Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Тимофеєв Євген Володимирович
студент групи РУ-181

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Розробка програмного забезпечення для виявлення фотопідробок, виконаних засобами нейронних мереж

Спеціальність:
125 Кібербезпека

Спеціалізація, освітня програма:
Управління кібербезпекою

Керівник:
Зоріло Вікторія Вікторівна,
к.т.н., ст.викл.

Одеса – 2022

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Рівень вищої освіти перший (бакалаврський)
Спеціальність 125 – Кібербезпека
Освітня програма – Управління кібербезпекою

ЗАТВЕРДЖУЮ
Завідувач кафедри КБПЗ

_____
д.т.н.,проф. А.А.Кобозєва
____ _____2022р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Тимофеєву Євгену Володимировичу

1.Тема роботи: *Розробка програмного забезпечення для виявлення фотопідробок, виконаних засобами нейронних мереж*,
керівник роботи *Зоріло Вікторія Вікторівна, к.т.н.,ст.викл.,*
затверджені наказом ректора від „17” 05. 2022 р. №168-в .
2.Зміст роботи*: сучасний стан проблеми виявлення Deepfake, аналіз алгоритму детектування Deepfake, розробка програмного продукту, охорона праці.*
3. Перелік ілюстративного матеріалу: *зображення для демонстрації результатів обчислювального експерименту, презентація.*

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
Охорона праці	доц. Ярова І.А.	Завдання Видав	Завдання прийняв

6. Дата видачі завдання "_____"_____2022 р.

КАЛЕНДАРНИЙ ПЛАН

з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз джерел з теми випускної кваліфікаційної роботи*	*15.11.2021*	*виконано*
2	*Обґрунтування вибору рішення. Збір даних*	*15-12-2021*	*виконано*
3	*Аналіз основних аспектів реалізації алгоритмів*	*11-01-2022*	*виконано*
4	*Аналіз систем програмування для реалізації програмного продукту*	*20-02-2022*	*виконано*
5	*Розроблення проекту інтерфейсу програмного продукту*	*30-03-2022*	*виконано*
6	*Підготовка тексту роботи*	*11-05-2022*	*виконано*
7	*Підготовка презентації та доповіді*	*15-05-2022*	*виконано*
8	*Попередній захист*	*20-05-2022*	*виконано*
9	*Нормоконтроль, рецензування*	*20-05-2022*	*виконано*

Здобувач вищої освіти _____ *Тимофеєв Є.В.*

Керівник роботи _____ *Зоріло В.В.*

ЗАВДАННЯ

на розробку розділу «Охорона праці»

Тимофеєву Євгену Володимировичу, група РУ-181

Навчально-науковий інститут інформаційної безпеки, радіоелектроніки та телекомунікацій

Кафедра кібербезпеки та програмного забезпечення

Тема роботи *Розробка програмного забезпечення для виявлення фотопідробок, виконаних засобами нейронних мереж*

Зміст розділу:

1 Аналіз умов праці і вибір основних заходів виробничої безпеки.

2 Аналіз пожежної безпеки. Вибір заходів та засобів пожежної безпеки.

Керівник роботи

_____ (*Зоріло В.В.*)

«____» _____ 2022 р.

Консультант з охорони праці

_____(Ярова І.А)

«____» _____ 2022 р.

АНОТАЦІЯ

Кваліфікаційна робота на тему «Розробка програмного забезпечення для виявлення фотопідробок, виконаних засобами нейронних мереж» на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 – Кібербезпека, спеціалізація, освітня програма: Кібербезпека виконана в обсязі 61 сторінки і містить 17 рисунків, 2 таблиці, 1 додаток та 27 джерел за переліком посилань.

Метою роботи є виявлення порушень цілісності медіа файлів, виконаних за технологією Deepfake, шляхом розробки програмного додатку з використанням нейронної мережі.

В результаті аналізу існуючих систем та методів розв’язання даної задачі створена згорткова нейронна мережа для виявлення зображень, які містять Deepfake.

У даній роботі проводиться аналіз методів навчання нейронних мереж, параметрів і гіперпараметрів, здатних краще і швидше вирішити проблеми навчання нейронних мереж. Важливою частиною реалізації такої системи є набір даних, тобто зображення, які далі надходять в алгоритм детектування Deepfake. Практично підтверджено, що на точність результату також впливають самі дані та кількість даних.

Результати роботи можуть бути використані судовими лабораторіями та експертами-криміналістами при перевірці цифрових зображень та відео на наявність їх фальсифікації.

Можливими напрямками подальших досліджень є продовження роботи з аналізу впливу різних видів несанкціонованого втручання у цифрове зображення з метою змінення його змісту та значення.

НЕЙРОННІ МЕРЕЖІ, ЗГОРТКОВІ МЕРЕЖІ, AUTOENCODER, GAN, DEEPFAKE, ЦИФРОВЕ ЗОБРАЖЕННЯ, МАШИННЕ НАВЧАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ, АЛГОРИТМ.

ANNOTATION

Qualification work on "Software development for detection of photo forgeries made by neural networks" for the first (bachelor's) level of higher education in in 125 – Cybersecurity, specialization educational program Cybersecurity is written volume contains 61 pages and 17 figures, 2 table, 1 appendix and 27 sources for references.

The aim of the work is to identify violations of the integrity of media files made by Deepfake technology, by developing a software application using a neural network.

The result of the analysis of existing systems and methods of solving this problem is a convolutional neural network that was created to detect images that contain Deepfake.

This paper analyzes the methods of learning neural networks, parameters and hyper parameters that can better and faster solve the problem of learning neural networks. The important part of the implementation of such a system is dataset, that are images that fed into the Deepfake detection algorithm. It is practically confirmed that the accuracy of the result is also affected by the data itself and the amount of data.

The results can be used by forensic laboratories and forensic experts to check digital images and videos for falsification.

Possible areas of further research are the continuation of work on the analysis of the impact of various types of unauthorized interference in digital images in order to change its content and meaning.

NEURAL NETWORKS, ROLLER NETWORKS, AUTOENCODER, GAN, DEEPFAKE, DIGITAL IMAGE, MACHINE LEARNING, ARTIFICIAL INTELLIGENCE, ALGORITHM.

ЗМІСТ

ВСТУП 8

1 СУЧАСНИЙ СТАН ПРОБЛЕМИ ВИЯВЛЕННЯ DEEPFAKE 10

1.1 Нейронні мережі 10

1.2 Опис технології Deepfake 14

1.3 Методи виявлення Deepfake 18

2.1 Вибір моделі нейронної мережі 21

2.2 Навчання нейронної мережі 25

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ 29

3.1 Обгрунтування вибору програмного середовища 29

3.2 Реалізація алгоритму детектування Deepfake 30

3.3 Опис програмного продукту 34

3.4 Системні вимоги для успішної роботи розробленої програми 36

4 ОХОРОНА ПРАЦІ 37

ВИСНОВКИ 46

ПЕРЕЛІК ПОСИЛАНЬ 47

Додаток А. Лістинг програмного продукту 50

ВСТУП

Глибоке навчання є ефективною та корисною технікою, яка широко застосовується в різних областях, включаючи комп'ютерний зір, машинний зір та обробку природної мови. Deepfake використовує технологію глибокого навчання, щоб маніпулювати зображеннями та відео людини, які люди не можуть відрізнити від реальних. Deepfake зазвичай створюють двома способами. Перший спосіб – за допомогою генеративної змагальної мережі, або GAN, яка використовує дві окремі нейронні мережі, які працюють разом, навчаючись вивчати характеристики реальних зображень, щоб вони могли створювати переконливі фальшиві. Другий спосіб – за допомогою алгоритму штучного інтелекту (ШІ), який має назву «інкодер» і працює шляхом запуску знімків облич двох людей через інкодер для знаходження схожості між цими знімками та запуску декодера, який отримує зображення обличчя та міняє їх місцями.

Останніми роками було проведено багато досліджень, щоб зрозуміти, як працює Deepfake, і було запроваджено багато підходів, заснованих на глибокому навчанні, для виявлення відео або зображень, які містили Deepfake.

Досягнення в області Deepfake однаково вражають і турбують. У чужих руках ця технологія може бути використана для поширення дезінформації та підриву суспільної довіри майже як науково-фантастичний тип крадіжки особистих даних, коли ви можете змусити будь-кого сказати що завгодно, і це навпаки. Deepfake може викликати розкол на політичній арені, маючи великий потенціал для підриву політичних систем, наприклад, як фейкове відео, яке з'явилося у березні 2022 року, на якому президент України Володимир Зеленський каже українським солдатам здатися під час російської вторгнення.

У зв'язку з цим можна констатувати, що задача детектування Deepfake є дуже важливою та потребує застосування нових для цієї галузі досліджень,

розробки нових алгоритмів детектування Deepfake та вдосконалення вже існуючих методів.

Мета даної роботи – виявлення порушень цілісності медіа файлів, виконаних за технологією Deepfake, шляхом розробки програмного додатку з використанням нейронної мережі.

Для досягнення даної мети необхідно вирішити наступні задачі:

1. Аналіз принципів роботи Deepfake та подібних алгоритмів.
2. Розробка системи детектування Deepfake за допомогою нейронної мережі.
3. Розробка програмного забезпечення.

Об’єкт дослідження – процес створення моделі для розпізнавання Deepfake.

Предмет дослідження – математичні, програмні засоби і методи, завдяки …???

Результати, отримані в роботі, опубліковано в фаховому виданні «Інформатика та математичні методи в моделюванні» [1].

1 СУЧАСНИЙ СТАН ПРОБЛЕМИ ВИЯВЛЕННЯ DEEPFAKE

1.1 Нейронні мережі

Глибокі нейромережі являються областю машинного навчання, що невпинно розвивається. Розвиток цього напряму поштовхнув на появу великої кількості алгоритмів, побудованих на глибокому навчанні. Нейромережі використовують усі основні алгоритми для генерації підробних зображень.

Нейронні мережі відображають поведінку людського мозку, дозволяючи комп’ютерним програмам розпізнавати закономірності та вирішувати загальні проблеми в області ШІ, машинного навчання та глибокого навчання.

Нейронні мережі, також відомі як штучні нейронні мережі (ШНМ) або змодельовані нейронні мережі (ЗНМ), є підмножиною машинного навчання і лежать в основі алгоритмів глибокого навчання. Їх назва та структура натхненні людським мозком, імітуючи спосіб, яким біологічні нейрони сигналізують один одному.

Штучні нейронні мережі (ШНМ) складаються з шарів вузлів, що містять вхідний шар, один або кілька прихованих шарів і вихідний шар. Кожен вузол або штучний нейрон з’єднується з іншим і має відповідну вагу та поріг. Якщо вихід будь-якого окремого вузла перевищує вказане порогове значення, цей вузол активується, надсилаючи дані на наступний рівень мережі. В іншому випадку дані не передаються на наступний рівень мережі.

Нейронні мережі покладаються на навчальні дані, щоб з часом вивчати та покращувати свою точність. Однак, як тільки ці алгоритми навчання будуть максимально налаштовані на точність, вони стануть потужними інструментами в інформатиці та штучному інтелекті, що дозволить нам класифікувати та кластерувати дані з високою швидкістю. Завдання розпізнавання мовлення або зображення можуть займати хвилини та години, якщо порівнювати з ручною ідентифікацією експертів. Однією з найвідоміших нейронних мереж є пошуковий алгоритм Google.

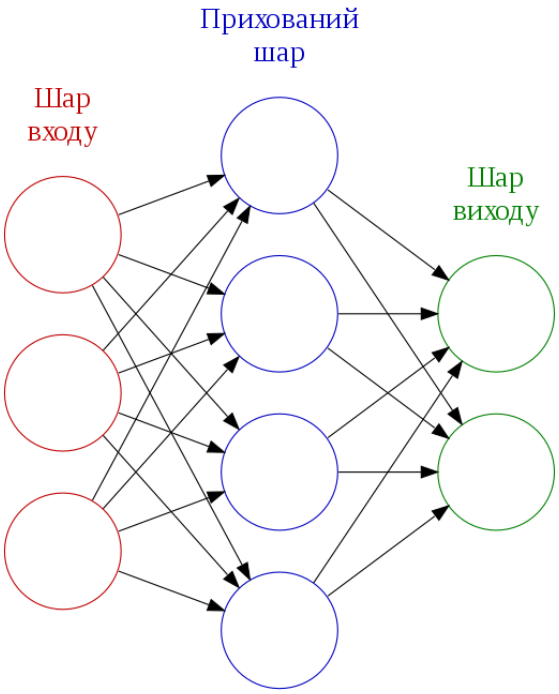

Рисунок 1.1 – Структура нейронної мережі

Інформація протікає через нейронну мережу двома способами. Коли вона навчається (тренується) або працює нормально (після навчання), шаблони інформації надходять у мережу через вхідні блоки, які запускають шари прихованих блоків, а ті, у свою чергу, надходять до вихідних. Ця поширена конструкція називається мережею прямого зв’язку. Не всі підрозділи працюють постійно. Кожен блок отримує вхідні дані від одиниць зліва, і вхідні дані помножуються на вагу з'єднань, по яких вони рухаються. Кожен блок сумує всі вхідні дані, які він отримує таким чином, і (у найпростішому типі мережі), якщо сума перевищує певне порогове значення, блок запускає одиниці, до яких він підключений (ті, що праворуч) .

Щоб нейронна мережа навчалася, має бути задіяний елемент зворотного зв’язку. Нейронні мережі тренуються за допомогою процесу зворотного зв’язку, який називається зворотним поширенням. Це включає порівняння результату, який виробляє мережа, з результатом, який вона повинна була виробляти, і використання різниці між ними для зміни ваги зв’язків між одиницями в мережі, працюючи від вихідних одиниць через приховані блоки

до вхідних. З часом зворотне розповсюдження змушує мережу вчитися, зменшуючи різницю між фактичним і передбачуваним виходом до точки, де обидва вони точно збігаються.

Нейронні мережі можна розділити на різні типи, які використовуються для різних цілей. Хоча це не вичерпний список типів, наведене нижче буде репрезентувати найпоширеніші типи нейронних мереж:

1. Персептрон – найстаріша нейронна мережа, створена Френком Розенблатом у 1958 році. Вона має один нейрон і є найпростішою формою нейронної мережі.

2. Нейронні мережі прямого зв’язку або багатошарові персептрони – мережі, які складаються з вхідного шару, прихованого шару або шарів і вихідного шару. Важливо зазначити, що ці нейронні мережі насправді складаються з сигмовидних нейронів, а не з персептронів, оскільки більшість проблем реального світу є нелінійними. Дані зазвичай подаються в ці моделі для їх навчання, і вони є основою для комп’ютерного зору, обробки природної мови та інших нейронних мереж.

3. Згорткові нейронні мережі (ЗНМ) подібні до мереж з прямим зв’язком, але вони зазвичай використовуються для розпізнавання зображень, візерунків та/або комп’ютерного зору. Ці мережі використовують принципи лінійної алгебри, зокрема множення матриці, для визначення шаблонів у зображенні. Вони мають три основних типи шарів, а саме:

- згортковий шар;
- об’єднуючий шар;
- повнозв’зний шар.

Згортковий шар є першим шаром згорткової мережі. У той час як за згортковими шарами можуть слідувати додаткові згорткові шари або шари об’єднання, повністю пов’язаний шар є останнім шаром. З кожним шаром ЗНМ збільшується у своїй складності, ідентифікуючи більші частини зображення. Більш ранні шари зосереджені на простих елементах, таких як кольори та краї. Коли дані зображення просуваються через шари ЗНМ, вона

починає розпізнавати більші елементи або форми об’єкта, поки нарешті не ідентифікує передбачуваний об’єкт.

Згортковий шар є основним будівельним блоком CNN, і саме там відбувається більшість обчислень. Для цього потрібно кілька компонентів, а саме вхідні дані, фільтр і карта об’єктів. Припустимо, що вхідним буде кольорове зображення, яке складається з матриці пікселів у 3D. Це означає, що вхідні дані будуть мати три виміри – висоту, ширину та глибину – які відповідають RGB-зображенню. Згортковий шар має детектор функцій, також відомий як ядро або фільтр, який переміщається по сприйнятливих полях зображення, перевіряючи, чи є функція теперішньою. Цей процес відомий як згортка.

Детектор ознак – це двовимірний (2-D) масив ваг, який представляє частину зображення. Хоча вони можуть відрізнятися за розміром, розмір фільтра зазвичай є матрицею 3×3; це також визначає розмір рецептивного поля. Потім фільтр застосовується до області зображення, і між вхідними пікселями та фільтром обчислюється крапковий добуток. Цей точковий добуток потім подається у вихідний масив. Після цього фільтр зміщується на один крок, повторюючи процес, поки ядро не охопить усе зображення. Остаточний вихід із ряду точкових добутків із входу та фільтра відомий як карта ознак, карта активації або згорнута функція.

Об’єднання шарів, також відоме як зниження дискретизації, зменшує розмірність, зменшуючи кількість параметрів у вхідних даних. Подібно до згорткового шару, операція об’єднання об’єднує фільтр по всьому входу, але різниця в тому, що цей фільтр не має жодних ваг. Замість цього ядро застосовує функцію агрегації до значень у сприйнятливому полі, заповнюючи вихідний масив.

Назва повнозв'язного шару влучно описує себе. У повнозв’язному шарі кожен вузол вихідного шару підключається безпосередньо до вузла попереднього шару. Цей шар виконує завдання класифікації на основі ознак, витягнутих через попередні шари та їх різні фільтри.

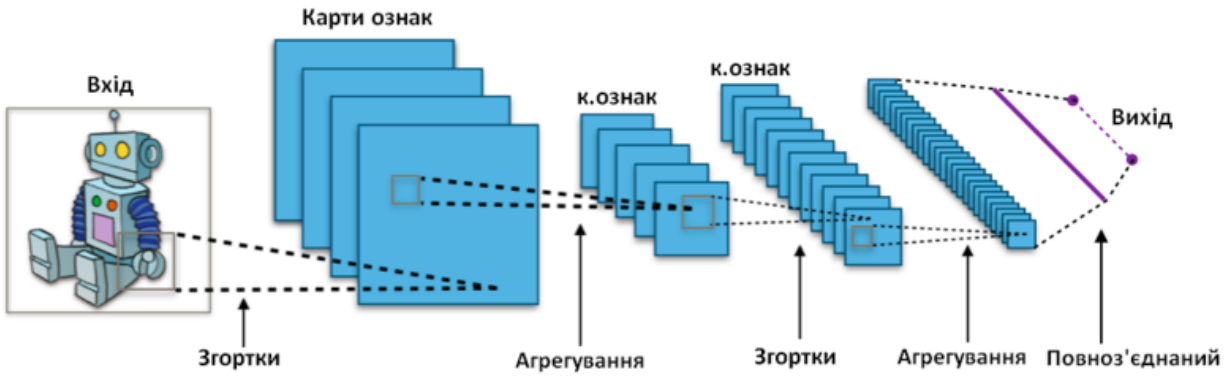

Рисунок 1.2 – Структура згорткової нейронної мережі

4. Рекурентні нейронні мережі ідентифікуються за їх петлями зворотного зв'язку. Ці алгоритми навчання в основному використовуються під час використання даних часових рядів для прогнозування майбутніх результатів, таких як прогнози фондового ринку або прогнозування продажів.

1.2 Опис технології Deepfake

На перший погляд технологія Deepfake може здаватися веселим, навіть корисним інструментом. Навіть високобюджетні фільми, використовували технологію Deepfake, щоб повернути молодші версії улюблених персонажів. Але переваги цієї технології мінімальні в порівнянні з потенційною небезпекою, яку вона несе. Сьогодні ця технологія використовується для порнографії, фейкових новин, містифікацій, знущань, фінансового шахрайства тощо. У марті 2022 року дипфейк президента України Володимира Зеленського, який закликає своїх солдатів скласти зброю, потрапив на зламаний український новинний веб-сайт, а потім став вірусним, перш ніж було визнано, що він є фейком. Хоча точні збитки від цього інциденту залишаються невідомими, він продемонстрував одне з найнебезпечніших застосувань цієї технології. Якби це відео не було швидко позначено як фейк, воно могло б привести до втрат і змінити хід конфлікту. На більш інтимному та особистому рівні цю саму технологію можна

використовувати як форму фішингу на робочому місці. Хоча хакери зазвичай використовуються у текстовій формі, щоб видавати себе за вашого боса чи колегу, тепер хакери можуть зателефонувати вам у Zoom, видавати себе за вашого боса та змусити вас придбати предмети або передати конфіденційну інформацію.

Термін «Deepfake» походить від «Deep Learning» (Глибоке навчання) і «Fake» (Фейк), і він описує конкретні фотореалістичні відео або зображення, створені за підтримки глибокого навчання. Це слово було названо на честь анонімного користувача Reddit наприкінці 2017 року, який застосував методи глибокого навчання, щоб замінити обличчя людини в порнографічних відео, використовуючи обличчя іншої людини, і створив фотореалістичні підроблені відео. Для створення таких підроблених відео використовувалися дві нейронні мережі: генеративна мережа та дискримінаційна мережа з технікою FaceSwap. Генеративна мережа створює підроблені зображення за допомогою кодера та декодера. Дискримінаційна мережа визначає автентичність знову створених зображень. Комбінація цих двох мереж називається Генеративною змагальною мережею (GAN).

Генеративні змагальні мережі (GAN) — це алгоритмічні архітектури, які використовують дві нейронні мережі, налаштовуючи одну проти іншої (тому «змагальні»), щоб генерувати нові синтетичні екземпляри даних, які можуть передаватись як реальні дані. Вони широко використовуються при генерації зображень, відео та голосу.

GAN були представлені в статті Яна Гудфеллоу[2] та інших дослідників з Університету Монреаля, включаючи Йошуа Бенджіо, у 2014 році. Згадуючи GAN, керівник дослідження ШІ Facebook Янн Лекун назвав змагальне навчання «найцікавішою ідеєю за останні 10 років у Машинному Вченні».

Потенціал GAN як для добра, так і для зла величезний, оскільки вони можуть навчитися імітувати будь-який розподіл даних. Тобто GAN можна навчити створювати світи, моторошно схожі на наш, у будь-якій області: зображення, музика, мова, проза. У певному сенсі вони художники-роботи, і

їхні результати вражають – навіть пронизливі. Але їх також можна використовувати для створення підробленого медіа-контенту, і вони є технологією, що лежить в основі Deepfake.

GAN працює наступним чином:

Одна нейронна мережа, яка називається нові генератором, генерує екземпляри даних, а інший, дискримінатор, оцінює їх на достовірність; тобто дискримінатор вирішує, чи є кожен екземпляр даних, який він переглядає, до фактичного набору даних для навчання чи ні.

Наприклад, ми збираємось генерувати рукописні цифри, подібні до тих, що містяться в наборі даних MNIST [2], який береться з реального світу. Метою дискримінатора, коли він показує екземпляр із справжнього набору даних MNIST, є розпізнавання тих, які є автентичними. Тим часом генератор створює нові синтетичні образи, які передає дискримінатор. Це робиться в надії, що вони будуть визнані справжніми, навіть якщо вони підроблені. Мета генератора – генерувати прохідні рукописні цифри: брехати, не будучи спійманим. Мета дискримінатора – ідентифікувати зображення, яке надходить від генератора, як підробку. Ось кроки, які виконують GAN.

- Генератор приймає випадкові числа і повертає зображення.
- Це згенероване зображення подається в дискримінатор разом із зображенням потоку, беручи з фактичного набору даних, що відповідає дійсності.
- Дискримінатор як реальні, так і підроблені зображення та повертає ймовірність, число від 0 до 1, чому 1 означає прогноз достовірності, а 0 – підробку.

Отже, маємо подвійний цикл зворотного зв’язку:

Дискримінатор знаходиться в циклі зворотного зв’язку з основною істинністю зображення, яку ми знаємо. Генератор знаходиться в петлі зворотного зв'язку з дискримінатором.

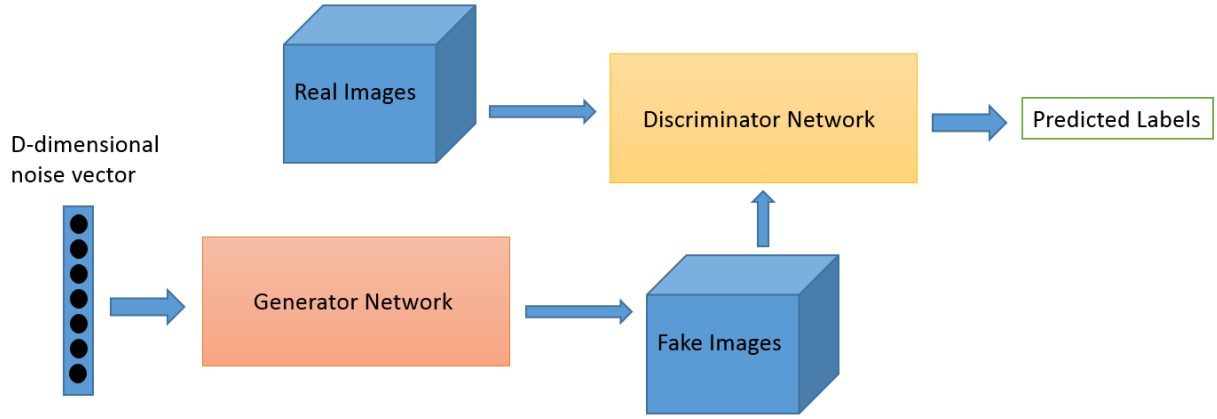

Рисунок 1.3 – Схема GAN

Буде корисним порівняти генеративні змагальні мережі з іншими нейронними мережами, такими як автокодери та варіаційні автокодери.

Автокодери [3] кодують вхідні дані як вектори. Вони створюють приховане або стиснене представлення необроблених даних. Вони корисні для зменшення розмірності; тобто вектор, що служить прихованим уявленням, стискає вихідні дані в меншу кількість помітних вимірів. Автокодери можна об'єднати з так званим декодером, який дозволяє відновлювати вхідні дані на основі їх прихованого представлення.

Варіаційні автокодери – це генеративний алгоритм, який додає додаткове обмеження для кодування вхідних даних, а саме, що приховані уявлення нормалізуються. Варіаційні автокодери здатні як стискати дані, як автокодер, так і синтезувати дані, як GAN. Однак, хоча GAN генерує дані з дрібними деталями, зображення, створені VAE, мають тенденцію бути більш розмитими.

Багато додатків для створення дипфейків існують вже кілька років.

FakeApp – це перший метод, який широко використовувався для створення дипфейків. Цей метод здатний міняти обличчя на відео за допомогою структури парування автокодер-декодер, розробленої користувачем Reddit. Подібно до GAN, FakeApp складається з автокодера, який використовується для створення прихованих рис зображень людського обличчя, і декодера, який використовується для повторного вилучення рис

для зображень обличчя людини. Ця проста техніка є потужною, оскільки вона здатна створювати надзвичайно реалістичні підроблені відео, які людям важко відрізнити від справжнього.

VGGFace – це ще одна популярна техніка створення дипфейку, заснована на генеративної змагальної мережі (GAN). Архітектура VGGFace покращена шляхом додавання двох шарів, які називаються втратою протиборства та втратою сприйняття. Ці шари додаються до автоматичного кодування-декодера, щоб фіксувати приховані особливості зображень обличчя, такі як рухи очей, щоб створити більш правдоподібні та реалістичні підроблені зображення.

CycleGAN – це метод створення дипфейку, який витягує характеристики одного зображення та створює інше зображення з такими ж характеристиками за допомогою архітектури GAN. Цей метод застосовує функцію втрати циклу, яка дає змогу вивчати приховані особливості. На відміну від FakeApp, CycleGAN є неконтрольованим методом, який може виконувати перетворення зображення в зображення без використання парних прикладів. Іншими словами, модель вивчає особливості колекції зображень з джерела та цілі, які не повинні бути пов’язані один з одним.

1.3 Методи виявлення Deepfake

Протягом останніх років методи глибокого навчання успішно застосовуються для виявлення підроблених зображень. Однак сучасні методи глибокого навчання зображення не можуть бути безпосередньо застосовані для виявлення підроблених відео через наявність значної втрати інформації кадру після стиснення відео. Методи виявлення дипфейків можна поділити на дві основні категорії: завдяки аналізу Біологічний аналіз одиниць та завдяки аналізу просторових і часових особливостей.

Біологічний аналіз поодиноких осіб. Юезен Лі [4] представив новий підхід, заснований на природній мережі для виявлення фальшивих облич у відео. Цей метод передбачає моргання очей для виявлення підроблених відео,

що є важливою фізичною ознакою, за допомогою якої можна розрізняти підроблені відео. Щоб досягти цього, цей метод використовує згорткову нейронну мережу з рекурсивною нейронною мережею для виявлення фізіологічних сигналів, таких як рух очей і моргання. Потім модель використовує двійковий класифікатор для визначення стану закритих і відкритих очей. Цей підхід перевірено за допомогою набору даних під назвою ‘eye-blinking’(моргання очей), який сканується з Інтернету. Набір даних з морганням очей — це перший доступний набір даних, спеціально розроблений для виявлення моргання очей. Результати експерименту демонструють ефективність запропонованого підходу у виявленні помилкових зображень.

Інші біологічні сигнали, такі як серцебиття, як було показано, є надійним провісником для реального відео. Ціфці[5] та інші розробив модель на основі генеративної змагальної мережі (GAN), яка може виявляти джерело глибокого фейку, аналізуючи «серцебиття» глибоких підробок. Запропонована модель починається з кількох мереж детекторів, де вхідним сигналом для цієї моделі є реальне відео. Потім пара реалістичного відео та підробленого відео призначається іншому шару, який називається реєстрацією, який витягує цікаві ділянки обличчя і біологічні сигнали для створення клітин фотоплетизмографії (ФПГ). Тут осередки ФПГ є просторово-часовими вікнами, які містять кілька облич, витягнутих за допомогою детектора облич. Останній шар відповідає за класифікацію відео як підробленого чи справжнього. Автори використали кілька загальнодоступних наборів даних для перевірки своєї моделі. Результат показує, що моделі активують точність 97,3% у виявленні глибоких фейків.

Аналіз просторових і часових ознак. Більшість сучасних методів виявлення глибоких фейків використовують лише один відеокадр[6]. Фактично, маніпуляції з відео можна виконувати з кількома функціями на рівні кадру. Останнім часом багато досліджень показали, що аналіз тимчасової послідовності між кадрами може успішно допомогти відрізнити

реальне відео від підробленого. У цій роботі автори представили тимчасову модель для виявлення фейкових відео. У моделі спочатку використовується згорткова нейронна мережа для виділення ознак кадру. Згодом ці ознаки передаються на шар ДКЧП (довгої короткочасної пам'яті) для аналізу тимчасової послідовності для маніпуляції обличчя між кадрами. Нарешті, функція softmax використовується для класифікації відео як реального чи підробленого. Для оцінки було зібрано колекцію з 600 відео з кількох веб-сайтів. Результати експерименту свідчать про ефективність цієї моделі для виявлення глибоких фейків.

На основі попередньої версії Cycle-GAN [7] було представлено новий підхід під назвою Recycle-GAN [8], який використовує умовні генеративні змагальні мережі для злиття просторових і часових даних. Результати оцінки показують, що поєднання просторових і часових обмежень може дати ефективний результат. Крім того, також було запропоновано новий підхід [9], заснований на рекурентній згортковій мережі. Підхід складається з двох етапів аналізу: етап обробки обличчя з подальшим виявленням маніпуляції обличчям. Під час обробки кадрування та вирівнювання обличчя витягується за допомогою мережі просторового трансформатора. Потім вихідні дані з попередніх етапів передаються для виявлення маніпуляцій обличчям за допомогою рекурентної згорткової мережі, де аналізується тимчасова інформація по кадрах.

У даному розділі проведено аналіз принципу роботи технології Deepfake та найвідоміших існуючих типів нейронних мереж, які можуть використовуватися для створення цієї технології, також проаналізовані сучасні методи виявлення Deepfake. Проведений аналіз показав, що згорткова нейронна мережа є найкращим варіантом для розпізнавання зображень, тому доцільно використовувати саме її для побудови архітектури.

Таким чином, у першому розділі вирішені задачі один та два з переліку завдань, поставлених у кваліфікаційній роботі.

2 АНАЛІЗ АЛГОРИТМУ ІДЕНТИФІКАЦІЇ DEEPFAKE

2.1 Вибір моделі нейронної мережі

Модель нейронної мережі заснована на статті, опублікованій Даріусом Афчаром та ін. у 2018 році [2]. Це двійковий класифікатор, побудований як відносно неглибока згорткова нейронної мережі, навчена класифікувати зображення на один із двох класів. Один клас відноситься до «реальних» зображень (зображення реальних людей), а інший — до «підроблених» зображень (зображень, створених Deepfake ШІ).

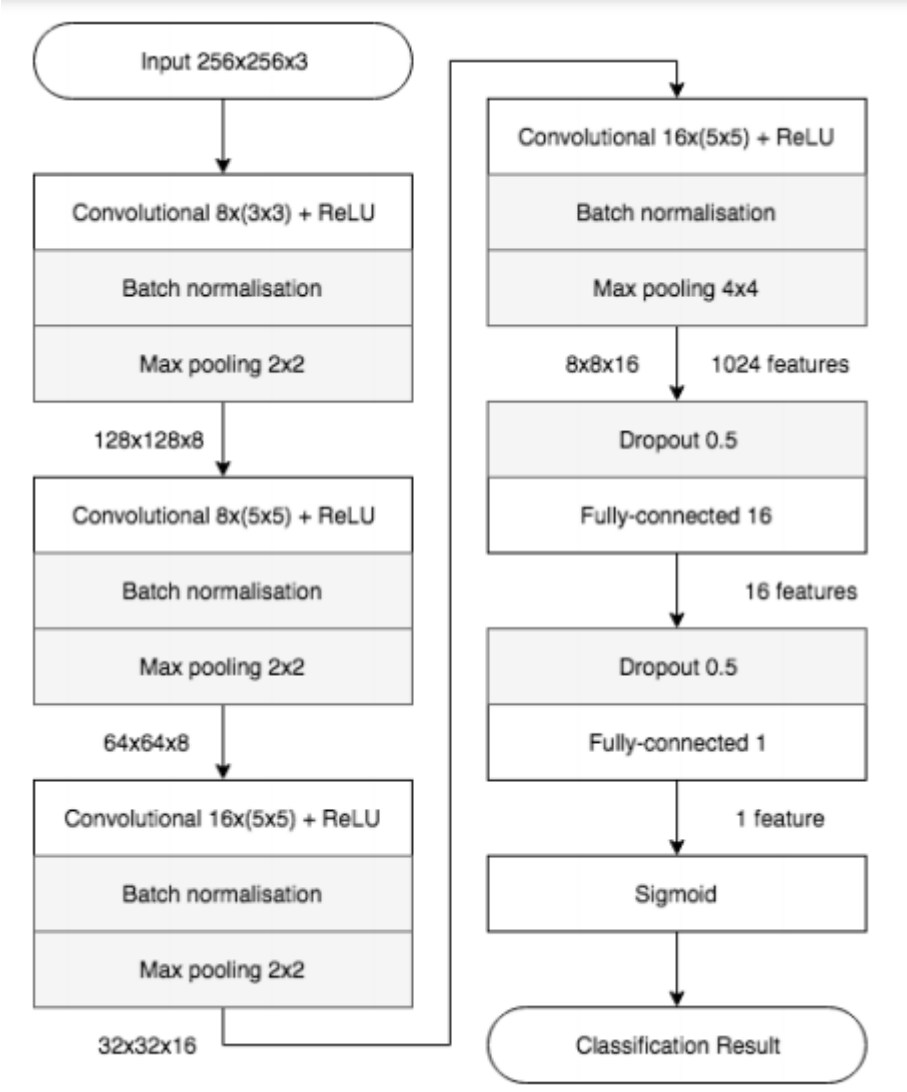

Рисунок 2.1 – Модель нейронної мережі

- Вхідний шар 3 X 256 X 256 з масштабуванням вхідних даних на 255 і збільшеннями доданими до нього.
- Згортковий шар із 8 фільтрами, розміром 3 x 3 і кроком 1, за яким слідує максимальний об’єднуючий шар розміром 2 x 2.
- Згортковий шар із 8 фільтрами, розміром 5 x 5 і кроком 1, за яким слідує максимальний об’єднуючий шар розміром 2 x 2.
- Два згорткових шари з 16 фільтрами, розміром 5 x 5 і кроком 1, за якими слідують максимальні шари об’єднання з вікном об’єднання 2 x 2.
- Повнозв'язний шар з 16 нейронами.
- Повністю підключений вихідний шар з одним блоком і сигмовидною активацією.

Згортковий шар, представлений conv2d, є найважливішою частиною. Тут встановлюється розмір та кількість фільтрів, які використовуватимуться у згортці. Кожен фільтр є окремим елементом зображення, наприклад, горизонтальна лінію. Під час згортки цей фільтр проходить по зображенню, щоб оцінити, якою мірою певні області цього зображення відповідають фільтру. Після згорткового шару слідує шар пакетної нормалізації. Пакетна нормалізація – це новий метод підвищення швидкості та стабільності нейронних мереж. Він працює шляхом нормалізації вхідних даних для кожного шару мережі, що зменшує взаємозалежність між параметрами даного шару та вхідним розподілом наступного шару. Ця взаємозалежність називається внутрішнім коваріантним зрушенням і надає дестабілізуючу дію на процес навчання. Останній шар наших згорткових блоків – це шар пулу. Саме на рівні пулу значно зменшується розмірність даних, що значно прискорює обчислення. У даній моделі використовується максимальний пул цього шару, що означає зменшення області значень пікселів до максимального значення цієї області. Незважаючи на те, що ця архітектура ретельно дотримується, були проведені експерименти з різними функціями активації. Зокрема, крім використання стандартної активації ReLU, також були проведені експерименти з ELU [2] та LeakyReLU [3].

ReLU є функцією активації вибору, оскільки немає очевидного ризику мертвих нейронів. Крім того, існує активація LeakyReLU після повністю підключеного шару з 16 одиниць. За цим немає жодної видимої причини, крім того, що використовує газета.

Dropout – випадання, додається після повністю підключеного шару з 16 нейронами для боротьби з перенавчанням. Перенавчання — одна з проблем глибоких нейронних мереж (Deep Neural Networks, DNN), яка полягає в наступному: модель добре пояснює лише приклади з навчальної вибірки, адаптуючись до навчальних прикладів, замість того, щоб вчитися класифікувати приклади, що не брали участь у навчанні (втрата здатності до узагальнення). За останні роки було запропоновано безліч рішень проблеми перенавчання, але одне з них перевершило всі інші, завдяки своїй простоті та чудовим практичним результатам; це рішення – Dropout. Головна ідея Dropout – замість навчання однієї DNN навчити ансамбль кількох DNN, а потім усереднити отримані результати. Мережі для навчання виходять за допомогою виключення з мережі (dropping out) нейронів із ймовірністю p, таким чином, ймовірність того, що нейрон залишиться в мережі, становить q=1-p. «Виключення» нейрона означає, що при будь-яких вхідних даних або параметрах він повертає 0. Виключені нейрони не роблять свій внесок у процес навчання на жодному з етапів алгоритму зворотного поширення помилки (backpropagation); тому виняток хоча б одного з нейронів рівносильне навчанню нової нейронної мережі.

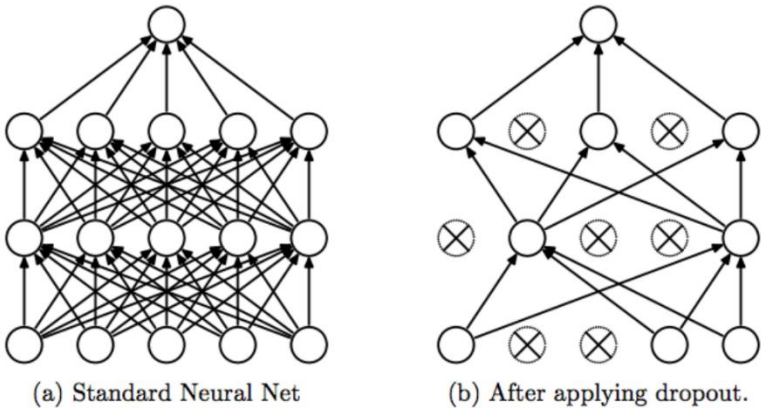

Рисунок 2.2 – НМ a) –до застосування Dropout, b) – після Dropout

Flatten (згладжування) – це перетворення даних в одновимірний масив для введення їх у наступний шар. Вирівнюються вихідні дані згорткових шарів, щоб створити один довгий вектор ознак. І це пов’язано з остаточною моделлю класифікації, яка називається повністю зв’язаним шаром. Іншими словами, відбувається поміщення всіх піксельних даних в один рядок і з’єднання з останнім шаром.

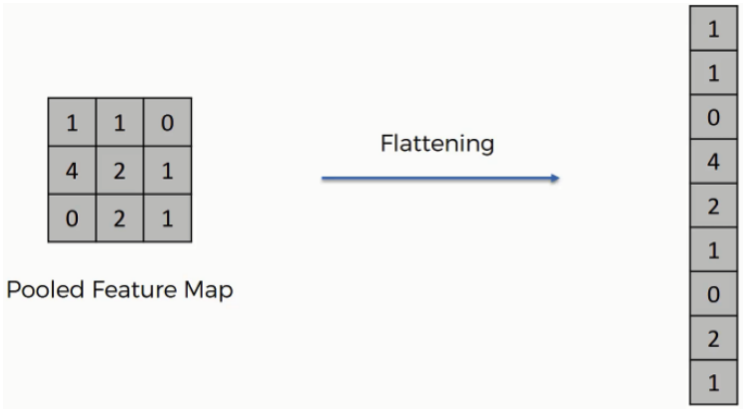

Рисунок 2.3 – Приклад застосування Flatten

Dense (повнозв’язний шар) – це простий шар нейронів, в якому кожен нейрон отримує вхідні дані від усіх нейронів попереднього шару, тому його називають щільним.

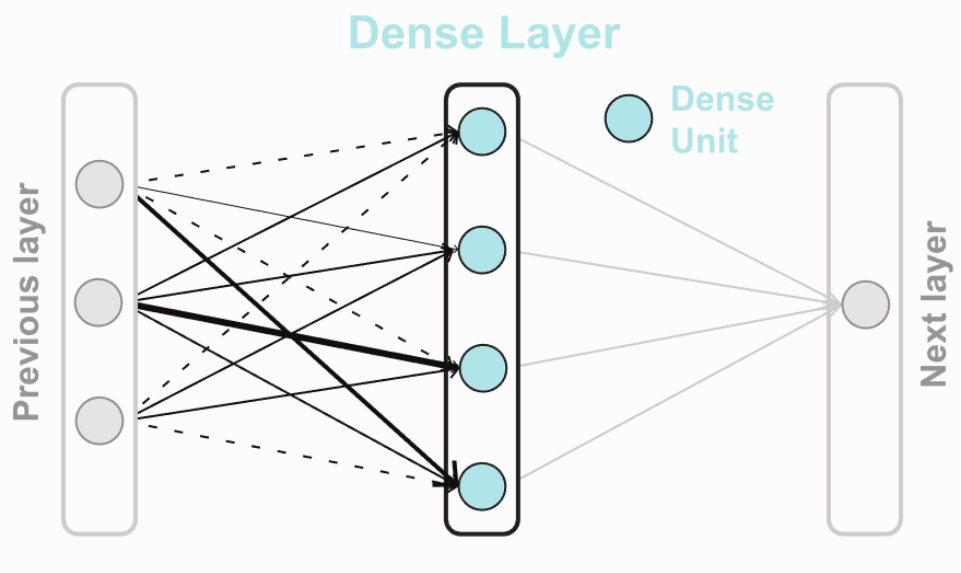

Рисунок 2.4 – Приклад застосування Dense

Повнозв’язний шар використовується для класифікації зображень на основі вихідних даних із згорткових шарів. Робота одного нейрона. Шар містить кілька таких нейронів.

2.2 Навчання нейронної мережі

Для навчання нейронної мережі був використаний набір даних діпфейків, який складався з зображень, які були вилучені близько зі 175 існуючих відео, знятих з популярних платформ діпфейків, та модифікований, а саме з кожного зображення було вилучено обличчя і таким чином був створений новий набір даних для навчання нейронної мережі. Для вилучення облич використовувався метод HOG (гістограми орієнтованих градієнтів), який є дескриптором ознак і використовується в комп'ютерному баченні та обробці зображень з метою виявлення об'єктів. Ця техніка підраховує випадки орієнтації градієнта в локалізованій частині зображення. Цей метод дуже схожий на гістограми орієнтації країв і перетворення ознак, інваріантне масштабу. Дескриптор HOG фокусується на структурі або формі об'єкта. Це краще, ніж будь-який дескриптор краю, оскільки він використовує величину, а також кут градієнта для обчислення ознак. Для областей зображення він генерує гістограми, використовуючи величину та орієнтацію градієнта. Градієнти розраховуються в межах зображення на блок. Блок розглядається як піксельна сітка, в якій градієнти складаються з величини і напрямку зміни інтенсивності пікселя всередині блоку.

а) б)

Рисунок 2.5 – Зображення з набору даних а і б: а) - оригінальне зображення; б) модифіковане зображення

Модифікація набору даних була зроблена для збільшення якості навчання нейронної мережі. Фінальний набір даних складає 11 780 зображень, які діляться на реальні зображення та діпфейки. Вхідними даними є зображення розміром 256×256×3, де 256×256 – висота і ширина зображення в пікселях, 3 – кількість кольорових каналів.

Для навчання був використаний оптимайзер Adam – один із найефективніших алгоритмів оптимізації у навчанні нейронних мереж. Він поєднує в собі ідеї RMSProp [4] та оптимізатора імпульсу.

Після аналізу великої кількості тестувань на більш малому наборі даних, який складав 1000 зображень, при швидкості навчання рівної 0.001 та чотирьом епохам було ідентифіковано, що найкращі результати у тренуванні та тестуванні показала модель нейронної мережі, де шар Dropout змінений з 0,5 до 0,48:

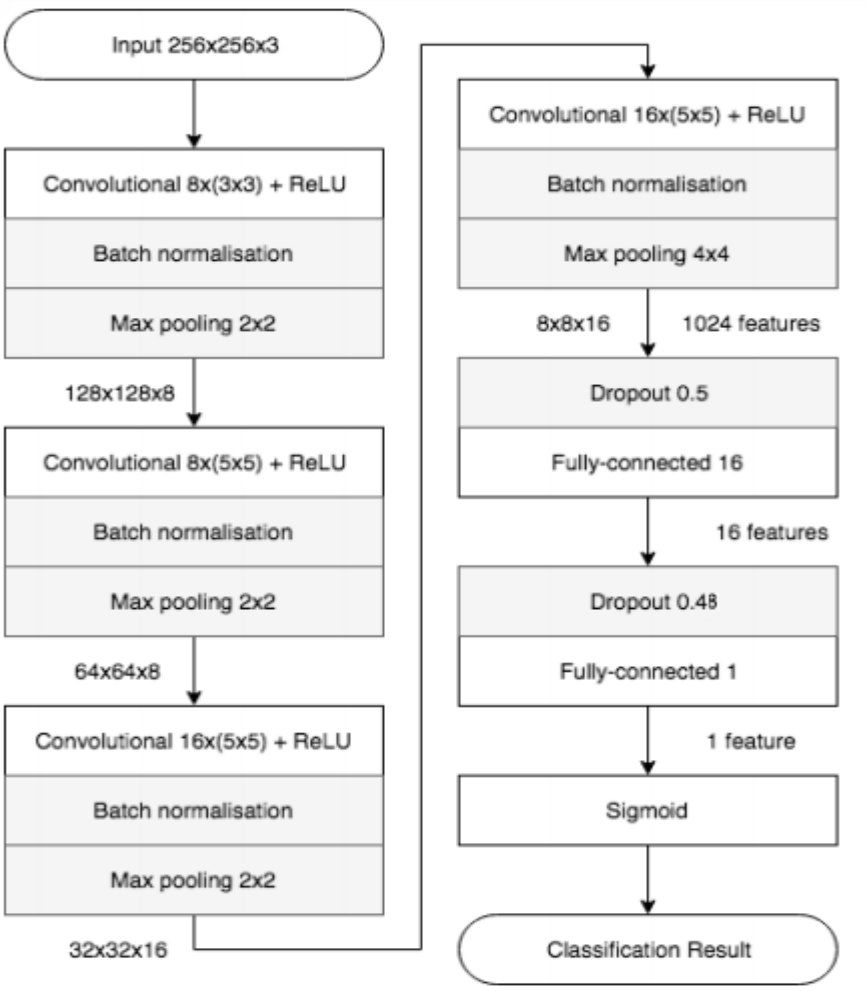

Рисунок 2.6 – Змінена модель нейронної мережі

А саме після 2 епохи була встановлена максимальна точність підтвердження (validation accuracy) – 98,57% та мінімальна втрата підтвердження (validation loss) – 1,26%. В той час на моделі нейронної мережі [2]: максимальна точність підтвердження (validation accuracy) – 97,58% та мінімальна втрата підтвердження (validation loss) – 1,93%.

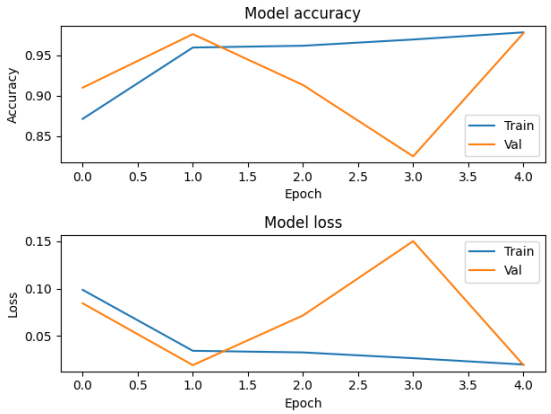

Рисунок 2.7 – Графік тестового навчання моделі нейронної мережі [2]

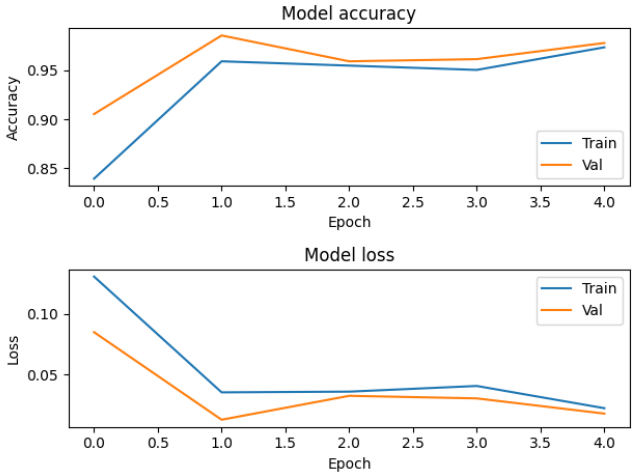

Рисунок 2.8 – Графік тестового навчання модифікованої моделі нейронної мережі

При більшому розмірі набору даних, а саме – 11780 зображень, при швидкості навчання рівної 0.001 та 30 епохам у модифікованої моделі, найкращий результат було отримано на 28 епосі :

loss	2,44 %
acc	96,89%
val_loss	7,34 %
val_acc	91,17%

Таблиця 2.1 – Результат навчання моделі модифікованої моделі нейронної мережі

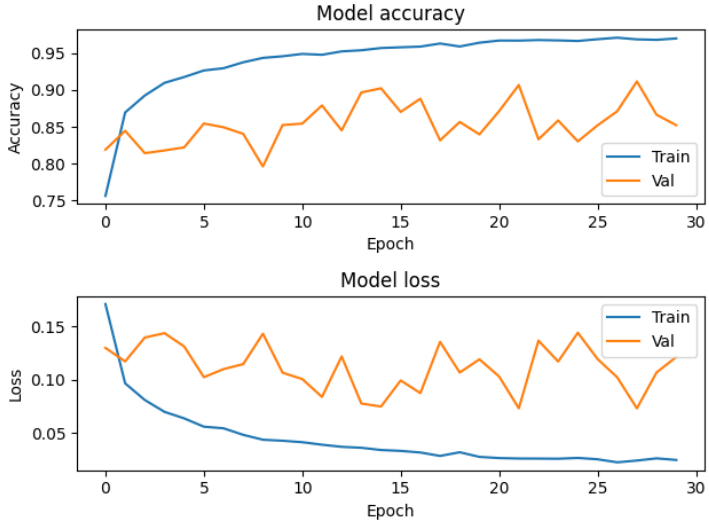

Рисунок 2.9 – Графік тестового навчання модифікованої моделі нейронної мережі

У даному розділі описана модель нейронної мережі та описані методи обробки зображень для створення набору даних, який в подальшому був використаний для навчання та перевірки нейронної мережі.

Точність класифікації створеної моделі є 91,17%, що є досить великим показником, але крім того, завдяки модифікації набору даних вдалося підвищити точність класифікації зображень, які не входять до використаного для навчання моделі набору даних.

Таким чином, у другому розділі вирішена задача три з переліку завдань, поставлених у кваліфікаційній роботі.

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Обгрунтування вибору програмного середовища

Програмний код для навчання нейронної мережі, описаної в другому розділі та методів для обробки вхідного реалізовано мовою програмування Python в програмному середовищі PyCharm 2022.1.1 за допомогою інтерпретатора anaconda3.

PyCharm – інтегроване середовище розробки мови Python. Надає засоби для аналізу коду, графічний налагоджувач, інструмент для запуску юніт-тестів і є дуже зручним завдяки рефакторінга та автодоповнення коду.

Віртуальне середовище, створене за допомогою Anaconda. Anaconda – дистрибутив мов програмування Python і R [5], що включає набір популярних вільних бібліотек, об'єднаних проблематиками науки про дані та машинне навчання. Основна мета – постачання єдиним узгодженим комплектом найбільш необхідних відповідному колу користувачів тематичних модулів (таких як NumPy [6], SciPy [7], Astropy [8] та інших) з вирішенням залежностей і конфліктів, що виникають, які неминучі при одиночній установці.

Основна особливість дистрибутива – оригінальний менеджер роздільної здатності залежностей conda з графічним інтерфейсом Anaconda Navigator, що дозволяє відмовитися від стандартних менеджерів пакетів (таких, як pip для Python). Дистрибутив завантажується один раз, і вся наступна конфігурація, у тому числі встановлення додаткових модулів, може проводитись в офлайні. Крім того, забезпечується можливість ведення декількох ізольованих середовищ з роздільною роздільною здатністю версійних залежностей у кожній.

Дистрибутив Anaconda має широкі можливості під'єднання модулів (більше 1500), зокрема, власним Conda та менеджером віртуального середовища. Графічний інтерфейс, Anaconda Navigator слугує графічною альтернативою інтерфейсові командного рядка (CLI).

Велика різниця між Conda та менеджером пакетів Pip полягає в тому, як управляти залежностями під'єднуваних пакунків, що є суттєвим викликом при роботі з Data Science у Python та головною причиною появи Conda.

Коли Pip встановлює деякий потрібний клієнтові пакет, то він автоматично встановлює весь перелік залежних від нього пакетів Python, не перевіряючи при цьому, чи вони будуть конфліктувати з раніше встановленими пакунками. Через це користувач із коректно встановленим, наприклад, Google Tensorflow, може виявити, що той різко перестає працювати: Pip при інсталяції нового пакета встановить інакшу версію NumPy (якого потребують одночасно і встановлюваний пакунок і вже наявний Tensorflow), наприклад 3.7, тоді як Tensorflow коректно працюватиме лише з 3.6. У деяких випадках може спочатку здаватися, що новий пакет працює як слід, але насправді він даватиме відмінні від правильних результати, що буде видно лише у деяких подробицях.

На відміну від цього, Conda структурно аналізує все поточне програмне середовище і вже після цього встановлює новий пакет, враховуючи всі обмеження сумісностей, версій різних пакунків, вірніше пропонує поєднуваний набір пакетів. У деяких випадках Conda попередить користувача про неможливість одночасного використання деяких пакетів. Як наслідок, тепер користувач може мати, наприклад, Tensorflow саме версії 2,0 або новішої, обираючи зручний для себе варіант враховуючи розуміння особливостей версій кожного з пакетів. Графічний інтерфейс для програми розроблений за допомогою GUI Tkinter [9] – єдиний фреймворк GUI, який вбудований в стандартну бібліотеку Python.

3.2 Реалізація алгоритму детектування Deepfake

Вхідними даними алгоритму є цифрове зображення або відео, яке підозрюється на наявність фальсифікації, виконаної шляхом застосування технології Deepfake. Тобто інтерфейс програмного продукту має містити

об’єкти для завантаження зображення та відео, що є підозрюваним на наявність Deepfake.

Для того, щоб перейти до реалізації алгоритму детектування Deepfake, необхідно спочатку встановити бібліотеки TensorFlow та Dlib.

Найважливішою бібліотекою для реалізації поставленої задачі є бібліотека TensorFlow. TensorFlow – це бібліотека для машинного навчання, групи технологій, яка дозволяє навчати штучний інтелект вирішенню різних завдань. Бібліотека розроблена на Python з використанням швидкого та продуктивного C++ для вирішення математичних завдань. Тому вона ефективно працює зі складними обчисленнями. Ця бібліотека включає безліч інструментів для різних напрямків ML, але найчастіше використовується для роботи з нейронними мережами. Нейронні мережі складаються із програмних елементів – «нейронів» та зв'язків між ними, і такий пристрій дозволяє їм навчатися. TensorFlow працює зі звичайними та глибокими нейронними мережами різних типів: рекурентними, згортковими тощо. Також вона використовується для машинного та глибокого навчання.

Приклади використання технологій – розпізнавання природної мови, зображень та рукописних текстів, різноманітні завдання класифікації чи кластеризації, обробка великих даних.

У TensorFlow моделі представлені за допомогою графів – математичних абстракцій, що складаються з вершин та шляхів між ними. Граф можна порівняти із схемою доріг між різними точками. У програмуванні це зазвичай необхідно під час вирішення «маршрутних» завдань і за створенні нейронних мереж.

TensorFlow працює з тензорами – багатовимірними структурами даних у векторному, тобто спрямованому просторі. Вони використовуються в лінійній алгебрі та фізиці. Звідси походить назва бібліотеки. За допомогою тензорів описуються шляхи графа, а вершини це математичні операції.

Обчислення TensorFlow виражаються як потоки даних через граф. Це означає, що інформація «рухається» за графом, передається шляхами від вершини до вершини.

Бібліотека може працювати на потужностях звичайного центрального процесора (CPU) або використовувати потужності графічного процесора (GPU). Режим перемикається у коді. Існує спеціальний тензорний процесор TPU, створений розробниками бібліотеки, ним можна скористатися через хмарні сервіси Google.

Переваги TensorFlow.

Високий рівень абстракції. Бібліотека написана так, що не треба думати про технічну реалізацію абстрактних понять. Можна зосередитись на описі логіки програми та на математиці, а спосіб реалізації обчислень – завдання TensorFlow, а не програміста. Це полегшує розробку та дозволяє сконцентруватися на важливих завданнях.

Інтерактивна технологія. TensorFlow дозволяє працювати з компонентами моделі окремо і створювати її на ходу, при цьому окремо перевіряти кожен елемент. Це зручніше, ніж описувати граф як єдину монолітну структуру. Підхід робить розробку більш інтерактивною – структуру можна гнучко налаштовувати та змінювати.

Гнучкість. TensorFlow можна використовувати для створення нейронних мереж, для глибокого навчання та інших напрямів ML. Його функції різноманітні на вирішення широкого спектра завдань. Завдяки графам та тензорам у TensorFlow можна легко зобразити складну математичну структуру. Гнучкість стосується не тільки функцій, а й технічної сторони питання: бібліотека працює і з центральним, і з графічним процесором, її легко використовувати з іншими інструментами машинного навчання. Наприклад, TensorFlow застосовують із API Keras.

Keras – відкрита нейромережна бібліотека, написана мовою Python та спроєктована для уможливлення швидких експериментів з мережами глибинного навчання, її зосереджено на тому, щоби вона була зручною в

користуванні, модульною та розширюваною. У 2017 році команда TensorFlow Google вирішила підтримувати Keras в основній бібліотеці TensorFlow. Шоллє (основний автор бібліотеки Keras) пояснив, що Keras було замислено більше як інтерфейс, аніж як самостійну систему машинного навчання. Вона пропонує високорівневий, інтуїтивніший набір абстракцій, який робить розробку глибинно-нейромережних моделей простою незалежно від використовуваного обчислювального тилу.

Dlib – багатоплатформова бібліотека загального призначення написана на мові C++. Вона була розроблена під суттєвим впливом ідей проектування за контрактом та компонентно-орієнтованого програмування. Таким чином, вона є, перш за все, набором незалежних програмних компонент. Це відкрите програмне забезпечення, яке випускається під ліцензією Boost Software. Оскільки розробка почалася ще в 2002 році, Dlib містить широкий спектр інструментів. Станом на 2016 рік вона містить програмні компоненти для роботи з комп'ютерними мережами, потоками, графічні інтерфейси користувача, структурами даних, лінійною алгеброю, машинним навчанням, обробки зображень, добуванням даних, XML та парсингу тексту, числової оптимізації, Баєсовими мережами та багато іншого. В останні роки, основний розвиток припав на створення широкого спектра статистичних інструментів машинного навчання і в 2009 році Dlib було опубліковано в Journal of Machine Learning Research. Відтоді вона використовується у різних областях. Ця бібліотека містить модуль face_recognition, який, на основі описаного у другому розділі методу HOG, виділяє усі обличчя на зображенні. Наприклад, за допомогою цього модулю можна створити нові зображення, які будуть містити обличчя людей, які були на оригінальному зображенні (Рис. 3.1).

Рисунок 3.1 – Приклад роботи модуля face_recognition

Вхідні зображення складаються з коефіцієнтів RGB від 0 до 255, але такі значення були б занадто високими для нашої моделі для обробки (з огляду на типову швидкість навчання), тому ми націлюємо значення від 0 до 1, масштабуючи за допомогою функції 1/255.

Після цього, застосовується нейронна мережа для прогнозування наявності Deepfake, де, за допомогою методу predict() з бібліотеки Keras генерується вихідний прогноз для вхідного зображення, результатом якого є наявність чи ненаявність діпфейку на зображенні.

Для того, щоб застосувати алгоритм детектування Deepfake для відео, виконуються наступні кроки:

− Відео ділиться на кадри.

− Ідентифікується кількість унікальних облич (людей) на усіх кадрах за допомогою модуля face_recognition.

− Для кожної групи унікальних облич створюється окрема директорія та кожна група копіюється до цієї директорії.

− Алгоритм детектування Deepfake застосовується до кожного обличчя кожної групи окремо, після чого, за умови, що нейронна мережа прогнозує менше 20% кадрів з ймовірністю більше 35% реальності кадру (ненаявності Deepfake), вважається, що відео містить Deepfake.

3.3 Опис програмного продукту

Після запуску розробленого програмного продукту на екрані з’являється вікно, де знаходяться всі елементи управління (рис. 3.2).

Для перевірки зображення чи відео на наявність Deepfake треба виконати наступні дії. Слід натиснути на кнопку «1.Open mediafile» та обрати потрібне зображення чи відео. Після проведеної операції обране відео чи зображення буде відкрито для попереднього перегляду. Після перегляду необхідно закрити відео чи зображення та натиснути кнопку «2.Check». У результаті чого формується прогноз чи містить відео чи зображення діпфейк чи ні, який буде представлений у вигляді строки під лейблом «Result» замість

надпису «Prediction will be here». За допомогою кнопки «Show mediafile» можна додатково переглянути відео чи зображення, обране за допомогою кнопки «1.Open mediafile».

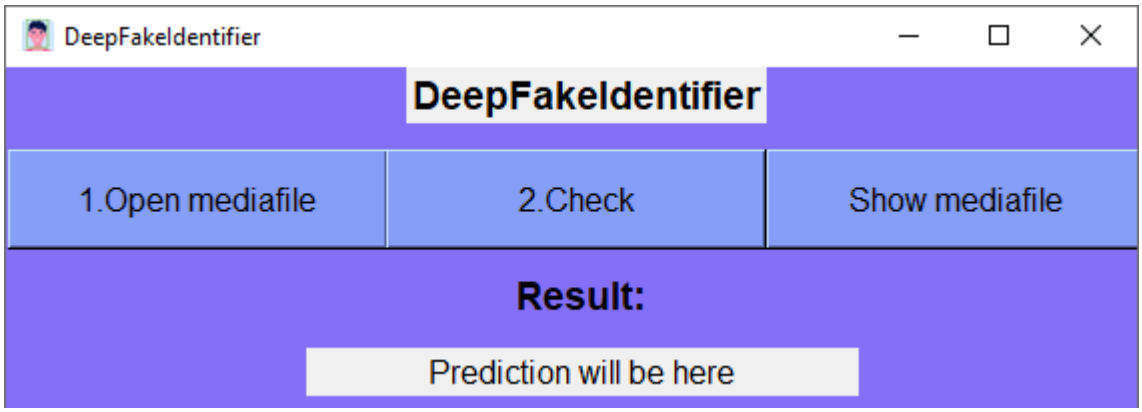

Рисунок 3.2 – Інтерфейс програмного продукту

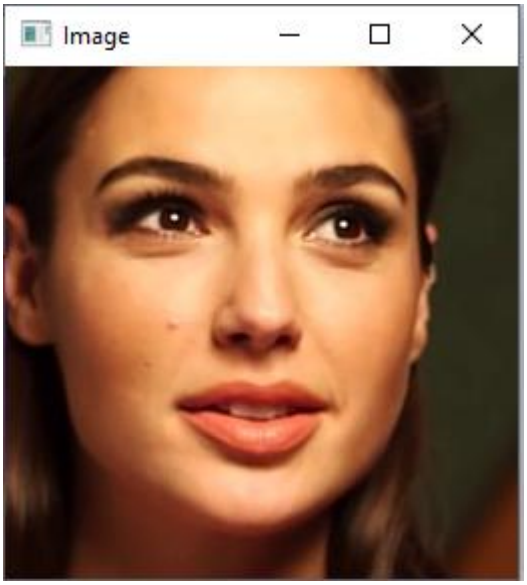

Рисунок 3.3 – Вікно для попереднього перегляду зображення

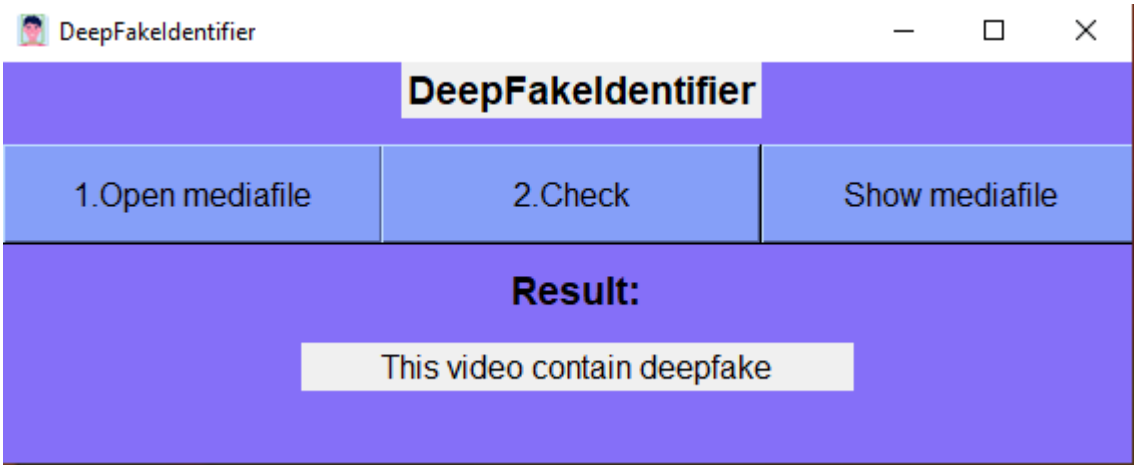

Рисунок 3.4 – Інтерфейс із результатом, що відео містить діпфейк

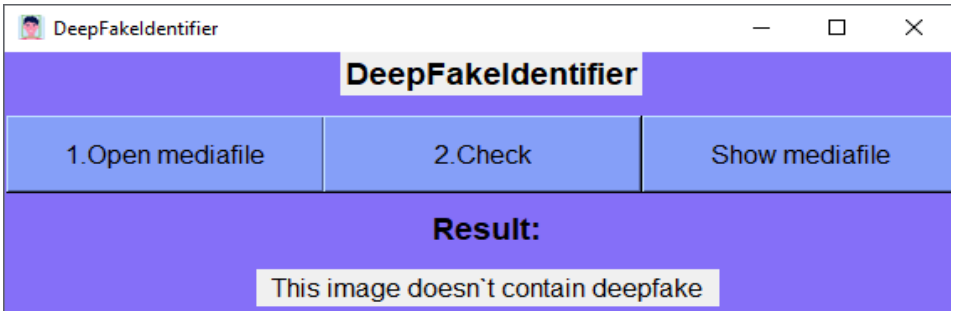

Рисунок 3.5 – Інтерфейс із результатом, що зображення не містить діпфейк

3.4 Системні вимоги для успішної роботи розробленої програми

Для успішної роботи даного програмного продукту необхідно встановити віртуальне середовище Anaconda та встановити пакети TensorFlow, Keras та Dlib. Для оптимального встановлення програмного забезпечення повинні бути виповнені наступні вимоги. ОС: Windows 10. Процесор: AMD A8-7600 Radeon R7, 10 Compute Cores 4C+6G 3.10 GHz. Місце на диску: не менше 10 ГБ. RAM: 8 ГБ.

Таким чином, у розділі три описано розробку програмного продукту реалізації алгоритму детектування Deepfake. Для написання програмного продукту було обрано програмне середовище PyCharm 2022.1.1, віртуальне середовище Anaconda, встановлені пакети TensorFlow, Keras, Dlib за вже описані в розділі переваги. Програмний продукт є універсальним для будь-яких вхідних даних, надійним та працездатним у критичних ситуаціях. Розроблений інтерфейс є простим, зрозумілим та зручним для користувача.

Також в даному розділі приведено приклади роботи розробленої програми, коли відео чи зображення містили та не містили діпфейк. Також описано детальну інструкцію щодо користування розробленою програмою та вимоги програмного забезпечення для його швидкої та надійної роботи.

В ході виконання роботи в третьому розділі було вирішено четвертий пункт поставлених задач, тобто було реалізовано описаний у другому розділі алгоритм детектування діпфейків .

4 ОХОРОНА ПРАЦІ

Охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

Метою охорони праці є забезпечення безпечних , нешкідливих і сприятливих умов праці. Стіл з розміщеним на ньому персональним комп’ютером і необхідною периферією (монітор, клавіатура, миша і т.д.) є робочим місцем, яке дуже часто зустрічається у сучасній промисловості. Таке робоче місце має багато небезпечних та шкідливих факторів виробництва.

В даному розділі охорони праці були виявлені основні небезпечні і шкідливі виробничі фактори при роботі для заданого робочого місця. Згідно ГОСТ 12.0.003-74 [19], небезпечними та шкідливими факторами виробництва для роботи з персональним комп’ютером є:

- відсутність або нестача природного світла;
- недостатня освітленість робочої зони;
- нерівномірний розподіл яскравості в полі зору;
- підвищений рівень шуму на робочому місці;
- підвищена чи знижена температура повітря робочої зони;
- підвищена чи знижена вологість повітря;
- тривале статичне напруження;
- підвищене значення напруги в електричному ланцюзі, замикання якого може статися через тіло людини;

При роботі за монітором в неоптимальних умовах можливо завдати шкоди органам зору. Виконання зорової роботи за несприятливих умов освітлення призводить до зниження зорової працездатності. Зорова втома прискорює розвиток загальної втоми в організмі працівника і значною мірою відображається на якісних і кількісних виробничих показниках.

Найважливішими факторами, що зумовлюють зниження зорової працездатності, є: недостатні рівні освітленості, нерівномірність розподілу яскравості на робочому місці та в приміщенні загалом, наявність у полі зору сліпучої яскравості.[20]

Освітлення дає сприятливий психофізіологічний ефект, впливає на працездатність людини і на безпеку праці. Раціональне виробниче освітлення є показником естетики виробництва й високого рівня культури праці та фактором, що значною мірою зумовлює безпеку праці та створює у працівників певний психологічний тонус, попереджує зорову і загальну втому, сприяє високопродуктивній праці. Низький рівень освітленості, засліплююча дія джерел світла, пульсація світлового потоку, відбиті полиски від полірованих блискучих предметів, нерівномірне освітлення робочої зони можуть порушити правильне сприйняття навколишніх предметів і призвести до травматизму.

Однак, слід відзначити, що збільшення рівня освітленості має свою оптимальну межу, поза якою вона починає чинити засліплювальну дію і несприятливо впливати на зорове сприйняття. Властивість поверхонь, що світяться і мають високу яскравість, порушувати зорове сприйняття називають блискучістю, а психофізіологічні зміни, що відбуваються при цьому, і суб’єктивні відчуття — засліпленням. Блискучість може викликати несприятливі зміни не тільки у зоровому аналізаторі, а й у центральній нервовій системі (гальмування, зниження лабільності, працездатності, активності тощо). При її впливі на орган зору працівника насамперед порушується контрастна чутливість ока. Що менший контраст об’єкта розпізнавання з фоном і його кутові розміри, то сильніше виражена несприятлива дія засліплення.

У промисловості застосовуються три види освітлення: природне, штучне комбіноване [21].

Природне освітлення зумовлюють прямі сонячні промені й дифузне світло небосхилу. Природнє освітлення поділяється на: бокове (одно – або

двостороннє), що здійснюється через світлові отвори (вікна) в зовнішніх стінах; верхнє – через ліхтарі та отвори в дахах і перекриттях; комбіноване – поєднання верхнього та бокового освітлення.

Штучне освітлення може бути загальним та комбінованим. Загальним називають освітлення, при якому світильники розміщуються у верхній зоні приміщення (не нижче 2,5 м над підлогою) рівномірно (загальне рівномірне освітлення) або з урахуванням розташування робочих місць (загальне локалізоване освітлення).

Комбіноване освітлення складається із загального та місцевого. Його доцільно застосовувати при роботах з високої точності, а також, якщо необхідно створити певний або змінний в процесі роботи напрямок світла. Для місцевого освітлення робочих місць слід використовувати світильники з непросвічуючими відбивачами. Світильники повинні розташовуватися так, щоб їх елементи, які світяться, не влучали в поле зору працюючих на освітленому робочому місці і на інших робочих місцях. Застосування лише місцевого освітлення не допускається з огляду на небезпеку виробничого травматизму та професійних захворювань.

За будівельними нормами і правилами ДБН В.2.5-28-2006 необхідно, щоб усі виробничі, підсобні, складські та допоміжні приміщення були забезпечені денним світлом (для приміщень з постійним перебуванням людей). Для офісного приміщення ідеальним варіантом буде суміщене джерело освітлення, при якому недостатнє за нормами природне освітлення доповнюється штучним освітленням загального типу, регламентоване ДСанПін 2.2.4.171-10. Такий підхід зумовлений потребою напруження зору при роботі з дисплеєм. Це надає можливість регулювати рівень освітленості протягом дня в різні пори року. Таким чином, легко можна досягнути нормованого рівню освітленості на робочому столі – 300-500 лк.

Тривале неправильне положення призводить до напруження м'язів шиї, голови, рук і плечей, остеохондрозу. Тривале неправильне положення ще призводить до застою крові в тазових органах і, як наслідок, до простатиту і

геморою. Малорухливий спосіб життя також призводить до ожиріння. Остеохондроз виникає при порушенні міжхребцевих дисків, яке призводить до випинання в якусь сторону (грижі міжхребцевого диска).

Необхідно дотримуватись наступних правил [22]:

- Сидіти потрібно прямо, держати спину рівно.
- Монітор потрібно розмістити на відстані 70 см.
- Не закидати ногу на ногу, стопи мають твердо стояти на підлозі.
- Коліна під столом повинні бути зігнуті під кутом 90 градусів, а відстань до крісла має рівнятися розміру кулака.
- Стіл має бути на 2-3 см вище рівня ліктів, плечі не напружені, руки повинні бути зігнуті під кутом 90 градусів.
- Не варто часто використовувати підлокітники, якщо вони розташовані не під правильним кутом.
- Час від часу необхідно робити перерви, щоб трошки розім'ятися. Можна сходити на невелику прогулянку до сусіднього офісу, або зробити пару вправ на розтяжку.

Робота за комп’ютером, передбачає багато годин використання дисплею на день, тому в нагоду стають краплі для очей. Вони допоможуть краще змочити очне яблуко і принаймні частково запобігти від сухості та втоми очей.

Шум. Шум з фізіологічної точки зору – це шкідливий дратівливий чинник, що впливає на органи слуху і весь організм людини і створює значне навантаження на нервову систему людини а також погіршує працездатність людини [23].

Відволікаючий шум можна розділити на три групи:

– Шум, створюваний самими співробітниками, що включає розмови, кроки, а також звуки, створювані пересуванням меблів і закриваються дверима.

– Звуки, що йдуть від технічних установок будівлі. До них відносяться системи опалення, гідравліки, вентиляції та кондиціонування, а також шум, видаваний ліфтами.

– Шум від офісної техніки, без якої неможлива організація праці в офісі. Цю групу складають комп'ютери, принтери, сканери, сервери, шредери, телефони та інші пристрої.

В ДСН 3.3.6.037–99 вказано, що гранично допустимий рівень звуку в офісі, який забезпечує відсутність ризику набуття вад слуху і майже не впливає на працездатність та стан здоров'я працівників, становить 50 дБА.

Зовнішні шуми знижують шляхом розміщення на стінах звукопоглинаючого покриття. До засобів звукоізоляції належать акустичні екрани, мінеральна вата, акустичний поролон, штучні поглиначі. Також гарним способом зниження фонового шуму є індивідуальна гарнітура, яка щільно прилягає до голови, тим самим перешкоджає потраплянню звукових хвиль до вуха.

Санітарно-гігієнічні вимоги до умов праці в офісних приміщеннях. Площу приміщень, в яких розташовують персональні комп'ютери, визначають відповідно до нормативних документів. Виділення достатнього особистого простору для роботи дуже важливо і для нормального психічного стану людини. Відповідно до ДСанПін 2.2.4.171-10 [24] з розрахунку на одне робоче місце, обладнане ПК, встановлено такі норми:

- Площа на одне робоче місце має становити не менше ніж 6,0 кв. м.
- Об'єм на одне робоче місце має становити не менше ніж 20,0 куб. м.
- Відстані між бічними поверхнями ПК не менше ніж 1,2 м.
- Відстані від тильної поверхні одного ПК та екрана від іншого – 2,5 м.
- Висота робочої поверхні робочого столу з ВДТ має регулюватися в межах 0,68 – 0,8 м.
- Робочі місця повинні бути розташовані на відстані не менше ніж 1 м від стіни з вікном.

Мікроклімат. Мікроклімат виробничого середовища суттєво впливає на стан організму працівника, його працездатність протягом робочого дня, зміни. Показники температури, відносної вологості, швидкості руху повітря, теплового випромінювання нагрітих поверхонь характеризують клімат внутрішнього середовища виробничого приміщення. В процесі трудової діяльності людина перебуває у тепловій взаємодії з виробничим середовищем.

Для робочої зони виробничих приміщень встановлюються оптимальні та допустимі мікрокліматичні умови з урахуванням важкості виконуваної роботи та періоду року згідно з ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень».

Оптимальні – це комплекс мікрокліматичних чинників, які в умовах тривалої та систематичної дії на людину створюють комфортні теплові відчуття та збереження нормального теплового стану організму без напруження механізмів терморегуляції, а допустимі можуть викликати дискомфортні відчуття та зміни теплового стану організму, однак вони швидко минають і нормалізуються за рахунок напруження механізмів терморегуляції в межах фізіологічних пристосувальних можливостей [25].

Робота в офісі передбачає категорію робіт «Легка-1а». В таблиці 1 наведені оптимальні значення мікроклімату в теплий і холодний період року.

Таблиця 4.1 – Оптимальні умови мікроклімату для категорії роботи «Легка-1а»

Період року	Температура повітря, градуси	Відносна вологість, %	Швидкість руху повітря, м/с
Холодний	22 - 24	40 - 60	< 0,1
Теплий	23 - 25	40 - 60	< 0,1

Для підтримки температури в офісному приміщені використовуються кондиціонери, вентилятори і обігрівачі, все залежить від пори року.

Оптимальні значення вологості повітря досягаються розміщенням спеціальних зволожувачів повітря.

Електробезпека [26]. Електробезпека це система організаційних і технічних заходів, засобів та способів, які забезпечують захист від шкідливої і небезпечної дії електричного струму, електричної дуги, електромагнітного поля та статичної електрики.

Великий обсяг електрообладнання в типовому офісі може наражати працівників на серйозну небезпеку електричного струму, включаючи удари, опіки та пожежу. Для забезпечення електробезпеки необхідно звернути увагу на наступні заходи.

Облаштування електромережі, зокрема: правильний розподіл навантаження на всі приміщення офісу, правильний розподіл електромережі за призначенням (наприклад: освітлення – це одна група, робоча зона – інша), якість самих комплектуючих електромережі (розетки, вимикачі, лампи, світильники), потенціал для збільшення навантаження (на випадок створення додаткових робочих місць чи розширення компанії), використання офісного обладнання, в якому електроенергія застосовується за призначенням згідно з технічними рекомендаціями виробника.

Виважений підхід до питання використання стаціонарних або мобільних електрогенераторів для зменшення енергозалежності.

Проведення інструктажів з охорони праці з питань електробезпеки.

Пожежна небезпека [27] – можливість виникнення та (або) розвитку пожежі в будь-якій речовині, процесі, стані. Слід зазначити, що пожеж безпечних не буває, якщо вони і не створюють прямої загрози життю та здоров'ю людини, то завдають чи призводять до значних матеріальних втрат. Коли людина перебуває в зоні впливу пожежі, то вона може потрапити під дію наступних небезпечних та шкідливих факторів.

Метою пожежної безпеки об’єкта є попередження виникнення пожежі на визначеному чинними нормативами рівні, а у випадку виникнення пожежі

– обмеження її розповсюдження, своєчасне виявлення, гасіння пожежі, захист людей і матеріальних цінностей.

Для забезпечення безпечної роботи співробітників в офісі, приміщення має відповідати всім вимогам пожежної безпеки. Клас передбачуваної пожежі - “Е”, оскільки в приміщеннях встановлено системи, підключені до джерела електроенергії. Для співробітників дуже важливо виконання елементарних правил пожежної безпеки під час перебування на робочому місці (в офісі). Адже безвідповідальне ставлення до таких, здавалося б, дрібниць, як кинутий недопалок чи залишений без нагляду електрообігрівач, може спричинити пожежу.

Часто займання відбувається через неправильне зберігання в приміщенні легкозаймистих речовин, спалах електропроводки через перевантаження електромережі, неакуратне поводження з вогнем у місцях приготування їжі.

Рекомендується.

Меблі та обладнання необхідно розміщувати таким чином, щоб забезпечувався вільний евакуаційний прохід до дверей виходу з приміщення (завширшки не менше 1 м).

Евакуаційні шляхи та виходи необхідно постійно утримувати вільними, нічим не захаращувати.

Електромережі, електроприлади і апаратуру експлуатувати тільки у справному стані з урахуванням вказівок та рекомендацій підприємств-виготовлювачів.

У разі виявлення пошкоджень електромереж, вимикачів, розеток та інших електровиробів слід негайно вимкнути їх та вжити необхідних заходів щодо приведення в пожежобезпечний стан.

Документи, папір та інші горючі матеріали слід зберігати на відстані не менше 1 м від електрощитів; 0,5 м від електросвітильників; 0,6 м від сповіщувачів автоматичної пожежної сигналізації та 0,15 м від приладів центрального водяного опалення. Засоби протипожежного захисту слід утримувати у справному стані. Усі працівники повинні вміти користуватись

наявними вогнегасниками, іншими первинними засобами пожежогасіння, знати місце їх знаходження.

Відстань від найбільш віддаленого місця приміщення до місця розташування вогнегасника не повинна перевищувати 20 м.

ВИСНОВКИ

В даній кваліфікаційній роботі було проведено огляд сучасного стану проблеми виявлення зображень та відео, фальсифікованих за допомогою технології DeepFake.

В ході дослідження було обґрунтовано вибір алгоритму та моделі нейронної мережі для виявлення DF та визначено оцінку його ефективності.

Програмно реалізовано алгоритм виявлення DeepFake як фальсифікації цифрового зображення чи відео.

Розроблений програмний продукт є простим у використанні.

Усі задачі, поставлені в роботі, вирішено. Мету досягнуто.

Предметом подальшого розвитку даної роботі є вдосконалення нейромережі за рахунок збільшення набору даних для навчання.

ПЕРЕЛІК ПОСИЛАНЬ

1. Зоріло В.В., Тимофєєв Є.В. Виявлення фото підробок, виконаних засобами нейронних мереж. Інформатика та математичні методи в моделюванні. Том 12, № 1-2. 2022. С.10-19.

2. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. 2014. Generative Adversarial Networks.

3. MNIST URL: https://paperswithcode.com/dataset/mnist.

4. Branislav Holländer (2020) Autoencoders: Overview of Research and Applications.

5. Li, Y., Chang, M.-C. and Lyu, S. (2018) In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking. 2018 IEEE International Workshop on Information Forensics and Security (WIFS), Hong Kong, 11-13 December 2018, 1-7.

6. Ciftci, U.A., Demir, I. and Yin, L. (2020) How Do the Hearts of Deep Fakes Beat? Deep Fake Source Detection via Interpreting Residuals with Biological Signals. 2020 IEEE International Joint Conference on Biometrics (IJCB), Houston, 28 September-1 October 2020, 1-10.

7. de Lima, O., Franklin, S., Basu, S., Karwoski, B. and George, A. (2020) Deepfake Detection Using Spatiotemporal Convolutional Networks.

8. Zhu, J.-Y., Park, T., Isola, P. and Efros, A.A. (2017) Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. Proceedings of the IEEE International Conference on Computer Vision, Venice, 22-29 October 2017, 2223-2232.

9. Bansal, A., Ma, S., Ramanan, D. and Sheikh, Y. (2018) Recycle-GAN: Unsupervised Video Retargeting. Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, 23-28 August 2018, 119-135.

10. Sabir, E., Cheng, J., Jaiswal, A., AbdAlmageed, W., Masi, I. and Natarajan, P. (2019) Recurrent Convolutional Strategies for Face Manipulation

Detection in Videos. CVPR Workshops.

11. Afchar, Darius, et al. Mesonet: a compact facial video forgery detection network.

12. Djork-Arné Clevert, Thomas Unterthiner, & Sepp Hochreiter. (2015). Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs).

13. Andrew L. Maas. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models.

14. Реалізація та порівняння оптимізаторів моделей у глибокому навчанні URL: https://habr.com/ru/company/skillfactory/blog/525214/

15. R (мова програмування) URL: https://uk.wikipedia.org/wiki/R_(мова_програмування).

16. NumPy documentation URL: https://numpy.org/doc/stable/

17. SciPy documentation URL: https://docs.scipy.org/doc/scipy/

18. Astropy Documentation URL: https://docs.astropy.org/en/stable/

19. Tkinter – Python interface to Tcl/Tk URL: https://docs.python.org/3/library/tkinter.html#the-packer

20. ГОСТ 12.0.003-74

21. Освітлення на робочому місці URL: https://oppb.com.ua/news/pogane-osvitlennya-na-robochomu-misci-mozhe-pryzvesty-do-nevrozu-0

22. Освітлення виробничих приміщень URL:

https://oppb.com.ua/news/yak-vplyvaye-na-robotu-osvitlennya-vyrobnychyh-prymishchen-0

23. Як правильно сидіти за комп’ютером URL:

https://jysk.ua/blog/yak-pravilno-siditi-za-robochim-stolom

24. Березуцький В.В., Васьковец Л.А., Горбенко В.В., В.Ф. Райко В.Ф., Янчик О.Г. Основи професійної безпеки та здоров’я людини. Харків: НТУ “ХПІ”, 2018. 553с.

25. Наказ про затвердження Державних санітарних норм та правил "Гігієнічні вимоги до води питної, призначеної для споживання людиною"

URL:

https://zakon.rada.gov.ua/laws/show/z0452-10#Text

26. Мікроклімат виробничих приміщень та його вплив на працездатність URL:

https://oppb.com.ua/news/mikroklimat-vyrobnychyh-prymishchen-ta-yogo-vplyv-na-pracezdatnist

27. Про заходи з електробезпеки для працівників офісу URL:

https://oppb.com.ua/articles/pro-zahody-z-elektrobezpeky-dlya-pracivnykiv-ofisu

28. Заходи пожежної безпеки у службових приміщеннях URL:

https://city-adm.lviv.ua/lmr/socialni-iniciativi/2097-zakhody-pozhezhnoi-bezpeky-u-sluzhbovykh-prymishchenniakh-ofisakh

Додаток А. Лістинг програмного продукту

```
    # Файл для навчання моделі НМ
    from tensorflow.keras.layers import Input, Dense, Flatten,
Conv2D, MaxPooling2D, BatchNormalization, Dropout, LeakyReLU
    from tensorflow.keras.preprocessing.image import
ImageDataGenerator
    from tensorflow.keras.callbacks import ModelCheckpoint
    from tensorflow.keras.optimizers import Adam
    from tensorflow.keras.models import Model
    import matplotlib.pyplot as plt

    image_dimensions = {'height':256, 'width':256,
'channels':3}
    train= ImageDataGenerator(rescale=1/255)
    validation= ImageDataGenerator(rescale=1/255)
    train_dataset =train.flow_from_directory('F:/train_set/',
                                             target_size=(256,
256),
                                             batch_size=1,

class_mode='binary'
                                             )
    validation_dataset                                       =
train.flow_from_directory('F:/validation/',
                                             target_size=(256,
256),
                                             batch_size=1,

class_mode='binary'
                                             )
    model_checkpoint= ModelCheckpoint(f"My_final_Model.h5",
                                      save_best_only=True,
                                      monitor='val_loss',
                                      verbose=1
```

```
                                        )
    class Classifier:
        def __init__():
            self.model = 0
        def predict(self, x):
            return self.model.predict(x)
        def load(self, path):
            self.model.load_weights(path)
    class MyModel(Classifier):
        def __init__(self, learning_rate=0.001):
            self.model = self.init_model()
            optimizer = Adam(learning_rate=learning_rate)
            self.model.compile(optimizer=optimizer,
                               loss='mean_squared_error',
                               metrics=['acc'])
            self.model_fit = self.model.fit(train_dataset,
                                   epochs=30,
                                   batch_size=32,

validation_data=validation_dataset,
                                   callbacks=[model_checkpoint])
            plt.figure(1)
            plt.subplot(211)
            plt.plot(self.model_fit.history['acc'])
            plt.plot(self.model_fit.history['val_acc'])
            plt.title('Model accuracy')
            plt.ylabel('Accuracy')
            plt.xlabel('Epoch')
            plt.legend(['Train', 'Val'], loc='lower right')
            plt.subplot(212)
            plt.plot(self.model_fit.history['loss'])
            plt.plot(self.model_fit.history['val_loss'])
            plt.title('Model loss')
            plt.ylabel('Loss')
            plt.xlabel('Epoch')
```

```
            plt.legend(['Train', 'Val'], loc='upper right')
            plt.tight_layout()
            plt.show()
        def init_model(self):
            x = Input(shape=(image_dimensions['height'],
                             image_dimensions['width'],
                             image_dimensions['channels']))
            x1 = Conv2D(8, (3, 3), padding='same',
activation='relu')(x)
            x1 = BatchNormalization()(x1)
            x1 = MaxPooling2D(pool_size=(2, 2),
padding='same')(x1)
            x2 = Conv2D(8, (5, 5), padding='same',
activation='relu')(x1)
            x2 = BatchNormalization()(x2)
            x2 = MaxPooling2D(pool_size=(2, 2),
padding='same')(x2)
            x3 = Conv2D(16, (5, 5), padding='same',
activation='relu')(x2)
            x3 = BatchNormalization()(x3)
            x3 = MaxPooling2D(pool_size=(2, 2),
padding='same')(x3)
            x4 = Conv2D(16, (5, 5), padding='same',
activation='relu')(x3)
            x4 = BatchNormalization()(x4)
            x4 = MaxPooling2D(pool_size=(4, 4),
padding='same')(x4)
            y = Flatten()(x4)
            y = Dropout(0.5)(y)
            y = Dense(16)(y)
            y = LeakyReLU(alpha=0.1)(y)
            y = Dropout(0.48)(y)
            y = Dense(1, activation='sigmoid')(y)
            return Model(inputs=x, outputs=y)
    model = MyModel()
```

```
    # Файл для вилучення кадрів з відео
    import os
    import cv2
    def extract(path):
        if os.path.exists('data'):
            for root, dirs, files in os.walk('data',
topdown=False):
                for name in files:
                    os.remove(os.path.join(root, name))
                for name in dirs:
                    os.rmdir(os.path.join(root, name))
        if os.path.exists('images'):
            for root, dirs, files in os.walk('images',
topdown=False):
                for name in files:
                    os.remove(os.path.join(root, name))
                for name in dirs:
                    os.rmdir(os.path.join(root, name))
        cam = cv2.VideoCapture(path)
        try:
            if not os.path.exists('data'):
                os.makedirs('data')
        except OSError:
            print('Error: Creating directory of data')
        currentframe = 0
        while (True):
            ret, frame = cam.read()
            if ret:
                name = './data/frame' + str(currentframe) +
'.jpg'
                print('Creating...' + name)
                cv2.imwrite(name, frame)
                currentframe += 1
            else:
```

```
                break
        cam.release()
        cv2.destroyAllWindows()

    # Файл для класифікації унікальних облич
    import face_recognition
    from PIL import Image
    import os
    import glob
    def face_classifier():
        #Создание массива для класификации лиц
        images_enc=[]
        print("Создание массива для класификации лиц")
        for i in range(len(glob.glob('data/*'))):
            img_path = f"data/frame{i}.jpg"
            img1= face_recognition.load_image_file(img_path)
            for j in range
(len(face_recognition.face_encodings((img1)))):
                print(f"i={i} ,j={j}")
                img1_encodings =
face_recognition.face_encodings(img1)[j]
                s=0
                if len(images_enc)>0:
                    for enc in images_enc:
                        result=
face_recognition.compare_faces([enc],img1_encodings)
                        if result==[True]:
                            s+=1
                    if s==0:
                        images_enc.append(img1_encodings)
                else: images_enc.append(img1_encodings)
        print (len(images_enc))

        # Создание папки для каждого уникального лица
        for e in range (len(images_enc)):
```

```
            try:
                if not
os.path.exists(f"images/meson_P/{e}_face"):
                    os.makedirs(f"images/meson_P/{e}_face")
            except OSError:
                print('Error: Creating directory of data')
        # Перемещение лиц в папки
        print("Перемещение лиц в папки")
        for q in range(len(glob.glob('data/*'))):
            img_path = f"data/frame{q}.jpg"
            img1=face_recognition.load_image_file(img_path)
            for j in range
(len(face_recognition.face_encodings((img1)))):
                for i in range(len(images_enc)):
                    img1_encodings =
face_recognition.face_encodings(img1)[j]
                    result =
face_recognition.compare_faces([images_enc[i]], img1_encodings)
                    print(f"result={result}")
                    if result == [True]:
                        top, right, bottom, left =
face_recognition.face_locations(img1)[j]
                        print(f"{j}=j,{i}=i")
                        face_img = img1[top:bottom, left:right]
                        pil_img = Image.fromarray(face_img)

pil_img.save(f"images/meson_P/{i}_face/{j}face_{q}img.jpg")
        return len(images_enc)

    # Файл для вилучення облич з цифрового зображення
    import face_recognition
    import os
    from PIL import Image
    def face_image(path):
        img_path = path
```

```
        count = 0
        faces = face_recognition.load_image_file(img_path)
        face_location = face_recognition.face_locations(faces)
        for face_location in face_location:
            top, right, bottom, left = face_location
            print(face_location)
            face_img = faces[top:bottom, left:right]
            pil_img = Image.fromarray(face_img)
            try:
                if not os.path.exists(f'images'):
                    os.makedirs(f'images')
                if not os.path.exists(f'images/meson_P'):
                    os.makedirs(f'images/meson_P')
            except OSError:
                print('Error: Creating directory of data')
            pil_img.save(f"images/{count}_rface_img.jpg")
            count += 1
        print(f"Found {count} face(s)in photo")

    # Головний файл з інтерфейсом
    import keras.models
    from      tensorflow.keras.preprocessing.image      import
ImageDataGenerator
    from video_faces_analizer import video_faces_analizer
    from face_from_image import face_image
    from extraction import extract
    import warnings
    from tkinter import *
    from tkinter import filedialog
    import tkinter.font as tkFont
    import os,shutil,glob,cv2
    path=""
    dfpath='C:/Users/timof/OneDrive/Рабочий
стол/DEEPF/test_images/df'
    def open_file():
```

```
    global path
    for file in glob.glob(dfpath + "/*"):
        os.remove(file)
    for file in glob.glob("images" + "/*.jpg"):
        os.remove(file)
    path = filedialog.askopenfilename(initialdir=f"images")
    lrestext['text']="Prediction will be here"
    if path.endswith(".mp4"):
        cap = cv2.VideoCapture(path)
        if (cap.isOpened() == False):
            print("Error opening video  file")
        while (cap.isOpened()):
            ret, frame = cap.read()
            if ret == True:
                cv2.imshow('Video', frame)
                if cv2.waitKey(250) & 0xFF == ord('q'):
                    break
            else:break
        cap.release()
      cv2.destroyAllWindows()
    else:
        if (path.endswith("images/*") == False):
            shutil.copy(path, "images")
            str = os.path.basename(path)
        newpath = f"images/{str}"
        image = cv2.imread(newpath)
        cv2.imshow('Image', image)
        cv2.waitKey(0)
        cv2.destroyAllWindows()
def check():
    global path, dfpath
    for file in glob.glob(dfpath + "/*"):
        os.remove(file)
    if path.endswith(".mp4"):
        extract(path)
```

```
            mass = video_faces_analizer()
            opr=False
            for im in mass:
                if im =="deepfake":
                    opr= True
            if opr ==True:
                lrestext['text']= "This video contain deepfake"
            else:    lrestext['text']=  "This    video  doesn`t
contain deepfake"
        else:
            face_image(path)
            shutil.copy(path, dfpath)
            with warnings.catch_warnings():
                warnings.simplefilter("ignore")
                model                                          =
keras.models.load_model('My_new_Model.h5')
            dataGenerator   =   ImageDataGenerator(rescale=1.   /
255)
            generator = dataGenerator.flow_from_directory(
                'C:/Users/timof/OneDrive/Рабочий
стол/DEEPF/test_images/',
                target_size=(256, 256),
                batch_size=1,
                class_mode='binary')
            X, y = generator.next()
            if (100 * model.predict(X)[0][0])>=50:
                ltext = f"This image doesn`t contain deepfake"
            else: ltext = f"This image  contain deepfake"
            lrestext['text']= ltext
    def play():
        global path
        if path.endswith(".mp4"):
            cap = cv2.VideoCapture(path)
            if (cap.isOpened() == False):
                print("Error opening video  file")
```

```
            while (cap.isOpened()):
                ret, frame = cap.read()
                if ret == True:
                    cv2.imshow('Video', frame)
                    if cv2.waitKey(250) & 0xFF == ord('q'):
                        break
                else:break
            cap.release()
            cv2.destroyAllWindows()
        else:
            if (path.endswith("images/*")== False):
                shutil.copy(path,"images")
                str=os.path.basename(path)
            newpath=f"images/{str}"
            image = cv2.imread(newpath)
            cv2.imshow('Image', image)
            cv2.waitKey(0)
            cv2.destroyAllWindows()
    root= Tk()
    root.title("DeepFakeIdentifier")
    root.geometry("565x200")
    root.iconbitmap("C:/Users/timof/OneDrive/Рабочий
стол/favicon.ico")
    root['background']='#856ff8'
    b1=Button(root,text="1.Open    mediafile",height=2,width=20,
font="16",background="#859ff8",command=open_file)
    b1.place(x=1,y=41)
    b2=                              Button(root,text="2.Check",
height=2,width=20,font="16",background="#859ff8",command=check)
    b2.place(x=190,y=41)
    b3=            Button(root,text="Show            mediafile",
height=2,width=20,font="16",background="#859ff8",command=play)
    b3.place(x=380,y=41)
    bfont = tkFont.Font(family="Arial", size=14, weight="bold")
    lmain=Label(root, text="DeepFakeIdentifier")
```

```
    lmain.configure(font=bfont)
    lmain.place(x=200,y=0)
    l1=Label(root,text="Result:",background="#856ff8",width=20,
height=1)
    l1.configure(font=bfont)
    l1.place(x=165,y=100)
    lrestext      =Label(root,text="Prediction      will      be
here",width=30, height=1,font="14")
    lrestext.place(x=150,y=140)
    root.mainloop()
```