МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії програмного забезпечення

_____ Євген ДАВИДЕНКО

«___» _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

СИСТЕМА РОЗПІЗНАВАННЯ ФЕЙКОВИХ ЗОБРАЖЕНЬ НА ОСНОВІ НЕЙРОННИХ МЕРЕЖ

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Микита КОЛЕСНІКОВ**

«__» _____ 2024 р.

Керівник роботи

канд. техн. наук,

доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 2024 р.

Миколаїв – 2024

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Другий (магістерський)
Освітній ступінь	Магістр
Спеціальність	Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії
програмного забезпечення
_____Євген ДАВИДЕНКО
«____»_____2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Колеснікова Микити Олександровича
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи
___Система розпізнавання фейкових зображень на основі нейронних мереж__

Затверджена наказом ЧНУ ім. Петра Могили від «04»вересня 2024 р. №__220__

2. Строк представлення кваліфікаційної роботи «___»_____2024 р.

3. Очікуваний результат роботи та початкові дані якщо такі потрібні
Очікуваним результатом є розробка система розпізнавання фейкових зображень на основі нейронних мереж

4. Перелік питань, що підлягають розробці:

- дослідження предметної області та аналіз існуючих аналогів;
- формування специфікації вимог до системи;

– визначення архітектури для проєктування системи;

– проєктування та моделювання системи;

– тренування моделі ШІ;

– розробка системи;

– тестування роботи системи;

– проведення аналізів результатів роботи системи.

5. Перелік графічних матеріалів
Презентація до КМР

6. Завдання до спеціальної частини

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ *Євген ДАВИДЕНКО*
Особистий підпис

Здобувач _____ *Микита КОЛЕСНІКОВ*
Особистий підпис

Дата видачі завдання « ____»_____20_____р

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Система розпізнавання фейкових зображень на основі нейронної мережі

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КМР	01.09.2024	04.09.2024	виконано
2.	Пошук літератури за темою кваліфікаційної роботи	04.09.2024	09.09.2024	виконано
3.	Складання календарного плану роботи	09.09.2024	14.09.2024	виконано
4.	Аналіз предметної області	14.09.2024	21.09.2024	виконано
5.	Розробка проєктних рішень	22.09.2024	01.10.2024	виконано
6.	Моделювання та конструювання ПЗ	03.10.2024	10.10.2024	виконано
7.	Тренування моделі ШІ	12.10.2024	20.10.2024	виконано
8.	Кодування розробленого ПЗ	21.10.2024	07.11.2024	виконано
9.	Тестування ПЗ, аналіз результатів тестування	08.11.2024	14.11.2024	виконано
10.	Оформлення КМР та презентації КМР	15.11.2024	29.11.2024	виконано
11.	Попередній захист КМР	02.12.2024	02.12.2024	виконано
12	Рецензування	03.12.2024	08.12.2024	виконано
13	Завершення оформлення КМР та презентації	09.12.2024	12.12.2024	виконано
14	Захист КМР	19.12.2024	19.12.2024	

Здобувач _____ **Микита КОЛЕСНІКОВ**

«__» _____ 2024 р.

Керівник роботи

канд. техн. наук,

доцент _____ **Євген ДАВИДЕНКО**

«__» _____ 2024 р.

АНОТАЦІЯ

до кваліфікаційної магістерської роботи

«Система розпізнавання фейкових зображень на основі нейронних мереж»

Здобувач 608 групи: Колесніков Микита

Керівник: канд. техн. наук, доцент Давиденко Євген

Дана робота присвячена розробці системи для надання послуг у формі розпізнавання фейкових зображень на основі нейронних мереж.

Об’єкт дослідження – процес аналізу зображень для виявлення їх можливої нереальності.

Предмет дослідження – методи та засоби реалізації системи для розпізнавання фейкових зображень з використанням нейронних мереж.

Метою роботи є покращення кібербезпеки користувачів вебмережі шляхом розробки ПЗ з використанням нейронних мереж для виявлення зображень в мережі Інтернет, які вводять в оману.

Завдання кваліфікаційної роботи складається з наступних пунктів:

– проведення аналізу необхідних критеріїв інформації, необхідних для пошуку та аналізу фейкових зображень;

– створення датасету для навчання нейронних мереж, який складається з зображень різного рівня складності, змісту, а також аугментований шляхом зміни кольорів, розмиття, повороту тощо зображень;

– аналіз існуючих можливості для навчання нейронних мереж та вибір найкращого методу;

– моделювання ПЗ для перевірки зображень на предмет їх фейковості;

– тренування моделі ШІ з метою виявлення фейкових зображень;

– розробка прототипу програмного забезпечення для виявлення фейкових зображень;

– тестування та аналіз роботи створеного програмного забезпечення.

Кваліфікаційна робота складається зі вступу, 4 розділів, висновків та переліку джерел посилань.

У вступі зазначено актуальність теми, науково-практичне значення, мета, об’єкт та предмет роботи.

У першому розділі проведено огляд та аналіз сучасного стану технологій в даній сфері, а також аналіз існуючих методів і засобів для вирішення завдань кваліфікаційної роботи. Оглянуто різновиди видів нейронних мереж, їх переваги та недоліки.

У другому розділі проведено моделювання системи, з розробкою діаграм, які показують роботу системи.

У третьому розділі було розроблено архітектуру системи, обґрунтовано використання мов програмування, бібліотек, а також вибір датасету для тренування моделі ШІ та проведено розробку UML-діаграм.

У четвертому розділі показано процес тренування моделі ШІ та розробка системи. Наприкінці розділу показано тестування системи.

У висновках проаналізовано результати виконаних робіт по кожному розділу кваліфікаційної роботи.

Кваліфікаційна магістерська робота викладена на 77 сторінках, містить 4 розділи, 36 ілюстрації, 3 таблиць, 20 джерел в переліку посилань.

Ключові слова: *розробка системи, нейронні мережі, штучний інтелект, аналіз зображень, аналіз штучним інтелектом, кібербезпека*.

ABSTRACT

of the Master's Thesis

«A system for recognizing fake images based on neural networks»

Student: Mykyta Kolesnikov

Supervisor: Candidate of Technical Sciences, Associate Professor Yevhen Davydenko

This work is devoted to the development of a system for providing services in the form of recognition of fake images based on neural networks.

The object of research is the process of analyzing images to identify their possible unreality.

The subject of research is methods and means of implementing a system for recognizing fake images using neural networks.

The goal of the work is to improve the cyber security of web users by developing software using neural networks to detect misleading images on the Internet.

The task of the qualification work consists of the following items:

– conducting an analysis of the necessary criteria of information necessary for the search and analysis of fake images;

– creation of a dataset for training neural networks, which consists of images of different levels of complexity and content, as well as augmented by changing colors, blurring, rotating, etc. of images;

– analysis of existing opportunities for training neural networks and selection of the best method;

– software simulation for checking images for their fakeness;

– training of an AI model to detect fake images;

– development of a software prototype for detecting fake images;

– testing and analysis of the work of the created software.

The qualification work consists of an introduction, 4 sections, conclusions and a list of reference sources.

The relevance of the topic, scientific and practical significance, purpose, object and subject of the work are indicated in the introduction.

In the first section, a review and analysis of the current state of technology in this field, as well as an analysis of existing methods and tools for solving the tasks of qualification work, is carried out. Various types of neural networks, their advantages and disadvantages are reviewed.

In the second chapter, the system is modeled, with the development of diagrams that show the operation of the system.

In the third chapter, the system architecture was developed, the use of programming languages, libraries, and the selection of a dataset for training the AI model were justified, and UML diagrams were developed.

The fourth chapter shows the AI model training process and system development. At the end of the chapter, system testing is shown.

In the conclusions, the results of the work performed on each section of the qualification work are analyzed.

The qualifying master's thesis is laid out on 77 pages, contains 4 chapters, 36 illustrations, 3 tables, 20 sources in the list of references.

Keywords: *system development, neural networks, artificial intelligence, image analysis, artificial intelligence analysis, cyber securit*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ – Програмне забезпечення

КМР – Кваліфікаційна магістерська робота

CNN – Convolutional Neural Networks

GAN – Generative Adversarial Networks

RNN – Recurrent Neural Networks

LSTM – Long short-term memory

ЗМІ – Засоби масової інформації

AI – Artificial intelligence

JS – JavaScript

TS – TypeScript

UI – User Interface

API – Application Programming Interface

ТЗ – Технічне завдання

ВСТУП

Актуальність теми.

У сучасному світі потік інформації, який спричинений стрімким розвитком комп'ютерних систем від стаціонарних до портативних, є настільки величезним, що за ним не услідкують навіть сотні тисяч людей. Кожен день в мережу Інтернет завантажується величезна кількість інформації, від корисної до нелегальної та небезпечною. До небезпечної інформації відносяться і фейкові зображення, які можуть використовуватись як з ціллю «тролінгу» інших користувачів, так і з ціллю шантажу, маніпуляції або розповсюдження неправдивої інформації серед користувачів вебпростору. Система розпізнавання фейкових зображення тримає на меті допомогти користувачам розрізняти такі зображення від реальних та допомогти не стати жертвами інших користувачів Інтернету. Для даних цілей використовуються навчені на наборах інформації нейронні мережі, які позбавлені суб'єктивності людського ока та оперують лише вичисленнями та реальними фактами, артефактами та іншої інформацією на зображеннях.

Науково-практичне значення.

На данному етапі розвитку цивілізації кібербезпека грає одну з найважливіших ролей в будь-якому суспільстві, саме тому її потрібно розвивати будь-якими можливими методами. Навчання та подальше використання нейронної мережі набагато спрощує розпізнавання неправдивості візуальної частини інформації, що дозволяє убезпечити користувачів мережі Інтернет від інших користувачів з поганими намірами, а можливість подальшого розвитку нейронних мереж на новій інформації дозволяє не стояти на місці та адаптуватись до нових викликів та виявляти більш якісні фейкові зображення.

Метою кваліфікаційної роботи є покращення кібербезпеки користувачів вебмережі шляхом розробки ПЗ з використанням нейронних мереж. Дане програмне забезпечення дозволить з великою вірогідністю визначати неправдиві зображення, шляхом повного аналізу їх змісту, що забезпечити більшу безпеку користувачів під час користування мережою Інтернет.

Об'єктом кваліфікаційної роботи є процес аналізу зображень для виявлення їх можливої фейковості.

Предметом дослідження є методи та засоби реалізації системи для розпізнавання фейкових зображень з використанням нейронних мереж.

Для досягнення визначеної мети необхідно вирішити такі ***завдання***:

– провести аналіз необхідних критеріїв інформації, необхідних для пошуку та аналізу фейкових зображень;

– створити датасет для навчання нейронних мереж, який має складатись з зображень різного рівня складності, змісту, а також аугментований шляхом зміни кольорів, розмиття, повороту тощо зображень;

– проаналізувати існуючі можливості для навчання нейронних мереж та вибір найкращого методу;

– змоделювати ПЗ для перевірки зображень на предмет їх фейковості;

– натренуванти модель ШІ з метою виявлення фейкових зображень;

– розробити прототип програмного забезпечення для виявлення фейкових зображень;

– протестувати та проаналізувати роботу створеного програмного забезпечення.

Сфера застосування результатів

Результати роботи можуть бути використані в різних сферах, де важлива автентичність та достовірність візуальних даних: журналістика та ЗМІ, соціальні мережі, сфера кібербезпеки, судова сфера, маркетингова сфера тощо.

Апробація результатів КМР відбулась під час XXVII Всеукраїнської науково-практичної конференції «Могилянські читання – 2024», Миколаїв, 06-10 листопада, 2024 р. (Додаток А).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Структурні і функціональні особливості об’єкта дослідження

Структурні особливості об’єкта дослідження:

– основною структурною одиницею системи є модель нейронної мережі, яка може використовувати різні архітектури для розпізнавання фейкових зображень. Серед архітектур можна виділити Convolutional Neural Networks (CNN); Generative Adversarial Networks (GAN) та Recurrent Neural Networks (RNN);

– для тренування нейронної мережі використовується датасет, який включає в себе велику кількість візуальних даних, а саме реальні та фейкові зображення;

– перед тренування нейронної мережі виконується попередня обробка вхідних зображень, що включає в себе нормалізацію зображень, їх масштабування тощо;

– у системі використовується метрика точності прогнозу, яка показує результат роботи моделі;

– для взаємодії системи з користувачем використовується графічний інтерфейс вебзастосунку, який дозволяє авторизуватись в застосунок, завантажувати зображення, отримувати результати аналізу, перевіряти результати старіших аналізів зображень.

Функціональні особливості об’єкта дослідження:

– система розділена на модулі, що дозволяє працювати окремо з кожним з них у випадку необхідності їх заміни, оновлення тощо;

– основна функція системи – визначати, чи завантажене користувачем зображення є фейковим, використовуючи попередньо навчену модель нейронної мережі;

– заплановано, що система матиме змогу донавчати модель за допомогою нових даних, отриманих під час роботи з користувачами (а саме використовуючи завантажені ними зображені).

1.2 Огляд і аналіз сучасного стану інформаційних технологій у даній предметній області

На даний момент сфера кібербезпеки, яка пов’язана з роботою з фейковою інформацією розвивається стрімкими кроками, створюючи нові інструменти та рішення, які дозволяють користувачам інтернету користуватись тими чи іншими методами виявлення фейкових зображень. Такі інструменти різняться за використанням ШІ від класичних алгоритмів комп’ютерного зору до глибинних нейронних мереж, які використовують архітектури CNN чи GAN.

Аналіз існуючих аналогів

NVIDIA's Fake Image Detection System

Одним із провідних гравців у галузі є компанія NVIDIA (рис. 1.1), яка розробляє інструменти для виявлення підроблених зображень. Компанія використовує свої потужні GPU та фреймворки для машинного навчання для створення високоточних моделей. Особливістю підходу NVIDIA є використання Convolutional Neural Networks (CNN), що спеціалізуються на аналізі пікселів зображення для виявлення найдрібніших аномалій, характерних для фейкових зображень.

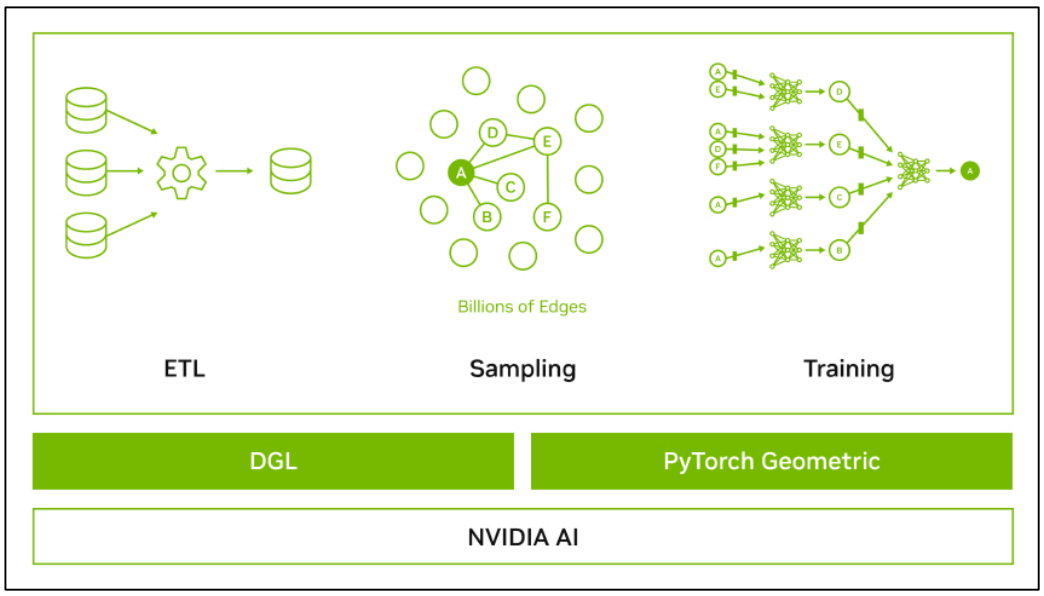


Рисунок 1.1 – NVIDIA AI

Основні особливості:

– використання CNN для розпізнавання структурних аномалій у зображеннях;

– можливість масштабування системи для обробки великих масивів зображень;

– інтеграція з хмарними сервісами для забезпечення масштабованості.

Deepware Scanner (рис. 1.2)

Це застосунок для виявлення підроблених зображень і відео, створених за допомогою технології DeepFake. Інструмент використовує штучний інтелект для аналізу аудіовізуальних файлів і виявлення модифікацій, характерних для DeepFake. Система здатна аналізувати відео та зображення, використовуючи CNN для детектування текстурних аномалій, а також фонові зміни.

Рисунок 1.2 – Deepware

Особливості:

– оцінка глибинних моделей для визначення змін в оригінальних зображеннях;

– аналіз зображень на предмет реалістичності текстур та природності кольорів;

– підтримка аналізу як зображень, так і відео.

Sensity AI (рис. 1.3)

Це платформа для виявлення медіа-підробок, що надає інструменти для автоматичного аналізу зображень і відео на предмет модифікацій, виконаних за допомогою нейронних мереж. Ця система використовує машинне навчання для виявлення аномалій у зображеннях, таких як неприродні контури або зміни в текстурі шкіри.

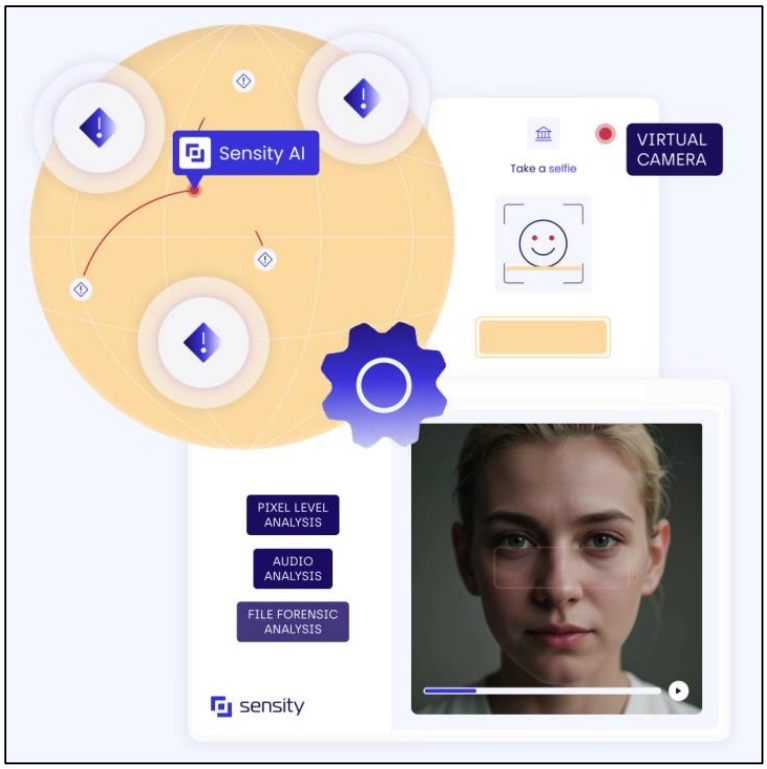


Рисунок 1.3 – Sensity AI

Особливості:

– аналіз зображень у реальному часі з високою точністю;

– інтеграція з медіа-платформами та соціальними мережами для масової перевірки контенту;

– виявлення підробок на основі глибинних ознак, що характерні для фейкових зображень.

Аналіз ефективності та підходів

Нейронні мережі для розпізнавання зображень

Усі згадані аналоги використовують різні архітектури нейронних мереж, серед яких домінують Convolutional Neural Networks та Generative Adversarial

Networks. CNN дозволяє системам ідентифікувати специфічні ознаки фейкових зображень, такі як неприродні текстури або аномальні контури. Деякі системи використовують CNN, щоб зосередитися на локальних ознаках, а GANs допомагають генерувати фейкові зображення для тренування систем.

Покадровий аналіз та аналіз у реальному часі

Такі інструменти, як Deepware Scanner, використовують кадровий аналіз для виявлення підробок у відео. Це підвищує точність системи, дозволяючи виявити навіть найменші зміни між кадрами або в конкретних кадрах. Однак це вимагає значних обчислювальних ресурсів і часу для аналізу, що може стати проблемою при масштабних завданнях.

Інтеграція з іншими сервісами

Платформи типу Sensity AI пропонують інструменти для інтеграції з соціальними мережами та іншими медіа-платформами, що дозволяє автоматизувати процес виявлення фейкових зображень у потоках новин або користувацькому контенті. Це корисно в контексті боротьби з дезінформацією та забезпеченням надійності інформаційних потоків.

Переваги та недоліки існуючих аналогів

Переваги:

– висока точність. Використання глибинних нейронних мереж дозволяє досягати високої точності виявлення фейкових зображень, особливо у випадку складних маніпуляцій, таких як зміна текстур або контурів облич;

– аналіз у реальному часі. Деякі системи, як Sensity AI, дозволяють проводити аналіз зображень у реальному часі, що є критично важливим для соціальних мереж;

– інтеграція з платформами. Можливість інтеграції з існуючими медіа-системами забезпечує масовий аналіз даних у великих масштабах.

Недоліки:

– висока вартість ресурсів. Використання глибинних нейронних мереж та обчислювальних потужностей вимагає значних ресурсів, особливо при аналізі великих обсягів даних;

– вразливість до нових видів підробок. Фейкові зображення, створені новими версіями GANs або іншими алгоритмами, можуть бути важче розпізнані, що вимагає постійного вдосконалення систем;

– обмеженість щодо різноманіття контенту. Більшість систем спеціалізуються на розпізнаванні підробок лише в певних категоріях (наприклад, обличчя), що може обмежувати їхню ефективність для інших типів зображень.

1.3 Огляд і аналіз існуючих методів і засобів вирішення завдань кваліфікаційної роботи магістра

Для вирішення завдань КМР використовуються методи і засоби, які дозволяють виявляти фейки серед зображень, сюди входять глибинні нейронні мережі, методи комп’ютерного зору тощо.

Методи розпізнавання фейкових зображень

Convolutional Neural Networks (CNN)

Одним із найбільш поширених методів є згорткові нейронні мережі (CNN) (рис. 1.4). Вони здатні виділяти специфічні ознаки зображень за допомогою згорток і шарів pooling, що дозволяє виявляти тонкі текстурні аномалії, які можуть свідчити про підробку.

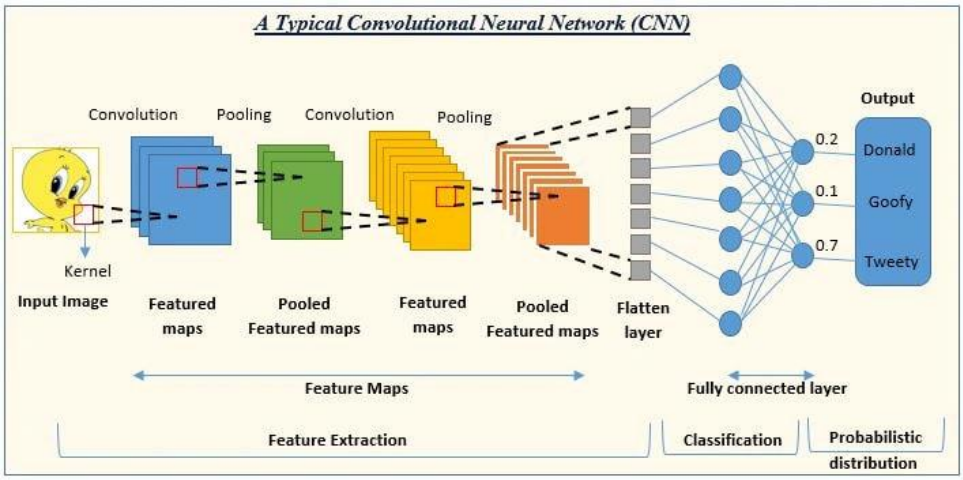


Рисунок 1.4 – Структура роботи CNN

Переваги:

– автоматичне виділення ознак. CNN автоматично навчається виділяти важливі характеристики, такі як межі, текстури та кольорові аномалії;

– гнучкість. CNN може бути застосованою до різних типів зображень, включаючи складні об'єкти та фейкові маніпуляції.

Недоліки:

– вимогливість до обчислювальних ресурсів. Глибинні CNN потребують значних обчислювальних ресурсів для навчання;

– чутливість до якості даних. Мережі можуть бути неефективними, якщо якість вхідних даних низька.

Generative Adversarial Networks (GANs)

Це нейронні мережі, що використовуються як для генерації фейкових зображень, так і для їхнього розпізнавання. GAN складається з двох частин: генератора, що створює фейкові зображення, та дискримінатора, який намагається їх виявити (рис. 1.5). Цей підхід забезпечує більш високу точність у боротьбі з фейками, оскільки мережі навчаються одночасно і створювати, і розпізнавати підробки.

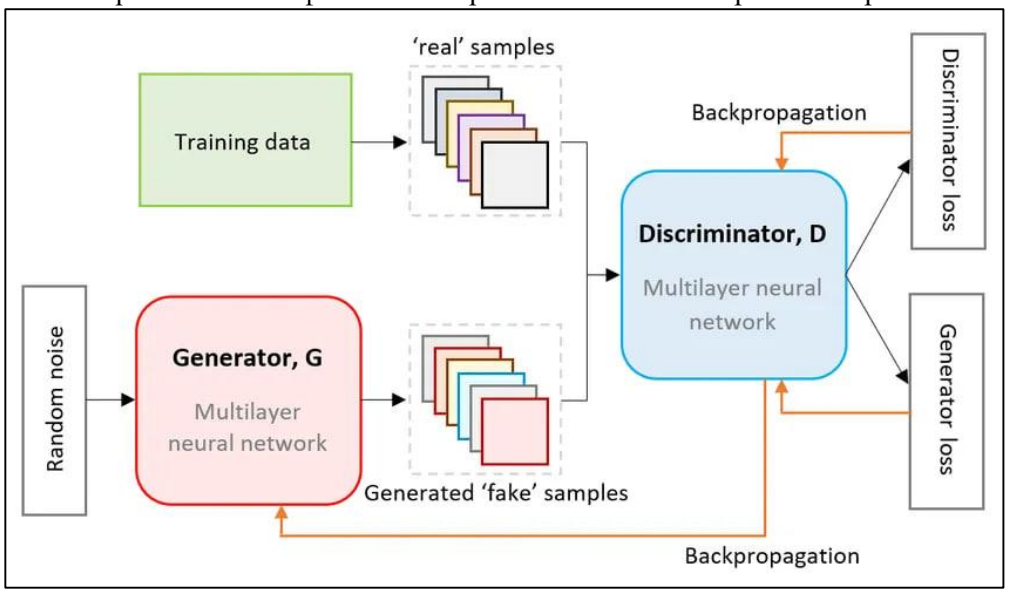


Рисунок 1.5 – Структура GAN

Переваги:

– підвищена точність. GANs здатні навчитися розпізнавати підробки, генеруючи та імітуючи їх;

– адаптивність. Мережі можуть адаптуватися до нових типів фейкових зображень.

Недоліки:

– ресурсовимогливе навчання. Навчання GANs є складним і потребує значної кількості даних та обчислювальних ресурсів;

– складність моделювання. GAN може навчитися вводити в оману дискримінатор, що робить виявлення фейків менш точним.

Рекурентні нейронні мережі (RNN) та LSTM

Для аналізу відео або послідовностей кадрів у відео використовуються Рекурентні нейронні мережі (RNN) (рис. 1.6) та Long Short-Term Memory (LSTM) (рис. 1.7). Вони дозволяють враховувати часові взаємозв'язки між кадрами, що допомагає виявляти фейкові зміни, які можуть бути непомітними на окремих кадрах.

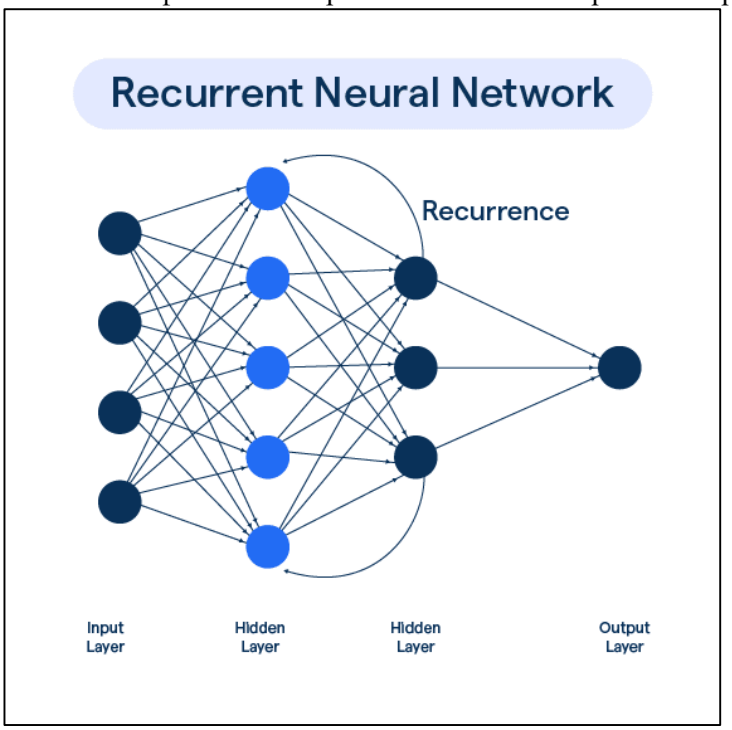


Рисунок 1.6 – Структура RNN

Переваги:

– врахування контексту. RNN та LSTM можуть обробляти послідовності даних, зокрема відео, що дозволяє враховувати зміни в часі;

– застосування до відео-аналітики. Ці мережі ефективні для виявлення фейків у відеоматеріалах.

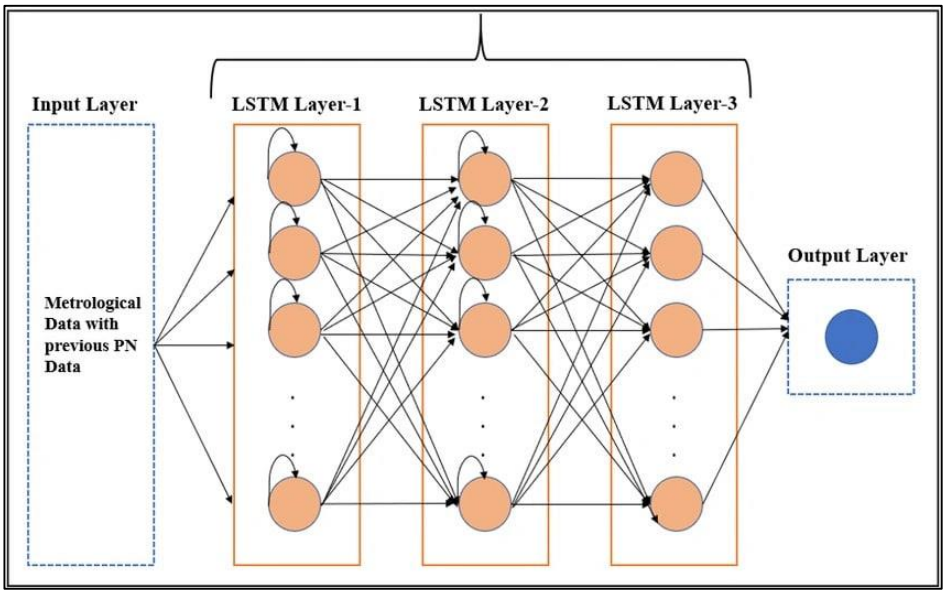


Рисунок 1.7 – Структура LSTM

Недоліки:

– тривалий процес навчання. Такі мережі потребують значного часу та ресурсів для навчання;

– проблеми з довготривалою залежністю. Навіть LSTM може стикатися з проблемами при обробці довгих послідовностей даних.

Інші підходи та технології

Аналіз метаданих

Метадані зображень (EXIF-дані) (рис. 1.8) можуть містити інформацію про зміни, зроблені над зображенням, такі як редагування, зміна розмірів або форматування.

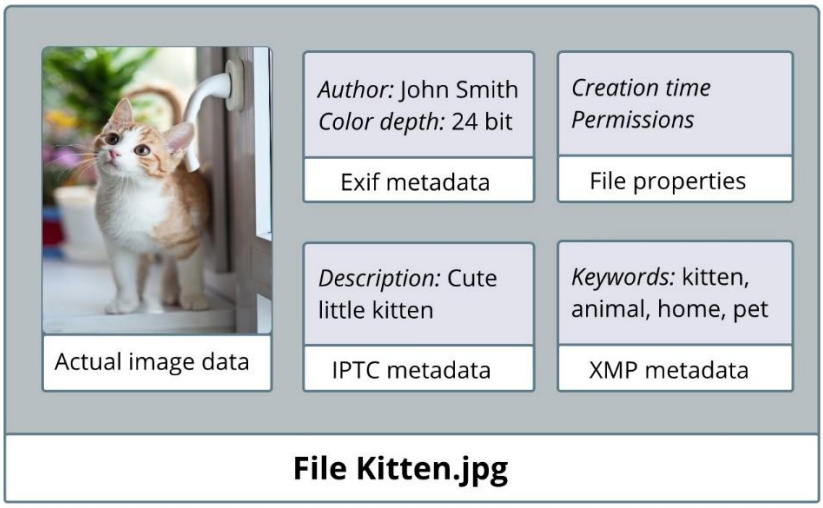


Рисунок 1.8 – Метадані зображення, які не видні користувачеві

Аналіз таких даних може допомогти виявити підробку без необхідності складного комп'ютерного аналізу зображення.

Переваги:

– швидкість аналізу. Аналіз метаданих не вимагає значних ресурсів і може бути виконаний дуже швидко;

– простота реалізації. Метод не потребує складних алгоритмів або нейронних мереж.

Недоліки:

– низька точність. Якщо метадані були видалені або змінені, цей метод не буде ефективним;

– не виявляє глибокі фейки. Метадані не завжди дають змогу виявити складні маніпуляції зі зображеннями.

Методи обробки зображень

Класичні методи обробки зображень, такі як аналіз частотних компонентів, оцінка колірних моделей або аналіз текстур, можуть використовуватися для виявлення аномалій у зображенні, характерних для підробок.

Переваги:

– легка інтеграція. Ці методи можуть бути швидко інтегровані у систему;

– швидкість обробки. Часто такі методи працюють швидше, ніж нейронні мережі.

Недоліки:

– менша точність. Такі методи менш ефективні для виявлення складних фейкових зображень, зокрема створених за допомогою GANs;

– застарілість для сучасних загроз. З розвитком технологій фейків класичні методи втрачають свою актуальність.

1.4 Обґрунтування та вибір підходів до виконання завдань кваліфікаційної магістерської роботи

На основі аналізу існуючих методів можна зробити висновок, що для розв’язання задач розпізнавання фейкових зображень доцільно використовувати більш новітні методи для створення нейронних мереж. Використання нейронних мереж дозволить:

– автоматично виявляти підробки, аналізуючи великі обсяги зображень;

– покращити точність і швидкість порівняно з традиційними методами аналізу зображень;

– адаптувати систему для виявлення нових типів фейків.

Зокрема, для виконання завдань цієї роботи пропонується використати такі підходи:

– Convolutional Neural Networks (CNN) для детекції аномалій та виявлення підроблених елементів на зображеннях;

– Generative Adversarial Networks (GANs) для тренування системи на нових підробках та адаптації до змін у методах створення фейків;

– класифікаційні алгоритми, такі як дерева рішень, для ідентифікації типів підробок та визначення оптимальних стратегій їх розпізнавання.

1.5 Специфікація вимог до програмного забезпечення

1. ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

1.1 Призначення системи, для якої розробляється програмне забезпечення

Система розпізнавання фейкових зображень призначена для виявлення та ідентифікації фейкових або редагованих зображень. Головною сферою її використання є сфера кібербезпеки.

1.2 Погодження, що ухвалені в програмній документації

Документація погоджується із стандартами ISO/IEC 25010 та методологіями машинного навчання для побудови нейронних мереж. Інтерфейси та архітектура системи базуються на загальноприйнятих протоколах і стандартах для систем розпізнавання зображень.

1.3 Межі проєкту ПЗ

Проєкт обмежений аналізом лише цифрових зображень у форматах JPG, PNG, JPEG. Проєкт не включає в себе аналіз відеофайлів або реальних зображень, отриманих з апаратних пристроїв у режимі реального часу.

2. ЗАГАЛЬНИЙ ОПИС

2.1 Сфера застосування

– вільний доступ для будь-яких користувачів мережі Інтернет;

– платформи новин для перевірки достовірності зображень;

– соціальні мережі.

2.2 Характеристики користувачів

– звичайні користувачі мережі Інтернет;

– фахівці з кібербезпеки;

– журналісти;

– аналітики даних.

2.3 Загальна структура і склад системи

Система складається з:

– нейронної мережі для розпізнавання фейкових зображень;

– бази даних для зберігання результатів перевірок;

– інтерфейсу користувача для взаємодії з системою.

2.4 Загальні обмеження

– система обробляє лише статичні зображення (відсутня підтримка графічного формату GIF та відеоматеріалів);

– максимальний розмір файлу – 50 Мб;

– система може обробляти лише один файл від одного користувача за перевірку.

3. ФУНКЦІЇ СИСТЕМИ

3.1 Функція виявлення фейкових зображень

3.1.1 Опис функції

Система аналізує вхідне зображення за допомогою нейронної мережі та визначає, чи є воно фейковим або справжнім.

3.1.2 Вхідна і вихідна інформація

Вхідна інформація: цифрове зображення у форматі JPG, PNG, JPEG.

Вихідна інформація: результат (справжнє або фейкове зображення) з ймовірністю точності.

3.1.3 Функціональні вимоги

– система повинна аналізувати зображення не більше ніж за 20 секунд;

– точність розпізнавання має бути не менше 80%;

– повідомлення про результат перевірки має містити ймовірність фейкового зображення (наприклад, ймовірність фейку 18%);

– система має виводити на екран критерії, за якими було розпізнано фейковість зображення, наприклад метадані (опціонально).

4. ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Джерела і зміст вхідної інформації (даних)

Джерелами даних є датасети фейкових і справжніх зображень для тренування нейронної мережі.

4.2 Нормативно-довідкова інформація (класифікатори, довідники тощо)

Система має використовувати стандартизовані класифікатори зображень та референсні набори даних для навчання моделі.

4.3 Вимоги до способів організації, збереження та ведення інформації

Дані повинні зберігатися у захищеному сховищі, доступ до якого обмежений лише авторизованими користувачами. Дані повинні бути організовані за категоріями: фейкові та справжні зображення.

5. ВИМОГИ ДО ТЕХНІЧНОГО ЗАБЕЗПЕЧЕННЯ

Для функціонування системи необхідні сервери з високою обчислювальною потужністю та підтримкою GPU для тренування і розгортання нейронних мереж. Можливе використання хмарних сервісів для обчислень.

6. ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

6.1 Архітектура програмної системи

Система базується на модульній архітектурі з поділом на компоненти: нейронні мережі, база даних та інтерфейс користувача.

6.2 Системне програмне забезпечення

Операційні системи з браузером та доступом до мережі Інтернет (Linux, Windows, macOS).

6.3 Мережне програмне забезпечення

Система підтримує мережеві з'єднання для віддаленої обробки даних та доступу до зовнішніх ресурсів.

6.4 Програмне забезпечення ведення інформаційної бази

Система повинна використовувати реляційні бази даних (MySQL, PostgreSQL) для зберігання інформації про зображення та результати аналізу з можливістю їх подальшого редагування, експорту та видалення.

6.5 Мова і технологія розробки ПЗ

Мова програмування — Python (з використанням бібліотек TensorFlow або PyTorch для нейронних мереж).

7. ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

7.1 Інтерфейс користувача

Інтерфейс має бути зрозумілим для користувача та мінімалістичним, з простою формою для завантаження зображень та отримання результатів.

7.2 Апаратний інтерфейс

Система може інтегруватися з камерами або іншими пристроями для завантаження зображень (за потреби).

7.3 Програмний інтерфейс

Система повинна підтримувати API для інтеграції з іншими системами через REST.

7.4 Комунікаційний протокол

Система має використовувати HTTPS для безпечної передачі даних.

8. ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

8.1 Доступність

Система повинна працювати 24/7 з мінімальним часом простою та бути доступною з невеликими обмеженнями для всіх користувачів.

8.2 Супроводжуваність

Система повинна мати докладну документацію для майбутньої підтримки та розвитку.

8.3 Переносимість

Можливість розгортання на різних платформах, локальних серверах або хмарних сервісах.

8.4 Продуктивність

Час аналізу одного зображення для одного користувача не повинен перевищувати 20 секунд.

8.5 Надійність

Система повинна витримувати навантаження одночасної обробки кількох зображень.

8.6 Безпека

Всі дані повинні передаватися в зашифрованому вигляді. Система повинна мати механізми захисту від атак, таких як DDoS.

Висновки до розділу 1

У першому розділі КМР було визначено об’єкт та предмет дослідження, а також визначено структурні і функціональні особливост об’єкта дослідження. Було проаналізовано, що використання нейронних мереж та датасетів для їх тренувань допоможе досягти мети кваліфікаційної роботи.

Далі було проведено огляд і аналіз сучасного стану інформаційних технологій у даній предметній області, шляхом пошуку та аналізу вже існуючих рішень на тему КМР. Було виділено переваги та недоліки даних рішень, що дозволяє сформувати важливі критерії для успішної роботи майбутнього продукту.

Після цього було проведено огляд і аналіз існуючих методі і засобів для вирішення завдань КМР. Тут можна виділити аналіз методів створення нейронних мереж для тих чи інших задач, їх переваги та недоліки. На основі цього аналізу обгрунтовується вибір тих чи інших рішень для вирішення поставлених задач.

Наприкінці розділу було розписано специфікацію вимог програмного забезпечення. Було описано призначення системи, описані особливості документації продукту, описано сферу застсування, можливих користувачів системи, її структуру, а також обмеження. Корокто було розказано про функції майбутньої системи та їх особливості, як наприклад функція виявлення фейкових зображень.

2 МОДЕЛЮВАННЯ СИСТЕМИ

2.1 Функція виявлення фейкових зображень

Функція виявлення фейкових зображень є ключовою функцією розроблюваної системи. Ця функція призначена для ідентифікації візуального контенту, який був штучно створений або зазнав суттєвих маніпуляцій. Система аналізує:

- неприродні особливості текстур і кольорових переходів;
- порушення в геометрії та перспективі об'єктів;
- сліди цифрової обробки та компресії;
- невідповідності в EXIF-даних;
- аномалії в зображеннях облич (для портретних фотографій).

Ця функціональність критично важлива для протидії поширенню візуальної дезінформації, забезпечення цілісності медіа-контенту та підтримки довіри до цифрових зображень.

Технологічна база функції включає в себе:

- глибинні нейромережеві архітектури для аналізу візуальних паттернів;
- алгоритми комп'ютерного зору для виділення специфічних ознак;
- методи спектрального аналізу для виявлення артефактів обробки.

Ці підходи дозволяють провести багаторівневий аналіз зображень та сформувати обґрунтований висновок щодо їх автентичності.

Архітектура системи складається з наступних ключових компонентів:

Модуль підготовки даних:

- стандартизація форматів вхідних зображень;
- застосування методів покращення якості для подальшого аналізу (опціонально).

Модуль перевірки зображення:

- застосування навченої нейромережевої моделі;
- формування фінального вердикту щодо автентичності.

Система управління даними:

– ведення бази зразків фейкових зображень;

– збереження результатів аналізу для подальшого навчання системи.

Інтерфейси взаємодії:

– REST API для інтеграції з зовнішніми системами;

– вебпортал для ручного завантаження та перевірки зображень.

Модуль зворотного зв'язку:

– механізм врахування оцінок користувачів;

– система автоматичного донавчання на нових даних.

Технологічний стек реалізації:

– Python як основна мова розробки;

– Keras, PyTorch та Tensorflow для імплементації нейромережевих моделей;

– OpenCV для обробки зображень;

– React, Angular або Vue.js для розробки користувацького інтерфейсу.

Забезпечення прозорості:

– генерація текстових пояснень щодо виявлених ознак фейковості (опціонально).

Перспективи розвитку:

– розширення функціоналу для аналізу відеоконтенту;

– поступовий розвиток нейромережі для аналізу, з використанням контенту користувачів.

Адаптивність:

– реалізація механізмів онлайн-навчання для пристосування до нових методів фальсифікації.

Ця функція забезпечує потужний інструментарій для виявлення та класифікації фейкових зображень, сприяючи підвищенню довіри до візуальної інформації в цифровому просторі.

2.2 Огляд вимог до інформаційного забезпечення

Джерела інформації

Для ефективної роботи системи виявлення фейкових зображень необхідно інтегрувати різноманітні джерела даних. Ці джерела включають в себе набори реальних та фейкових зображень, їх метадані, а також інформації про існуючі методи створення фейкових зображен. Система повинна мати можливість обробляти зображення різних підтримуваних системою форматів, аналізувати їх метадані та враховувати контекст зображення для підвищення точності аналізу зображення.

Збір даних

Основніс дані для навчання та тестування системи збираються з різних джерел:

– відкриті набори даних з маркованими фейковими та справжніми зображеннями;

– синтетичні дані, створені за допомогою генеративних моделей штучного інтелекту;

– зображення із соціальних мереж та новинних ресурсів;

– спеціалізовані набори даних для конкретних типів фейків (наприклад, deepfakes).

Для кожного типу даних використовуються специфічні методи збору та обробки:

– API соціальних мереж для отримання зображень та метаданих;

– вебскрейпінг для збору даних з новинних ресурсів;

– генеративні моделі штучного інтелекту для створення синтетичних даних.

Сторонні API

Для покращення точності аналізу система може використовувати зовнішні API:

– сервіси перевірки метаданих зображень;
– API для аналізу цифрових відбитків камер;
– сервіси розпізнавання облич для виявлення маніпуляцій з портретами.

Обробка даних

Попередня обробка зображень включає:

– нормалізацію розміру та формату;
– видалення шуму та артефактів;
– покращення контрасту та чіткості.

Для цього використовується бібліотека обробки зображень OpenCV.

Оптимальний формат зберігання даних: зображення зберігаються в форматі, що підтримує швидкий доступ для аналізу. Метадані та результати аналізу зберігаються в реляційній базі даних, такій як MySQL.

2.3 Нормативно-довідкова інформація

Нормативно-довідкова інформація включає дані про характеристики різних типів камер, стандарти обробки зображень та типові ознаки фейкових зображень.

Джерела нормативної інформації

Джерела нормативної інформації включають в себе:

– технічні специфікації виробників камер та графічних редакторів;
– бази даних відомих методів створення фейків.

Класифікація інформації

Система використовує стандартизовані класифікатори для різних типів фейків, методів їх створення та ознак маніпуляцій з зображеннями.

Зберігання нормативних даних

Нормативні дані зберігаються у реляційній базі даних у вигляді структурованих таблиць, що дозволяє швидко отримати доступ до інформації у разі необхідності.

2.4 Організація збереження та ведення інформації

Ефективне управління даними є критичним для функціонування системи розпізнавання фейків. Архітектура системи забезпечує не лише надійне зберігання поточних зображень та результатів аналізу, але й створює основу для постійного вдосконалення алгоритмів через накопичення та аналіз історичних даних.

Структура сховища даних

Для даної системи було обрано MySQL як основну систему управління базами даних. MySQL пропонує оптимальний баланс між продуктивністю, надійністю та простотою використання, що ідеально підходить для обробки різноманітних типів даних, пов'язаних з аналізом зображень.

Для забезпечення швидкого доступу до даних при зростанні обсягів інформації буде впроваджено систему індексації та партиціонування таблиць в MySQL. Це дозволить ефективно працювати з мільйонами записів без втрати швидкодії при виконанні запитів.

Керування життєвим циклом даних

Для оптимізації використання ресурсів буде розроблено систему переміщення неактивних даних у холодне сховище. Старі результати аналізу та рідко використовувані навчальні дані архівуються у стиснутому форматі та зберігаються на окремих серверах, залишаючись доступними для глибокого аналізу чи повторного навчання моделей.

Захист даних

Безпека даних забезпечується комплексним підходом:

– використання SSL/TLS для шифрування даних при передачі;

– застосування вбудованих функцій шифрування MySQL для захисту чутливих даних у базі;

– впровадження системи управління доступом на основі ролей (RBAC) для контролю прав користувачів.

Для гарантування цілісності та доступності даних буде використано комбінацію повних та інкрементальних резервних копій:

– щоденні інкрементальні копії для швидкого відновлення останніх змін;

– щотижневі повні резервні копії для забезпечення цілісності всієї бази даних;

– автоматизоване тестування відновлення даних для перевірки надійності резервних копій.

Така архітектура забезпечує надійну основу для зберігання та обробки даних у розроблюваній системі, дозволяючи ефективно масштабувати систему та підтримувати її постійне вдосконалення.

2.5 Побудова IDEF0 діаграми

IDEF0 – це методологія моделювання процесів, яка використовується для опису функціональних аспектів систем, процесів або організацій. Назва IDEF0 походить від "Integration Definition for Function Modeling" і є частиною сімейства стандартів IDEF, розроблених у 1970-х роках для покращення процесів аналізу, проектування та управління в різних сферах діяльності.

Основні характеристики IDEF0 діаграм:

– функціональна орієнтація. IDEF0 фокусується на функціях або процесах, які виконуються в системі, а не на об'єктах чи даних;

– структурованість. Діаграми представлені у вигляді ієрархії, де вищі рівні показують загальні функції, а нижчі деталізують їхні складові;

– модульність. Кожна функція представлена як окрема блок-схема, що полегшує розуміння та аналіз складних систем.

Основні компоненти IDEF0 діаграми:

– блок (функція). Прямокутник, що представляє конкретну функцію або процес. Кожен блок має унікальний ідентифікатор (наприклад, A-0, A1, A2 і т.д.);

– вхідні дані (Inputs). Стрілки, що входять зліва або зверху блоку, які представляють ресурси або інформацію, необхідні для виконання функції;

– вихідні дані (Outputs). Стрілки, що виходять справа або знизу блоку, які показують результати виконання функції;

– контроль (Controls). Стрілки, що входять зверху блоку, які представляють умови, правила або інструкції, що регулюють виконання функції;

– механізми (Mechanisms/Resources). Стрілки, що входять знизу або зліва блоку, які показують ресурси або засоби, необхідні для виконання функції (наприклад, обладнання, персонал).

Переваги IDEF0 діаграм:

– чіткість та зрозумілість. Графічне представлення дозволяє легко зрозуміти структуру та функціонування системи;

– масштабованість. Можливість деталізації процесів на різних рівнях абстракції;

– уніфікація. Використання стандартної нотації забезпечує уніфікований підхід до моделювання;

– полегшення комунікації. Спрощує обмін інформацією між різними учасниками проекту або організації.

Недоліки та обмеження IDEF0 діаграм:

– складність при великій кількості функцій. Велика система може призвести до дуже складних діаграм, що ускладнює їхнє розуміння;

– не фокусується на даних. IDEF0 більше орієнтований на функції та процеси, ніж на структуру даних, що може вимагати додаткових моделей для повного опису системи;

– вимагає спеціальних навичок. Для ефективного створення та аналізу IDEF0 діаграм необхідно володіти специфічними знаннями та досвідом.

На рисунку 2.1 нижче зображено діаграму IDEF0 для системи розпізнавання фейкових зображень.

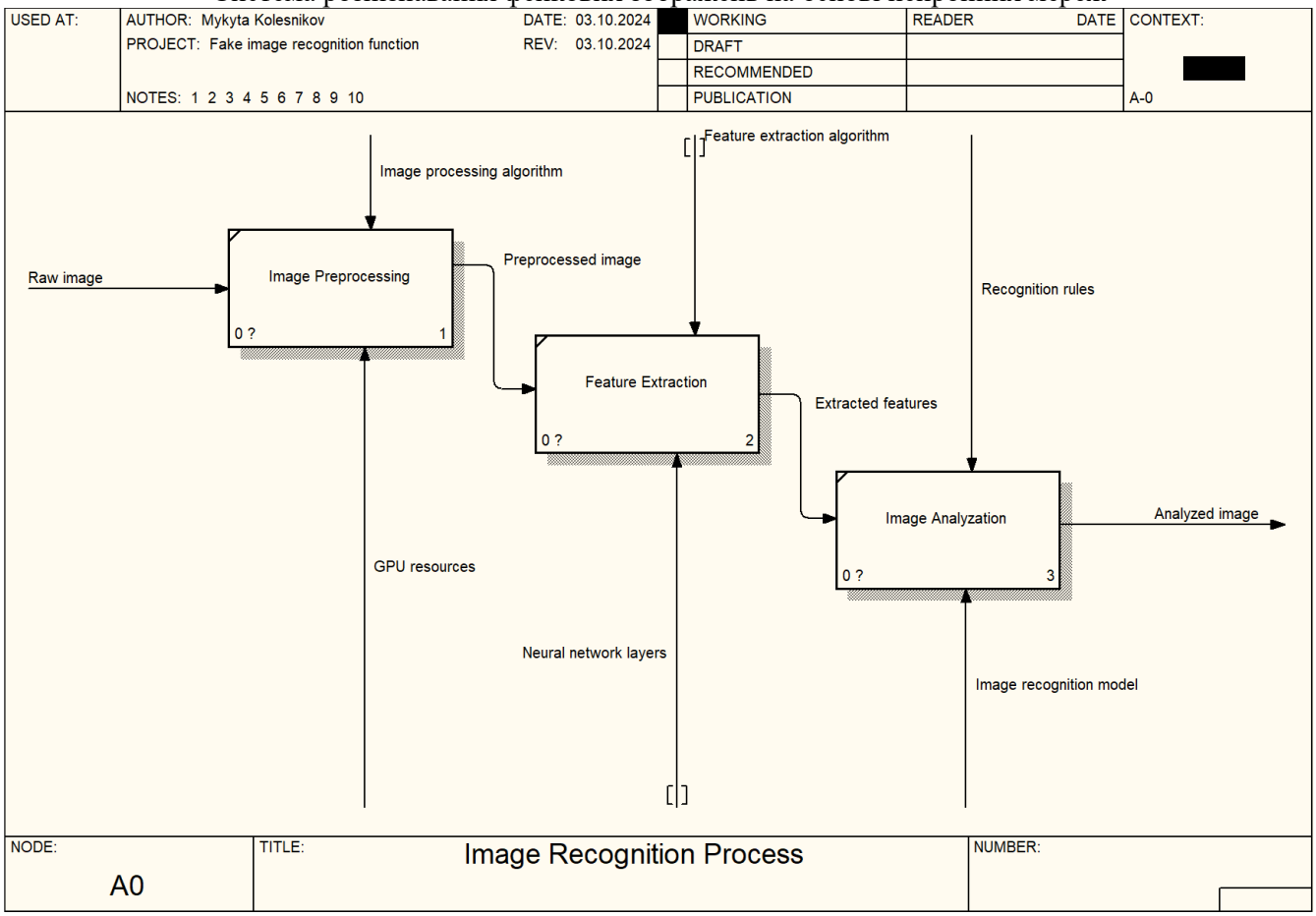


Рисунок 2.1 – Діаграма IDEF0 для системи розпізнавання фейків

На даній діаграмі показано процес розпізнавання та аналізу зображення. На ній зображені наступні елементи системи:

1. Image Preprocessing (попередня обробка картинки) – це перший етап, під час якого вхідна картинка нормалізується під певний стандарт для більш зручної перевірки. Вхідний потік – це сама картинка. Механізм – ресурси графічного процесору серверу. Контроль – алгоритм попередньої обробки зображення. На виході отримуємо стандартизоване зображення.

2. Feature Extraction (виділення особливостей) – на наступному етапі на стандартизованому зображенні знаходяться певні особливості, у вигляді аномалій, артефактів тощо. Вхідний потік – стандартизована картинка. Механізм – шари нейронної мережі. Контроль – алгоритм виділення особливостей. На виході отримуємо знайдені особливості та аномалії.

3. Image analyzation (аналіз картинки) – фінальний етап, на якому після отримання особливостей і артефактів картинки – модель нейронної мережі приходить до висновки, чи є картинка фейком, чи реальною. Вхідний потік – особливості і артефакти картинки. Механізм – модель нейронної мережі. Контроль – правила розгляду та аналізу чистоти картинки. На виході отримуємо проаналізоване зображення.

Потік інформації є безперервним між блоками. Як тільки інформація виходить з одного блоку – вона входить в наступний.

2.6 Побудова DFD діаграми

DFD (Data Flow Diagrams) – це інструмент моделювання, який використовується для візуалізації потоку даних у системі, що відображає, як дані входять, обробляються і виходять з системи. На відміну від IDEF0, який фокусується на функціях, DFD акцентує увагу на даних і процесах, які працюють з цими даними. DFD є одним із найбільш використовуваних методів для опису інформаційних систем.

Основні елементи DFD

– Процеси (Processes). Це основні операції або функції, які виконуються над даними. В DFD вони представлені у вигляді кола або прямокутника з заокругленими кутами. Кожен процес має унікальний номер і назву, яка чітко описує, що саме робить процес;

– Потоки даних (Data Flows). Це стрілки, що показують рух даних між процесами, зовнішніми сутностями та сховищами даних. Вони можуть представляти передання даних між елементами, наприклад, від клієнта до процесу або від процесу до бази даних;

– Сховища даних (Data Stores). Це місця, де зберігаються дані для подальшої обробки. В DFD вони зазвичай зображаються у вигляді двох паралельних ліній. Сховища можуть бути базами даних, файлами чи будь-якими іншими засобами для збереження інформації;

– Зовнішні сутності (External Entities). Це зовнішні об'єкти або організації, з якими взаємодіє система. Вони не є частиною системи, але можуть надсилати або отримувати дані. Зображуються у вигляді прямокутників.

Рівні деталізації DFD

– Контекстна діаграма (Context Diagram). Це найвищий рівень DFD, що представляє всю систему як один процес. Вона показує зовнішні сутності, які взаємодіють із системою, та потоки даних між ними;

– Рівень 1 (Level 1 DFD). Цей рівень деталізує основні процеси всередині системи, показуючи їх взаємодію з даними та зовнішніми сутностями. На рівні 1 система розбивається на кілька підпроцесів;

– Нижчі рівні (Level 2, Level 3, і т.д.). Деталізують кожен підпроцес, якщо необхідно показати ще більше деталей і зрозуміти, як працюють окремі компоненти системи.

Правила побудови DFD

– чіткість потоків даних. Потоки даних повинні чітко показувати рух інформації, і кожна стрілка повинна представляти реальний потік даних;

– односторонній напрямок потоків. Потоки даних мають один напрямок, щоб показувати, звідки дані надходять і куди направляються;

– унікальні назви процесів. Кожен процес повинен мати унікальну назву, яка чітко описує його функцію;

– зовнішні сутності не повинні взаємодіяти безпосередньо. Потоки даних не повинні йти від однієї зовнішньої сутності до іншої без участі процесу.

На рисунку 2.2 зображено DFD діаграму системи розпізнавання фейкових зображень.

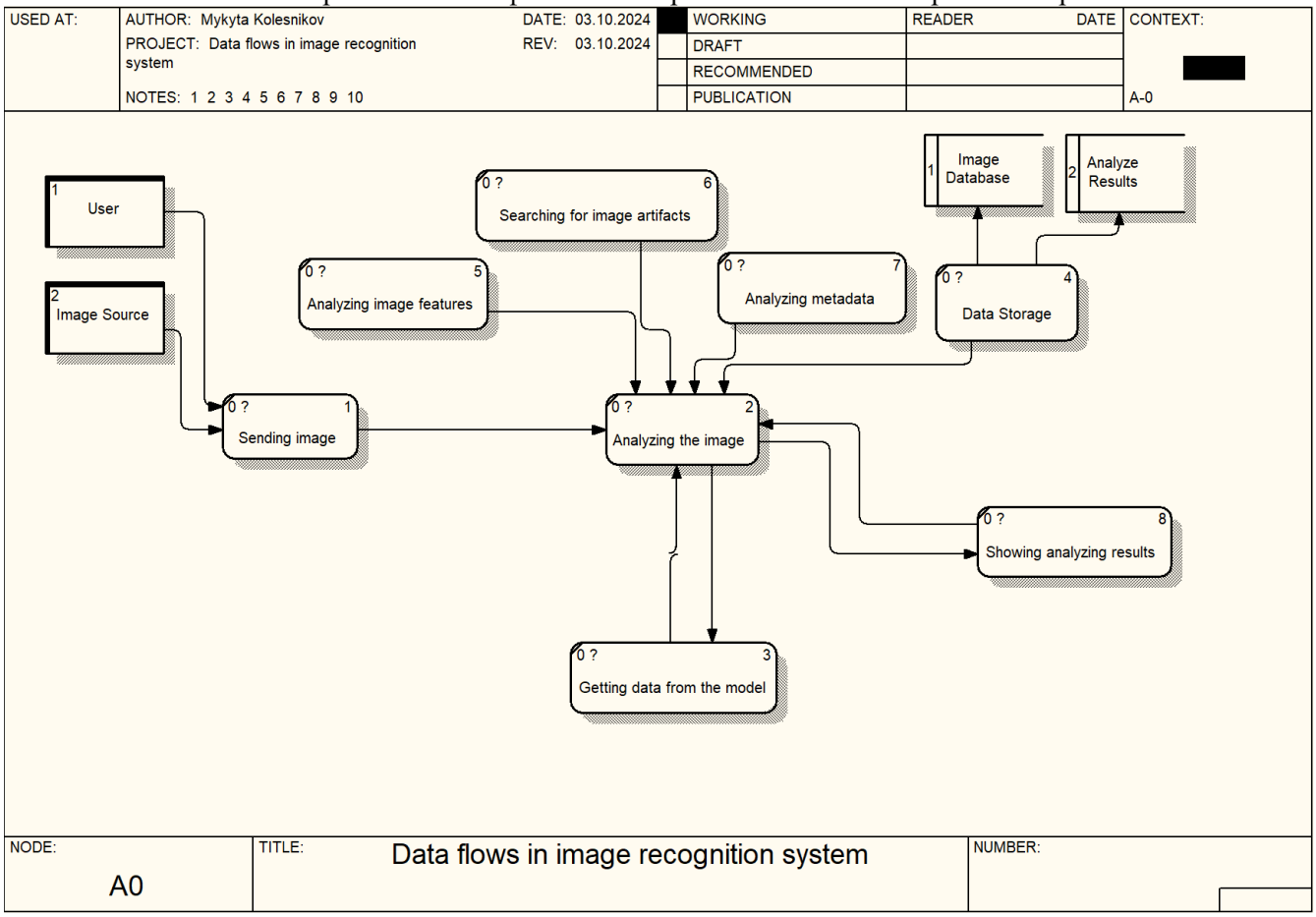


Рисунок 2.2 – DFD-діаграма системи розпізнавання фейків

User (користувач). Разом з Image Source є початковими елементами діаграми, які ведуть до першого процесу. Він відправляє зображення на аналіз до системи.

Процес 1. Sending Image (відправка зображення). Користувач завантажує зображення на сервер для подальшої перевірки його на реальність.

Процес 2. Analyzing the image (аналіз зображення). Є головним блоком системи, оскільки взаємодіє з іншими процесами та в кінці роботи видає результат. Цей блок є самим процесом аналізу зображення.

Процес 3. Getting data from the model (отримання даних від моделі). Під час даного процесу отримується інформація від моделі нейронної мережі, що використовується для аналізу зображень.

Процес 4. Data Storage (сховище даних). Тут відбувається пошук, чи не відбувався аналіз даного зображення до цього моменту, а також його збереження для майбутнього покращення моделі.

Процес 5. Analyzing image features (аналіз особливостей зображення). Під час даного процесу відбувається аналіз зображення на предмет певних особливостей, що відрізняють його від інших зображень.

Процес 6. Searching for image artifacts (пошук артефактів зображення). На даному етапі відбувається пошук артефактів на зображенні, які не є природніми або через які зображення виглядає дивно. Також відбувається пошук можливих місць редагування зображення.

Процес 7. Analyzing metadata (аналіз метаданих). Під час цього етапу відбувається аналіз метаданих зображення, для отримання більшої інформації про нього, а також чи було воно редаговане.

Процес 8. Showing analyzing results (показ результатів аналізу). Є фінальним процесом, під час якого на екран користувача виводиться інформація про можливість фейковості зображення.

Image Database (база даних зображень) та Analyze Results (результати аналізу). Вони є зовнішніми джерелами інформації. В базі даних зберігаються зображення, які вже проходили до цього перевірку та які будуть використані для подальшого навчання моделі нейронної мережі. Також в них зберігаються результати аналізів зображень.

Потоки даних:

- відні потоки: Надсилання зображення користувачем;
- вхідні потоки: Повідомлення про можливість фейку зображення;
- вутрішні потоки: Поетапний аналіз зображення.

Ця діаграма DFD показує, як різні компоненти взаємодіють у процесі обробки користувацького запиту та надання рекомендацій на основі аналізу характеристик ПК і вимог ігор.

Висновки до розділу 2

Другий розділ КМР було присвячено моделюванню розроблюваного ПЗ для аналізу зображень та пошуків фейків серед них. Було описано основну функцію

пошуку фейків серед зображень, вимоги до неї, опис її роботи та яким чином матиме вигляд архітектура ПЗ.

Далі було сформовано вимоги до інформаційного забезпечення, нормативно-довідкову інформацію, а також розглянуто організацію збереження та ведення інформації розроблюваного ПЗ. Було визначено, що організація бази даних відбуватиметься за допомогою реляційних баз даних, а саме MySQL, що дозволить ефективно звертатись за інформацією в необхідний момент.

В кінці розділу було сформовано IDEF0 та DFD діаграму, на яких показано потоки інформації, а також процеси, які присутні у розроблюваній системі. Було визначено вхідні та вихідні дані, а також механізми роботи програмного забезпечення. На діаграмі IDEF0 було описано елементи попередньої обробки зображень, виділення особливостей і артефактів із зображень, а також сам аналіз зображення. На діаграмі DFD було показано перехідні процесу в системі, що розробляється, а саме повний цикл від відправки самого зображення на перевірку користувачем, до його перевірки, збереження в базі даних та повернення до користувача.

Сформовані діаграми та вся супутня інформація дозволяють ефективно змоделювати системи, опираючись на її майбутні елементи, функціонал та процес роботи кожного елементу системи.

3 АРХІТЕКТУРА, МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Розробка архітектури програмного забезпечення

Архітектура для системи розпізнавання фейкових зображень, що розробляється, використовує модульний підхід. Дане рішення дозволяє розділити систему на декілька ключових компонентів, які ефективно працюють над власними задачами, не втручаючись в простір один одного, а обмінюючись по необхідності даними. Це дозволяє інтегрувати нейронні мережі із сучасними вебтехнологіями для створення функціональної системи.

Модуль обробки даних

Даний модуль відповідає за завантаження зображень та їх попередню обробку. Серед функцій даного модулю можна виділити нормалізацію та масштабування зображень, а також підлаштування зображень під один спільний формат для перевірки, щоб мінімізувати проблеми, які пов’язані з різницею форматів зображень.

Модуль нейронної мережі

Даний модуль реалізує основну функціональність для розпізнавання фейкових зображень. Він містить в собі модель нейронної мережі, яка натренована з використанням бібліотек Keras та Tensorflow. В свою чергу нейронна мережа використовує архітектуру CNN, оскільки вона є найбільш підходящою для виконання поставлених завдань, а саме пошуку аномалій та графічний артефактів на зображенням, для подальшого прийняття рішень щодо нереальності даних зображень. Модель навчена на тренувальній виборці, яка складається з 70+ тисяч фейкових та 70+ тисяч реальних зображень. В подальшому планується донавчати модель нейронної мережі, що дозволить зробити даний модуль, а також нейронну мережу більш точної в своїх вимірах.

Модуль серверної частини

Даний модуль забезпечує взаємодію між клієнтською частиною та модулем нейронної мережі. Сервер обробляє запити у вигляді зображень, які надсилає йому користувач на перевірку, передає дані зображення на нормалізацію та їх аналіз і в кінці циклу повертає результати про реальність даного зображення. Також даний модуль зберігає логи та результати аналізу в базу даних, що дозволяє користувачеві в будь-який момент звернутись по минулі результати перевірок його зображень.

Модуль клієнтської частини

Даний модуль відповідає за те, що бачить користувач. Він забезпечує зручний інтерфейс для взаємодії користувача із системою. Використовуючи вебзастосунок, користувач може завантажити зображення, отримати результати аналізу або продивитись минулі результати аналізів його зображень.

Модуль зберігання даних

Даний модуль відповідає за зберігання зображень, логів та аналізів зображень на предмет їх фейковості. Даний компонент забезпечує зберігання історії використання системи індивідуально для кожного користувача, а також дозволяє використати збережені дані для подальшого донавчання моделі.

Взаємодія даних модулів відбувається наступним чином:

– користувач завантажує зображення через вебзастосунок;

– клієнтський модуль передає зображення на сервер з використання API;

– сервер обробляє запит та передає зображення в модуль нейронної мережі;

– модель нейронної мережі виконує аналіз зображення та на його основі класифікує зображення як реальне, або фейкове;

– результати передаються на сервер, зберігають в базі даних та відправляються на клієнтську частину для можливості перегляду результатів користувачем.

3.2 Вибір технологій, мов програмування та датасету

Для розробки системи розпізнавання фейкових зображень з використанням нейронних мереж, було використано наступні технології:

- HTML;
- JS;
- TS;
- CSS;
- Python;
- Angular;
- Node.js;
- Keras;
- Tensorflow.

Обрання технологій для необхідної системи, що розробляється – є необхідною складовою для створення успішного робочого рішення, оскільки це в першу чергу впливає на архітектуру та створення самої системи.

Кожна з обраних технологій або мов програмування пропонують свої власні рішення або інструменти для вирішення поставлених задач в найбільш ефективний спосіб, який буде зручно використовувати розробнику під час розробки системного рішення.

Тобто для відносно невеликих проєктів гарним рішенням було б обрання відносно невеликого стеку технологій, які б дозволяли впровадити готове рішення в найкоротші терміни, хоч і пожертвувавши майбутньою швидкістю роботи застосунку. Але б і ризики повільної роботи системи були б нівельовані, оскільки розроблюваний продукт мав би відносно невеликий розмір, що саме по собі дозволяло би використовувати набагато менше ресурсів та надавати більшу продуктивність роботи системи. В іншому ж випадку, якщо розроблювана система є великою за обсягом – то ефективно було б використовувати рішення, які хоч і є складнішими за своїм розумінням та будовою, але надавали би набагато кращу

оптимізацію, а також простір для майбутніх надбудов у застосунку, у вигляді створення нового функціоналу, а також оновлення та розширення вже існуючого.

Також бажано підбирати технології та мови програмування з врахуванням архітектури системи. Наприклад якщо планується створення відносно невеликої за архітектурою системи з мінімальною підтримкою в майбутньому, то тоді краще відмовитись від великих технологій, фреймворків або тяжких в своєму розумінні мов програмувань. І навпаки, якщо архітектура системи є великою, а також для її створення необхідні комплексні програмні рішення, а також рішення, які в майбутньому дозволять створювати надбудови в системі – то кращим рішенням вже буде використання технологій складніше, які дозволяти досягти необхідних результатів зі створення масштабної архітектури системи.

І звісно, однієї з основних причин обґрунтування вибору тих чи інших технологій для розробки програмного рішення – є час, який встановлений для розробки продукту. Але тут також є два варіанти розвитку подій. Або наприклад використовувати технології, які дозволяють швидке розгортання системи, яку в подальшому буде важче підтримувати (такий підхід використовують, якщо система не планується в майбутньому з певними надбудовами та розширенням функціоналу), або використовувати програмні рішення, які хоч і не дозволять доволі швидко розгорнути програмний продукт, але дозволять в подальшому спокійно вносити в нього зміни, оскільки цьому буде сприяти його власна архітектура.

Тому з цих роздумів можна виявити, що основою будь-якого успішного продукту – є обрання правильних технологій, мов програмування, а також в даному випадку і датасету для тренування моделі штучного інтелекту, адже чим більш підходящим є такий набір даних – тим ефективнішою буде робота нейронних мереж на базі застосунку.

Серед основних для використання мов програмувань було обрано Python, JavaScript та TypeScript.

Python (рис. 3.1) було обрано для розробки модуля машинного навчання через низку причин. Серед цих причин – його простий синтаксис, наявність широкої підтримки від інших розробників на форумах (оскільки дана мова програмування є однією з найпоширеніших у світі), а також простота використання.

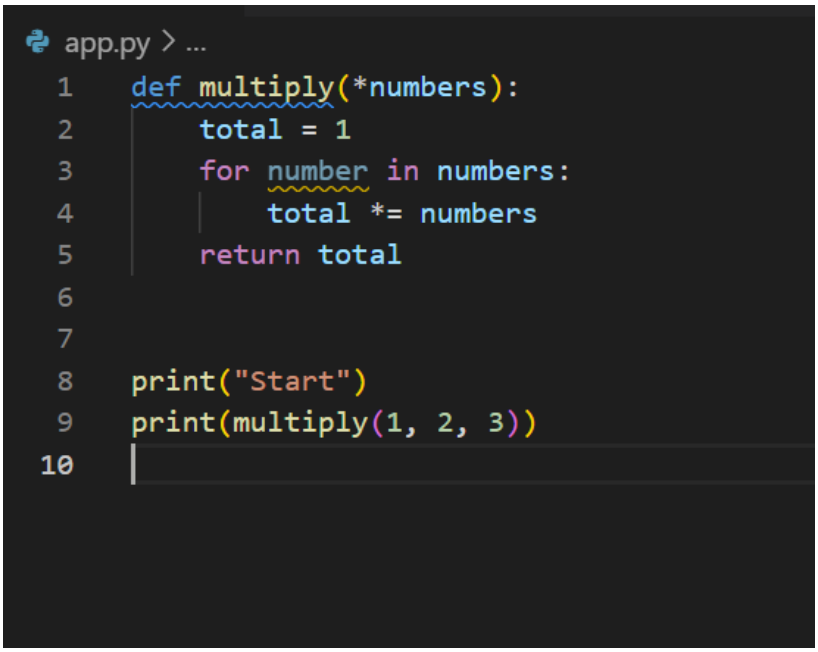


Рисунок 3.1 – Синтаксис Python

Також великою перевагою даної мови програмування є величезна кількість розроблених для неї бібліотек, в які входять різноманітні рішення від математичних операцій, до навчання моделей нейронних мереж. Більшість таких бібліотек мають добре створену документацію, яка дозволяє швидко ввести розробників в курс діла, а також допомагає влитися їм швидко в розробку або підтримку систем.

JavaScript (рис. 3.2) є однією з найбільш поширених мов програмування у світі. Вона дозволяє створювати інтерактивні та динамічні вебзастосунки та є доволі універсальною, оскільки може використовуватись як з клієнтської, так і з серверної сторони, як наприклад при використанні на основі платформи Node.js.

```
<script>
    var test = 'test-value';

    $('body').on('click', '#some_button', function () {
        var $this = $(this);
        $this.append('<i class="icon-spinner icon-spin"></i>');

        $.ajax({
            url: test,
            data: {
                test: 'test'
            }
        });
    });
</script>
```

Рисунок 3.2 – Синтаксис JavaScript

Також дана мова програмування є підтримуваною всіма основними і не тільки браузерами та є, можна сказати, певним стандартом для використання під час веброзробки. Для даної мови існує величезна кількість різноманітних фремворків та бібліотек від найменших та мінімальних по функціоналу до величезних за своїм обсягом та документацією (наприклад Angular, React або Vue.js), які дозволяють створювати комплексні вебзастосунки, незалежно від вимог до розроблюваних систем.

TypeScript (рис. 3.3) є надбудовою над JavaScript, яка додає статичну типізацію та інші нові можливості, що робить процес розробки набагато більш передбачуваним та безпечним.

```
// declare a function (using function expression)
const sum = function( a: number, b: number ): number {
    return a + b;
}

// print sum of `1` and `2
console.log( 'sum( 1, 2 ) =>', sum( 1, 2 ) );

// operate on the return value
console.log( 'sum( 1, 2 ) + 7 =>', sum( 1, 2 ) + 7 );
console.log( 'sum( 1, 2 ).toFixed(2) =>', sum( 1, 2 ).toFixed( 2 ) );
```

Рисунок 3.3 – Синтаксис TypeScript

TS дозволяє явно визначати типи змінних, що дозволяє зменшити кількість помилок, які можуть виникнути при неправильному використанні типів. Особливо це корисно під час роботи над великими проєктами, де код регулярно зазнає змін, а також де регулярно додають новий функціонал.

Також через те, що TS є надбудовою над JS, то це означає, що код JavaScript сумісний із TypeScript. Це дозволяє у разі необхідності без особливих проблем переносити проєкти на нову надбудову.

Для тренування та роботи зі штучним інтелектом використовуються бібліотеки TensorFlow та Keras.

TensorFlow (рис. 3.4) це бібліотека для машинного навчання та штучного інтелекту. Вона використовується для широкого асортименту задач, але в основному існує для тренування та роботи зі штучним інтелектом. Це також один із найбільш популярних фреймворків для глибокого машинного навчання (наряду з PyTorch).

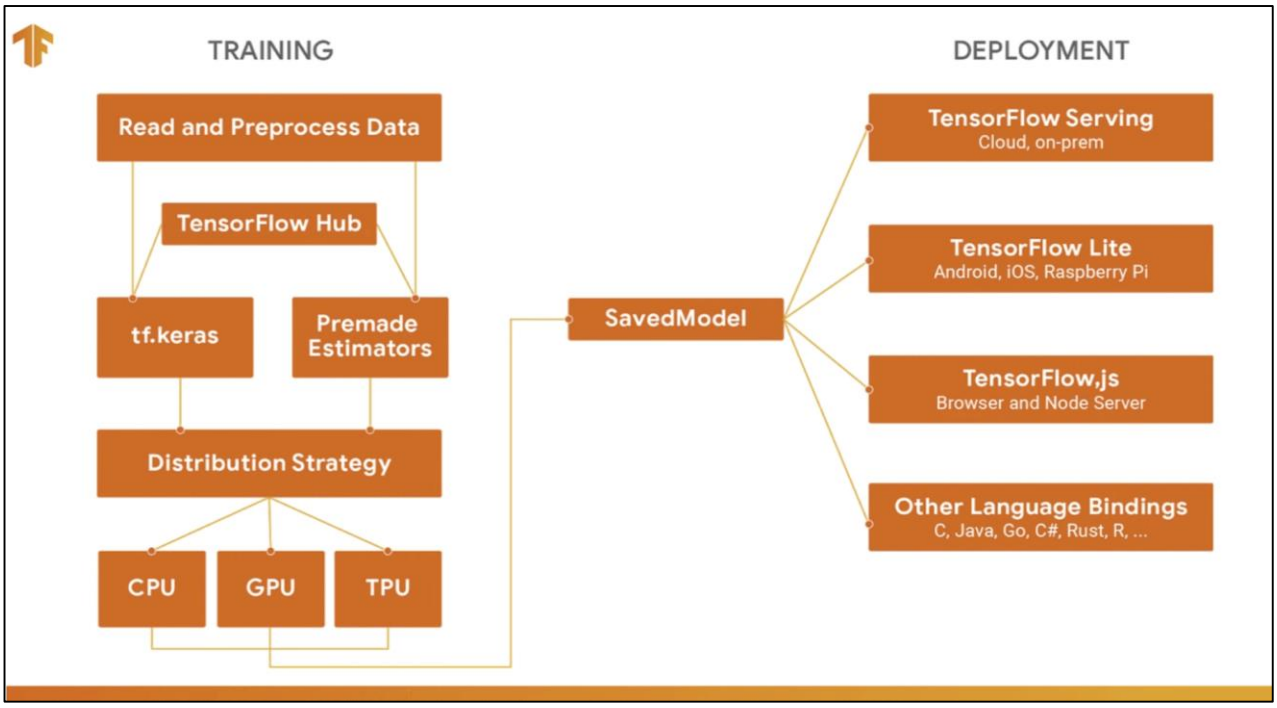


Рисунок 3.4 – Структура TensorFlow

Дана бібліотека використовується широким спектром мов програмування, таких як Python, JavaScript, C++, Java тощо.

Keras (рис. 3.5) є бібліотекою з відкритим програмним кодом, яка надає інтерфейс для роботи зі штучними нейронними мережами. Дана бібліотека раніше була незалежним програмним забезпеченням, але з часом її інтегрували до бібліотеки Tensorflow.

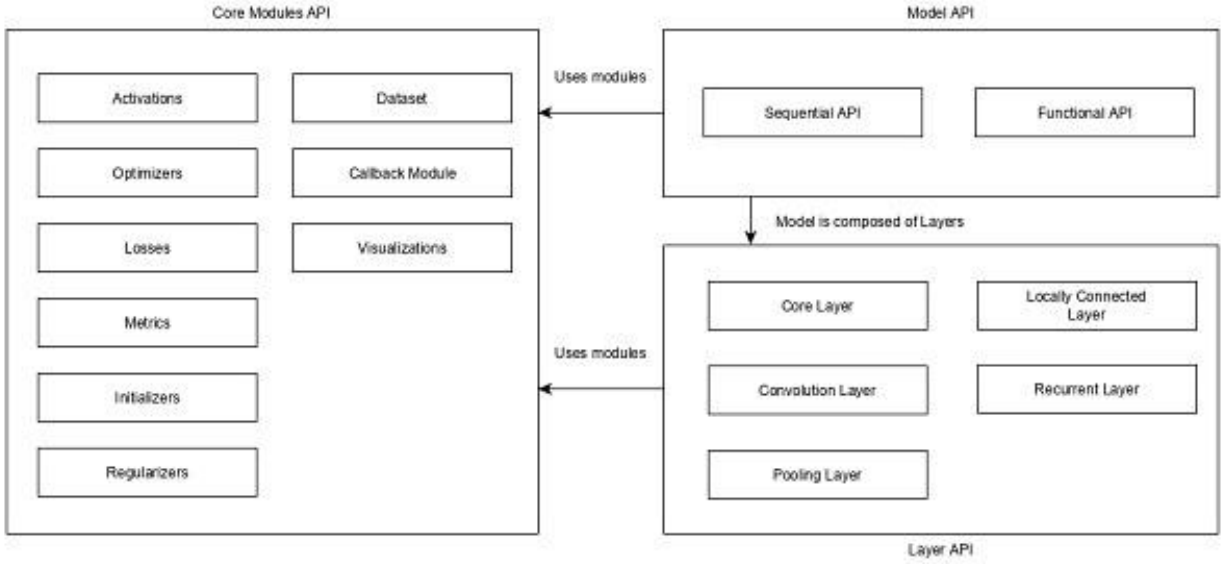


Рисунок 3.5 – Структура модулів Keras

Дана бібліотека дозволяє дуже швидко почати експерименти та роботу з глибокими нейронними мережами, а сам функціонал відрізняється дружелюбністю до користувача, модульністю та розширенням можливостей TensorFlow.

3.3 Розробка UML-діаграм

На етапі проєктування ПЗ найкращим способом показати ті чи інші майбутні функції програмного продукту є з використанням різноманітних діаграм. Серед всіх відомих видів діаграм найбільше виділяються UML-діаграми, які є еталоном зручності для використання. Завдяки своїх простоті, навички з їх побудови легкі в опануванні, а самі вони надають тому, хто моделює систему, змогу найкраще описати різноманітні процеси, які відбуваються в системі, а також зрозуміло зобразити саму структуру ПЗ.

UML – це певний стандарт в моделюванні, який надає набір графічних символів та перелік правил, на основі яких доволі зрозуміло та просто створювати моделі програмних систем.

Унікальність UML у тому, що він не прив’язаний до конкретної мови програмування чи технології. Це означає, що UML можна використовувати для будь-якої системи, від простих вебзастосунків до складних розподілених систем. UML діаграми можна створювати як на етапі проєктування, щоб зрозуміти, як система має виглядати, так і для існуючої програми, щоб краще розібратися в її структурі. Використання UML допомагає уникнути плутанини, яка може виникнути під час розробки складних проєктів. Вони слугують своєрідною картою, яка показує, як усе взаємодіє, і допомагають знайти слабкі місця в дизайні ще до того, як код буде написаний.

Також особливістю даних діаграм є їх варіативність. Ця мова моделювання включає в себе різні види діаграм, серед них:

- діаграми варіантів використання;
- діаграми класів;
- діаграми послідовностей;
- діаграми станів;
- діаграми пакетів;
- діаграми розгортання;
- діаграми компонентів, тощо.

Кожен різновид діаграм має свою особливу специфікацію та використовується при моделюванні певних елементів системи.

Для визначення основних сутностей застосунку та зв’язків між ними використовується діаграма класів. Діаграми класів — це один із найважливіших інструментів для моделювання програмних систем у рамках об'єктно-орієнтованого підходу. Вони дозволяють зрозуміти структуру системи, зобразивши основні елементи — класи, їхні властивості, методи, а також зв’язки між ними. Головна перевага діаграм класів — це їхня універсальність. Вони можуть бути використані

як на етапі проєктування системи, щоб передбачити її структуру, так і під час аналізу вже існуючого програмного коду, щоб краще зрозуміти його архітектуру. Завдяки цьому вони є важливим інструментом для комунікації між членами команди розробників, оскільки дають змогу говорити "однією мовою", спрощуючи обговорення складних технічних рішень.

Створена діаграма має 8 основних сутностей (рис. 3.6).

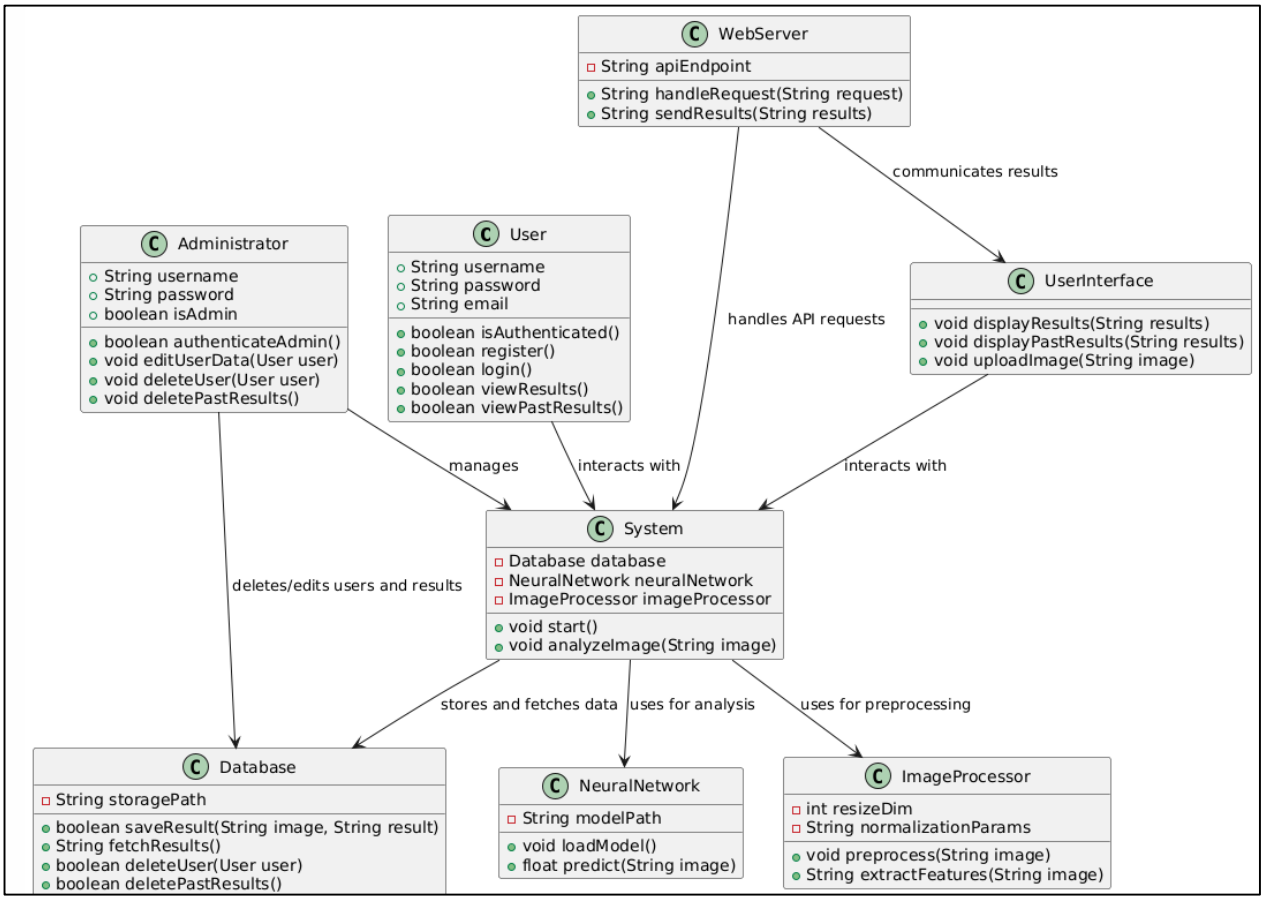


Рисунок 3.6 – Діаграма класів

У центрі всієї системи знаходиться клас System, який координує процеси між усіма іншими класами. Він використовує базу даних для збереження та отримання інформації, обробник зображень для їх попередньої підготовки до перевірки та приведення їх до певного стандарту, а також саму нейромережу для аналізу зображень.

Користувачі взаємодіють із системою через «UserInterface», де вони можуть завантажувати зображення та переглядати результати аналізу. Запити користувачів

опрацьовуються класом «WebServer», який передає інформацію між інтерфейсом і системою, а також відповідає за передачу результатів.

Для управління системою передбачений клас «Administrator», який дозволяє адміністраторам редагувати дані користувачів та видаляти історію їх старих результатів. Окремий клас «User» відповідає за звичайних користувачів, які можуть входити в систему, реєструватися в ній та переглядати свої попередні результати.

Система також включає в себе клас «Database», який використовується для зберігання результатів аналізу, та клас «NeuralNetwork», що здійснює аналіз з використанням попередньо натренованої моделі нейронної мережі. Клас «ImageProcessor» займається попередньою обробкою зображень перед їх аналізом, приводячи їх до нормалізованого формату.

Діаграма компонентів відображає залежності між компонентами програмного забезпечення, включаючи компоненти вихідних кодів, бінарні компоненти, та компоненти, що можуть виконуватись. Модуль програмного забезпечення може бути представлено як компоненту. Деякі компоненти існують під час компіляції, деякі — під час компонування, а деякі під час роботи програми.

Діаграма компонентів відображає лише структурні характеристики, для відображення окремих екземплярів компонент слід використовувати діаграму розгортування.

Діаграма компонентів системи виявлення фейкових зображень складається з 4 модулів, які пов’язані між собою (рис. 3.7).

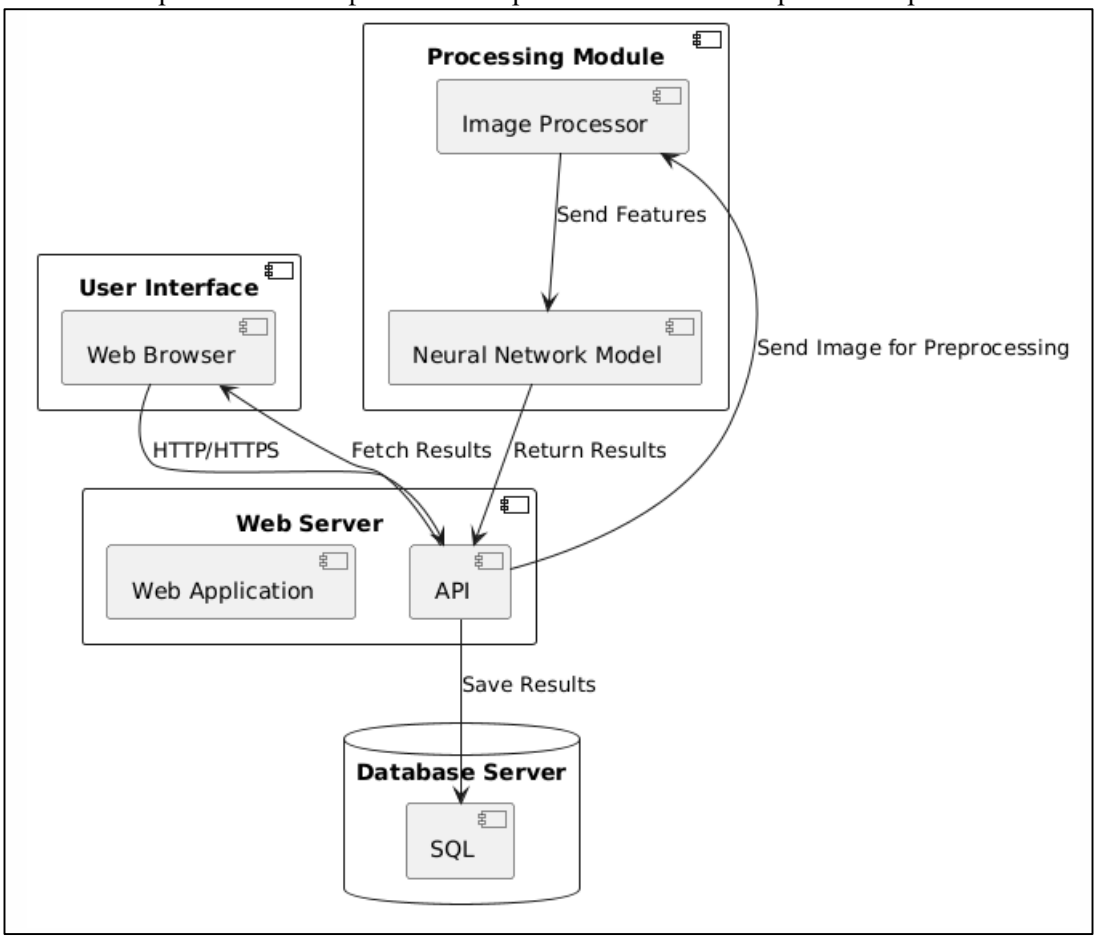


Рисунок 3.7 – Діаграма компонентів

Користувач взаємодіє із системою через «User Interface», який представлений через браузер, який надсилає запити до «Web Server» за допомогою HTTPS протоколу.

«Web Server» є центральним компонентом, який містить вебдодаток і API. Він отримує зображення від користувача, передає їх на обробку, а також відповідає за збереження і отримання результатів з «Database Server».

Для аналізу зображень використовується «Processing Module», який включає два основні елементи: «Image Processor» і «Neural Network Model». Зображення спочатку проходить через модуль попередньої обробки «Image Processor», який готує його для аналізу, а потім обробляється нейромережею, яка здійснює аналіз зображення. Оброблені дані повертаються через API до сервера.

«Database Server» відіграє роль сховища для збереження результатів обробки, які можуть бути запитані через сервер і повернуті користувачеві у вебінтерфейсі.

Діаграма розгортання (рис. 3.8) це діаграма, на якій відображаються обчислювальні вузли під час роботи програми, компоненти, та об'єкти, що виконуються на цих вузлах. Компоненти відповідають представленню робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються; натомість, їх можна відобразити на діаграмах компонент. Діаграма розгортання відображає робочі екземпляри компонент, а діаграма компонент, натомість, відображає зв'язки між типами компонент.

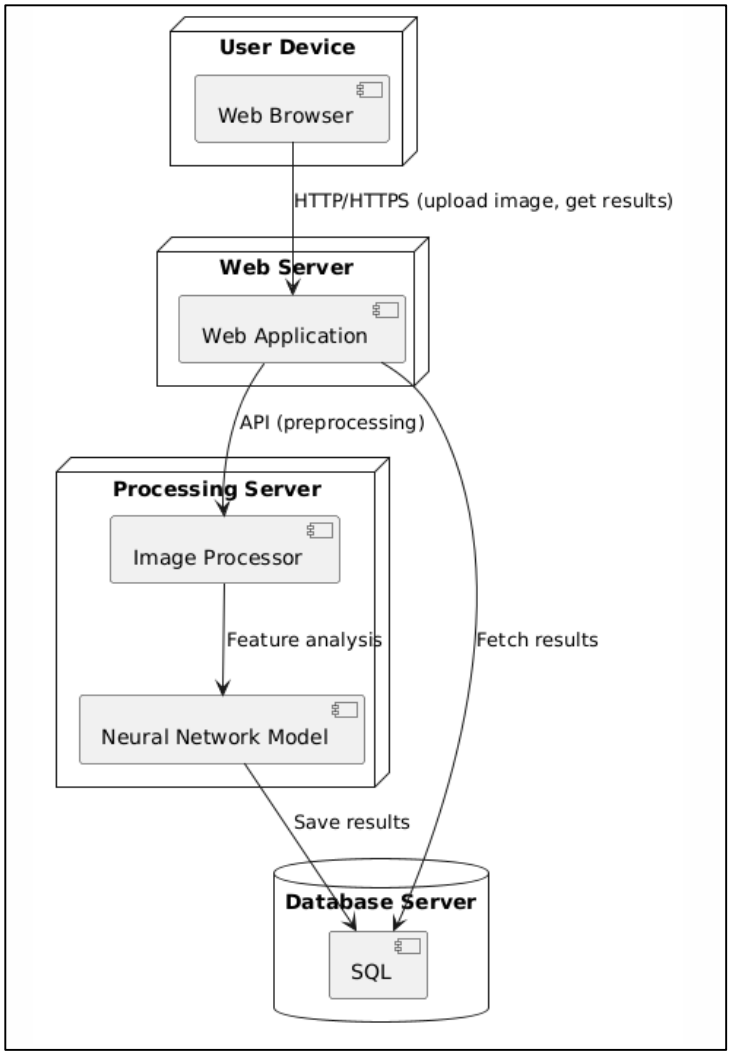


Рисунок 3.8 – Діаграма розгортання

Було створено діаграму пакетів. Діаграми пакетів відображають залежності між пакетами, які складають модель.

Діаграми пакетів можуть використовувати пакети, які містять прецеденти для ілюстрації функціональності програмного забезпечення системи.

Діаграма пакетів системи виявлення фейкових зображень складається з 4 основних пакетів (рис. 3.9).

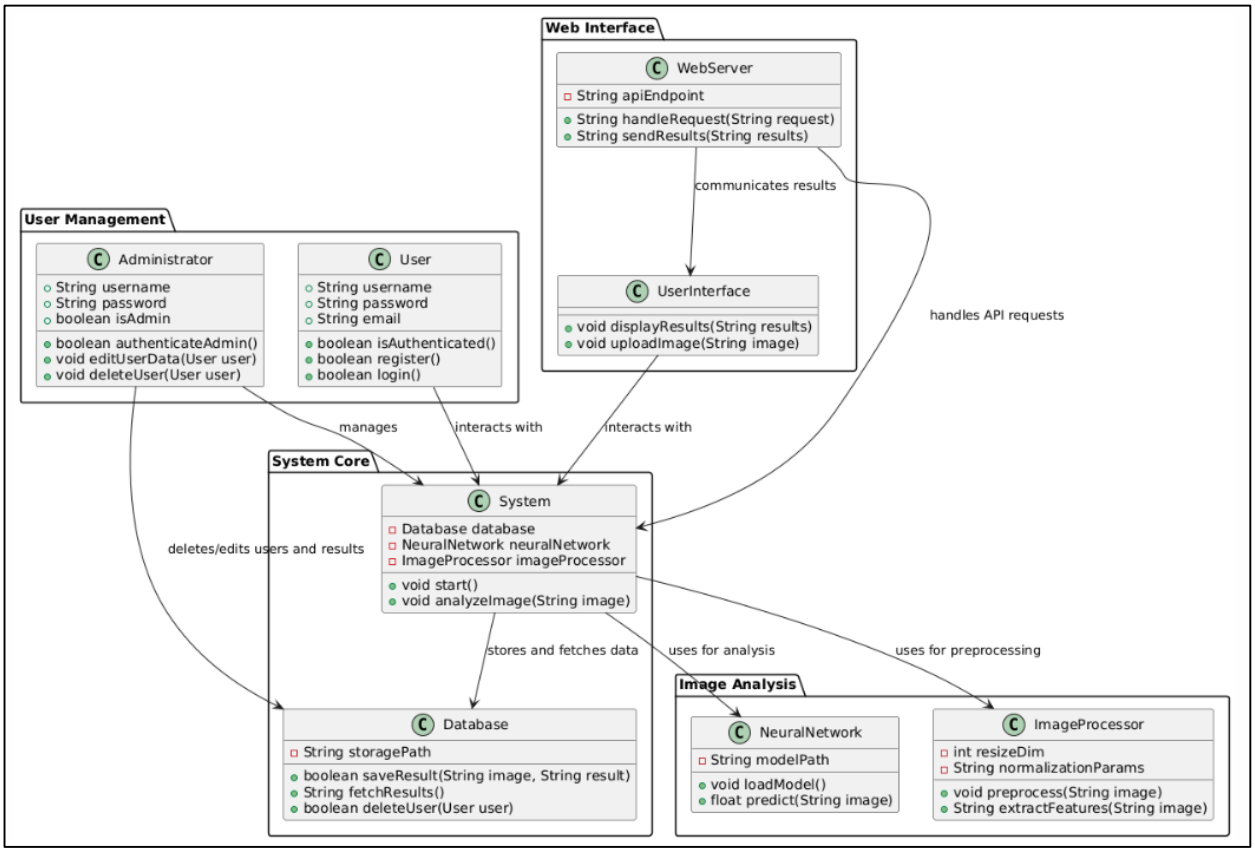


Рисунок 3.9 – Діаграма пакетів

Далі було створено діаграму прецедентів для системи (рис. 3.10). Діаграма прецедентів представляє собою графічний показ можливої взаємодії того чи іншого користувача з системою залежно від ролі користувача. Взаємодії на таких діаграмах показані або колами, або еліпсами, в той час як актори показані фігурками чоловічків, зроблених з палочок.

Перевагою використання даної діаграми при плануванні архітектури системи є її простота, що допомагаю в комунікаціях із людьми, які далекі від технічних питань.

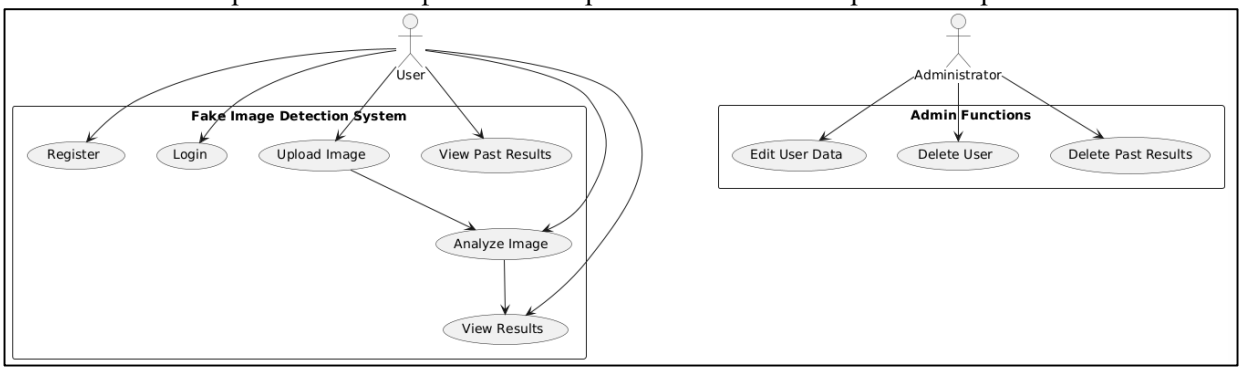


Рисунок 3.10 – Діаграма прецедентів

Далі було створено діаграму станів для системи розпізнавання (рис. 3.11). Діаграма станів – це спрямовані графи у вузлах яких позначаються стани та які поєднані між собою стрілками, які позначають дії-переходи між цими станами.

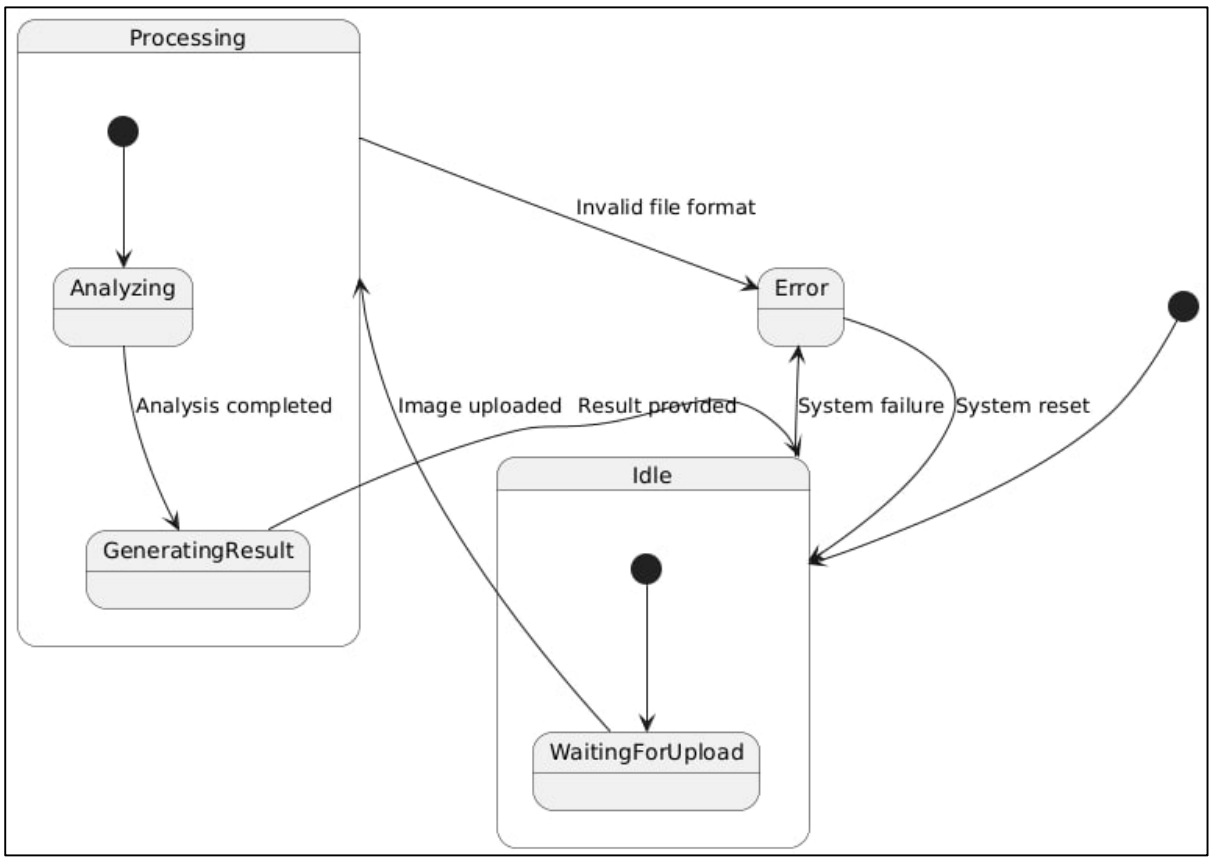


Рисунок 3.11 – Діаграма станів

Наостанок було створено діаграму послідовностей (рис. 3.12). Діаграма послідовностей вказує на взаємодію процесів у певній часовій послідовності. На

такій діаграмі показані задіяні процеси та об’єкти в системі, а також послідовність дій, які необхідні для виконання тих чи інших дій у певній функції системи.

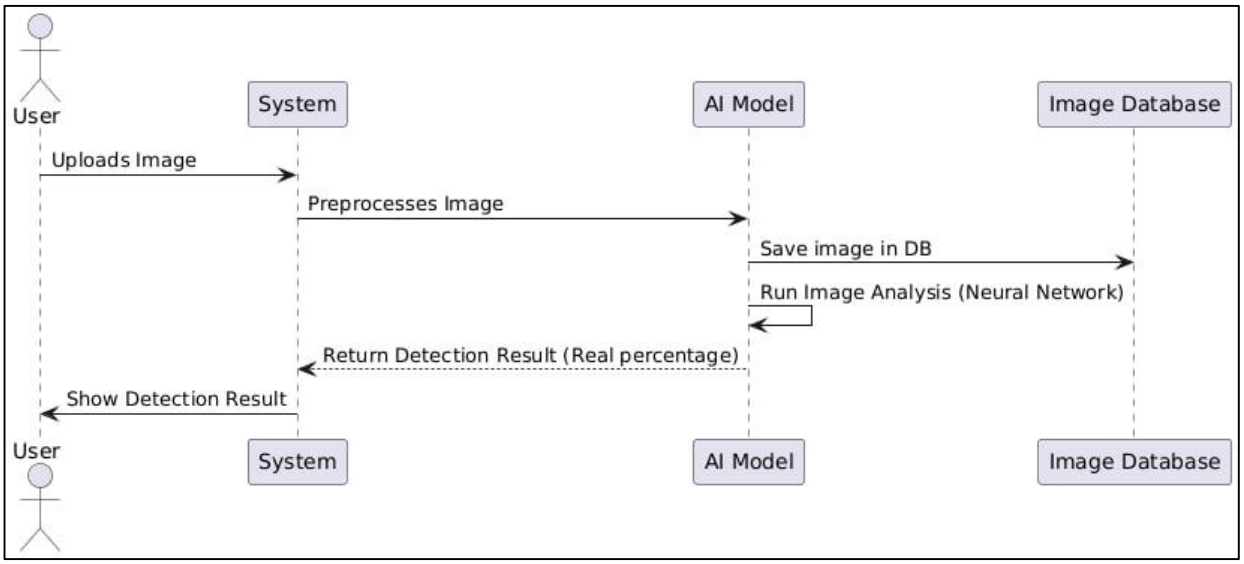


Рисунок 3.12 – Діаграма станів

3.4 Опис інтерфейсів програмного забезпечення

В даній системі передбачено використання двох видів інтерфейсів, а саме користувацького інтерфейсу (UI) та програмного інтерфейсу (API).

Користувацький інтерфейс вирішено зробити мінімалістичний, для уникнення його перевантаження, та в подальшому уникнення ускладненого серфінгу користувача по системі. UI умовно розділений, залежно від сторінок, на яких перебуває користувач, але серед основних елементів взаємодії можна виділити наступні:

– поле для завантаження зображення (з підтримкою форматів PNG, JPG та JPEG);

– кнопка відправки зображення на аналіз;

– вікно входу та реєстрації у системі;

– меню з усіма доступними сторінками системи;

– історія перевірки зображень користувачем у вигляді колажу в особистому кабінеті юзера

– відображення ймовірності фейковості зображення у відсотках після перевірки тощо.

Висновки до розділу 3

У третьому розділі кваліфікаційної магістерської роботи було створено UML-діаграми, які відображають роботу та внутрішню архітектуру системи, що розробляється, а також дозволить надалі покращити розробку ПЗ та полегшить внесення нового функціоналу до системи. Серед розроблених діаграм можна виділити: діаграму класів, діаграму компонентів, діаграму розгортання та діаграму пакетів.

Даний етап є найважливішим перед початком розробки програмного забезпечення. Все по причині того, що під час створення даних діаграм – вся інформація структуризується у зручному вигляді, що дозволяє повернутись до неї під час будь-якого етапу розробки застосунку, тим самим економлячи час та нерви розробників.

Також було проведено вибір мов програмування, бібліотек та датасету для тренування моделі нейронної мережі. Вибір було проведено з попереднім визначенням критеріїв та вимог функціоналу до даних елементів. Більша частина проблем під час розробки систем будь-якого виду пов’язані саме через неправильний вибір мов програмувань та бібліотек для них для реалізації програмного забезпечення.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПРОВЕДЕННЯ ТЕСТУВАННЯ

4.1 Розробка дизайну вебсистеми

Однією з важливих частин при розробці проєкту є створення його дизайну. Для розробки дизайну було використано спеціальний програмний застосунок з підтримкою хмарної бази даних «Figma».

Figma (рис. 4.1) – це одне з найпопулярніших рішень на ринку дизайну. Цей застосунок має потужний інструментарій для роботи над дизайном та прототипуванням будь-яких продуктів. Він дозволяє командам, в які входять не тільки дизайнери, а й розробник, працювати над вебдизайном і прототипами продуктів, навіть не знаходячись в одному місці. Даний програмний продукт дозволяє в реальному часі працювати великій кількості людей одночасно над одним і тим самим дизайном, що дуже корисно у випадку віддаленої роботи або аутсорсингу.

Для початку було розроблено дизайн вікон авторизації користувача. А саме було створено вікно входу користувача (рис. 4.1).

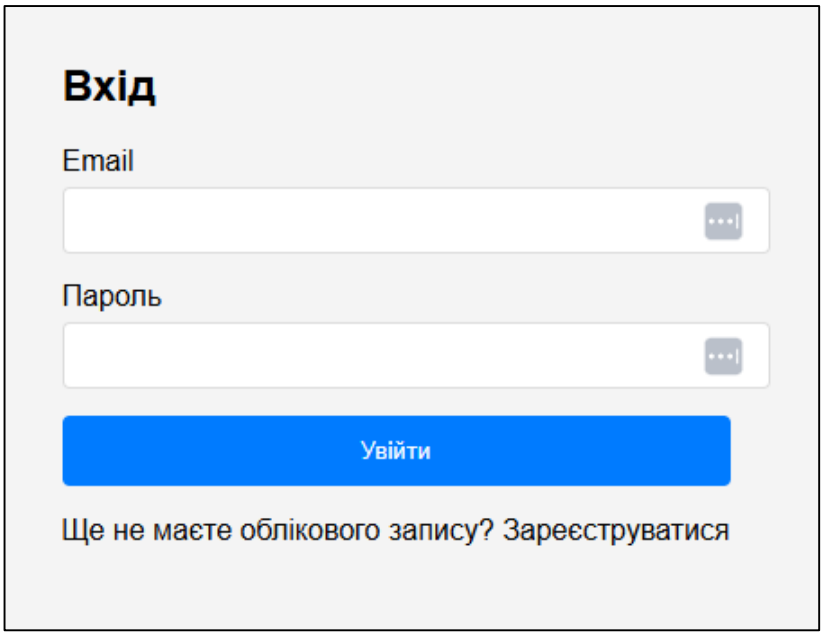


Рисунок 4.1 – Вікно входу користувача

А після було створено вікно реєстрації користувача (рис. 4.2).

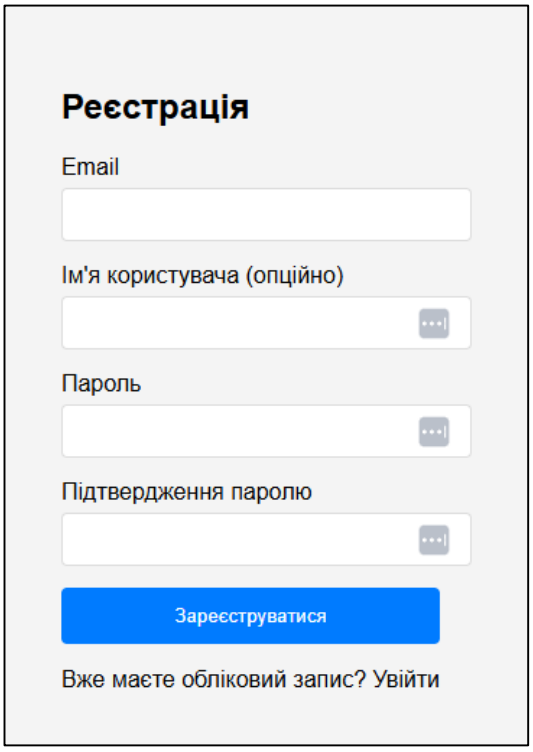


Рисунок 4.2 – Вікно реєстрації користувача

Після авторизації, користувач потрапляє в меню для перевірки зображень (рис. 4.3). Оскільки на даний момент продукт не планується випускати в публічний простір, то дизайн було зроблено максимально мінімалістичним. В подальшому, за потреби – він буде змінюватись.

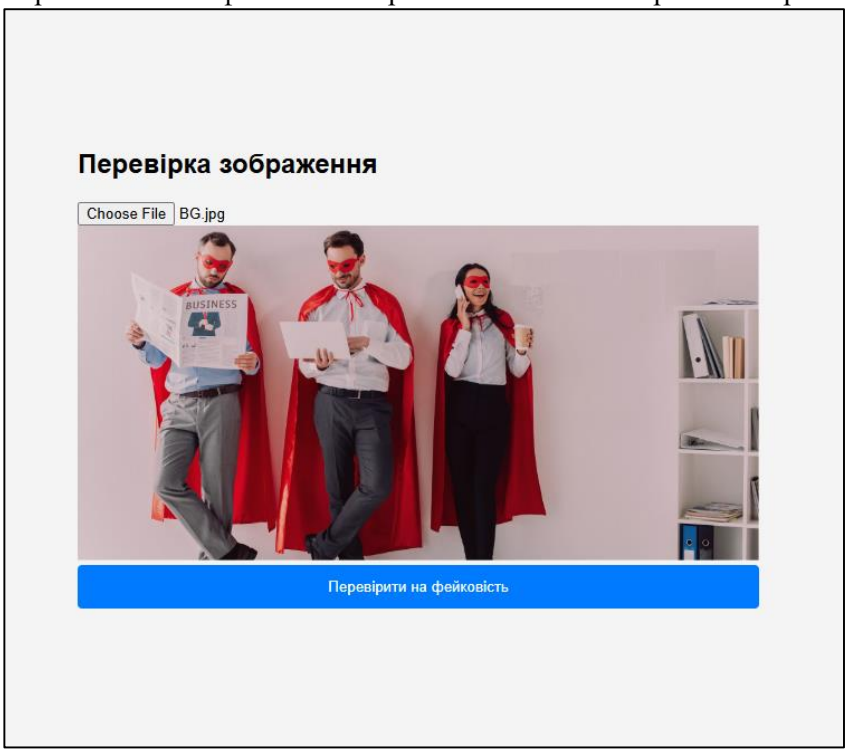


Рисунок 4.3 – Вікно перевірки зображення

4.2 Тренування моделі нейронної мережі

Для тренування моделі нейронної мережі було використано датасет, взятий із відкритого доступу в мережі Інтернет. Заплановано, що в подальшому нейронна мережа буде «дотреновуватись» вже на основі отриманих зображень на перевірку від користувачів.

Датасет зображень включає в себе більше 120 тисяч різноманітних зображень умовно розділених на дві категорії – фейкові зображення (зображення згенеровані штучним інтелектом, а також зображеня, які було відредаговано у спеціалізованих графічних редакторах), а також реальні зображення, які не піддавались ніякій обробці. На рис. 4.4 зображено фейкові зображення, які входять в даний датасет.

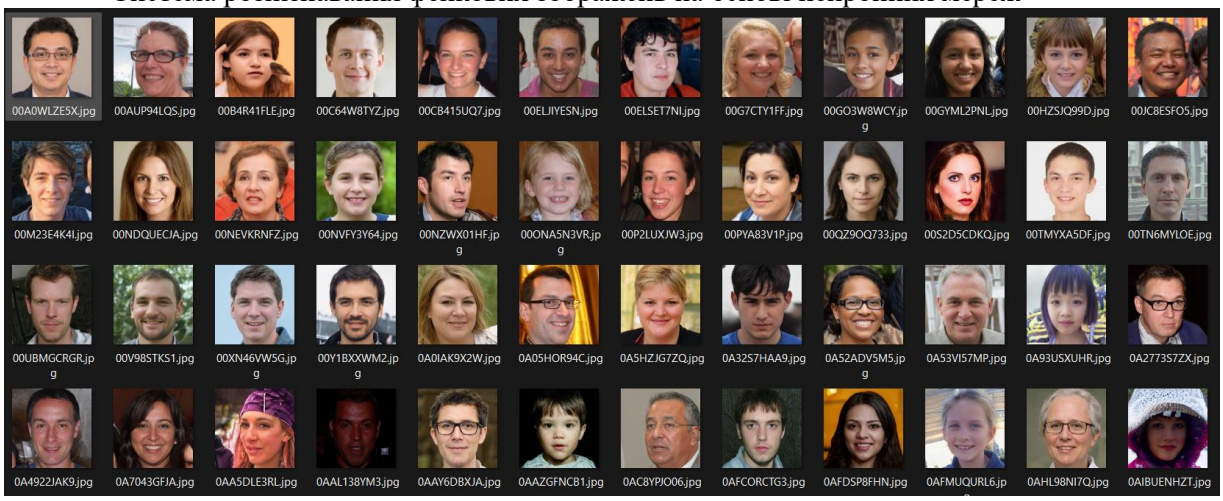


Рисунок 4.4 – Фейкові зображення, що входять до датасету

На рис. 4.5 зображено реальні зображення, які також входять в тренувальний датасет.

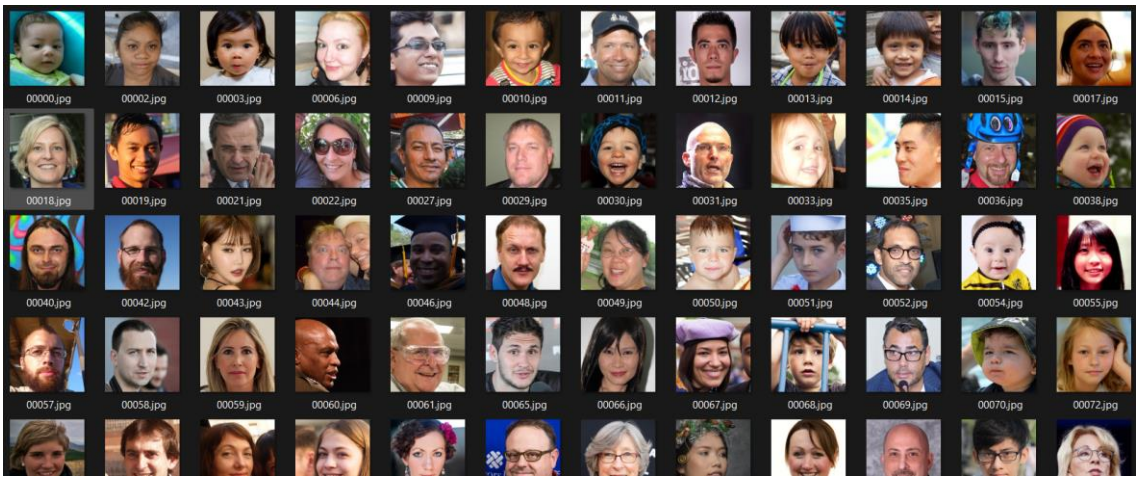


Рисунок 4.5 – Реальні зображення, що входять до датасету

Тренування моделі ШІ відбувалось з використанням бібліотек TensorFlow та Keras на мові програмування Python. Для початку було завантажено всі необхідні бібліотеки, а опісля було прописано код для тренування моделі:

```
train_fake_path = r"E:\FAKE-REAL\archive\real_vs_fake\real-vs-fake\train\fake"
train_real_path = r"E:\FAKE-REAL\archive\real_vs_fake\real-vs-fake\train\real"
test_fake_path = r"E:\FAKE-REAL\archive\real_vs_fake\real-vs-fake\test\fake"
test_real_path = r"E:\FAKE-REAL\archive\real_vs_fake\real-vs-fake\test\real"
img_size = (150, 150)  # Image dimensions
batch_size = 32
```

```
train_datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)
train_generator = train_datagen.flow_from_directory(
    directory=os.path.dirname(train_fake_path),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary',
    subset='training')
validation_generator = train_datagen.flow_from_directory(
    directory=os.path.dirname(train_fake_path),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary',
    subset='validation'
test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    directory=os.path.dirname(test_fake_path),
    target_size=img_size,
    batch_size=batch_size,
    class_mode='binary')
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=validation_generator
)
```

З кожною епохою тренування моделі нейронної мережі, відсоток помилки зменшувався, в той час як відсоток точності зростав. Це можна побачити на рис. 4.7.

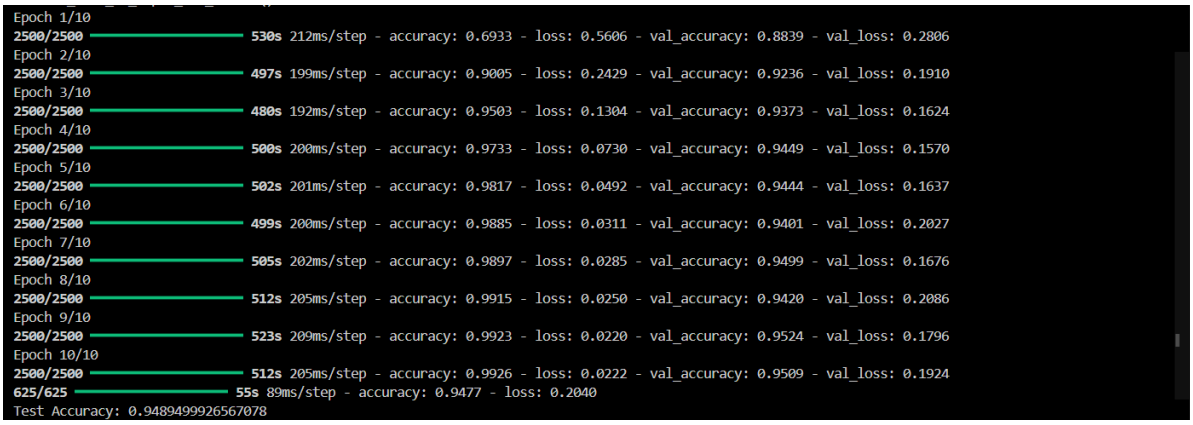


Рисунок 4.7 – Поепоховий ріст точності моделі

Також на рис. 4.8 продемонстровані графіки точності моделі в залежності від пройденої епохи навчання.

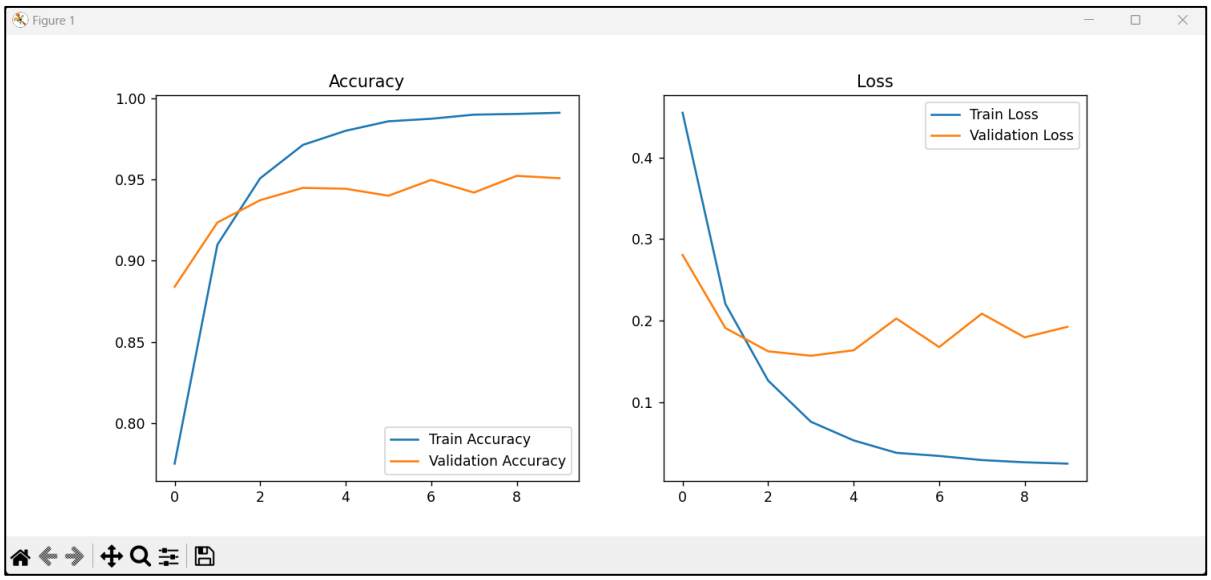


Рисунок 4.8 – Графіки точності моделі

4.3 Створення системи

Програмне забезпечення було створено з використанням технології Angular. Для початку було створено сервіс авторизації. Його код наведено нижче:

```
export class AuthService {
  private storageService = inject(StorageService<User>);
```

```
private router = inject(Router);


register(email: string, password: string, username?: string): boolean {
  const existingUser = this.getUserByEmail(email);
  if (existingUser) {
    return false;
  }
  const newUser: User = {
    id: uuidv4(),
    email,
    password,
    username,
    createdAt: new Date()
  };
  this.storageService.save(`user_${newUser.id}`, newUser);
  this.setCurrentUser(newUser);
  return true;
}
login(email: string, password: string): boolean {
  const users = this.storageService.getAll();
  const user = users.find(u =>
    u.email === email && u.password === password
  );
  if (user) {
    this.setCurrentUser(user);
    return true;
  }
  return false;
}
logout(): void {
  localStorage.removeItem('currentUser');
  this.router.navigate(['/login']);
}
getCurrentUser(): User | null {
  return this.storageService.get('currentUser');
}
private setCurrentUser(user: User): void {
  this.storageService.save('currentUser', user);
}
private getUserByEmail(email: string): User | null {
```

```
    const users = this.storageService.getAll();
    return users.find(u => u.email === email) || null;
  }
  isAuthenticated(): boolean {
    return !!this.getCurrentUser();
  }
}
```

Сервіс авторизації складається з декількох методів, а саме:

– login() – метод отримує електронну пошту та пароль в якості вхідних даних від користувача та шукає даного користувача в уже існуючих зареєстрованих користувачах. У випадку, якщо такий користувач знаходиться, то користувача перенаправляють на головне меню. Якщо ні – то користувачеві видається помилка з причиною, чому його не запустили в головне меню;

– register() – метод отримує електронну пошту, юзернейм та пароль користувача в якості вхідних даних. Якщо дана користувачем інформація відсутня в базі, тоді створюється новий запис користувача в базі та в подальшому він може використовувати попередньо введені дані для входу в систему;

– logout() – метод, який викликається, коли користувач хоче вийти із системи. Після виходу із системи, користувача перенаправляє на сторінку входу в систему;

– getCurrentUset() – метод, який перевіряє, чи авторизований хтось на даний момент в системі із пристрою відвідувача системи;

– setCurrentUser() – метод, який встановлює, що користувач авторизувався в систему;

– getUserByEmail() – метод, який перевіряє, чи існує акаунт за поштою, яку вказав користувач під час реєстрації;

Далі було створено сервіс зберігання даних користувачів. Оскільки даний продукт поки не планувався випускатись в публічний простір – було вирішено створити зберігання даних тимчасовим в localStorage. Його код наведено нижче:

```
@Injectable({
  providedIn: 'root'
})
```

```
export class StorageService<T> implements StorageInterface<T> {
  save(key: string, data: T): void {
    localStorage.setItem(key, JSON.stringify(data));
  }
  get(key: string): T | null {
    const item = localStorage.getItem(key);
    return item ? JSON.parse(item) : null;
  }

  remove(key: string): void {
    localStorage.removeItem(key);
  }
  getAll(): T[] {
    return Object.keys(localStorage)
      .map(key => {
        try {
          return this.get(key);
        } catch (e) {
          console.error(`Error parsing key "${key}":`, e);
          return null;
        }
      })
      .filter(item => item !== null) as T[];
  }
}
```

Зберігання даних складається з декількох методів:

– save() – метод, який в якості вхідних параметрів отримує інформацію про дані, які треба зберегти в localStorage та опісля зберігає ці дані;

– get() – метод, який в якості вхідного параметру отримує ключ для пошуку по ньому значення в localStorage та повертає дане значення;

– remove() – метод, який отримує ключ в якості вхідного значення на по цьому ключу шукає та видаляє те значення;

– getAll() – метод, який отримує всі можливі пари ключ-значення, які збережені в localStorage та зв’язані з системою.

Опісля було розроблено компонент завантаження зображення у сервіс. Нижче наведено код завантаження зображення:

```
@Component({
  selector: 'app-upload',
  standalone: true,
  imports: [CommonModule],
  templateUrl: './upload.component.html'
})
export default class UploadComponent {
  selectedFile: File | null = null;
  detectionResult: number | null = null;
  imagePreview: string | null = null;
  private imageDetectionService = inject(ImageDetectionService);
  onFileSelected(event: Event) {
    const input = event.target as HTMLInputElement;
    if (input.files && input.files.length > 0) {
      this.selectedFile = input.files[0];
      const reader = new FileReader();
      reader.onload = (e) => {
        this.imagePreview = e.target?.result as string;
      };
      reader.readAsDataURL(this.selectedFile);
    }
  }
  async checkImage() {
    if (this.selectedFile) {
      try {
        this.detectionResult = await this.imageDetectionService.detectFakeImage(
          this.selectedFile
        );
      } catch (error) {
        console.error('Помилка детекції зображення', error);
      }
    }
  }
}
```

В даному випадку роботу виконують два основних методи:

– onFileSelected() – це метод, який обробляє вибір файлу користувачем, зберігає даний файл у selectedFile та генерує прев’ю зображення;

– checkImage() – це асинхронний метод, який використовує сервіс ImageDetectionService для виявлення фейкових зображень.

Далі було створено найголовніший сервіс у всій системі, а саме сервіс розпізнавання фейкових зображень. Код сервісу наведено нижче:

```
async detectFakeImage(file: File): Promise<number> {
  if (!this.model) {
    await this.loadModel();
  }
  const base64 = await this.fileToBase64(file);
  const img = await this.preprocessImage(base64);
  const prediction = this.model?.predict(img) as tf.Tensor;
  const result = prediction.dataSync()[0];
  this.saveImageCheckResult(file.name, base64, result);
  return result;
}
private async fileToBase64(file: File): Promise<string> {
  return new Promise((resolve, reject) => {
    const reader = new FileReader();
    reader.readAsDataURL(file);
    reader.onload = () => resolve(reader.result as string);
    reader.onerror = error => reject(error);
  });
}
private async preprocessImage(base64: string): Promise<tf.Tensor> {
  const img = new Image();
  img.src = base64;
  await new Promise(resolve => { img.onload = resolve; });
  const tensor = tf.browser.fromPixels(img)
    .resizeBilinear([224, 224])
    .toFloat()
    .expandDims();
  return tensor;
}
private saveImageCheckResult(fileName: string, imageBase64: string, result:
number): void {
  const currentUser = this.authService.getCurrentUser();
  if (currentUser) {
    const imageCheck: ImageCheck = {
      id: uuidv4(),
```

```
        userId: currentUser.id,
        fileName,
        result,
        imageBase64,
        checkedAt: new Date()
      };
      this.storageService.save(`image_check_${imageCheck.id}`, imageCheck);
    }
  }
```

На цьому сервісі треба зупинитись детальніше, оскільки він є основою системи. Він має основні методи:

– loadModel() – це асинхронний метод для завантаження моделі нейронної мережі для виявлення фейкових зображень. Дана модель завантажується з файлу «fake_real_detector.h5», яка була попередньо натренована з використанням підготовленого датасету та з використанням бібліотеки Keras. У випадку помилки завантаження – у консоль виводиться помилка та її причина;

– detectFakeImage() – це асинхронний головний метод для виявлення фейковості зображення. Спочатку відбувається завантаження моделі, якщо вона ще не завантажена, далі відбувається перетворення зображення в base64 та попередня обробка зображення. Опісля відбувається сама перевірка з використанням моделі, після чого відбувається зберігання результатів перевірки;

– fileToBase64() – це приватний асинхронний метод, який перетворює завантажений файл у формат base64 з FileReader;

– preprocessImage() – це приватний асинхронний метод, який виконує попередню обробку зображень для його перевірки моделлю нейронної мережі. Він спочатку завантажує зображення, а потім змінює його розмір до 224x224 пікселів;

– saveImageCheckResult() – це приватний метод, який виконує збереження результатів перевірки зображення;

– getUserImageChecks() – це метод, який повертає список всіх минулих перевірок користувачів.

Попередньо використовуючи бібліотеку TensorFlow.js треба перетворити натреновану модель штучного інтелекту в формат, підтримуваний бібліотекою. На рис. 4.11 зображено вікно з результатом перевірки одного з зображень.

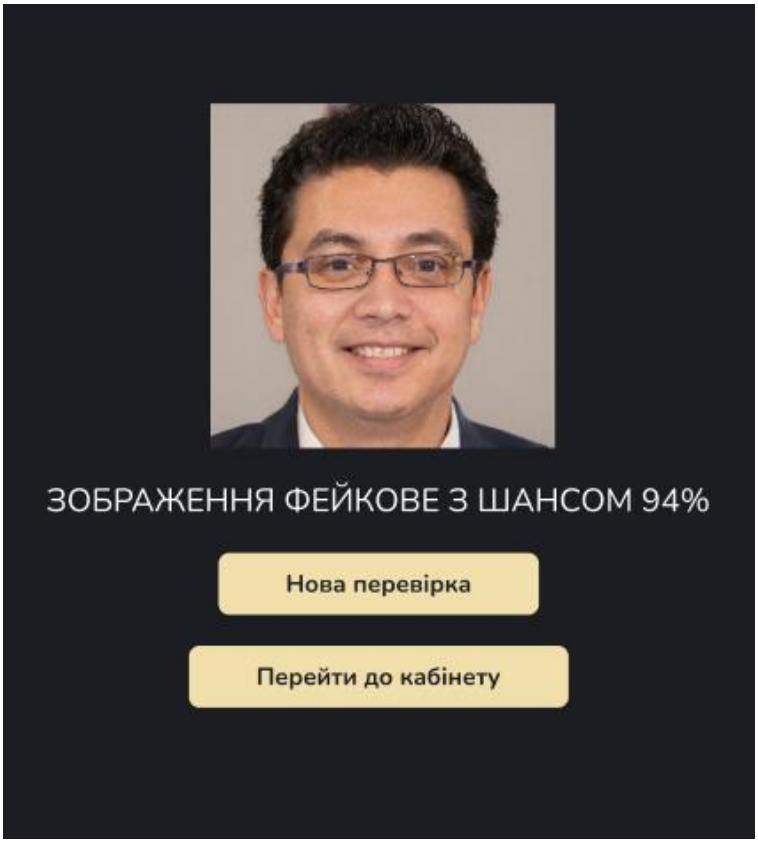


Рисунок 4.11 – Вікно з результатом перевірки зображення

Користувач має можливість перейти до власного кабінету, в якому він може переглянути результати минулих перевірок зображень (рис. 4.12).

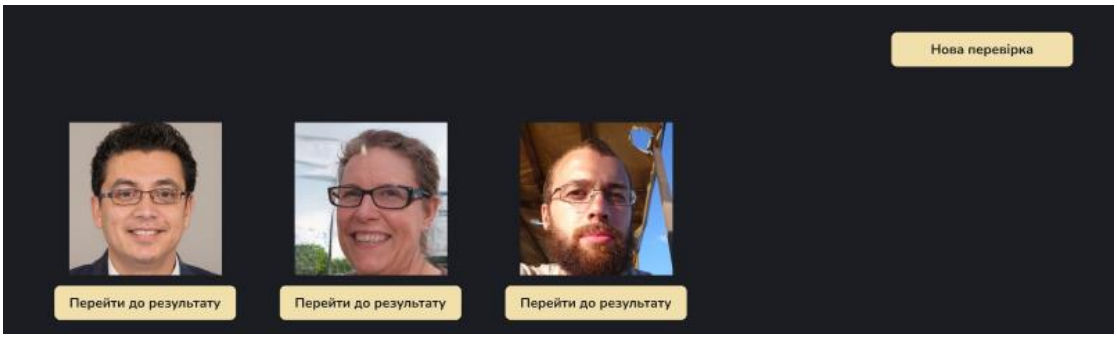


Рисунок 4.12 – Історія перевірок зображень

На виході було отримано готову систему з розпізнавання фейкових зображень.

4.4 Тестування системи

Після розробки систему та верифікації того, що всі необхідні її елементи працюють так, як треба – можна приступити до фінального етапу розробки, а саме тестування програмного забезпечення. Даний етап можна вважати ледве не найголовнішим у всьому процесу розробки ПЗ, оскільки саме він має на меті віднайти та визначити всі можливі недоліки, недопрацювання та помилки в системі. Такі помилки можуть різнитись від найменш неприємних (наприклад неправильне відображення якогось елемента на певній роздільній здатності екрану) до таких, що повністю ламають роботу системи, а то і девайс користувача. Окрім того, тестування також допомагає віднайти елементи та відрізки коду системи, які можна було б оптимізувати для поліпшення швидкодії роботи програмного забезпечення на різних пристроях користувачів. В таблицях нижче приведені тестування, які пройшли під час різних сценаріїв роботи системи.

Таблиця 4.1 – Авторизація користувача в систему

Діючі особи	Користувач
Мета	Авторизація в акаунт в системі
Передумова	Користувач не авторизований в системі
Успішні сценарії: – користувач переходить до форми логіну автоматично при заході в систему; – користувач заповнює свої особисті дані (електронна пошта та пароль); – користувач натискає кнопку «Увійти»; – система перевіряє дані, які надіслав користувач; – система перенаправляє користувача в його особистий кабінет.	
Сценарій успішний. Користувач потрапляє у свій особистий кабінет	

Кінець таблиці 4.1

Розширенні сценарії: – користувач не заповнює всі поля для входу. Результат – авторизація безуспішна, система видає повідомлення про помилку; – користувач вводить неправильні дані (електронну пошту або пароль). Результат – авторизація провалена, система видає повідомлення про помилку; – користувач вводить дані, яких не існує в системі. Результат – авторизація не проходить, система видає помилку щодо входу та пропонує створити акаунт.
Усі сценарії розширення успішно виконані.

Таблиця 4.2 – Реєстрація користувача в системі

Діючі особи	Користувач
Мета	Реєстрація акаунта в системі
Передумова	Користувач не авторизований в системі
Успішні сценарії: – користувач переходить до форми реєстрації при заході в систему; – користувач заповнює свої особисті дані (електронна пошта, пароль, перевірка паролю, нікнейм); – користувач натискає кнопку «Зареєструватись»; – система перевіряє дані, які надіслав користувач; – система видає повідомлення про успішну реєстрацію та перенаправляє в особистий кабінет.	
Сценарій успішний. Користувач створює новий акаунт та потрапляє в особистий кабінет.	
Розширенні сценарії: – користувач не заповнює всі поля для реєстрації. Результат – реєстрація не дозволена, система видає повідомлення про помилку;	

Кінець таблиці 4.2

– користувач вводить дані, які не валідуються (електронну пошту в неправильному форматі, тощо). Результат – реєстрація не дозволена, система видає повідомлення про помилку; – користувач вводить дані, які вже існують в системі. Результат – користувачу не дозволена реєстрація, система видає помилку та пропонує увійти в попередньо зазначений акаунт.
Усі сценарії розширення успішно виконані.

Таблиця 4.3 – Перевірка зображення на фейковість

Діючі особи	Користувач
Мета	Дізнатись реальність зображення
Передумова	Користувач авторизований в системі та має в наявності зображення для перевірки
Успішні сценарії: – користувач переходить на головну сторінку системи; – користувач натискає на кнопку «Завантажити зображення» та обирає зображення зі списку своїх файлів; – користувач натискає на кнопку «Перевірити зображення»; – система перевіряє чи відповідає зображення правильному розширенню файлу; – система надає нейронній мережі зображення на перевірку; – нейронна мережа надає відповідь системі після перевірки зображення; – система виводить відсоток можливості фейку користувачеві.	
Сценарій успішний. Користувач перевіряє зображення, яке його цікавить.	
Розширенні сценарії: – користувач надсилає пошкоджений файл. Результат – система видає помилку та просить користувача надіслати не пошкоджений файл;	

Кінець таблиці 4.3

– користувач надсилає файл в неправильному формату. Результат – система видає помилку та показує перелік форматів зображень, які підтримує система; – користувач надсилає занадто величезний файл. Результат – система видає помилку та просить користувача надіслати файл меншого розміру.
Усі сценарії розширення успішно виконані.

Далі було розглянуто всі 4 можливі варіанти результатів, які може видати система, а саме: розпізнане фейкове зображення, нерозпізнане фейкове зображення, розпізнане реальне зображення та нерозпізнане реальне зображення.

Для початку було перевірене зображення, яке є реальним (рис. 4.13).

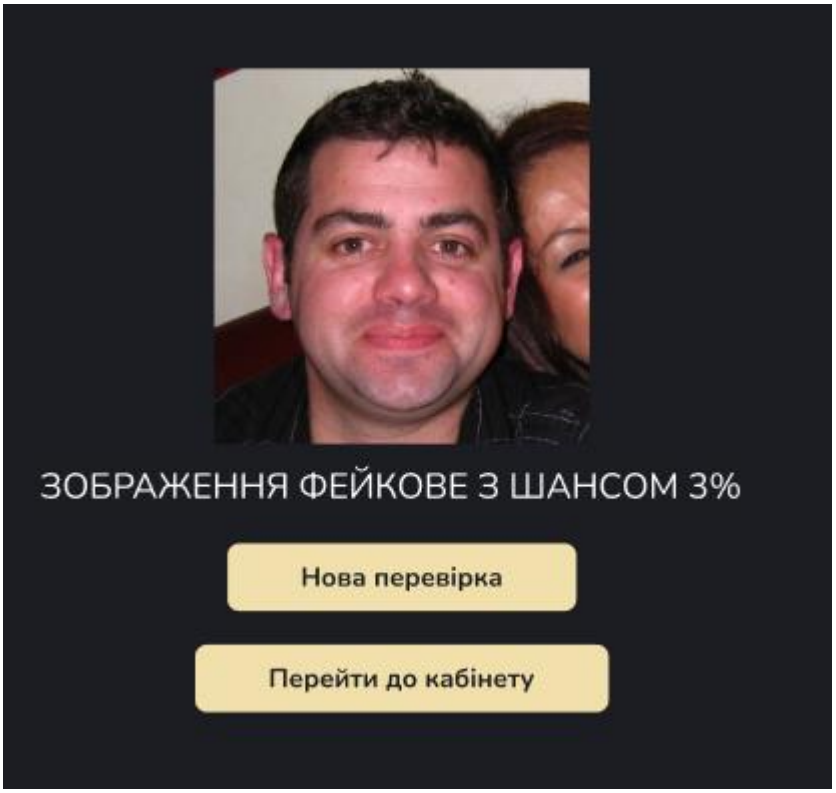


Рисунок 4.13 – Правильно розпізнане реальне зображення

Як можна судити з цієї перевірки – зображення було розпізнано правильно. Все через те, що чітко видно контури обличчя людини на зображенні, а також воно доволі гарної якості, без видозмінених кольорів, ідеальне по якравості та обличчя людини нічим не закрите.

Тепер роздивимось варіант, коли система не розпізнає правильне зображення (рис. 4.14).

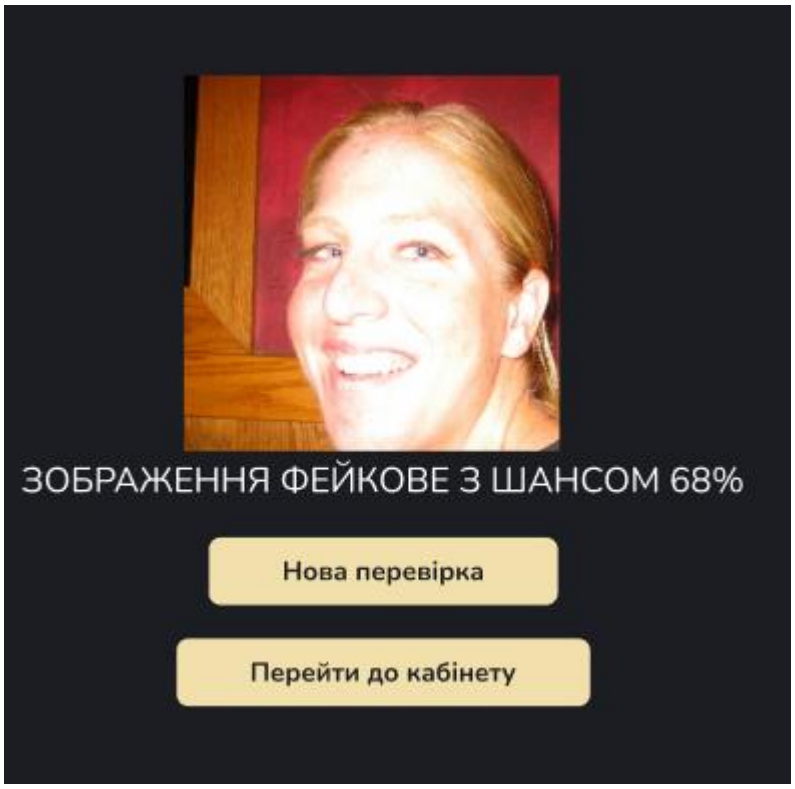


Рисунок 4.14 – Неправильно розпізнане реальне зображення

В даному випадку, система розпізнала зображення неправильно. Все по причині того, що ракурс фото не охоплює повних контурів обличчя людини, а також найголовнішою проблемою є надмірна яскравість зображення, через яку навіть після нормалізації фото, нейронній мережі було важко розпізнати правильно приналежність зображення до правильної категорії.

Далі розглянемо варіанти фейкових зображень. Для початку проаналізуємо кейс із правильним розпізнаванням (рис. 4.15).

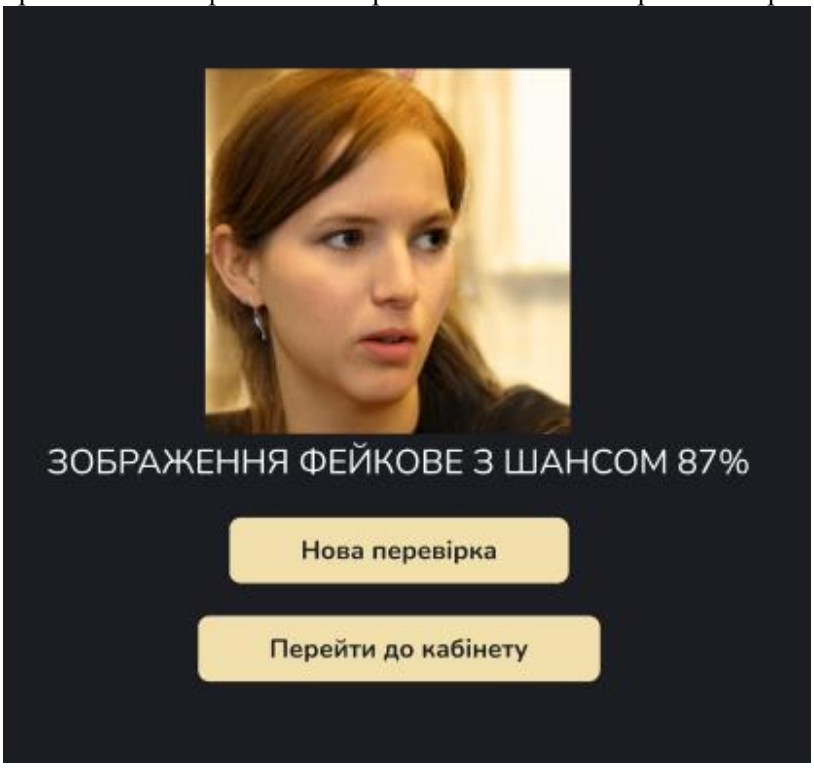


Рисунок 4.15 – Правильно розпізнане фейкове зображення

В даному випадку зображення було розпізнано правильно, це сталось по декільком причинам:

– волосся людини на зображенні виглядає розмазаним, неначе його недомалював штучний інтелект;

– очі людини на зображенні мають забагато артефактів, що вказує на можливі маніпуляції або генерацію зображення нейронною мережею;

– зуби на зображенні виглядають змазаними, що також вказує на зовнішній вплив на зображення.

Але система не є ідеальною і завжди є причини розвиватись далі, тому розглянемо ситуацію, коли система не змогла правильно розпізнати фейкове зображення (рис. 4.16).

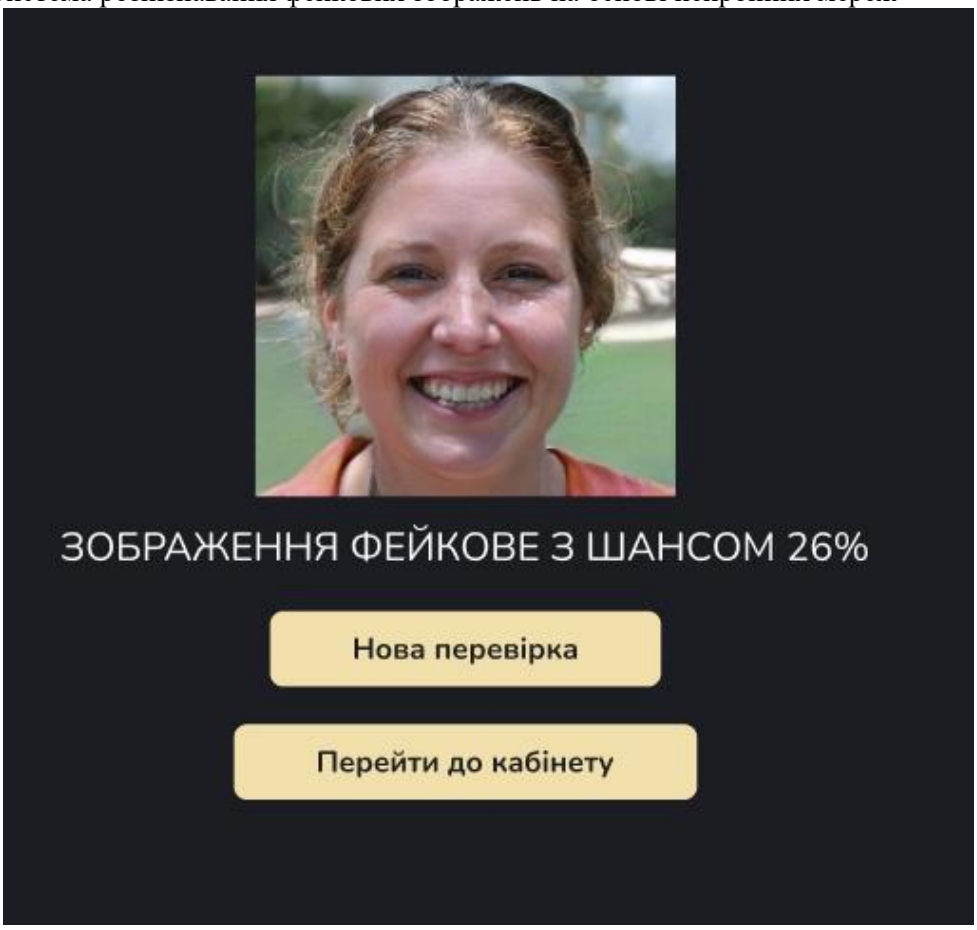


Рисунок 4.16 – Неправильно розпізнане фейкове зображення

Як можна побачити, то дане зображення було розпізнано неправильно, оскільки насправді воно є фейковим. Даний результат можливо був спричинений наступними факторами:

– волосся людини на фото є деталізований та не має розмиття, кожну волосину можна спокійно роздивитись;

– міміка максимально зімітована людська, передані всі ямочки та складки на обличчі, що робить розпізнавання зображення важчим, оскільки дана міміка також є важливим аспектом під час розпізнавання;

– риси обличчя та його форма нагадують майже ідеально людські, що також спричиняє неправильні результати в системі;

– зуби та губи є дуже деталізованими, що також викликає неправильні роздуми у нейронних мережах.

Висновки до розділу 4

У четвертому розділі кваліфікаційної магістерської роботи було розглянуто процес створення системи від етапу її дизайну до реалізації, в якості програмного забезпечення. На початку було створено мокапи сторінок та елементів системи, на основі яких в подальшому було здійснено розробку користувацького інтерфейсу системи. Було створено дизайни для входу в систему, реєстрації, дизайн головного меню, дизайн вікна перевірки зображення, а також особистого кабінету користувача.

Було продемонстровано процес тренування моделі нейронної мережі на основі обраного датасету з зображеннями, також показано прогрес точності моделі протягом кожної епохи тренування.

Для використання натренованої моделі нейронної мережі було використано бібліотеку TensorFlow.js, яка дозволила конвертувати модель у формат JSON, що вирішило проблему з підтримкою розширення моделі для використання в системі, та дозволило імплементувати нейронну мережу в систему.

Наприкінці було проведено тестування розробленої системи, що дало змогу перевірити систему на можливі помилки, а також перевірити її на правильність її функціональної частини. Тестування було пройдено успішно, що вказує на те, що програмний продукт було розроблено успішно.

ВИСНОВКИ

Під час виконання кваліфікаційної магістерської робот було розроблено систему виявлення фейкових зображень на основі нейронних мереж. Для досягнення поставленої мети роботи було виконано наступні завдання:

– проведення аналізу необхідних критеріїв інформації, необхідних для пошуку та аналізу фейкових зображень;

– пошук датасету для навчання нейронних мереж, який складається з зображень різного рівня складності, змісту;

– аналіз існуючих можливостей для навчання нейронних мереж та вибір найкращого методу;

– моделювання ПЗ для перевірки зображень на предмет їх фейковості;

– тренування модель ШІ з метою виявлення фейкових зображень;

– розробка прототипу програмного забезпечення для виявлення фейкових зображень;

– тестування та аналіз роботи створеного програмного забезпечення.

На початку виконання роботи було проаналізовано існуючі програмні рішення на ринку розпізнавачів фейкових зображень. Було описано їх переваги та недоліки, на основі яких було сформульовано подальше ТЗ, за яким розроблялась система.

Далі було створено IDEF0 та DFD діаграми, на яких було описано функціональні аспекти системи, що розроблялась. На IDEF0 діаграмі було описно процес підготовки зображення та його аналізу на предмет фейковості. На DFD діаграмі було показано всі потоки дані у системі розпізнавання зображень,

Для успішного виконання цих робіт було створено UML-діаграм ддя показу архітектури системи:

– діаграму класів;

– діаграму розгортання;

– діаграму компонентів;

– діаграму пакетів;
– діаграму прецедентів;

Після завершення роботи над проєктом було проведено його тестування, що дозволило знайти всі можливі помилки під час його роботи та їх було виправлено. Тестування системи включало перевірку її функціональності, продуктивності, стабільності та безпеки. Особливу увагу приділено оцінці якості моделі ШІ для виявлення фейкових зображень. Було виконано тестування на незалежному датасеті для оцінки точності, чутливості та специфічності моделі.

Проведений аналіз результатів роботи системи продемонстрував її здатність ефективно розпізнавати фейкові зображення з високим рівнем точності, хоч і невеликий проблем у випадках, коли зображення мають низьку якість, або коли неможливо повністю розпізнати контури персонажа на зображенні. Це свідчить про правильний вибір архітектури нейронної мережі, методів її тренування та обробки даних, але також і вказує на те, що систему треба дотреновувати вже на основі даних, отриманих від користувачів для того, щоб добитись її максимальної ефективності.

Таким чином, розроблена система відповідає поставленій меті і завданням роботи, забезпечує ефективне виявлення фейкових зображень та демонструє хороший потенціал для її використання у сфері медіа та журналістики, кібербезпеки та просто серед звичайних користувачів мережі Інтернет.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Aristeidis Bampakos, Pablo Deeleman. Learning Angular - Fourth Edition: A no-nonsense guide to building web applications with Angular. Packt Publishing; 4th edition. 2023. 446 p.

2. Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media; 3rd edition. 2022. 816 p.

3. Thomas Hathaway, Angela Hathaway. Data Flow Diagrams - Simply Put!: Process Modeling Techniques for Requirements Elicitation and Workflow Analysis (Advanced Business Analysis Topics). CreateSpace Independent Publishing Platform. 2016. 118 p.

4. David A. Marca, Clement L. McGowan. IDEF0 and SADT: A Modeler's Guide. OpenProcess, Inc. 2005. 392 p.

5. Seth Weidman. Deep Learning from Scratch: Building with Python from First Principles. O'Reilly Media; 1st edition. 2019. 250 p.

6. Adam Freeman. Pro Angular 9: Build Powerful and Dynamic Web Apps. Appress; 4th edition. 2020. 1368 p.

7. Amita Kapoor, Antonio Gulli, Sujit Pal. Deep Learning with TensorFlow and Keras – Third Edition: Build and deploy supervised, unsupervised, deep, and reinforcement learning models. Packt Publishing; 3rd edition. 2022. 698 p.

8. Francois Chollet. Deep Learning with Python, Second Edition. Manning; 2nd edition. 2021. 504 p.

9. Oliver Duerr, Beate Sick, Elvis Murina. Probalistic Deep Learning: With Python, Keras and TensorFlow Probability. Manning; 1st edition. 2020. 296 p.

10. Vishal Rajput. Ultimate Neural Network Programming with Python: Create Powerful Modern AI Systems by Harnessing Neural Networks with Python, Keras, and TensorFlow. Orange Education Pvt Ltd. 2023. 401 p.

11. Zed A. Shaw. Learn Python the Hard Way. Addison-Wesley Professional; 5th edition. 2024. 352 p.

12. Hayden Van Der Post. Keras: Master Deep Learning with Keras. Reactive Publishing; 5th edition. 2024. 654 p.

13. Anirudh Koul, Siddha Ganju, Meher Kasam. Practical Deep Learning for Cloud, Mobile, and Edge: Real-World AI & Computer-Vision Projects Using Python, Keras & Tensorflow. O'Reilly Media; 1st edition. 2019. 583 p.

14. Dan Vanderkam. Effective TypeScript: 83 Specific Ways to Improve Your TypeScript. O'Reilly Media; 2nd edition. 2024. 410 p.

15. Josh Goldberg. Learning TypeScript: Enhance Your Web Development Skills Using Type-Safe JavaScript. O'Reilly Media; 1st edition. 2022. 317 p.

16. Jennifer Robbins. Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics. O' Reilly Media; 5th edition. 2018. 808 p.

17. Jason Beaird, Alex Walker, James George. The Principles of Beautiful Web Design. SitePoint; 4th edition. 2020. 282 p.

18. Nathan Rozentals. Mastering TypeScript – Fourth Edition: Build enterprise-ready, modular web applications using TypeScript 4 and modern frameworks. Packt Publishing; 4th edition. 2021. 538 p.

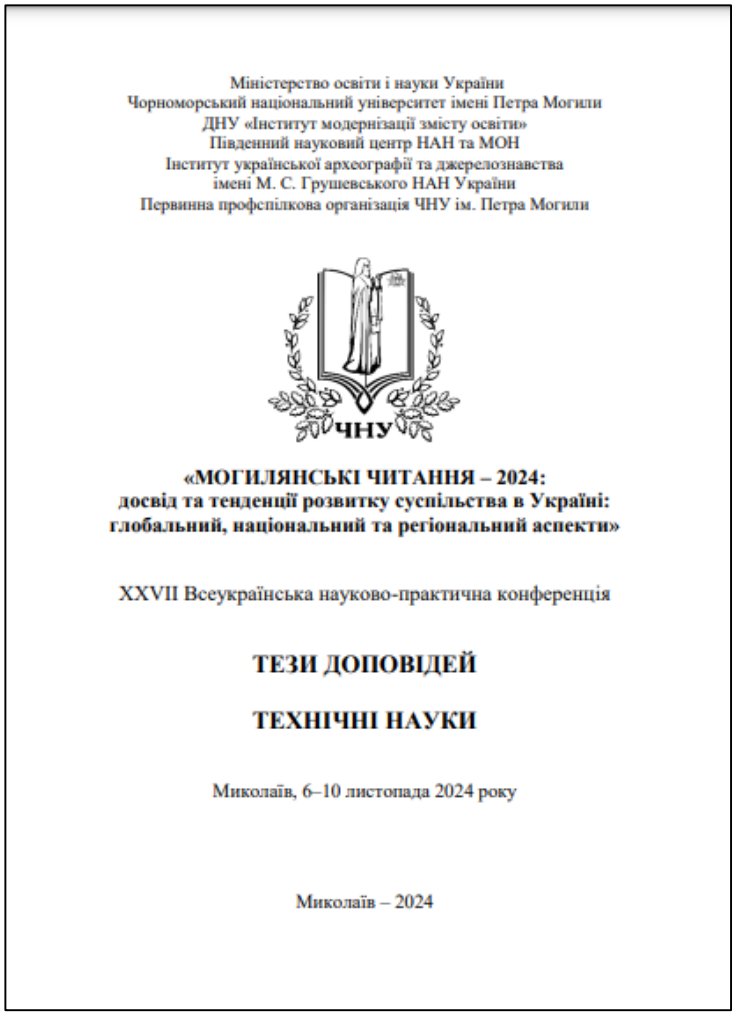
19. Greg Lim. TypeScript Crash Course for Beginners: Boost your Javascript projects with TypeScript: Learn about core types, generics, TypeScript and more. 2024. 91 p.

20. TransformaTech Institute. Understanding Deep Learning: Building Machine Learning Systems with PyTorch and TensorFlow: From Neural Networks (CNN, DNN, GNN, RNN, ANN, LSTM, GAN) to Natural Language Processing (NLP). Independently published. 2024. 397 p.

ДОДАТОК А

Апробація кваліфікаційної роботи

Результати досліджень були представлені на наступній конференції «Могилянські читання – 2024 : досвід та тенденції розвитку суспільства в Україні : глобальний, національний та регіональний аспекти. Технічні науки : XXVII Всеукр. наук.-практ. конф. : 6–10 листоп. 2024 р., м. Миколаїв : тези / М-во освіти і науки України ; ЧНУ ім. Петра Могили ; ДНУ «Ін-т модернізації змісту освіти» ; Півд. наук. Центр НАН та МОН України ; Ін-т укр. археографії та джерелознавства ім. М. С. Грушевського НАН України; Первинна профспілкова орг. ЧНУ ім. Петра Могили. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2024. – 280 с.»



Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ ім. Петра Могили

ЧНУ

**«МОГИЛЯНСЬКІ ЧИТАННЯ – 2024:
досвід та тенденції розвитку суспільства в Україні:
глобальний, національний та регіональний аспекти»**

XXVII Всеукраїнська науково-практична конференція

ТЕЗИ ДОПОВІДЕЙ

ТЕХНІЧНІ НАУКИ

Миколаїв, 6–10 листопада 2024 року

Миколаїв – 2024

Рисунок А.1 – Обкладинка збірника тез конференції Могилянські читання - 2024