and Blink Count
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]

        # Compute Eye Aspect Ratio
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2.0

        # And plot its contours
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
```

Рис.4.15 “код”

```
    cv2.putText(frame,'Number of Faces : ' + str(len(rects)),(40, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, 155, 1)

    # When everything is done, release the capture
    cv2.imwrite("released_faces.jpg", frame)
    return face_reports
```

Рис.4.16 “збереження фотографії”

Додаємо ще число облич на фото і показуємо цей “фрейм”, в результаті ми повертаємося до логіки боту.

За допомогою нашої бібліотеки для роботи з Telegram bot API (pyTelegramBotApi) ми відправляємо нашому користувачеві емоційний репорт на кожне знайдене особа, а так само нову оброблену фотографію відкриваємо наше нове фото, потім відповідаємо на повідомлення в телеграмі.

```
6     photo = open('released_faces.jpg', 'rb')
7     bot.reply_to(message, result)
8     bot.send_photo(message.chat.id, photo)
```

Рис.4.17 “Відповідь на повідомлення”

Вигляд зі сторони користувача:

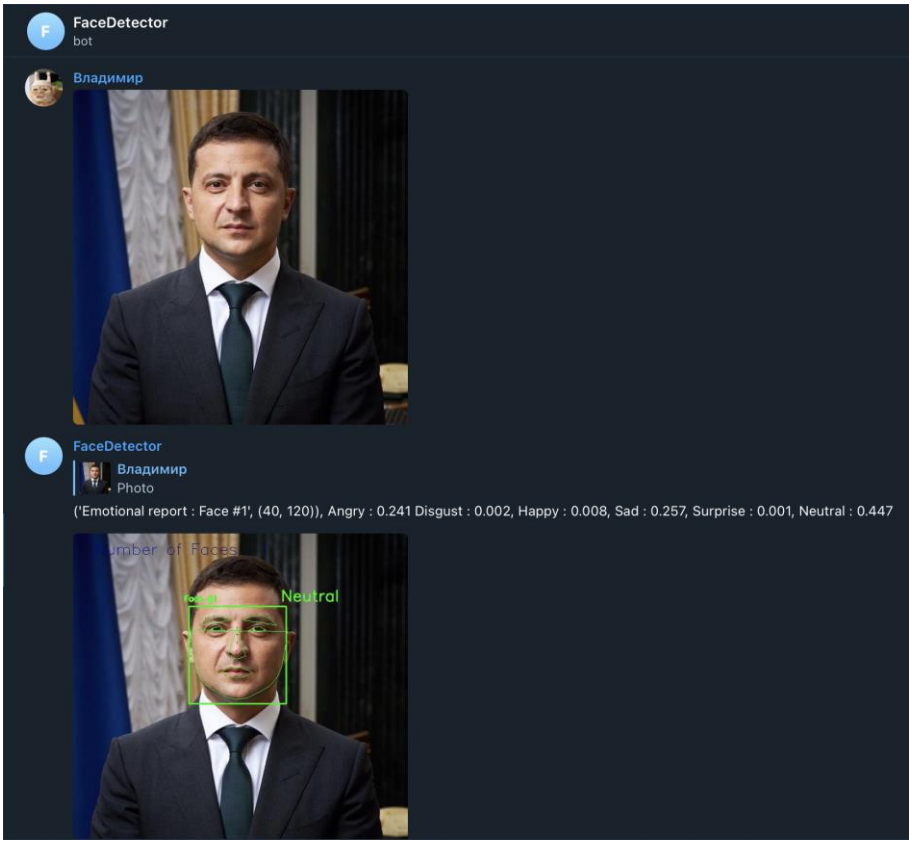

Рис.4.18 “приклад роботи”

Бот відправляє hash та фото, всередині цього hash ми маємо детальну інформацію стосовно нашого фото.

ВИСНОВКИ

В результаті ми створили телеграм бота, який розпізнає обличчя та емоції. Ми вивчили сучасну бібліотеку OpenCv, яка використовується в Machine Learning і яка має понад дві тисячі різноманітних алгоритмів. Також ми навчилися робити найпростіший комп'ютерний зір, що дозволяє нам розробити багато функціоналу. І в результаті для зручності наших користувачів ми зробили телеграм бота, що є сучасним рішенням та невеликою заміною front-end частини. Ми вивчили telegram bot API та навчилися користуватися бібліотекою для нього.

github репозиторій - https://github.com/ptiforka/emoji_detector_telegram_bot

СПИСОК ЛІТЕРАТУРИ І НАУКОВІ СТАТТІ

1. Документація OpenCv [Електронний ресурс] – Режим доступу : https://pypi.org/project/opencv-python/
2. Репозеторій OpenCv [Електронний ресурс] – Режим доступу : https://github.com/skvark/opencv-python
3. Документація Pandas [Електронний ресурс] – Режим доступу : https://khashtamov.com/ru/pandas-introduction/
4. Приклади розробки[Електронний ресурс] – Режим доступу https://www.pyimagesearch.com/start-here/
5. Приклади компьютерного зору [Електронний ресурс] – Режим доступу https://realpython.com/tutorials/computer-vision/
6. «Python. Карманный справочник», Марк Лутц
7. «Изучаем Python», Марк Лутц
8. Документація PyTelegramBotApi [Електронний ресурс] – Режим доступу https://github.com/eternnoir/pyTelegramBotAPI#pytelegrambotapi
9. Документація NumPy [Електронний ресурс] – Режим доступу https://numpy.org/
10. Розгляд методу AdaBoost [Електронний ресурс] – Режим доступу https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/

Додаток А

файл find_faces.py

```
### General imports ###
from __future__ import division

import numpy as np
import pandas as pd
import cv2

from time import time
from time import sleep
import re
import os

import argparse
from collections import OrderedDict

### Image processing ###
from scipy.ndimage import zoom
from scipy.spatial import distance
import imutils
from scipy import ndimage

import dlib

import tensorflow as tf
from imutils import face_utils
```

```
import requests

global shape_x
global shape_y
global input_shape
global nClasses

def process_image(image) :

    shape_x = 48
    shape_y = 48
    input_shape = (shape_x, shape_y, 1)
    nClasses = 7

    thresh = 0.25
    frame_check = 20

    def eye_aspect_ratio(eye):
        A = distance.euclidean(eye[1], eye[5])
        B = distance.euclidean(eye[2], eye[4])
        C = distance.euclidean(eye[0], eye[3])
        ear = (A + B) / (2.0 * C)
        return ear

    def detect_face(frame):

        #Cascade classifier pre-trained model
        cascPath = 'Models/Landmarks/face_landmarks.dat'
```

```
        faceCascade = cv2.CascadeClassifier(cascPath)

        #BGR -> Gray conversion

        frame = cv.imread("file_froN.jpg")
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        #Cascade MultiScale classifier
        detected_faces =
faceCascade.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=6,
                                        minSize=(shape_x, shape_y),
                                        flags=cv2.CASCADE_SCALE_IMAGE)
        coord = []

        for x, y, w, h in detected_faces :
            if w > 100 :
                sub_img=frame[y:y+h,x:x+w]
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0, 255,255),1)
                coord.append([x,y,w,h])

        return gray, detected_faces, coord

    def extract_face_features(faces, offset_coefficients=(0.075, 0.05)):
        gray = faces[0]
        detected_face = faces[1]

        new_face = []

        for det in detected_face :
```

```
        #Region dans laquelle la face est d√©tect√©e
        x, y, w, h = det
        #X et y correspondent √† la conversion en gris par gray, et w, h
correspondent √† la hauteur/largeur

        #Offset coefficient, np.floor takes the lowest integer (delete border of
the image)
        horizontal_offset = np.int(np.floor(offset_coefficients[0] * w))
        vertical_offset = np.int(np.floor(offset_coefficients[1] * h))

        #gray transforme l'image
        extracted_face = gray[y+vertical_offset:y+h, x+horizontal_offset:x-
horizontal_offset+w]

        #Zoom sur la face extraite
        new_extracted_face = zoom(extracted_face, (shape_x /
extracted_face.shape[0],shape_y / extracted_face.shape[1]))
        #cast type float
        new_extracted_face = new_extracted_face.astype(np.float32)
        #scale
        new_extracted_face /= float(new_extracted_face.max())

        new_face.append(new_extracted_face)

    return new_face


(lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
(rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
```

```
    (nStart, nEnd) = face_utils.FACIAL_LANDMARKS_IDXS["nose"]
    (mStart, mEnd) = face_utils.FACIAL_LANDMARKS_IDXS["mouth"]
    (jStart, jEnd) = face_utils.FACIAL_LANDMARKS_IDXS["jaw"]

    (eblStart, eblEnd) =
face_utils.FACIAL_LANDMARKS_IDXS["left_eyebrow"]
    (ebrStart, ebrEnd) =
face_utils.FACIAL_LANDMARKS_IDXS["right_eyebrow"]

    model = tf.keras.models.load_model('Models/EmotionXCeption/video.h5')
    face_detect = dlib.get_frontal_face_detector()
    predictor_landmarks  =
dlib.shape_predictor("Models/Landmarks/face_landmarks.dat")

    #Launch la capture video
    video_capture = cv2.VideoCapture(0)

    # Capture frame-by-frame
    frame = cv2.imread("file_from_tg.jpg")
    #ret, frame = video_capture.read()

    face_index = 0

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    rects = face_detect(gray, 1)
    #gray, detected_faces, coord = detect_face(frame)

    face_reports = ""
```

```
    for (i, rect) in enumerate(rects):

        shape = predictor_landmarks(gray, rect)
        shape = face_utils.shape_to_np(shape)

        # Identify face coordinates
        (x, y, w, h) = face_utils.rect_to_bb(rect)
        face = gray[y:y+h,x:x+w]

        #Zoom on extracted face
        face = zoom(face, (shape_x / face.shape[0],shape_y / face.shape[1]))

        #Cast type float
        face = face.astype(np.float32)

        #Scale
        face /= float(face.max())
        face = np.reshape(face.flatten(), (1, 48, 48, 1))

        #Make Prediction
        prediction = model.predict(face)
        prediction_result = np.argmax(prediction)
        prediction_result

        # Rectangle around the face
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

        cv2.putText(frame, "Face #{}".format(i + 1), (x - 10, y - 10),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
```

```
        for (j, k) in shape:
            cv2.circle(frame, (j, k), 1, (0, 0, 255), -1)

        # 1. Add prediction probabilities
        #cv2.putText(frame, "----------------",(40,100 + 180*i),
cv2.FONT_HERSHEY_SIMPLEX, 0.5, 155, 0)
        emotional_report = "Emotional report : Face #" + str(i+1),(40,120 + 180*i)
        angry = "Angry : " + str(round(prediction[0][0],3))
        disgust = "Disgust : " + str(round(prediction[0][1],3))
        fear = "Fear : " + str(round(prediction[0][2],3))
        happy = "Happy : " + str(round(prediction[0][3],3))
        sad = "Sad : " + str(round(prediction[0][4],3))
        surprise = "Surprise : " + str(round(prediction[0][5],3))
        neutral = "Neutral : " + str(round(prediction[0][6],3))

        face_reports += f"{emotional_report}, {angry} {disgust}, {happy}, {sad},
{surprise}, {neutral}"
        # 2. Annotate main image with a label
        if prediction_result == 0 :
            cv2.putText(frame, "Angry",(x+w-10,y-10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        elif prediction_result == 1 :
            cv2.putText(frame, "Disgust",(x+w-10,y-10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        elif prediction_result == 2 :
            cv2.putText(frame, "Fear",(x+w-10,y-10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        elif prediction_result == 3 :
```

```
            cv2.putText(frame, "Happy",(x+w-10,y-10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        elif prediction_result == 4 :
            cv2.putText(frame, "Sad",(x+w-10,y-10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        elif prediction_result == 5 :
            cv2.putText(frame, "Surprise",(x+w-10,y-10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
        else :
            cv2.putText(frame, "Neutral",(x+w-10,y-10),
cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)

        # 3. Eye Detection and Blink Count
        leftEye = shape[lStart:lEnd]
        rightEye = shape[rStart:rEnd]

        # Compute Eye Aspect Ratio
        leftEAR = eye_aspect_ratio(leftEye)
        rightEAR = eye_aspect_ratio(rightEye)
        ear = (leftEAR + rightEAR) / 2.0

        # And plot its contours
        leftEyeHull = cv2.convexHull(leftEye)
        rightEyeHull = cv2.convexHull(rightEye)
        cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
        cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)

        # 4. Detect Nose
        nose = shape[nStart:nEnd]
```

```
        noseHull = cv2.convexHull(nose)
        cv2.drawContours(frame, [noseHull], -1, (0, 255, 0), 1)

        # 5. Detect Mouth
        mouth = shape[mStart:mEnd]
        mouthHull = cv2.convexHull(mouth)
        cv2.drawContours(frame, [mouthHull], -1, (0, 255, 0), 1)

        # 6. Detect Jaw
        jaw = shape[jStart:jEnd]
        jawHull = cv2.convexHull(jaw)
        cv2.drawContours(frame, [jawHull], -1, (0, 255, 0), 1)

        # 7. Detect Eyebrows
        ebr = shape[ebrStart:ebrEnd]
        ebrHull = cv2.convexHull(ebr)
        cv2.drawContours(frame, [ebrHull], -1, (0, 255, 0), 1)
        ebl = shape[eblStart:eblEnd]
        eblHull = cv2.convexHull(ebl)
        cv2.drawContours(frame, [eblHull], -1, (0, 255, 0), 1)

    cv2.putText(frame,'Number of Faces : ' + str(len(rects)),(40, 40),
cv2.FONT_HERSHEY_SIMPLEX, 1, 155, 1)

    # When everything is done, release the capture
    cv2.imwrite("released_faces.jpg", frame)
    return face_reports

def main():
```

```
    show_webcam()

if __name__ == "__main__":
    main()
```

Файл main.py

```
import telebot
import time
import sys
import live_face
import requests
from flask import Flask, request

server = Flask(__name__)
API_TOKEN = "1818910210:AAFHAb9ffRA8k7akf48dOBUpwwpS6lK5cgY"
bot = telebot.TeleBot(API_TOKEN, parse_mode=None) # You can set
parse_mode by default. HTML or MARKDOWN

@bot.message_handler(commands=['start', 'help'])
def send_welcome(message):
    bot.reply_to(message, "Send image for detection faces   –æ")

@bot.message_handler(content_types=['photo'])
def handle_docs_audio(message):
    file_id = message.json["photo"][-1]["file_id"]
    file_info = bot.get_file(file_id)
```

```
    url = 'https://api.telegram.org/file/bot{0}/{1}'.format(API_TOKEN,
file_info.file_path)
    file =
requests.get('https://api.telegram.org/file/bot{0}/{1}'.format(API_TOKEN,
file_info.file_path))
    response = requests.get(url, stream=True)

    with open("file_from_tg.jpg", "wb") as handle:
        for data in response.iter_content():
            handle.write(data)
    result = find_face.process_image("file_from_tg.jpg")
    time.sleep(1)
    photo = open('released_faces.jpg', 'rb')
    bot.reply_to(message, result)
    bot.send_photo(message.chat.id, photo)

@server.route("/")
def webhook():
    bot.remove_webhook()
    bot.set_webhook(url = SERVER_URL + TOKEN)
    return "!", 200

bot.polling()
```

файл requirements.txt

```
absl-py==0.1.13
aiogram==2.3
aiohttp==3.6.2
aioredis==1.3.0
```

appnope==0.1.0
APScheduler==3.6.1
argon2-cffi==20.1.0
astor==0.6.2
astroid==2.2.5
async-generator==1.10
async-timeout==3.0.1
attrs==19.3.0
autopep8==1.4.4
Babel==2.7.0
backcall==0.2.0
beautifulsoup4==4.8.0
bleach==1.5.0
bs4==0.0.1
certifi==2019.9.11
cffi==1.14.3
chardet==3.0.4
Click==7.0
cycler==0.10.0
decorator==4.4.2
defusedxml==0.6.0
entrypoints==0.3
fastconf==1.0.0
gast==0.2.0
greenlet==0.4.15
grpcio==1.11.0
gTTS==2.0.3
gTTS-token==1.1.3
hiredis==1.0.0

html5lib==0.9999999
idna==2.8
idna-ssl==1.1.0
importlib-metadata==2.0.0
ipykernel==5.3.4
ipython==7.16.1
ipython-genutils==0.2.0
ipywidgets==7.5.1
isort==4.3.17
jedi==0.17.2
Jinja2==2.11.2
jsonschema==3.2.0
jupyter==1.0.0
jupyter-client==6.1.7
jupyter-console==6.2.0
jupyter-core==4.6.3
jupyterlab-pygments==0.1.1
lazy-object-proxy==1.3.1
Markdown==2.6.11
MarkupSafe
matplotlib==2.2.0
mccabe==0.6.1
mistune==0.8.4
mpmath==1.1.0
msgpack==0.6.2
multidict==4.5.2
nbclient==0.5.0
nbconvert==6.0.6
nbformat==5.0.7

nest-asyncio==1.4.0

nose==1.3.7

notebook==6.1.4

numpy

olefile==0.44

opencv-contrib-python==4.4.0.46

opencv-python==4.4.0.44

pandas==1.1.2

pandocfilters==1.4.2

parso==0.7.1

peewee==3.13.1

peewee-migrations==0.3.18

pexpect==4.8.0

pickleshare==0.7.5

Pillow==4.3.0

prometheus-client==0.8.0

prompt-toolkit==3.0.7

protobuf==3.5.2.post1

ptyprocess==0.6.0

pycodestyle==2.5.0

pycparser==2.20

Pygments==2.7.1

pylint==2.3.1

pynvim==0.4.0

pyparsing==2.2.0

pyrsistent==0.17.3

python-dateutil==2.8.1

pytz==2019.3

PyYAML==5.1.2

pyzmq==19.0.2

qtconsole==4.7.7

QtPy==1.9.0

requests==2.21.0

scipy>=1.0

Send2Trash==1.5.0

six==1.12.0

soupsieve==1.9.2

SQLAlchemy==1.3.11

sympy==1.6.2

tensorboard==1.7.0

tensorflow==2.5.0

termcolor==1.1.0

terminado==0.9.1

testpath==0.4.4

tornado==6.0.4

traitlets==4.3.3

typed-ast==1.3.4

typing-extensions==3.7.4

tzlocal==2.0.0

urllib3==1.24.1

virtualenv==16.7.7

wcwidth==0.2.5

Werkzeug==0.15.3

widgetsnbextension==3.5.1

wrapt==1.11.1

yarl==1.3.0

zipp==3.2.0

cmake

argparse

dlib

imutils

tika

wave

nltk

gensim

seaborn

scikit-learn

dill

keras

wordcloud

dlib

pyTelegramBotAPI