МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМУВАННЯ ТА МАТЕМАТИКИ

До захисту допускається
завідувач кафедри _____
Лифар В.О.
«____»_____ 20__ р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Дослідження загроз інформаційної безпеки систем зберігання даних

Освітній рівень“Магістр”
Спеціальність 126 “Інформаційні системи та технології”

Науковий керівник роботи: _____ (підпис) Марченко Д.М. (ініціали, прізвище)

Студент: _____ (підпис) Панфілов Є.В. (ініціали, прізвище)

Група: ІСТ-20дм

Сєвєродонецьк 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет 122 Інформаційних технологій та електроніки

Кафедра Програмування та математики

Освітній рівень Магістр

Спеціальність 126 “Інформаційні системи та технології”
(шифр і назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри _____

_____ В.О.Лифар

«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Панфілову Євгену Віталійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Дослідження загроз інформаційної безпеки систем зберігання даних

керівник проекту (роботи) Марченко Дмитро Миколайович, д.т.н., проф.
(прізвище, м.’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «30» 11 2021 р. № 182/15.16

2. Строк подання студентом роботи 20.12.2021

3. Вихідні дані до роботи Матеріали науково-дослідної практики, науково-методична література; дані інтернет-мережі;

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Визначення проблем захисту інформації в мережі інтернет, методи захисту інформації, розробка нового методу захисту, висновки

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) Електронна презентація

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Розробка технічного завдання	8.11.2021-13.11.2021	
2	Аналіз літератури з досліджуваної проблеми	14.11.2021-19.11.2021	
3	Аналіз технічних засобів	20.11.2021-24.11.2021	
4	Аналіз існуючих систем	25.11.2021-30.11.2021	
4	Реалізація системи захисту	1.12.2021-11.12.2021	
5	Оформлення пояснювальної записки та презентації	12.12.2021-19.12.2021	

Студент _____ Панфілов Є.В.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Марченко Д.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Панфілов Є.В. Дослідження загроз інформаційної безпеки систем зберігання даних.

Метою роботи є аналіз проблем захисту інформації у мережі інтернет, аналіз існуючих методів захисту інформації, створення власного методу захисту від DeepFake технології.

Об’єкт дослідження - методи та аналіз існуючих проблем захисту інформації, створення рішення для захисту однієї із проблем інформаційної загрози - DeepFake.

Робота присвячена складанню концепту виявлення підробленого відео – DeepFake, з подальшим розширенням та створенням веб сайту для захисту людей та компаній від даної глобальної проблеми.

Було проведено дослідження існуючих методів і засобів захисту від подробленої інформації і відео та розроблене рішення, що дозволяє швидко виявити підроблене відео створене за допомогою технології DeepFake.

ABSTRACT

Panfilov EV Investigation of information security threats to storage systems.

The purpose of the work is to analyze the problems of information protection on the internet, analysis if existing methods of information protection, creating your own method of protection against DeepFake technology.

The object of research is methods and analysis of existing information security problems, creating a solution to protect one of the problems of information threat - DeepFake.

The work is devoted to the concept of detecting fake video - DeepFake, with the further expansion and creation of a website to protect people and companies from this global problem.

A study of existing methods and means of protection against counterfeit information and video was conducted and a solution was developed that allows you to quickly detect fake video created using DeepFake technology.

ЗМІСТ

ВСТУП

Актуальність. У зв'язку з розвитком інформаційних технологій та комп'ютеризацією економіки одним із найважливіших питань у діяльності компанії стає забезпечення інформаційної безпеки.

Інформаційна безпека – це збереження та захист інформації, а також її найважливіших елементів, у тому числі системи та обладнання, призначені для використання, заощадження та передачі цієї інформації. Іншими словами, це набір технологій, стандартів та методів управління, які необхідні для захисту інформаційної безпеки.

Об’єкт дослідження: процес створення системи розпізнавання обличчя для запобігання створенню підроблених відео.

Предмет дослідження: побудова системи розпізнавання підроблених відео за допомогою технології DeepFake.

Мета дослідження: забезпечення інформаційної безпеки – захистити інформаційні дані та підтримуючу інфраструктуру від випадкового чи навмисного втручання, що може спричинити втрату даних або їх несанкціоновану зміну.

Задачі дослідження:

1. Ознайомитися з принципом створення штучного інтелекту, його можливостей і актуальності застосування у даному проекту.
2. Сформувати список актуальних та потрібних технологій і мов, для розробки власної системи для розпізнавання підроблених відео.
3. Розробити концепт системи розпізнавання DeepFake відео у мережі.

АНАЛІТИЧНИЙ ОГЛЯД

1.1. Контроль інформаційної безпеки.

Вибір та впровадження відповідних видів контролю безпеки допоможе організації знизити ризик до допустимих рівнів. Виділяють такі види контролю:

- **Адміністративний**. Адміністративний вид контролю складається із затверджених процедур, стандартів та принципів. Він формує рамки для ведення бізнесу та управління людьми. Закони та нормативні акти, створені державними органами, також є одним із видів адміністративного контролю. Інші приклади адміністративного контролю включають політику корпоративної безпеки, паролів, найму та дисциплінарні заходи.
- **Логічний**. Логічні засоби управління (що ще називають технічними засобами контролю) базуються на захисті доступу до інформаційних систем, програмного забезпечення, паролях, брандмауерів, інформації для моніторингу та контролю доступу до систем інформації.
- **Фізичний**. Це контроль середовища робочого місця та обчислювальних засобів (опалення та кондиціювання повітря, димові та пожежні сигналізації, протипожежні системи, камери, барикади, огородження, замки, двері та ін.). [1]

1.2. Загрози інформаційної безпеки.

Загрози інформаційної безпеки можна розділити на:

- Природні (катаклізми, незалежні від людини: пожежі, урагани, повінь, удари блискавки тощо.).
- Штучні, які також поділяються на:
  - ненавмисні (вчиняються людьми з необережності чи незнання);

- навмисні (хакерські атаки, протиправні дії конкурентів, помста співробітників та ін.).

- Внутрішні (джерела загрози, що знаходяться усередині системи).
- Зовнішні (джерела загроз за межами системи)

Так як загрози можуть по-різному впливати на інформаційну систему, їх поділяють на пасивні (ті, що не змінюють структуру та зміст інформації) та активні (ті, що змінюють структуру та зміст системи, наприклад, застосування спеціальних програм). [2]

Найбільш небезпечні навмисні загрози, які все частіше поповнюються новими різновидами, що пов'язано насамперед із комп'ютеризацією економіки та поширенням електронних транзакцій. Зловмисники не стоять на місці, а шукають нові шляхи отримати конфіденційні дані та завдати втрат компанії.

Щоб убезпечити компанію від втрати коштів та інтелектуальної власності, необхідно приділяти більше уваги інформаційній безпеці. Це можливо завдяки засобам захисту в особі передових технологій.

1.3. Засоби захисту інформаційної безпеки.

Засоби захисту інформаційної безпеки - це набір технічних пристроїв, пристроїв, приладів різного характеру, які перешкоджають витоку інформації та виконують її функцію.

Засоби захисту інформації поділяються на:

- Організаційні. Це сукупність організаційно-технічних (забезпечення комп'ютерними приміщеннями, налаштування кабельної системи та ін.) та організаційно-правових (законодавча база, статут конкретної організації) коштів.
- Програмні. Ті програми, які допомагають контролювати, зберігати та захищати інформацію та доступ до неї.
- Технічні (апаратні). Це технічні види пристроїв, які захищають інформацію від проникнення та витоку.

- Змішані апаратно-програмні. Виконують функції як апаратних, і програмних засобів.

У зв'язку зі стрімким розвитком IT, все більш частими кібератаками, комп'ютерними вірусами та іншими загрозами, що з'являються найбільш поширеними і затребуваними на сьогоднішній день є програмні засоби захисту інформації.

Антивірусні програми - програми, які борються з комп'ютерними вірусами та відновлюють заражені файли.

Хмарний антивірус (CloudAV) – одне з хмарних рішень інформаційної безпеки, що застосовує легке програмне забезпечення агента на захищеному комп'ютері, вивантажуючи велику частину аналізу інформації в інфраструктуру провайдера. CloudAV – це також рішення для ефективного сканування вірусів на пристрої з невисокою обчислювальною потужністю для виконання самих сканувань. Деякі зразки хмарних антивірусних програм – це Panda Cloud Antivirus, Crowdstrike, Cb Defense та Immunet.

DLP (Data Leak Prevention) рішення – це захист від витоку інформації. Запобігання витоку даних (DLP) є набір технологій, спрямованих на запобігання втраті конфіденційної інформації, яка відбувається на підприємствах по всьому світу. Успішна реалізація цієї технології потребує значної підготовки та ретельного технічного обслуговування. Підприємства, які бажають інтегрувати та впроваджувати DLP, мають бути готовими до значних зусиль, які, якщо вони будуть виконані правильно, можуть значно знизити ризик для організації.

Криптографічні системи – перетворення інформації таким чином, що її розшифрування стає можливим лише за допомогою певних кодів або шифрів (DES – Data Encryption Standard, AES – Advanced Encryption Standard). Криптографія забезпечує захист інформації та іншими корисними програмами, включаючи покращені методи автентифікації, дайджести повідомлень, цифрові підписи та зашифровані мережеві комунікації. Старі,

менш безпечні програми, наприклад Telnet і протокол передачі файлів (FTP), повільно замінюються безпечнішими програмами, такими як Secure Shell (SSH), які використовують зашифровані мережеві комунікації. Бездротовий зв'язок може бути зашифрований за допомогою таких протоколів, як WPA/WPA2 або старіший (і менш безпечний) WEP. Дротові комунікації (такі як ITU-T G.hn) захищені з використанням AES для шифрування та X.1035 для автентифікації та обміну ключами. Програмні програми, такі як GnuPG або PGP, можна використовувати для шифрування інформаційних файлів та електронної пошти.

Міжмережні екрани (брандмауери або файрволи) – пристрої контролю доступу до мережі, призначені для блокування та фільтрації мережного трафіку. Брандмауери зазвичай класифікуються як мережеві чи хост-сервери. Мережеві брандмауери на базі мережі розташовані на шлюзових комп'ютерах LAN, WAN та інтрамережах. Це або програмні пристрої, що працюють на апаратних засобах загального призначення, або апаратні пристрої брандмауера. Брандмауери пропонують інші функції для внутрішньої мережі, яку вони захищають, наприклад, є сервером DHCP або VPN для цієї мережі. Одним із найкращих рішень як для малих, так і для великих підприємств є міжмережні екрани CheckPoint.

VPN (Virtual Private Network). Віртуальна приватна мережа (VPN) дає можливість визначити та використовувати для передачі та отримання інформації приватну мережу в рамках загальнодоступної мережі. Таким чином, програми, що працюють за VPN, є надійно захищеними. VPN дозволяє підключитися до внутрішньої мережі на відстані. За допомогою VPN можна створити спільну мережу для територіально віддалених підприємств. Що стосується окремих користувачів мережі – вони також мають свої переваги використання VPN, тому що можуть захищати власні дії за допомогою VPN, а також уникати територіальних обмежень та використовувати проксі-сервери, щоб приховати своє місцезнаходження.

Proxy-server (Проксі-сервер) – це певний комп'ютер або комп'ютерна програма, яка є зв'язуючою ланкою між двома пристроями, наприклад, як комп'ютер та інший сервер. Проксі-сервер можна встановити на одному комп'ютері разом із сервером брандмауера, або на іншому сервері. Плюси проксі-сервера в тому, що його кеш може служити всім користувачам. Інтернет-сайти, які є найчастіше запитуваними, найчастіше знаходяться в кеші проксі, що, безперечно, зручно для користувача. Фіксування своїх взаємодій проксі-сервером є корисною функцією для виправлення неполадок.

Системи моніторингу та управління інформаційною безпекою, SIEM. Щоб виявляти і реагувати на загрози інформаційної безпеки, що виникають, використовується рішення SIEM, яке виконує збір та аналіз подій з різних джерел, таких як міжмережеві екрани, антивіруси, IPS, оперативні системи тощо. Завдяки системі SIEM у компаній з'являється можливість централізовано зберігати журнали подій та корелювати їх, визначаючи відхилення, потенційні загрози, збої у роботі IT-інфраструктури, кібератаки тощо.

1.4. Проблема додатку «Дія»

Все частіше і частіше в Telegram з'являються канали, де молодим людям, які не досягли повноліття, пропонується додаток, що імітує сервіс "Дія". Підлітки можуть завантажити його за невелику плату собі на смартфони, вписати необхідні дані про себе, відкоригувавши вік і додавши фото, і користуватися ним, наприклад, при покупці алкоголю або цигарок (продаж яких заборонений особам, які не досягли 18 років).

Дані з "Дія" рідко перевіряють

Часто ті, хто має перевіряти паспортні дані, просто не морочаться. Навіть іноді поліція не сканує ці QR-коди. Просто подивилися додатку, порівняли з фото, та на перевірка закінчується. Що вже говорити про касирів супермаркетів, вони часто не мають ні часу, ні бажання перевіряти дані з "Дія" на справжність.

Іноді касири для перевірки справжності просять покупців перегорнути відкриту "Дію" вліво-вправо, щоб переконатися, що це не скріншот. Однак на сьогоднішній день існує маса фейкових сервісів, які візуально нічим не відрізняються від сьогодення, і навіть частково функціонують як справжній сервіс. Ставка робиться на те, що QR-коди ніхто не перевіряє.

Перевірка QR-кодів Covid-сертифікатів там, де перевірка паспорта не потрібна, тобто поза вокзалами та аеропортами, — заняття безглузде. Простий приклад: охоронець у магазині просить вас пред'явити QR-код. Ви його показуєте, і охоронець отримує інформацію про те, що ви вакцинувалися і що ваше ім'я – Іванов Іван Іванович. Але як охоронець може перевірити достовірність цієї інформації, якщо у вас немає паспорта (а носити його з собою ви не зобов'язані), це питання безпеки залишається відкритим.

Збільшується кількість підробок "Дія" та Covid-сертифікатів

Якщо є попит, є і пропозиція. І поки це так, шахраї штампуватимуть фейки заради наживи, а люди купуватимуть їх, щоб не вакцинуватися, але мати необхідний документ і купувати ті товари, які хочуть. Фейкових сервісів досить велика кількість, і їх з'являтиметься дедалі більше (тільки в одному Telegram-каналі "Фейк Дія" понад 20 тис. користувачів, які заплатили гроші за встановлення підробки).

З QR-кодами для автентифікації Covid-сертифікатів - та ж історія безпеки. Деякі клініки вводять дані з помилками до електронної системи охорони здоров'я (ЕСОЗ), і потім під час перевірки результати отримують некоректні. Деякі медики навмисно генерують "ліві" коди для того, щоб потім їх продати. І це проблема не лише України. Як нещодавно було помічено, на такому шахрайстві потрапили німецькі фармацевти.

Корінь проблеми безпеки в тому, що при розробці сервісів спочатку неправильно було поставлено завдання. До уваги не брали те, що знайдуться люди, які зможуть обійти обмеження, сподівалися, що все працюватиме правильно і всі користуватимуться продуктом правильно. [3]

1.5. Технологія Deepfake як загроза інформаційній безпеці

Deepfake (від англ. Deep learning - «глибоке навчання» і fake - «фальшивий») - реалістична маніпуляція аудіо- та відеоматеріалами за допомогою штучного інтелекту. Ця технологія змушує людину говорити те, чого вона не вимовляла, і робити те, чого вона ніколи не робила.

Технологія Deepfake сягає корінням у далекі дев'яності. У той час подібними інструментами мали лише експерти зі спецефектів у кіноіндустрії. Згодом технологію було доопрацьовано в інтернет-спільноті, і програмне забезпечення для створення дипфейків з'явилося у відкритому доступі. Останнім часом технологія Deepfake привертає велику увагу через її використання у фінансових махінаціях, розіграшах та фальшивих новинах (fake news).

Deepfake застосовує можливості штучного інтелекту для синтезу людського зображення: об'єднує кілька знімків, на яких людина відображена з різних ракурсів і з різним виразом обличчя, і робить відеопотік. Аналізуючи фотографії, спеціальний алгоритм навчається тому, як виглядає і може рухатись людина. При цьому працюють дві нейромережі. Перша з них генерує зображення, а друга відповідає за пошук відмінностей між ними та справжніми зразками. Якщо друга нейромережа виявляє підробку, зображення відправляється назад першою для поліпшення.

Deepfake працює за допомогою відкритих алгоритмів машинного навчання та бібліотек, що дозволяє досягти найвищої якості контенту. Нейросеть отримує зображення з бібліотеки та навчається за допомогою роликів на відеохостингах. Штучний інтелект тим часом зіставляє фрагменти вихідних портретів з тим, що є на відео, і виходить правдоподібний матеріал.

1.6. Чим небезпечна технологія Deepfake

У 2018 році з'явилася перша публічна програма для підміни осіб під назвою FakeApp, яка відкрила дипфейк-інструмент простому обивателю. Набори для створення такого контенту знаходяться відтоді у вільному

доступі та легкі у освоєнні. Будь-яка людина, яка має доступ до інтернету, вільний час, цілі та мотивацію, може в режимі реального часу створювати фальшивий контент та наповнювати їм канали соціальних мереж.

Перші підроблені відеоролики почали з'являтися ще 2017 року. Спочатку більшість дипфейк-контенту виглядали просто кумедно: це були в основному аматорські відеофайли, створені з використанням безкоштовних інструментів і що містили особи знаменитостей, накладені на порнографічні записи. Однак за кілька років технологія розвинулася настільки, що створювані з її допомогою матеріали стали переконливими. Широку популярність вона здобула після того, як один із користувачів сайту Reddit з нікнеймом Deepfakes розмістив у себе на сторінці відео, де обличчя президента вдало замінили на обличчя актриси Галь Гадот.

Дипфейки є серйозною загрозою, оскільки такого роду контент є, по суті, інформаційною атакою. IT-аналітики заявляють, що технологія Deepfake може стати найнебезпечнішою у цифровій сфері за останні десятиліття. З поширенням дипфейків з'явилися випадки дискредитації відомих людей, фотографій яких багато в мережі «Інтернет». У світі політики технологія Deepfake може бути використана як зброя проти окремих діячів та цілих партій, щоб маніпулювати громадським сприйняттям, впливати на вибори чи навіть на фондовий ринок.

Як приклад гучного дипфейку можна навести опубліковане 2018 року фальшиве відео з екс-президентом США Бараком Обамою, де він нібито ображає нинішнього главу американської держави. Відеоролик був зроблений за допомогою програм FakeApp та Adobe After Effects.

У середині 2019 року в Мережі з'явилося сфабриковане відео з Марком Цукербергом, який нібито відверто описав поточний стан справ із персональними даними людей. У тому ж році в інтернеті опублікували дипфейк-відео зі спікером палати представників конгресу США Ненсі Пелосі. Автор ролика за допомогою технологій штучного інтелекту змінив мову Пелосі так, що вона погано вимовляла слова, і користувачі, які

переглянули відео, вважали, що політик перебуває у стані алкогольного сп'яніння. Все це переросло у великий скандал, і лише через деякий час було доведено, що Пелосі мова була згенерована комп'ютером.

У червні 2019 року користувачі побачили алгоритм DeepNude, який навчився роздягати дівчат на фотографіях. Якщо раніше подібні програми просто підміняли обличчя, то новий алгоритм самостійно перемальовує тіло героїні знімка. Достатньо було завантажити зображення у програму та кілька секунд почекати.

Описані події змусили уряд та промисловість Сполучених Штатів виявляти та обмежувати незаконне використання дипфейків. У жовтні 2019 року губернатор Каліфорнії Гевін Ньюсом підписав документ AB 730, відомий як «антидіпфейк-законопроект», в якому йдеться про заборону розповсюдження дипфейк-відео під час передвиборчої кампанії у 2020 році.

На додаток до всього перерахованого, існують і технології підробки голосів. Голосові дипфейки також використовуються кіберзлочинцями — зокрема для того, щоб переконати співробітників будь-якої компанії здійснити несанкціоновані дії. Далі у статті ми розповімо про це докладніше.

1.7. Deepfake як загроза для інформаційної безпеки бізнесу

Крім політичних інформаційних воєн Deepfake створює і ризики у сфері інформаційної безпеки корпоративного сектора. Дипфейки виявились непоганим інструментом у руках шахраїв.

Наприклад, цільовий фішинг (англ. spearphishing) спрямований на те, щоб обдурити конкретних співробітників певної компанії та змусити їх виконати вручну якусь операцію - оплату підробленого платіжного доручення, відправлення документів, ручне скидання облікових даних користувача для кіберзлочинця. Такі дії, як правило, важче виявити технічно (з точки зору засобів захисту інформації), оскільки електронний лист не містить жодних підозрілих посилань або вкладень і зазвичай використовується в поєднанні з атакою Business E-mail Compromise (BEC),

коли хакери отримують контроль над корпоративним обліковим записом електронної пошти та можуть надсилати листи з легітимних адрес. За даними ФБР, за останні три роки атаки типу BEC обійшлися компаніям по всьому світу більш ніж 26 млрд доларів США.

Дипфейки мають можливість перевантажувати подібні операції. Наприклад, співробітник компанії отримує електронний лист від генерального директора з проханням вжити деяких фінансових заходів, потім йому надходить текстове послання з мобільного номера того ж топ-менеджера, а потім і аудіоповідомлення, де голос генерального директора називає працівника на ім'я і посилається на попередні розмови з ним. Швидше за все, співробітник не задумається про те, чи це дипфейк чи ні.

Як приклад такої атаки можна навести інцидент, який стався у серпні 2019 року в одній компанії, що працює у сфері енергетики. Було встановлено, що кіберзлочинець використав технологію створення дипфейків для здійснення шахрайських дій на суму 220 000 євро. Зловмисник вийшов на зв'язок із фінансовим відділом як керуючий із Німеччини. Він змодельованим голосом попросив терміново переказати гроші на рахунок в Угорщині, і бізнес-партнери не мали жодного приводу не вірити тому, хто телефонує.

1.8. Як боротися з DeepFake

Проблема дипфейків видається дуже складною. Оскільки вони створюються за допомогою штучного інтелекту, для боротьби з ними слід використати щось аналогічне.

В даний час ринок ІБ не пропонує спеціальних технологій та рішень для захисту від дипфейків. Тим не менш, певні кроки в цьому напрямі здійснюються. Наприклад, Facebook, Microsoft та дослідники з американських університетів розпочали розробку інструментів для виявлення подібних підробок, а також у співпраці з Amazon вирішили провести конкурс Deepfake Detection Challenge на найкращий спосіб визначення дипфейк-відео.

Управління перспективних досліджень проектів Міністерства оборони США (DARPA) запустило алгоритм виявлення підроблених відеоматеріалів. У статті "In Ictu Oculi: Exposing AI Generation Fake Face Videos by Detecting Eye Blinking" розповідається про те, як аналіз частоти моргання може допомогти виявити дипфейк. Ідея така: зазвичай у відкритому доступі важко знайти фотографії людини в момент моргання, так що нейронної мережі просто нема на чому вчитися генерувати подібні кадри. Крім того, у оригіналу та підробки можуть відрізнятися деякі примітні частини обличчя (підборіддя, брови, вилиці, вуса та борода, ластовиння та родимі плями); будь-яка невідповідність - свідчення дипфейка.

Для того, щоб мінімізувати ризики цільових фішингових атак за допомогою дипфейків у корпоративному середовищі, необхідно інформувати користувачів про нові типи шкідливої активності та бути напоготові в ситуаціях, коли поведінка співрозмовника в телефонній розмові або голосовому повідомленні видається незвичайною. Крім того, рекомендується:

- використовувати багатофакторну автентифікацію співробітників, електронний підпис для захисту повідомлень електронної пошти
- відстежувати факти наявності програм для створення дипфейків на комп'ютерах користувачів та спроби пошуку таких додатків у мережі «Інтернет», звертати особливу увагу на подібних працівників та проводити щодо них внутрішні перевірки
- мінімізувати кількість каналів комунікацій компанії
- забезпечити узгоджене поширення інформації
- обмежити фото та відеоконтент за участю керівних осіб підприємства
- розробити план реагування на дезінформацію (за аналогією з інцидентами безпеки)

- організувати централізований моніторинг каналів та звітність
- всередині компанії та для зв'язку з контрагентами застосовувати практику введення усних паролів, кодових слів або контрольних питань, відповідь на які відома лише двом сторонам
- стежити за новими способами виявлення дипфейків та методами боротьби з ними

1.9. Приклади DeepFake-додатків та інструментів

Безкоштовне програмне забезпечення DeepFaceLab є одним з найбільш популярних програм для створення дипфейків і використовує нові нейронні мережі для заміни осіб у відео. Програма розміщена на GitHub, вихідний код викладено користувачем iperov. За словами розробника, понад 95% глибоких підробок відео створюється за допомогою DeepFaceLab.

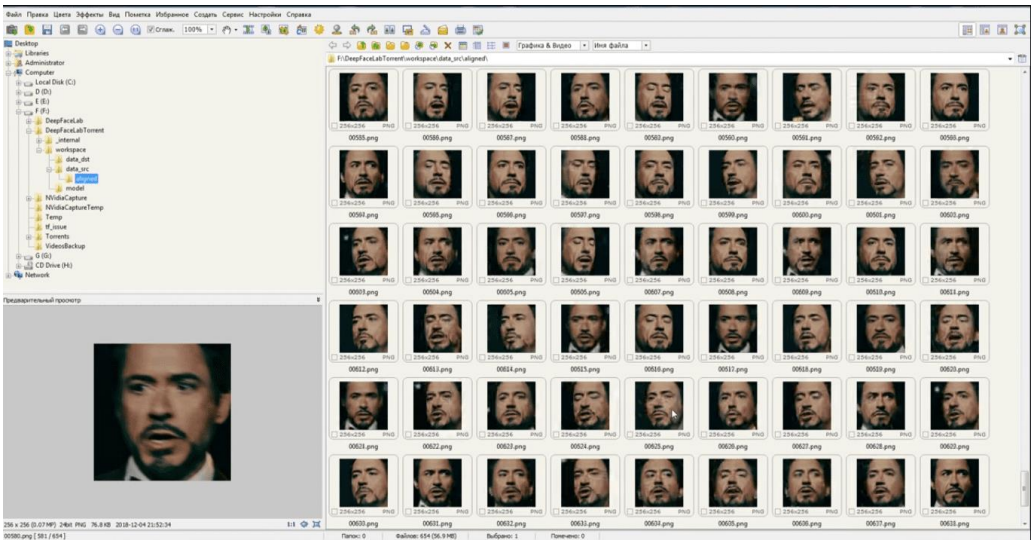

Рис. 1.1. Інтерфейс програми DeepFaceLab.

Doublicat. Розробники сервісу Reflect у грудні 2019 року представили безкоштовний додаток Doublicat для Android та iOS. Програма дозволяє реалістично міняти обличчя на двох фотографіях. Для створення зображення потрібно зробити селфі, а потім вибрати якийсь із знаменитих GIF-мемов і вставити туди своє обличчя. На початку січня програма Doublicat стала вірусною. Про нього написали десятки західних видань.

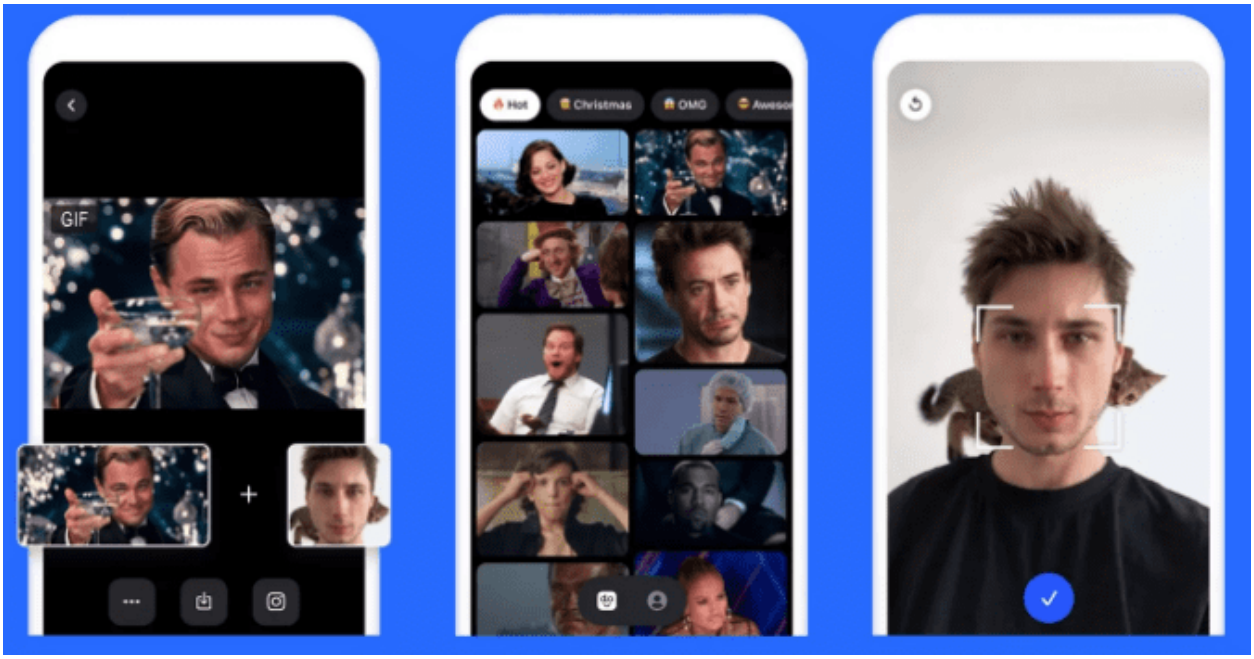

Рис. 1.2. Обробка зображення для створення GIF-дипфейку в Doublicat.

Безкоштовний китайський дипфейк-додаток Zao для Android та iOS може замінити обличчя відомого героя або актора на обличчя користувача в будь-якому уривку популярного фільму, серіалу, шоу або кліпу, і при цьому не знадобиться вдаватися в тонкощі та нюанси монтажу та обробки відео. Zao дозволяє модулювати голоси знаменитостей та накладати своє обличчя на тіло актора у сцені. Додаток надає безліч відеокліпів та вбрання, а також великі можливості для вивчення. Найчастіше результат виглядають не ідеальним, але кумедним.

Багато користувачів вважали, що програма може виявитися небезпечною, оскільки в політиці конфіденційності зазначено, що компанія-розробник Momo вправі застосовувати згенеровані відеоролики у власних цілях (тобто їй належить весь створений програмою контент).

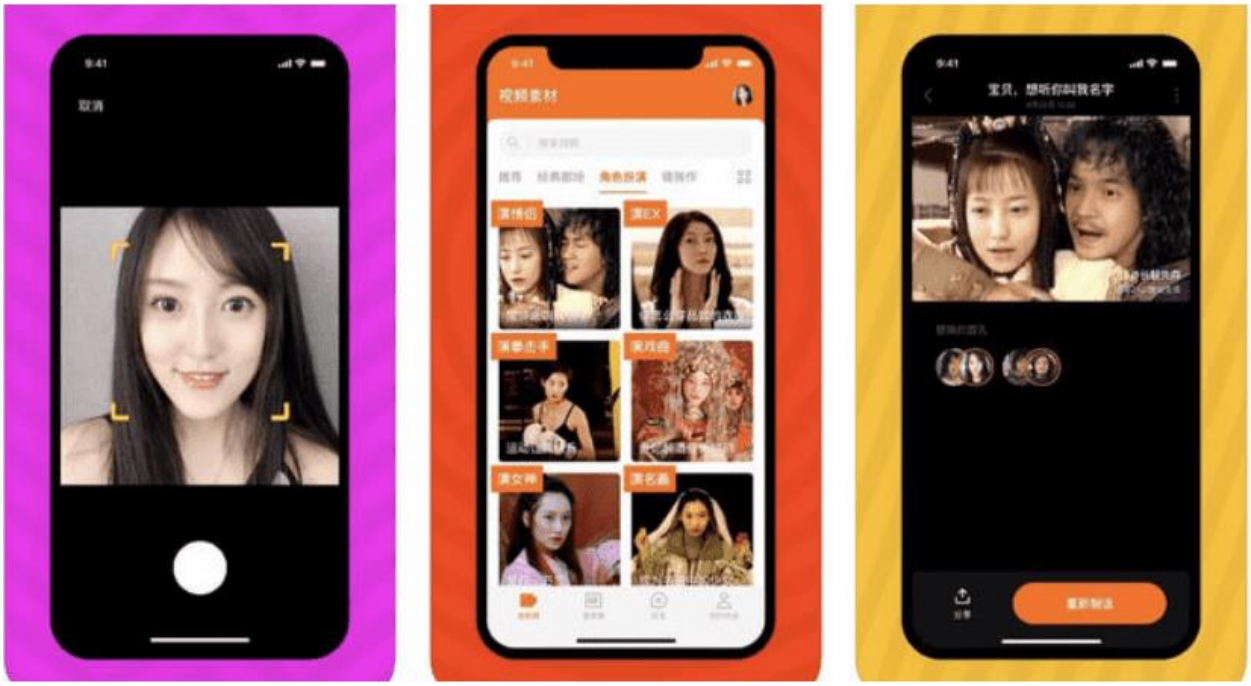

Рис. 1.3. Обробка зображення для створення відео-дипфейку в Zao.

З поширенням технології Deepfake з'явилася небезпека дискредитації будь-якого користувача, чий знімок чи голос є у загальному доступі. Технології створення дипфейків постійно вдосконалюються, і вже через кілька років можна очікувати появи таких підробок, що настільки природно виглядають або звучать, що виявити їх можна буде тільки після ретельного аналізу з використанням штучного інтелекту.

При цьому найперспективніші напрями використання дипфейків — політичні війни та шахрайство. Крім того, з урахуванням постійного вдосконалення технологій дипфейки можуть також нашкодити і судовій практиці — щодо довіри до аудіо- та відеоматеріалів доказової бази (диктофонних записів, файлів відеореєстраторів тощо). [4]

1.10. Способи вияву DeepFake

Останнім часом ми постійно стикаємося з технологією Deepfake, яка дозволяє підміняти обличчя на фото чи відео на будь-які інші: наприклад, ролики з нібито Томом Крузом вийшли настільки якісними, що в них повірили мільйони людей. І тепер дослідники з Університету Буффало

знайшли спосіб, який дозволяє з 94% ймовірністю виявити такі Deepfake-фото.

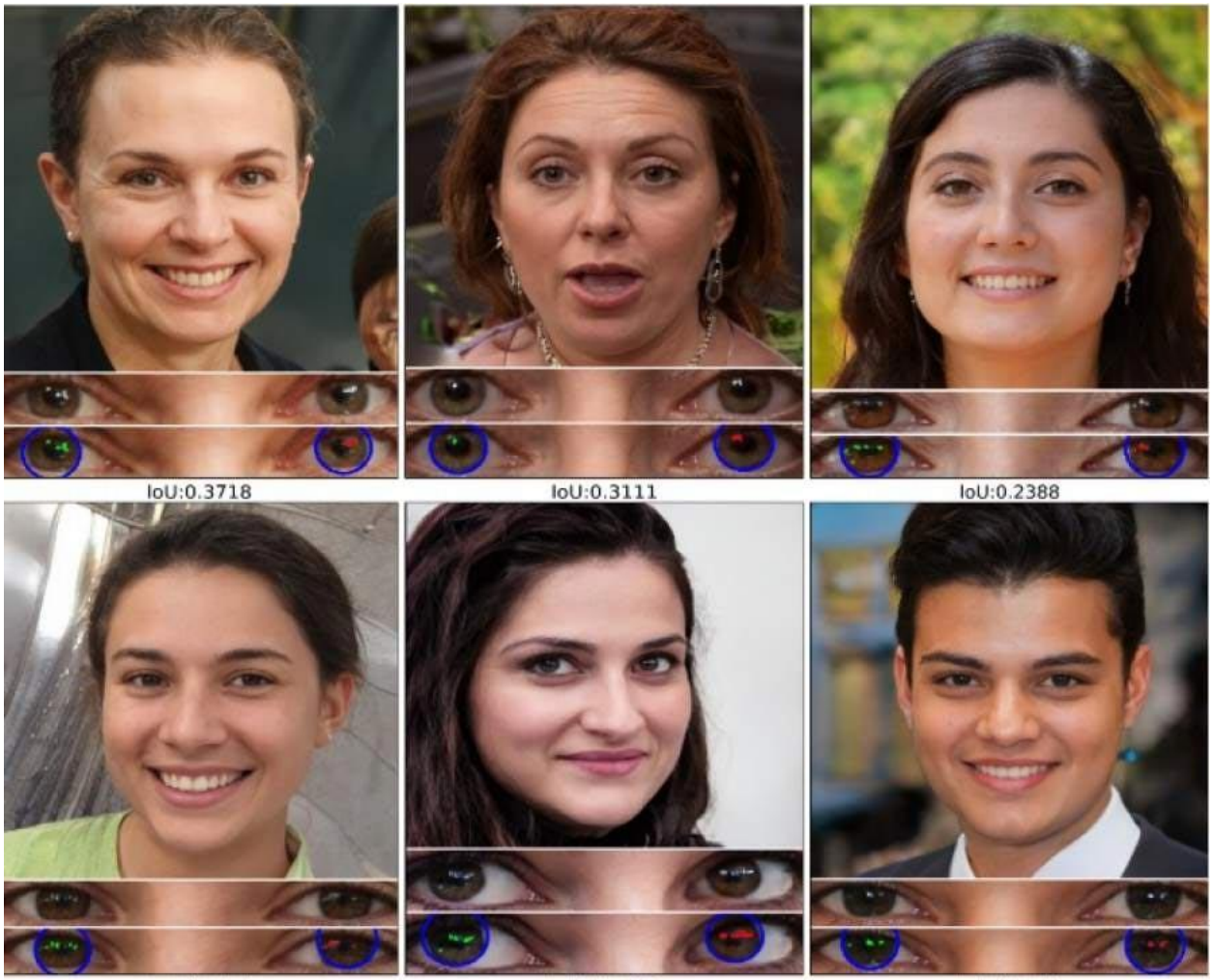

Рис. 1.4. Спосіб DeepFake виявлення по відблиску в очах.

Вся справа в тому, що на більшості фото (якщо вони тільки не зняті в темряві) можна виявити відображення в очах, які, зрозуміло, будуть однакові на реальних фотографіях. А ось на Deepfake-фото це не так, тому що нейромережа при створенні дипфейку використовує велику кількість цифрових фотографій зі своїх баз, що призводить до різних відбиття в очах.

Дослідники випробували свій алгоритм на нейромережі StyleGAN2, яка вміє створювати «несправжніх» людей, і при хорошому висвітленні точність виявлення дипфейків становила значні 94%. На жаль, цей показник виявляється відчутно гіршим на темних фото, особливо з низькою роздільною здатністю, яка не дозволяє нормально розглянути очі.

Також потрібно пам'ятати, що і в реальності відображення в очах можуть відрізнятися, якщо людина, наприклад, має чубок, або одне з очей при фотографуванні опинилося в тіні. Однак, все ще такий алгоритм є непоганим рішенням для боротьби з повальною модою на Deepfake-фото.

1.11. Google веде активну боротьбу з DeepFake

Google вийшла на активну боротьбу із дипфейками. IT-корпорація створить базу фальшивих відео, за допомогою якої алгоритми навчаться автоматично визначати фейкові записи.

Розвиток технологій ставить перед суспільством нові питання безпеки. Йдеться не лише про фінансовий чи інформаційний хакінг: репутація та приватне життя користувачів теж можуть опинитися під загрозою. Однією з основних інструментів, створюють передумови для нової кіберзлочинності, є дипфейки – відеоролики, створювані з допомогою штучного інтелекту. Використовуючи можливості машинного навчання, алгоритм аналізує безліч фотографій, а потім, використовуючи отримані дані, замінює особи на відеороликах.

Щоб зрозуміти серйозність того, що відбувається, досить згадати дипфейк, опублікований BuzzFeed, де "Барак Обама" ображає колишнього президента США Дональда Трампа. Відео, створене за допомогою програми Fakeapp та графічного редактора Adobe After Effects, виглядає реалістично та наочно демонструє небезпечний потенціал дипфейків. Діапазон їх використання у перспективі просто величезний – від фейкових новин до репутаційних роликів та підроблених стартапів.

Найбільша небезпека дипфейків полягає в тому, що технології створення таких відео знаходяться у відкритому доступі, а результат інколи неможливо ідентифікувати візуально. У спробі зупинити загрозу, що росте, фахівці Google вирішили бити дипфейки їхньою ж зброєю. Розпізнаватиме відео, створені за допомогою штучного інтелекту, буде сам штучний інтелект.

Для цього програмісти завантажили в датасет Deep Fake Detection 363 оригінальні відео за участю добровольців, і на їх основі створили понад 3 тисячі дипфейків. Цієї бази буде достатньо для того, щоб розпочати навчання алгоритмів. Надалі датасет буде поповнюватися новим матеріалом для вивчення – це допоможе максимально автоматизувати процес розпізнавання та звести кількість помилок нанівець, заявляють фахівці Google.

Штучний інтелект все краще працює із зображеннями, отже, все краще справляються зі своєю роботою дипфейк-алгоритми. Фахівці побоюються, що скоро не можна буде на око визначити фейкові ролики, тому розвивають проекти для їхнього автоматичного розпізнавання.

Для датасета Deep Fake Detection компанії створили 363 оригінальні відео, в яких знялися добровольці. На їх основі зробили понад 3 тисячі дипфейк-відео із залученням алгоритмів Deepfakes, Face2Face, FaceSwap та NeuralTextures. Набір Deep Fake Detection став частиною проекту FaceForensics, який вивчає роботу алгоритмів для заміни осіб.

Компанії будуть поповнювати датасет. Доступний він лише дослідникам та за спеціальним запитом. Раніше Facebook для боротьби з фейковими відео об'єднався з Microsoft та вченими з Partnership on AI. Сторони оголосили про конкурс Deepfake Detection Challenge на найкраще рішення щодо виявлення deepfakes. [5]

МОДЕЛЬ ПРОЕКТУ ТА ПОСТАНОВКА ЗАДАЧІ

Вибухове зростання глибокого фейкового відео та його незаконне використання є серйозною загрозою для демократії, справедливості та громадської довіри. У зв'язку з цим зростає попит на, виявлення та знешкодження таких відео.

Виявлення глибоких підроблених відео за допомогою виявлення артефактів деформації обличчя, використовуючи підхід для виявлення артефактів шляхом порівняння створених областей обличчя та їх прилеглих областей із спеціальною моделлю згорткової нейронної мережі.

Розкриття підроблених відео, створених штучним інтелектом, шляхом виявлення моргання очей описує новий метод розкриття підроблених Deep Fake відео, створених за допомогою моделей глибокої нейронної мережі. Метод заснований на виявленні моргання очей у відео, що є фізіологічним сигналом, який погано представлений у синтезованих підроблених відео. Метод оцінюється за контрольними показниками наборів даних виявлення миготіння очей і показує багатообіцяючу продуктивність при виявленні відео, створених за допомогою програмного забезпечення Deep Fake на основі Deep Neural Network.

Метод використовує лише відсутність моргання як підказку для виявлення. Однак для виявлення глибокої підробки необхідно враховувати деякі інші параметри, як-от стискання зубів, зморшки на обличчі тощо.

Існує багато інструментів для створення глибокої підробки, але для виявлення глибокої підробки навряд чи є доступний інструмент. Наш підхід до виявлення Deep Fake стане великим внеском у уникнення просочування Deep Fake у всесвітній мережі. Ми надамо веб-платформу для користувача, щоб завантажити відео та класифікувати його як підробку чи справжнє. Цей проект можна розширити від розробки веб-платформи до плагіна для браузера для автоматичного виявлення глибоких підробок. Навіть велика програма, як-от WhatsApp, Facebook може інтегрувати цей проект зі своєю програмою для легкого попереднього виявлення Deep Fake перед відправкою

іншому користувачеві. Однією з важливих цілей є оцінка його продуктивності та прийнятності з точки зору безпеки, зручності, точності та надійності. Наш метод зосереджений на виявленні всіх типів Deep Fake, як-от моргання Deep Fake, зморшки Deep Fake та границі світла Deep Fake. рис.2 представляє просту системну архітектуру пропонованої системи.

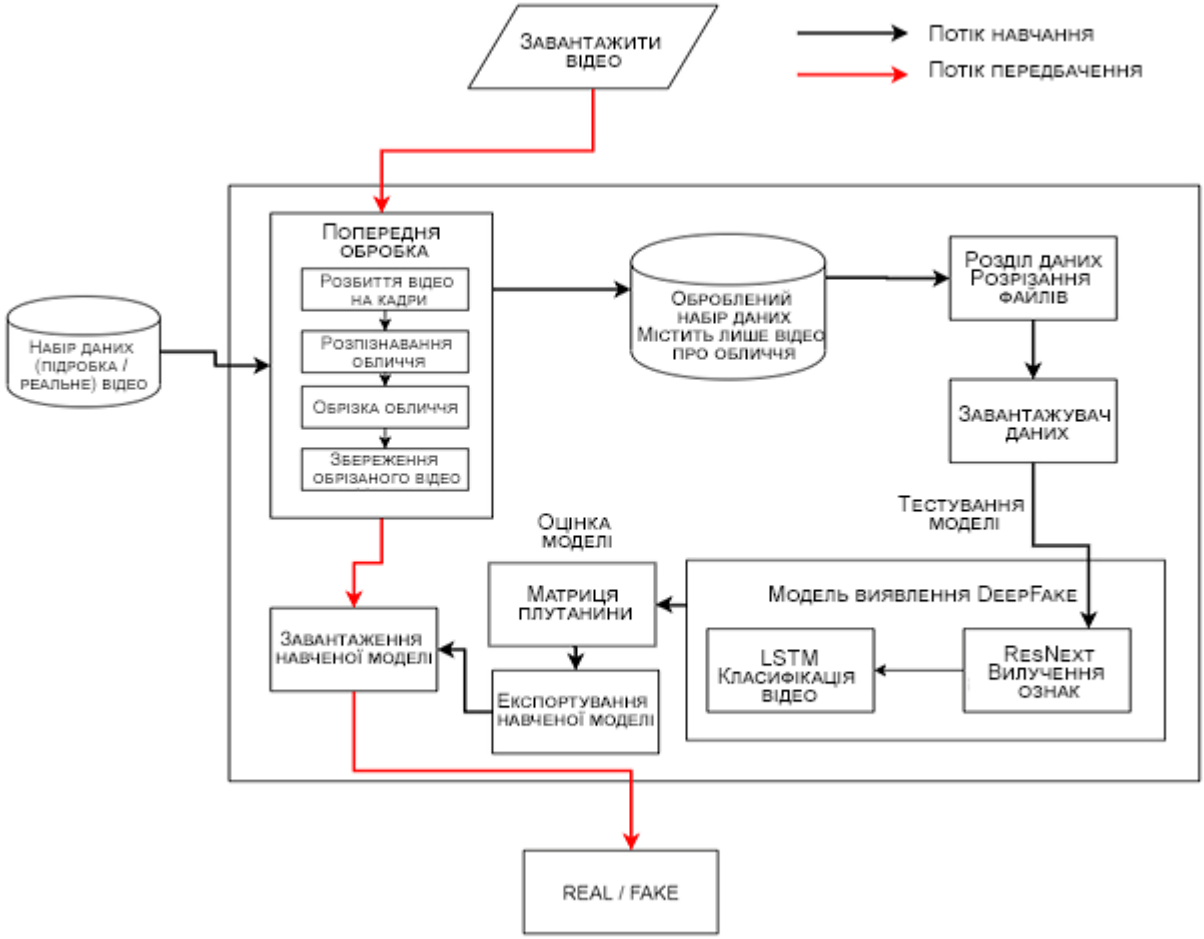

Рис. 2. Системна архітектура виявлення DeepFake.

Метод заснований на спостереженнях, що поточний алгоритм Deep Fake може генерувати лише зображення обмеженої роздільної здатності, які потім необхідно додатково трансформувати, щоб відповідати обличчям, які потрібно замінити у вихідному відео.

Цей проект спрямований на виявлення дипфейків відео з використанням таких методів глибокого навчання, як ResNext та LSTM. Ми досягли виявлення глибоких підробок за допомогою трансферного навчання,

в якому попередньо навчена CNN ResNext використовується для отримання вектора ознак, а потім шар LSTM навчається з використанням цих функцій.

Ми використовуємо змішаний набір даних, який складається з відео з різних джерел наборів даних, таких як YouTube, FaceForensics++, набір даних для виявлення глибоких підробок (DeepFake Detection Challenge Dataset). Наш нещодавно підготовлений набір даних містить 50% оригінального відео та 50% маніпуляційних фейкових відео. Набір даних розділений на 70% поїздів і 30% тестових наборів.

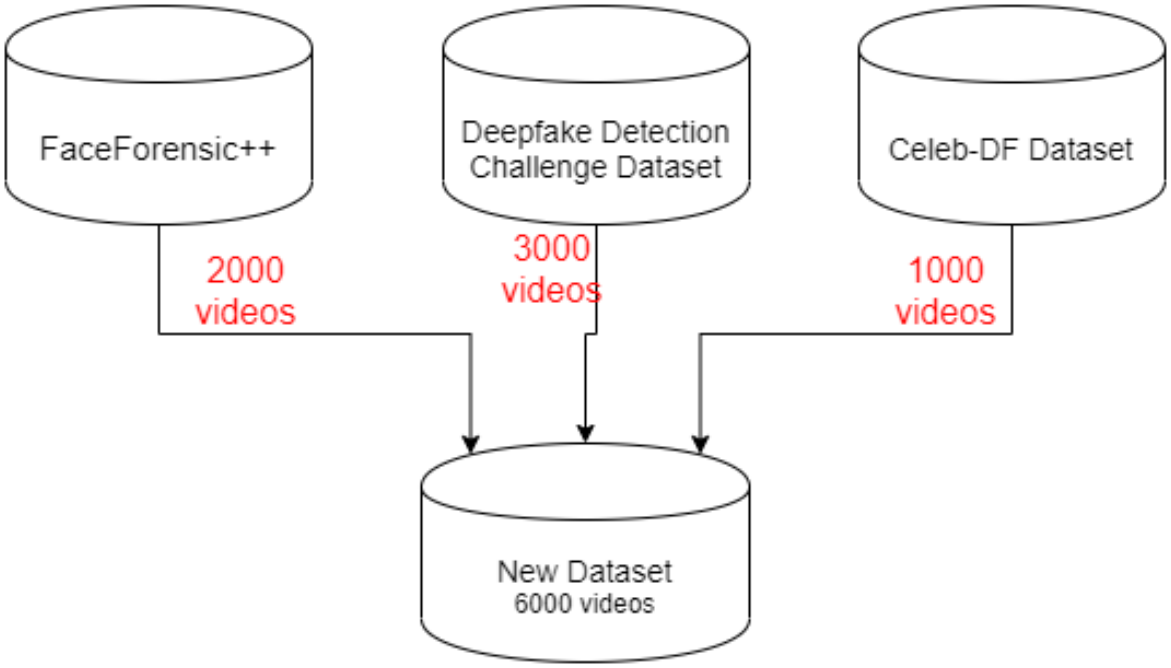

Рис. 2.1 Змішаний набір даних для обробки.

Попередня обробка набору даних включає поділ відео на кадри. Далі слід розпізнавання обличчя та обрізання кадру з виявленим обличчям. Для підтримки рівномірності кількості кадрів обчислюється середнє значення відео набору даних і створюється новий набір даних обрізаного обробленого обличчя, що містить кадри, рівні середньому. Кадри, у яких немає граней, ігноруються під час попередньої обробки.

Для обробки 10-секундного відео зі швидкістю 30 кадрів в секунду, тобто всього 300 кадрів, буде потрібно багато обчислювальної потужності.

Тому в експериментальних цілях ми пропонуємо використовувати лише перші 100 кадрів для навчання моделі. [6]

Нове відео передається навченій моделі для передбачення. Нове відео також попередньо оброблено, щоб передати формат навченої моделі. Відео розбивається на кадри з подальшим обрізанням обличчя, і замість збереження відео в локальному сховищі обрізані кадри безпосередньо передаються навченій моделі для виявлення.

Рис. 2.2 Попередня обробка відео.

Результатом моделі стане те, чи є відео глибоким фейком, чи реальне відео, а також впевненість моделі. Один із прикладів показаний на малюнку 2.3.

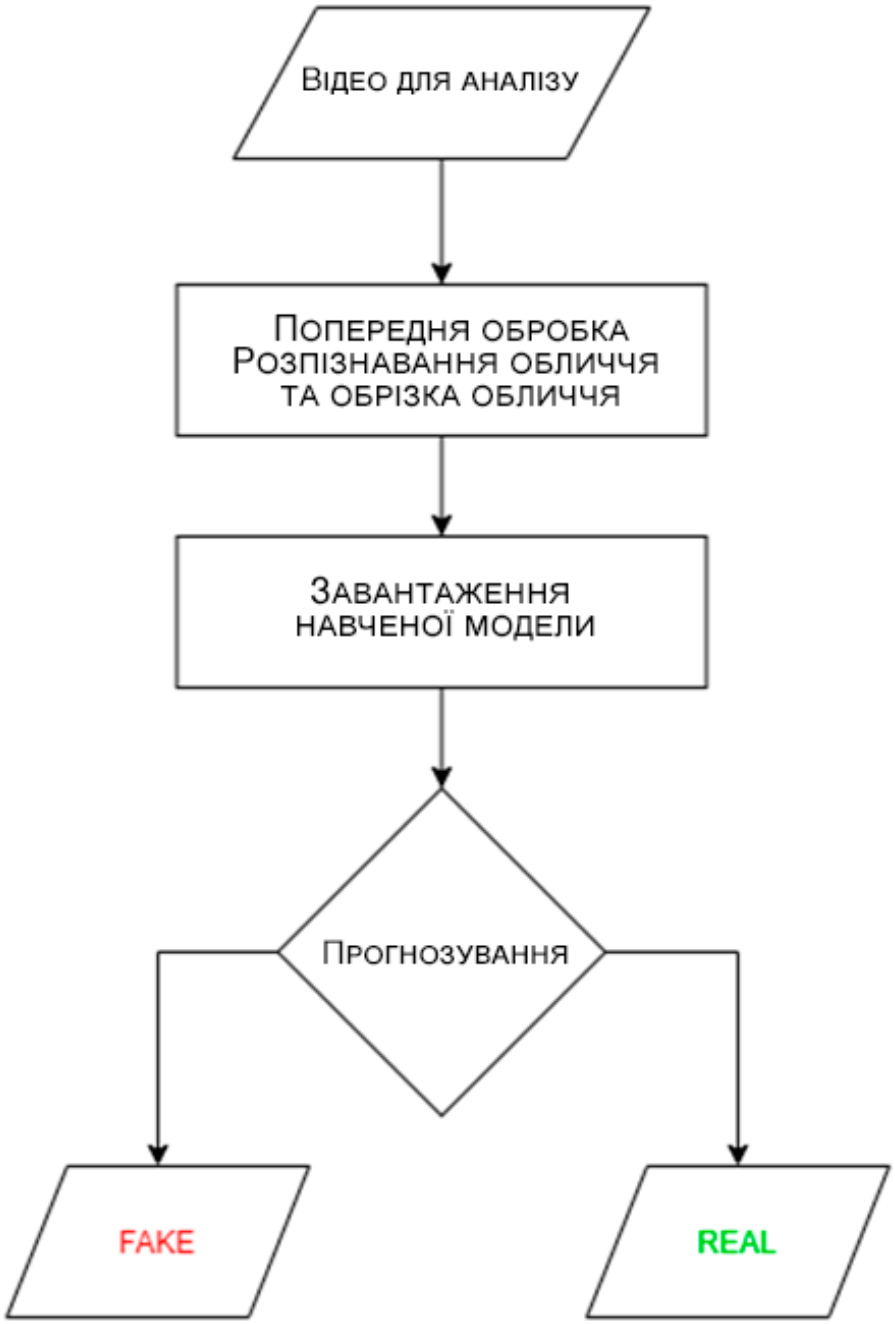

Рис. 2.3. Схема методу прогнозування.

ВИБІР МЕТОДІВ ТА РЕАЛІЗАЦІЯ ПОСТАВЛЕННИХ ЗАДАЧ

Даний проект буде реалізований в основі мовою програмування Python, популярного фреймворку Django, дуже важливим модулем OpenCV, а також базою даних SQLite.

Мова програмування Python.

Python - високорівнева мова програмування загального призначення, орієнтована на підвищення продуктивності розробника і читання коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Рис. 3. Логотип мови програмування Python.

Python підтримує структурне, об'єктно-орієнтоване, функціональне, імперативне і аспектно-орієнтоване програмування. Основні архітектурні риси - динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень, високорівневі структури даних. Підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети.

Еталонної реалізацією Python є інтерпретатор CPython, що підтримує більшість активно використовуваних платформ [7]. Він поширюється під вільною ліцензією Python Software Foundation License, що дозволяє

використовувати його без обмежень в будь-яких додатках, включаючи пропрієтарні.

Python - активно розвивається мова програмування, нові версії з додаванням / зміною мовних властивостей виходять приблизно раз в два з половиною роки. Язик не піддавався офіційній стандартизації, роль стандарту де-факто виконує CPython, що розробляється під контролем автора мови. На даний момент Python займає третє місце в рейтингу TIOBE з показником 8,5%.

Особливості Python.

Python, як і будь-яка інша мова програмування, має свої відмінні риси. Отже, можна виділити наступні:

- Кросплатформеність. Python – це інтерпретована мова, його інтерпретатори існують для багатьох платформ. Тому з запуском його на будь-який ОС не повинно виникнути проблем.
- З Python доступна величезна кількість сервісів, середовищ розробки, і фреймворків.
- Можливість підключити бібліотеки, написані на C. Це дозволяє підвищити ефективність, поліпшити швидкість роботи.
- Наявність самих різних джерел інформації про Python. Не важно буде знайти відповідь на питання, які виникають, так існую багато безкоштовної літератури, навчальних відео-посібників, готових початкових кодів та шаблонів для роботи у відкритому доступі.

Переваги Python щодо інших мов програмування.

Python легко конкурує з іншими мовами програмування, так як має безліч переваг. По-перше, це зрозумілий і простий синтаксис. Особливо добре він для новачків. Можна створити цікаві програми, і при цьому не доведеться сидіти тижнями, вивчаючи складний синтаксис.

Динамічна типізація - це одне з головних достоїнств мови Python. Для новачків це можливість спростити написання коду і уникнути безлічі

фатальних помилок і багів в роботі. Також в Python немає операторних дужок, з розставленими яких найчастіше виникають складності.

За швидкістю виконання програм, коли це стосується великих повномасштабних проектів, Python, звичайно ж, не лідер. Тут мінусом є і автоматичне керування пам'яттю, і повна динамічна типізація. Python поступається значно таких мов як Java, C, C ++, але і в той же час з легкістю дає фору JavaScript, Ruby, PHP. Підключення бібліотек, написаних на C і можливість попередньої компіляції коду в байт-код - все це дозволяє поліпшити швидкодію.

Python - це мова програмування, затребувана сьогодні і з великим потенціалом в майбутнє. Сьогодні ринок праці потребує кваліфікованих фахівців зі знаннями Python.

Фреймворк Django.

Django — це програмний каркас з багатими можливостями, що підходить для розробки складних сайтів та веб-застосунків, написаний мовою програмування Python.

Рис. 3.1 Логотип Фреймворку Django.

Django — фреймворк для веб-застосунків мовою Python. Один з основних принципів фреймворку - DRY (don't repeat yourself). Веб-системи на

Django будуються з одного або кількох програм, які рекомендується робити відчужуваними та підключається. Це одна з помітних архітектурних відмінностей цього фреймворку від інших (наприклад, Ruby on Rails). Також, на відміну від багатьох інших фреймворків, обробники URL Django конфігуруються явно (за допомогою регулярних виразів), а не автоматично задаються зі структури контролерів.

Django проектувався для роботи під управлінням Apache (з модулем mod_python) та з використанням PostgreSQL як база даних. В даний час, окрім PostgreSQL, Django може працювати з іншими СУБД: MySQL (MariaDB), SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere та Oracle. Для роботи з базою даних Django використовує власний ORM, у якому модель даних описується класами Python, і генерується схема бази даних.

Архітектура Django схожа на "Модель-Представлення-Контролер" (MVC). Контролер класичної моделі MVC приблизно відповідає рівню, який Django називається Представлення (View), а презентаційна логіка Представлення реалізується в Django рівнем Шаблонів (Templates). Через це рівневу архітектуру Django часто називають "Модель-Шаблон-Представлення" (MTV).

Спочатку розробка Django велася для забезпечення зручнішої роботи з ресурсами новин, що досить сильно позначилося на архітектурі: фреймворк надає ряд засобів, які допомагають у швидкій розробці веб-сайтів інформаційного характеру. Наприклад, розробнику не потрібно створювати контролери та сторінки для адміністративної частини сайту, у Django є вбудований додаток для керування вмістом, який можна включити в будь-який сайт, зроблений на Django, і який може керувати відразу кількома сайтами на одному сервері. Адміністративна програма дозволяє створювати, змінювати та видаляти будь-які об'єкти наповнення сайту, протоколюючи всі вчинені дії, та надає інтерфейс для управління користувачами та групами (з пооб'єктним призначенням прав).

Веб-фреймворк Django використовується в таких великих та відомих сайтах, як Instagram, Disqus, Mozilla, The Washington Times, Pinterest, lamoda та ін.

Деякі можливості та переваги Django:

- Використання Python як мови програмування. Це не ідеальна і швидка мова програмування, проте вона досить проста з синтаксичної точки зору, що автоматично дає низький поріг входження. Також ми маємо всю його потужність мета-програмування, велику бібліотеку класів, відмінну документацію і досить компактний та інтуїтивно зрозумілий синтаксис.
- Чудова документація. Документація Django відома своєю зручністю та поширеністю, в той же час дуже компактною, але зрозумілою інформацією – безліч прикладів, пояснень, і найголовніше – відкритий вихідний код, який дуже добре написаний.
- Вбудований ORM (Object-relational mapper). Звичайно, є більш гнучкі та потужні бібліотеки, що забезпечують проектування реляційних даних в об'єкти, але вирішує свої завдання Django ORM відмінно. Найголовніше, в більшості випадків абсолютно не потрібне використання SQL-синтаксису у виразах, що автоматично знижує ризик появи SQL-injection вразливості.
- Підтримка MTV (Model-Template-View). Даний патерн проектування дуже близький до класичного MVC, і найголовніше, що він дозволяє це добре відокремлювати бізнес-логіку від дизайну.

На базі Django розроблено досить багато готових рішень, що розповсюджуються під вільною ліцензією, серед яких системи управління інтернет-магазинами, універсальні системи управління змістом, а також більш вузькоспрямовані проекти. [8]

База даних SQLite.

SQLite - це вбудована бібліотека, яка реалізує автономний, безсерверний, нульовий конфігурації, механізм транзакції СУБД SQL. Це база даних, яка налаштована на нуль, що означає, як інші бази даних, які вам не потрібно налаштовувати у вашій системі.

Рис. 3.2. Логотип бібліотеки бази даних SQLite.

SQLite не є автономним процесом, як інші бази даних, ви можете зв'язати його статично або динамічно відповідно до вашої вимоги з вашою програмою. SQLite безпосередньо звертається до своїх файлів зберігання.

Слово «вбудований» (embedded) означає, що SQLite не використовує парадигму клієнт-сервер, тобто двигун SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а являє собою бібліотеку, з якою програма компонується, і двигун стає складовою програми. Таким чином, як протокол обміну використовуються виклики функцій (API) бібліотеки SQLite. Такий підхід зменшує накладні витрати, час відгуку та спрощує програму. SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси та дані) в єдиному стандартному файлі на тому комп'ютері, на якому виконується програма. Простота реалізації досягається рахунок того, що перед початком виконання транзакції запису весь файл, що зберігає базу даних, блокується; ACID-функції досягаються у тому числі створення файлу журналу.

Декілька процесів або потоків можуть одночасно без будь-яких проблем читати дані з однієї бази. Запис до бази можна здійснити лише у тому випадку,

якщо жодних інших запитів на даний момент не обслуговується; інакше спроба запису закінчується невдачею, й у програму повертається код помилки. Іншим варіантом розвитку подій є автоматичне повторення спроб запису протягом заданого інтервалу часу.

Старі версії SQLite були спроектовані без будь-яких обмежень, єдиною умовою було те, щоб база даних містилася в пам'яті, в якій всі обчислення проводилися за допомогою 32-розрядних цілих чисел. Це створювало певні проблеми. Через те, що верхні межі не були визначені і належним чином протестовані, часто виявлялися помилки при використанні SQLite в досить екстремальних умовах. Тому в нових версіях SQLite було введено межі, які тепер перевіряються разом із загальним набором тестів. [9]

Особливості SQLite:

- SQLite не вимагає окремого процесу сервера або системи для роботи (без сервера).
- SQLite поставляється з нульовою конфігурацією, що означає відсутність необхідності в налаштуванні або адмініструванні.
- Повна база даних SQLite зберігається в одному крос-платформному диску.
- SQLite дуже маленький і легкий, менше 400KiB повністю налаштований або менше 250KiB з додатковими функціями.
- SQLite є автономним, що означає відсутність зовнішніх залежностей.
- Транзакції SQLite повністю сумісні з ACID, забезпечуючи безпечний доступ до кількох процесів або потоків.
- SQLite підтримує більшість функцій мови запитів, знайдених у стандарті SQL92 (SQL2).
- SQLite написаний на ANSI-C і надає простий та простий у використанні API.

- SQLite доступний у UNIX (Linux, Mac OS-X, Android, iOS) та Windows (Win32, WinCE, WinRT).

Команди SQLite. Стандартні команди SQLite взаємодії з реляційними базами даних аналогічні SQL. Це CREATE, SELECT, INSERT, UPDATE, DELETE та DROP.

Назва	Опис
CREATE	Створення нової таблиці, представлення таблиці чи інший об'єкт у базі даних.
ALTER	Змінює об'єкт бази даних, такий як таблиця.
DROP	Видаляє всю таблицю, представлення таблиці чи іншого об'єкта у базі даних.
INSERT	Створює запис
UPDATE	Змінює записи
DELETE	Видаляє записи
SELECT	Витягує певні записи з однієї або кількох таблиць

Табл. 1. Основні команди SQLite.

Бібліотека OpenCV.

OpenCV (Open Source Computer Vision Library, бібліотека комп'ютерного зору з відкритим кодом) — бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Реалізована C/C++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов. Може вільно використовуватися в академічних та комерційних цілях – поширюється за умов ліцензії BSD.

Рис. 3.3. Логотип бібліотеки OpenCV

Другий великий апдейт OpenCV було випущено у жовтні 2009 року. OpenCV 2 включає серйозні зміни в інтерфейсі C++, спрямовані на спрощення, поліпшення безпеки, введення нових функцій і збільшення продуктивності (особливо для багатоядерних систем). Офіційні релізи випускаються кожні шість місяців, а технологія ведеться незалежної російської командою за підтримки комерційних корпорацій. [10]

Приклад розпізнавання обличчя за допомогою Python та OpenCV.

Обличчя може розглядатися як унікальна особистість людини. Люди по всьому світу мають унікальні особи та риси обличчя. Воно відіграє важливу роль у взаємодії з іншими людьми у суспільстві. З огляду на ці факти, розпізнавання осіб здійснюється у реальному світі.

Систему розпізнавання обличчя можна визначити як технологію, яка може ідентифікувати або перевірити людину з цифрового зображення або відеоджерела шляхом порівняння та аналізу моделей на основі контурів особи людини.

Починаючи з середини 1900-х років, вчені працюють над використанням комп'ютерів для розпізнавання людських осіб. Розпізнавання осіб набуло значної уваги від дослідників через його широке застосування у реальному світі наприклад FaceID у компанії Apple. [11]

Оскільки обличчя – це унікальний спосіб ідентифікації людей, розпізнавання обличчя привертає до себе велику увагу та швидко зростає у всьому світі для забезпечення надійного та надійного захисту. Він набуває

великого значення для корпоративних компаній та державних організацій через високий рівень безпеки та надійності.

В даний час вважається, що розпізнавання обличчя має більше переваг у порівнянні з іншими біометричними системами, такими як відбитки долонь і відбитки пальців, оскільки розпізнавання обличчя не вимагає будь-якої взаємодії з людиною і може здійснюватися без відома людини, яка може бути дуже корисною для виявлення людської діяльності, виявленої в різні програми безпеки, такі як аеропорт, виявлення злочинців, відстеження осіб, криміналістика тощо.

За ці роки було багато методів, що використовуються для реалізації моделей розпізнавання обличчя, але завдяки штучному інтелекту це полегшило наше життя. Використовуючи глибоке навчання (частина штучного інтелекту), надаючи достатні дані, можна легко створити систему розпізнавання обличчя. Ми використовуємо OpenCV для створення простої моделі розпізнавання обличчя.

Дані для початку: у нас є тренувальні дані, що складаються з кількох зображень знаменитостей, таких як М.С. Доні, Вірат Кохлі та актор Прабхас. Ці зображення розташовані в трьох різних папках, і кожна папка складається з зображень тільки однієї людини, і вони названі в числовому порядку так:

Рис. 3.4. Необхідні бібліотеки с обличчями.

Після імпортування всіх бібліотек визначимо деякі функції, які використовуються для виявлення осіб.

За заданим зображенням наша перша функція розпізнає обличчя на зображенні та повертає зображення у градаціях сірого з рамкою навколо обличчя. Тут, у цій функції, ми спочатку перетворимо наше тестове зображення, яке є кольоровим зображенням, у зображення у відтінках сірого, використовуючи OpenCV, а потім завантажуємо класифікатор Haar, який ми спочатку завантажили. Цей файл використовується для виявлення обличчя на нашому зображенні.

```
import cv2
import os
import numpy as np
```

Рис. 3.5. Підключаємо необхідні модулі та бібліотеки.

Наша друга функція приймає каталог як вхідні дані і повертає грані разом з їх мітками. Ця функція завантажує кожне зображення одне за одним у підкаталоги наших навчальних даних, виявляє особи на кожному зображенні, обрізає особи на кожному зображенні та повертає особи з їхніми ідентифікаторами.

```
def faceDetection(test_img):
    gray_img=cv2.cvtColor(test_img,cv2.COLOR_BGR2GRAY)#convert color image to grayscale
    face_haar_cascade=cv2.CascadeClassifier('HaarCascade/haarcascade_frontalface_default.xml')#Load haar classifier
    faces=face_haar_cascade.detectMultiScale(gray_img,scaleFactor=1.32,minNeighbors=5)#detectMultiScale returns rectangles

    return faces,gray_img
```

Рис. 3.6. Функція розпізнавання обличчя.

Наші функції, що залишилися, використовуються для навчання нашого класифікатора Haar, малювання обмежуючих рамок навколо граней і для введення тексту на полях.

```
def labels_for_training_data(directory):
    faces=[]
    faceID=[]

    for path,subdirnames,filenames in os.walk(directory):
        for filename in filenames:
            if filename.startswith("."):
                print("Skipping system file")#Skipping files that startwith .
                continue

            id=os.path.basename(path)#fetching subdirectory names
            img_path=os.path.join(path,filename)#fetching image path
            print("img_path:",img_path)
            print("id:",id)
            test_img=cv2.imread(img_path)#loading each image one by one
            if test_img is None:
                print("Image not loaded properly")
                continue
            faces_rect,gray_img=faceDetection(test_img)#Calling faceDetection function to return faces detected in particular image
            if len(faces_rect)!=1:
               continue #Since we are assuming only single person images are being fed to classifier
            (x,y,w,h)=faces_rect[0]
            roi_gray=gray_img[y:y+w,x:x+h]#cropping region of interest i.e. face area from grayscale image
            faces.append(roi_gray)
            faceID.append(int(id))
    return faces,faceID
```

Рис. 3.7. Функція повертає частину обличчя із його міткою.

Після того, як всі функції визначені, ми продовжимо та навчимо нашу модель і пройдемо тестове зображення, щоб розпізнати обличчя на тестовому зображенні.

```
def train_classifier(faces,faceID):
    face_recognizer=cv2.face.LBPHFaceRecognizer_create()
    face_recognizer.train(faces,np.array(faceID))
    return face_recognizer

#Below function draws bounding boxes around detected face in image
def draw_rect(test_img,face):
    (x,y,w,h)=face
    cv2.rectangle(test_img,(x,y),(x+w,y+h),(255,0,0),thickness=5)

#Below function writes name of person for detected label
def put_text(test_img,text,x,y):
    cv2.putText(test_img,text,(x,y),cv2.FONT_HERSHEY_DUPLEX,2,(255,0,0),4)
```

Рис. 3.8. Функція для тестової моделі.

Далі передаємо наступне зображення MS Dhoni як тестове зображення.

```
#This module takes images  stored in diskand performs face recognition
test_img=cv2.imread('TestImages/dhoni.jpg')#test_img path
faces_detected,gray_img=fr.faceDetection(test_img)
print("faces_detected:",faces_detected)
```

Рис. 3.9. Функція обробки тестового фото.

Після тренування нашої моделі ми пройдемо тестове зображення. На цьому останньому кроці наша модель виявляє обличчя на тестовому зображенні і малює рамку навколо обличчя, що обмежує, і намагається передбачити людину на зображенні. Наш кінцевий результат після запуску цього коду виглядає на Рис. 3.10.

```
for face in faces_detected:
    (x,y,w,h)=face
    roi_gray=gray_img[y:y+h,x:x+h]
    label,confidence=face_recognizer.predict(roi_gray)#predicting the label of given image
    print("confidence:",confidence)
    print("label:",label)
    fr.draw_rect(test_img,face)
    predicted_name=name[label]
    if(confidence>37):#If confidence more than 37 then don't print predicted face text on screen
        continue
    fr.put_text(test_img,predicted_name,x,y)

resized_img=cv2.resize(test_img,(1000,1000))
cv2.imshow("face dtecetion tutorial",resized_img)
cv2.waitKey(0)#Waits indefinitely until a key is pressed
cv2.destroyAllWindows
```

Рис. 3.10. Кінцева функція розпізнавання обличчя.

Приклади застосування цієї системи у світі:

- У світі розпізнавання обличчя широко використовують у системах спостереження.
- Це також використовується у розкритті злочину та судовій експертизі. Федеральне бюро розслідувань США використовує розпізнавання обличчя для ідентифікації підозрюваних за посвідченнями водія. Камери, обладнані ШІ, також були випробувані у Англії для виявлення контрабанди у в'язницях.
- Розпізнавання обличчя також використовується у платежах для забезпечення безпечних та надійних онлайн-платежів.
- Він використовується у мобільних телефонах для розблокування. Це ефективний спосіб захистити особисті дані та забезпечити, щоб

у разі крадіжки телефону зловмисник залишив недоступними конфіденційні дані.

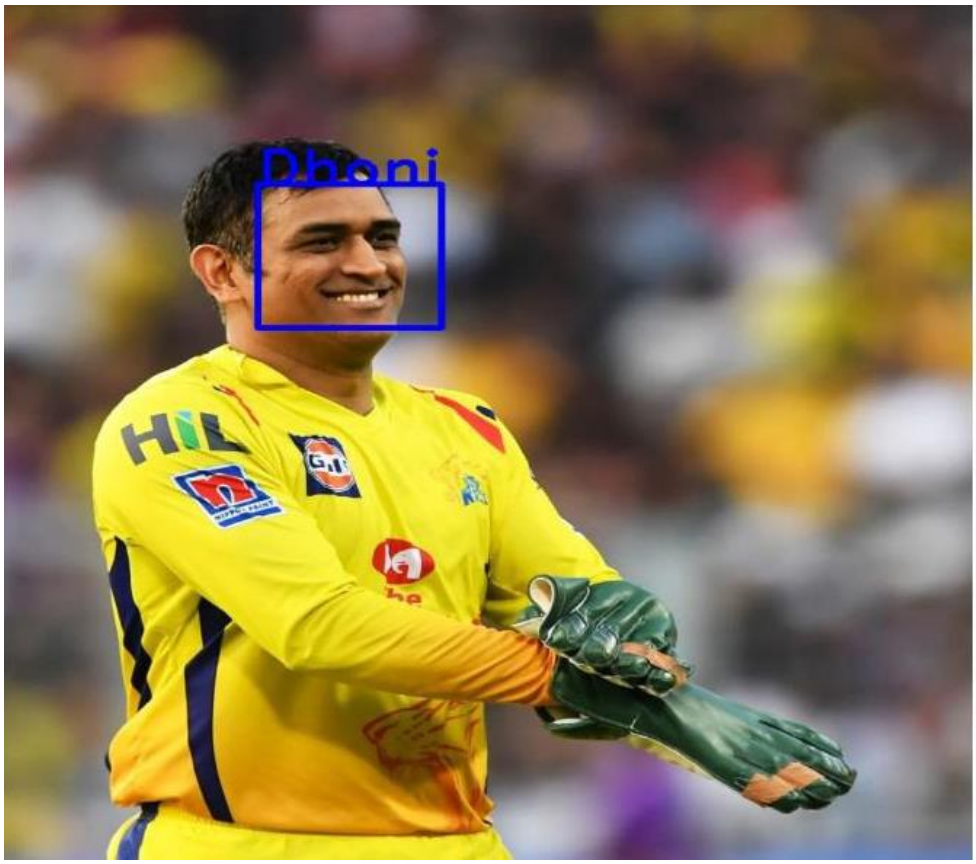

Рис. 3.11. Результат функції розпізнавання обличчя.

Розпізнавання обличчя є цікавою системою 21 століття, OpenCV зробив його неймовірно простим і легким, щоб його кодувати. Щоб мати повністю працюючу програму розпізнавання обличчя, потрібно всього лише кілька рядків коду.

Більш просунуті алгоритми розпізнавання обличчя реалізовані за допомогою комбінації OpenCV і машинного навчання. [12]

Майбутній концепт інтернет ресурсу покращеної системи розпізнавання DeepFake відео.

Таким чином зібравши всю інформацію в єдине, ми змогли скласти концепт майбутнього ресурсу, для розпізнавання підроблених DeepFake відео. На даний момент розроблена система на Python з використанням багатьох бібліотек (Повний список бібліотек у додатку В). У майбутньому система матиме свій інтернет ресурс, приклад ресурсу вказаний нижче на концептуальних рисунках.

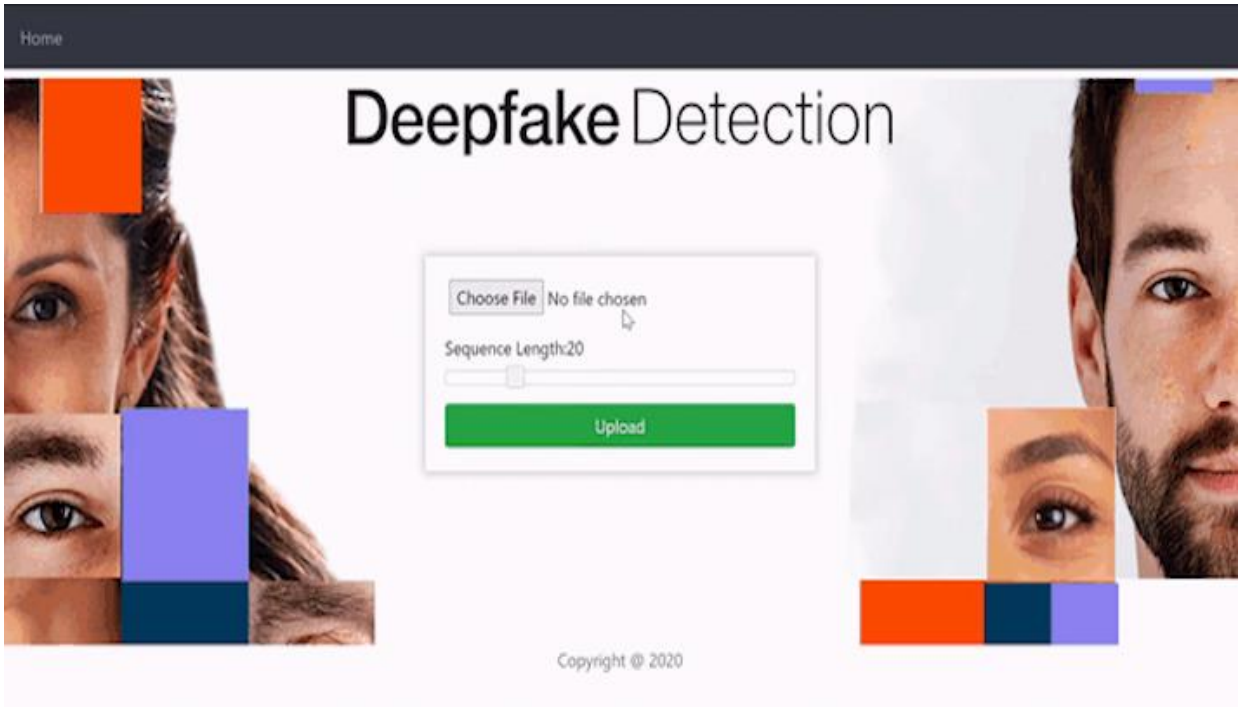

Рис. 4. Головна сторінка ресурсу, із можливістю вибору відео для перевірки.

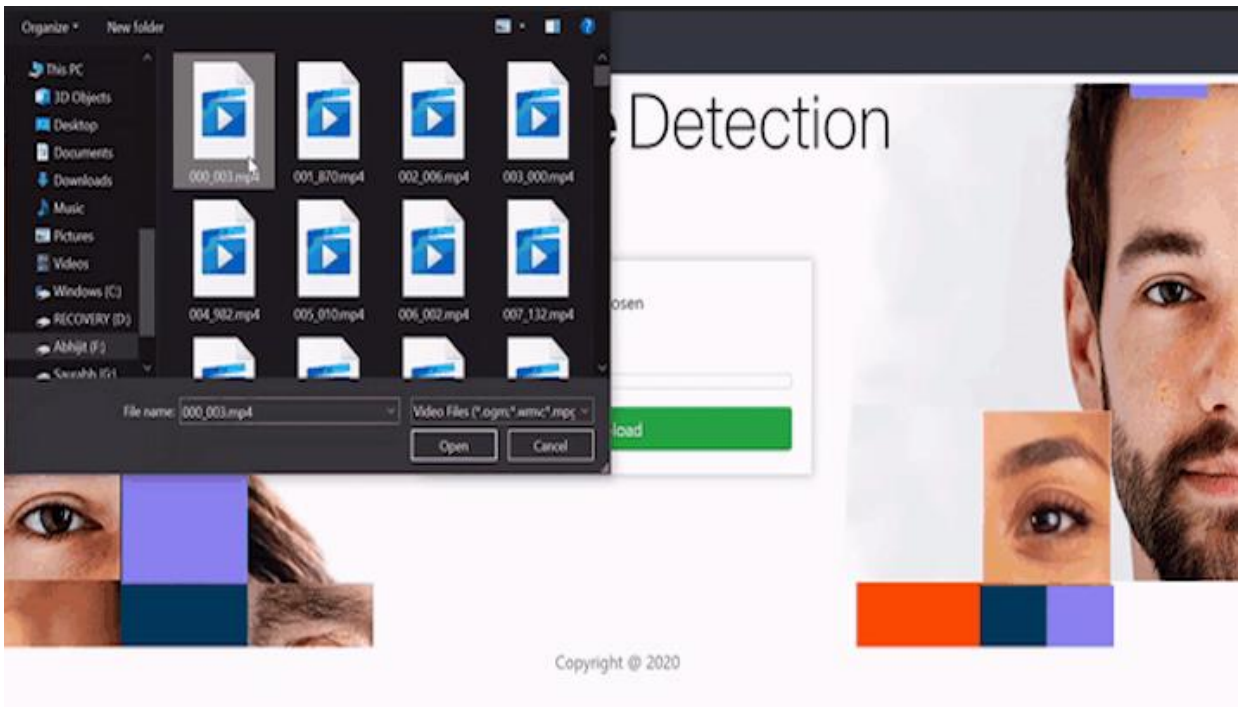

Рис. 4.1. Вікно вибору відео з файлів комп'ютера або хмарного диска.

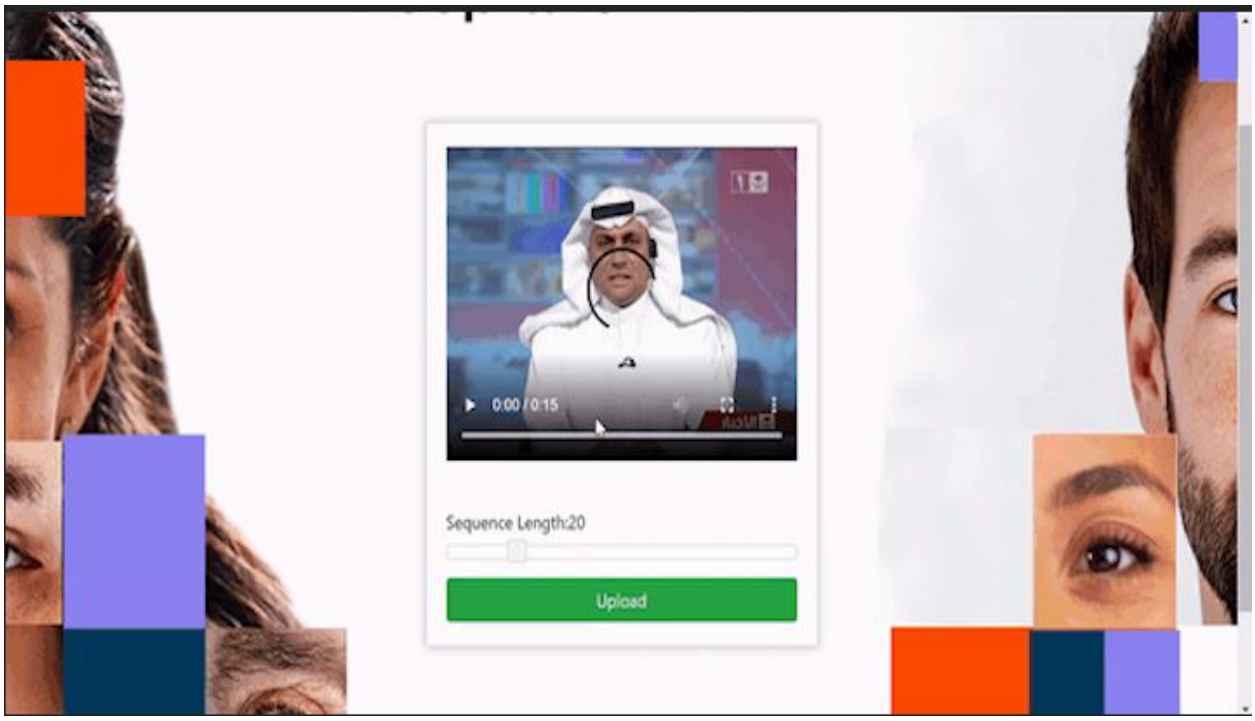

Рис. 4.2. Завантаження відео на наш ресурс та його прев'ю.

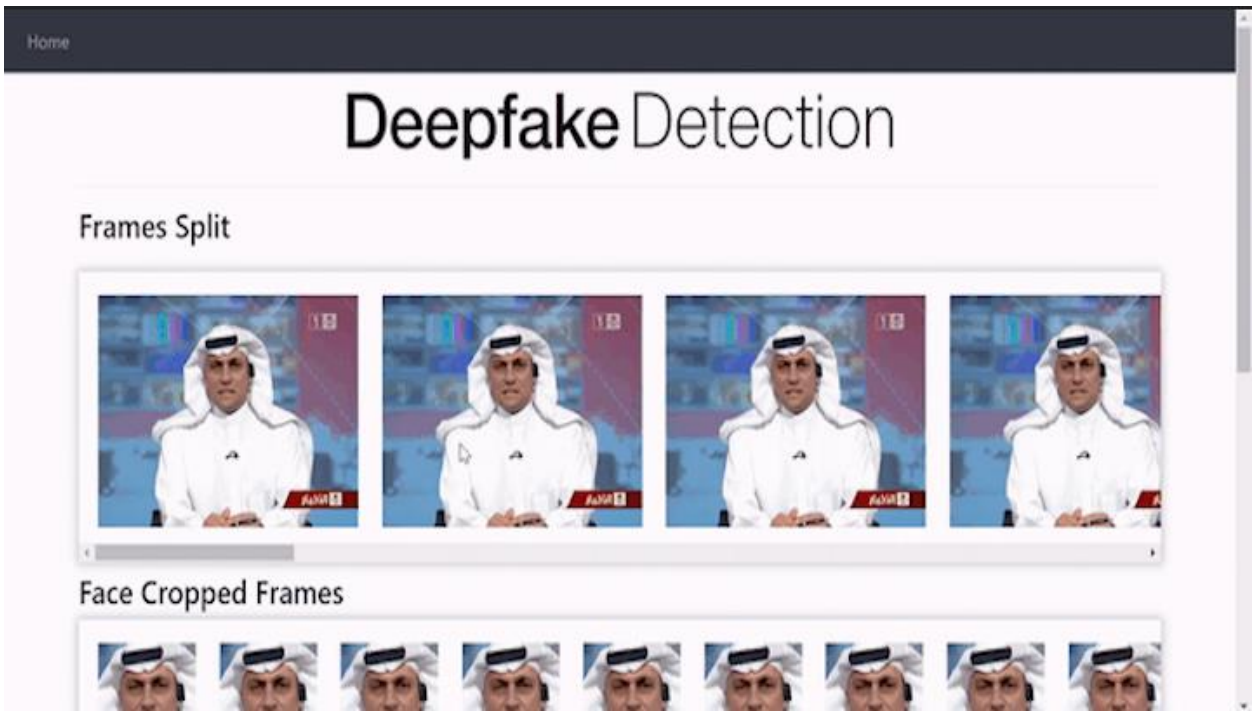

Рис. 4.3. Відео розрізається на фрагменти, котрі може побачити користувач.

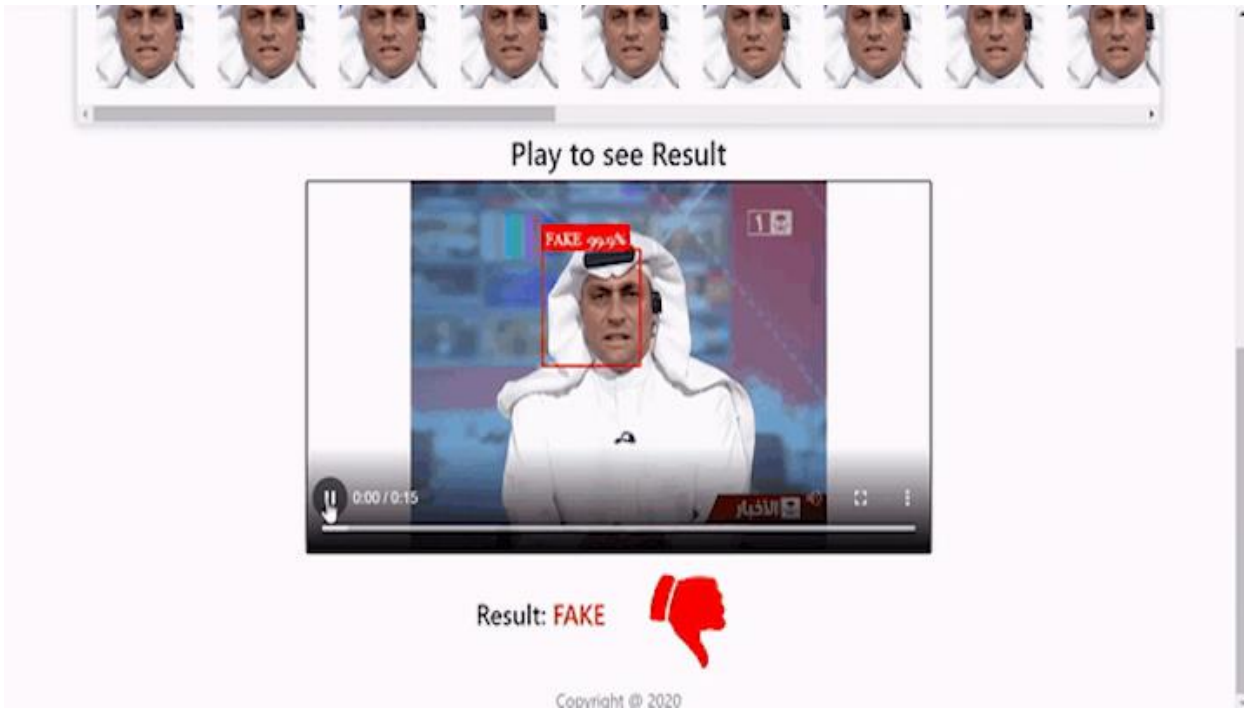

Рис. 4.4. Внизу сторінки користувач бачить результат перевірки.
У цьому випадку відео Fake (підроблене)

Цей проект спрямований на виявлення DeepFake відео з використанням таких методів глибокого навчання, як ResNext та LSTM. Ми досягли виявлення глибоких підробок за допомогою трансферного навчання, в якому попередньо вивчена CNN ResNext використовується для отримання векторів ознак, а потім шар LSTM навчається з використанням цих функцій.

Для простоти розуміння проект структурований у наступному форматі:

- Django Application
- Model Creation

1. Додаток Django
    - Цей каталог складається з програми нашої роботи, створеної django. Де користувач може завантажити відео та передати його моделі для прогнозу. Навчена модель виконує прогноз, і результат відображається на екрані.
2. Створення моделі
    - Цей каталог містить покроковий процес створення та навчання моделі виявлення DeepFake з використанням нашого підходу.

ВИСНОВКИ

У сучасному світі інформаційний ресурс став одним із найпотужніших важелів економічного розвитку. З поширенням технології Deepfake виникла небезпека дискредитації будь-якого користувача, чия фотографія або голос є в загальному доступі. Технології створення DeepFake постійно вдосконалюються, і вже за кілька років очікується появи таких підробок, що настільки природно виглядають або звучать, що виявити їх можна буде тільки після ретельного аналізу з використанням штучного інтелекту.

При цьому найперспективніші напрями використання DeepFake — політичні війни та шахрайство. Крім того, з урахуванням постійного вдосконалення технологій DeepFake можуть також зашкодити і судовій практиці — щодо довіри до аудіо- та відеоматеріалів доказової бази.

В даний час ринок інформаційної безпеки не пропонує спеціалізованих рішень для захисту від DeepFake. Однак у нашому проекті ми показали можливу систему для вирішення подібних проблем, ця система і подібні до них ще не використовуються в офіційних засобах виявлення DeepFake, але це вже як одне з рішень.

У магістерській роботі були проведені дослідження щодо актуальності та доцільності системи інформаційної безпеки, а саме створення системи вияву підроблених відео - DeepFake.

Було проведено аналіз існуючих систем, а також інших проблем інформаційної безпеки.

Як подальший розвиток цього проекту, буде розроблений інтернет ресурс, в якому будь який користувач зможе перевірити відео на його оригінальність та виявити підроблене відео.

ПЕРЕЛІК ПОСИЛАНЬ

1. Інформаційна безпека [Електронний ресурс] - https://pirit.biz/reshenija/informacionnaja-bezopasnost
2. Визначення інформаційної безпеки [Електронний ресурс] - https://searchinform.ru/informatsionnaya-bezopasnost/
3. Чим небезпечний додаток «Дія» [Електронний ресурс] - https://focus.ua/digital/496164-gosudarstvo-v-gastronime-chem-opasno-poyavlenie-lzhe-dii-dlya-pokupki-alkogolya-podrostkami
4. Чим небезпечна система DeepFake [Електронний ресурс] - https://www.anti-malware.ru/analytics/Threats_Analysis/Deepfakes-as-a-information-security-threat#part2
5. Deepfake Detection Challenge [Електронний ресурс] - https://center2m.ru/google-deepfake
6. Deepfake Video Detection using Neural Networks [Електронний ресурс] - *https://github.com/abhijitjadhav1998/Deepfake_detection_using_deep_learning/blob/master/Documentation/IJSRDV8I10860.pdf*
7. Інтерпретатор Python - CPython [Електронний ресурс] - https://www.python.org/about/
8. Фреймворк Django [Електронний ресурс] - https://habr.com/ru/hub/django/
9. SQLite – бібліотека для швидкої бази даних [Електронний ресурс] - https://habr.com/ru/post/149356/
10. Бібліотека комп'ютерного зору OpenCV [Електронний ресурс] - https://ru.wikipedia.org/wiki/OpenCV
11. Система розпізнавання обличчя Apple [Електронний ресурс] - https://support.apple.com/en-us/HT208109
12. Майбутнє систем розпізнавання осіб [Електронний ресурс] - https://www.superdatascience.com/blogs/opencv-face-recognition

ДОДАТКИ

ДОДАТОК А. ЗМІСТ ОСНОВНОГО ФАЙЛУ ПРОЕКТУ

```
from django.shortcuts import render, redirect
import torch
import torchvision
from torchvision import transforms, models
from torch.utils.data import DataLoader
from torch.utils.data.dataset import Dataset
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import face_recognition
from torch.autograd import Variable
import time
import sys
from torch import nn
import json
import glob
import copy
from torchvision import models
import shutil
from PIL import Image as pImage
import time
from django.conf import settings
from .forms import VideoUploadForm

index_template_name = 'index.html'
predict_template_name = 'predict.html'
```

```
im_size = 112

mean=[0.485, 0.456, 0.406]

std=[0.229, 0.224, 0.225]

sm = nn.Softmax()

inv_normalize =  transforms.Normalize(mean=-
1*np.divide(mean,std),std=np.divide([1,1,1],std))


train_transforms = transforms.Compose([

                        transforms.ToPILImage(),

                        transforms.Resize((im_size,im_size)),

                        transforms.ToTensor(),

                        transforms.Normalize(mean,std)])


class Model(nn.Module):


   def __init__(self, num_classes,latent_dim= 2048, lstm_layers=1 , hidden_dim = 2048,
bidirectional = False):

      super(Model, self).__init__()

      model = models.resnext50_32x4d(pretrained = True)

      self.model = nn.Sequential(*list(model.children())[:-2])

      self.lstm = nn.LSTM(latent_dim,hidden_dim, lstm_layers,  bidirectional)

      self.relu = nn.LeakyReLU()

      self.dp = nn.Dropout(0.4)

      self.linear1 = nn.Linear(2048,num_classes)

      self.avgpool = nn.AdaptiveAvgPool2d(1)


   def forward(self, x):

      batch_size,seq_length, c, h, w = x.shape

      x = x.view(batch_size * seq_length, c, h, w)

      fmap = self.model(x)
```

```
        x = self.avgpool(fmap)
        x = x.view(batch_size,seq_length,2048)
        x_lstm,_ = self.lstm(x,None)
        return fmap,self.dp(self.linear1(x_lstm[:,-1,:]))


class validation_dataset(Dataset):
    def __init__(self,video_names,sequence_length=60,transform = None):
        self.video_names = video_names
        self.transform = transform
        self.count = sequence_length

    def __len__(self):
        return len(self.video_names)

    def __getitem__(self,idx):
        video_path = self.video_names[idx]
        frames = []
        a = int(100/self.count)
        first_frame = np.random.randint(0,a)
        for i,frame in enumerate(self.frame_extract(video_path)):
            #if(i % a == first_frame):
            faces = face_recognition.face_locations(frame)
            try:
              top,right,bottom,left = faces[0]
              frame = frame[top:bottom,left:right,:]
            except:
              pass
            frames.append(self.transform(frame))
```

```
            if(len(frames) == self.count):
                break
        """
        for i,frame in enumerate(self.frame_extract(video_path)):
            if(i % a == first_frame):
                frames.append(self.transform(frame))
        """
        # if(len(frames)<self.count):
        #   for i in range(self.count-len(frames)):
        #        frames.append(self.transform(frame))
        #print("no of frames", self.count)
        frames = torch.stack(frames)
        frames = frames[:self.count]
        return frames.unsqueeze(0)

    def frame_extract(self,path):
      vidObj = cv2.VideoCapture(path)
      success = 1
      while success:
          success, image = vidObj.read()
          if success:
              yield image

def im_convert(tensor, video_file_name):
    """ Display a tensor as an image. """
    image = tensor.to("cpu").clone().detach()
    image = image.squeeze()
    image = inv_normalize(image)
    image = image.numpy()
```

```
    image = image.transpose(1,2,0)
    image = image.clip(0, 1)
    # This image is not used
    # cv2.imwrite(os.path.join(settings.PROJECT_DIR, 'uploaded_images',
video_file_name+'_convert_2.png'),image*255)
    return image

def im_plot(tensor):
    image = tensor.cpu().numpy().transpose(1,2,0)
    b,g,r = cv2.split(image)
    image = cv2.merge((r,g,b))
    image = image*[0.22803, 0.22145, 0.216989] +  [0.43216, 0.394666, 0.37645]
    image = image*255.0
    plt.imshow(image.astype(int))
    plt.show()


def predict(model,img,path = './', video_file_name=""):
  fmap,logits = model(img.to('cuda'))
  img = im_convert(img[:,-1,:,:,:], video_file_name)
  params = list(model.parameters())
  weight_softmax = model.linear1.weight.detach().cpu().numpy()
  logits = sm(logits)
  _,prediction = torch.max(logits,1)
  confidence = logits[:,int(prediction.item())].item()*100
  print('confidence of prediction:',logits[:,int(prediction.item())].item()*100)
  return [int(prediction.item()),confidence]

def plot_heat_map(i, model, img, path = './', video_file_name=''):
  fmap,logits = model(img.to('cuda'))
```

```
  params = list(model.parameters())
  weight_softmax = model.linear1.weight.detach().cpu().numpy()
  logits = sm(logits)
  _,prediction = torch.max(logits,1)
  idx = np.argmax(logits.detach().cpu().numpy())
  bz, nc, h, w = fmap.shape
  #out = np.dot(fmap[-1].detach().cpu().numpy().reshape((nc, h*w)).T,weight_softmax[idx,:].T)
  out = np.dot(fmap[i].detach().cpu().numpy().reshape((nc, h*w)).T,weight_softmax[idx,:].T)
  predict = out.reshape(h,w)
  predict = predict - np.min(predict)
  predict_img = predict / np.max(predict)
  predict_img = np.uint8(255*predict_img)
  out = cv2.resize(predict_img, (im_size,im_size))
  heatmap = cv2.applyColorMap(out, cv2.COLORMAP_JET)
  img = im_convert(img[:,-1,:,:,:], video_file_name)
  result = heatmap * 0.5 + img*0.8*255
  # Saving heatmap - Start
  heatmap_name = video_file_name+"_heatmap_"+str(i)+".png"
  image_name = os.path.join(settings.PROJECT_DIR, 'uploaded_images', heatmap_name)
  cv2.imwrite(image_name,result)
  # Saving heatmap - End
  result1 = heatmap * 0.5/255 + img*0.8
  r,g,b = cv2.split(result1)
  result1 = cv2.merge((r,g,b))
  return image_name

# Model Selection
def get_accurate_model(sequence_length):
    model_name = []
```

```
    sequence_model = []
    final_model = ""
    list_models = glob.glob(os.path.join(settings.PROJECT_DIR, "models", "*.pt"))
    for i in list_models:
        model_name.append(i.split("\\")[-1])
    for i in model_name:
        try:
            seq = i.split("_")[3]
            if (int(seq) == sequence_length):
                sequence_model.append(i)
        except:
            pass

    if len(sequence_model) > 1:
        accuracy = []
        for i in sequence_model:
            acc = i.split("_")[1]
            accuracy.append(acc)
        max_index = accuracy.index(max(accuracy))
        final_model = sequence_model[max_index]
    else:
        final_model = sequence_model[0]
    return final_model

ALLOWED_VIDEO_EXTENSIONS = set(['mp4','gif','webm','avi','3gp','wmv','flv','mkv'])

def allowed_video_file(filename):
    #print("filename" ,filename.rsplit('.',1)[1].lower())
    if (filename.rsplit('.',1)[1].lower() in ALLOWED_VIDEO_EXTENSIONS):
```

```
        return True
    else:
        return False
def index(request):
    if request.method == 'GET':
        video_upload_form = VideoUploadForm()
        if 'file_name' in request.session:
            del request.session['file_name']
        if 'preprocessed_images' in request.session:
            del request.session['preprocessed_images']
        if 'faces_cropped_images' in request.session:
            del request.session['faces_cropped_images']
        return render(request, index_template_name, {"form": video_upload_form})
    else:
        video_upload_form = VideoUploadForm(request.POST, request.FILES)
        if video_upload_form.is_valid():
            video_file = video_upload_form.cleaned_data['upload_video_file']
            video_file_ext = video_file.name.split('.')[-1]
            sequence_length = video_upload_form.cleaned_data['sequence_length']
            video_content_type = video_file.content_type.split('/')[0]
            if video_content_type in settings.CONTENT_TYPES:
                if video_file.size > int(settings.MAX_UPLOAD_SIZE):
                    video_upload_form.add_error("upload_video_file", "Maximum file size 100 MB")
                    return render(request, index_template_name, {"form": video_upload_form})

            if sequence_length <= 0:
                video_upload_form.add_error("sequence_length", "Sequence Length must be greater
than 0")
                return render(request, index_template_name, {"form": video_upload_form})
```

```
            if allowed_video_file(video_file.name) == False:
                video_upload_form.add_error("upload_video_file","Only video files are allowed ")
                return render(request, index_template_name, {"form": video_upload_form})


            saved_video_file = 'uploaded_file_'+str(int(time.time()))+"."+video_file_ext
            with open(os.path.join(settings.PROJECT_DIR, 'uploaded_videos', saved_video_file),
'wb') as vFile:
                shutil.copyfileobj(video_file, vFile)


            request.session['file_name'] = os.path.join(settings.PROJECT_DIR, 'uploaded_videos',
saved_video_file)
            request.session['sequence_length'] = sequence_length
            return redirect('ml_app:predict')
        else:
            return render(request, index_template_name, {"form": video_upload_form})


def predict_page(request):
    if request.method == "GET":
        if 'file_name' not in request.session:
            return redirect("ml_app:home")
        if 'file_name' in request.session:
            video_file = request.session['file_name']
        if 'sequence_length' in request.session:
            sequence_length = request.session['sequence_length']
        path_to_videos = [video_file]
        video_file_name = video_file.split('\\')[-1]
        video_file_name_only = video_file_name.split('.')[0]
        video_dataset = validation_dataset(path_to_videos,
sequence_length=sequence_length,transform= train_transforms)
        model = Model(2).cuda()
```

```
    model_name = os.path.join(settings.PROJECT_DIR,'models',
get_accurate_model(sequence_length))
    models_location = os.path.join(settings.PROJECT_DIR,'models')
    path_to_model = os.path.join(settings.PROJECT_DIR, model_name)
    model.load_state_dict(torch.load(path_to_model))
    model.eval()
    start_time = time.time()
    # Start: Displaying preprocessing images
    print("<=== | Started Videos Splitting | ===>")
    preprocessed_images = []
    faces_cropped_images = []
    cap = cv2.VideoCapture(video_file)

    frames = []
    while(cap.isOpened()):
        ret, frame = cap.read()
        if ret==True:
            frames.append(frame)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        else:
            break
    cap.release()

    for i in range(1, sequence_length+1):
        frame = frames[i]
        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        img = pImage.fromarray(image, 'RGB')
        image_name = video_file_name_only+"_preprocessed_"+str(i)+'.png'
        image_path = os.path.join(settings.PROJECT_DIR, 'uploaded_images', image_name)
```

```
            img.save(image_path)
            preprocessed_images.append(image_name)
        print("<=== | Videos Splitting Done | ===>")
        print("--- %s seconds ---" % (time.time() - start_time))
        # End: Displaying preprocessing images


        # Start: Displaying Faces Cropped Images
        print("<=== | Started Face Cropping Each Frame | ===>")
        padding = 40
        faces_found = 0
        for i in range(1, sequence_length+1):
            frame = frames[i]
            #fig, ax = plt.subplots(1,1, figsize=(5, 5))
            face_locations = face_recognition.face_locations(frame)
            if len(face_locations) == 0:
                continue
            top, right, bottom, left = face_locations[0]
            frame_face = frame[top-padding:bottom+padding, left-padding:right+padding]
            image = cv2.cvtColor(frame_face, cv2.COLOR_BGR2RGB)

            img = pImage.fromarray(image, 'RGB')
            image_name = video_file_name_only+"_cropped_faces_"+str(i)+'.png'
            image_path = os.path.join(settings.PROJECT_DIR, 'uploaded_images',
video_file_name_only+"_cropped_faces_"+str(i)+'.png')
            img.save(image_path)
            faces_found = faces_found + 1
            faces_cropped_images.append(image_name)
        print("<=== | Face Cropping Each Frame Done | ===>")
        print("--- %s seconds ---" % (time.time() - start_time))
```

```
    # No face is detected
    if faces_found == 0:
        return render(request, predict_template_name, {"no_faces": True})


    # End: Displaying Faces Cropped Images
    try:
        heatmap_images = []
        for i in range(0, len(path_to_videos)):
            output = ""
            print("<=== | Started Predicition | ===>")
            prediction = predict(model, video_dataset[i], './', video_file_name_only)
            confidence = round(prediction[1], 1)
            print("<=== |  Predicition Done | ===>")
            # print("<=== | Heat map creation started | ===>")
            # for j in range(0, sequence_length):
            #     heatmap_images.append(plot_heat_map(j, model, video_dataset[i], './',
video_file_name_only))
            if prediction[0] == 1:
                output = "REAL"
            else:
                output = "FAKE"
            print("Prediction : " , prediction[0],"==",output ,"Confidence : " , confidence)
            print("--- %s seconds ---" % (time.time() - start_time))


        return render(request, predict_template_name, {'preprocessed_images':
preprocessed_images, 'heatmap_images': heatmap_images, "faces_cropped_images":
faces_cropped_images, "original_video": video_file_name, "models_location":
models_location, "output": output, "confidence": confidence})
    except:
        return render(request, 'cuda_full.html')
```

```
def about(request):
    return render(request, about_template_name)


def handler404(request,exception):
    return render(request, '404.html', status=404)
def cuda_full(request):
    return render(request, 'cuda_full.html')
```

ДОДАТОК Б. ЗМІСТ ДОПОМІЖНОГО ФАЙЛУ ПРОЕКТУ DJANGO SETTINGS

```
Django settings for project_settings project.

import os


# Build paths inside the project like this: os.path.join(BASE_DIR, ...)

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Build paths inside the project like this: os.path.join(PROJECT_DIR, ...)

PROJECT_DIR = os.path.abspath(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))



# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/


# SECURITY WARNING: keep the secret key used in production secret!

SECRET_KEY = '@)0qp0!&-vht7k0wyuihr+nk-b8zrvb5j^1d@vl84cd1%)f=dz'


# SECURITY WARNING: don't run with debug turned on in production!
```

```
DEBUG = True


# Change and set this to correct IP/Domain
ALLOWED_HOSTS = []



# Application definition

INSTALLED_APPS = [
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'ml_app.apps.MlAppConfig'
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
ROOT_URLCONF = 'project_settings.urls'


TEMPLATES = [
  {
    'BACKEND': 'django.template.backends.django.DjangoTemplates',
    'DIRS': [os.path.join(PROJECT_DIR, 'templates')],
    'APP_DIRS': True,
    'OPTIONS': {
      'context_processors': [
        'django.template.context_processors.debug',
        'django.template.context_processors.request',
        'django.contrib.messages.context_processors.messages',
        'django.template.context_processors.media'
      ],
    },
  },
]


WSGI_APPLICATION = 'project_settings.wsgi.application'



# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases


DATABASES = {
```

```
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": os.path.join(PROJECT_DIR, 'db.sqlite3'),
    }
}



# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = False

USE_L10N = False

USE_TZ = False



# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.0/howto/static-files/

STATIC_URL = '/static/'
```

```
STATICFILES_DIRS = [
    os.path.join(PROJECT_DIR, 'uploaded_images'),
    os.path.join(PROJECT_DIR, 'static'),
    os.path.join(PROJECT_DIR, 'models'),
]

CONTENT_TYPES = ['video']
MAX_UPLOAD_SIZE = "104857600"

MEDIA_URL = "/media/"

MEDIA_ROOT = os.path.join(PROJECT_DIR, 'uploaded_videos')
```

ДОДАТОК В. ЗМІСТ ОСНОВНИХ БІБЛІОТЕК ДЛЯ ПРОЕКТУ

REQUIREMENTS.TXT

```
altair==4.1.0
asgiref==3.2.7
astor==0.8.1
attrs==19.3.0
backcall==0.1.0
base58==2.0.0
bleach==3.1.4
blinker==1.4
cachetools==4.0.0
```

certifi==2019.11.28

chardet==3.0.4

click==7.1.1

cmake==3.16.3

colorama==0.4.3

cycler==0.10.0

decorator==4.4.2

defusedxml==0.6.0

Django==3.0.5

dlib==19.19.0

docutils==0.15.2

entrypoints==0.3

enum-compat==0.0.3

face-recognition==1.3.0

face-recognition-models==0.3.0

future==0.18.2

google==2.0.3

google-api-core==1.16.0

google-api-python-client==1.8.0

google-auth==1.12.0

google-auth-httplib2==0.0.3

googleapis-common-protos==1.51.0

httplib2==0.17.0

idna==2.9

ipykernel==5.2.0

ipython==7.13.0

ipython-genutils==0.2.0

ipywidgets==7.5.1

jedi==0.16.0

Jinja2==2.11.1

jmespath==0.9.5

json5==0.9.4

jsonschema==3.2.0

jupyter-client==6.1.2

jupyter-core==4.6.3

jupyterlab==2.0.1

jupyterlab-server==1.0.7

kiwisolver==1.1.0

MarkupSafe==1.1.1

matplotlib==3.2.1

mistune==0.8.4

nbconvert==5.6.1

nbformat==5.0.4

notebook==6.0.3

numpy==1.18.2

opencv-python==4.2.0.32

packaging==20.3

pandas==1.0.3

pandocfilters==1.4.2

parso==0.6.2

pathtools==0.1.2

pickleshare==0.7.5

Pillow==7.0.0

prometheus-client==0.7.1

prompt-toolkit==3.0.5

protobuf==3.11.3

pyasn1==0.4.8

pyasn1-modules==0.2.8

pycodestyle==2.5.0

pydeck==0.3.0b3

Pygments==2.6.1

pyparsing==2.4.6

pyrsistent==0.16.0

python-dateutil==2.8.1

pytz==2019.3

pywin32==227

pywinpty==0.5.7

PyYAML==5.3.1

pyzmq==19.0.0

requests==2.23.0

rsa==4.0

s3transfer==0.3.3

Send2Trash==1.5.0

six==1.14.0

soupsieve==2.0

sqlparse==0.3.1

terminado==0.8.3

testpath==0.4.4

toml==0.10.0

toolz==0.10.0

torch==1.4.0

torchvision==0.5.0

tornado==5.1.1

traitlets==4.3.3

tzlocal==2.0.0

uritemplate==3.0.1

urllib3==1.25.8

validators==0.14.2

watchdog==0.10.2

wcwidth==0.1.9

webencodings==0.5.1

widgetsnbextension==3.5.1