Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Галузь знань 12 – Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Метод та програмно-технічні засоби виявлення дипфейків»

КвРКІП. 170139.21.01.08 ПЗ

Виконав: студент 2 курсу, група КІ2м-21-1 Підпис Дмітрієв Б.В. Ініціали, прізвище

Керівник доктор техн. наук, професор Науковий ступінь, вчене звання Підпис Яцків В.В. Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КІІС, д.т.н., проф.
Т.О. Говорущенко
10 05 2023 р.

Хмельницький, 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень магістр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма освітньо-наукова програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Г.О.Говорущенко

"01" 09 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Дмітрієву Богдану Васильовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Метод та програмно-технічні засоби виявлення дипфейків

Керівник проекту (роботи) Яцків В.В. , д.т.н., професор

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 09.01.2023 р. № 1

2. Строк подання студентом проекту (роботи) на кафедру 01.05.2023 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз відомих методів та рішень виявлення дипфейків

Огляд фреймворків та засобів, які використалися у роботі

Розробка методу та програмно-технічного засобу виявлення дипфейків

Тестування та експериментальні дослідження

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів кваліфікаційної роботи магістра

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІІС		

7. Дата видачі завдання «_06_»_____09_____2022р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи магістра	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики КвРМ з керівником	05.09.2022	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	05.10.2022	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	05.11.2022	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	05.12.2022	виконано
5	Робота над науковою статтею	05.01.2023	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.02.2022	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	05.04.2023	виконано
8	Оформлення пояснювальної записки згідно вимог	15.04.2023	виконано
9	Попередній захист ДРМ		
10	Захист ДРМ на засіданні ЕК	18.04.2023	виконано
		До 10.05.2023	

Студент _____ Б.В. Дмітрієв
Підпис Ініціали, прізвище

Керівник роботи _____ В.В. Яцків
Підпис Ініціали, прізвище

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: Метод та програмно-технічні засоби виявлення дипфейків

Автор роботи: Дмітрієв Богдан Васильович

Керівник роботи: Яцків Василь Васильович

Пояснювальна записка: 71 с., 19 рис., 7 табл., 3 дод., 81 джерел.

НЕЙРОННА МЕРЕЖА, DEEPFAKE, GAN, МАШИННЕ НАВЧАННЯ, ЗГОРТКОВІ МЕРЕЖІ, ШТУЧНИЙ ІНТЕЛЕКТ

Об’єктом дослідження є процес виявлення дипфейків.

Предметом дослідження є методи, програмні та апаратні засоби виявлення дипфейків.

Метою кваліфікаційної роботи магістра є розробка методу з використанням готових програмно-технічних засобів для виявлення медіафайлів, які було змінено за допомогою технології дипфейків.

Для розв’язання поставлених задач використовувалися методи машинного навчання, методи побудови нейронних мереж та методи виявлення дипфейків.

Наукова новизна отриманих результатів:

− набув подальшого розвитку метод виявлення дипфейків, який базується на навчанні нейронної мережі з використанням набору даних, який містить як фальшиві, так і справжні відеодані при цьому забезпечує ефективне виявлення дипфейків та відкриваючи можливості для подальших досліджень;

− набула подальшого розвитку інформаційна технологія штучного інтелекту для тестування та оцінка запропонованого методу надаючи емпіричні докази його ефективності та надійності.

На основі проведених досліджень розроблена архітектура і компоненти програмного забезпечення, що дозволило реалізувати метод виявлення фейкових відео. Особливою частиною реалізації цієї архітектури є набір даних, тобто зображення, що проходять алгоритм детектування Deepfake. Обґрунтовано, що на точність результату впливають самі файли та їх кількість.

Практична значимість отриманих результатів полягає у тому, що вони можуть допомогти судовими лабораторіями та експертами-криміналістами при аналізі відео на наявність їх фальсифікації.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ШІ – штучний інтелект

НМ – нейронна мережа

НД – набір даних

ВСТУП

У сучасному світі високих технологій все більше людей використовують ШІ для створення інформації, яка може бути невірною або неправдивою. Одним із найпоширеніших типів фейкових новин є дипфейки – зображення, відео чи аудіо, які були змінені, щоб ввести глядача чи слухача в оману [1-3].

Тому для виявлення дипфейків потрібні ефективні методи та програмні засоби. Це питання стало особливо важливим, оскільки неправдиві дані можуть мати серйозні наслідки, такі як поширення оманливих новин або вплив на результати виборів. Вони можуть бути використані для дискредитації політичних опонентів, підриву довіри до органів влади та інших організацій, а також для поширення дезінформації та неправдивої інформації про здоров'я та медицину.

В останні роки поширення дипфейків досягло катастрофічних масштабів. Сайти соціальних мереж, зокрема Facebook і Twitter, уже розгортають функції, які дозволяють користувачам повідомляти про публікації, які містять невірну інформацію. Однак це залишається складним питанням, особливо для організацій та влади [4, 5]

Більше того, виявилося, що розпізнавання дипфейків – це складне завдання, оскільки вони можуть бути створені дуже професійно та вміло. Це означає, що необхідно розробити нові методи та технології для ефективного виявлення дипфейків, що створює виклик для вчених і розробників програмного забезпечення.

Існує глобальний попит на інструменти, які допомагають швидко та ефективно виявляти дипфейки. Це може допомогти виявити фейкові новини до того, як вони поширяться в масових ЗМІ. Навіть якщо фейкові новини вже поширилися, виявлення глибоких фейків може зменшити їхній вплив і полегшити поширення справжніх новин пізніше [6-8].

Таким чином, метою дослідження є розробка та тестування ефективного методу виявлення дипфейків, який може допомогти ідентифікувати фейкові фотографії та стримувати їх поширення. У цій роботі розглядаються різні методи

та засоби програмного забезпечення та техніки для виявлення дипфейків також експериментальне дослідження їх результативності. Ми дослідимо, як можна використовувати машинне навчання та інші методи для виявлення дипфейків у зображеннях, відео та аудіо записах. Крім того, ми проведемо порівняльний аналіз ефективності різних методів виявлення дипфейків.

Об'єктом дослідження є процес виявлення дипфейків.

Предметом дослідження є методи і програмне забезпечення, а також технічні засоби виявлення дипфейків.

Тому ця робота є вагомим внеском у вирішення проблеми дипфейків, які стали серйозною загрозою для довіри до інформації та ЗМІ загалом. Ми будемо досліджувати не тільки способи виявлення дипфейків, але й їхні наслідки для людей і суспільства, а також можливі застосування в інших сферах.

1 АНАЛІЗ ВІДОМИХ МЕТОДІВ ТА РІШЕНЬ ВИЯВЛЕННЯ ДИПФЕЙКІВ

1.1 Поняття діпфейків

Мережі постій Коли ми говоримо про ШІ, він часто згадує світ роботів або футуристичних технологій. Проте ШІ вже є частиною нашого повсякденного життя [1, 2].

ШІ (AI) – це передова галузь інформатики, присвячена автоматизації поведінки, яка зазвичай асоціюється з людським інтелектом. Як це буває з будь-якою складною наукою, багато понять виділяються з неї [3, 4]. Є багато видів навчання про які ми поговоримо трішки пізніше, але основне що ми маємо знати, що ця галузь призвела за до створення нових типів мереж, так як - Генераційні змагальні мережі (GAN). Одна з новітніх технологій, що пропонує великий потенціал у багатьох моментах використання, від створення ваших минулих зображень і посилення вашого голосу до надання різноманітних застосувань у медицині та інших галузях промисловості.

Генеруюча змагальна мережа (GAN) – це модель машинного навчання, у якій дві нейронні мережі конкурують одна з одною, щоб стати точнішими у своїх прогнозах. GAN зазвичай працюють без нагляду та використовують кооперативну гру з нульовою сумою для навчання. Дві нейронні мережі, які утворюють GAN, називаються генератором і дискримінатором. Генератором є згорткова НМ, а дискримінатором є деконволюційна НМ. Метою генератора є штучне створення вихідних даних, які можна легко прийняти за реальні. Мета дискримінатора полягає в тому, щоб визначити, які виходи, які він отримує, були створені штучно. По суті, GAN створюють власні навчальні дані. У міру того, як цикл зворотного зв’язку між конкуруючими мережами продовжується, генератор почне видавати високоякісний вихід, а дискримінатор стане краще позначати дані, які були створені штучно, опис цього в електронному ресурсі [6 – 8].

Першим кроком у створенні GAN є визначення бажаного кінцевого результату та збір початкового навчального набору даних на основі цих параметрів.

Потім ці дані рандомізуються та вводяться в генератор, доки він не досягне базової точності у створенні вихідних даних. Після цього згенеровані зображення завантажуються в дискримінатор разом із фактичними точками даних з вихідної концепції. Дискримінатор фільтрує інформацію та повертає ймовірність від 0 до 1, щоб представити автентичність кожного зображення (1 корелює зі справжнім, а 0 – із підробкою). Потім ці значення вручну перевіряються на успішність і повторюються, доки не буде досягнуто бажаного результату. За допомогою нього будуються найскладніші підроблені фотографії, відео і.т.д. ,такої технології, як діпфейк.

Deepfake, як і інші алгоритми глибокого навчання, покладаються на нейронні мережі, які, просто кажучи, є програмною конструкцією, яка намагається імітувати функціонування людського мозку. Deepfakes вимагає зразків вихідних даних, а також кодера та декодера. Універсальний кодер використовується для аналізу та порівняння ключових характеристик вихідних даних, якими можуть бути зображення, відео, текст або аудіофайл. Дані розбиваються на латентний простір нижчого виміру, а кодер навчається знаходити шаблони. Декодер — це навчений алгоритм, який використовує специфікації цілі, щоб потім порівняти та порівняти два зображення. У результаті алгоритм накладає ознаки джерела на зображення цілі, що призводить до підроблених даних [9].

Основною архітектурою, яка забезпечує високу точність і функціонування технології deepfake, є генеративна змагальна мережа (GAN), яка є частиною декодера. Як правило, кодер використовується для виділення прихованих рис обличь або цікавої області із зображень, тоді як декодер використовується для реконструкції обличчя. У процесі зміни обличчя між цільовим і вихідним зображеннями під час створення глибоких фейків знадобляться дві пари декодера та кодера, де кожен спочатку навчається на вихідному, а потім на цільовому зображенні [10]. Що робить GAN такими унікальними та точними, так це робота та спільна робота генератора та дискримінатора. Генератор створює нове зображення з прихованого представлення вихідних даних (рисунок 1.1). З іншого боку, дискримінатор намагається якомога точніше розрізнити щойно згенеровані та

оригінальні реальні дані та визначає, чи згенероване зображення чи ні. Оскільки обидві мережі виконують змагальне навчання для оптимізації своїх цілей на основі їх функції втрат, генератор і дискримінатор продовжують працювати разом, щоб постійно підвищувати свою точність. Застосовність є дуже потужною завдяки постійному покращенню продуктивності та векторній арифметиці в латентному просторі. Крім того, GAN можуть створювати нові набори даних із подібним розподілом і статистикою, як основний НД, який використовується для навчання алгоритму. Дискримінатор дізнається про розподіл даних, у результаті чого створюється модель, яка може виводити нові реалістичні зразки [11, 12].

Рисунок 1.1 – Високорівневий опис потокової архітектури GAN

Реальна вибірка даних є джерелом, тоді як вибірка синтетичних даних є цільовою. Вхідний шум поєднується з міткою в архітектурі GAN, а структура навчання GAN дозволяє використовувати функцію умовних втрат GAN у генерації синтетичних цільових зображень.

У цьому підрозділі можна розглянути детальніше різні види діпфейків, що можуть бути створені за допомогою різноманітних технологій та алгоритмів

машинного навчання. Діпфейки можуть мати різноманітні форми, включаючи фотографії, відео, аудіо та текст. Вони можуть бути створені з різних мотивів, таких як політична пропаганда, реклама, кібербулінг, розваги та злочинні дії, такі як шахрайство та шантаж. Наприклад, можна описати наступні типи [13]:

1. Діпфейки на зображеннях. Цей тип включає різні методи створення фальшивих зображень, такі як глибокі нейронні мережі, генеративно-змагальні мережі та інші. Для створення фальшивих зображень використовують технології машинного навчання, що стають все більш реалістичними. Ці фальшиві зображення можуть бути створені шляхом обробки фотографій або відео, що містять людей, тварин або об'єкти, що з'являються в реальному житті.

2. Діпфейки на відео, які є рухомим еквівалентом фотодіпфейків. Це означає, що машинне навчання використовується для створення фальшивих відео. Механізми на відео можуть бути створені шляхом зміни контенту відео, додавання або видалення об'єктів, заміни обличь людей, зміни заднього плану тощо. Такі типи можуть бути використані для створення фальшивих новин, відео-мемів або навіть фальшивих відео-спогадів.

3. Діпфейки на аудіо. Такі створені за допомогою глибоких нейронних мереж, що можуть розпізнавати голос та імітувати його. Одним з найбільш небезпечних видів є голосові діпфейки. Вони звучать, як реальний голос людини. Штучні звуки на аудіо можуть бути для створення фальшивих голосів, які можуть бути використані в обмані при телефонних розмовах, а також для фальшивих звукових записів.

4. Діпфейки на текстах. Це фейки текста, створені за допомогою алгоритмів машинного навчання, що можуть генерувати фальшиві текстові повідомлення або статті. Такі на текстах можуть бути використані для поширення фальшивих новин, обману або впливу на громадську думку.

5. Діпфейки з елементами віртуальної реальності - цей вид використовується для створення віртуального світу або реальності, яку неможливо відрізнити від реальної. Використовують технології для створення фальшивих

сцен, образів та реалістичних ігор. В таких діпфейках може бути створено цілі віртуальні світи, що повністю не існують у реальності.

Оскільки діпфейки можуть мати серйозні наслідки, такі як вплив на громадську думку, порушення приватності, збитки від кібершахрайства та багато іншого, важливо розробляти ефективні методи та програмно-технічні засоби для їх виявлення та боротьби з ними.

1.2 Огляд існуючих методів виявлення діпфейків

У цьому пункті ми розглянемо різні методи виявлення діпфейків, які були розроблені на сьогоднішній день. Вони можуть бути класифіковані в залежності від типу діпфейку, що вони виявляють, та від технології, яка використовується для виявлення. Одним з ключових завдань виявлення діпфейків є розробка ефективних та надійних методів, що дозволяють точно визначити, чи є зображення дійсним або фальшивим. Існує багато методів виявлення діпфейків, кожен з яких має свої переваги та недоліки.

1.2.1 Виявлення діпфейків на основі аналізу зображень

Один з найбільш поширених методів виявлення діпфейків на основі аналізу зображень полягає в знаходженні аномальностей в зображенні. Цей метод заснований на тому, що фотографії, які були змінені або створені за допомогою штучного інтелекту, мають різні аномалії, які можуть бути виявлені за допомогою аналізу зображення. Аналіз зображень є одним з найбільш ефективних способів виявлення діпфейків, оскільки діпфейки зазвичай засновані на використанні зображень або відео. Основною метою аналізу зображень є знайдення незвичайних артефактів, які можуть вказувати на те, що зображення було змінене або підроблене.

Один з основних методів аналізу зображень - це перевірка пікселів зображення на наявність аномалій. Наприклад, деякі фейки можуть бути створені

шляхом злиття двох зображень в одне, може бути помітна чітка грань між двома різними частинами зображення. Другий метод аналізу зображень полягає в аналізі особливих точок, таких як кути, які можуть вказувати на те, що зображення було підроблене (рисунок 1.2) [14].

Іншим ефективним методом є аналіз експозиції та освітлення на зображенні. Деякі діпфейки можуть мати різний рівень яскравості, контрастності або кольорової насиченості, що відрізняє їх від оригіналу ,полягає в аналізі особливостей зображення, таких як шум, зміна розміру. Ці параметри можуть бути виміряні та порівняні з відповідними параметрами оригінального зображення, що дозволяє виявляти змінені зображення. Аналіз цих параметрів може допомогти виявити зміни, що вказують на підробку.

Крім того, можна використовувати методи машинного навчання для виявлення діпфейків на основі аналізу зображень. Наприклад, можна тренувати моделі нейронних мереж на великій кількості оригінальних та підроблених зображень і використовувати їх для автоматичного виявлення діпфейків на нових зображеннях.

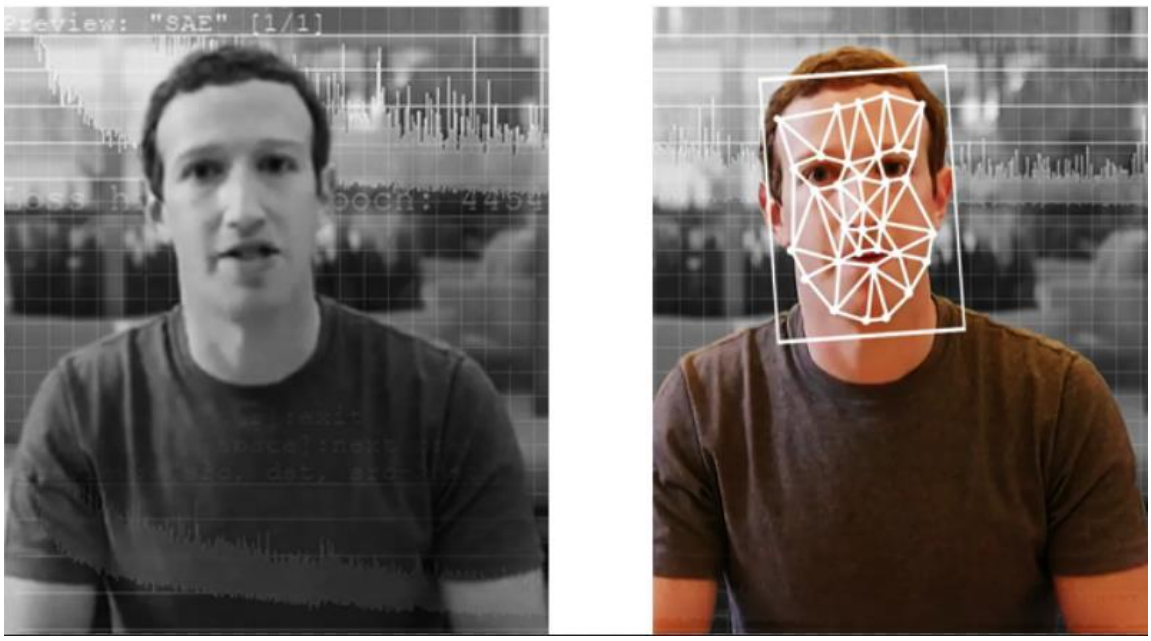


Рисунок 1.2 – Приклад реалізації методу аналізу зображення

Загалом, аналіз зображень є важливим інструментом для виявлення діпфейків, і цей метод може бути ефективним. Однак, важливо зазначити, що з точки зору технологій можуть бути створені настільки реалістично, що вони можуть бути дуже важкими до виявлення. Тому дослідження та розвиток нових методів виявлення діпфейків досі продовжується.

1.2.2 Виявлення діпфейків на основі аналізу відео

Виявлення діпфейків на основі аналізу відео - це процес виявлення фальсифікованого вмісту у відеофайлі, що було створено шляхом штучної модифікації вмісту з метою створення хибного сприйняття. Один з найпоширеніших методів виявлення діпфейків у відео полягає у застосуванні аналізу руху на відео. Цей метод вимірює швидкість, напрям та прискорення об'єктів у відео, що дозволяє виявляти аномалії в русі, що можуть бути показником фальсифікації. Інші методи виявлення діпфейків у відео включають аналіз спектрів кольору та яскравості відео, дослідження глибини зображення, аналіз різниці між реальним та фальсифікованим відео та використання машинного навчання та штучного інтелекту.

Інші методи, такі як аналіз відхилень частоти та аналіз показників температури, також можуть бути застосовані для виявлення діпфейків у відео. Однак, більшість методів виявлення у відео мають свої обмеження та не можуть забезпечити повного виявлення всіх видів у відео.

Також є більш старіші методи, такі як аналіз водяного знака (watermark). Цей метод використовує водяні знаки, щоб захистити авторські права на зображення та перешкодити його фальсифікації. Водяні знаки - це вбудована інформація в зображення, що неможливо помітити звичайним зором. Якщо водяний знак виявляється в неправильному місці або зображення не містить водяного знака взагалі, це може бути індикатором фальсифікації. Інший метод - аналіз стенографії [15].

Стенографія - це метод приховування інформації в зображеннях, текстових документах та інших типах файлів, не змінюючи зовнішнього вигляду файлу. Якщо зображення містить приховану інформацію, то це може свідчити про фальсифікацію.

1.2.3 Виявлення діпфейків на основі аналізу звуку

Глибоке аудіо – наступний рубіж для шахрайства з компрометацією компаній, і шахраї стають все більш звичним шляхом обману отримувати доступ до корпоративних коштів. Зі зростанням популярності та можливостей глибоких аудіофейків, розробка засобів захисту від глибоких фейків, які використовуються в підлих цілях, є більш важливою, ніж будь-коли. Мета реалізована за допомогою Python і техніки CNN. У модель подається НД зображення зразка аудіочастотного аналізу. Модель використовує візуальні представлення аудіокліпів, які потім подаються в resnet34 і навчаються для досягнення точної точності, щоб розрізнити справжній звук від підробленого. Останні прориви в глибокому навчанні та інших суміжних сферах.

Коли для створення синтетичного звуку використовується «клонований» голос, який, можливо, неможливо відрізнити від голосу реальної людини, це називається аудіо підробкою. Крім того, оскільки багато записів мовлення – це телефонні дзвінки низької якості (або записані в шумному середовищі), аудіо підробки може бути ще важче виявити. Що гірша якість звуку, то важче виявити сигнали про те, що голос несправжній. Нещодавно інтегровані ШІ розробки в більшості технік продемонстрували їх потужну здатність створювати реалістичні голоси. Фальшиві голоси не чутні для людського вуха і часто використовуються для створення реалістичних і природних фейків, які підкреслюють серйозні ризики для нашої спільноти. У Resemble і Descript є безкоштовні онлайн-демонстрації, які кожен може спробувати. Ви просто записуєте фрази, які з’являються на екрані, і модель вашого голосу створюється за лічені хвилини. Ви можете подякувати штучному інтелекту (ШІ) — зокрема, алгоритмам глибокого навчання — за те, що

ви можете зіставити записане аудіо з текстом і розшифрувати фонеми, з яких складається ваш голос [16]. Потім він наближає слова, використовуючи отримані лінгвістичні будівельні елементи .

Однією з найбільш ефективних технік виявлення діпфейків є аналіз звукових ознак. Звук може бути важливим джерелом інформації про те, чи є відео або аудіозапис реальним. Наприклад, дослідники можуть проаналізувати амплітуду звукових сигналів відео та порівняти їх зі звуком, який має бути присутнім у відповідній ситуації. Якщо звук не відповідає очікуваному звуку, це може свідчити про те, що відео є діпфейком.

Для досягнення методу було використано та впроваджено Python та глибоке навчання. Вхідними даними цієї роботи є аудіо- чи відеодані, а модель навчена розпізнавати аспекти, унікальні для генерації та виявлення голосу. Щоб відрізнити реальне від фальшивого, використовується техніка глибокого навчання. Частотні графіки аудіо кліпів включено в НД. Потім НД тренується за набором перевірки, який подається в для виведення точності, а не за допомогою аналізу часової області, оскільки частотний спектр людини більше порівнюється з його природним голосом, ніж з його виголошенням [17].

Ефективні та надійні детектори для синтезованих фальшивих голосів, з іншого боку, все ще знаходяться в зародковому стані та не можуть ефективно вирішити цю зростаючу проблему. У цьому методі потрібно провести експерименти з трьома наборами даних (включаючи комерційні продукти від Google, Baidu та інших), які містять англійську та китайську мови, щоб підтвердити високі показники виявлення Deep Sonar (середня точність 98,1 відсотка) і низький рівень помилок (приблизно 2% помилок) у виявленні фальшивих голосів. Крім того, значні результати тестування показують, що він стійкий до маніпуляційних атак (наприклад, перетворення голосу та додаткові шуми реального світу). Крім того, замість того, щоб піддаватися численним артефактам, представленим у синтезі, наше дослідження дає нове розуміння того, як застосовувати пропозиції точного виявлення [18].

Інший метод полягає у використанні різних статистичних методів для аналізу звуку, таких як мел-частотний кепстральний аналіз (MFCC). Цей метод використовується для порівняння характеристик звуку у фейках зі звуком, що спостерігається у відео, яке є справжнім.

Також існують методи, які використовують ШІ для виявлення змін у голосовому тоні та ритмі мовлення, що може свідчити про те, що відео було підроблено. Такі методи можуть бути особливо ефективними для виявлення недоліків з використанням синтезованого голосу. Методи аналізу звукових ознак можуть бути особливо ефективними, які використовуються для розповсюдження неправдивої інформації або дискредитації осіб. Однак, важливо пам'ятати, що жоден метод виявлення не є ідеальним і кращим результатом можна досягти, комбінуючи різні методи та техніки.

Одним із найпоширеніших методів аналізу звукових ознак є спектральний аналіз. Він базується на розкладанні звукового сигналу на окремі складові за допомогою перетворення Фур'є. Після цього можна визначити частотний склад звукового сигналу та порівняти його з відомими звуками. Інший метод - це аналіз формант. Форманти - це резонансні піки у звуковому сигналі, які виникають під час вимови різних голосних звуків [19].

Крім того, існують методи аналізу звуку на основі експертної оцінки, коли спеціалісти слухають звуковий запис та визначають, чи є він діпфейком чи ні. Однак цей метод є менш точним і об'єктивним, ніж методи аналізу звукових ознак, які базуються на математичних алгоритмах.

1.2.4 Виявлення діпфейків на основі аналізу тексту

Для виявлення текстових діпфейків використовуються різні методи аналізу тексту.

Існують методи аналізу тексту на основі мовленнєвих або письмових особливостей, які можуть вказувати на те, що текст був згенерований штучним інтелектом. Основними методами виявлення тексту є (таблиця 1.1).

Таблиця 1.1 – Основні методи виявлення тексту

Машинне навчання на основі прикладів	Цей метод полягає в тому, щоб створити модель машинного навчання на основі прикладів текстових діпфейків та оригінальних текстів. Модель будується
Аналіз стилістики тексту	Цей метод полягає в аналізі стилістики тексту для виявлення ознак, які можуть вказувати на його фальсифікацію. Наприклад, аналіз різноманітної лексики, синтаксичної структури, вживання пунктуації, повторення слів і фраз що може допомогти виявити текстовий діпфейк.
Аналіз мовних елементів	Цей метод використовується для аналізу мовних елементів, таких як інтонація, енергетика мовлення, ритм ,тощо. Він може виявити зміни в стилі, які можуть свідчити про фальсифікацію тексту.
Використання мовних моделей	Цей метод використовує моделі для виявлення діпфейків. Такі базуються на використанні відомостей про те, які слова можуть з'являтися в тексті та як вони пов'язані між собою. Ці моделі можуть бути використані для виявлення несподіваних комбінацій слів у тексті.

Всі ці методи можуть бути використані для аналізу тексту та виявлення діпфейків. При цьому важливо враховувати особливості конкретного відео та його контексту, щоб забезпечити якнайбільш точний та ефективний аналіз. Перелік з статті [20].

1.2.5 Методи аналізу біометричних ознак

Біометричні ознаки людини, такі як відбиток пальця, обличчя, голос тощо, можуть бути використані для виявлення діпфейків. Це пов'язано з тим, що

біометричні ознаки є унікальними для кожної людини, тому можуть слугувати як ідентифікатором особи на відміну від створених штучних даних.

Одним з методів аналізу біометричних ознак є аналіз обличчя. Цей метод полягає в порівнянні відео-або фотозображень людини з відомими зразками обличь для визначення автентичності. Для цього використовуються алгоритми комп'ютерного зору, які аналізують різні ознаки обличчя, такі як розташування очей, форма носа, розмір губ тощо. Ці ознаки відображаються в числовій формі і можуть бути порівняні з відомими зразками обличь для визначення автентичності [21-24].

Ще одним методом аналізу біометричних ознак є аналіз голосу. Цей метод полягає в порівнянні звукових записів голосу з відомими зразками голосу для визначення автентичності.

Для цього використовуються алгоритми обробки сигналів, які аналізують різні ознаки голосу, такі як частота голосу, інтонування тощо. Ці ознаки також відображаються в числовій формі і можуть бути порівняні з відомими зразками голосу для визначення автентичності.

Інші біометричні ознаки можуть також бути використані для виявлення діпфейків з використанням відповідних алгоритмів обробки даних.

Технологію біометрії обличчя з жвавістю можна використовувати для перевірки того, що хтось є справжньою особою, а не для атаки на презентацію, тобто визначити, коли демонструється відео чи фото. Але, як ми бачили, зараз технологія досягла такого рівня, коли зловмисники можуть вводити безпосередньо у відеопотік, минаючи деякі перевірки живучості, згідно в [25].

Хтось вважає, що для пом’якшення цього важливі рішення, які можуть перевірити найскладнішу частину присутності користувача в режимі реального часу.

Один вчений заявляє, що біометрія, особливо технологія розпізнавання обличчя, може допомогти вирішити проблему фальшивих кандидатів на роботу. Однак не так багато пристроїв постачаються з технологією розпізнавання обличчя.

1.3 Порівняльний аналіз методів виявлення діпфейків

У роботі проведено порівняльний аналіз методів виявлення діпфейків, що включає методи аналізу звукових ознак, методи аналізу тексту та методи аналізу біометричних ознак.

Аналіз показав, що кожен з методів має свої переваги та недоліки:

1) Методи аналізу звукових ознак:

Переваги: швидкі та ефективні виявлення діпфейків на основі аналізу акустичних характеристик;

Недоліки: можуть бути вразливі помилкам, зумовленим спотворенням звукової інформації;

2) Методи аналізу тесту:

Переваги: використовують машинне навчання та аналіз структури тексту. Вони можуть бути ефективними виявленням діпфейків, що створені штучним інтелектом;

Недоліки: вони можуть бути неефективними виявленням фейкам, що створені людиною;

3) Методи аналізу біометричних ознак:

Переваги: використовують технології розпізнавання обличчя та голосу. Вони можуть бути ефективними виявленням діпфейків, що використовуються для імітації обличчя або голосу;

Недоліки: вони можуть бути менш ефективними виявленням діпфейків, що використовуються для імітації інших біометричних ознак, таких як підпис або відбиток пальця;

Таким чином, для ефективного виявлення діпфейків може бути доцільно використовувати комбінацію різних методів аналізу, що дозволяє скористатись перевагами кожного методу та уникнути його недоліків [26].

1.4 Висновки

В першому розділі було дано визначення поняття штучного інтелекту, як він працює та що він утворює. Також проаналізовано будову мережі GAN, та нейронної мережі, адже це головне в побудові та виявлення фейкових відео.

Розкрито поняття діпфейк, як він створюється, та яку шкоду приносить. Проаналізовано основні методи виявлення таких фейків, як аналіз відео, зображень, звуку та тексту, також біометричних ознак. Була проведена порівняльна характеристика методів, що в свою чергу дало зрозуміти, як саме можна вдосконалити та реалізовувати власний метод.

2 ОГЛЯД ФРЕЙМВОРКІВ ТА ЗАСОБІВ, ЯКІ ВИКОРИСТАЛИСЯ У РОБОТІ

2.1 Огляд комерційних та відкритих програмно-технічних засобів

Огляд програмно-технічних засобів виявлення діпфейків можна розділити на дві категорії: комерційні та відкриті засоби. Комерційні програмно-технічні засоби виявлення діпфейків часто мають розвинені функціональні можливості та гарантії безпеки. Однак, вони зазвичай є дуже дорогими та вимагають високої кваліфікації персоналу для їх використання.

Комерційні програмні інструменти для виявлення глибоких фейків часто призначені для використання у великих організаціях або державних установах. Ці інструменти можуть включати алгоритми машинного навчання та інші передові технології для аналізу відео- та аудіо- даних у режимі реального часу.

Комерційні програмно-технічні засоби виявлення діпфейків часто мають розвинені функціональні можливості та гарантії безпеки. Однак, вони зазвичай є дуже дорогими та вимагають високої кваліфікації персоналу для їх використання.

До комерційних програмно-технічних засобів виявлення діпфейків належать:

1. Cellebrite UFED Cloud Analyzer: засіб для аналізу цифрових слідів в хмарних сервісах, що дозволяє знайти докази діпфейку.

2. Veracity: платформа для виявлення діпфейків, що базується на штучному інтелекті та відкритому джерелі. Забезпечує точне та швидке виявлення діпфейків [27].

З іншого боку, інструменти з відкритим вихідним кодом для виявлення підробок часто є безкоштовними і доступними для всіх, хто має доступ до Інтернету. Ці інструменти можуть бути не такими досконалими, як їхні комерційні аналоги, але вони все одно можуть бути ефективними у виявленні певних типів глибоких підробок [28].

Відкриті програмно-технічні засоби, дозволяють розробникам та дослідникам досліджувати та вдосконалювати їх. Однак, вони можуть мати

обмежені функціональні можливості та бути менш ефективними у виявленні деяких типів діпфейків.

До відкритих програмно-технічних засобів виявлення діпфейків належать багато засобів (таблиця 2.1).

Таблиця 2.1 – Основні відкриті засоби виявлення діпфейків.

FaceForensics++	Це НД криміналістики, що складається з 1000 оригінальних відео послідовностей, які були оброблені за допомогою чотирьох автоматизованих методів обробки обличчя: Deepfakes, Face2Face, FaceSwap і NeuralTextures.
Deepfake Detection Challenge	Набір засобів, що розробляється спільнотою з метою збору даних та створення більш ефективних алгоритмів виявлення діпфейків.
OpenFace	Бібліотека для розпізнавання облич та аналізу емоцій, яка може бути використана для виявлення діпфейків
NeuralTextures	Це новий графічний примітив, який може мати довільну розмірність і зберігати високорозмірний вивчений вектор ознак на піксель.

Важливо зазначити, що хоча комерційні програмні інструменти можуть мати більш просунуті функції та можливості, вони також можуть бути дорогими і недоступними для невеликих організацій або приватних осіб. Інструменти з

відкритим вихідним кодом, хоч і є безкоштовними та доступними, можуть вимагати більше технічних знань для ефективного використання.

Загалом, існує багато варіантів виявлення глибоких підробок - від комерційного програмного забезпечення до інструментів з відкритим вихідним кодом. Вибір правильного інструменту залежить від таких факторів, як бюджет, технічний досвід і конкретні потреби та вимоги.

2.1.1 Deepfake Detection Challenge

Deepfake Detection Challenge (DFDC) - це конкурс, започаткований Facebook у 2019 році у співпраці з академічними дослідниками для розробки автоматизованих методів виявлення "глибоких фейків". Мета конкурсу - розробити інноваційні та ефективні технології виявлення фейків, які допоможуть боротися з поширенням дезінформації. База даних DFDC містить тисячі відео, якими маніпулювали за допомогою методів глибокого навчання для створення реалістичного, але фейкового контенту. Перед учасниками стоїть завдання розробити алгоритми для виявлення таких маніпуляцій, а їхня ефективність оцінюється на основі набору тестових даних [29].

У конкурсі є два напрямки: один для виявлення відео, в яких використовуються вже існуючі обличчя, а другий - для виявлення відео, в яких використовуються нові обличчя, згенеровані за допомогою штучного інтелекту. Конкурс привернув значну увагу дослідників машинного навчання та комп'ютерного зору: на першому етапі в ньому взяли участь понад 2 000 учасників з усього світу. Результатом челенджу стала розробка кількох перспективних методів глибокого виявлення фейків, зокрема на основі машинного навчання, розпізнавання обличчя та інших підходів. Переможці конкурсу нагороджуються призами на загальну суму 1 мільйон доларів, з яких 500 000 доларів отримає найкращий виконавець у кожному з напрямків. DFDC вже призвів до значного прогресу в технології виявлення глибоких фейків і стимулював подальші дослідження в цій галузі.

Розробка ефективних технологій виявлення фейків має важливе значення для боротьби з поширенням дезінформації в цифрову епоху, і DFDC є важливим кроком вперед у цьому напрямку. Однак деякі експерти критикували конкурс за те, що він не представляє належним чином весь спектр методів виявлення підробок, а також за те, що він занадто зосереджений на виявленні, а не на запобіганні. Незважаючи на цю критику, Deepfake Detection Challenge зробив значний внесок у розвиток технологій виявлення підробок і продовжує надихати на подальші дослідження в цій галузі.

2.1.2 XceptionNet

XceptionNet (eXtreme Inception) – це архітектура глибокої згорткової нейронної мережі, яку представив Франсуа Шолле у 2017 році. Вона заснована на архітектурі Inception і має більш ефективну топологію мережі. Назва "Xception" розшифровується як "Extreme Inception", що означає, що вона використовує ту саму концепцію розділення просторових і канальних операцій, але виводить її на екстремальний рівень, використовуючи згортки, що розділяються за глибиною, в кожному шарі згортки. XceptionNet використовує схожий підхід до мережі Inception, але замість стандартних згорток, вона використовує згортки, що розділяються в глибину, які є більш ефективними в обчислювальному плані[29].

Основна ідея згорток, що розділяються за глибиною, полягає в тому, щоб розділити стандартну згортку на дві окремі операції: згортки за глибиною і точкові згортки. Глибинні згортки застосовують один фільтр до кожного вхідного каналу окремо, тоді як точкові згортки об'єднують результати глибинних згорток за допомогою згортки 1x1. Такий підхід значно зменшує кількість необхідних параметрів і обчислень, зберігаючи при цьому високий рівень точності.

Однією з головних переваг XceptionNet є його ефективність, як з точки зору обчислювальних ресурсів, так і з точки зору використання пам'яті. Це робить його добре придатним для розгортання на пристроях з обмеженими ресурсами, таких як мобільні телефони та вбудовані системи [30]. XceptionNet також показав свою

ефективність у різноманітних завданнях комп'ютерного зору, що виходять за рамки класифікації зображень, включаючи виявлення об'єктів і семантичну сегментацію.

XceptionNet демонструє найсучаснішу продуктивність у різних завданнях комп'ютерного зору, включаючи класифікацію зображень, виявлення об'єктів і сегментацію. У контексті виявлення глибоких підробок, XceptionNet використовується як базова архітектура для декількох методів виявлення глибоких підробок, включаючи НД FaceForensics++ і DeepFake Detection Challenge. Кілька досліджень показали високу точність при використанні XceptionNet для виявлення глибоких підробок, особливо в поєднанні з іншими методами виявлення, такими як узгодженість кадрів і збільшення руху.

Отже, XceptionNet – це потужна архітектура глибокої згорткової нейронної мережі, яка демонструє високу продуктивність у різних завданнях комп'ютерного зору, включаючи виявлення глибоких підробок. Її ефективність і точність роблять її перспективним кандидатом для майбутніх досліджень у сфері виявлення глибоких підробок.

Xception – це згорткова НМ, яка має 71 рівень. Ви можете завантажити попередньо навчену версію мережі, навчену більш ніж мільйону зображень із бази даних ImageNet. Попередньо навчена мережа може класифікувати зображення за 1000 категоріями об’єктів, наприклад клавіатура, миша, олівець і багато тварин. У результаті мережа навчилася багатим представленням функцій для широкого діапазону зображень. Мережа має вхідний розмір зображення 299 на 299 (рис 1.3). Загалом, якщо вважається, що кожен шар DNN виділяє певну функцію, то розташовувати ці шари один над одним не дуже найкраща ідея [31].

Загалом, якщо вважається, що кожен шар DNN виділяє певну функцію, то розташовувати ці шари один над одним не дуже найкраща ідея. Глибокі мережі схильні до переобладнання, а об’єднання кількох згорткових операцій разом збільшує вартість навчання мережі. Інша проблема полягає в тому, що кожен тип шару витягує різний тип інформації, як ми дізнаємося, яка трансформація (ядра) надає найбільш корисну інформацію для DNN?

Модуль Inception обчислює кілька різних перетворень (рисунок 2.1) над тим самим входом, а потім, нарешті, об’єднує весь вихід, що дозволяє моделі вирішити, які функції використовувати та в якій кількості. Є одна проблема. Це все ще обчислювально неефективно через згортки. Ці звивини відбуваються не тільки просторово, але й по глибині. Таким чином, для кожного додаткового фільтра ми повинні виконати згортку над вхідною глибиною, щоб обчислити лише одну вихідну карту, і через це глибина стає величезним вузьким місцем у DNN.

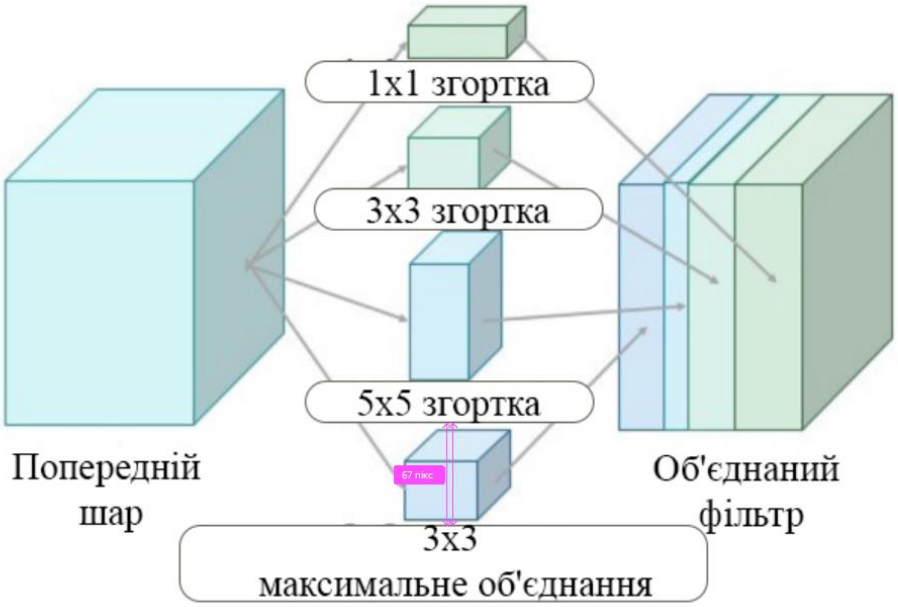


Рисунок 2.1 – Обчислення перетворень модулем Inception

Глибокі мережі схильні до переобладнання, а об’єднання кількох згорткових операцій разом збільшує вартість навчання мережі (рисунок 2.2).

net = xceptionповертає мережу Xception, навчену на наборі даних ImageNet..

net = xception('Weights','imagenet') повертає мережу Xception, навчену на наборі даних ImageNet. Цей синтаксис еквівалентний net = xception.

lgraph = xception('Weights','none') повертає ненавчену мережеву архітектуру Xception. Ненавчена модель не потребує пакета підтримки.

```
net = xception
net = xception('Weights','imagenet')

lgraph = xception('Weights','none')
```

Рисунок 2.2 – Синтаксис функції Xception

2.1.4 FaceForensics++

FaceForensics++ - це масштабний НД і бенчмарк для виявлення підробки обличь. Він був випущений у 2019 році як розширення оригінального набору даних FaceForensics. НД містить понад 1 000 оригінальних відео, 5 639 підроблених відео, згенерованих чотирма найсучаснішими методами маніпуляцій з обличчям, і понад 500 оригінальних і підроблених відео зі знаменитостями. Підроблені відео включають три типи маніпуляцій з обличчям: Deepfake, Face2Face та нейронні текстури. Цей набір даних вніс в цю тематику такі внески:

1. Автоматизований тест для де-захисту при довільному стисненні для стандартизованого порівняння, включаючи базову лінію людини.
2. Новий великомасштабний НД маніпуляційних зображень обличчя, що складається з більш ніж 1.8 мільйонів зображень.
3. 1000 відео з незайманими (тобто справжніми) джерелами та отримати реальну правду, щоб увімкнути контрольоване навчання.
4. Велика оцінка найсучаснішої ручної роботи, навчився читати підробки в різних сценаріях.
5. Найсучасніший метод виявлення підробок, адаптований до маніпуляції з обличчям.

Бенчмарк FaceForensics++ оцінює ефективність методів виявлення підробок з точки зору їхньої здатності правильно класифікувати відео з маніпуляціями як фейкові. Бенчмарк включає два протоколи оцінювання: внутрішньо- та крос-НД. Під час внутрішньо набірного оцінювання ефективність методів виявлення

оцінюється на тому самому наборі даних, на якому вони були навчені. При оцінюванні між наборами даних методи виявлення оцінюються на іншому наборі даних, ніж той, на якому вони навчалися.

Існують різні публічні реалізації DeepFakes, зокрема, FakeApp та faceswapgithub. Обличчя в цільовій послідовності замінюється обличчям, яке спостерігалося у вихідному відео або колекції зображень. Метод базується на двох автокодерах зі спільним кодуванням, які навчаються реконструювати навчальні зображення вихідного та цільового обличчя відповідно. Для обрізання та вирівнювання зображень використовується детектор облич. Щоб створити фальшиве зображення, навчені кодер і декодер вихідного обличчя застосовуються до цільового обличчя. Вихід автокодера потім змішується з рештою зображення за допомогою пуассонівської обробки зображень [40].

Розглядається виявлення підробок як задачу покадрової бінарної класифікації маніпульованих відеозаписів. Для всіх експериментів розділено НД на:

1) фіксований;

2) навчальний;

3) валідаційний;

4) тестовий набір, що складаються з 720, 140 та 140 відео відповідно.

Для оцінки ефективності роботи людей у завданні виявлення підробок, проведено дослідження користувачів, в якому взяли участь 204 учасники, переважно студенти комп'ютерних факультетів університетів. Це формує базову лінію для автоматизованих методів виявлення підробок: Після короткого ознайомлення з бінарною задачею користувачам пропонується класифікувати випадково вибрані зображення з нашого тестового набору. Вибрані зображення відрізняються як за якістю зображення, так і за методом маніпуляції; використано розподіл оригінальних і підроблених зображень у співвідношенні 50:50. Оскільки кількість часу на перевірку зображення може бути важливою, а також для імітаційного сценарію, коли користувач витрачає обмежену кількість часу на одне зображення, як це часто трапляється в соціальних мережах, зазвичай

встановлювали часовий ліміт у 2,4 або 6 секунд, після чого приховували зображення. Після цього користувачів запитували, чи є показане зображення "справжнім" чи "фейковим". Щоб переконатися, що користувачі витрачають на перевірку наявний час, питання ставилося після того, як зображення з'являлося на екрані, а не під час спостереження. Розроблено дослідження таким чином, щоб воно займало лише кілька хвилин на кожного учасника, показуючи 60 зображень, що в результаті дає колекцію з 12240 рішень людини:

На рисунку 2.3 показано результати дослідження на всіх рівнях якості, демонструючи кореляцію між якістю відео та здатністю виявляти фейки. З нижчою якістю відео ефективність роботи людини знижується в середньому з 68,7% до 58,7%. На графіку наведено усереднені показники за всіма часовими інтервалами, оскільки різні часові обмеження не призвели до суттєвих відмінностей у результатах спостережень [32].

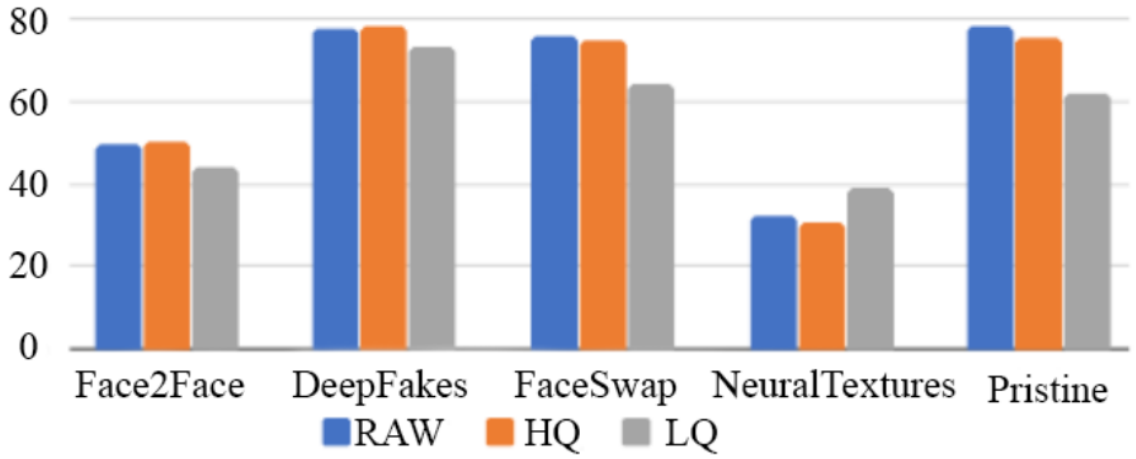


Рисунок 2.3 – Синтаксис функції Xception

За допомогою бенчмарку FaceForensics++ було оцінено кілька найсучасніших методів виявлення, зокрема MesoNet, XceptionNet і EfficientNet. Цей бенчмарк також надихнув на розробку нових методів виявлення, таких як Capsule-Forensics та двопотокові надуті 3D згорткові мережі (I3D).

Загалом, FaceForensics++ став важливим еталоном для оцінки ефективності методів виявлення підробок обличчя і сприяв розробці нових методів виявлення маніпуляцій із зображеннями.

2.1.5 NeuralTextures

NeuralTextures - це тип глибокого фейку, який фокусується на текстурі обличчя людини, а не на її рисах. Цей метод передбачає навчання нейронної мережі генерувати нові текстури для обличчя, які потім можна накласти на наявне зображення або відео, щоб створити реалістичну на вигляд фальшивку. В результаті виходить глибока підробка, яку важко виявити, оскільки НМ може генерувати нові текстури, які відповідають освітленню і затіненню оригінального зображення.

Однією з головних проблем глибокого виявлення підробок за допомогою NeuralTextures є те, що згенеровані зображення можуть мати ті самі риси обличчя та характеристики, що й оригінальні зображення, що ускладнює виявлення підроблених зображень. Однак цей підхід все ще перебуває на ранніх стадіях розвитку і потребує подальших досліджень і тестування для підвищення його точності та ефективності.

Щоб вирішити цю проблему, дослідники запропонували використовувати моделі глибокого навчання для аналізу текстури зображення та виявлення неправильних елементів , які можуть свідчити про глибоку підробку. Розроблено різні методи, включаючи аналіз узгодженості текстур обличчя на різних кадрах, вивчення розподілу значень пікселів на зображенні та пошук аномалій у процесі генерації текстур. Крім того, деякі дослідники запропонували використовувати генеративні змагальні мережі (GAN) для створення контрзаходів, які можуть ідентифікувати та видаляти нейронні текстури із зображень та відео.

Як і у випадку з іншими методами глибокого підроблення, розробка нейронних текстур викликала занепокоєння щодо можливості використання зловмисниками цієї технології в недобрих цілях, таких як політичні маніпуляції або

шантаж. Дослідники та політики працюють над розробкою кращих методів виявлення та зменшення ризиків, пов'язаних з глибокими підробками, зокрема з NeuralTextures.

2.1.6 OpenFaceForensics

OpenFaceForensics – це інструмент на основі глибокого навчання для виявлення маніпуляцій із зображеннями та відео. Він також є як фреймворк глибокого навчання з відкритим вихідним кодом, який розробила команда дослідників з Неаполітанського університету Федеріко II та Університету Салерно. Використовує поєднання традиційних методів обробки зображень і згорткових нейронних мереж (CNN) для виявлення та локалізації маніпуляцій з обличчям. Вона використовує комбінацію контрольованих і неконтрольованих методів навчання для виявлення закономірностей у даних і класифікації відео як справжніх або фальшивих. Інструмент працює, витягуючи з вхідного зображення або відео орієнтири та ознаки обличчя, які потім використовуються для навчання CNN, щоб класифікувати, чи є зображення або відео об'єктом маніпуляцій, чи ні [33].

OpenFaceForensics виявився ефективним у виявленні низки методів маніпуляцій з обличчям, включаючи підміну обличь і відтворення обличчя. Він також перевершує інші сучасні методи з точки зору точності та надійності.

Однією з ключових переваг OpenFaceForensics є його відкритий характер, здатність виявляти глибоко підроблені відео, які були піддані маніпуляціям з використанням різних технік, включаючи заміну обличчя, синхронізацію губ і відтворення обличчя. Фреймворк демонструє високу точність у виявленні широкого спектру підроблених відео, в тому числі створених за допомогою сучасних технологій, таких як NeuralTextures, про який розповідалось раніше. Все це дозволяє дослідникам і розробникам модифікувати і налаштовувати інструмент відповідно до своїх конкретних потреб. Він доступний на GitHub за ліцензією MIT. Приклад роботи можна глянути на (рисунок 2.4).

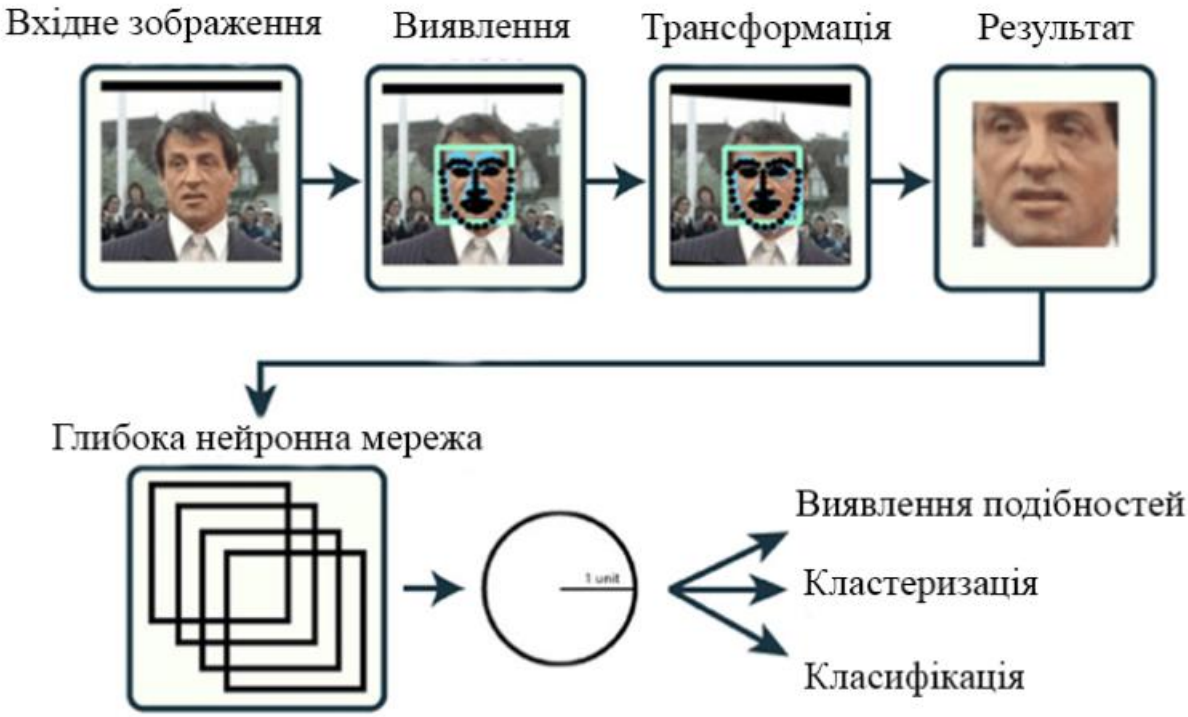


Рисунок 2.4 – Синтаксис функції Xception

Фреймворк вже використовувався в ряді досліджень і має потенціал стати цінним інструментом для виявлення та зменшення ризиків, пов'язаних з глибоко підробленими відео. Однак, як і інші методи, засновані на глибокому навчанні, OpenFaceForensics не застрахований від ворожих атак, коли зловмисник навмисно маніпулює зображенням або відео, щоб уникнути виявлення. Поточні дослідження зосереджені на розробці більш надійних методів, які можуть протистояти таким атакам [34].

2.2 Фреймворки машинного навчання

Фреймворки машинного навчання - це програмні бібліотеки, які надають інструменти та інтерфейси для побудови, навчання та розгортання моделей машинного навчання. Ці фреймворки спрощують процес розробки та впровадження моделей машинного навчання, надаючи високорівневі абстракції та

заздалегідь створені компоненти для таких поширених завдань, як попередня обробка даних, вилучення ознак та оцінка моделі.

2.2.1 TensorFlow

TensorFlow – це фреймворк машинного навчання з відкритим вихідним кодом, розроблений Google точніше командою Google Brain і вперше опублікована в 2015 році. Є бібліотекою з відкритим кодом для чисельних обчислень і великомасштабного машинного навчання. TensorFlow об’єднує низку моделей і алгоритмів машинного та глибокого навчання (так званих нейронних мереж) і робить їх корисними за допомогою загальних програмних метафор [20]. Він використовує Python або JavaScript, щоб забезпечити зручний зовнішній API для створення додатків, одночасно виконуючи ці додатки на високопродуктивній C++.

Також може навчати та запускати глибокі нейронні мережі для класифікації рукописних цифр, розпізнавання зображень, вбудовування слів, рекурентних нейронних мереж, моделей послідовності для машинного перекладу, обробки природної мови та Моделювання на основі PDE (рівняння в частинних похідних). Найкраще те, що TensorFlow підтримує масштабне прогнозування виробництва з тими самими моделями, що використовуються для навчання.

TensorFlow дозволяє розробникам створювати графи потоку даних – структури, які описують, як дані переміщуються через граф або ряд вузлів обробки. Кожен вузол на графі представляє математичну операцію, а кожне з’єднання або ребро між вузлами є багатовимірним масивом даних або тензором [20].

Фреймворк надає все це для програміста за допомогою мови Python. Ця мова проста у вивченні та з нею легко працювати, надає зручні способи вираження того, як абстракції високого рівня можна об’єднати. TensorFlow підтримується на версіях Python від 3.7 до 3.10, і хоча він може працювати на попередніх версіях.

Вузли та тензори в TensorFlow є об’єктами Python, а програми TensorFlow самі є програмами Python. Фактичні математичні операції, однак, не виконуються в Python. Бібліотеки перетворень, доступні через фреймворк, написані як

високопродуктивні двійкові файли C++. Мова програмування просто спрямовує трафік між частинами та забезпечує абстракції програмування високого рівня, щоб з’єднати їх разом.

Загалом він широко використовується для розробки нейронних мереж та інших моделей машинного навчання. TensorFlow пропонує високорівневий Keras API, а також низькорівневі API для більш складних випадків використання.

2.2.2 PyTorch

PyTorch – ще один популярний фреймворк машинного навчання з відкритим вихідним кодом, який набув популярності в останні роки. Він широко використовується для розробки моделей глибокого навчання і пропонує динамічні обчислювальні графіки, що полегшує налагодження та експерименти з моделями.

Заснований на мові програмування Python і бібліотеці Torch. Torch — це бібліотека ML з відкритим кодом, яка використовується для створення глибоких нейронних мереж і написана на мові сценаріїв Lua. Це одна з кращих платформ для досліджень глибокого навчання. Інфраструктура створена для прискорення процесу між дослідженням прототипування та розгортанням. Фреймворк PyTorch підтримує понад 200 різних математичних операцій. Популярність його продовжує зростати, оскільки він спрощує створення моделей штучних нейронних мереж [19].

Фреймворк в основному використовується вченими в галузі даних для досліджень і додатків ШІ. PyTorch випускається під модифікованою ліцензією BSD. PyTorch є пітонічним за своєю природою, що означає, що він дотримується стилю кодування, який використовує унікальні можливості Python для написання коду, який можна буде прочитати [19]. Він також популярний завдяки використанню динамічних графіків обчислень. Це дозволяє розробникам, науковцям та монтажникам нейронних мереж запускати та тестувати частину коду в режимі реального часу, не чекаючи на написання всієї програми.

PyTorch надає такі ключові функції:

1) тензорне обчислення;

2) середовище TorchScript;

3) обчислення динамічного графіка;

4) автоматична диференціація;

5) підтримка Python;

6) змінна;

7) параметр;

8) модуль;

9) функції.

Використання PyTorch може надати такі переваги:

- Пропонує розробникам просту в освоєнні та кодуванні структуру на основі Python.
- Забезпечує легке налагодження за допомогою популярних інструментів Python. Пропонує масштабованість і добре підтримується на основних хмарних платформах.
- Надає невелику спільноту, зосереджену на відкритому коді.
- Експортує моделі навчання у стандартний формат Open Neural Network Exchange (ONNX).
- Пропонує зручний інтерфейс.
- Надає опцію зовнішнього інтерфейсу C++ .
- Включає багатий набір потужних API, які розширюють бібліотеку PyTorch.

2.2.3 scikit-learn

Scikit-learn – популярна бібліотека Python для машинного навчання. Він надає вибір ефективних інструментів для машинного навчання та статистичного моделювання, включаючи класифікацію, регресію, кластеризацію та зменшення розмірності через узгоджений інтерфейс у Python. Ця бібліотека, яка в основному написана на Python, побудована на NumPy, SciPy і Matplotlib . Вона пропонує широкий спектр алгоритмів для контрольованого та неконтрольованого навчання,

а також інструменти для попередньої обробки даних, вибору моделей та оцінювання [35].

Спочатку він називався scikits.learn і спочатку був розроблений Девідом Курнапо як проект літнього коду Google у 2007 році. Пізніше, у 2010 році група творців з FIRCA (Французький дослідницький інститут у Комп'ютерні науки та автоматизація), вивели цей проект на інший рівень і зробили перший публічний випуск 1 лютого 2010 року.

Деякі з ключових особливостей scikit-learn включають (таблиця 2.1).

Таблиця 2.2 – Особливості scikit-learn.

Послідовний API	Надає послідовний API для всіх своїх моделей, що дозволяє легко перемикатися між моделями або використовувати їх у комбінації.
Широкий спектр алгоритмів	Включає широкий спектр алгоритмів для класифікації, регресії, кластеризації та зменшення розмірності, а також інструменти для вибору та вилучення ознак.
Простота у використанні	Розроблений таким чином, щоб бути простим у використанні, з інтуїтивно зрозумілими API і хорошою документацією.
Масштабованість	Розроблений для масштабування до великих наборів даних, з інструментами для паралельної та спеціальної обробки .

Загалом, scikit-learn є чудовим вибором для завдань машинного навчання на Python, особливо для початківців або тих, хто шукає бібліотеку з широким набором інструментів та алгоритмів.

2.2.4 MXNet

MXNet – це масштабований та ефективний фреймворк для машинного навчання, розроблений Amazon. Він пропонує високорівневий Gluon API, а також низькорівневі API для більш складних випадків використання.

MXNet дозволяє змішувати символічне та імперативне програмування, щоб максимізувати як ефективність, так і продуктивність. Він створений на основі динамічного планувальника залежностей, який автоматично розпаралелює як символічні, так і імперативні операції на льоту. Крім того, рівень оптимізації графіка робить символічне виконання швидким і ефективним для пам'яті. Бібліотека MXNet є портативною та легкою. Він прискорюється за допомогою графічних процесорів NVIDIA Pascal™ і масштабується між кількома графічними процесорами та декількома вузлами, що дозволяє швидше тренувати моделі [21].

Apache MXNet пропонує такі ключові функції та переваги:

1) гібридний інтерфейс: імперативний символічний гібридний API Gluon забезпечує простий спосіб створення прототипу, навчання та розгортання моделей без шкоди для швидкості навчання.

2) масштабованість: розроблений з нуля для хмарної інфраструктури, MXNet використовує розподілений сервер параметрів, який може досягти майже лінійного масштабу за допомогою кількох GPU або CPU.

3) екосистема: MXNet має інструменти та бібліотеки для комп'ютерного зору, обробки природної мови, часових рядів тощо.

4) мови: мови, що підтримуються MXNet, включають Python, C++, R, Scala, Julia, Matlab і JavaScript. MXNet також компілюється в C++.

Загалом, MXNet – це потужний і гнучкий фреймворк для глибокого навчання, який пропонує широкий спектр функцій та інструментів для побудови та розгортання нейронних мереж [36]. Масштабованість і продуктивність роблять його популярним вибором як для дослідницьких, так і для виробничих середовищ.

2.2.5 Caffe

Caffe – це фреймворк для глибокого навчання, розроблений Берклійським центром зору та навчання (BVLC). Він широко використовується для класифікації зображень та виявлення об'єктів і пропонує API на C++ та Python. Це фреймворк з відкритим вихідним кодом, який широко використовується для класифікації зображень, сегментації та інших завдань комп'ютерного зору.

Фреймворк спочатку розроблявся для дослідницьких цілей, але набув популярності в індустрії завдяки своїй високій продуктивності та масштабованості. Caffe підтримує різні архітектури нейронних мереж, включаючи згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN) та повністю зв'язані мережі. Він також надає різноманітні алгоритми оптимізації, такі як стохастичний градієнтний спуск (SGD), адаптивна оцінка моменту (Adam) та Adagrad [24].

Однією з ключових особливостей Caffe є підтримка прискорення графічного процесора, що дозволяє пришвидшити час навчання та виведення. Caffe також дуже портативний і може працювати на різних платформах, включаючи CPU та GPU. Caffe має велику та активну спільноту розробників, які роблять свій внесок у її розвиток та надають підтримку користувачам. Він також має багату екосистему попередньо навчених моделей, інструментів і розширень, які можна легко інтегрувати в існуючі робочі процеси [37].

Загалом, Caffe – це потужна і гнучка платформа для глибокого навчання, яка добре підходить для широкого спектру завдань комп'ютерного зору. Простота використання, масштабованість і висока продуктивність роблять його популярним вибором як серед дослідників, так і серед практиків.

2.2.6 Theano

Theano – ще один популярний фреймворк для глибокого навчання, який відомий своєю високою ефективністю в оптимізації та виконанні математичних обчислень. Він написаний мовою Python і був розроблений Монреальським

інститутом алгоритмів навчання (MILA) при Монреальському університеті в 2007 році. Theano в основному використовується для створення та оптимізації математичних виразів, які включають багатовимірні масиви. Це популярний вибір для дослідників і розробників, яким потрібно створювати та оптимізувати чисельні алгоритми на Python [37].

Theano розроблений, щоб бути швидким і ефективним, і використовує символьний підхід до обчислень для створення математичних виразів. Цей підхід передбачає створення обчислювального графа, який потім оптимізується для ефективного виконання на CPU або GPU. Theano має ряд особливостей, які роблять його добре придатним для задач машинного навчання, включаючи автоматичну диференціацію, динамічну оптимізацію та підтримку складних структур даних.

Однією з головних переваг використання Theano є його здатність автоматично диференціювати вирази, що полегшує створення та оптимізацію складних математичних моделей. Theano також має функцію динамічної оптимізації, яка дозволяє оптимізувати вирази на льоту на основі даних, що використовуються. Це може допомогти підвищити швидкість і ефективність моделей глибокого навчання.

Theano підтримує роботу як з CPU, так і з GPU, що робить його добре придатним для використання у високопродуктивних обчислювальних середовищах. Він також має гнучку та модульну архітектуру, що дозволяє розробникам легко розширювати та налаштовувати фреймворк для задоволення своїх конкретних потреб.

Ще однією перевагою Theano є його інтеграція з іншими популярними фреймворками глибокого навчання, включаючи TensorFlow та Keras. Це дозволяє розробникам скористатися функціями та перевагами цих фреймворків, одночасно використовуючи Theano з його унікальними можливостями.

Однак слід зазначити, що Theano не оновлювався з 2017 року, а його розробка значною мірою припинена. В результаті деякі розробники перейшли на новіші фреймворки, такі як TensorFlow та PyTorch, які пропонують більш розширені можливості та більшу підтримку з боку спільноти розробників. Тим не менш,

Theano залишається потужним інструментом для дослідників і розробників, яким потрібно створювати та оптимізувати чисельні алгоритми на Python.

2.2.7 Microsoft Cognitive Toolkit

Microsoft Cognitive Toolkit (CNTK): CNTK - це фреймворк глибокого навчання з відкритим вихідним кодом, розроблений компанією Microsoft. Він розроблений як високомасштабований та ефективний фреймворк для навчання та оцінки глибоких нейронних мереж.

Однією з головних особливостей CNTK є його здатність розподіляти навчання між декількома машинами, що дозволяє навчати дуже великі моделі на дуже великих наборах даних. CNTK також підтримує прискорення як на CPU, так і на GPU, що робить її універсальним вибором для додатків глибокого навчання.

CNTK пропонує ряд різних інтерфейсів програмування, включаючи Python, C++ та C#/.NET. Python API є найбільш часто використовуваним інтерфейсом і надає високорівневий інтерфейс для побудови нейронних мереж та виконання навчання і висновків [38].

CNTK надає широкий спектр готових архітектур нейронних мереж, включаючи мережі прямого поширення, згорткові мережі та рекурентні мережі. Ці архітектури можна легко налаштовувати та комбінувати для створення більш складних моделей [38].

На додаток до своїх основних функцій, CNTK також надає ряд високорівневих інструментів та утиліт для глибокого навчання, включаючи завантаження та попередню обробку даних, інструменти візуалізації та утиліти для розподіленого навчання.

Загалом, CNTK є потужним і гнучким фреймворком для глибокого навчання, який добре підходить для широкомасштабного навчання та оцінки нейронних мереж. Здатність розподіляти навчання між декількома машинами та підтримка прискорення як на CPU, так і на GPU роблять його привабливим вибором для додатків глибокого навчання.

2.2.8 Torch

Torch – це популярний фреймворк машинного навчання з відкритим вихідним кодом, який спочатку був розроблений в лабораторії Facebook AI Research (FAIR). Torch базується на мові програмування Lua, яка відома своєю простотою, легкістю у використанні та швидкістю. Torch широко використовується в дослідницькій спільноті та професіоналами галузі для глибокого навчання, навчання з підкріпленням та інших завдань машинного навчання.

Torch надає широкий спектр інструментів та бібліотек для побудови та навчання нейронних мереж, включаючи модулі для згорткових нейронних мереж (CNN), рекурентних нейронних мереж (RNN), мереж з довгою короткочасною пам'яттю (LSTM) та багато інших. Torch також надає гнучкий і модульний фреймворк для побудови складних моделей з використанням різних будівельних блоків, таких як шари, активації та оптимізатори [39].

Однією з ключових переваг Torch є його динамічний граф обчислень, який дозволяє розробникам змінювати модель "на льоту" під час навчання. Це дозволяє легко експериментувати з різними мережевими архітектурами та гіперпараметрами, а також налагоджувати та оптимізувати модель. Torch також надає простий у використанні інтерфейс для графічного прискорення, що дозволяє користувачам скористатися перевагами сучасних графічних процесорів для прискорення навчання та висновків. Torch можна легко інтегрувати з популярними бібліотеками GPU, такими як CUDA, cuDNN та NCCL, що робить його чудовим вибором для великомасштабних проектів машинного навчання [39].

Torch має велику і активну спільноту розробників і користувачів, а також безліч попередньо навчених моделей і прикладів коду, доступних в Інтернеті. Екосистема Torch також включає кілька високорівневих бібліотек та інструментів, таких як PyTorch - популярний фреймворк для глибокого навчання, побудований на основі Torch.

Загалом, Torch – це потужний і гнучкий фреймворк машинного навчання, який добре підходить як для дослідницьких, так і для виробничих середовищ. Його

простота, модульність та динамічний графік обчислень роблять його привабливим вибором для розробників, які хочуть легко створювати та навчати складні нейронні мережі.

2.2.9 Keras

Keras – це програмна бібліотека з відкритим вихідним кодом, яка надає високорівневий нейромережевий API мовою Python. Вона розроблена як зручна, модульна та розширювана, що робить її ідеальним інструментом для дослідників та практиків у галузі глибокого навчання. Keras була розроблена Франсуа Шолле у 2015 році і зараз є частиною проекту TensorFlow.

Keras підтримує як згорткові, так і рекурентні нейронні мережі, а також їх комбінації. Він також включає ряд вбудованих функцій втрат, функцій активації та алгоритмів оптимізації, що дозволяє легко навчати та оцінювати моделі для широкого спектру завдань. Крім того, Keras можна запускати як на CPU, так і на GPU, що дозволяє користувачам використовувати переваги обчислювальної потужності свого обладнання.

Однією з головних переваг Keras є її простота. Бібліотека розроблена так, щоб бути легкою у використанні, з простим API, який дозволяє користувачам швидко будувати і навчати нейронні мережі. Keras також включає ряд утиліт для попередньої обробки даних, таких як обробка зображень і тексту, що робить її зручним вибором для широкого спектру застосувань.

Keras також має високу модульність і розширюваність. Бібліотека дозволяє користувачам створювати складні моделі, комбінуючи готові шари та модулі або створюючи власні. Це дозволяє легко експериментувати з різними архітектурами та налаштовувати моделі під конкретні завдання. [39]

Ще однією перевагою Keras є його інтеграція з TensorFlow. Keras можна використовувати як високорівневий API для побудови моделей у TensorFlow, що дозволяє користувачам скористатися перевагами продуктивності та

масштабованості TensorFlow, водночас отримуючи вигоду від простоти використання та модульності Keras.

В цілому, Keras є потужним і гнучким інструментом для побудови моделей глибокого навчання. Його простота і модульність роблять його ідеальним вибором як для дослідників, так і для практиків, а інтеграція з TensorFlow робить його цінним доповненням до будь-якого набору інструментів для глибокого навчання.

2.3 Висновки

В даному розділі описано засоби що будуть використовуватися при реалізації власної системи. Описано різні засоби, розкрито основні характеристики та функціонал.

Розглянуто фреймворки, які необхідні для розробки системи та розкрито функціонал, який вони можуть реалізувати. Це важливо, адже потрібно підібрати підхід для реалізації методу виявлення діпфейків. Для цього ми вияснили які можуть бути варіанти використання, та їхні можливі функції в моделі нейронної мережі.

3 РОЗРОБКА МЕТОДУ ТА ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ ВИЯВЛЕННЯ ДІПФЕЙКІВ

3.1 Розробка методу виявлення діпфейків

Перед початком формування моделі нейронної мережі, був сформований метод для виявлення діпфейків. Він був вибраний саме під можливості тих фреймворків та засобів, що були використані в даній роботі.

На першому етапі виявлення відбувається з відео матеріалів, для цього був сформований НД, який був включений в робочу модель. Він був основою для навчання нейронної мережі. Вибір набору даних є важливим фактором, який може вплинути на продуктивність нейронної мережі. Важливо вибрати НД, який містить достатню кількість справжніх і фальшивих відео та охоплює широкий спектр типів відео та сценаріїв.

На рисунку 3.1 показані два стоп-кадра на різних відео, на одному точно є реальне відео а на іншому діпфейк.

Запропонований метод базується на навчанні нейронної мережі з використанням набору даних, який містить як фальшиві, так і справжні відеодані. Важлива також кількість матеріалу для реалізації даного методу, адже навчання буде відбуватись більш точніше та якісніше.

Візуально можна побачити артефакти, шуми, текстури та сильну контрастність порівняно з реальним відео матеріалом також інші відмінності. Але часто візуально важко розрізнити, або матеріалів для перевірки може бути велика кількість, тому власноруч важко буде це робити. Для цього можна налаштувати модель, яка буде цим займатися.

Отже, для початку роботи згорткової нейронної мережі, потрібно зібрати енну кількість фейкових відео та реальних. Адже в фейкових відео є подібності до змін в піксельному складі відео, тому це буде швидше та простіше.

Рисунок 3.1 – Приклад стоп-кадру фейкового відео та реального

Далі метод передбачає обробку підробленого відео або реального шляхом розбиття його на окремі кадри та перетворення кожного кадру на матрицю. Цей метод має на меті виявити відмінності між підробленим відео та справжнім шляхом аналізу байтів з матриці кожного кадру. Тобто на кожному відео утворюється стоп-кадр який перевіряється в діапазоні в 0.5 секунд.

Першим кроком є вилучення кожного кадру з фейкового та реального відео. Далі на кожному кадрі виділяється обличчя та пікселі які там знаходяться заповнюється в відповідні розміри, отже можна сказати що фон видаляється. Таку дію можна зробити за допомогою гістограми орієнтованих градієнтів, також відома як HOG. Це дескриптором ознак, подібним до Canny Edge Detector та SIFT (Scale Invariant and Feature Transform). Він використовується в комп'ютерних зорових якостей та обробці картинок з метою виявлення деталей. Техніка підраховує випадки градієнтної орієнтації в певній частині зображення.

Цей метод подібний на гістограми орієнтації країв, або на масштабне інваріантне перетворення об’єктів (SIFT). Дескриптор HOG фокусується на структурі ,може також на формі об'єкта. Він є найкращим, ніж будь-який дескриптор краю, адже він використовує величину, а також кут градієнта для прорахунку характеристик. Для областей зображення створює гістограми, використовуючи величину та орієнтацію самого градієнта. Вони розраховуються в області зображення на блок. Він вже в свою чергу розглядається як піксельна сітка, в якій градієнти створені з величини та напрямку зміни яскравості пікселя всередині блоку [40].

Для обчислення HOG-характеристик зображення першим кроком є поділ зображення на невеликі прямокутні області, які називаються комірками. У кожній комірці обчислюються градієнти зображення за допомогою фільтрів (рисунок 3.2). Величини та орієнтації градієнтів потім використовуються для побудови гістограми орієнтацій градієнтів, причому частини гістограми представляють різні діапазони орієнтацій градієнтів.

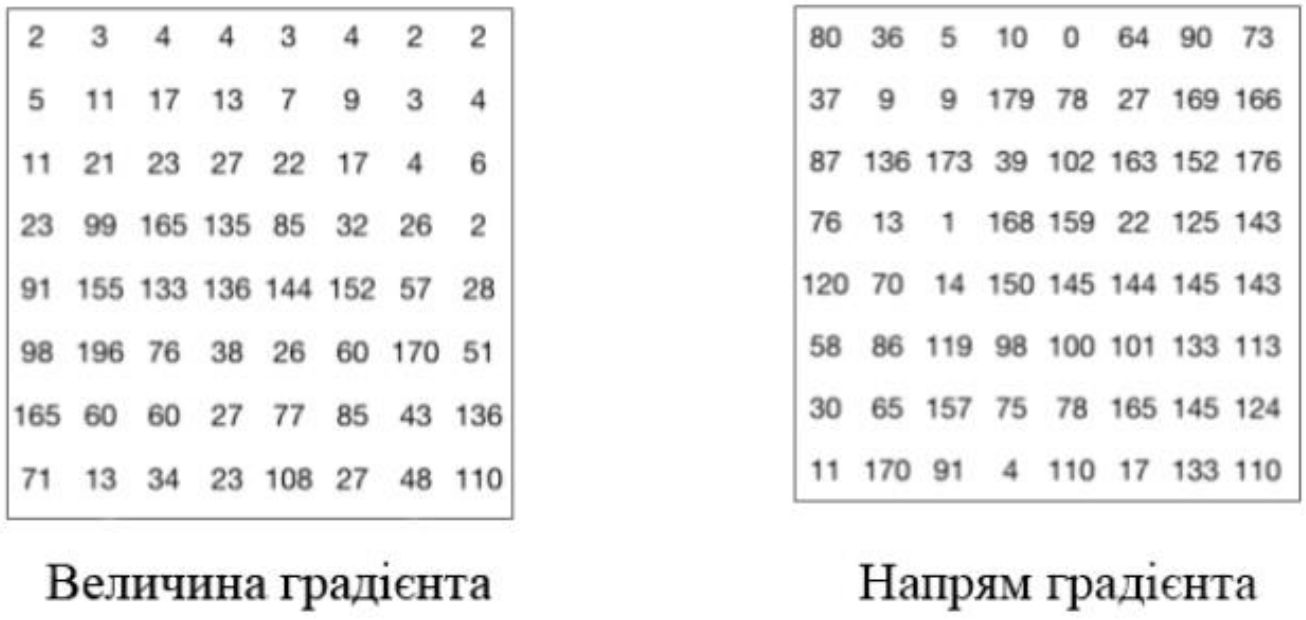


Рисунок 3.2 – Приклад розміщення градієнтів

Гістограми орієнтацій градієнта з усіх комірок у межах більшої області, яка називається блоком, потім об'єднуються для створення вектора ознак, який

представляє цей блок. Цей процес повторюється для всіх блоків на зображенні, в результаті чого створюється набір векторів ознак, які описують розподіл напрямків градієнта в різних областях зображення.

Нарешті, ці вектори ознак можуть бути використані як вхідні дані для алгоритму машинного навчання, такого як машина опорних векторів (SVM) для класифікації об'єктів на зображенні. В методі кожен таке зображення створюється в розмірі 128×128 та додається в відповідні файли.

Далі потрібно ці пікселі подати в читання . Це досягається шляхом застосування алгоритму, який перетворює кожен піксель зображення на цифру, а саме в систему байтів в результаті чого утворюється матриця з різними числами. Але ці числа можуть досягати великих значень, тому була проведена оптимізація, після якого діапазон цих чисел став від 0 до 1. Цей процес повторюється для кожного кадру відео, в результаті чого створюється колекція матриць.

Покращення набору даних була зроблена для збільшення якості навчання мережі CNN. Кінцевий НД включає велика кількість зображень, які діляться на реальні зображення та діпфейки. Початковими файлами є зображення розміром 128×128×3, де 128×128 – висота і ширина зображення в пікселях, 3 – кількість кольорових каналів.

Далі будується НМ, яка навчається за допомогою підходу керованого навчання. НД розбивається на дві частини: навчальну та перевірочну. Навчальний набір використовується для навчання мережі за допомогою зворотного поширення зі стохастичним градієнтним спуском. Під час навчання мережа вчиться розрізняти справжні та підроблені відео на основі ознак, витягнутих з кожного кадру.

Я використовую відповідну функцію втрат, яка спонукає мережу мінімізувати різницю між передбачуваними та реальними мітками. Також використовується крок попередньої обробки для оптимізації кадрів зображення. Цей крок включає перетворення кожного кадру в числа для спрощення представлення даних і зменшення обчислювальної складності [41]. Це допомагає гарантувати, що нейромережа може точно виявляти відмінності між справжніми та підробленими відео.

На кожній нейронній мережі вже є сформовані вагові значення за рахунок набору даних з ImageNet. Адже як вже ми знаємо, це великий НД позначених зображень, які використовуються для навчання та оцінювання моделей комп’ютерного зору. Він нам дасть вже з самого початку встановити вагові коефіцієнти для більш точної роботи нейронної мережі. Але важливо що значення коефіцієнтів може змінюватися після встановлення цієї функції.

В роботі CNN мережі є 3 шари у яких відбувається навчання. Кожен такий шар побудований за рахунок певних фреймворків та функцій, що спрощує сам процес навчання та покращує його точність. Наприклад в перший шар була вбудована GoogleNet, також відома як Inception v1, - це архітектура глибокої нейронної мережі, розроблена дослідниками Google. Вона була розроблена для підвищення ефективності моделей глибокого навчання, особливо з точки зору обчислювальної складності та вимог до пам'яті. Архітектура GoogleNet використовує комбінацію згорткових шарів, шарів об'єднання та початкових модулів для вилучення ознак із вхідних зображень. Початкові модулі є ключовим нововведенням в архітектурі і складаються з декількох паралельних згорткових шарів з різними розмірами фільтрів. Це дозволяє мережі виокремлювати ознаки на різних масштабах і зменшити кількість необхідних параметрів. Її можна легко імпортувати і використовувати для задач класифікації зображень або адаптувати для інших задач комп'ютерного зору, таких як виявлення об'єктів або сегментація [4,42].

Щоб уникнути перенавчання під час роботи мережі, ми відстежуємо втрати валідації та використовуємо ранню зупинку, коли втрати валідації починають зростати. Це допомагає гарантувати, що НМ не стане занадто спеціалізованою на навчальних даних і зможе узагальнювати нові дані. Після завершення навчання ми оцінюємо продуктивність нейронної мережі за допомогою валідаційного набору. Цей набір містить дані, які не були використані під час навчання, і дозволяє оцінити здатність мережі узагальнювати нові дані. Ми оцінюємо продуктивність мережі за допомогою стандартних метрик, таких як точність.

У цьому методі дуже важливо добре підібрати параметри навчання, наприклад, кількість епох навчання, розмір мережі, функції активації тощо. Також важливо правильно вибрати НД для навчання, враховуючи його розмір та різноманітність, а також збалансувати кількість фейкових та оригінальних відео в ньому. У цілому, метод виявлення діпфейків на основі нейронних мереж є досить ефективним і точним, але потребує достатньо великої кількості часу та ресурсів для навчання мережі та підготовки НД. Оптимізуючи кадри зображень і використовуючи підхід керованого навчання, можна навчити мережу, яка може точно виявляти відмінності між справжніми і підробленими відео.

Розроблений метод легко адаптується до різних типів підроблених відео і може бути використаний як у програмних, так і в апаратних засобах для виявлення підробок. Однак, враховуючи, що з кожним роком розвитку машинного навчання та зростанням обчислювальних можливостей, цей метод стає все більш доступним та швидким в застосуванні.

3.2 Архітектура програмно-технічного засобу для виявлення діпфейків

Для реалізації програмно-технічного засобу було застосоване програмне середовище яке дало змогу реалізувати даний метод для візуального перегляду всіх елементів, та перевірки ефективності роботи. Також потрібно ,щоб це середовище підтримувало всі фреймворки та різні методи

Для цього я вирішив використати Visual Studio Code (VS Code) – це безкоштовний редактор коду з відкритим вихідним кодом, розроблений компанією Microsoft. Вперше він був випущений у квітні 2015 року і швидко став одним з найпопулярніших редакторів коду серед розробників. VS Code підтримує широкий спектр мов програмування, включаючи JavaScript, Python, C++ та багато інших [56].

Хост розширень VS Code взаємодіє з оболонкою Electron за допомогою набору чітко визначених API. Така архітектура дозволяє VS Code бути дуже розширюваним. Основні можливості Visual Studio Code приведені в таблиці 3.1.

Таблиця 3.1 – Основні можливості Visual Studio Code.

Кросплатформеність	Visual Studio Code доступний для Windows, macOS та Linux. Це дозволяє легко використовувати один і той самий редактор коду.
Інтегрована налагодження	Visual Studio Code має вбудовану підтримку для налагодження коду. Вона підтримує різні сценарії налагодження, включаючи локальне
Розширення та маркетплейс	VS Code має багату екосистему розширень та плагінів, які можна використовувати для розширення його функціональності. Маркетплейс VS Code містить тисячі розширень, які можна встановити лише кількома кліками.
Легкий та швидкий	VS Code - це легкий редактор коду, оптимізований для швидкості роботи. Він побудований на основі фреймворку Electron, що дозволяє йому працювати на різних платформах, забезпечуючи при цьому роботу.
Інтуїтивно зрозумілий інтерфейс користувача	Visual Studio Code має сучасний та інтуїтивно зрозумілий користувацький інтерфейс, в якому легко орієнтуватися. Він має бічну панель, яка забезпечує швидкий доступ до важливих функцій.
Розширення та маркетплейс	VS Code має багату екосистему розширень та плагінів, які можна використовувати для розширення його функціональності.

Visual Studio Code - це потужний і розширюваний редактор коду, який широко використовується розробниками по всьому світу. Його легка і швидка архітектура в поєднанні з широкою мовною підтримкою та інтуїтивно зрозумілим інтерфейсом, роблять його популярним вибором для розробки додатків на широкому спектрі мов програмування і платформ (рисунок 3.3). Завдяки багатій

екосистемі розширень та плагінів, VS Code можна легко налаштувати відповідно до конкретних потреб окремих розробників та команд [56].

Отже, в цьому інструменті використано з широкого спектру мов програмування саме Python. Щоб почати використовувати Python у VS Code, спочатку потрібно встановити розширення Python. Це розширення надає багато корисних функцій, таких як завершення коду, підсвічування синтаксису та можливості налагодження. Воно також включає вбудований лінтер, який може допомогти мені виявити та виправити помилки у коді. Однією з найпотужніших функцій VS Code є можливості налагодження. Розширення Python забезпечує повністю інтегрований досвід налагодження, дозволяючи крокувати по коду, встановлювати точки зупинки та перевіряти змінні [43]. Це може заощадити багато часу при налагодженні складних Python-додатків.

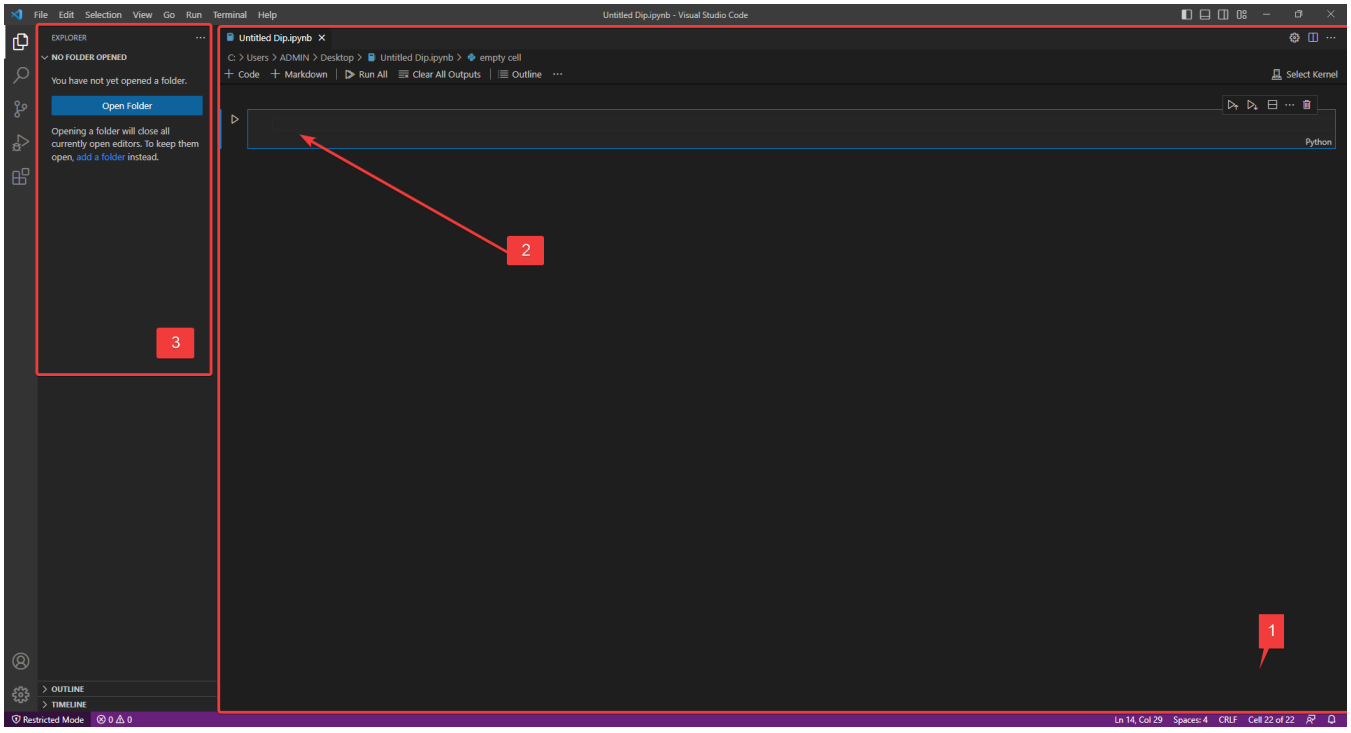


Рисунок 3.3 – Приклад інтерфейсу

На рисунку ми можемо спостерігати такі програмні області:

1) Робоча область, місце де ми можемо взаємодіяти з блоком коду, для якого відведено найбільше місця. Назва функцій та інше підсвічується різними

кольорами. Тут же вказано шлях до файлу, а трохи вище розташовані вкладки-файли, якими можна переміщатися. За допомогою них ми можемо додавати не просто один файл із кодом, а цілий проект;

2) Область в якій відбувається запис коду в блок, їх може бути декілька і кожен з них окремо запускається;

3) Ліва панель управління, яка включає 5 основних вкладок: «Провідник», «Пошук», «Система управління версіями», «Запуск коду» і «Розширення».

Інші області є панелями інструментів та сервісів.

Таким чином, Visual Studio Code надає потужне та гнучке середовище для розробки на Python. Завдяки багатому набору функцій і розширень воно є чудовим вибором для Python розробників усіх рівнів.

Тому, на початку роботи було сформовано блоки в інструментарії Visual Studio Code де окремо в кожному блоці писались функції для всієї НМ. Спершу були включенні в проект бібліотеки, які є обов'язковим елементом для реалізації даного методу та й всієї системи (рисунок 3.4).

```
[1]
import dlib
import cv2
import os
import re
import json
from pylab import *
from PIL import Image, ImageChops
```

Рисунок 3.4 – Імпорт бібліотек в робочий проект

Далі потрібно створити папки або каталоги в яких буде розміщений НД з діпфейками (рисунок 3.5). Для цього використовується така функція як os.makedirs(), використовується для рекурсивного створення каталогу або набору вкладених каталогів. Функція отримує два аргументи: ім'я та режим. Аргумент name вказує каталог, який потрібно створити, включно з необхідними батьківськими каталогами, і може бути абсолютним або відносним шляхом.

Аргумент mode визначає дозволи, які буде встановлено для каталогу, і є необов'язковим.

```
os.makedirs('./deepfake/dataset')
os.makedirs('./deepfake/dataset/real')
os.makedirs('./deepfake/dataset/fake')
```

Рисунок 3.5 – Створення папок для набору даних

Як можна спостерігати, було створено три каталоги: загальний НД, реальні відео та фейкові.

Наступним кроком потрібно розбити відео на фрейми та занести отримані зображення в папки, базуючись на метаданих. Для цього можна використати такий метод з бібліотки OpenCV:

cap.read() – це метод, який дозволяє зчитувати кадри з відеофайлу або камери. Метод повертає два значення: ret і frame.ret - це булеве значення, яке вказує, чи було успішно прочитано кадр, чи ні. Якщо ret дорівнює True, то кадр успішно прочитано, а якщо False, то кадрів для читання більше немає. frame - масив, який містить дані зображення кадру у вигляді значень пікселів. Тому було зроблено це наступним чином:

```
while cap.isOpened():
frameId = cap.get(1)
ret, frame = cap.read()
```

Наступним кроком потрібно зробити так, щоб створювалися фрейми з кожного відео з часом в 0,5 секунди, для цього можна використати умову frameId, в якій встановлюємо потрібні параметри. Та вже після цього використовується HOG, а точніше деякі його функції.

В роботі використано цикл для перебору кожного виявленого обличчя у кадрі за допомогою змінної face_rects. Для кожного обличчя код витягує координати рамки (прямокутника, що оточує обличчя) за допомогою методів left(), top(), right() і bottom() об'єкта. Потім код обрізає область вихідного кадру, визначену рамкою, за допомогою нарізки масиву і присвоює її змінній зображення. Воно обрізане та містить лише область обличчя.

Наступним кроком потрібно зберегти зображення у вказаному каталозі, у відповідному розмірі, для цього використовується cv2.imwrite – це функція з бібліотеки OpenCV, яка використовується для збереження зображення на диск. Далі зображення буде збережено у форматі PNG cv2.resize(crop_img, (128, 128))- змінює розмір обрізаного зображення обличчя до квадратної форми з розмірами 128x128 пікселів. Зображення зі зміненим розміром буде збережено на диск. Повна частина коду:

```
cv2.imwrite(real_directory+vid.split('.')[0]+'_'+str(count)+'.png',cv2.resize(crop_img, (128, 128)))
```

Далі ми можемо провести навчання нейронної мережі для виявлення глибоких підробок за допомогою набору даних справжніх і підроблених зображень. Спочатку потрібно сформувати код який буде зчитувати зображення з каталогів справжніх і підроблених за допомогою функції load_img з модуля перед процесування з бібліотеки Keras. Потім кожне зображення перетворюється в масив чисел з плаваючою комою за допомогою функції img_to_array. Потім масив нормалізується шляхом ділення кожного значення на 255.0 для масштабування значень пікселів у діапазоні від 0 до 1.

Справжні та підроблені зображення зберігаються окремо у списках X та Y, де X містить згладжені масиви зображень, а Y містить двійкові мітки для кожного зображення (1 для справжнього, 0 для підробленого).

Списки X та Y потім використовуються для навчання НМ за допомогою послідовної моделі Кераса. Модель має три шари: вхідний шар, прихований шар зі

128 вузлами та вихідний шар з одним вузлом (оскільки це задача бінарної класифікації).

Нарешті, модель оцінюється на перевірочній множині, яка в даному випадку збігається з навчальною множиною. Бінарні мітки для валідаційної множини зберігаються у файлі Y_val_org.

Робимо нормалізацію, для того щоб спростити навантаження НМ на систему та зробити розбиття на дані для тренування моделі та дані для тестів. Для цього я реалізував таку форму нормалізації для створених масивів в мережі.

```
X = np.array(X)
Y = to_categorical(Y, 2)
```

У наведеному фрагменті коду змінні X та Y створюються як масиви NumPy. Масив X містить дані сплющених зображень як справжніх, так і фальшивих зображень. Масив Y містить відповідні мітки для кожного зображення, де 1 означає справжнє, а 0 - підроблене.

Масиви X та Y передаються через функцію to_categorical() з Keras, яка перетворює масив Y у формат з однократним кодуванням. Це означає, що кожна мітка (або 0, або 1) представлена у вигляді двійкового вектора довжиною 2, де індекс, що відповідає мітці, дорівнює 1, а всі інші індекси дорівнюють 0. Це корисно для задач класифікації де є багато класів, де вихідні класи не є взаємовиключними.

Наприклад, якщо Y мав значення [0, 1, 1, 0, 0, 1] до проходження через to_categorical(), він перетворюється на двовимірний масив форми (6, 2), де кожна мітка представлена у вигляді бінарного вектора [[1, 0], [0, 1], [0, 1], [1, 0], [1, 0], [0, 1]].

Також в моделі є функція зміни форми на 128×128×3, це дуже важливо, щоб реалізувати даний метод пошуку діпфейків. Для цього створився рядок коду X = X.reshape(-1, 128, 128, 3), що перетворює масив вхідних даних X у 4D тензор форми (кількість_зразків, висота, ширина, канали), де:

1) кількість_зразків – загальна кількість зображень у наборі даних;
2) висота і ширина – розміри кожного зображення;
3) канали – кількість колірних каналів у зображенні (у цьому випадку 3 для RGB).

Значення -1 у функції reshape є заповнювачем, який вказує Numpy обчислити необхідний розмір автоматично на основі загальної кількості елементів у масиві та інших заданих розмірів. У цьому випадку це означає, що кількість зразків буде автоматично розраховано на основі загального розміру X та інших заданих розмірів. Таким чином, змінений масив X - це 4D тензор форми (кількість_зразків, 128, 128, 3), який можна використовувати як вхідні дані для згорткової НМ (CNN) для навчання моделі глибокого навчання для виявлення глибоких підробок.

Також є функція, яка поділяє тестові та тренувальні дані, це важливо, щоб НМ не переплутала дані з набору даних. Для цюого використовується rain_test_split(), яка йде з бібліотеки sklearn, що використовується для розділення НД на навчальні та валідаційні набори.

X_train, X_val, Y_train, Y_val = train_test_split(X, Y, test_size = 0.2, random_state=5)

У фрагменті коду X - це вхідний масив даних, а Y - вихідний або цільовий масив. Параметр test_size використовується для визначення розміру валідаційного набору, який дорівнює 0.2, тобто для валідації буде використано 20% від загального обсягу даних. Параметр random_state використовується для ініціалізації внутрішнього генератора випадкових чисел, який гарантує, що при кожному запуску коду генерується однакова послідовність випадкових чисел, що забезпечує відтворюваність.

Функція повертає чотири масиви: X_train, X_val, Y_train та Y_val. X_train та Y_train - це вхідні та вихідні масиви навчання відповідно, а X_val та Y_val - вхідні та вихідні масиви перевірки відповідно.

Наступник кроком потрібно створити НМ , щоб вони змогла повністю задовільнити метод виявлення діпфейків. Для цього створюється шар мережі з встановленими вже вагами на кожен нейрон з використанням набору даних з ImageNet. Як ми знаємо він вже в собі має велику кількість перевірених даних на діпфейки та реальні, що дає нам дає точність при перевірці.

Загалом, модель InceptionResNetV2 – це архітектура глибокої згорткової нейронної мережі, яка використовує модуль Inception та залишкові зв'язки для підвищення точності задач класифікації зображень.

Модуль Inception – це блок вилучення ознак, який використовує декілька згорткових шарів з різними розмірами ядра і паралельно виконує операції об'єднання для вилучення ознак на різних масштабах. Виходи цих паралельних згорткових шарів об'єднуються і передаються на наступний шар.

Залишковий зв'язок дозволяє моделі вивчати залишкові функції, які можуть бути використані для обходу шарів і полегшують вивчення складних функцій.

Використовуючи ці два методи разом, модель InceptionResNetV2 здатна досягти найсучаснішої продуктивності в задачах класифікації зображень.

Встановлення googleNet_model.trainable = True робить всі шари в попередньо навченій моделі здатними до навчання. Це означає, що в процесі навчання ваги і зміщення цих шарів можуть бути оновлені відповідно до даних і функції втрат поточного завдання, на додаток до остаточного шару класифікації, який ми додали до моделі. Це може допомогти поліпшити продуктивність моделі шляхом точного налаштування попередньо навчених шарів під конкретну задачу. Однак це також може збільшити ризик перенавчання, якщо навчальні дані обмежені або якщо модель занадто складна.

Щоб створити НМ з 3 шарами, для нашого методу потрібно її зробити послідовною, для цього потрібно використати Sequential() – це клас у Keras, який дозволяє створити нейромережеву модель, що являє собою лінійний стек шарів. Ми можемо додавати шари до цієї моделі, викликаючи метод .add() на об'єкті моделі.

Ось коротке пояснення кожного доданого шару в коді (таблиця 3.2)

Таблиця 3.2 – Опис кожного шару в функції Sequential

googleNet_model	Це попередньо навчена модель InceptionResNetV2, яка вже була навчена на мільйонах зображень. Вона використовується як екстрактор ознак, де вихідні дані цієї моделі подаються на наступні шари нейронної мережі.
GlobalAveragePooling2D	Цей шар використовується для обчислення середнього значення кожної карти ознак на виході попереднього шару.
Dense	Цей шар є повністю зв'язаним шаром, який бере вихідні дані попереднього шару і застосовує до них лінійну операцію. У цьому коді є два щільних шари. Перший має 1024 одиниці і використовує функцію активації ReLU, а другий має 512 одиниць і також використовує функцію активації ReLU.

В третьому шарі під назвою Dense ми який є щільний до моделі з 2 вихідними одиницями і використовуємо функцію активації softmax. Функція softmax зазвичай використовується для задач багатокласової класифікації і повертає розподіл ймовірностей для кожного класу. У нашому випадку ми маємо два класи: справжній та підроблений, і виходом функції softmax буде ймовірність того, що вхідне зображення належить до кожного класу. Вихід цього шару подається на вихід моделі, де використовується для обчислення втрат та оновлення ваг під час

навчання. Під час виведення вихід цього шару використовується для прогнозування нових даних.

Далі був організований оптимізатор Adam , з вказаними параметри для навчання даної моделі, для покращення швидкості.

Adam (Adaptive Moment Estimation) – це алгоритм оптимізації, який використовується в глибокому та машинному навчанні. Він є розширенням стохастичного градієнтного спуску (SGD), який використовує ковзні середні значення параметрів для визначення швидкості навчання.

Алгоритм відстежує експоненціальне спадаюче середнє минулих градієнтів і експоненціальне спадаюче середнє минулих квадратів градієнтів. Ці оцінки використовуються для оновлення параметрів моделі на кожній ітерації. Швидкість навчання також адаптивно коригується на основі першого та другого моментів градієнтів.

Перевагами оптимізатора Адама є швидка збіжність, хороше узагальнення та стійкість до спотворених даних. Він широко використовується в глибокому навчанні завдяки здатності обробляти великі обсяги даних і складні моделі. Однак важливо правильно налаштувати гіперпараметри відповідно до конкретної задачі, що розв'язується. model.compile() – це метод у Keras, який налаштовує процес навчання моделі, що включає визначення функції втрат, оптимізатора та метрик.

У цьому конкретному випадку функцією втрат є бінарна перехресна ентропія, яка зазвичай використовується для задач бінарної класифікації. В якості оптимізатора використовується алгоритм Adam, який є адаптивним алгоритмом оптимізації швидкості навчання. Швидкість навчання встановлюється на 1e-5, а також задаються інші параметри, такі як beta_1, beta_2 та епсилон. Нарешті, для оцінки моделі використовується метрика точності. Після того, як модель скомпільована, вона готова до навчання за допомогою методу model.fit(), який приймає навчальні дані і запускає процес навчання із заданою конфігурацією.

Отже, була зроблена готова модель яка реалізувала мето виявлення фейкових відео. Після запуску НМ, можемо спостерігати за створенням всіх шарів та їхні зв’язки (рисунок 3.6).

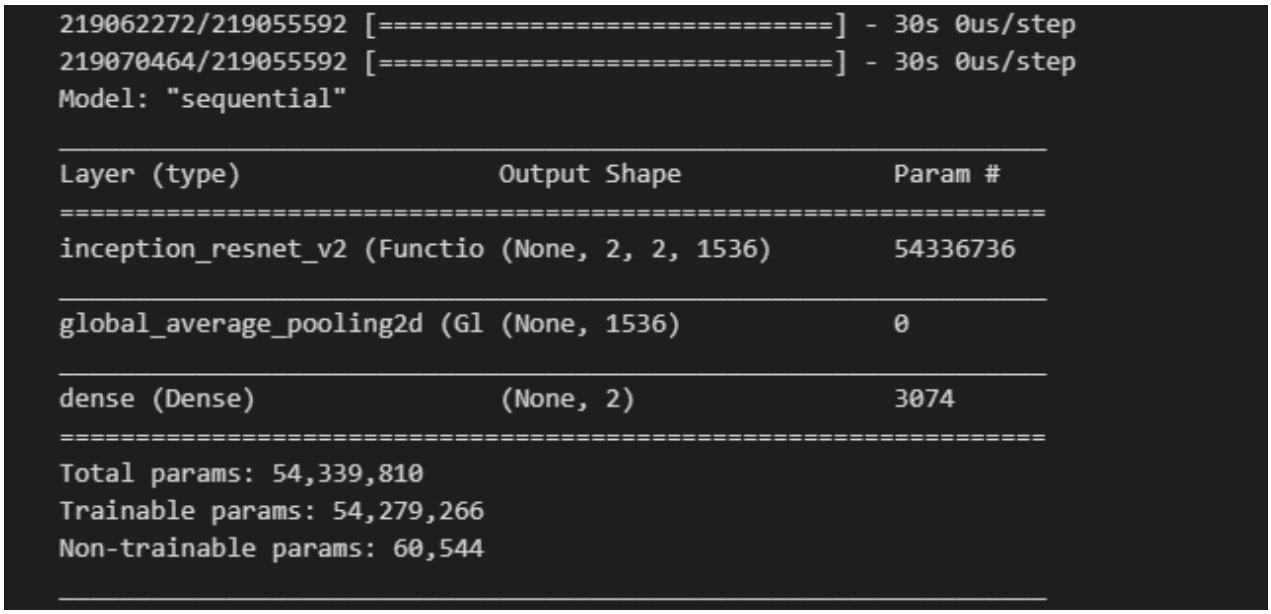


Рисунок 3.6 – Результат запуску НМ

Як можемо спостерігати Тут модель має три шари:

1) шар InceptionResNetV2 з вихідною формою (None, 2, 2, 1536), який має 54 336 736 параметрів, що навчаються.

2) шар GlobalAveragePooling2D з вихідною формою (None, 1536), який не має параметрів, що навчаються.

3) шар Dense з формою виходу (None, 2), який має 3,074 параметри, що навчаються.

Загальна кількість параметрів, що навчаються в моделі, становить 54,339,810.

Ще до навчання НМ потрібно додати зупинку, якщо виникнуть якісь проблеми, щоб навчання не відбувалось негативним.

Та вже потім зробити процес навчання на певну кількість раз, в даному випадку на 20 раз. Для цього був написаний код.

Він навчає модель на навчальних даних X_train та Y_train, використовуючи метод підгонки об'єктної моделі Keras.

У ньому вказується кількість епох для навчання (EPOCHS) та розмір партії даних (BATCH_SIZE). Під час навчання використовуються валідаційні дані X_val та Y_val для моніторингу роботи моделі на даних, яких вона раніше не бачила.

Метод fit повертає об'єкт історії, який містить втрати та точність моделі під час навчання для кожної епохи. Об'єкт early_stopping є екземпляром зворотного виклику EarlyStopping, який використовується для дострокової зупинки навчання, якщо втрата валідації не покращується протягом певної кількості епох (терпіння).

Таким чином, цей код навчає модель на наданих даних і контролює її роботу, використовуючи дані валідації. Послідовно НД проходить всі 20 епосів, та на кожному з них виділяється певна кількість точності, та змінюється вага значення нейрону.

Якщо продуктивність моделі не покращується протягом певної кількості епох, навчання припиняється достроково. Хід навчання зберігається в об'єкті історії для аналізу та візуалізації. Результат роботи цього коду (рисунок 3.7).

```
Epoch 1/20
30/30 [==============================] - 180s 6s/step - loss: 0.6403 - accuracy: 0.7299 - val_loss: 0.7087 - val_accuracy: 0.7059
Epoch 2/20
30/30 [==============================] - 185s 6s/step - loss: 0.5455 - accuracy: 0.8151 - val_loss: 0.6477 - val_accuracy: 0.7433
Epoch 3/20
30/30 [==============================] - 189s 6s/step - loss: 0.4676 - accuracy: 0.8679 - val_loss: 0.5961 - val_accuracy: 0.7687
Epoch 4/20
30/30 [==============================] - 193s 6s/step - loss: 0.3994 - accuracy: 0.8967 - val_loss: 0.5478 - val_accuracy: 0.7741
Epoch 5/20
30/30 [==============================] - 194s 6s/step - loss: 0.3455 - accuracy: 0.9228 - val_loss: 0.5019 - val_accuracy: 0.7874
Epoch 6/20
30/30 [==============================] - 195s 7s/step - loss: 0.2805 - accuracy: 0.9472 - val_loss: 0.4666 - val_accuracy: 0.8302
Epoch 7/20
30/30 [==============================] - 194s 6s/step - loss: 0.2251 - accuracy: 0.9585 - val_loss: 0.4216 - val_accuracy: 0.8449
Epoch 8/20
30/30 [==============================] - 190s 6s/step - loss: 0.1821 - accuracy: 0.9716 - val_loss: 0.3932 - val_accuracy: 0.8570
Epoch 9/20
30/30 [==============================] - 199s 7s/step - loss: 0.1410 - accuracy: 0.9749 - val_loss: 0.3553 - val_accuracy: 0.8703
Epoch 10/20
30/30 [==============================] - 205s 7s/step - loss: 0.1060 - accuracy: 0.9850 - val_loss: 0.3180 - val_accuracy: 0.8877
Epoch 11/20
30/30 [==============================] - 192s 6s/step - loss: 0.0881 - accuracy: 0.9840 - val_loss: 0.3108 - val_accuracy: 0.8877
Epoch 12/20
30/30 [==============================] - 190s 6s/step - loss: 0.0677 - accuracy: 0.9866 - val_loss: 0.3077 - val_accuracy: 0.8850
Epoch 13/20
...
Epoch 19/20
30/30 [==============================] - 179s 6s/step - loss: 0.0276 - accuracy: 0.9943 - val_loss: 0.2983 - val_accuracy: 0.9051
Epoch 20/20
30/30 [==============================] - 179s 6s/step - loss: 0.0256 - accuracy: 0.9930 - val_loss: 0.3149 - val_accuracy: 0.9024
```

Рисунок 3.7 – Результат запуску НМ

Навчання моделі завершилося успішно. Модель навчалася на 20 епохах з розміром партії 100. Точність навчання досягла 99,3%, а точність валідації склала 90,24% в останній епосі. Втрати на навчальній вибірці значно зменшилися з 0,64 в

першій епосі до 0,0256 в останній епосі. Аналогічно, втрати на перевірочній множині зменшилися з 0,70 у першій епосі до 0,31 у фінальній епосі.

Загалом, модель, досягла хорошої точності і низьких втрат як на навчальній, так і на валідаційній вибірках, що вказує на те, що вона може добре узагальнювати нові дані.

3.3 Перелік використаних фреймворків у моделі

Розроблена робота включає в себе велику кількість бібліотек та фреймворків, в цьому розділі буде розглянуто які саме були використані та що з них було взято для реалізації методу.

Першим фреймворк використовується dlib – це сучасний інструментарій C++, що містить алгоритми машинного навчання та інструменти для створення складного програмного забезпечення на C++ для вирішення реальних проблем. Це програмне забезпечення з відкритим вихідним кодом, яке можна використовувати безкоштовно як для комерційних, так і для некомерційних цілей. dlib широко використовується для різних завдань, включаючи виявлення об'єктів, розпізнавання облич та сегментацію зображень.

В роботі цим набором даних було використано фронтальний детектор обличчя бібліотеки dlib для виявлення у кожному кадрі відео. Потім він обрізає виявлені обличчя і зберігає їх як png-зображення в директорії на основі мітки, вказаної у файлі metadata.json для відповідного відео.

Перед збереженням зображення змінюються до розміру (128, 128). Загалом, цей код готує НД обрізаних зображень обличь, витягнутих з навчальних відео для глибокого виявлення фейків.

Другий фрейм os: використовується для переліку всіх файлів з розширенням .png в обох директоріях. Функції використовується для перетворення зображень у масиви, та для завантаження зображень з каталогу. Потім зображення нормалізуються шляхом ділення кожного пікселя на 255.0, а мітки присвоюються залежно від того, чи є зображення справжнім, чи підробленим.

Третій використовується фреймворк TensorFlow: фреймворк машинного навчання з відкритим вихідним кодом, розроблений компанією Google. TensorFlow надає різноманітні інструменти та бібліотеки для побудови та навчання моделей машинного навчання, включаючи моделі глибокого навчання. У цьому конкретному коді для побудови моделі використовується API Keras від TensorFlow.

Також Keras: це високорівневий API для побудови та навчання моделей глибокого навчання, який працює поверх TensorFlow (та інших фреймворків). Він надає зручний та інтуїтивно зрозумілий інтерфейс для визначення та налаштування моделей глибокого навчання. Конкретна архітектура моделі, що використовується в цьому коді, - InceptionResNetV2, яка є архітектурою глибокої згорткової нейронної мережі, що досягла найсучаснішої продуктивності на різноманітних завданнях комп'ютерного зору.

Наступний NumPy: Це бібліотека чисельних обчислень на мові Python, яка використовується для роботи з масивами та матрицями. У цьому коді вона використовується для маніпулювання даними.

Далі sklearn: бібліотека машинного навчання на Python, яка використовується для попередньої обробки даних, моделювання та оцінки. У цьому коді використовується для розділення даних на навчальні та валідаційні набори.

OpenCV: популярна бібліотека комп'ютерного зору, яка надає функції для обробки зображень і відео, а dlib - бібліотека C++, що використовується для машинного навчання, обробки зображень і завдань комп'ютерного зору. Код також використовує бібліотеку Os для маніпулювання файлами та каталогами і бібліотеку json для завантаження метаданих про відео.

3.4 Висновки

В третьому розділі досягнута основна мета роботи, а саме розроблено метод виявлення діпфейків. Описано всі необхідні елементами та функції. Вибрали спосіб оптимізації, який покращив роботу методу.

Розроблено програмно-технічний засіб, в якому реалізовано розроблений метод, з використанням функціоналу Visual Studio Code, безкоштовного редактору коду з відкритим вихідним кодом. В ньому за допомогою мови Python було реалізовано всі основні моменти. Використали функції оптимізаторів, таких як Adam, та інші. Далі створили модель НМ, в якій є три шари і вона є послідовними.

Порівняли результати перевірки тренувальних даних з НД, та валідаційних. Дійшли висновку що метод працює правильно, та виконує свою роботу.

В результаті було переглянуто фреймворки, які використовувались у роботі. При реалізації були використані функції з таких бібліотек як, OpenCV, TensorFlow і т.д.

4 ТЕСТУВАННЯ ТА ЕКСПЕРЕМЕНТАЛЬНІ ДОСЛІДЖЕННЯ

4.1 Реалізація методу виявлення діпфейків в іншій моделі НМ

Задля перевірки методу була створена ще одна модель НМ, яка в тестуванні дасть змогу порівняти ефективність даного методу, та його масштабованість. Ця модель повторює метод виявлення діпфейків, та всі його алгоритми, але змінюється сама структура нейронної мережі.

Щоб її реалізувати я вирішив використати трошки інші функції фреймворків, для експерименту. Цей код визначає модель глибокого навчання для класифікації зображень з використанням методу навчання з переносом. Зокрема, він використовує попередньо навчену згорткову НМ під назвою VGG16 як базову модель для вилучення ознак.

Виклик функції VGG16 з аргументом include_top=False вказує, що останній шар попередньо навченої моделі (який використовується для класифікації) слід виключити з моделі. Аргумент «weights» вказує, що модель має бути ініціалізована попередньо навченими вагами на наборі даних ImageNet. Аргумент input_shape визначає форму вхідних зображень, які будуть оброблятися моделлю. Нарешті, аргумент classes має значення 2, що вказує на те, що модель є бінарним класифікатором.

Далі за допомогою виклику функції послідовної мережі, створюється нова модель з назвою secondModel. Базова модель base_model додається першим шаром до цієї моделі. Потім до secondModel додається шар Flatten(), який перетворює вихідні дані base_model з 3D-тензора в 1D-тензор.

Результатом цього коду є узагальнена модель Keras Sequential з двома шарами.

Перший шар - це попередньо навчена модель VGG16 з видаленими верхніми шарами (повністю з'єднаними шарами) і вагами, завантаженими з набору даних ImageNet.

Другий шар – це шар Flatten, який бере вихідні дані моделі VGG16 і перетворює їх на одномірний вектор. Зведення відображає вихідні форми кожного

шару і кількість параметрів, що навчаються і не навчаються, в моделі. Загальна кількість параметрів у моделі становить 14,714,688, і всі вони піддаються навчанню.

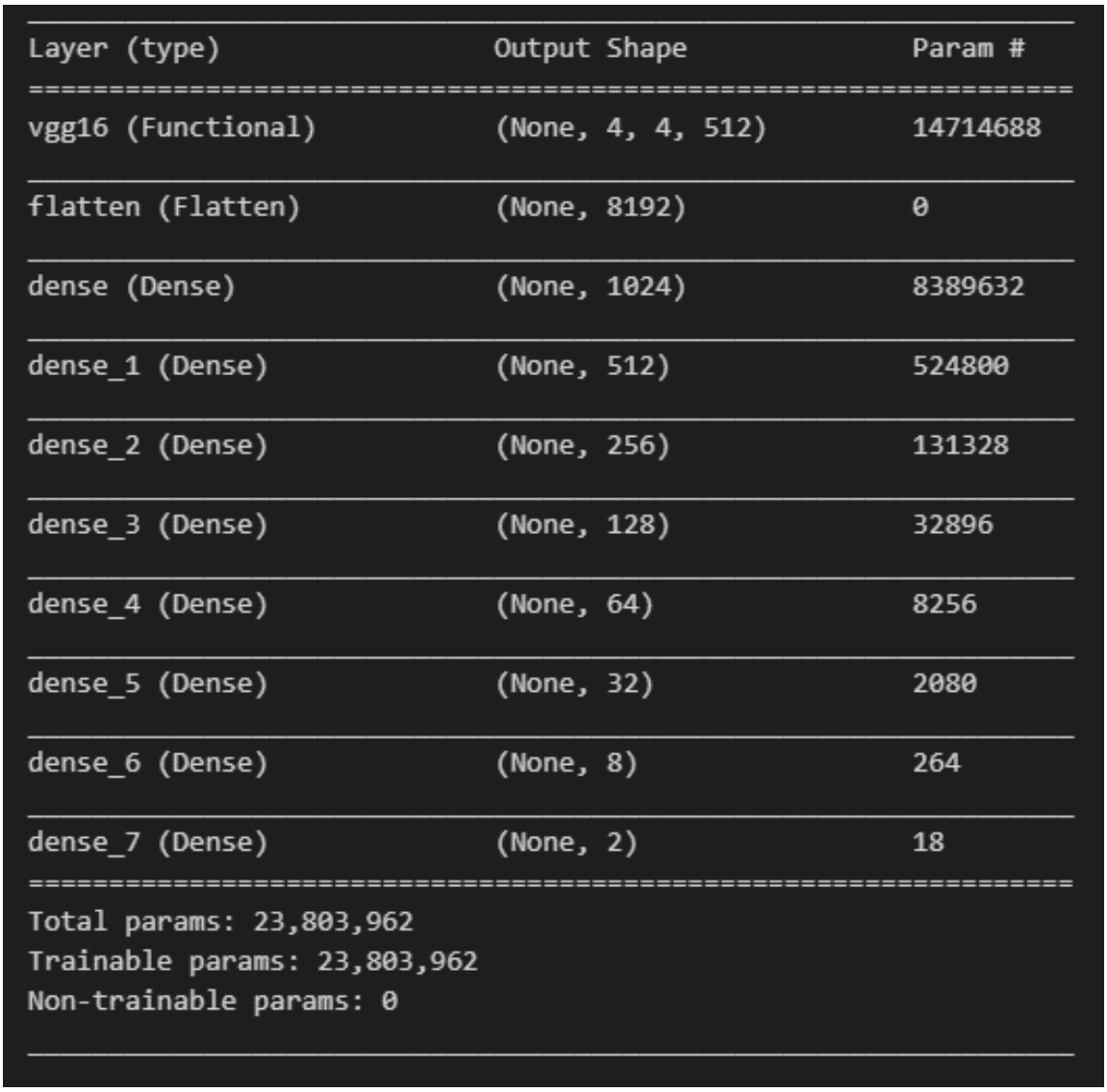
Ось результат відображення архітектури та кількості параметрів моделі (рисунок 4.1).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 4.1 – Результат запуску НМ в іншій моделі

Далі для дослідження можна визначити нейромережеву модель за допомогою бібліотеки Keras. Модель будується шляхом додавання декількох шарів щільно з'єднаних нейронів (рисунок 4.2):

1) перший шар має 1024 нейрони і приймає вхід розмірністю 2;
2) другий шар має 512 нейронів;
3) третій шар має 256 нейронів;
4) четвертий шар має 128 нейронів;
5) п'ятий шар має 64 нейрони;
6) шостий шар має 32 нейрони;
7) сьомий шар має 8 нейронів;
8) останній шар має 2 нейрони, яка виводить ймовірності для бінарної класифікації.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 4, 4, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 1024)              8389632
_________________________________________________________________
dense_1 (Dense)              (None, 512)               524800
_________________________________________________________________
dense_2 (Dense)              (None, 256)               131328
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_6 (Dense)              (None, 8)                 264
_________________________________________________________________
dense_7 (Dense)              (None, 2)                 18
=================================================================
Total params: 23,803,962
Trainable params: 23,803,962
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 4.2 – Результат запуску НМ за допомогою бібліотеки Keras

Отримана модель має загалом 23,803,962 параметри, всі з яких піддаються навчанню. Архітектуру моделі можна візуалізувати як послідовність шарів, де вхідні дані по черзі проходять через кожен шар для отримання кінцевого результату. У висновку надається інформація про кожен шар, включаючи його тип, форму вихідного сигналу та кількість параметрів.

Продовжуючи експеримент, можна створити код, який навчає модель нейронної мережі за допомогою функції fit() з Keras. Функція буде працювати протягом вказаної кількості епох (в реалізації буде 5 епох), протягом яких модель буде навчена на навчальних даних і будуть обчислені метрики втрат і точності. Дані валідації будуть використані для оцінки того, наскільки добре модель узагальнює

нові дані. Змінна history міститиме історію навчання, яку можна використовувати для візуалізації втрат і точності під час навчання та валідації.

4.2 Тестування моделей на точність виявлення діпфейків

Головне що є при реалізації методу та його засобів, це є результат. Тому було проведено тестування всіх моделей, та які були отримані значення.

В головній моделі ми отримали такі результати, що при навчанні на 20 епохах з розміром партії 100. Точність навчання була 99,3%, а точність перевірки випадкових відео склала 90,24% в останній епосі.

Втрати на тренувальній вибірці сильно зменшилися з 0,64 в початковій епосі до 0,0256 в останній. Також, втрати на цій яку перевіряють зменшилися з 0,70 на початковій епосі до 0,31 у останній.

Результатами експериментальної моделі, де ми реалізували метод виявлення діпфейків, з іншою НМ, ми отримаємо такі значення (рисунок 4.3).

```
Epoch 1/5
15/15 [==============================] - 417s 28s/step - loss: 0.5035 - accuracy: 0.7977 - val_loss: 0.5028 - val_accuracy: 0.7888
Epoch 2/5
15/15 [==============================] - 419s 28s/step - loss: 0.4844 - accuracy: 0.7977 - val_loss: 0.4823 - val_accuracy: 0.7888
Epoch 3/5
15/15 [==============================] - 443s 30s/step - loss: 0.4644 - accuracy: 0.7977 - val_loss: 0.4559 - val_accuracy: 0.7888
Epoch 4/5
15/15 [==============================] - 457s 31s/step - loss: 0.4392 - accuracy: 0.7984 - val_loss: 0.4221 - val_accuracy: 0.8035
Epoch 5/5
15/15 [==============================] - 451s 30s/step - loss: 0.4059 - accuracy: 0.8181 - val_loss: 0.3875 - val_accuracy: 0.8463
```

Рисунок 4.3 – Результат перевірки НМ після проходження всього набору даних

На виході показано точність навчання та перевірки, а також втрати для нейромережевої моделі для 5 епох. Модель навчалася на даних з розміром партії 200 та валідацією 0.2.

Для кожної епохи на виході відображаються втрати при навчанні, точність навчання, втрати при валідації та точність перевірки. Втрати при навчанні та валідації зменшуються, а точність навчання та перевірка зростають з кожною

епохою. Модель досягла точності 84,63% на валідаційних даних після 5 епох навчання.

Можна скласти висновок, що початкова модель показала кращі результати, порівнюючи експериментальну, але й кількість епох була використана більшою. Тому наступну перевірка буде проводитись саме з початковою моделлю.

Наступним кроком буде показати роботу методу на випадковому одному відео, для чистоти перевірки. Для цього створили код, який знов реалізує метод виявлення діпфейків. Він ініціалізує вхідну форму для моделі, а потім переходить до читання відеофайлу за допомогою OpenCV.

Він використовує попередньо навчений детектор обличчя з бібліотеки dlib для виявлення у кожному кадрі відео. Якщо обличчя виявлено, він обрізає його з кадру, змінює розмір до 128x128, перетворює його в масив NumPy, нормалізує його і прогнозує вираз за допомогою попередньо навченої моделі нейронної мережі. Потім передбачений вираз виводиться на консоль. Цей процес повторюється для кожного кадру відео.

Результат роботи був виведений набір бінарних чисел [0,1,1,0,0,0,0,0,0,0], модель знаходить клас обличчя як 0 або 1, і результат (передбачений клас) виводиться на консоль. На основі даних модель прогнозує 0 для перших 7 кадрів і 1 для наступних 3 кадрів.

Для перегляду ефективності методу, потрібно побудувати графік. Для цього було включено бібліотеку Matplotlib – це бібліотека візуалізації даних у Python, яка використовується для створення статичних, інтерактивних та анімованих візуалізацій у Python. Вона надає різноманітні інструменти та функції для побудови графіків та візуалізацій, таких як лінійні діаграми, точкові діаграми, гістограми, 3D-діаграми та багато іншого. Matplotlib добре налаштовується і може використовуватися разом з іншими бібліотеками, такими як NumPy, Pandas та Seaborn, для створення інформативних та візуально привабливих графіків.

Тому було створено графік з двома підграфіками за допомогою цієї бібліотеки. Перший графік показує значення точності навчання та валідації для 20 епох (рисунок 4.4).

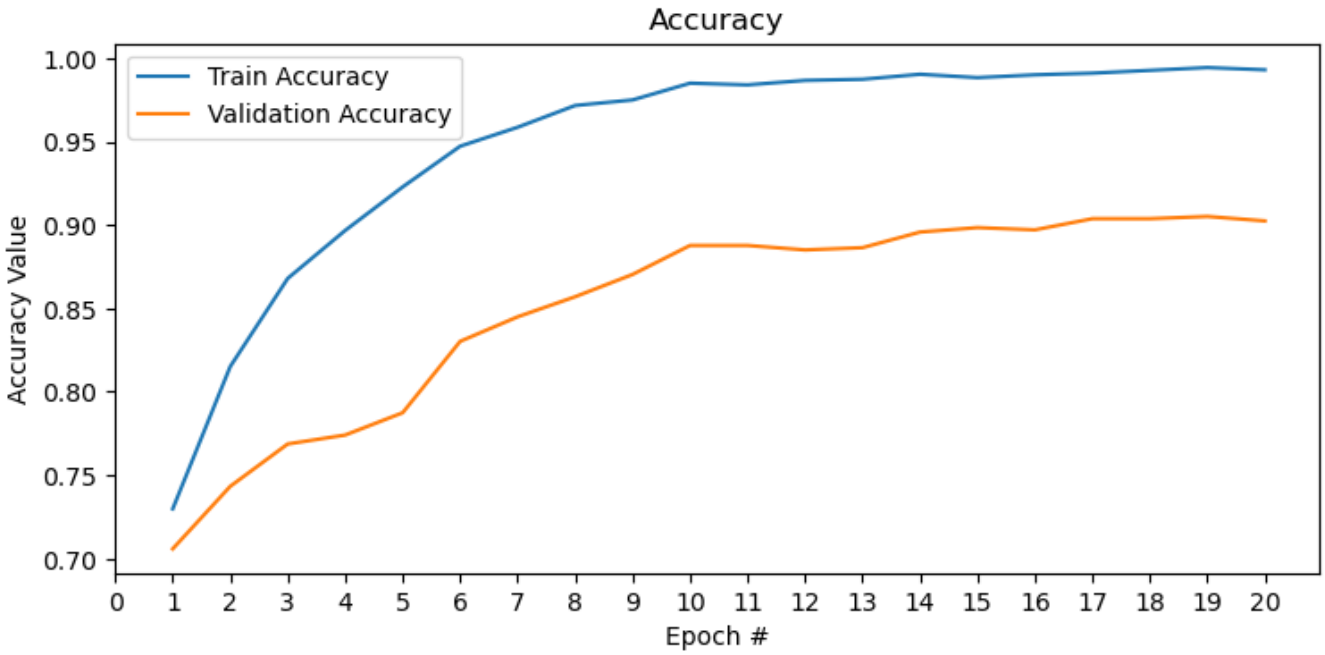


Рисунок 4.4 – Графік значення точності навчання та валідації.

Другий графік показує значення втрат навчання та валідації для 20 епох (рисунок 4.5).

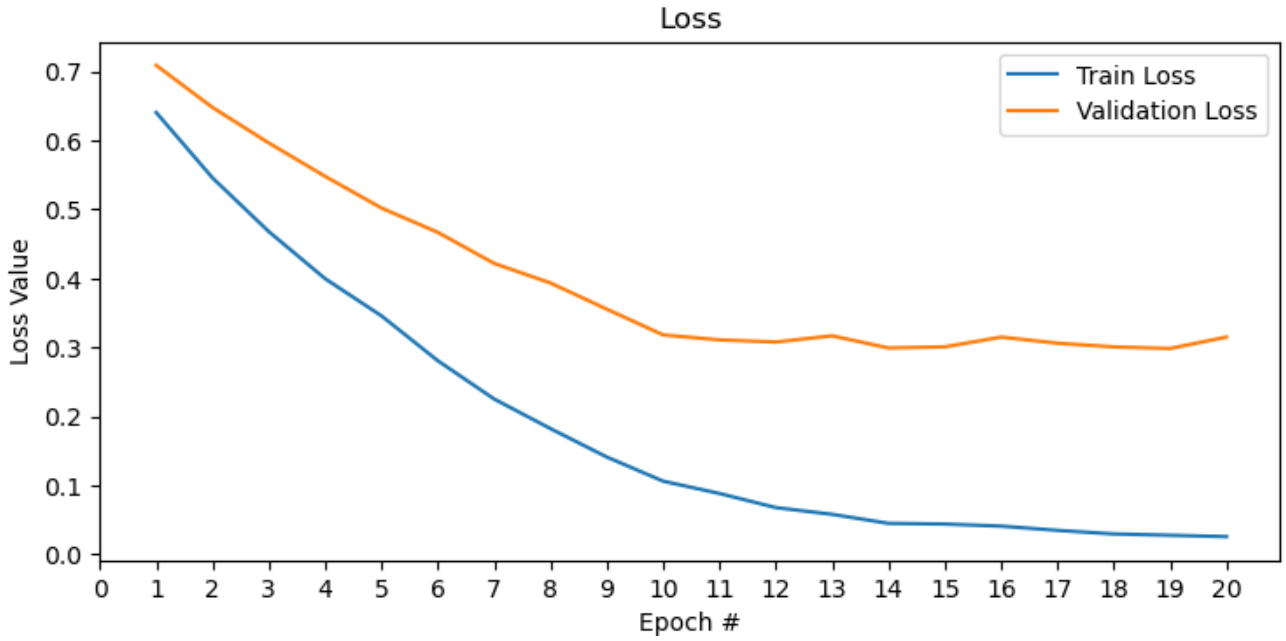


Рисунок 4.4 – Графік значення втрат навчання та валідації.

Для побудови графіків точності та втрат використовується об'єкт історії, який повертається методом підбору моделі. Вісь x на кожному підграфіку представляє епохи, а вісь y - значення точності та втрат. Код також встановлює мітки і назви

піддіаграм і рисунка. Нарешті, він виводить певне значення для кожної піддіаграми.

Як можна спостерігати за графіком, це частинка гілки параболи, яка равномірно піднімається вверх, що говорить про постійне та стабільне навчання набору даних на кожному з епох, тобто метод та програмно-технічний засіб працює правильно та без перешкод.

4.3 Тестування за допомогою матриці невідповідностей

Перед тестуванням потрібно зрозуміти, що таке матриця невідповідностей. Матриця невідповідностей – це таблиця, яка використовується для оцінки ефективності моделі класифікації на наборі тестових даних, для яких відомі істинні значення. Вона також відома як матриця помилок [44].

Матриця помилок має два виміри, і кожен вимір представляє передбачуваний клас і фактичний клас даних. Існує чотири можливих результати задачі бінарної класифікації:

1) істинно позитивний (ІП): Коли фактичний клас є позитивним і модель правильно прогнозує його як позитивний;

2) хибнопозитивний (ХП): коли фактичний клас є негативним, але модель помилково прогнозує його як позитивний;

3) хибнонегативний (FN): Коли фактичний клас є додатним, але модель помилково пророкує його як від'ємний;

4) істинно від'ємний (TN): Коли фактичний клас є від'ємним, а модель правильно прогнозує його як від'ємний.

Ці результати використовуються для заповнення клітинок матриці помилок. Рядки матриці представляють фактичний клас, а стовпці - передбачений клас.

Головна діагональ матриці представляє правильно класифіковані екземпляри, в той час як елементи поза діагоналлю представляють неправильно класифіковані екземпляри. Діагональні елементи матриці дають загальну кількість правильних класифікацій для кожного класу, тоді як елементи поза діагоналлю

дають кількість неправильних класифікацій [57].

Матриця невідповідностей може бути використана для обчислення різних оціночних метрик для класифікаційної моделі, включаючи точність, достовірність, пригадування і оцінку F1 [44].

Точність – це відсоток правильних прогнозів від загальної кількості прогнозів. Прецизійність – це відношення істинних спрацьовувань до суми істинних спрацьовувань і помилкових спрацьовувань.

Відтворення – це відношення істинно-позитивних результатів до суми істинно-позитивних і хибно-негативних. Показник F1 – це середнє гармонійне значення точності та пригадування.

Таким чином, матриця невідповідностей є важливим інструментом для оцінювання ефективності класифікаційної моделі. Вона надає детальну розбивку прогнозів і фактичних міток класів, що дозволяє оцінити точність, достовірність, пригадування та інші важливі метрики. Використовуючи інформацію з матриці, дослідники можуть приймати обґрунтовані рішення для підвищення продуктивності моделей класифікації в різних областях і додатках.

Для реалізації цієї матриці, була створена модель яка це реалізує. Цей код визначає функцію print_confusion_matrix, яка приймає два аргументи так та ні. Вона обчислює матрицю помилок за допомогою функції з scikit-learn, а потім виводить значення true positive, false positive, false negative і true negative. Потім вона створює теплову карту матриці помилок за допомогою seaborn і відображає її за допомогою matplotlib.

Нарешті, функція викликається з двома аргументами. Прогнозовані значення отримуються шляхом застосування методу predict() навченої моделі на X. Також є функція, що використовується для отримання індексу максимального значення вздовж вказаної осі, яка представляє прогнозовану мітку класу для кожної вхідної вибірки. Отримана матриця помилок виводиться на екран (рисунок 4.5).

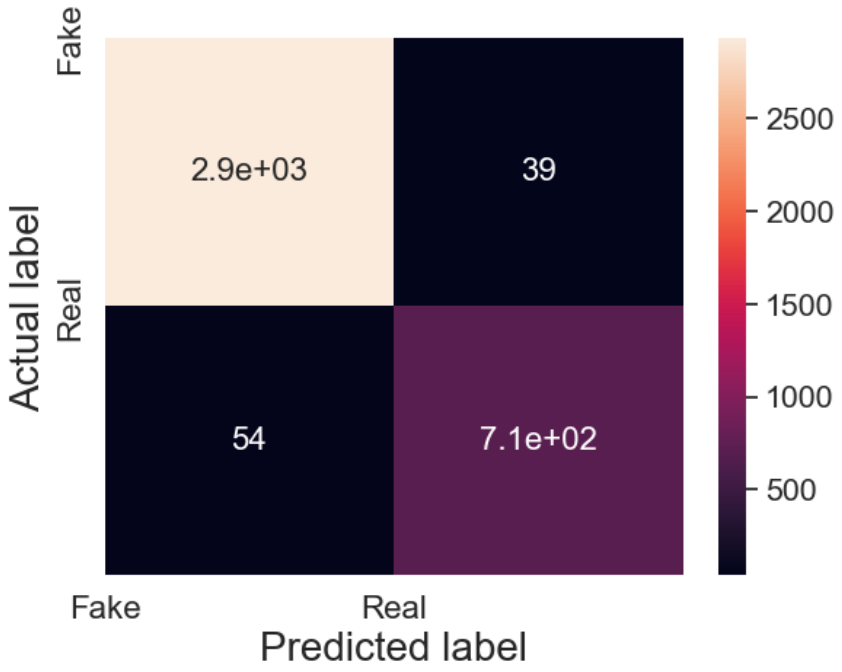


Рисунок 4.5 – Матриця невідповідностей для основної моделі НМ

В результаті виконання матриці, можемо спостерігати такі значення (таблиця 4.1).

Таблиця 4.1 – Результати матриці невідповідностей

Істинно позитивний	2948
Хибно позитивний	39
Хибно негативний	54
Істинно негативний	709

Можемо спостерігати, що хибних значень мінімальна кількість, отже метод працює дуже добре.

Для експерименту перевіримо матрицю невідповідностей іншої модель НМ, та отримаємо результат (рисунок 4.6).

Можемо спостерігати, що в другі моделі НМ значення відрізняються в гіршу сторону, а саме велика кількість хибних значень в перевірці діпфейків, що були реальними відео, а не фейковими, та навпаки істино реальні відео значно мала кількість, порівняно з основною моделлю НМ яку я використовував. Тому цей

метод значно гірший ніж попередник, не зважаючи на його простоту та простіший спосіб реалізації.

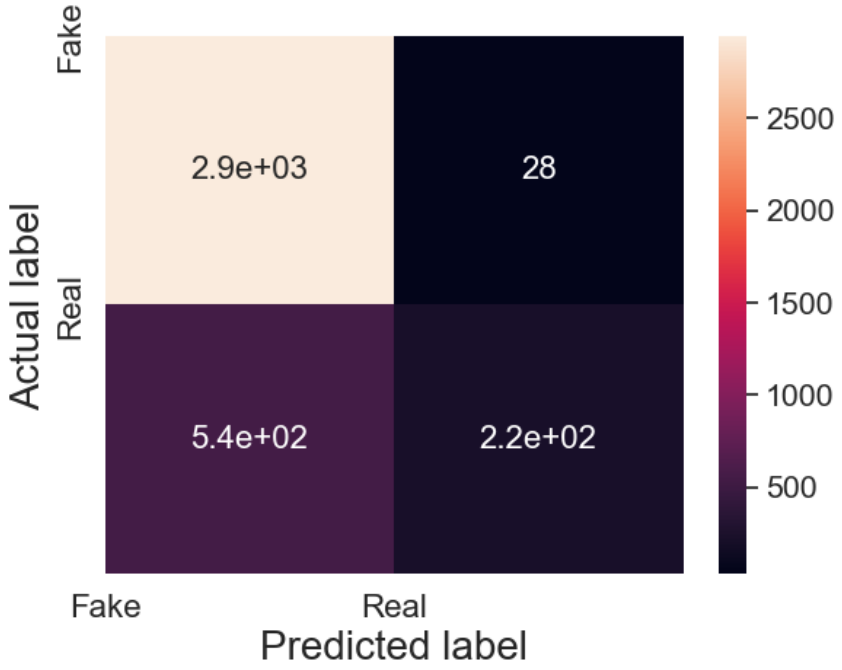


Рисунок 4.6 – Матриця невідповідносте для експериментальної моделі НМ

В результаті виконання цієї матриці, можемо переглянути такі значення (таблиця 4.2).

Таблиця 4.2 – Результати матриці невідповідностей експериментальна.

Істинно позитивний	2948
Хибно позитивний	28
Хибно негативний	545
Істинно негативний	218

4.4 Висновки

В цьому розділі проведено дослідження методу виявлення діпфейків. Розроблено альтернативний метод НМ для перевірки роботи, який був простіший в роботі та використовував менше епох перевірки.

Проведено тестування цих моделей, з використання НД, які підібрані для нього, тобто тестові та валідаційні дані. Проте тестування показало, що основний метод є ефективнішим та точнішим, порівняно з експериментальним.

Також була створена матриця невідповідностей, що дала можливість точно показати рух нейронів в мережі, та як формувався вибір, чи це фейк чи реальне відео.

Результати показали що перший метод був точнішим та ефективнішим, порівняно з простішим експериментальним.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи проведено глибокий аналіз підробок, включаючи всебічний огляд існуючих методів їх виявлення. Також запропоновано і розроблено власний метод виявлення підробок з використанням як програмних, так і апаратних засобів, які згодом були протестовані і оцінені. Результати тестів демонструють ефективність запропонованого підходу і свідчать про те, що метод є цінним внеском у сферу виявлення підробок.

Окрім розробки та дослідження методу, в роботі визначено область в якій можуть бути проведені дослідження з метою подальшого вдосконалення виявлення підробок. Зокрема, матриця неповноцінностей, представлена в роботі, може бути корисним інструментом для визначення сфер, де існуючі методи виявлення є мало ефективними, і де можуть бути проведені додаткові дослідження для усунення цих недоліків.

Виявлення недоліки в існуючих методах виявлення підробок свідчить про те, що в цій галузі ще є значний простір для вдосконалення, і робота може слугувати корисною відправною точкою для майбутніх досліджень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. De keersmaecker J., Roets A. ‘Fake news’: Incorrect, but hard to correct. The role of cognitive ability on the impact of false information on social impressions. *Intelligence*. 2017. Vol. 65. P. 107–110.

2. Figueira A., Oliveira L. The current state of fake news: challenges and opportunities. *Procedia Computer Science*. 2017. Vol. 121, P. 817–825.

3. Fletcher J. Deepfakes, Artificial Intelligence, and Some Kind of Dystopia: The New Faces of Online Post-Fact Performance. *Theatre Journal*. 2018. Vol. 70, No 4. P. 455–471.

4. Floridi L. Artificial Intelligence, Deepfakes and a Future of Ectypes. *Philosophy & Technology*. 2018. Vol. 31, No 3. P. 317–321.

5. Wagner T.L., Blewer A. The Word Real Is No Longer Real: Deepfakes, Gender, and the Challenges of AI-Altered Video. *Open Information Science*. 2019. Vol. 3, No 1 P. 32–46.

6. Basic concepts of artificial intelligence. URL: https://www.opentrends.net/en/article/basic-concepts-artificial-intelligence (дата звернення: 13.02.2023).

7. Generative adversarial network (GAN). URL: https://www.techtarget.com/searchenterpriseai/definition/generative-adversarial-network-GAN (дата звернення: 20.01.2023)

8. Is Seeing Still Believing? Leveraging Deepfake Technology for Livestock Farming. URL: https://www.frontiersin.org/articles/10.3389/fvets.2021.740253/full#F1 (дата звернення: 03.02.2023)

9. PyTorch. URL: https://www.techtarget.com/searchenterpriseai/definition/PyTorch (дата звернення: 10.03.2023)

10. What is TensorFlow? The machine learning library explained. URL: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html (дата звернення: 18.02.2023)

11. Bates M. E. Say What? 'Deepfakes' Are Deeply Concerning. *Online Searcher*. 2018. P. 64.

12. Chawla R. Deepfakes: How a pervert shook the world. *International Journal of Advance Research and Development*. 2019. Vol.4. P. 4–8.

13. AI, Deep Learning, and Machine Learning: A Primer. URL: https://a16z.com/2016/06/10/ai-deep-learning-machines/ (дата звернення: 20.03.2023).

14. Spivak R. “Deepfakes”: The newest way to commit one of the oldest crimes. *The Georgetown Law Technology Review*. 2019. Vol. 3, No 2. P. 339–400.

15. Andrew L. Maas. Rectifier Nonlinearities Improve Neural Network Acoustic Models. 2013.

16. Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao.Joint face detection and alignment using multitask cascadedconvolutional networks. *IEEE Signal Processing Letters*. Oct 2016. Vol 23, No 10. P. 1499–1503.

17. Deepfakes: Can Biometric Authentication Defeat the New Cybersecurity Nightmare?. URL: https://www.spiceworks.com/it-security/identity-access-management/articles/deepfake-biometric-security-measure/ (дата звернення: 14.02.2023).

18. Chollet F., Xception. Deep learning with depthwise separable convolutions. In. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017. P. 1251– 1258.

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

20. Chierchia G., et al. A bayesian-MRFapproach for PRNU-based image forgery detection. *IEEE Trans. Inf. Forensics Secur*. 2014, Vol. 9, No. P. 4554– 567.

21. Is There Any Difference Between Scikit-Learn and Sklearn? URL: https://towardsdatascience.com/scikit-learn-vs-sklearn-6944b9dc1736 (дата звернення: 11.03.2023)

22. Apache MXNet. URL: https://www.nvidia.com/en-us/glossary/data-science/mxnet/ (дата звернення: 24.02.2023)

23. What Is Caffe? URL: https://builtin.com/machine-learning/caffe (дата звернення: 14.03.2023)

24. Top 9 Deep Learning Frameworks Worth Learning. URL: https://www.codingdojo.com/blog/deep-learning-frameworks (дата звернення: 20.02.2022)

25. The Microsoft Cognitive Toolkit. URL: https://learn.microsoft.com/en-us/cognitive-toolkit/ (дата звернення: 17.08.2022)

26. Korshunova I., et al. Fast face-swap using convolutional neural networks. In. *IEEE International Conference on Computer Vision (ICCV)*. 2017. P. 3697– 3705.

27. What is Visual Studio Code? Microsoft's extensible code editor. URL: https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html (дата звернення: 13.03.2023).

28. Understanding Confusion Matrix. URL: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62 (дата звернення: 29.03.2023).

29. Campolo A, Sanfilippo MR, Whittaker M, Crawford K. AI Now 2017 Report. *AI Now Institute at New York University*, 2017. P. 37 .

30. Aldwairi M., & Alwahedi, A. 2018. Detecting Fake News in Social Media Networks. *Procedia Computer Science*.2018. P. 215–222.

31. Anwar S., Milanova M., Anwer M., & Banihirwe A. Perceptual Judgments to Detect Computer Generated Forged Faces in Social Media. In F. Schwenker, & S. Scherer (Eds.), *Multimodal Pattern Recognition of Social Signals in Human-Computer-Interaction. MPRSS 2018*. Lecture Notes in Computer Science, 2019.vol. 11377. Springer, Cham.

32. Britt M. A., Rouet J.-F., Blaum D., & Millis K. A Reasoned Approach to Dealing With Fake News. *Policy Insights from the Behavioral and Brain Sciences*. 2019. P. 94–101.

33. Cybenko A. K., & Cybenko G. *AI and Fake News. IEEE Intelligent Systems*. 2018. P. 3–7.

34. What is PyTorch, and How Does It Work: All You Need to Know. URL: https://www.simplilearn.com/what-is-pytorch-article (дата звернення: 12.02.2023)

35. Theano vs TensorFlow : A Quick Comparision of Frameworks. URL: https://www.edureka.co/blog/theano-vs-tensorflow/ (дата звернення: 13.03.2023)

36. Yongyi Lu, Yu-Wing Tai, and Chi-Keung Tang. Conditionalcyclegan for attribute guided face image generation. *In Eu-ropean Conference on Computer Vision*.2018.

37. Zhihe Lu, Zhihang Li, Jie Cao, Ran He, and Zhenan Sun.Recent progress of face image synthesis. *In IAPR Asian Con-ference on Pattern Recognition*. 2017.

38. Nicolas Rahmouni, Vincent Nozick, Junichi Yamagishi, and Isao Echizen. Distinguishing computer graphics from nat-ural images using convolution neural networks. *In IEEEWorkshop on Information Forensics and Security*. 2017. P. 1–6.

39. Supasorn Suwajanakorn, Steven M. Seitz, and IraKemelmacher-Shlizerman. Synthesizing Obama: learninglip sync from audio. *ACM Transactions on Graphics (TOG).* 2017. Vol. 36, No 4.

40. Peng Zhou, Xintong Han, Vlad I. Morariu, and Larry S.Davis. Two-stream neural networks for tampered face de-tection. *In IEEE Computer Vision and Pattern RecognitionWorkshops*. 2017. P.1831–1839.

41. Justus Thies, Michael Zollh ofer, and Matthias Nießner. De-ferred neural rendering: Image synthesis using neural tex-tures. *ACM Transactions on Graphics 2019 (TOG).*

42. Astropy Documentation. URL: https://docs.astropy.org/en/stable/ (дата звернення: 15.10.2022).

43. SciPy documentation. URL: https://docs.scipy.org/doc/scipy/ (дата звернення: 12.10.2022).

44. NumPy documentation. URL: https://numpy.org/doc/stable/ (дата звернення: 15.11.2022).

45. AlphaGo. URL: https://deepmind.com/research/case-studies/alphago-the-story-so-far (дата звернення: 15.01.2023).

46. 4 Types of Artificial Intelligence. URL: https://www.bmc.com/blogs/artificial-intelligence-types/ (дата звернення: 15.02.2023).

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

48. Sabir E., Cheng J., Jaiswal A., AbdAlmageed W., Masi I. and Natarajan P. Recurrent Convolutional Strategies for Face Manipulation 48 Detection in Videos. *CVPR Workshops*. 2019.

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

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

51. De Lima O., Franklin S., Basu S., Karwoski B. and George A. *Deepfake Detection Using Spatiotemporal Convolutional Networks*. 2020.

52. Ciftci U.A., Demir I. and Yin L. 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, pp.1-10.

53. Grigory Antipov, Moez Baccouche, and Jean-Luc Duge-lay. Face aging with conditional generative adversarial net-works. *In IEEE International Conference on Image Process-ing*, 2017.

54. Thies J., et al. Face2face: real-time face capture and reenactment of RGBvideos. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016. P. 2387– 2395.

55. Suwajanakorn S., Seitz S.M., Kemelmacher-Shlizerman, Synthesizing I. Obama: learning lip sync from audio. *ACM Trans, Graph*. 2017. Vol. 36, No 4. P. 1– 13.

56. Szegedy C., et al. Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2015. P. 1– 9.

57. Abadi M. et al. Tensorflow: A system for large-scale machine learning. *Proceedings of the USENIX Conference on Operating Systems Design and Implementation*. Nov. 2016. Vol. 16. P. 265-283.

58. Antipov G.,. Baccouche M. and Dugelay J.-L. Face aging with conditional generative adversarial networks. 2017. arXiv:1702.01983.

59. Averbuch-Elor H. et al. Bringing portraits to life. *ACM Transactions on Graphics*. Nov. 2017. Vol. 36, No. 6. P. 196:1-196:13.

60. Bestagini P. et al.. Local tampering detection in video sequences. *Proceedings of the IEEE International Workshop on Multimedia Signal Processing*. 2013. P. 488-493.

61. Brundage M. et al. The malicious use of artificial intelligence: Forecasting prevention and mitigation. Feb. 2018. arXiv:1802.07228.

62. Conotter V., Bodnari E., Boato G. and Farid H., Physiologically-based detection of computer generated faces in video. *Proceedings of the IEEE International Conference on Image Processing*. Oct. 2014. P. 248-252.

63. Donahue J. et al., Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Apr. 2017. Vol. 39, No. 4, P. 677-691.

64. Farid H. Photo Forensics. *MIT Press Ltd*. 2016.

65. Goodfellow I. et al., Generative adversarial nets. *Advances in Neural Information Processing Systems*. Dec. 2014. P. 2672-2680.

66. Güera D., Yarlagadda S. K.,. Bestagini P, Zhu F.,. Tubaro S and Delp E. J., Reliability map estimation for cnn-based camera model attribution. *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*. Mar. 2018.

67. Hochreiter S. and Schmidhuber J., Long short-term memory. *Neural Computation*. Nov. 1997. Vol. 9, No. 8. P. 1735-1780.

68. Isola P.,. Zhu J. Y, Zhou T. and. Efros A. A, Image-to-image translation with conditional adversarial networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. July 2017. P. 5967-5976.

69. Lample G. et al., Fader networks: Manipulating images by sliding attributes. *Advances in Neural Information Processing Systems*. Dec. 2017. P. 5967-5976.

70. Laptev I., Marszalek M., Schmid C. and Rozenfeld B., Learning realistic human actions from movies. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. June 2008. P. 1-8.

71. Liao Y., Wang Y. and Liu Y., Graph regularized auto-encoders for image representation. *IEEE Transactions on Image Processing*. June 2017. Vol. 26, No. 6. P. 2839-2852.

72. McLaughlin N., Rincon J. M. d. and Miller P., Recurrent convolutional network for video-based person re-identification. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. June 2016. P. 1325-1334.

73. Qian Y. et al., Recurrent color constancy. *Proceedings of the IEEE International Conference on Computer Vision*. Oct. 2017. P. 5459-5467.

74. Raghavendra R., Raja K. B., Venkatesh S. and Busch C., Transferable deep-cnn features for detecting digital and print-scanned morphed face images. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. July 2017. P. 1822-1830.

75. Rahmouni N., Nozick V., Yamagishi J. and Echizen I., Distinguishing computer graphics from natural images using convolution neural networks. *Proceedings of the IEEE Workshop on Information Forensics and Security*. Dec. 2017. P. 1-6.

76. Digital Image Forensics., *New York: Springer*. 2013.

77. Szegedy C. et al., Rethinking the inception architecture for computer vision. *Proeedings of the IEEE Conference on Computer Vision and Pattern Recognition*. June 2016. P. 2818-2826.

78. Tewari A. et al., Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. *Proceedings of the IEEE International Conference on Computer Vision Workshops*. Oct. 2017. P. 1274-1283.

79. Thies J. et al., Face2Face: Real-time face capture and reenactment of rgb videos. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* June 2016. P. 2387-2395.

80. Upchurch P. et al., Deep feature interpolation for image content changes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. July 2017. P. 6090-6099.

81. Zhou P. et al., Two-stream neural networks for tampered face detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. July 2017. P. 1831-1839.

ДОДАТОК А
(обов'язковий)

ЛІСТИНГ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ ДЛЯ ВИЯВЛЕННЯ ДІПФЕЙКІВ

Модуль «Додавання фреймворків».

```
import dlib
import cv2
import os
import re
import json
from pylab import *
from PIL import Image, ImageChops
```

Модуль «Розбиття відео на фрейми».

```
train_data_directory = "./train_sample_videos"

real_directory = "./deepfake/dataset/real/"

fake_directory = "./deepfake/dataset/fake/"



with open(os.path.join(train_data_directory, 'metadata.json'), 'r')
as file:

    data = json.load(file)

list_of_train_data = [f for f in os.listdir(train_data_directory) if
f.endswith('.mp4')]

detector = dlib.get_frontal_face_detector()

for vid in list_of_train_data:

    count = 0

    cap = cv2.VideoCapture(os.path.join(train_data_directory, vid))
```

```
    frameRate = cap.get(5)
    while cap.isOpened():
        frameId = cap.get(1)
        ret, frame = cap.read()
        if ret != True:
            break
        if frameId % ((int(frameRate)+1)*1) == 0:
            face_rects, scores, idx = detector.run(frame, 0)
            for i, d in enumerate(face_rects):
                x1 = d.left()
                y1 = d.top()
                x2 = d.right()
                y2 = d.bottom()
                crop_img = frame[y1:y2, x1:x2]
                if data[vid]['label'] == 'REAL':

cv2.imwrite(real_directory+vid.split('.')[0]+'_'+str(count)+'.png',
cv2.resize(crop_img, (128, 128)))
                elif data[vid]['label'] == 'FAKE':

cv2.imwrite(fake_directory+vid.split('.')[0]+'_'+str(count)+'.png',
cv2.resize(crop_img, (128, 128)))
                count+=1
```

Модуль «Нормалізація та розбиття на дані для тренування моделі та дані для тестів».

```
input_shape = (128, 128, 3)
data_dir = './deepfake/dataset'

real_data = [f for f in os.listdir(data_dir+'/real') if
f.endswith('.png')]
fake_data = [f for f in os.listdir(data_dir+'/fake') if
f.endswith('.png')]

X = []
Y = []

for img in real_data:

X.append(img_to_array(load_img(data_dir+'/real/'+img)).flatten() /
255.0)
    Y.append(1)
for img in fake_data:

X.append(img_to_array(load_img(data_dir+'/fake/'+img)).flatten() /
255.0)
    Y.append(0)

Y_val_org = Y

#Normalization
X = np.array(X)
Y = to_categorical(Y, 2)

#Reshape
X = X.reshape(-1, 128, 128, 3)

#Train-Test split
```

```
    X_train, X_val, Y_train, Y_val = train_test_split(X, Y,
test_size = 0.2, random_state=5)
```

Модуль «Побудова три шарової НМ».

```
from tensorflow.keras.applications import InceptionResNetV2

from tensorflow.keras.layers import Conv2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Dropout

from tensorflow.keras.layers import InputLayer

from tensorflow.keras.layers import GlobalAveragePooling2D

from tensorflow.keras.models import Sequential

from tensorflow.keras.models import Model

from tensorflow.keras import optimizers

from tensorflow.keras.callbacks import ReduceLROnPlateau,
EarlyStopping


googleNet_model = InceptionResNetV2(include_top=False,
weights='imagenet', input_shape=input_shape)

googleNet_model.trainable = True

model = Sequential()

model.add(googleNet_model)

model.add(GlobalAveragePooling2D())

model.add(Dense(units=2, activation='softmax'))

model.compile(loss='binary_crossentropy',

              optimizer=optimizers.Adam(lr=1e-5, beta_1=0.9,
beta_2=0.999, epsilon=None, decay=0.0, amsgrad=False),

              metrics=['accuracy'])
```

```
model.summary()
```

Модуль «Побудова графіків».

```
f, (ax1, ax2) = plt.subplots(1, 2, figsize=(20, 4))

t = f.suptitle('Pre-trained InceptionResNetV2 Transfer Learn with
Fine-Tuning & Image Augmentation Performance ', fontsize=12)

f.subplots_adjust(top=0.85, wspace=0.3)


epoch_list = list(range(1,EPOCHS+1))

ax1.plot(epoch_list, history.history['accuracy'], label='Train
Accuracy')

ax1.plot(epoch_list, history.history['val_accuracy'],
label='Validation Accuracy')

ax1.set_xticks(np.arange(0, EPOCHS+1, 1))

ax1.set_ylabel('Accuracy Value')

ax1.set_xlabel('Epoch #')

ax1.set_title('Accuracy')

l1 = ax1.legend(loc="best")


ax2.plot(epoch_list, history.history['loss'], label='Train Loss')

ax2.plot(epoch_list, history.history['val_loss'], label='Validation
Loss')

ax2.set_xticks(np.arange(0, EPOCHS+1, 1))

ax2.set_ylabel('Loss Value')

ax2.set_xlabel('Epoch #')

ax2.set_title('Loss')

l2 = ax2.legend(loc="best")
```

Модуль «Матриці невідповідностей».

```
def print_confusion_matrix(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    print('True positive = ', cm[0][0])
    print('False positive = ', cm[0][1])
    print('False negative = ', cm[1][0])
    print('True negative = ', cm[1][1])
    print('\n')
    df_cm = pd.DataFrame(cm, range(2), range(2))
    sn.set(font_scale=1.4) # for label size
    sn.heatmap(df_cm, annot=True, annot_kws={"size": 16}) # font
size
    plt.ylabel('Actual label', size = 20)
    plt.xlabel('Predicted label', size = 20)
    plt.xticks(np.arange(2), ['Fake', 'Real'], size = 16)
    plt.yticks(np.arange(2), ['Fake', 'Real'], size = 16)
    plt.ylim([2, 0])
    plt.show()

print_confusion_matrix(Y_val_org, np.argmax(model.predict(X),
axis=1))
```

ДОДАТОК Б
(обов’язковий)

Копія тез доповіді на Всеукраїнській науково-практичній конференції Актуальні Проблеми Комп’ютерних Наук (АПКН-2021)

Дмітрієв Б. В., Яцків В. В. Дослідження технологій виявлення дипфейків // АПКН–2022 (Хмельницький, 18-19 жовтня 2022). С. 109-111

УДК 004.4

Дмітрієв Б.В., Яцків В.В.

Хмельницький національний університет

МЕТОД ТА ПРОГРАМНО-ТЕХНІЧНІ ЗАСОБИ ВИЯВЛЕННЯ ДИПФЕЙКІВ

Розглянуто підходи виявлення фальшивих медіафайлів, створених шляхом зміни наявного відеоматеріалу, який дозволяє точніше та швидше дізнатися реальність переглянутого матеріалу. Результати дослідження можна спрямувати для оптимізації вже готової системи виявлення Діпфейків. Очікується, що метод виявлення зможе розпізнавати найскладніші зміни обличчя чи голосу людини.

The approaches of fake media files created by changing the existing video material, which allows to more accurately and quickly obtain the reality of the viewed material, are considered. The results of the study can be directed to optimize the already ready Deepfake detection system. The method is expected to be able to recognize the most complex changes in a person's face or voice.

В епоху інформаційної ери, майже кожний у світі знайомий з інтернетом та його можливостями, при цьому технології активно розвиваються, зокрема, появляється тенденція до використання штучного інтелекту. Можливості технології (artificial intelligence, AI) немає кінця, адже вона можуть вдосконалюватись та навчатись. Більшість користувачів не розуміють як зловмисники можуть їх обманути за допомогою глибоких нейронних мереж (DNN) для зміни особи людини у відеоматеріалах, що може привести до жахливих наслідків.

Метою роботи є вдосконалення методу з використанням вже готових програмно-технічних засобів для виявлення медіафайлів, які було змінено оманним шляхом за допомогою технології побудови діпфейків.

Діпфейки можуть бути шкідливими, але створити діпфейки, які важко виявити, нелегко. Створення діпфейків сьогодні вимагає використання графічного процесора (GPU). Щоб створити переконливий діпфейк, може бути достатньо графічного процесора ігрового типу вартістю кілька тисяч доларів. Програмне забезпечення для створення діпфейків безкоштовне, з відкритим кодом і легко завантажується. Однак значні навички редагування графіки та дубляжу звуку, необхідні для створення правдоподібного діпфейку, ще не є поширеними. Крім того, робота, необхідна для створення такого глибокого фейку, потребує витрат часу від декількох тижнів до місяців, щоб навчити модель і виправити недоліки. Цей постійний розвиток дозволить зробити глибокі фейки все легшими для менш

досвідчених користувачів, з більшою точністю та більшим потенціалом для створення правдоподібних фейкових медіа [1].

Щоб ефективно виявити фейковий відеоматеріал використовується ряд різних методик, але всі вони мають певні недоліки. Вже важко спостерігати за візуальними змінами та рухами обличчя у поставленому матеріалі, тому виявити фейк майже нереально. Адже розробка обманних відео та аудіо-матеріалів завжди покращується та ускладнює виявлення. Тому найдоцільніше використовувати методи нейронних мереж. Одним із способів, є використання мережі , яка має назву - Siamese neural network (SNN).

Цей клас архітектур нейронних мереж, містить дві або більше ідентичних підмереж. «Ідентичні» тут означає, що вони мають однакову конфігурацію з однаковими параметрами та вагою. Оновлення параметрів відображається в обох підмережах і використовується для пошуку подібності між входами шляхом порівняння векторів ознак. Сіамські мережі використовують лише кілька зображень, щоб отримати кращі прогнози. Здатність навчатися з дуже малою кількістю даних зробила сіамські мережі більш популярними в останні роки.

Традиційно нейронна мережа вчиться передбачати кілька класів. Це створює проблему, коли нам потрібно додати або видалити нові класи до даних. У цьому випадку ми повинні оновити нейронну мережу та перенавчити її на всьому наборі даних. Крім того, глибокі нейронні мережі потребують великого обсягу даних, на яких можна навчатися. SNN, з іншого боку, вивчають функцію подібності. Таким чином, ми можемо навчити SNN бачити, чи збігаються два зображення. Цей процес дозволяє класифікувати нові класи даних без перенавчання мережі [2].

Для реалізації нейронної мережі, потрібно її спершу навчити, для цього використовується така послідовність кроків:

1. Ініціалізуйте мережу, функцію втрати та оптимізатор.
2. Передайте в мережу перше зображення пари.
3. Відправте через мережу друге зображення пари.
4. Обчисліть втрати, використовуючи вихідні дані першого та другого зображень,
5. Зворотне поширення втрат для обчислення градієнтів нашої моделі.
6. Оновіть ваги за допомогою оптимізатора.
7. Зберегти модель.

Дві нейронні мережі є перцептронами прямого зв'язку та використовують зворотне поширення помилок під час навчання; вони працюють паралельно в тандемі та порівнюють свої результати в кінці, як правило, через косинус відстані. Вихід, згенерований виконанням сіамської нейронної мережі, можна вважати семантичною подібністю між прогнозованим представленням двох вхідних векторів [3].

Переглянувши інформацію про нейронну мережу «SNN» можна виділити ряд переваг, але також маємо певну кількість недоліків та ризиків, серед переваг можемо виділити [2, 4]:

1) більш стійка до класового дисбалансу;
2) добре поєднувати з найкращим класифікатором;
3) навчання на семантичній подібності;

Недоліки використання мережі «SNN»:

1) потрібно більше часу на навчання;
2) не виводить ймовірності;

Практичні випадки реального використання сіамських мереж включають розпізнавання обличчя, перевірку підпису, Крім того, сіамські мережі можна навчати з надзвичайно малою кількістю даних, що робить можливими більш просунуті програми, такі як одноразове та багатократне навчання.

Отже, проведений аналіз методу виявлення діпфейків за допомогою нейронних мереж, а саме сіамської мережі «SNN». Подальші дослідження будуть спрямовані для покращення ефективності виявлення фейкових відеоматеріалів, що унеможливить зловмисникам ввести в оману переглядачів відео.

Перелік посилань

1. How Easy Is It to Make and Detect a Deepfake? 2022. https://insights.sei.cmu.edu/blog/how-easy-is-it-to-make-and-detect-a-deepfake/
2. A Friendly Introduction to Siamese Networks. 2022. https://builtin.com/machine-learning/siamese-network
3. Chicco Davide. Siamese neural networks: An overview. Artificial Neural Networks, 2021. P.73-94.
4. Berlemont, S., Lefebvre, G., Duffner, S., & Garcia, C. Class-balanced siamese neural networks. Neurocomputing, 2018, 273. P. 47-56.

ДОДАТОК В
(обов'язковий)

ПРЕЗЕНТАЦІЯ ДОПОВОДІ

Хмельницький національний університет
Факультет програмування та комп'ютерних і телекомунікаційних систем
Кафедра комп'ютерної інженерії та інформаційних систем

МЕТОД ТА ПРОГРАМНО-ТЕХНІЧНІ ЗАСОБИ ВИЯВЛЕННЯ ДИПФЕЙКІВ

Доповідач:
Дмітрієв Богдан
Науковий керівник:
д.т.н.,проф. Яцків В.В.

МЕТА

- Метою кваліфікаційної роботи магістра є розробка методу з використанням програмно-технічних засобів для виявлення медіафайлів, які було змінено за допомогою технології дипфейків.

ОБ’ЄКТ та ПРЕДМЕТ ДОСЛІДЖЕННЯ

- Процес сканування та виявлення відмінностей у відео які створені технологією дипфейк

- методи, програмні та апаратні засоби виявлення дипфейків.

3

НАУКОВА НОВИЗНА

набув подальшого розвитку метод виявлення дипфейків, який базується на навчанні нейронної мережі з використанням набору даних, який містить як фальшиві, так і справжні відеодані при цьому забезпечує ефективне виявлення дипфейків та відкриваючи можливості для подальших досліджень;

набула подальшого розвитку інформаційна технологія штучного інтелекту для тестування та оцінка запропонованого методу надаючи емпіричні докази його ефективності та надійності.

4

ПРАКТИЧНІ ЗНАЧИМІСТЬ

- Дає змогу побудувати програмний додаток, використовуючи сучасну архітектуру згорткової мережі, яка в свою чергу може допомогти судовими лабораторіями та експертами-криміналістами при аналізі відео на наявність їх фальсифікації.

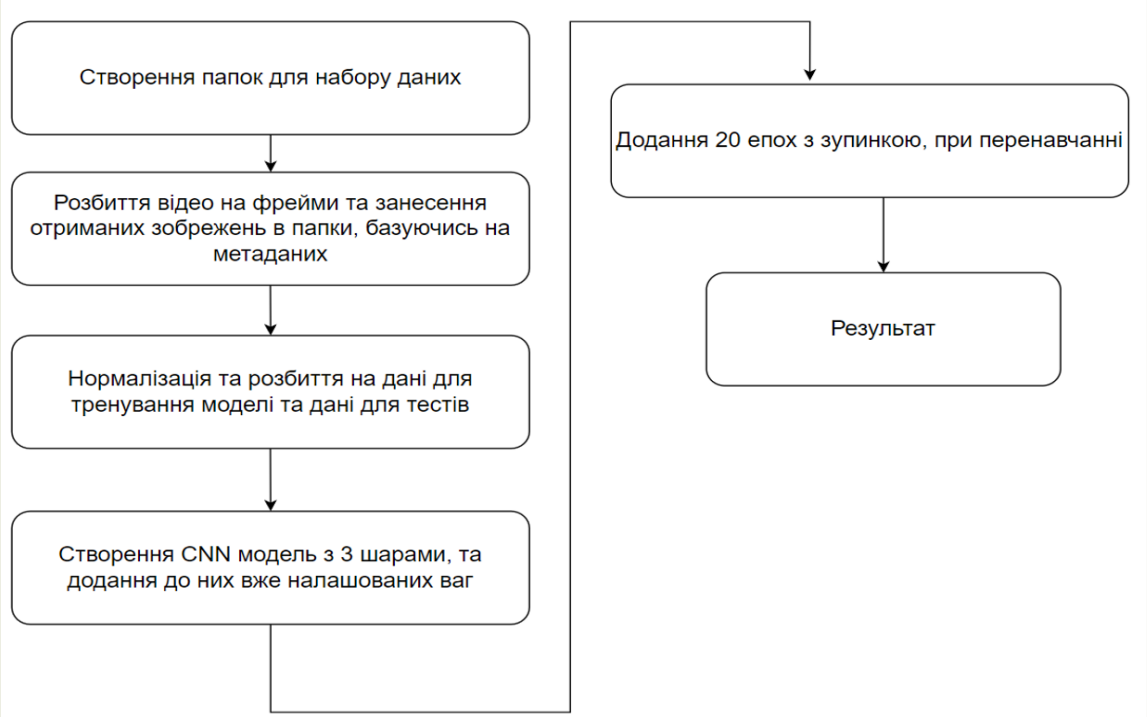
5

МЕТА І ЗАДАЧІ ДОСЛІДЖЕННЯ

- Дослідити існуючі методи виявлення дипфейків.
- Зробити аналіз сучасних прогромно-технічних засобів.
- Розробити метод виявлення фейкових медіафайлів
- Зробити модель НМ, на основі методу
- Реалізувати метод використовуючи набір данних та модель .

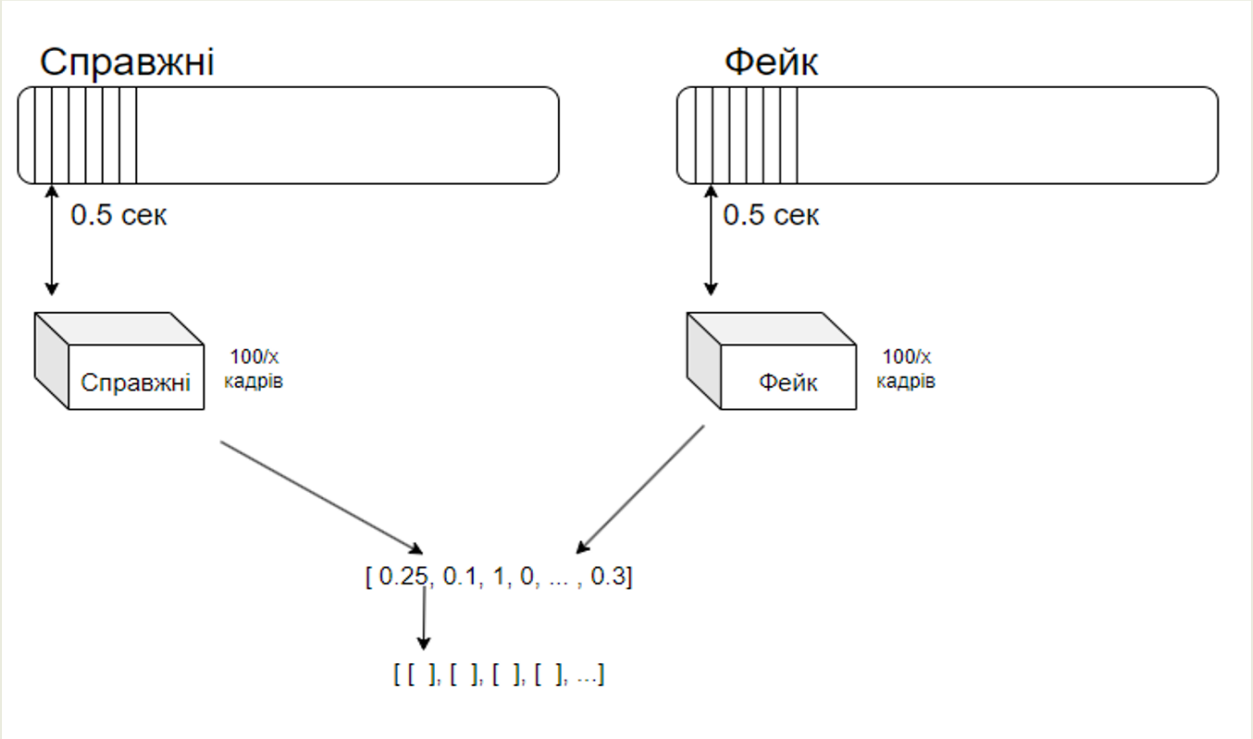
6

РЕАЛІЗАЦІЯ МЕТОДУ ВИЯВЛЕННЯ ДИПФЕЙКІВ

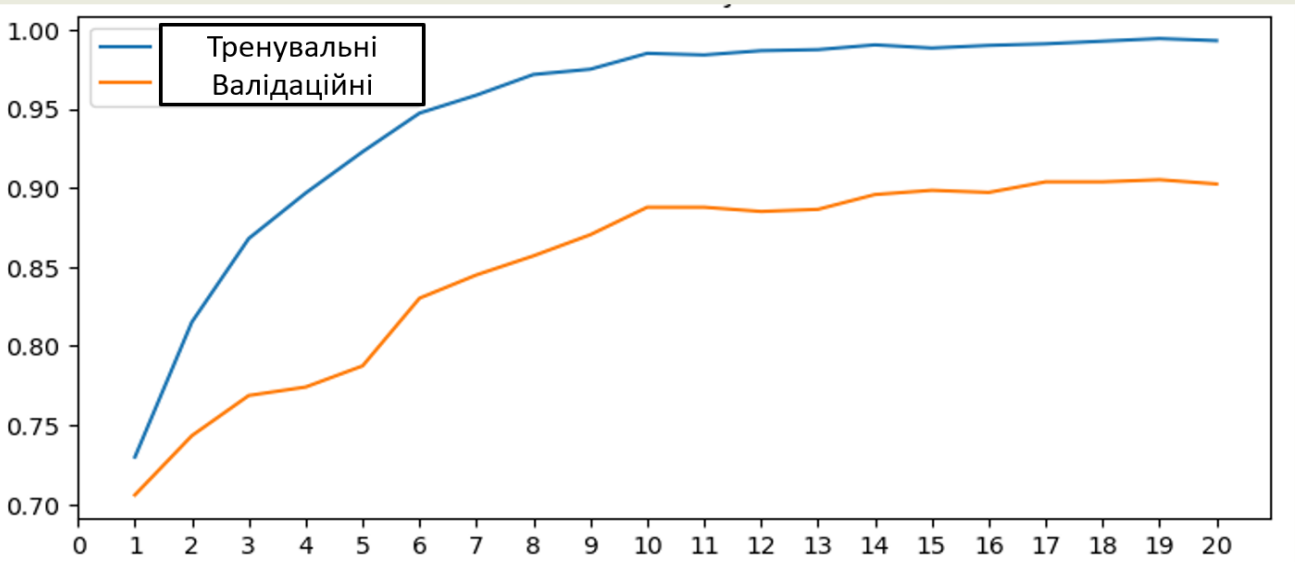


7

СПОСІБ СТВОРЕННЯ ФРЕЙМІВ



ГРАФІК ТОЧНОСТІ



9

ВИСНОВКИ

- В першому розділі було дано визначення поняття штучного інтелекту, як він працює та що він утворює. Також проаналізовано будуву мережі GAN, та нейронної мережі, адже це головне в побудові та виявлення фейкових відео.

- В другому розділі описано засоби що будуть використовуватися при реалізації власної системи. Описано різні засоби, розкрито основні характеристики та функціонал. Розглянуто фреймворки, які необхідні для розробки системи та розкрито функціонал, який вони можуть реалізувати.

10

ВИСНОВКИ

- В третьому розділі досягнута основна мета роботи, а саме розроблено метод виявлення дипфейків. Описано всі необхідні елементами та функції. Вибрали спосіб оптимізації, який покращив роботу методу. Розроблено програмно-технічний засіб, в якому реалізовано розроблений метод.

- В четвертому розділі проведено дослідження методу виявлення дипфейків. Розроблено альтернативний метод НМ для перевірки роботи, який був простіший в роботі та використовував менше епох перевірки. Проведено тестування цих моделей, з використання НД, які підібрані для нього, тобто тестові та валідаційні дані. Також створена матриця невідповідностей.

11

НАЯВНІ ПУБЛІКАЦІЇ

За темою кваліфікаційної роботи опублікована одна теза у збірнику матеріалів конференції «Збірника наукових праць Конференції АПКН-2022»

12

ДЯКУЮ ЗА УВАГУ!

UNICHECK

Ім'я користувача:
Кафедра КІ

ID перевірки:
1014947924

Дата перевірки:
06.05.2023 10:32:53 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
06.05.2023 10:33:12 EEST

ID користувача:
100005591

Назва документа: **Дмітрієв_Метод та програмно-технічні засоби виявлення дипфейків**

Кількість сторінок: **85** Кількість слів: **17264** Кількість символів: **132512** Розмір файлу: **1.90 MB** ID файлу: **1014641116**

1.69%
Схожість

Найбільша схожість: **0.54%** з джерелом з Бібліотеки (ID файлу: **1011053481**)

1.31% Джерела з Інтернету	79	Сторінка 87
0.93% Джерела з Бібліотеки	103	Сторінка 87

0.09% Цитат

Цитати	3	Сторінка 88
Посилання	1	Сторінка 88

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	8

Sat May 06 09:39:22 EEST 2023, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 1.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

	Документ		Сумарний збіг по Базі Даних	
ID: 113065 Назва: МКР Метод та програмно-технічні засоби виявлення дипфейків Додано в БД: 2023-05-06 Автора: Дмітрієв Б.В. Керівники: Яцків В.В. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	115047	999	1534 (1%)	12 (1%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Дмітрієв Богдан Васильович

Тема: Метод та програмно-технічні засоби виявлення дипфейків.

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг дипломної роботи:

Кількість листів креслень – ; кількість сторінок записки 102

1. Короткий зміст роботи та прийнятих рішень У роботі запропоновано метод та програмно-технічний засіб виявлення дипфейків

2. Висновок про відповідність роботи дипломному завданню Кваліфікаційна робота магістра відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі дано визначення поняття штучного інтелекту, як він працює та його застосування. Також проаналізовано будову мережі GAN, та нейронної мережі, адже це головне в побудові та виявлення фейкових відео. У другому розділі описано засоби що будуть використовуватися при реалізації власної системи. Описано різні засоби, розкрито основні характеристики та функціонал. У третьому розділі досягнута основна мета роботи, а саме розроблено метод виявлення дипфейків. Описано всі необхідні елементами та функції. Вибрали спосіб оптимізації, який покращив роботу методу.. У четвертому розділі проведено дослідження методу виявлення дипфейків, розроблено альтернативний метод НМ для перевірки роботи.

4. Позитивні сторони роботи: Запропонований метод виявлення дипфейків які є відео- матеріалами, який оптимізований під великі обсяги набору даних. Створено засіб який реалізовує цей метод та модель нейронної мережі, що в свою

чергу зменшить час пошуку фальсифікацій та підвищить їх точність виявлення.

5. Негативні сторони роботи: В роботі не в повній мірі досліджено методи виявлення дипфейків, також не досконало реалізовано програмно-технічний засіб, щоб наглядно показати роботу конкретними відеоматеріалами.

6. Оцінка графічного оформлення та пояснювальної записки роботи: –

7. Відгук про роботу в цілому: В загальному робота виконана на нормальному рівні. Весь матеріал кваліфікаційної роботи структурований, чіткий та послідовний. Усі розділи роботи йдуть у вірній послідовності, що дозволяє чітко розуміти викладений матеріал в рамках даної кваліфікаційної роботи.

8. Інші зауваження: –

9. Оцінка дипломної роботи:

Розглянувши позитивні та негативні сторони представленої дипломної роботи вважаю, що робота заслуговує оцінки «добре» 4,0 (С)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) доктор технічних наук, професор, зав. кафедри комп'ютерних наук Бармак Олександр Володимирович.

"10" травня 2023р.

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Дмітрієв Богдан Васильович
ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-21-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

10 травня 2023року
дата

підпис

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Метод та програмно-технічні засоби виявлення дипфейків

Автор: Дмітрієв Богдан Васильович

Спеціальність: 123 – Компютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: Яцків Василь Васильович, д.т.н, професор

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих рішень, які не описують безпосередньо авторське дослідження, а тільки слугують як приклад та початок створення роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) деякими виявленими збігами системи, були використані джерела, які розпізнали як плагіат;

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 1.69% і адресується до 182 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ В. В. Яцків

Гарант ОП _____ О. С. Савенко

Завідувач кафедри КІПСч _____ Т. О. Говорущенко