

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки**

До захисту допущено
Завідувач кафедри
_____ Дмитро ЛАНДЕ
(підпис)

24 червня 2025 р.

**Дипломна робота
на здобуття ступеня бакалавра
за освітньо-професійною програмою «Системи, технології та математичні
методи кібербезпеки»
спеціальності 125 «Кібербезпека»**

на тему: Методика виявлення фейкових зображень в задачах безпеки
кібернетичного простору

Виконав (-ла): здобувач вищої освіти **IV** курсу, групи ФБ-12
(шифр групи)

Козак Владислав Миколайович
(прізвище, ім'я, по батькові) (підпис)

Керівник к. ф.-м. н., ст. викл. каф. ІБ Рибак Олександр Владиславович
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент к. ф.-м. н., доцент каф. ММЗІ, Хмельницький М. О.
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ – 2025 року

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки**

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 125 «Кібербезпека»

Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Дмитро ЛАНДЕ
(підпис)

24 червня 2025 р.

**ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти**

Козаку Владиславу Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методика виявлення фейкових зображень в задачах безпеки кібернетичного простору,

керівник роботи Рибак Олександр Владиславович, к. ф-м. н., ст. викл. каф. ІБ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «26» травня 2025 р. № 1761-с

2. Термін подання роботи здобувачем вищої освіти «17» червня 2025 р.

3. Вихідні дані до роботи

Попередні дослідження та літературні джерела за темою роботи.

4. Зміст роботи

Теоретичні основи фейкових зображень, їх види, методи створення та вплив на кіберпростір, аналіз сучасних методів виявлення фальсифікованих зображень, зокрема аналіз рівня помилок та використання нейронних мереж, розробка методики виявлення фейкових зображень на основі комбінації ELA та згорткових нейронних мереж, реалізація програмного забезпечення для автоматизованого аналізу зображень.

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація.

6. Дата видачі завдання: 20 травня 2025 року.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Отримання завдання для дипломної роботи	20.05.2025	Виконано
2	Огляд наукових джерел	21.05.2025 – 27.05.2025	Виконано
3	Написання першого розділу	28.05.2025 – 02.06.2025	Виконано
4	Написання другого розділу	03.06.2025 – 07.06.2025	Виконано
5	Написання третього розділу	08.06.2025 – 12.06.2025	Виконано
6	Оформлення дипломної роботи	13.06.2025 – 14.06.2025	Виконано
7	Створення презентації для передзахисту	15.06.2025	Виконано
8	Передзахист	17.06.2025	Виконано
9	Внесення фінальних правок	18.06.2025-22.06.2025	Виконано
10	Захист	24.06.2025	

Здобувач вищої освіти

(підпис)

Владислав КОЗАК
(Власне ім'я, ПРІЗВИЩЕ)

Керівник роботи

(підпис)

Олександр РИБАК
(Власне ім'я, ПРІЗВИЩЕ)

РЕФЕРАТ

Обсяг роботи 60 сторінок, 22 ілюстрації, 10 таблиць, 3 додатки, 10 джерел літератури.

Об'єкт дослідження: Цифрові зображення, що можуть бути піддані маніпуляціям з метою створення фейкових візуальних матеріалів.

Предмет дослідження: Методика виявлення фейкових зображень на основі використання згорткових нейронних мереж у поєднанні з аналізом рівня похибки.

Мета дослідження: Розробка та реалізація методики виявлення фейкових зображень із використанням аналізу рівня помилки та застосування алгоритмів машинного навчання.

Методи дослідження: Аналіз наукових джерел, вивчення методів обробки зображень, методи глибокого навчання, експериментальне дослідження ефективності.

Отримані результати: Розроблено методику виявлення фейкових зображень, що поєднує аналіз рівня помилок з глибоким навчанням на основі згорткових нейронних мереж, що забезпечує високу точність класифікації. Реалізовано програмний комплекс, який містить модуль попередньої обробки зображень з алгоритмом аналізу рівня похибки, архітектуру згорткової нейронної мережі, систему навчання та валідації моделі, інтерфейс для класифікації нових зображень.

Ключові слова: фейкові зображення, аналіз рівня похибки, згорткова нейронна мережа.

ABSTRACT

The volume of the thesis is 60 pages, 22 illustrations, 10 tables, 3 appendices, 10 sources of literature.

Research object: digital images that can be manipulated to create fake visual materials.

The subject of the research: methodology for detecting fake images based on the use of convolutional neural networks in combination with error level analysis.

The purpose of the research: development and implementation of a method for detecting fake images using error level analysis and the application of machine learning algorithms.

Research methods: analysis of scientific sources, study of image processing methods, deep learning methods, experimental study of effectiveness.

Obtained results: a method for detecting fake images has been developed that combines error rate analysis with deep learning based on convolutional neural networks, which ensures high classification accuracy. A software package has been implemented that includes an image preprocessing module with an error level analysis algorithm, a convolutional neural network architecture, a model training and validation system, and an interface for classifying new images.

Keywords: fake images, error level analysis, convolutional neural network.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	8
ВСТУП.....	9
1 АНАЛІЗ СТАНУ ПРОБЛЕМИ ТА СУЧАСНИХ МЕТОДІВ ВИЯВЛЕННЯ ФЕЙКОВИХ ЗОБРАЖЕНЬ	11
1.1 Поняття фейкових зображень і їх роль у кіберпросторі.....	11
1.2 Методи створення фейкових зображень	13
1.3 Огляд сучасних методів і засобів виявлення фальсифікованих зображень	14
1.4 Проблематика виявлення фейкових зображень	19
Висновки до розділу 1	21
2 ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ МЕТОДИКИ ВИЯВЛЕННЯ ПІДРОБЛЕНИХ ЗОБРАЖЕНЬ.....	22
2.1 Основи комп'ютерного зору та обробки зображень.....	22
2.2 Застосування методу Error Level Analysis	23
2.3 Використання нейронних мереж для аналізу маніпуляцій із зображеннями.....	24
2.4 Архітектура методики виявлення фейкових зображень на основі ELA та CNN	26
Висновки до розділу 2	30
3 РОЗРОБКА І РЕАЛІЗАЦІЯ МЕТОДИКИ ВИЯВЛЕННЯ ФЕЙКОВИХ ЗОБРАЖЕНЬ	31
3.1 Вибір середовища розробки та інструментів	31
3.2 Вибір датасету та підготовка зображень	32
3.3 Реалізація алгоритму ELA.....	33
3.4 Створення моделі на основі CNN	34
3.5 Навчання та тестування моделі, побудова метрик.....	39
3.6 Використання методики для виявлення згенерованих зображень.....	49
3.7 Використання моделі для класифікації зображень	50

	7
Висновки до розділу 3	52
ВИСНОВКИ	53
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ	54
ДОДАТОК А PYTHON 3 КОД ORGINIZE_DATA.PY	55
ДОДАТОК Б PYTHON 3 КОД TRAIN_MODEL.PY	56
ДОДАТОК В PYTHON 3 КОД PREDICT_ELA.PY	59

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ELA (Error Level Analysis) – аналіз артефактів стиснення в цифрових даних з втратним стисненням.

EXIF (Exchangeable image file format) – стандарт, що визначає формат опису допоміжної метаінформації для файлів.

RGB (Red, Green, Blue) – колірна модель, яка описує спосіб синтезу кольору.

JPEG (Joint Photographic Experts Group) – формат цифрових зображень, що використовує стиснення з втратою якості.

CNN (Convolutional Neural Network) – згорткова нейронна мережа, клас глибоких штучних нейронних прямого поширення.

ВСТУП

Актуальність роботи:

У сучасних умовах швидкого розвитку цифрових технологій, візуальні дані стали одним із основних носієм інформації. Із вдосконаленням методів обробки зображень також зростає і їх підроблена кількість. Фейкові зображення активно використовуються для вчинення кіберзлочинів, проведення атак соціальної інженерії та поширення паніки серед громадян.

У контексті кібербезпеки здатність автоматично виявляти фейкові зображення є дуже важливою. Виявлення відредагованої певним чином візуальної інформації може сприяти попередженню кіберзлочинів, допомагати у розслідуванні інцидентів та захисті інформаційної інфраструктури.

Тому є необхідність розробити ефективну методику виявлення фейкових зображень, яка здатна класифікувати зображення за ознаками автентичності.

Мета дослідження: розробка та реалізація методики виявлення фейкових зображень із використанням аналізу рівня помилки та застосування алгоритмів машинного навчання.

Завдання дослідження:

1. Проаналізувати сучасні методи виявлення фейкових зображень.
2. Розробити методику для виявлення фейкових зображень.
3. Провести експериментальне дослідження ефективності методики.

Об'єкт дослідження: цифрові зображення, що можуть бути піддані маніпуляціям з метою створення фейкових візуальних матеріалів.

Предмет дослідження: методика виявлення фейкових зображень на основі використання згорткових нейронних мереж у поєднанні з аналізом рівня похибки.

Методи дослідження: аналіз наукових джерел, вивчення методів обробки зображень, методи глибокого навчання, експериментальне дослідження ефективності.

Практичне значення одержаних результатів: Розроблена методика може бути використана для створення інструментів перевірки цифрових зображень у ЗМІ, соціальних мережах, сферах кібербезпеки, а також у системах автоматизованого моніторингу контенту. Методика здатна посилити захист інформаційного простору та знизити ризики поширення фальсифікованих візуальних матеріалів.

1 АНАЛІЗ СТАНУ ПРОБЛЕМИ ТА СУЧАСНИХ МЕТОДІВ ВИЯВЛЕННЯ ФЕЙКОВИХ ЗОБРАЖЕНЬ

1.1 Поняття фейкових зображень і їх роль у кіберпросторі

Фейкові зображення – це цифрові матеріали, які були свідомо модифіковані або синтезовані з метою введення в оману. До них належать як редаговані зображення, наприклад, з використанням графічних редакторів, як Photoshop, так і повністю створенні з нуля синтетично за допомогою моделей штучного інтелекту.

Фальсифікація цифрових зображень стає все більш доступною через швидкий розвиток технологій редагування. Зображення можуть бути легко модифіковані, але при цьому візуально ніяким чином не відрізнитись від оригіналу.

Завдяки значному вдосконаленню обчислювальних та мережевих технологій, за останні кілька років спостерігається значне зростання доступності, удосконалення та передачі цифрових зображень за допомогою таких технологій обробки зображень, як цифрові камери, сканери, програми для редагування фотографій та програмні пакети. Однак ця технологія також використовується для маніпулювання цифровими зображеннями та створення підробок, які важко відрізнити від автентичних фотографій. Таким чином, проблема встановлення автентичності зображення стала більш складною через легку доступність цифрових зображень та безкоштовне завантаження програмного забезпечення для редагування, що призводить до зменшення довіри до цифрових фотографій.

Серед найпоширеніших типів підроблення зображень виділяють такі техніки, як злиття, копіювання-переміщення та видалення об'єктів. Злиттям називають вставку фрагмента, скопійованого з іншого зображення, тоді як копіювання-переміщення – це дублювання об'єкта в межах того самого зображення. Видалення означає вилучення певного об'єкта із зображення з

подальшим розширенням навколишнього фону для приховування слідів втручання.

Підробка зображень зазвичай здійснюється з метою приховати певні об'єкти на зображенні, щоб створити хибний доказ або створити зображення приємнішим для зовнішнього вигляду.

Вплив фейкових зображень в різних галузях суспільства:

- У галузі медицини звіти пацієнтів є конфіденційними та завжди повинні бути автентичними. Медичні зображення у більшості випадків створюються як доказ нездоров'я або заяву про хворобу. Оскільки зазвичай такі зображення можуть бути пов'язані з великими сумами грошей, людей може спокусити їх фальсифікація з метою отримання виплати за медичне страхування. Також медичні зображення часто подаються як докази у суді для пом'якшення покарання або уникнення відповідальності. Тому фальсифікація медичних зображень становить серйозну загрозу безпеці та правовому захисту громадян.
- У сфері освіти, підробка зображень призводить до поширення недостовірної інформації, що спричиняє передачу хибних даних до навчальних установ. Досить часто студенти вдаються до підробки власних документів для отримання особистої вигоди. Такі дії підривають безпеку управління в освітніх закладах.
- У сфері електронної комерції більшість транзакцій здійснюється через Інтернет, будь то грошові перекази, покупки, оплата рахунків тощо. У цьому випадку безпека даних клієнтів є головним пріоритетом. Але багато неавторизованих користувачів маніпулюють даними, що призводить до серйозних злочинів. Тому в епоху цифрових технологій підробка зображень є серйозною загрозою.

Результати експериментального дослідження Nameleers та ін. [1], проведеного серед 1404 жителів США, показали, що використання мультимодальної дезінформації (текст у поєднанні з зображенням) виявилось

значно переконливішим, ніж суто текстова. Було встановлено, що така дезінформація сприймається як на 10-15% достовірніша.

Це свідчить про те, що фейкові зображення можуть посилювати вплив дезінформації, створюючи ілюзію правдивості навіть у випадках повністю сфабрикованого контексту.

1.2 Методи створення фейкових зображень

Існує безліч способів редагування зображень за допомогою різних інструментів і технологій, серед них є як і традиційні методи ручного редагування, так і більш сучасні підходи на основі штучного інтелекту. Основними можна виділити використання графічних редакторів, генеративних моделей та спеціалізованих програм для створення дипфейків.

Таблиця 1.1 – Основні методи створення фейкових зображень

Метод	Опис	Приклад інструментів та технологій
Графічне редагування	Ручне або частково автоматизоване редагування зображення для зміни його змісту.	Photoshop, GIMP
Копіювання-переміщення	Вставлення фрагмента зображення в інше місце цього ж зображення	Photoshop, Paint.NET
Злиття зображень	Об'єднання частин з різних зображень в одне	Photoshop, Deep Image Fusion
Видалення об'єктів	Вилучення об'єктів із фону з подальшим заповненням області	Content-Aware Fill

Кінець таблиці 1.1

Дипфейки (Deepfakes)	Заміна або синтез зображення людини на фото чи відео з використанням глибокого навчання.	DeepFaceLab, FaceSwap
Генеративні моделі	Генерація повністю нових зображень на основі моделей штучного інтелекту	GANs, DALL-E

В таблиці 1.1 видно, що значна частина фейкових зображень створюється за допомогою традиційних засобів графічного редагування. Незважаючи на активний розвиток автоматизованих методів синтезу зображень, ручне редагування все ще залишається поширеним способом фальсифікації зображень. Особливу небезпеку становлять саме професійно відредаговані зображення, які складно відрізнити від оригінальних без спеціальних інструментів. Такі зображення часто не мають очевидних артефактів, а зміни можуть бути настільки дрібними, що помітити їх без глибокого аналізу неможливо.

1.3 Огляд сучасних методів і засобів виявлення фальсифікованих зображень

Виявлення фальсифікацій на зображеннях ґрунтується на аналізі ознак, які з'являються в наслідок ручного або програмного втручання в структуру файлу. Сучасні підходи до детектування можна поділити на дві категорії: аналітичні методи, які вивчають зміст і технічні характеристики зображення, і автоматизовані, що базуються на алгоритмах машинного навчання.

Один із найпростіших аналітичних методів – це аналіз метаданих. Більшість цифрових фотографій має EXIF-дані, які містять технічну інформацію про умови створення зображення: модель камери, геолокація, дата і час зйомки,

а також програмне забезпечення за допомогою якого файл було змінено чи збережено. Цей тип аналізу дозволяє виявити перші ознаки втручання в зображення. Якщо у фотографії відсутня інформація про модель, або ж запис про редагування у якомусь графічному редакторі, це може свідчити про можливу підробку. Нижче наведений приклад полів для аналізу метаданих (рис. 1.1)

```

Date/Time Original      : 2008:10:22 16:28:39
Create Date            : 2008:10:22 16:28:39
Modify Date            : 2008:11:01 21:15:07
Orientation             : Horizontal (normal)
ISO                    : 64
Image Height           : 480
Image Width            : 640
Software               : Nikon Transfer 1.1 W
Make                   : NIKON
Camera Model Name      : COOLPIX P6000
GPS Latitude           : 43 deg 28' 2.81" N
GPS Longitude          : 11 deg 53' 6.46" E

```

Рисунок 1.1 – EXIF-дані зображення

Перевагою аналізу метаданих є його простота і доступність – існує безліч інструментів, які дозволяють швидко переглянути вбудовану інформацію. Проте основним недоліком цього методу є те, що метадані можна легко видалити або ж змінити за допомогою будь-якого редактора. Також, соціальні мережі часто автоматично видаляють або змінюють EXIF-інформацію під час завантаження, що ускладнює її використання як доказ автентичності (рис. 1.2)

Крім того, наявність або відсутність певного запису в метаданих не завжди прямо свідчить про фальсифікацію – вона лише вказує на можливість редагування, тому цей метод зазвичай використовується у поєднанні з іншими підходами.

Social Network	Photo (Mobile)	Video (Mobile)	Photo (Web)	Video (Web)
Discord	✓	N/A	✓	✓
GitHub (Comments)	N/A	N/A	✗	N/A
GitLab (Comments)	N/A	N/A	✓	N/A
Google Chat	✓	✗	✗	✗
Imgur	✓	N/A	✓	✓
LinkedIn	✓	✓	✓	✓
Medium	✓	N/A	✓	N/A
Microsoft Teams	✓	✗	✗	✗
Reddit	✓	N/A	✓	N/A
Signal	✓	✓	N/A	N/A
Slack	✓	✗	✓	✗
Strava	✓	N/A	✓	N/A
Telegram	✓	✓	✗	✗
Tumblr	✓	✓	✓	✓
Twitter	✓	✓	✓	✓
WhatsApp	✓	✓	✓	✗
Wire	✗	✗	✗	✗
Yammer	✓	✗	✗	✗

Рисунок 1.2 – Соціальні мережі, що вилучають EXIF-дані [2]

Ще одним із аналітичних методів детекції фальсифікацій є аналіз цифрового шуму. Цифрові зображення, створені камерами, завжди містять певний рівень шуму – це природне явище, яке виникає через фізичні обмеження сенсорів. Особливо шум помітний у фотографіях, які були зроблені при поганому освітленні або з високим ISO. Цей шум має випадковий, але передбачуваний характер і є рівномірно розподіленим по всьому зображенню.

Стабільність розподілу шуму дозволяє використовувати його як індикатор справжності зображення. У не редагованому зображенню шум буде відповідати

загальній структурі сцени. На фальсифікованому ж зображенню, характеристики шуму в редагованих зонах можуть змінюватись.

Окрім аналізу метаданих та цифрового шуму, які переважно спираються на технічні характеристики зображення, існують й інші методи, здатні виявляти фальсифікації на основі змін у структурі самого файлу. Одним із таких методів є аналіз рівня похибки (ELA) – візуальний підхід, який дозволяє виявляти локальні зміни, внесені під час редагування зображення (рис. 1.3)

Суть методу полягає у вивченні відмінностей у рівні стиснення різних ділянок зображення. При використанні стиснення з втратами зазвичай застосовується рівномірно до набору даних, таких як зображення, що призводить до рівномірного рівня артефактів стиснення [3].

Альтернативно, дані можуть складатися з частин з різними рівнями артефактів стиснення. Ця різниця може виникати через те, що різні частини неодноразово піддавалися одному й тому ж стисненню з втратами різну кількість разів, або через те, що різні частини піддавалися різним видам стиснення з втратами. Різниця в рівні артефактів стиснення в різних частинах даних може свідчити про те, що дані були відредаговані [3].

Щоб зробити типово слабкі артефакти стиснення більш помітними, дані, що підлягають аналізу, піддаються додатковому раунду стиснення з втратами, цього разу на відомому, рівномірному рівні, і результат віднімається від вихідних даних, що досліджуються. Отримане різницеве зображення потім перевіряється вручну на наявність будь-яких варіацій у рівні артефактів стиснення [3].

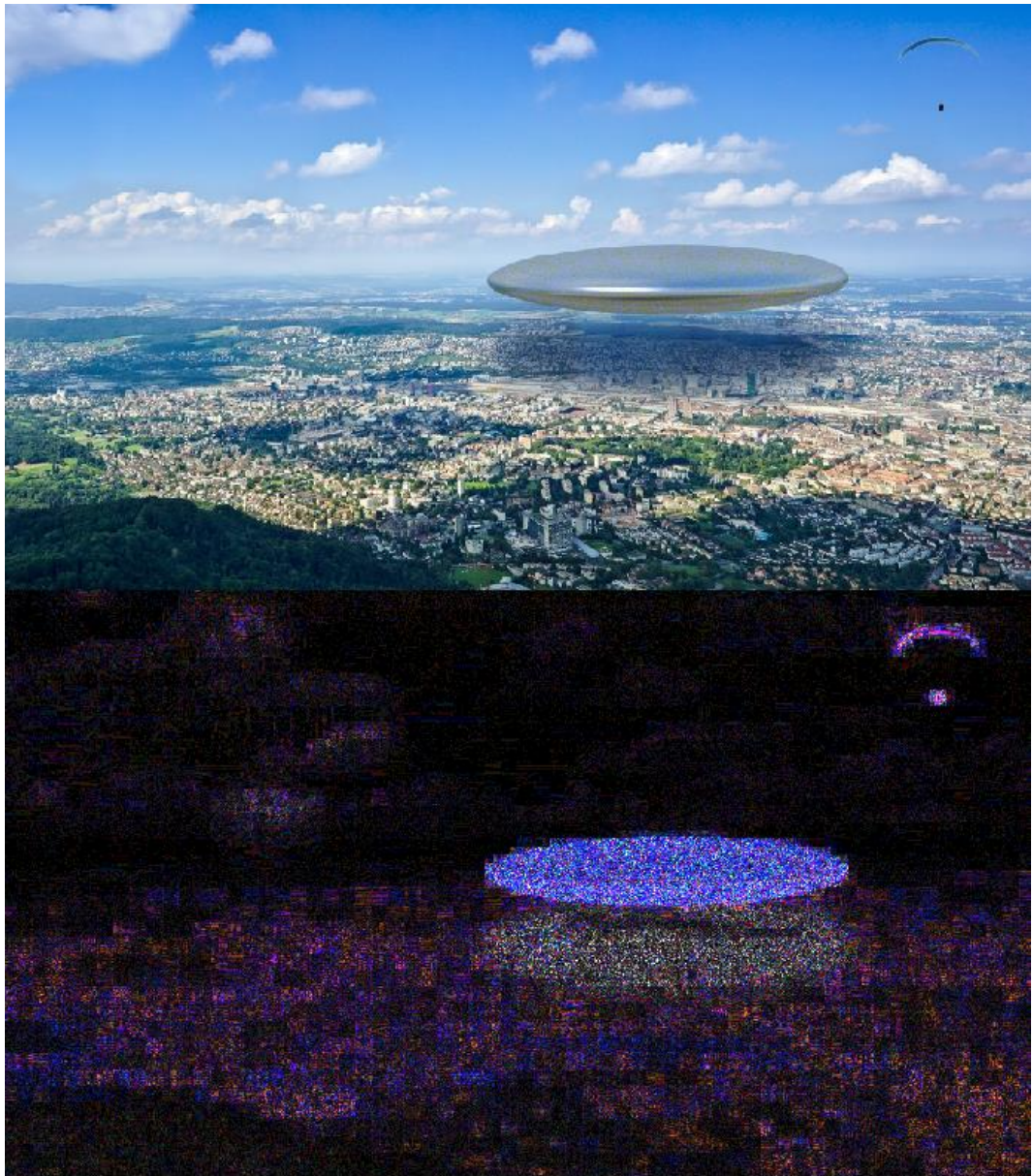


Рисунок 1.3 – Приклад виявлення змінених ділянок зображення за допомогою ELA

На сьогодні існує багато популярних програм та застосунків, які розроблені спеціально для автоматичного виявлення фейкових зображень, включно з дипфейками. Ці інструменти ґрунтуються на алгоритмах машинного навчання. В таблиці 1.2 наведено порівняльну характеристику найвідоміших засобів.

Таблиця 1.2 – Засоби автоматичного виявлення фейкових зображень

Засіб	Опис роботи	Переваги	Недоліки
Deerware Scanner	Використовує штучний інтелект для аналізу зображень і відео на наявність фальсифікацій.	Простий у використанні, підтримка мобільних пристроїв.	Може давати хибні спрацювання, залежить від якості вхідного зображення.
Hive Moderation	AI-платформа для виявлення підробок, включно з deepfake, за допомогою глибоких згорткових мереж.	Працює в реальному часі, API для інтеграції, підтримує відео та зображення.	Платна модель, вимагає підключення до інтернету, орієнтована на бізнес.
DeepFake-o-meter	Онлайн-сервіс для перевірки зображень на наявність deepfake через порівняння з відомими шаблонами.	Не потребує встановлення, доступний з браузера, простий інтерфейс.	Обмежена база даних, не гарантує високої точності для нових типів фейків.

1.4 Проблематика виявлення фейкових зображень

Не зважаючи на велику наявність інструментів та методів для виявлення фейкових зображень, у цій сфері все ще є дуже багато серйозних проблем. Редагування зображень може бути виконано настільки ретельно, що візуально помітити фальсифікацію майже не можливо. Традиційні методи детектування, що базуються на аналізі метаданих чи перевірки шумових аномалій часто є неефективними, особливо коли зловмисник попередньо захищає зображення від аналізу. Ще однією проблемою є відсутність універсального підходу. Різні типи

редагування на зображенні вимагають різних технік виявлення. Методи, що працюють на одному типі, можуть не спрацьовувати на інших.

Попри наявність автоматичних рішень детектувань фейкових зображень, більшість з них зосереджені саме на виявленні дипфейків, залишаючи поза увагою інші ручні типи підробок. Тому є необхідність у застосуванні більш гнучких автоматизованих методів аналізу зображень, які здатні виявляти навіть незначні втручання.

Висновки до розділу 1

У першому розділі було розглянуто природу фейкових зображень, їх вплив на сучасний інформаційний простір та роль у кіберзагрозах. Проаналізовано основні методи їх створення, зокрема з використанням програмних засобів редагування.

У результаті проведеного аналізу встановлено, що проблема розпізнавання фейкових зображень є актуальною в умовах зростання кількості фейкових матеріалів у цифровому середовищі.

Було розглянуто сучасні методи виявлення фальсифікацій, серед яких особливу увагу приділено підходам, що базуються на виявленні цифрових артефактів. Визначено переваги та недоліки методів, таких як аналіз рівня помилок (ELA), перевірка метаданих та аналіз цифрового шуму.

2 ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ МЕТОДИКИ ВИЯВЛЕННЯ ПІДРОБЛЕНИХ ЗОБРАЖЕНЬ

2.1 Основи комп'ютерного зору та обробки зображень

Комп'ютерний зір — це міждисциплінарна галузь, у якій вивчається те, як комп'ютери можуть отримувати високий рівень розуміння з цифрових зображень або відео. З точки зору інженерії, ця галузь прагне автоматизувати завдання, які може виконувати людська зорова система [4].

Комп'ютерний зір займається автоматичним вилученням, аналізом та розумінням корисної інформації з одного зображення або послідовності зображень. Він передбачає розробку теоретичної та алгоритмічної основи для досягнення автоматичного візуального розуміння [4].

Однією із найпопулярніших бібліотек комп'ютерного зору та обробки зображень є OpenCV (Open Source Computer Vision Library). Це відкрита бібліотека, яка написана на C++ з підтримкою Python.

OpenCV дозволяє виконувати наступні операції:

- Зчитування та відображення зображень та відео.
- Перетворення зображень у різні кольорові простори.
- Робота з фільтрами, трансформаціями.
- Застосування алгоритмів для аналізу форми та структури зображень.

Для зчитування зображення з файлу OpenCV надає функцію `cv2.imread()`. Шлях до файлу зображення передається як параметр функції. Метод повертає зображення у форматі BGR (синій, зелений, червоний), який відрізняється від більш поширеного формату RGB. Ця відмінність є важливою при роботі безпосередньо з даними зображення [5].

Трансформація зображень є важливим аспектом обробки зображень, який включає зміну геометрії зображення для досягнення бажаних ефектів або вилучення корисної інформації. Ці операції допомагають підготувати зображення до подальшої обробки та аналізу [5].

Одна з найпростіших трансформацій зображень це зміна розміру. Воно може бути вирішальним для зменшення обчислювального навантаження, необхідного для обробки великих зображень або для пристосування зображень до певних вимог до розміру. OpenCV надає для цієї мети функцію `cv2.resize()`, яка дозволяє вказати нові розміри зображення або коефіцієнт масштабування [5].

2.2 Застосування методу Error Level Analysis

Error Level Analysis – це корисна техніка у цифровій криміналістиці для виявлення маніпуляцій над зображеннями. ELA працює шляхом порівняння рівнів помилок, які з’являються на зображенні під час процесу стиснення. Алгоритм JPEG поділяє зображення на блоки і кожен з блоків стискається з різним рівнем помилки. ELA виявляє ці відмінності у стисканні та виділяє частини зображення які були змінені, висвітлюючи потенційне редагування.

Відмінності виділяються шляхом порівняння оригінального зображення та його повторно збереженою версію. Друга версія стискається з подібним рівнем помилок що й перша. Відмінності між цими версіями показує рівень помилок який був створений під час процесу стискання зображення. В областях, які не були змінені, рівень помилок малий, коли у редагованих областях він великий.

ELA також використовує техніку кольорового відображення для виділення областей зображення, які були змінені. Зазвичай, незмінені області зображення виділяються синім кольором, а області з вищим рівнем помилок – червоним. Ця кольорова карта дозволяє швидко ідентифікувати області зображення, які були підроблені.

Відображення рівня помилок залежить від вибраної якості під час стискання, тому важливо підібрати правильне значення (рис. 2.1). JPEG стискає зображення, виконуючи наступні етапи:

- Трансформація колірного простору (перетворення зображення з RGB у YCbCr, тобто розділення яскравості та кольору)
- Зменшення роздільної здатності кольорових каналів (Downsampling)

- Розбиття зображення на блоки 8×8
- Дискретне косинус-перетворення (DCT)
- Квантизація

Саме на етапі квантизації якість стиснення відіграє важливу роль. При високій якості стиснення коефіцієнти DCT округлюються з малою похибкою. При низькому – з великою, через що губляться дрібні деталі. Саме тому якість стиснення прямо впливає на ефективність роботи ELA.

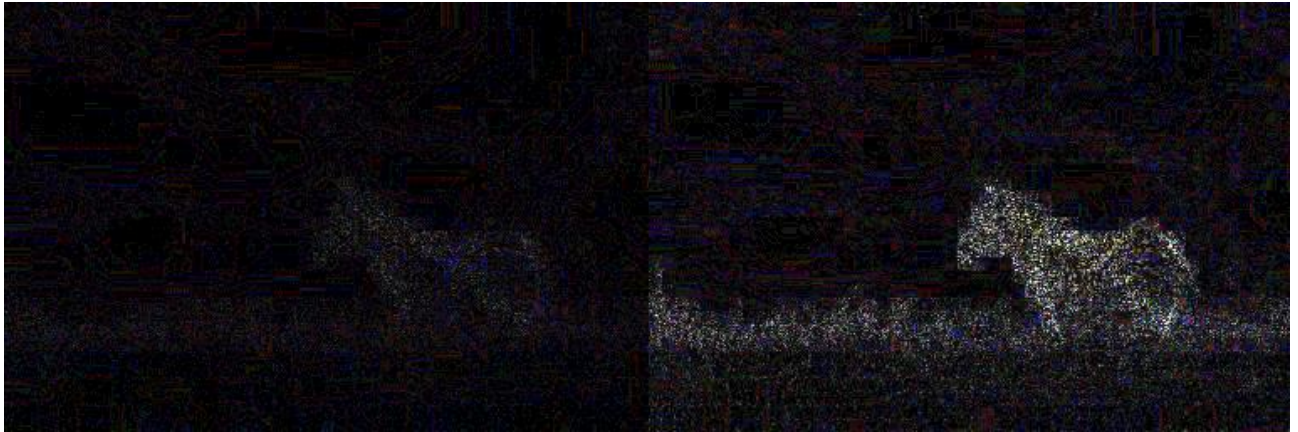


Рисунок 2.1 – Візуалізація алгоритму ELA для зображень, збережених з якістю 95% (ліворуч) та 80% (праворуч)

У багатьох дослідженнях ELA використовується як попередній етап для подачі даних у нейронну мережу. Оскільки ELA генерує карту рівнів помилок стиснення, цю карту можна подавати як вхідний канал для навчання моделі нейронної мережі. Це підвищує точність виявлення фальсифікацій порівняно з аналізом звичайного RGB-зображення.

2.3 Використання нейронних мереж для аналізу маніпуляцій із зображеннями

Нейронні мережі – це моделі машинного навчання, що імітують складні функції людського мозку. Ці моделі складаються з взаємопов'язаних вузлів або нейронів, які обробляють дані, вивчають закономірності та дозволяють виконувати такі завдання, як розпізнавання образів та прийняття рішень [6].

Існує багато різних типів нейронних мереж, кожен має свої особливості архітектури та підходи до обробки інформації. У таблиці 2.1 розглянуто найпоширеніші типи нейронних мереж, які застосовуються для виявлення маніпуляцій із зображеннями, їхні переваги та недоліки.

Таблиця 2.1 - Типи нейронних мереж для виявлення підроблених зображень

Тип нейронної мережі	Опис	Переваги	Недоліки
RNN (Recurrent Neural Network)	Аналізує послідовності даних, може застосовуватись до послідовностей кадрів відео.	Ефективна в обробці послідовностей	Не підходить для зображень без модифікації
CNN (Convolutional Neural Network)	Використовує згортки для автоматичного виділення ознак з зображень.	Висока точність, ефективність у роботі з візуальними даними	Потребує великих обсягів даних і часу навчання
Autoencoder	Самонавчальна модель для стискування і відновлення зображень	Може знаходити неочевидні відхилення від нормального патерну	Складно локалізувати конкретну змінену область
Capsule Networks	Альтернатива CNN з можливістю збереження позиційних зв'язків між ознаками.	Більш інформативне представлення об'єктів	Малопоширена, складна реалізація

З аналізу таблиці 2.1 можна зробити висновок, що згорткові нейронні мережі (CNN) є найбільш оптимальним вибором для виявлення маніпуляцій з зображеннями.

Згорткова нейронна мережа складається з вхідного шару, прихованих шарів та вихідного шару. Основну роботу в мережі виконують згорткові шари, які є ключовим компонентом прихованих шарів. Кожен згортковий шар містить набір фільтрів (ядер згортки), що ковзають по вхідному зображенню та обчислюють скалярний добуток між фільтром та локальною областю зображення. Ця операція, відома як згортка, дозволяє виявляти локальні особливості та шаблони в даних. Між згортковими шарами часто додають шари підвибірки (пулінгу), які зменшують розмірність даних, зберігаючи при цьому найважливіші ознаки. Найпоширенішими є *max-pooling*, що вибирає максимальне значення в області, та *average-pooling*, який обчислює середнє значення. [7].

Хоча CNN демонструє високу точність в своїй роботі, сама по собі вона не може виявити ознаки, необхідні для безпосереднього виявлення фейкового зображення. Для того, щоб протидіяти цій проблемі, можна скористатись попередньою обробкою вхідних даних через ELA.

2.4 Архітектура методики виявлення фейкових зображень на основі ELA та CNN

Для ефективного роботи методики виявлення фейкових зображень використовується послідовність етапів, кожен з яких виконує важливу роль в загальному процесі детектування маніпуляцій. Методика побудована на поєднанні аналізу рівня похибки та сучасних технологій глибинного навчання.

Комбіноване використання CNN та ELA покращує здатність системи виявляти підробки, зберігаючи гнучкість та ефективність у різних областях застосування та якостях зображень. Крім того, структура пропонує можливості

обробки в режимі реального часу, що робить її дуже застосовною в сценаріях, які вимагають швидкої та надійної автентифікації зображень [8].

Рисунок 2.2 ілюструє блокову діаграму методики виявлення фейкових зображень, описуючи послідовний потік даних від вхідних зображень до кінцевого результату.

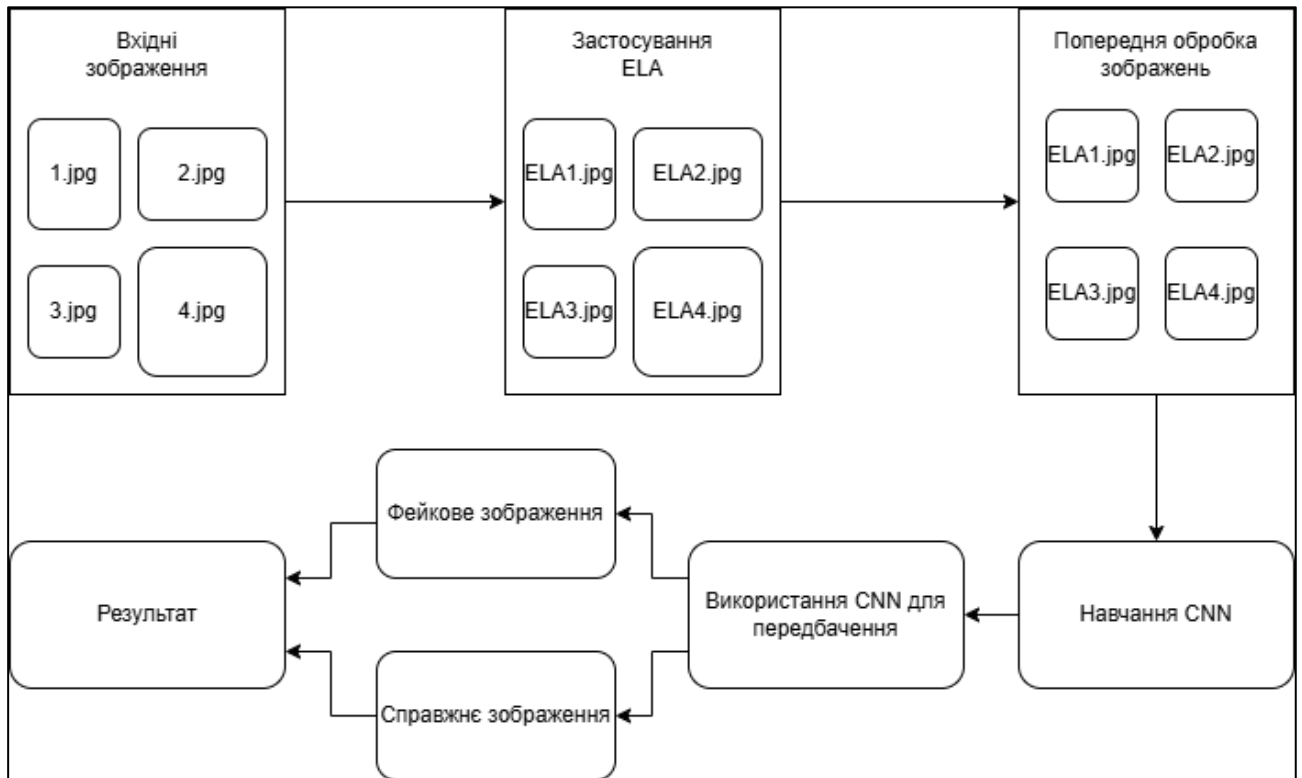


Рисунок 2.2 - Блок-схема методики виявлення фейкових зображень

Нижче наведено детальний опис кожного етапу:

1. Збір та підготовка даних

На початковому етапі збирається набір зображень для формування датасету, який має автентичні та редаговані приклади. Джерелами можуть бути відкриті набори даних. Попередньо потрібно впевнитись, що усі зображення приведені до одного формату JPEG. Ці зображення є базою для подальшого аналізу та навчання моделі.

2. Застосування методу ELA

Після формування датасету до кожного зображення застосовується метод ELA. Зображення повторно зберігається з певною якістю JPEG, відокремлюються відмінності між початковим і повторно збереженим варіантом.

Описати роботу цього етапу можна наступною формулою:

$$ELA(x, x') = |x - x'| \quad (2.1)$$

де x – оригінальне зображення;

x' – повторно стиснуте зображення.

В результаті, отримується візуальна помилкова мапа, у якій відредаговані області мають вищий рівень помилок. Ці ELA зображення будуть слугувати вхідним каналом для подальшого навчання моделі CNN.

3. Попередня обробка

Отримані на попередньому етапі ELA зображення проходять додаткову обробку для забезпечення сумісності з вимогами нейронної мережі:

- Усі зображення масштабуються до єдиного розміру, що очікує модель. Це потрібно, тому що нейронні мережі працюють із вхідними даними фіксованого розміру для забезпечення стабільності навчання.
- Зображення перетворюються у тривимірний масив чисел, що репрезентує кожен піксель. Кольорове зображення з фіксованим розмір має 3 канали (Red, Green, Blue). Після перетворення отримується масив розміром (Ширина, Висота, 3), де кожне значення – це інтенсивність пікселя від 0 до 255.
- Виконується нормалізація пікселів. Так як у зображеннях кожен піксель зазвичай представлений числом у діапазоні від 0 до 255, такий масштаб є досить великим для навчання моделі. Тому усі значення пікселів зводяться до стандартного діапазону від 0 до 1. Це робить зображення більш зручними для навчання нейронної мережі.

4. Навчання моделі CNN

Отримані масиви чисел, що представляють пікселі зображення поділяються на дві частини: тренувальні – власне для самого навчання моделі, та валідаційні – для перевірки її роботи. Для досягнення максимальної ефективності використовуються сучасні оптимізатори, функції втрат та метрики точності.

Процес навчання складається з багатьох епох, протягом яких модель вдосконалює свої параметри. Для запобігання перенавчанню також використовуються регуляризаційні методи.

5. Використання моделі CNN для передбачення

Після тренування моделі, її можна використовувати для класифікації нових зображень. Для цього, зображення повинне пройти таку саму попередню обробку, що й зображення під час навчання.

На основі сформованих ознак CNN дає ймовірність того, чи було зображення редаговане. Оцінка класифікації визначається сигмоїдною функцією:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2.2)$$

де x – зважена сума входів нейрона.

Цей підхід перетворює прогнози моделі на ймовірності між 0 та 1, що сприяє ефективній бінарній класифікації.

Висновки до розділу 2

У цьому розділі було розкрито теоретичні основи комп'ютерного зору, що є надзвичайно важливими для побудови систем автоматичного аналізу зображень. Особливу увагу приділено бібліотеці OpenCV, яка є ключовим інструментом для реалізації базових та складних операцій з обробки зображень – таких як зчитування, масштабування, перетворення кольорових просторів, фільтрація та ін.

Окремий акцент зроблено на використанні методу аналізу рівня помилок (ELA) як інструменту цифрової криміналістики для виявлення слідів маніпуляцій у JPEG-зображеннях. Цей метод дозволяє візуалізувати відмінності в рівнях похибок стиснення, які можуть свідчити про внесення змін до окремих фрагментів зображення. Його ефективність залежить від якості стиснення, а також від точності оцінки локальних змін у структурі зображення. Саме тому ELA є корисним попереднім етапом для глибших методів аналізу.

Також було розглянуто можливості використання нейронних мереж для аналізу автентичності зображень, зокрема таких типів, як згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN), автоенкодерів та Capsule Networks. З аналізу їхніх особливостей було встановлено, що CNN є найефективнішим інструментом у задачах виявлення фальсифікованих зображень завдяки здатності автоматично виділяти візуальні ознаки та адаптуватися до різних типів даних.

Розроблена архітектура методики виявлення фейкових зображень, яка базується на послідовному поєднанні аналізу рівня помилок (ELA) і глибокого навчання за допомогою CNN. Було детально описано кожен етап цього процесу — від попередньої обробки зображень та створення мап помилок ELA до навчання та застосування згорткової нейронної мережі для класифікації. Така комбінована методика забезпечує високу точність виявлення маніпуляцій на фейкових зображеннях.

3 РОЗРОБКА І РЕАЛІЗАЦІЯ МЕТОДИКИ ВИЯВЛЕННЯ ФЕЙКОВИХ ЗОБРАЖЕНЬ

3.1 Вибір середовища розробки та інструментів

Для реалізації методики виявлення фейкових зображень було обрано мову програмування Python 3.11. Це одна із найпопулярніших мов у сфері штучного інтелекту та обробки зображень, через простий синтаксис та наявність великої кількості бібліотек.

Як середовище розробки використовувався PyCharm, що забезпечує зручну інтерактивну роботу з кодом.

Для реалізації програми було використано такі бібліотеки:

- `os` – забезпечує доступ до функцій операційної системи, дозволяє працювати з файлами та каталогами.
- `cv2` – інтерфейс до бібліотеки OpenCV для обробки зображень.
- `sys` – дозволяє взаємодіяти з системними параметрами та отримувати доступ до аргументів командного рядка.
- `numpy` – бібліотека для роботи з багатовимірними масивами та виконання чисельних операцій.
- `matplotlib` – використовується для створення графіків, побудови візуалізацій результатів роботи програми.
- `sklearn` - набір інструментів машинного навчання: оцінка моделей, метрики, попередня обробка даних.
- `tensorflow` – фреймворк для побудови, навчання і використання нейронних мереж, особливо глибокого навчання.
- `seaborn` – розширення до `matplotlib`, призначене для красивої й інформативної візуалізації статистичних даних.
- `shutil` - модуль для роботи з файлами та папками: копіювання, переміщення, видалення каталогів.

3.2 Вибір датасету та підготовка зображень

Для навчання та валідації моделі нейронної мережі в цій дипломній роботі використовувався датасет CASIA 2.0 Image Tampering Detection Dataset [9].

Датасет був завантажений з платформи Kaggle. Набір даних складається з 7492 автентичних та 5125 фейкових зображень різних форматів розміщених у каталогах Au (Authentic) та Tr (Tampered) відповідно.

Редаговані зображення містять у собі різні типи змін, такі як вставка елементів з інших зображень, локальні маніпуляції, заміна фрагментів, ретушування та ін.

Для підготовки зображень було розроблено скрипт, який автоматично сортує та копіює тільки ті файли з датасету CASIA 2.0, що мають розширення .jpg, оскільки саме цей формат є придатним для подальшого аналізу методом ELA.

```
source_path = "dataset"
dest_path = "clean_dataset"

os.makedirs(name=f"{dest_path}/real", exist_ok=True)
os.makedirs(name=f"{dest_path}/fake", exist_ok=True)

def copy_images(src_folder, dst_folder): 2 usages
    count = 0
    for file in os.listdir(src_folder):
        if file.lower().endswith(".jpg"):
            shutil.copy(os.path.join(src_folder, file),
                        os.path.join(dst_folder, file))
            count += 1
    return count

real_count = copy_images(os.path.join(source_path, "Au"), os.path.join(dest_path, "real"))
fake_count = copy_images(os.path.join(source_path, "Tr"), os.path.join(dest_path, "fake"))

print(f"Кількість реальних зображень: {real_count}\n"
      f"Кількість фейкових зображень: {fake_count}")
```

Рисунок 3.1 – Скрипт для фільтрації та сортування зображень

Зображення з папки Au (authentic) були класифіковані як реальні та скопійовані до каталогу clean_dataset/real, тоді як зображення з папки Tr (tampered) були класифіковані як підроблені і збережені у clean_dataset/fake.

Для цього було створено нову структуру директорій, після чого реалізовано автоматичну фільтрацію та копіювання файлів за допомогою модуля shutil. Фільтрація здійснювалася шляхом перевірки розширення файлів .jpg, незалежно від регістру символів у назві.

В результаті роботи скрипту було отримано 7437 реальних та 2064 фейкових зображень:

```
Кількість реальних зображень: 7437
Кількість фейкових зображень: 2064
```

Рисунок 3.2 – Результат виконання скрипту сортування зображень

Як бачимо, кількість реальних зображень майже в 4 рази більше фейкових. Через таку незбалансованість класів в подальшому буде використовуватись балансування вагів під час навчання.

3.3 Реалізація алгоритму ELA

Для реалізації алгоритму ELA було створено функцію `convert_to_ela_image()`, яка майже повністю була реалізована за допомогою бібліотеки OpenCV.

На вхід функція приймає два параметри:

- `image_path` – шлях до зображення, над яким буде виконуватись алгоритм ELA;
- `quality` – якість, з якою буде стискатись зображення.

Далі наведений опис та реалізація основних етапів функції:

1. Вхідне зображення зчитується з заданого шляху в параметрі `image_path` за допомогою функції `cv2.imread()`. Далі воно записується в тимчасовий файл через функцію `cv2.imwrite()` з якістю `quality` з параметру функції. Тимчасове зображення відкривається та записується у змінну.

```
image = cv2.imread(image_path)
cv2.imwrite(temp_filename, image, params=[cv2.IMWRITE_JPEG_QUALITY, quality])
temp_image = cv2.imread(temp_filename)
```

Рисунок 3.3 – Повторне стиснення JPEG зображення

2. Різницю пікселів між вхідним та тимчасовим зображенням обраховує функція `cv2.absdiff()`. Таким чином отримується саме ELA

зображення. Далі воно перетворюється до типу `uint8`, щоб усі значення в масиві були цілими беззнаковими числами.

```
ela_image = cv2.absdiff(image, temp_image)
ela_image = ela_image.astype(np.uint8)
```

Рисунок 3.4 – Генерація ELA зображення

3. Для того щоб відмінності на ELA зображенні більш помітними, значення пікселів масштабуються до діапазону від 0 до 255. Це підсилює контраст і прокращує візуалізацію. Масштабування відбувається за наступною формулою:

$$image = \frac{(image - min) * 255}{max - min} \quad (3.1)$$

де `min` – мінімальне значення масиву;

`max` – максимальне значення масиву.

```
min_val = np.min(ela_image)
max_val = np.max(ela_image)
ela_image = (ela_image - min_val) * 255.0 / (max_val - min_val)
```

Рисунок 3.5 – Масштабування значень пікселів

4. В кінці усі тимчасові зображення видаляються. Функція повертає ELA зображення у вигляді `numpy`-масиву типу `uint8`.

Цей алгоритм є одним із головних етапів, так як він надає моделі можливість виділяти з зображень маніпуляції.

3.4 Створення моделі на основі CNN

Модель згорткової нейронної мережі створювалась за допомогою модуля Keras бібліотеки TensorFlow. Keras — це потужна та зручна бібліотека Python з відкритим кодом для розробки моделей глибокого навчання.

У даній роботі використовується модель `Sequential`, яка включає:

- Згорткові шари для виділення ознак із зображень ELA.
- Агрегувальні шари для зменшення розмірності та підвищення інваріантності до незначних знім.

- Повнозв'єднані шари для класифікації зображень.

Sequential дозволяє об'єднувати послідовну купу шарів в модель для лінійного потоку даних.

На рисунку 3.6 зображена архітектура згорткової нейронної мережі та деякі додаткові параметри.

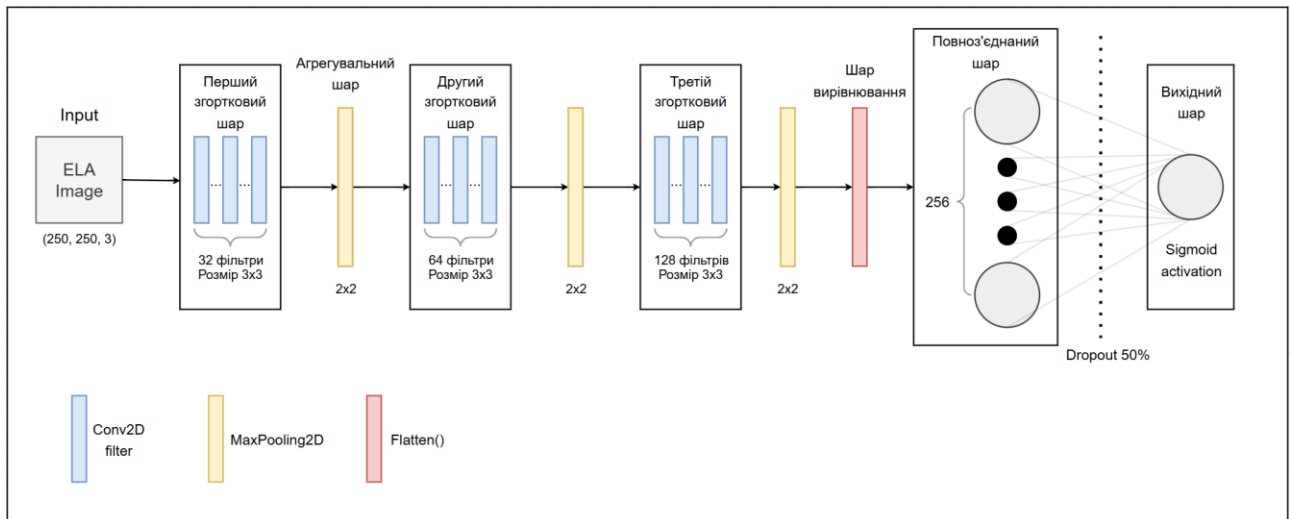


Рисунок 3.6 – Архітектура згорткової нейронної мережі

Детальне пояснення до кожного блоку архітектури:

- На вхід модель приймає ELA зображення у форматі (висота, ширина, канали). Було прийнято рішення встановити розмірність зображень 250x250 для балансу між деталізацією та обчислювальною ефективністю. Такий розмір дозволяє зберігти достатню кількість деталей для аналізу та не сильно збільшує обчислювальну складність. Також більша розмірність потребує більше пам'яті для тренування моделі, яка в цій роботі й так була використана по максимуму.
- Першим шаром у моделі йде згортковий шар Conv2D із 32 фільтрами розміром 3x3. Фільтри поступово застосовуються до кожного фрагменту зображення та виявляють певні структури. Цей шар націлений на аналіз простих ознак, таких як лінії, кути чи контури. В результаті, отримується активаційна карта (feature map), де високі значення вказують на наявність шуканої ознаки. Після згортки

застосовується функція активації ReLU (Rectified Linear Unit), яка додає нелінійність, відкидаючи від'ємні значення. ReLU прискорює навчання та прибирає слабкі активації, залишаючи лише корисні сигнали.

- Після згорткового шару йде агрегувальний шар MaxPooling2D розміром 2x2. Цей шар зменшує ширину та висоту активаційної карти, залишаючи лише найважливіші значення з вікна його розмірності. Таким чином зменшується кількість параметрів у нейронній мережі, що позитивно впливає швидкість обчислення та знижує ризик перенавчання.
- Комбінація із згорткового та агрегувального шару повторюється тричі. Відмінністю є лише те, що кількість фільтрів у кожному наступному згортковому шарі збільшується вдвічі. Таким чином другий згортковий шар має 64 фільтри, а третій – 128. Ці згорткові шари приймають та опрацьовують вихід із попередніх шарів. Поступове збільшення фільтрів у кожному з них дозволяє виявляти складніші ознаки, що вказують на маніпуляції у зображеннях.
- Після згорткових та агрегувальних шарів йде шар вирівнювання Flatten. Сам по собі цей шар не має навчальних параметрів. Його основна задача – перетворення багатовимірних даних у одновимірний вектор для подальшої обробки повноз'єднаними шарами. У моєму випадку він перетворює тензор з останнього агрегувального шару (ширина, висота, канали) у одновимірний вектор до наступного шару Dense.
- Наступним іде шар повноз'єднаний шар Dense. Він містить у собі 256 нейронів і функцію активації ReLU. Шар комбінує всі виявленні особливості і робить аналіз цих ознак. Велика кількість нейронів допомагає моделі вивчити складні взаємозв'язки між ознаками.

- Перед останнім шаром йде шар відсікання Dropout з коефіцієнтом 0.5. Він випадковим чином обирає 50% нейронів під час навчання та відключає їх, для того щоб уникнути перенавчання.
- Останнім шаром є повноз'єднаний шар з одним нейроном та сигмоїдною функції активації для бінарної класифікації зображень. Він визначає фінальний результат роботи нейронної мережі, а саме ймовірність того, що вхідне зображення фейкове або справжнє.

Таблиця 3.1 – Схематичне представлення архітектури Sequential CNN

Шар	Тип	Параметри	Вихідний розмір	Деталі
Input	Зображення	input_shape = (250, 250, 3)	250x250x3	Вхідні кольорові зображення
Conv2D	Згортковий шар	32 фільтри 3x3, ReLU	248x248x32	Виявлення базових ознак
MaxPooling2D	Субдискретизація	Вікно 2x2	124x124x32	Зменшення розмірності у 2 рази
Conv2D	Згортковий шар	64 фільтри 3x3, ReLU	122x122x64	Виявлення складніших ознак
MaxPooling2D	Субдискретизація	Вікно 2x2	61x61x64	Зменшення розмірності у 2 рази
Conv2D	Згортковий шар	128 фільтрів 3x3, ReLU	59x59x128	Виявлення високорівневих ознак

Кінець таблиці 3.1

MaxPooling2D	Субдискретизація	Вікно 2x2	29x29x128	Зменшення розмірності у 2 рази
Flatten	Вирівнювання	Відсутні	107648 (29x29x128)	Перетворення 3D даних у 1D
Dense	Повноз'єднаний шар	256 нейронів, ReLU	256	Комбінування ознак для класифікації
Dropout	Регуляризація	Rate = 0.5	256	Відключення 50% нейронів
Dense	Вихідний шар	1 нейрон, sigmoid	1	Бінарна класифікація

Також модель використовує оптимізатор Adam – метод стохастичного градієнтного спуску, який базується на адаптивній оцінці моментів першого і другого порядку та адаптивно оновлює ваги під час навчання. Як швидкість навчання (learning_rate) моделі встановлене типове значення 0.0001.

В якості функції втрат використовується бінарна кросс-ентропія:

$$L = -\frac{1}{N} \sum_{i=1}^N [y_i * \log(p_i) + (1 - y_i) * \log(1 - p_i)] \quad (3.2)$$

де N – кількість зображень в поточному батчі;

y_i – реальна мітка i -го зображення;

p_i – прогнозована ймовірність.

Функція вимірює розбіжність між прогнозованою ймовірністю та реальною міткою.

Реалізація створення моделі Sequential зображена на рисунку 3.7.

```
def create_model(input_shape): 1 usage
    model = Sequential([
        Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=input_shape),
        MaxPooling2D((2, 2)),
        Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Conv2D(filters=128, kernel_size=(3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(units=256, activation='relu'),
        Dropout(0.5),
        Dense(units=1, activation='sigmoid')
    ])
    model.compile(optimizer=Adam(learning_rate=0.0001),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model
```

Рисунок 3.7 – Модель Sequential

3.5 Навчання та тестування моделі, побудова метрик

Зображення з підготовленого датасету розподіляються на тренувальні та валідаційні, розподіленим даним даються маркування: справжнім – 1, та фейковим – 0. Розподілення даних виконувалось у співвідношенні 80% для тренування та 20% для валідації за допомогою функції `train_test_split()` бібліотеки `sklearn`:

```
X_train, X_val, y_train, y_val = train_test_split(*arrays: X, y, test_size=0.2, random_state=1, stratify=y)
print(f"\nTraining samples: {len(X_train)} (Real: {sum(y_train)}, Fake: {len(y_train)-sum(y_train)})")
print(f"\nValidation samples: {len(X_val)} (Real: {sum(y_val)}, Fake: {len(y_val)-sum(y_val)})")
```

Рисунок 3.8 – Розподілення даних на тренувальні та валідаційні

В результаті було отримано:

- 7600 зображень для тренування (з них 5949 – справжні, 1651 – фейкові)
- 1901 зображення для валідації (з них 1488 – справжні, 413 – фейкових)

Через такий дисбаланс класів використовувалося автоматичне обчислювання коригувальних ваг для навчання моделі за допомогою функції `compute_class_weight`:

```
class_weights = compute_class_weight(class_weight='balanced', classes=np.unique(y), y=y)
class_weight_dict = {0: class_weights[0], 1: class_weights[1]}
print(f"\nClass weights: {class_weight_dict}")
```

Рисунок 3.9 – Обчислення коригувальних ваг

Отримані значення ваг були наступними:

- Для класу 0: 2.301598837209302
- Для класу 1: 0.638765631302944

Для процесу навчання використовувалися зворотні виклики (callbacks), які допомагають автоматизувати певні дії під час тренування. Для покращення результатів було використано зворотні виклики `EarlyStopping` та `ModelCheckpoint`. `EarlyStopping` зупиняє навчання моделі автоматично, якщо її результати валідаційних даних перестають покращуватись.

Параметри `EarlyStopping`:

- `Patience = 5`, модель продовжить тренування ще 5 епох після останнього покращення. Якщо за цей час метрики не покращаться, то тренування зупиниться.
- `Restore_best_weights = True`, після зупинки модель буде мати свої найкращі ваги, а не останні.

`ModelCheckpoint` зберігає модель під час тренування, параметри:

- Назва файлу, у який буде збережено модель
- `Save_best_only = True`, зберігатись буде тільки модель, яка показала найкращі метрики.

Початковими параметрами навчання було встановлено 50 епох, батч містив у собі 32 зображення, ELA зображення стискалися з якістю 90%. В результаті роботи генерувався звіт за допомогою функції `classification_report()` з бібліотеки `sklearn`, який містив у собі інформацію з метриками, та матриця плутанини (Confusion Matrix).

В результаті перше навчання тривало 20 епох та було отримано наступні результати:

Таблиця 3.2 – Перший звіт навчання

Клас	Precision	Recall	F1-score	Support
Fake	0.87	0.89	0.88	413
Real	0.97	0.96	0.97	1488
Accuracy	-	-	0.95	1901
Macro avg	0.92	0.93	0.92	1901
Weighted avg	0.95	0.95	0.95	1901

Влучність (Precision) – частка всіх результатів, які модель класифікує як позитивні і які й справді такі. Обраховується за наступною формулою:

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)} \quad (3.3)$$

У задачі класифікації:

- True Positive – кількість зразків, які модель правильно класифікувала як позитивні.
- False Positive – кількість зразків, які модель помилково класифікувала як позитивні.
- False Negative – кількість зразків, які модель пропустила (позитивні, але класифіковані як негативні)
- True Negative – кількість зразків, які модель правильно класифікувала як негативні.

Повнота (Recall) – коефіцієнт істинно позитивних результатів, обраховується за такою формулою:

$$Recall = \frac{TP}{TP + FN} \quad (3.4)$$

F1-міра (F1-score) – середнє гармонійне влучності та повноти, обраховується за формулою:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (3.5)$$

Середнє арифметичне (Macro Avg) – власне середнє значення метрик (precision, recall, F1) для обох класів без урахування їхньої кількості.

Середнє зважене (Weighted Avg) – середнє значення метрик, зважене за кількістю зразків у кожному класі, враховує дисбаланс класів.

Точність (Accuracy) – загальна частка правильно класифікованих зразків, формула точності:

$$Accuracy = \frac{TP + TN}{\text{Усі зразки}} \quad (3.6)$$

З таблиці 3.2 видно, що модель демонструє високу загальну точність 0.95, проте через сильний дисбаланс класів ця метрика не є дуже показною. Для класу справжніх зображень, результати є набагато кращими, що пояснюється більшою кількістю навчальних даних для цього ж класу. Як видно, попри застосування методів балансування класів, модель продемонструвала суттєву асиметрію в якості класифікації.

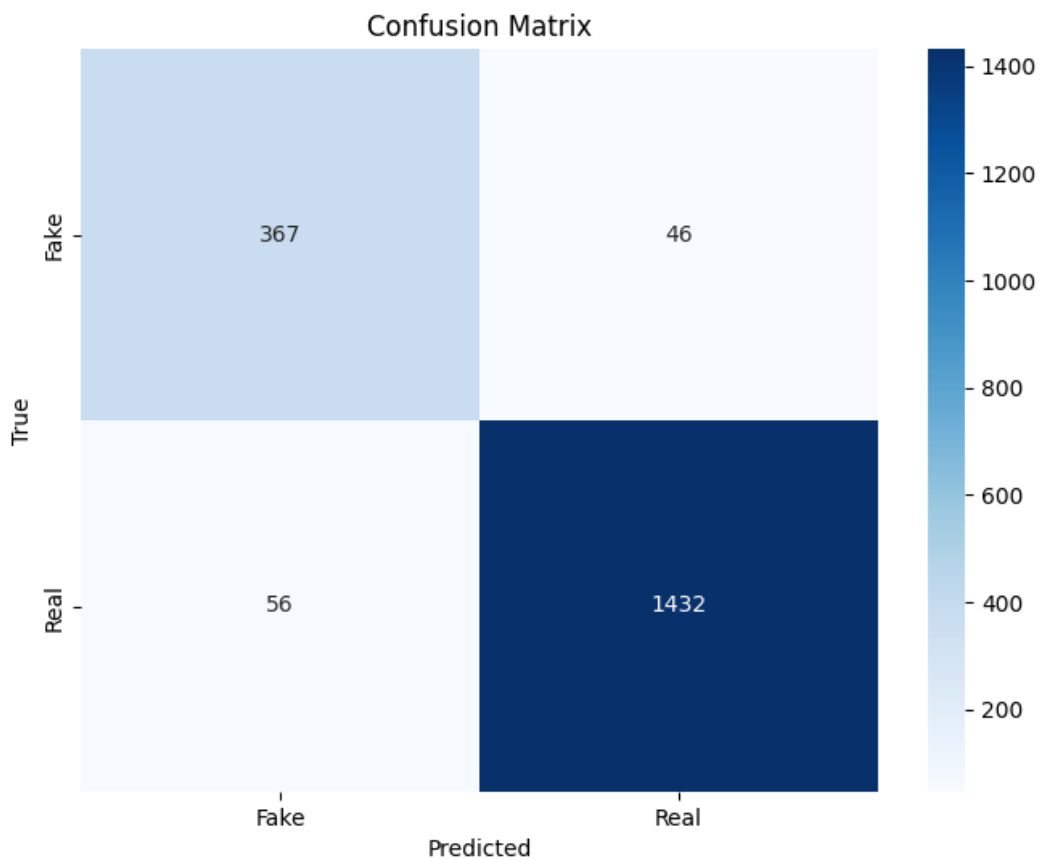


Рисунок 3.10 – Матриця невідповідностей

З матриці невідповідностей на рисунку 3.10 видно, що:

- True Negative: 1432 зображень були правильно ідентифіковані як справжні.
- True Positive: 367 фейкових зображень були правильно ідентифіковані як фейкові.
- False Negative: 46 фейкових зображень було невірнo класифіковано як справжні.
- False Positive: 56 справжніх зображень було помилково класифіковано як фейкові.

За результатами помітно, що модель навчилася добре розпізнавати справжні зображення, але відстає у класифікації фейкових зображень.

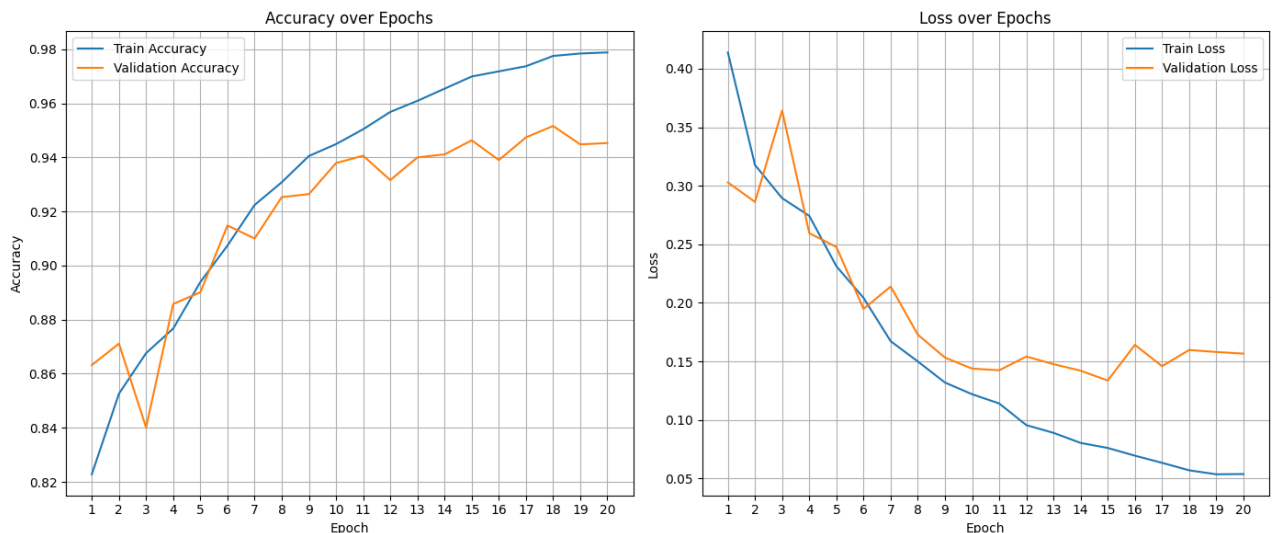


Рисунок 3.11 – Графіки втрат та точності впродовж епох

З графіків на рисунку 3.11 помітно, що після 15 епохи валідаційні точність та втрати перестали покращуватися, тому спрацював зворотній виклик EarlyStopping та значення вагів були повернені до 15 епохи.

На основі аналізу отриманих метрик було прийнято рішення про навчання моделі на збалансованому наборі даних, де класи будуть містити у собі зображення з однаковою кількістю зразків. Тому, кількість справжніх зображень була обмежена до 2064.

Повторне навчання моделі з однаковою кількістю зразків у кожному класі виявилось ефективним рішенням та продемонструвало більш збалансовані

значення метрик. Результати метрик повторного навчання зображенні у таблиці 3.3.

Таблиця 3.3 – Другий звіт навчання

Клас	Precision	Recall	F1-score	Support
Fake	0.93	0.94	0.94	413
Real	0.94	0.93	0.93	413
Accuracy	-	-	0.93	826
Macro avg	0.93	0.93	0.93	826
Weighted avg	0.93	0.93	0.93	826

Для обох класів Fake та Real досягнуто високу та збалансовану точність F1-score 0.94 та 0.93 відповідно. Тепер модель демонструє стабільні результати для обох класів без значних зміщень у сторону класу з більшою кількістю тренувальних даних.

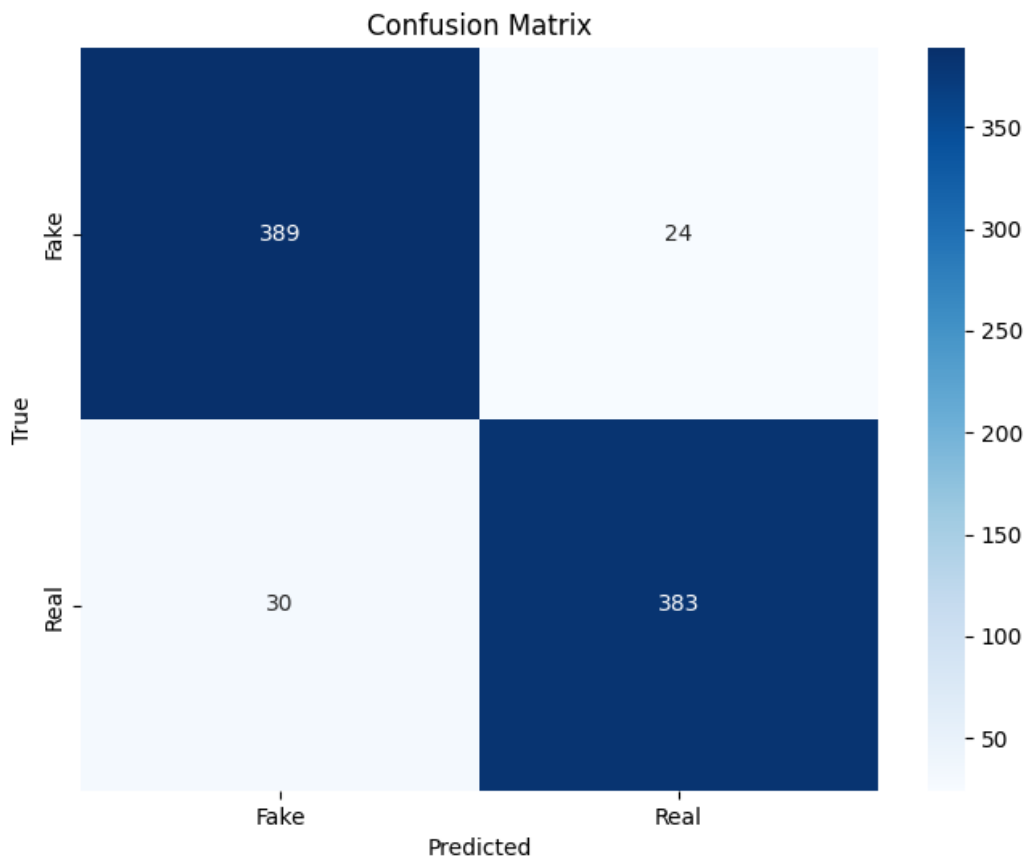


Рисунок 3.12 – Друга матриця невідповідностей

З 826 валідаційних зображень, відповідно до рисунку 3.12:

- True Negative: 383 зображень були правильно ідентифіковані як справжні.
- True Positive: 389 фейкових зображень були правильно ідентифіковані як фейкові.
- False Negative: 24 фейкових зображень було невірнo класифіковано як справжні.
- False Positive: 30 справжніх зображень було помилково класифіковано як фейкові.

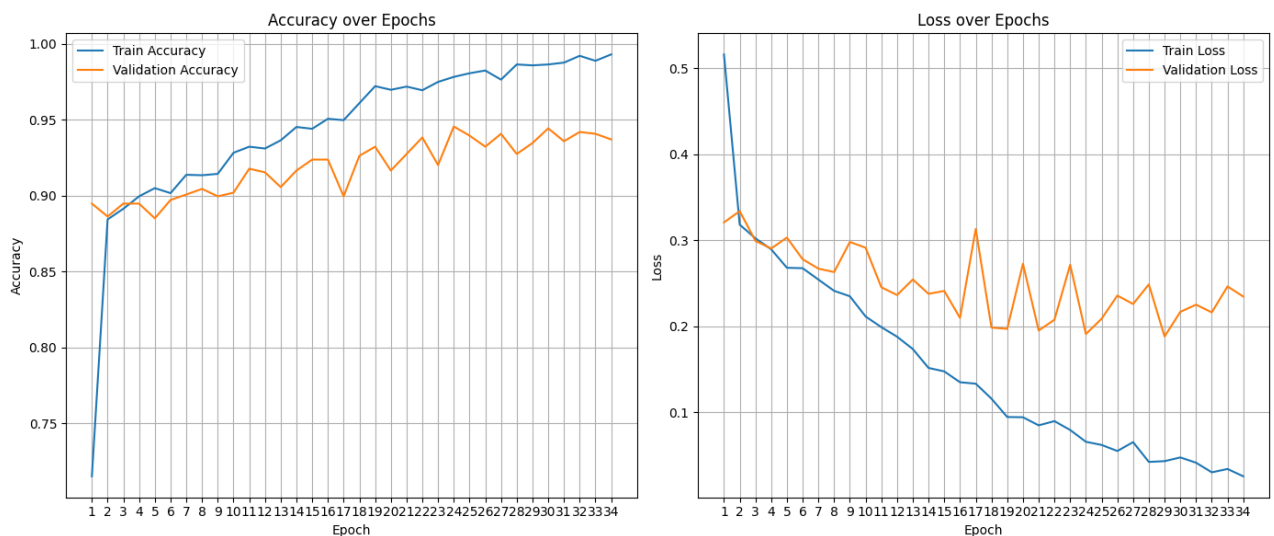


Рисунок 3.13 – Графіки втрат та точності другого навчання

Навчання було призупинено на 34 епосі та EarlyStopping повернув значення ваг до 29 епохи, в зв'язку з відсутністю покращень валідаційних значень.

Як експеримент, було проведено третє навчання моделі, проте з іншим рівнем стискання вхідних ELA зображень. Тепер вони стискались з 95% якості при генерації, що дозволило прибрати зайві ознаки з мап помилок для навчання моделі. Результати метрик третього навчання зображено у таблиці 3.4.

Таблиця 3.4 – Третій звіт навчання

Клас	Precision	Recall	F1-score	Support
Fake	0.95	0.97	0.96	413
Real	0.97	0.95	0.96	413

Кінець таблиці 3.4

Accuracy	-	-	0.96	826
Macro avg	0.96	0.96	0.96	826
Weighted avg	0.96	0.96	0.96	826

З таблиці 3.4 видно, що результати F1-score для обох класів збільшилися на 0.03 в порівнянні з минулим навчанням. Загалом було досягнуто точність передбачень моделі на валідаційних зображеннях у 96%

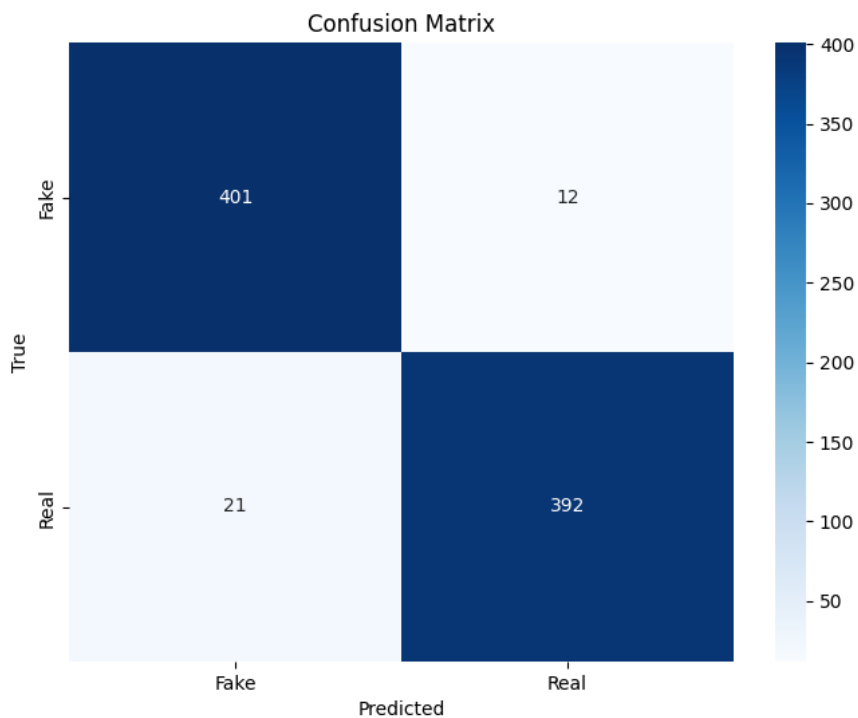


Рисунок 3.14 – Третя матриця невідповідностей

За матрицею невідповідностей з рисунку 3.14:

- 12 з 413 фейкових зображень були класифіковані як справжні.
- 21 з 413 справжніх зображень були класифіковані як фейкові.

Останнє навчання також тривало 34 епохи, значення ваг були повернуті до 29 епохи. Значення втрат та точностей протягом епох зображенні на рисунку 3.14.

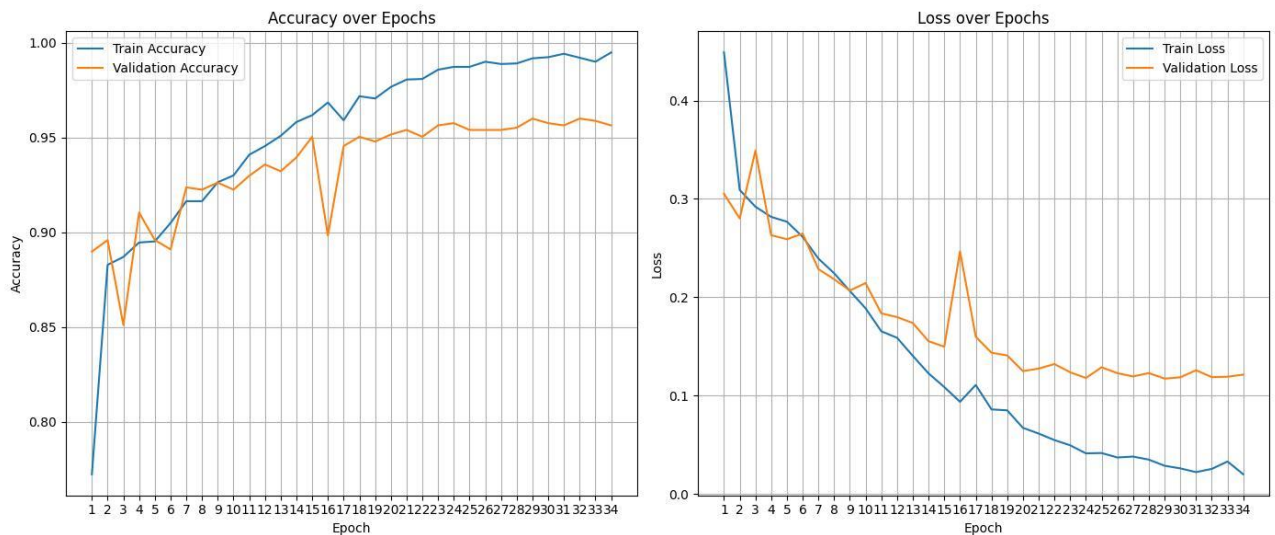


Рисунок 3.15 – Графіки втрат та точності третього навчання

Як додаткова оцінка ефективності розробленої в даній роботі архітектури згорткової нейронної мережі також було проведено порівняльне тестування з трьома популярними архітектурами – VGG16, ResNet0, EfficientNetB0. Їх порівняльні характеристики наведені у таблиці 3.5.

Таблиця 3.5 – Порівняльна таблиця архітектур

Характеристика	VGG16	ResNet50	EfficientNetB0
Кількість шарів	16 (13 згорткових + 3 повнозв'язні)	50 (з residual-блоками)	82 (з MBConv-блоками)
Вхідний розмір	224x224	224x224	224x224
Основні блоки	3x3 Conv + MaxPooling	Bottleneck Residual Blocks	MBConv
Тип пулінгу	MaxPooling 2x2	Global Average Pooling	Global Average Pooling

До кожної з архітектур використовувався однаковий підхід:

- Базова частина моделі, що була попередньо натренована на датасеті ImageNet, була імпортована без верхньої класифікаційної частини.
- Параметри базових шарів були заморожені, таким чином вони використовували уже навчені ваги на датасеті ImageNet.

- До базових шарів додавалась така ж сама комбінація повнозв'язних шарів, що використовувалась у створеній мною архітектурі, спеціально розроблених для бінарної класифікації.
- Додані повнозв'язні шари навчались на такому ж наборі даних що й власно створена модель.

Таким чином стало можливим використання цих попередньо навчених моделей до моєї задачі бінарної класифікації зображень.

Усі архітектури були навчені та протестовані на тій ж самій вибірці зображень, що й моя власна архітектура. Було використано 3302 тренувальних зображення (з них 1651 справжніх та фейкових) та 826 валідаційних зображення (з них 413 справжніх та фейкових).

Таблиця 3.6 – Порівняння результатів моделей

Архітектура	Точність (%)	False Positive	False Negative	Відсоток помилок	Валідаційних зображень
Власна	96.00	21	12	4%	826
VGG16	94.67	22	22	5.33%	826
ResNet50	95.16	30	10	4.84%	826
EfficientNetB0	95.28	19	20	4.72%	826

Як показали результати тестування, власна архітектура CNN продемонструвала найменший відсоток помилок класифікацій (4%) серед усіх розглянутих моделей. Це свідчить про доцільність застосування спрощених, але добре налаштованих моделей для класифікацій зображень за допомогою методу ELA.

3.6 Використання методики для виявлення згенерованих зображень

У рамках дослідження також було застосовано дану методику для виявлення зображень штучної генерації. Результати експериментального дослідження показали, що ELA у поєднанні з CNN ефективний для виявлення класичних фотоманіпуляцій, таких як ретушування, локальні зміни, вставка об'єктів з інших зображень. Проте, при застосуванні ELA для детектування згенерованих штучним інтелектом зображень, ефективність методу знижується. Це можна пояснити тим, що такі зображення створюються безпосередньо в цифровому вигляді і не проходять процес повторної компресії, що є ключовим для роботи ELA.

Для дослідження було використано невеликий відкритий набір даних AI Generated Images vs Real Images [10] з платформи Kaggle. Він містить у собі різні варіації справжніх зображень та зображень згенерованих штучним інтелектом. Навчання та валідація моделі виконувалась за тими ж принципами що й раніше. В результаті було отримано наступні значення метрик:

Таблиця 3.7 – Результати метрик детектування згенерованих зображень

Клас	Precision	Recall	F1-score	Support
Fake	0.56	0.64	0.60	80
Real	0.58	0.50	0.54	80
Accuracy	-	-	0.57	160
Macro avg	0.57	0.57	0.57	160
Weighted avg	0.57	0.57	0.57	160

Крім того, для якісної роботи ELA необхідно, щоб зображення було у форматі JPEG, оскільки метод спирається на специфіку стиснення цього формату. Згенеровані зображення, збережені у безвтратних форматах, не піддаються ефективному аналізу з використанням ELA.

Таким чином, ELA є корисним інструментом для первинного аналізу підозрілих зображень та може бути ефективно застосований у комплексі з CNN. Проте, для завдань виявлення штучно згенерованих зображень рекомендується

використовувати інші методи детектування, які здатні виявляти характерні артефакти генеративних моделей.

3.7 Використання моделі для класифікації зображень

Для використання натренованої моделі було створено окремий Python скрипт, який дозволяє класифікувати зображення. Скрипт приймає зображення, проводить з ним таку ж саму попередню обробку що й з зображеннями перед навчанням моделі та виводить рисунок, на якому зображені початкове та ELA зображення для порівняння, також на рисунку міститься інформація про передбачення моделі, а саме сама класифікація зображення та рівень довіри у відсотках. Повний код скрипту міститься у додатку В. На рисунках 3.16 та 3.17 зображено приклади використання скрипту для класифікації зображень.

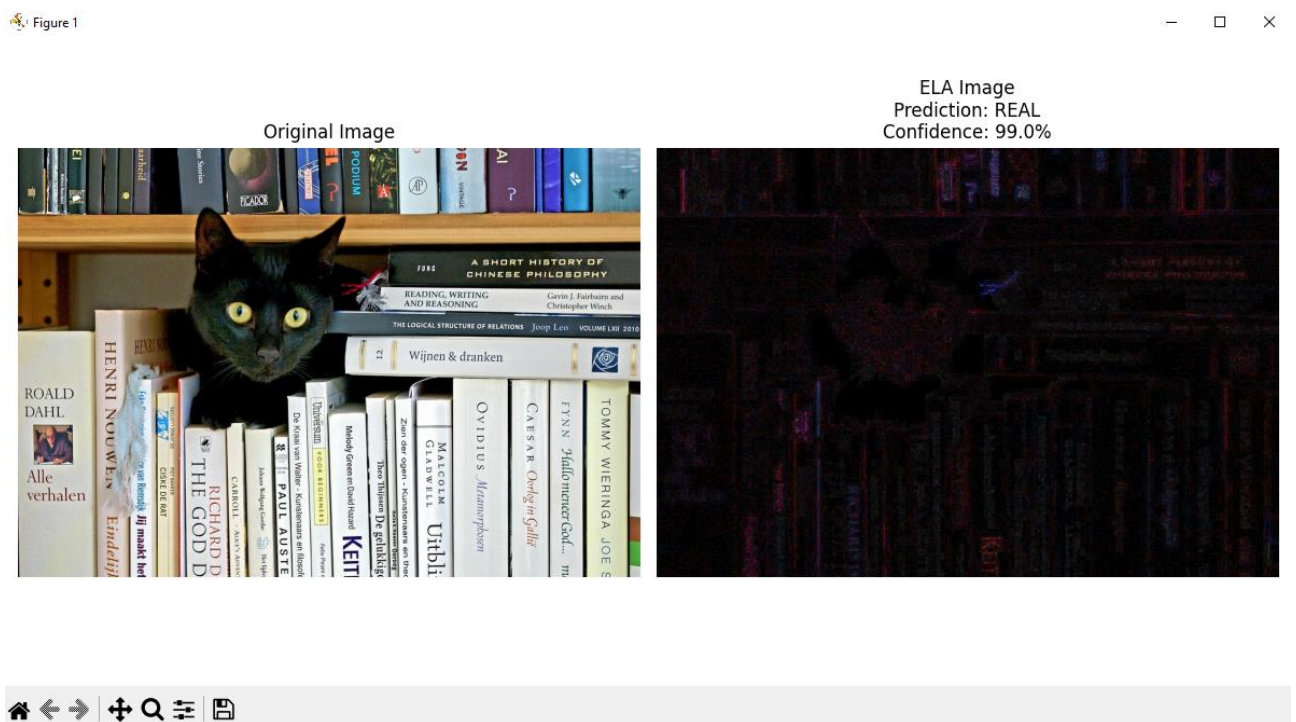


Рисунок 3.16 – Передбачення справжнього зображення

Figure 1

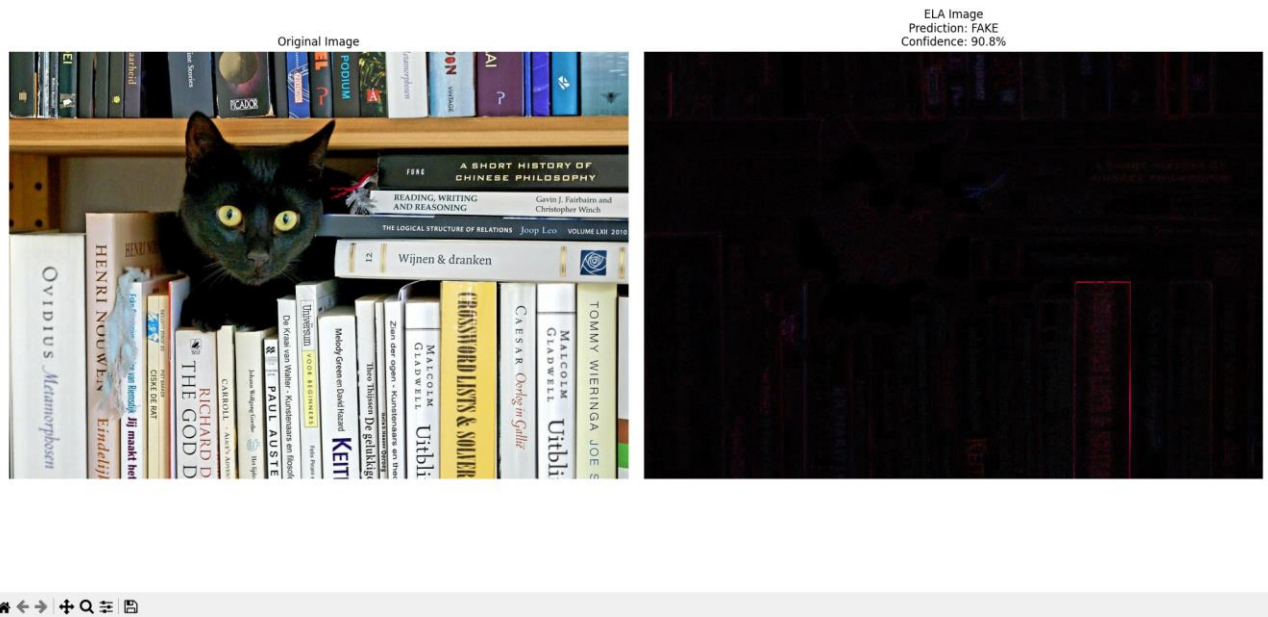


Рисунок 3.17 – Передбачення фейкового зображення

Для прикладу фейкового зображення було взято та вирізано книгу з іншого зображення та додано до зображення з рисунку 3.16.

Висновки до розділу 3

У ході реалізації методики виявлення фейкових зображень були обґрунтовано обрані мова програмування Python та середовище розробки PyCharm, що забезпечили зручність та ефективність розробки. Використання бібліотек OpenCV, TensorFlow/Keras, scikit-learn та NumPy дозволило реалізувати повний цикл обробки зображень — від підготовки даних до побудови й оцінювання моделі.

Обраний датасет CASIA 2.0 забезпечив надійну базу для навчання моделі, хоча потребував попереднього відбору та фільтрації для забезпечення відповідного формату та якості.

Впровадження ELA-аналізу стало важливим етапом у підвищенні точності виявлення маніпуляцій: метод візуалізації помилок JPEG-стиснення надав можливість моделі краще розрізняти справжні та підроблені зображення.

Побудована згорткова неймережа (CNN) із трьома згортковими шарами та повнозв'язаною частиною показала високу точність у класифікації, особливо після використання ELA з якістю JPEG 95%. Найкращий результат становив 96% точності при F1-оцінках 0.96 для обох класів, що свідчить про високу ефективність розробленої системи.

ВИСНОВКИ

У межах виконаної роботи було реалізовано ефективну методику виявлення фейкових зображень на основі глибокого навчання та аналізу рівня помилок (ELA). Дослідження охопило повний цикл розробки: від підбору датасету та підготовки даних до побудови та навчання згорткової нейронної мережі й оцінки її продуктивності.

В основі підходу лежить використання ELA-методу, який виявився ключовим для виявлення цифрових маніпуляцій, а також застосування згорткової нейронної мережі (CNN), що продемонструвала високу точність при класифікації зображень.

Результати експериментів показали, що модель здатна ефективно розрізняти справжні та підроблені зображення навіть у випадку маніпуляцій високої якості. Найкращий результат досягнуто після використання збалансованих даних і оптимізації якості ELA-зображень – точність класифікації сягнула 96%, а значення F1-score для обох класів становили 0.96.

Таким чином, поставлена мета – створити методику, яка здатна виявляти фейкові зображення була досягнута. Розроблений інструмент може бути використаний як основа для систем автоматизованого аналізу зображень у різних сферах, де важлива достовірність візуального контенту.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Hameleers M., Powell T. E., Van Der Meer T. G. L. A., Bos L. A Picture Paints a Thousand Lies? The Effects and Mechanisms of Multimodal Disinformation and Rebuttals Disseminated via Social Media. Political Communication. 2020. Vol. 37, Issue 2, pp. 281–301. DOI: 10.1080/10584609.2019.1674979
2. Woodhall S. Which Social Media Networks Remove EXIF Data? — URL: <https://stevenwoodhall.com/which-social-media-networks-remove-exif-data/>
3. Wikipedia contributors. Error level analysis. — URL: https://en.wikipedia.org/wiki/Error_level_analysis
4. Wikipedia contributors. Computer vision. — URL: https://en.wikipedia.org/wiki/Computer_vision
5. Obregon A. How to process images with Python using OpenCV. Medium. — URL: <https://medium.com/@AlexanderObregon/how-to-process-images-with-python-using-opencv-6894010d78e4>
6. GeeksforGeeks. Neural Networks - A Beginner's Guide. — URL: <https://www.geeksforgeeks.org/neural-networks-a-beginners-guide/>
7. Wikipedia contributors. Convolutional neural network. — URL: https://en.wikipedia.org/wiki/Convolutional_neural_network
8. More Shraddha. Enhancing Image Forgery Detection with Convolutional Neural Networks and Error Level Analysis. Journal of Information Systems Engineering and Management. 2025. Vol. 10, pp. 980-993. DOI: 10.52783/jisem.v10i27s.4757
9. Kaggle. CASIA 2.0 Image Tampering Detection Dataset. — URL: <https://www.kaggle.com/datasets/divg07/casia-20-image-tampering-detection-dataset>
10. Kaggle. AI-Generated Images vs Real Images Dataset. — URL: <https://www.kaggle.com/datasets/cashbowman/ai-generated-images-vs-real-images>

ДОДАТОК А PYTHON 3 КОД ORGINIZE_DATA.PY

```
import os
import shutil

source_path = "dataset"
dest_path = "clean_dataset"

os.makedirs(f"{dest_path}/real", exist_ok=True)
os.makedirs(f"{dest_path}/fake", exist_ok=True)

def copy_images(src_folder, dst_folder):
    count = 0
    for file in os.listdir(src_folder):
        if file.lower().endswith(".jpg"):
            shutil.copy(os.path.join(src_folder, file),
                        os.path.join(dst_folder, file))
            count += 1
    return count

real_count = copy_images(os.path.join(source_path, "Au"), os.path.join(dest_path, "real"))
fake_count = copy_images(os.path.join(source_path, "Tp"), os.path.join(dest_path, "fake"))

print(f"Кількість реальних зображень: {real_count}\n"
      f"Кількість фейкових зображень: {fake_count}")
```

ДОДАТОК Б PYTHON 3 КОД TRAIN_MODEL.PY

```

import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import seaborn as sns
from sklearn.utils.class_weight import compute_class_weight

def convert_to_ela_image(image_path, quality=95):
    temp_filename = 'temp_file.jpg'
    ela_filename = 'temp_ela_file.jpg'

    image = cv2.imread(image_path)
    cv2.imwrite(temp_filename, image, [cv2.IMWRITE_JPEG_QUALITY, quality])
    temp_image = cv2.imread(temp_filename)

    ela_image = cv2.absdiff(image, temp_image)
    ela_image = ela_image.astype(np.uint8)

    min_val = np.min(ela_image)
    max_val = np.max(ela_image)
    ela_image = (ela_image - min_val) * 255.0 / (max_val - min_val)

    if os.path.exists(temp_filename):
        os.remove(temp_filename)
    if os.path.exists(ela_filename):
        os.remove(ela_filename)

    return ela_image.astype(np.uint8)

def load_all_images(dataset_path, image_size=(250, 250)):
    real_images = []
    fake_images = []

    real_path = os.path.join(dataset_path, 'real')
    fake_path = os.path.join(dataset_path, 'fake')

    for img_name in os.listdir(real_path):
        img_path = os.path.join(real_path, img_name)
        ela_image = convert_to_ela_image(img_path)
        if ela_image is not None:
            resized = cv2.resize(ela_image, image_size)

```

```

    real_images.append(resized)

for img_name in os.listdir(fake_path):
    img_path = os.path.join(fake_path, img_name)
    ela_image = convert_to_ela_image(img_path)
    if ela_image is not None:
        resized = cv2.resize(ela_image, image_size)
        fake_images.append(resized)

X = np.array(real_images[:2064] + fake_images)
y = np.array([1] * 2064 + [0] * len(fake_images))

X = X.astype('float32') / 255.0

print(f"\nFinal dataset size: {len(X)} images (Real: {len(real_images)}, Fake:
{len(fake_images)})")
return X, y

def create_model(input_shape):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=input_shape),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(256, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')
    ])
    model.compile(optimizer=Adam(learning_rate=0.0001),
                  loss='binary_crossentropy',
                  metrics=['accuracy'])
    return model

def plot_confusion_matrix(y_true, y_pred, filename='conf_matrix_95ela.png'):
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                xticklabels=['Fake', 'Real'],
                yticklabels=['Fake', 'Real'])
    plt.xlabel('Predicted')
    plt.ylabel('True')
    plt.title('Confusion Matrix')
    plt.savefig(filename)
    plt.close()

def main():

    dataset_path = 'clean_dataset'

```

```

image_size = (250, 250)
batch_size = 32
epochs = 50

print("Loading images...")
X, y = load_all_images(dataset_path, image_size)

class_weights = compute_class_weight('balanced', classes=np.unique(y), y=y)
class_weight_dict = {0: class_weights[0], 1: class_weights[1]}
print(f"\nClass weights: {class_weight_dict}")

X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)
print(f"\nTraining samples: {len(X_train)} (Real: {sum(y_train)}, Fake: {len(y_train)-sum(y_train)})")
print(f"\nValidation samples: {len(X_val)} (Real: {sum(y_val)}, Fake: {len(y_val)-sum(y_val)})")

input_shape = (image_size[0], image_size[1], 3)
model = create_model(input_shape)
model.summary()

callbacks = [
    EarlyStopping(patience=5, restore_best_weights=True),
    ModelCheckpoint('model_95ela_2.h5', save_best_only=True)
]

history = model.fit(
    X_train, y_train,
    batch_size=batch_size,
    epochs=epochs,
    validation_data=(X_val, y_val),
    callbacks=callbacks,
    class_weight=class_weight_dict
)

print("\nEvaluating model...")
loss, accuracy = model.evaluate(X_val, y_val)
print(f"Validation Accuracy: {accuracy * 100:.2f}%")

y_pred = (model.predict(X_val) > 0.5).astype(int)

plot_confusion_matrix(y_val, y_pred)

report = classification_report(y_val, y_pred, target_names=['Fake', 'Real'])
print("\nClassification Report:")
print(report)

with open('classification_report_95ela.txt', 'w') as f:
    f.write(report)

if __name__ == "__main__":
    main()

```

ДОДАТОК В PYTHON 3 КОД PREDICT_ELA.PY

```

import cv2
import numpy as np
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt
import os
import sys

# Load your trained model
model = load_model('model_95ela_2.h5')

def convert_to_ela_image(image_path, quality=95):
    temp_filename = 'temp_ela.jpg'

    original = cv2.imread(image_path)
    if original is None:
        return None

    cv2.imwrite(temp_filename, original, [cv2.IMWRITE_JPEG_QUALITY, quality])
    compressed = cv2.imread(temp_filename)

    ela_image = cv2.absdiff(original, compressed)
    ela_image = ela_image.astype(np.uint8)
    min_val, max_val = np.min(ela_image), np.max(ela_image)
    ela_image = (ela_image - min_val) * 255.0 / (max_val - min_val)

    if os.path.exists(temp_filename):
        os.remove(temp_filename)

    return ela_image.astype(np.uint8)

def predict_image(image_path, model, threshold=0.5):
    ela_img = convert_to_ela_image(image_path)

    input_img = cv2.resize(ela_img, (250, 250))
    input_img = input_img.astype('float32') / 255.0
    input_img = np.expand_dims(input_img, axis=0)

    # Make prediction
    prediction = model.predict(input_img)[0][0]
    is_tampered = prediction < threshold

    # Display results
    plt.figure(figsize=(12, 6))

    plt.subplot(1, 2, 1)
    plt.imshow(cv2.cvtColor(cv2.imread(image_path), cv2.COLOR_BGR2RGB))

```

```

plt.title("Original Image")
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(ela_img, cmap='gray')
plt.title(
    f'ELA Image\nPrediction: {'FAKE' if is_tampered else 'REAL'}\nConfidence:
    {max(prediction, 1 - prediction) * 100:.1f}%')
plt.axis('off')

plt.tight_layout()
plt.show()

return is_tampered, prediction

if __name__ == "__main__":
    if len(sys.argv) > 1:
        image_path = sys.argv[1]
        is_fake, confidence = predict_image(image_path, model)
        print(f'\nResult: {'Tampered' if is_fake else 'Authentic'}')
    else:
        print("Usage: python predict_ela.py <image_path>")

```