ЛЬВІВСЬКИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ МІЖНАРОДНИХ ЕКОНОМІЧНИХ ВІДНОСИН
ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра комп'ютерних наук, вищої та прикладної математики

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розробка web-застосунку для покращення роздільної здатності зображення зі застосуванням нейромережі

Виконав: здобувач першого
(бакалаврського) рівня вищої освіти
ОПП "Комп'ютерні науки"
спеціальності 122 "Комп'ютерні науки"

_____ Ощипок Олексій Ігорович
(підпис)

Науковий керівник
_____ ст. викладач Крутяк М.Б.
(підпис)

Львів – 2025

ЛЬВІВСЬКИЙ ТОРГОВЕЛЬНО-ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

Факультет міжнародних економічних відносин
та інформаційних технологій
Кафедра комп'ютерних наук, прикладної та вищої математики
Перший (бакалаврський) рівень вищої освіти
Спеціальність 122 "Комп'ютерні науки"
Освітньо-професійна програма "Комп'ютерні науки"

“ЗАТВЕРДЖУЮ”

Завідувач кафедри комп'ютерних
наук, прикладної та вищої математики

_____ З. Г. Новосад
28.10. 2024 р.

Завдання
до кваліфікаційної роботи
Ощипку Олексію Ігоровичу
(прізвище, ім’я, по батькові)

1.Тема роботи **Розробка web-застосунку для покращення роздільної здатності зображення зі застосуванням нейромережі**
керівник роботи **Крутяк Марія Богданівна, *старший викладач***

затверджена наказом по Університету ***№ 1042/03 від "28" жовтня 2024 р***.

2.Термін здачі здобувачем вищої освіти роботи 5 червня 2025 р.
3. Вхідні дані до роботи: література, документація, матеріали та сайти Internet, що пов’язані з застосуванням нейромережі.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які підлягають розробці)
1) Дослідження методів покращення роздільної здатності зображень;
2) Аналіз можливостей нейромереж;
3) Опис розробленого web-застосунку.
5. Перелік графічного матеріалу до 12 слайдів у форматі PowerPoint
6. Консультанти розділів кваліфікаційної роботи

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

7. Дата видачі завдання 28.10. 2024 р.

Календарний план

№	Найменування етапів виконання кваліфікаційної роботи	Терміни виконання етапів	Примітка
1.	Отримання завдання і аналіз інформаційних джерел за темою роботи	до 11.12.2024	виконано
2.	Дослідження методів покращення роздільної здатності зображень	до 13.01.2025	виконано
3.	Аналіз можливостей нейромереж	до 07.02.2025	виконано
4.	Налагодження середовища для використання нейромережі та її навчання	до 01.04.2025	виконано
5.	Розробка web-застосунку	до 15.04.2025	виконано
6.	Тестування web-застосунку	до 22.04.2025	виконано
7.	Оформлення кваліфікаційної роботи. Подання роботи на перевірку керівнику	до 16.05.2025	виконано
8.	Доопрацювання кваліфікаційної роботи з врахуванням зауважень керівника. Подання роботи для перевірки на рівень запозичень	до 20.05.2025	виконано
9.	Подання кваліфікаційної роботи керівнику для отримання відгуку	до 27.05.2025	виконано
10.	Представлення кваліфікаційної роботи на зовнішнє рецензування. Подання кваліфікаційної роботи на кафедру комп'ютерних наук, прикладної та вищої математики	до 5.06.2025	виконано
11.	Попередній захист кваліфікаційної роботи	05.06.2025-06.06.2025	виконано
12.	Захист кваліфікаційної роботи на засіданні екзаменаційної комісії	згідно графіку роботи ЕК	

Здобувач вищої освіти Ощипок О. І.

Керівник роботи Крутяк М. Б.

Реферат

Робота складається із вступу, трьох розділів, висновків, списку використаних джерел з 16 посилань. Основний зміст роботи викладено на 80 сторінках і містить 9 рисунків та 3 таблиці.

Об’єктом дослідження є Web-технології;

Предметом дослідження є нейронні мережі, які застосовуються у задачах розпізнавання об'єктів на зображеннях;

Метою дослідження є створення програмного комплексу для автоматизації обробки зображень, виділення меж на зображенні у відтінках сірого.

В першому розділі досліджені методи покращення роздільної здатності зображень.

В другому розділі проведений аналіз можливостей нейромереж.

В третьому розділі розроблений web-застосунок покращення_роздільної здатності зображень

Перелік ключових слів: зображення, роздільна здатність, нейронні мережі.

Abstract

The work consists of an introduction, three sections, conclusions, a list of sources used with 16 references. The main content of the work is presented on 80 pages and contains 9 figures and 3 tables.

The object of the research is Web technologies;

The subject of the research is neural networks, which are used in the tasks of recognizing objects in images;

The purpose of the research is to create a software package for automating image processing, highlighting boundaries in grayscale images.

The first section investigates methods for improving image resolution;.

The second section analyzes the capabilities of neural networks.

In the third section, a web application for improving image resolution is developed

List of keywords: image, resolution, neural networks.

ЗМІСТ

ВСТУП 6

1. ДОСЛІДЖЕННЯ МЕТОДІВ ПОКРАЩЕННЯ РОЗДІЛЬНОЇ ЗДАТНОСТІ ЗОБРАЖЕНЬ 9

1.1 Загальна інформація 9

1.2. Засоби розробки WEB-додатків 16

1.3 Аналіз предметної області автоматизації обробки зображень виділення меж на зображенні у відтінках сірого 20

2. АНАЛІЗ МОЖЛИВОСТЕЙ НЕЙРОМЕРЕЖ 34

2.1. Огляд архітектури нейронних мереж для розпізнавання зображень 34

2.2. Типи нейронних мереж, що використовуються для розпізнавання зображень 40

2.3. Методи навчання нейронних мереж. 46

3. ОПИС РОЗРОБЛЕНОГО WEB-ЗАСТОСУНКУ 55

3.1. Структура HTML-документа 55

3.2. Стилізація інтерфейсу користувача (CSS) 59

3.3. Логіка взаємодії з користувачем (JavaScript) 69

ВИСНОВКИ 77

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 79

ВСТУП

Вплив всесвітньої комп'ютерної мережі Internet на сучасний світ не має історичних аналогів. Його сьогодення – це початок епохи електронного проникнення у всі сфери людського життя, це більше, аніж просто маркетингова кампанія, це основа нової філософії та нової ділової стратегії.

Цілком логічно припустити, що і з погляду реклами продукції чи послуги Інтернет – найбільш значущий ресурс. Більшість сучасних людей використовують Інтернет як найбільш доступне джерело інформації.

Web-технології повністю перевернули уявлення про роботу з інформацією та й з комп'ютером загалом. З'ясувалося, що традиційні параметри розвитку обчислювальної техніки – продуктивність, пропускна здатність, ємність пристроїв – не враховували головного "вузького місця" системи – інтерфейсу з людиною. Застарілий механізм взаємодії людини з інформаційною системою стримував впровадження нових технологій та зменшував вигоду від їх застосування. І тільки коли інтерфейс між людиною і комп'ютером був спрощений до природності сприйняття людиною, стався безпрецедентний вибух інтересу до можливостей обчислювальної техніки.

Зараз, коли кількість онлайнових користувачів оцінюється більш ніж у 500 мільйонів осіб (причому це число зростає зі швидкістю приблизно 7% на місяць), доводиться шукати способи використання нових технологій, які могли б сприяти в привернені та утриманні уваги користувачів, що подорожують безмежними просторами Web.

Ключовою сферою комп'ютерного зору, що знаходить своє застосування у багатьох сферах, включаючи медицину, автоматичне керування, автомобільну промисловість, робототехніку та багато інших, є розпізнавання зображень. При цьому завдання розпізнавання об'єктів на зображеннях є досить важким через складність і різноманітність зображень, які необхідно обробити.

До того, як нейронні мережі набули широкого поширення, розпізнавання об'єктів на зображеннях виконувалося за допомогою різних алгоритмів комп'ютерного зору, які були засновані на добуванні певних ознак із

зображення. Однак ці методи мали свої недоліки та обмеження: неможливість обробки великих обсягів даних, складність в аналізі зображень з різним контекстом та слабка можливість виявлення невідомих об'єктів.

Введення нейронних мереж у область розпізнавання зображень дозволило вирішити багато проблем. Нейронні мережі, на відміну класичних методів, не вимагають вилучення ознак із зображень вручну, а навчаються великих обсягах даних із відомими мітками. Вони також дозволяють обробляти зображення з різним контекстом, таким як фон, освітлення та масштаб, та можуть виявляти невідомі об'єкти.

Існує кілька типів нейронних мереж, які застосовуються у задачі розпізнавання об'єктів на зображеннях: згорткові нейронні мережі (Convolutional Neural Networks, CNNs), рекурентні нейронні мережі (Recurrent Neural Networks, RNNs) та глибокі нейронні мережі (Deep Neural Networks, DNNs). Кожен тип нейронної мережі має свої особливості та застосовується для вирішення відповідних завдань.

Актуальність теми: будь-яка форма обробки інформації, для якої вхідні дані представлені зображенням, наприклад фотографіями або відеокадрами називається обробкою зображень. Обробка зображень може здійснюватися як для отримання зображення на виході (наприклад, підготовка до поліграфічного тиражування, телетрансляції тощо), так і для отримання іншої інформації (наприклад, розпізнавання тексту, підрахунок числа і типу клітин у полі мікроскопа тощо);

Об’єкт дослідження: Web-технології;

Предмет дослідження: нейронні мережі, які застосовуються у задачах розпізнавання об'єктів на зображеннях;

Мета кваліфікаційної роботи: створення програмного комплексу для автоматизації обробки зображень виділення меж на зображенні у відтінках сірого.

Завдання:

- дослідження методів покращення роздільної здатності зображень;
- аналіз можливостей нейромереж;
- налагодження середовища для використання нейромережі та її навчання;
- розробка web-застосунку;
- тестування web-застосунку

Робота має **практичне значення**. В ній створений програмний комплекс для автоматизації обробки зображень виділення кордонів на зображенні у відтінках сірого.

Робота складається із вступу, трьох розділів, висновків, списку використаних джерел з 16 посилань. містить 3 таблиці 9 рисунків і викладена на 80 сторінках.

1. ДОСЛІДЖЕННЯ МЕТОДІВ ПОКРАЩЕННЯ РОЗДІЛЬНОЇ ЗДАТНОСТІ ЗОБРАЖЕНЬ

1.1 Загальна інформація

Інформація, що доступна користувачам мережі Internet, розташовується на комп'ютерах (Web-серверах), де встановлено спеціальне програмне забезпечення. Більшість цієї інформації організована як Web-сайти. Кожен з них має власну назву (адресу) в Internet. Web-сайт – це інформація, подана у певній формі, яка міститься на Web-сервері та має свою адресу. Для перегляду веб-сайтів на комп'ютері користувача застосовуються спеціальні програми, котрі називаються браузерами. Залежно від того, яку адресу сайту ми введемо у рядок "Адреса", браузер завантажуватиме у своє вікно відповідну інформацію [13].

Web-сайт складається із взаємопов'язаних Web-сторінок. Web-сторінка є текстовим файлом з розширенням *.htm, який містить текстову інформацію та спеціальні команди – HTML-коди, що визначають, в якому вигляді ця інформація відображатиметься у вікні браузера. Вся графіка, аудіо та відео інформація безпосередньо до Web-сторінки не належить і є окремою.

Сучасні web-додатки – це, переважно, портали, які надають послуги, якими не можна скористатися з іншого місця. Однією з незручностей подібних сервісів є складність обміну інформацією між користувачами. Зокрема, навіть контактні дані та іншу особисту інформацію доводиться на кожному такому сайті заново вводити.

Вимоги до Web-додатків [15]

Відправною точкою у web-проєкті є аналіз цілей сайту та функцій, які будуть запропоновані користувачеві.

Другим етапом буде побудова інформаційної архітектури сайту.

Після того, як будуть відомі всі матеріали сайту та його структура, можна перейти до дизайну навігації та сторінок

Переваги Web-додатків [14; 15]

1) Додаток абсолютно незалежний від тієї операційної системи, яка стоїть на комп'ютері користувача, тобто, фактично, є крос-платформним. Але є тут і один момент, який може створити певні складнощі при роботі з веб-додатками – це різні реалізації деяких специфікацій в браузерах, а також можливість налаштовувати різні параметри, на кшталт відображення шрифтів, що може призвести до некоректної роботи деяких додатків. Проте, слід зазначити, що цей мінус хоч і присутній, але істотного впливу на роботу більшості веб-додатків він не робить.

2) Сам факт існування веб-додатків повністю змінює метод поширення продукту. Тут виробники відходять від традиційних способів розповсюдження програмних продуктів шляхом продажу копій та встановлення їх на кожен комп'ютер користувачів. Тепер усе значно простіше: єдина версія програми розміщена на сервері, а всі користувачі мають доступ до неї, точніше, до її інтерфейсу користувача з будь-якого місця у світі. З будь-якого, де є Інтернет. При цьому користувачеві навіть не потрібно встановлювати нову версію програми – відразу після своєї появи вона доступна всім, причому багато хто може й не помітити будь-яких змін, тим паче якщо ці зміни не стосуються зовнішнього вигляду інтерфейсу. У всьому цьому чітко видно й позитивний момент для розробників — їм не потрібно перейматися сумісністю версій своїх додатків, оскільки всі користувачі одночасно отримують доступ і працюють з останньою версією програми.

3) Для користувача немає потреби встановлювати та налаштовувати програмне забезпечення – все вже встановлено на серверах та налаштовано розробниками. Все, що потрібно від користувача, це змінити на власний смак кілька ключових параметрів. Це дуже приємно для користувачів, оскільки більшість не любить возитися з налаштуваннями і віддає перевагу програмним продуктам, повністю готовим до використання одразу після їх інсталяції, хоча є й такі, що вважають за краще повністю налаштувати програму під свій смак і потреби. Проте у випадку веб-додатків ми позбавлені навіть процесу інсталяції.

4) Для роботи з додатком від користувача, за великим рахунком, нічого не потрібно. Хіба що комп'ютер та встановлений браузер. Але тут проблем немає – Інтернет-браузер уже є в будь-якій операційній системі, і для доступу до необхідного веб-додатку досить просто завантажити його URL-адресу в браузер. Використання веб-застосунків значною мірою знімає обмеження, що накладаються на апаратну частину комп'ютера. Тобто певні системні вимоги до ПК все ж таки є, але їх рівень автоматично досягнуто комп'ютером, щойно на ньому вже запущені ОС та браузер.

5) Враховуючи, що основна частина веб-програми сконцентрована на сервері в одному місці, набагато простіше займатися його налаштуванням, не потрібно утримувати величезні команди фахівців технічної підтримки, які займаються консультаціями користувачів та налаштуванням програми на комп'ютерах у всьому світі. Це набагато менш витратно у фінансовому плані і значно ефективніше. При цьому користувачеві невидима архітектура програми, у будь-який момент можна додати будь-яку кількість серверів, на яких встановлена основна складова програми, додати обчислювальні потужності, і цього навіть не помітить.

Таким чином, веб-додатки мають велику кількість плюсів за відсутності видимих мінусів, найбільшим і очевидним з яких є неможливість використання програм за відсутності доступу до Всесвітньої павутини.

Огляд методів підвищення роздільної здатності зображень.

Класичні методи підвищення роздільної здатності зображень, такі як інтерполяція, ґрунтуються на математичних алгоритмах, які оцінюють нові піксельні значення на основі сусідніх. Ці методи прості у впровадженні та швидкі, але часто мають обмеження в якості отриманого зображення. Розглянемо основні типи інтерполяції: [10-12]

1. Найближчого сусіда (Nearest Neighbor Interpolation)

Цей метод замінює новий піксель найближчим сусідом у вихідному зображенні.

Алгоритм:

Кожен новий піксель отримує значення існуючого пікселя, що знаходиться ближче за всі інші.

Переваги:

Надзвичайно швидкий і простий у реалізації.

Зберігає оригінальні кольори без усереднення.

Недоліки:

Створює "блокові" артефакти, особливо помітні на діагональних лініях.

Якість низька, не підходить для обробки складних текстур.

2. Лінійна інтерполяція (Bilinear Interpolation)

У цьому методі значення нового пікселя розраховується як зважене середнє значення чотирьох сусідніх пікселів.

Алгоритм:

Новий піксель отримує значення як двовимірну інтерполяцію по X та Y осях.

Переваги:

Плавніші переходи між пікселями.

Придатний для швидких обчислень із кращою якістю, ніж у найближчого сусіда.

Недоліки:

Розмиття зображення, особливо помітне на різких краях.

3. Бікубічна інтерполяція (Bicubic Interpolation)

Цей метод використовує 16 сусідніх пікселів, враховуючи як значення, так і похідні для отримання плавніших переходів.

Алгоритм:

Розраховуються ваги для кожного сусіднього пікселя, залежно від його відстані до нового пікселя.

Переваги:

Забезпечує значно краще відновлення деталей, ніж лінійна інтерполяція.

Зменшує розмиття.

Недоліки:

Повільніший, ніж методи найближчого сусіда та лінійний.

Може створювати артефакти "дзвону" навколо країв (ringing effect).

4. Ланцош інтерполяція (Lanczos Interpolation)

Цей метод використовує синусоїдальні функції для обчислення ваг і враховує більше сусідніх пікселів.

Алгоритм:

Використовує спеціальне вікно (наприклад, Lanczos-2 чи Lanczos-3) для зваженого усереднення.

Переваги:

Висока якість зображення, особливо для складних текстур.

Добре зберігає деталі.

Недоліки:

Висока обчислювальна вартість.

Іноді може створювати артефакти.

Таким чином, класичні методи підвищення роздільної здатності працюють добре для простих задач, але їх обмеження в деталізації та створення артефактів ускладнюють використання в реальних сценаріях, де потрібна висока якість. У таких випадках на допомогу приходять методи на основі глибокого навчання.

Сучасні методи підвищення роздільної здатності зображень на основі глибокого навчання (Deep Learning) забезпечують значно кращу якість у порівнянні з класичними підходами. Вони використовують великі обсяги даних та потужні обчислювальні ресурси для навчання моделей. Розглянемо основні методи [4-6].

1. Super-Resolution Convolutional Neural Network (SRCNN)

Це один із перших методів, який застосовує згорткові нейронні мережі (CNN) для суперроздільної здатності.

Архітектура:

SRCNN складається з трьох основних шарів:

1. **Patch Extraction and Representation:** Вхідне зображення розширюється (наприклад, за допомогою бікубічної інтерполяції), а потім виділяються локальні патчі.

2. **Non-Linear Mapping:** Застосовується нелінійна трансформація для відновлення деталей.

3. **Reconstruction:** Відновлення кінцевого зображення високої роздільної здатності.

Переваги:

- Простота реалізації.
- Значне покращення якості порівняно з класичними методами.

Недоліки:

- SRCNN може не враховувати глобальний контекст.
- Обмежена здатність відновлення дуже складних деталей.

2. Generative Adversarial Networks (GANs) для Super-Resolution (наприклад, ESRGAN)

GANs є більш сучасним і потужним підходом для підвищення роздільної здатності. ESRGAN (Enhanced Super-Resolution GAN) – вдосконалена версія SRGAN (Super-Resolution GAN).

Архітектура:

GAN складається з двох моделей:

1. **Генератор:** Створює зображення високої роздільної здатності зі зображення низької роздільної здатності.

2. **Дискримінатор:** Навчається відрізняти реальні зображення від згенерованих.

У ESRGAN додано кілька покращень:

Residual-in-Residual Dense Blocks (RRDB): Підвищують ефективність генератора.

Perceptual Loss: Враховує особливості зображення на рівні фільтрів високого порядку, забезпечуючи кращу якість візуального сприйняття.

Переваги:

Висока якість текстур та деталей.

Генерація реалістичних зображень, які виглядають природно.

Недоліки:

Велика потреба в обчислювальних ресурсах.

Можливість появи артефактів через суперечливість між генератором і дискримінатором.

3. Трансформери для обробки зображень (Image Transformers)

Останнім часом трансформери, такі як Vision Transformers (ViT) або Swin Transformers, почали активно застосовуватися для задач суперроздільної здатності [1, 2].

Архітектура:

Self-Attention: У трансформерах ключову роль відіграє механізм самоуваги, який дозволяє враховувати як локальні, так і глобальні залежності між пікселями.

Патчі: Зображення розбивається на невеликі блоки (патчі), які обробляються як послідовності.

Застосування в суперроздільній здатності:

SwinIR: Модель, яка використовує Swin Transformers для ефективного підвищення роздільної здатності.

U-формні архітектури: Для поліпшення відновлення деталей.

Переваги:

Добре справляються зі складними структурами та текстурами.

Враховують контекст на всьому зображенні.

Недоліки:

Високі обчислювальні витрати.

Необхідність великих наборів даних для навчання.

Порівняння методів підвищення роздільної здатності зображень наведено в таблиці 1.1.

Таблиця 1.1 – Порівняння методів роздільної здатності зображень

Метод	Якість деталей	Реалістичність	Обчислювальні витрати	Застосування
SRCNN	Середня	Середня	Низькі	Основні задачі
ESRGAN	Висока	Висока	Високі	Реалістичні текстури
Трансформери	Дуже висока	Дуже висока	Дуже високі	Комплексні текстури

1.2. Засоби розробки WEB-додатків

HTML

Мова розмітки гіпертексту (Hypertext Markup Language), або, як його частіше називають, HTML, – це мова комп'ютера, що лежить в основі World Wide Web (Всесвітньої мережі). Завдяки мові HTML будь-який текст можна розмітити, перетворивши його на гіпертекст з подальшою публікацією в Web.

Мова HTML має власний набір символів, за допомогою яких веб-браузери відображають сторінку. Ці символи, що звуться дескрипторами, включають елементи, необхідні для створення гіперпосилань [4; 7; 9].

Однією з відмінних рис HTML-документів є те, що сам документ містить лише текст, а всі інші об'єкти вбудовуються в нього в момент відображення браузером за допомогою спеціальних тегів і зберігаються окремо. При збереженні HTML-файлу в місці розміщення документа створюється папка, куди поміщаються супутні йому графічні елементи оформлення.

CSS

Каскадні таблиці стилів або CSS (від англійської Cascading Style Sheets) – це досить пізнє нововведення. Якщо сам HTML з'явився 1989 року, то таблиці стилів - лише 1997. Мало того, таблиці стилів не вважаються частиною HTML. Пов'язано це з тим, що WWW була створена вченими як засіб для обміну текстовими документами, а HTML був мовою, за допомогою якої ці документи створювали. Для науковців головним був зміст документа, а не його

оформлення. Тому перші версії HTML не включали навіть засобів для розміщення на Web-сторінках графічних зображень, не кажучи вже про складне форматування тексту. Але час минав, і в Інтернет прийшов продукт, який відразу ж вимагав від Web-дизайнерів «зробити його красиво». А Web-дизайнери, у свою чергу, вимагали від розробників стандарту HTML засобів, що полегшують роботу.

Каскадні таблиці стилів або просто таблиці стилів (CSS - Cascading Style Sheets) – це набір правил, що описують форматування різних фрагментів HTML-коду та зберігаються окремо від нього. Одне таке правило, що відображає форматування одного фрагмента або однотипної групи фрагментів коду, називається стилем. Таблиці стилів описуються спеціальною мовою CSS і зберігаються у спеціальних файлах з розширенням CSS, хоча можуть бути впроваджені в саму Web-сторінку.[5; 7;10]

Усього стандарт CSS визначає три способи задання стилю для елемента сторінки. Перерахуємо їх та дамо короткий опис кожного способу.

1. Зовнішня (або прив'язана) таблиця стилів. Стилі зберігаються в окремому файлі з розширенням CSS і прив'язуються до Web-сторінки за допомогою спеціального тегу <LINK>.

2. Внутрішня (чи впроваджена) таблиця стилів. Таблиця стилів в цьому випадку має той самий формат, що і зовнішня, але розташовується в секції з заголовком тієї ж Web-сторінки і розміщується всередині спеціального тегу <STYLE>.

3. Внутрішні (також вбудовані чи впроваджені) стилі. Визначення стилю поміщається у потрібний тег, для цього використовується спеціальний атрибут STYLE.

Правило каскадності: якщо якийсь стиль буде визначено одночасно два чи три рази, то визначення стилю, зробленому у зовнішній таблиці, додає визначення, зроблене у внутрішній таблиці.

А якщо визначення стосується того ж атрибуту, береться визначення, зроблене у внутрішній таблиці.

Ajax.

Ajax розшифровується як Asynchronous Javascript And XML (Асинхронні Javascript і XML) і технологією в строгому сенсі слова не є. Якщо у стандартному web-додатку обробкою всієї інформації займається сервер, тоді як браузер відповідає лише за взаємодію з користувачем, передачу запитів та виведення HTML, то в Ajax-додатку між користувачем і сервером з'являється ще один посередник – движок Ajax. Він визначає, які запити можна опрацювати "на місці", а за якими необхідно звертатися на сервер. Поведінка сервера також змінилася. Якщо раніше на кожен запит сервер видавав нову сторінку, то тепер він відсилає ті дані, які потрібні клієнту, а HTML з них прямо в браузері формує движок Ajax.

Асинхронність виявляється в тому, що далеко не кожен клік користувача доходить до сервера, причому зворотне теж справедливе — далеко не кожна реакція сервера обумовлена запитом користувача. Більшість запитів формує двигун Ajax, причому його можна написати так, що він буде завантажувати інформацію, передбачаючи дії користувача.[2; 4]

Де варто використовувати Ajax:

Форми.

Навігація у вигляді "дерева".

Пошук за файлами та папками.

JavaScript.

Мова програмування JavaScript розроблена фірмою Netscape для створення інтерактивних HTML-документів. Це об'єктно-орієнтована мова розробки додатків, що вбудовуються, що виконуються як на стороні клієнта, так і на стороні сервера. Синтаксис мови дуже схожий на синтаксис Java – тому його називають Java-подібним.

Основні сфери застосування JavaScript поділяються на наступні категорії: динамічне створення документа за допомогою сценарію;

Оперативна перевірка достовірності заповнюваних користувачем полів форм HTML до передачі їх на сервер;

Створення динамічних HTML-сторінок спільно з каскадними таблицями стилів та об'єктною моделлю документа;

Взаємодія з користувачем при вирішенні "локальних" завдань, що вирішуються додатком JavaScript, вбудованим в HTML-сторінку

React.

React (іноді React.js або ReactJS) — JavaScript-бібліотека з відкритим вихідним кодом для розробки інтерфейсів користувача.

React розробляється та підтримується Facebook, Instagram та спільнотою окремих розробників та корпорацій.

React може використовуватися для розробки односторінкових та мобільних додатків. Його мета – надати високу швидкість, простоту та масштабованість. Як бібліотека для розробки Користувальницьких інтерфейсів React часто використовуються з іншими бібліотеками, такими як Redux.

JetBrains WebStorm.

WebStorm – це середовище для розробки на JavaScript, яке ідеально підходить як для клієнтської частини, створення додатків на Node.js, так і для мобільних додатків на React Native.

Основна перевага WebStorm – зручний та інтелектуальний редактор для JavaScript, HTML та CSS, який також підтримує TypeScript, CoffeeScript, Dart, Less, Sass та Stylus, а також фреймворки, наприклад, Angular, React та Vue.js.

WebStorm, як і інші IDE на основі IntelliJ IDEA, робить розробку легшою та зручнішою. WebStorm забезпечує підсвічування та автодоповнення коду, перевіряє його на наявність помилок, допомагає швидко переміщатися проєктом та безпечно вносити зміни за допомогою рефакторингу. У WebStorm є інструменти для налагодження коду та інтеграція із системами контролю версій.

WebStorm по-справжньому розуміє структуру вашого проєкту та код, виявляє можливі проблеми ще до того, як ви відкрили проєкт у браузері, і пропонує їх вирішення [2]

Nodejs.

Nodejs (або просто Node) - це серверна платформа для роботи з JavaScript через двигун V8. JavaScript виконує дії на стороні клієнта, а Node – на сервері. За допомогою Node можна писати повноцінні програми. Node вміє працювати із зовнішніми бібліотеками, викликати команди з коду на JavaScript та виконувати роль веб-сервера.

Переваги Node над іншими серверними технологіями (Java, PHP, C#) досить прості: з Node легше масштабуватися. При одночасному підключенні до сервера тисяч користувачів Node працює асинхронно, тобто правильно розподіляє пріоритети та ресурси. Java ж, наприклад, виділяє на кожне з'єднання окремий потік.

Nodejs Express.

Nodejs Express - це веб-фреймворк, написаний на JavaScript і працюючий в середовищі виконання node.js. Цей модуль висвітлює деякі ключові переваги цього фреймворку, встановлення середовища розробки та виконання основних завдань веб-розробки та розгортання. Деякі переваги даного фреймворку:

- Дає змогу налаштувати посередників для відповіді на запити HTTP.

- Визначає таблицю маршрутизації, яка використовується для виконання різних дій на основі методу HTTP та URL-адреси.

- Дозволяє динамічно генерувати HTML-сторінки на основі передачі аргументів шаблонам.

1.3 Аналіз предметної області автоматизації обробки зображень виділення меж на зображенні у відтінках сірого

Обробка зображення, основні поняття та визначення.

Будь-яка форма обробки інформації, для якої вхідні дані представлені зображенням, наприклад фотографіями або відеокадрами називається обробкою

зображень. Обробка зображень може здійснюватися як для отримання зображення на виході (наприклад, підготовка до поліграфічного тиражування, телетрансляції і т.д.), так і для отримання іншої інформації (наприклад, розпізнавання тексту, підрахунок числа і типу клітин в полі мікроскопа і т.д.) [1].

Світло, що випромінюється або відображається об'єктами і проектується на ділянку сітківки нашого ока, має складний спектральний розподіл. Сітківка ока складається з рецепторів трьох видів, чутливих до різних областей видимого світла. Тому набору із трьох чисел достатньо, щоб описати колір. Плоске зображення – це функції залежності кольору від координат, причому координати набувають дискретних значень. Відомі два основні підходи до формування та зберігання зображень: растрова графіка та векторна графіка, також існують їх комбінації [2; 5; 8].

Сьогодні редагування зображень проводиться переважно на комп'ютері растровими редакторами в цифровому вигляді. Сучасні редактори не позбавлені недоліків, однак їх використання дозволяє вирішити більшість завдань, що виникають при редагуванні зображень.

Як джерела зображень у сучасному світі виступають: зображення з цифрового фотоапарата, негативні фотоплівки та слайди після оцифрування за допомогою сканера, фотобанки, сервери файлообміну та пошукові системи. На цих ресурсах нерідко можна зустріти зображення без обмежень використання.

Основні терміни, що використовуються для виділення кордонів на зображенні, наведені в глосарії, представленому в таблиці 1.2 [2]

Таблиця 1.2 – Глосарій предметної галузі

№	Термін	Визначення
1	Обробка зображень	Будь-яка форма обробки інформації, для якої вхідні дані представлені зображенням, наприклад фотографіями або відеокадрами.
2	Виділення кордонів	Термін у теорії обробки зображення та комп'ютерного зору, частково в області пошуку об'єктів та виділення об'єктів, ґрунтується на алгоритмах, що виділяють точки цифрового зображення, в яких різко змінюється яскравість або є інші види неоднорідностей.

Продовж. табл. 1.2

3	Градієнт	Вектор, своїм напрямом вказує напрямок якнайшвидшого зростання деякої величини, значення якої змінюється від однієї точки простору до іншої, а за величиною дорівнює швидкості зростання цієї величини в цьому напрямку.
4	Крапка	Найпростіший графічний примітив, що має нульову розмірність. Крапка характеризується лише координатами свого розташування.
5	Згладжування	Технологія, що використовується в обробці зображень з метою зробити межі кривих ліній візуально більш гладкими, прибираючи зубці, що виникають при растеризації на краях об'єктів
6	Вектор	Упорядкована пара точок, одна з яких називається початком, друга – кінцем вектора.
7	Піксель	Найменший логічний елемент двовимірного цифрового зображення у растровій графіці.
8	Растрове зображення	Являє собою сітку пікселів або кольорових точок на комп'ютерному моніторі, папері та інших пристроях і матеріалах, що відображають.
9	Графічний формат	Спосіб запису графічної інформації

Растрова графіка.

Растрова графіка описує зображення за допомогою кольорових точок, що називаються пікселями, розташованих на сітці [2]. Під час редагування растрової графіки відбуваються зміни пікселів. Растрова графіка залежить від роздільної здатності, бо інформація, що описує зображення, прив'язана до сітки певного розміру. Під час редагування растрової графіки якість її подання може змінюватись. Зокрема, зміна розмірів растрової графіки може спричинити розмиття країв зображення, оскільки пікселі перерозподілятимуться на сітці. Відображення растрової графіки на пристрої з нижчою роздільною здатністю, ніж роздільна здатність самого зображення, погіршить його якість [4].

Основою растрового представлення графіки є піксель (точка) з вказаним кольором. При описі, наприклад, червоного еліпса на білому тлі доводиться вказувати колір кожної точки як еліпса, так і фону. Зображення представлено у

вигляді великої кількості точок – чим їх більше, тим візуально якісніше зображення і більший розмір файлу. Тобто. одна й та сама картинка може бути представлена з кращою або найгіршою якістю відповідно до кількості точок на одиницю довжини – роздільною здатністю (зазвичай, точок на дюйм – dpi або пікселів на дюйм – ppi).

Окрім того, якість визначається ще й кількістю кольорів та відтінків, які може набувати кожна точка зображення. Чим більшою кількістю відтінків характеризується зображення, тим більше розрядів потрібно для їхнього опису. Червоний може бути кольором номер 001, а може й – 00000001. Таким чином, чим якісніше зображення, тим більший розмір файлу.

Растрове подання зазвичай використовують для фотографій з великою кількістю деталей або відтінків. На жаль, масштабування таких картинок у будь-який бік, як правило, погіршує якість. При зменшенні кількості точок губляться дрібні деталі та деформуються написи (щоправда, це може бути помітно при зменшенні візуальних розмірів самої картинки – тобто. збереженні роздільності). Додавання пікселів призводить до погіршення різкості та яскравості зображення, тому що новим точкам доводиться надавати відтінки, середні між двома й більше квітами, що межують. Поширені формати .tif, .gif, .jpg, .png, .bmp [13].

Виділення контурів.

Однією з основних та найважливіших цілей цифрової обробки зображень є розпізнавання об'єктів, наявних на них. Можливість розрізняти об'єкти закладена у високій інформативності зображення. Значною мірою саме це спонукає розробників різних інформаційних систем до використання зображень як способу подання результатів спостереження [3].

Разом з тим зображення, що пред'являються до обробки, містять багато надмірних малоінформативних даних, які займають, однак, великі обсяги пам'яті, і вимагають виконання великої кількості обчислень при спробі їх використання для розпізнавання. Теоретично існують методи розпізнавання образів, які дозволяють з'ясовувати ступінь інформативності тих чи інших

ознак. Проте на сьогодні ці методи не знайшли широкого застосування в розпізнаванні зображень.

Значно ширше застосовуються методи скорочення надмірності, що спираються на специфічні особливості зорового сприйняття зображень. Вважається, що суб'єктивне сприйняття сцени, що спостерігається, відбувається через її подання у вигляді окремих однорідних областей (тобто сегментацію) та виділення контурних ліній. Контурні, або граничні лінії поділяють на зображенні ділянки з різними властивостями, тому виділення контурів іноді розглядається як попередня обробка, спрямована на подальше виконання сегментації. Разом з тим, контур, що утворюється в деяких випадках, може й самостійно ефективно використовуватися для розпізнавання, оскільки інформація, що міститься в ньому, принаймні, з точки зору зорового сприйняття, цілком достатня для вирішення багатьох завдань такого типу [15].

Основною метою виявлення різких змін яскравості зображення є фіксація важливих подій та змін світу. Вони можуть відображати різні припущення про модель формування зображення, зміни яскравості зображення можуть вказувати на:

- зміни глибини;
- зміни орієнтації поверхонь;
- зміни у властивостях матеріалу;
- відмінність у освітленні сцени.

В ідеалі, результатом виділення контурів є набір з'єднаних кривих, які вказують на межі предметів, граней та відбитків на поверхні, а також криві, що показують зміни положення поверхонь. Таким чином, застосування фільтра виділення меж до зображення може значно зменшити об'єм даних, що підлягають обробці, через те, що відфільтрована частина зображення вважається менш значущою, а найважливіші структурні особливості зображення зберігаються. Однак, не завжди вдається виділити межі в зображеннях реального світу середньої складності. Контури, виділені з таких зображень, часто мають недоліки, такі як фрагментація, відсутність кордонів

або наявність помилкових меж, які не відповідають об'єкту, що досліджується [4].

Межі, виділені на двовимірному зображенні тривимірної сцени, можуть бути розділені на ті, що залежать або не залежать від точки огляду. Незалежні від точки огляду контури зазвичай відображають властивості, притаманні об'єктам тривимірної сцени, такі як колір поверхні та її форма. Залежні від точки огляду контури можуть змінюватися при зміні точки огляду та відображають геометрію сцени, як, наприклад, об'єкти, що перекриваються.

Звичайною межею може бути, наприклад, межа між ділянками червоного та жовтого кольору. З іншого боку, лінія може бути набором пікселів різного кольору на однотонному фоні. Тому лінія може мати кордони з кожного боку від неї [15; 16]

Актуальність виділення контурів зумовлена тим, що це є початковим етапом розпізнавання образів. У свою чергу, завдання розпізнавання образів розв'язується у таких сферах, як системи технічного зору, застосовні для розпізнавання тексту, виявлення дефектів виробів, розпізнавання облич, виявлення ДТП, управління автомобілем та інше.

Огляд існуючого програмного забезпечення для автоматизації виділення кордонів на зображенні.

В даний час існує безліч графічних редакторів для обробки зображень від багатофункціональних редакторів до вузькоспеціалізованих. Нижче розглянемо деякі сучасні редактори.

Adobe Photoshop (AP) – багатофункціональний графічний редактор, розроблений та розповсюджуваний фірмою Adobe Systems. Здебільшого працює з растровими зображеннями, проте має деякі векторні інструменти. Підтримується обробка зображень, з глибиною кольору 8 біт 16 біт, і 32 біт (використовуються числа одинарної точності з плаваючою комою). Можливе збереження у файлі додаткових елементів, як-от: напрямних, каналів, шляхів відсічення шарів, що містять векторні та текстові об'єкти. Файл може включати

профілі кольорів (ICC), функції перетворення кольору (transfer functions). Допускаються неквадратні пік селі [16].

Переваги:

- Photoshop надає користувачеві велику кількість функцій для обробки зображень;
- можливість роботи з різними графічними форматами;
- підтримує роботу з шарами.

Недоліки:

- висока вартість;
- складність в освоєнні.

GIMP (GNU Image Manipulation Program) – растровий графічний редактор, програма для створення та обробки растрової графіки та частковою підтримкою роботи з векторною графікою.

Типові завдання, які можна вирішувати за допомогою GIMP, включають створення графіки і логотипів, масштабування і кадрування фотографій, розмальовка, комбінування зображень з використанням шарів, ретушування і перетворення зображень в різні формати.

Переваги:

- має достатній функціональний набір для обробки зображень;
- безкоштовний;
- відкритий вихідний код;

– має можливість згортки із заданою матрицею.

Недоліки:

– не використовуються стандартні панелі керування Windows;

– інтерфейс, весь редактор складається лише із вікон інструментів.

Paint.NET – безкоштовний растровий графічний редактор малюнків та фотографій для Windows, розроблений на платформі .NET Framework. Підтримується робота із шарами, є можливість створення різноманітних

ефектів, чудово працює відкат. Дозволяє використовувати різні модулі, що завантажуються. Програма має російський інтерфейс.

Переваги:

– програма безкоштовна для поширення та використання;

– має багато модулів, що завантажуються;

Недоліки:

– потрібен час для освоєння інтерфейсу програмного продукту;

– виділення контуру йде окремим модулем, що завантажується.

Розробка математичної моделі обробки зображень та виділення контурів у відтінках сірого.

Оператори Собеля, Шарра, Прюїтта

Для виділення меж на зображенні у відтінках сірого застосовується безліч алгоритмів. Одним із таких алгоритмів є оператор Собеля.

Оператор Собеля є неточним наближенням градієнта зображення, але підходить для практичного застосування в багатьох задачах [5]. Точніше, оператор використовує значення інтенсивності тільки в околиці 3×3 кожного пікселя для одержання наближення відповідного градієнта зображення, і використовує тільки цілі значення вагових коефіцієнтів яскравості для оцінки градієнта [12].

Формула оператора Собеля (1.1): G_x і G_y – дві матриці, де кожна точка містить наближені похідні x і y. Вони обчислюються наступним чином шляхом множення матриці Gx і Gy та підсумовуванням обох матриць, в результаті отриманий результат записується в поточні координати x та y у нове зображення:

$$G = \sqrt{G_x^{\,2} + G_y^{\,2}}, \qquad (1.1)$$

Матриці G_y та G_x (1.2):

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A \text{ and } G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \quad (1.2)$$

Алгоритм оператора Прюіта [6] подібний до алгоритму оператора Собеля, за винятком використання іншої матриці (1.3):

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * A \text{ and } G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} * A \quad (1.3)$$

Алгоритм оператора Шарра подібний до алгоритму оператора Собеля, за винятком використання іншої матриці (1.4):

$$\begin{bmatrix} +3 & +10 & +3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix} \text{and} \begin{bmatrix} +3 & 0 & -3 \\ +10 & 0 & -10 \\ +3 & 0 & -3 \end{bmatrix} \quad (1.4)$$

Перехресний оператор Робертса

Перехресний оператор Робертса – один із алгоритмів виділення кордонів, який обчислює суму квадратів різниць між діагонально суміжними пікселями. Це може бути виконано згорткою зображення з двома ядрами[6]:

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \text{and} \begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \quad (1.5)$$

Перетворення кожного пікселя перехресним оператором Робертса може показати похідну зображення вздовж ненульової діагоналі, і комбінація цих перетворених зображень може розглядатися як градієнт від двох верхніх пікселів до двох нижніх. Оператор Робертса все ще використовується для швидкості обчислень, але він програє в порівнянні з альтернативами з його значною проблемою чутливості до шуму. Він дає лінії тонші, ніж інші методи виділення кордонів.

Внаслідок використання операції дискретного двовимірного диференціювання виходить нове значення, яке записується в нове фото.

Оператор Кенні.

Оператор Кенні використовує багатоступінчастий алгоритм виявлення широкого спектра кордонів у зображеннях [7].

Алгоритм детектора кордонів не обмежується обчисленням згладженого градієнта зображення. У контурі кордону залишаються лише точки максимуму градієнта зображення, а не максимальні точки, що лежать поряд із кордоном, видаляються. Тут також використовується інформація про спрямування кордону для того, щоб видаляти крапки саме поруч із кордоном та не розривати самий кордон поблизу локальних максимумів градієнта. Потім за допомогою двох порогів видаляються слабкі межі. Фрагмент кордону у своїй обробляється як ціле. Якщо значення градієнта де-небудь на фрагменті, що простежується, перевищить верхній поріг, то цей фрагмент залишається також «допустимою» кордоном і в тих місцях, де значення градієнта падає нижче цього порога, до тих пір поки вона не стане нижче нижнього порога. Якщо на всьому фрагменті немає жодної точки зі значенням великим верхнього порогу, він видаляється. Такий гістерезис дозволяє зменшити кількість розривів у вихідних межах. Включення в алгоритм Кенні шумопогашення з одного боку підвищує стійкість результатів, з другого – збільшує обчислювальні витрати і призводить до спотворення і навіть втрати подробиць кордонів. У алгоритмі застосовується розмиття зображення видалення шуму. Оператор Кенні використовує фільтр, який може бути добре наближений до першої похідної гауссіани.

$$B_x = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 5 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} * A \qquad (1.6)$$

Межі відзначаються там, де градієнт зображення набуває максимального значення. Вони можуть мати різний напрямок, тому алгоритм Кенні використовує чотири фільтри для виявлення горизонтальних, вертикальних та діагональних ребер у розмитому зображенні.

$$G = \sqrt{G_x^{\,2} + G_y^{\,2}}\,, \qquad (1.7)$$

$$\Theta = \arctan\left(\frac{G_y}{G_x}\right) \qquad (1.8)$$

Кут напрямку вектора градієнта округляється і може набувати таких значень: 0, 45, 90, 135.

Перед застосуванням детектора зазвичай перетворюють зображення на відтінки сірого.

Метод Лапласа.

Метод Лапласа [9] здійснює перемноження кожного елемента двовимірної апертури 3x3 на відповідний елемент так званої матриці Лапласа:

0¤	-1¤	0¤	¤	1¤	1¤	1¤	¤	-1¤	1¤	1¤	¤	1¤	1¤	1¤¤
-1¤	4¤	-1¤	¤	1¤	-2¤	1¤	¤	-1¤	-2¤	1¤	¤	-1¤	-2¤	1¤¤
0¤	-1¤	0¤	¤	-1¤	-1¤	-1¤	¤	-1¤	1¤	1¤	¤	-1¤	-1¤	1¤¤

Згортка зображення.

Фільтри, які були перераховані, а також безліч інших засновані на пакунку. Згортка - це операція, що показує "схожість" однієї функції з відбитою та зрушеною копією іншої. Поняття згортки узагальнюється для функцій.

Згортка – це операція обчислення нового значення вибраного пікселя, що враховує значення навколишніх пікселів. Для обчислення значення використовується матриця, яка називається ядром згортки. Зазвичай згортки є квадратною матрицею n*n, де n – непарне. Під час обчислення нового значення вибраного пікселя ядро згортки "прикладається" своїм центром до цього пікселя. Пікселі, що оточують, так само накриваються ядром. Далі

вираховується сума, де доданками є добутки значень пікселів на значення комірки ядра, що накрила цей піксель. Сума ділиться у сумі всіх елементів ядра згортки. Отримане значення є новим значенням вибраного пікселя. Якщо застосувати пакунок до кожного пікселя зображення, то в результаті вийде певний ефект, що залежить від обраного ядра пакунка.

Розробка структурно-функціональної моделі автоматизації обробки зображень та виділення на них меж у відтінках сірого на основі SADT-технології.

Бізнес-логіка – сукупність правил, принципів, залежностей поведінки об'єктів предметної галузі, реалізація правил та обмежень операцій, що автоматизуються. Реалізація предметної сфери в інформаційній системі [10].

Для опису бізнес логіки використовуємо методологію структурного аналізу та проектування SADT.

SADT – методологія структурного аналізу та проектування, що інтегрує процес моделювання, управління конфігурацією проекту, використання додаткових мовних засобів та керівництво проектом зі своєю графічною мовою [11].

Діаграма SADT складається з:

- Прямокутників (функції системи). Тут існує домінування, яке видно за розташуванням блоків. Кожен блок має номер відповідно до домінування. Домінування відбувається за часом або за управлінською функцією.

– Дуг – зображують взаємозв'язок між об'єктами.

Головною активністю предметної області є "Виділення кордонів на зображенні". На вхід подається вихідне зображення, вибір алгоритму та параметри згладжування.

Виконавцями активності є користувач системи. Усі дії виконавця мають бути регламентовані алгоритмами згладжування, алгоритмом перетворення до сірого, форматами, що підтримуються, та алгоритмом виділення кордонів. На виході отримуємо готове оброблене зображення із виділеними межами. У

таблиці 1.3 наведено опис контекстної структурно-функціональної діаграми нульового рівня основного бізнес-процесу.

Таблиця 1.3 – Опис контекстної структурно-функціональної діаграми нульового рівня процесу А0 "Виділення кордонів на зображенні"

№	Вхідні дані	Управління	Виконавець	Вихідні дані
А0	Початкове зображення.	Підтримуваний формат.	Користувач	Файл із обробленим зображенням
	Параметри згладжування. Вибір алгоритму.	Алгоритм згладжування. Алгоритм перетворення сірого. Алгоритм пошуку кордонів.		

При деталізації підсистеми або процесу деталізуюча діаграма як зовнішні джерела / приймачі даних може мати тільки ті компоненти (підсистеми, процеси, зовнішні сутності, накопичувачі даних), з якими має інформаційний зв'язок деталізована підсистема або процес на батьківській діаграмі.

Вимоги до програмної документації

Перелік програмної документації встановлюється стандартами. Нижче наведено список програмних документів та їх зміст.

Текст програми – запис програми з необхідними поясненнями та коментарями.

Опис програми – відомості про логічну структуру та функціонування програми.

Програма і методика випробувань – вимоги, підлягають перевірці під час випробування програми, і навіть порядок і способи контролю.

Технічне завдання – цей документ.

Пояснювальна записка – схема алгоритму, загальний опис алгоритму чи функціонування програми, і навіть обґрунтування прийнятих технічних і техніко-економічних рішень.

Експлуатаційні документи – опис застосування, посібник користувача.

Отже, у цьому розділі було проаналізовано принципи роботи систем, структуру та функціональність, а також взаємодію компонентів. Вивчено теоретичні основи розробки web-додатків. Розглянуто новітні веб-технології та етапи створення web-додатку, а також деякі питання щодо оптимізації та вибору засобів розробки. Розроблено структуру роботи web-додатків та технічно реалізовано даний алгоритм роботи.

Для розробки даного модуля вивчено різне програмне забезпечення та середовища розробки. Зокрема, HTML, CSS, JavaScript, React.JS, Node.js, Express.

Розглянуто інформацію, яка доступна користувачам Internet і розміщується на комп'ютерах (Web-серверах), де встановлено спеціальне програмне забезпечення. Значна частина цієї інформації організована як Web-сайти. Вимоги і переваги до Web-додатків. Оглянули методи підвищення роздільної здатності зображень і їх порівняли.

Наведено засоби розробки WEB-додатків.

Проведено аналіз предметної області автоматизації обробки зображень, виділення меж на зображенні у відтінках сірого, розглянули структурно-функціональної моделі автоматизації обробки зображень та виділення на них меж у відтінках сірого на основі SADT-технології. Описано контекстну структурно-функціональну діаграму нульового рівня процесу A0 "Виділення кордонів на зображенні".

2. АНАЛІЗ МОЖЛИВОСТЕЙ НЕЙРОМЕРЕЖ

2.1. Огляд архітектури нейронних мереж для розпізнавання зображень

Нейронні мережі – це математичні моделі, створені для обробки інформації, які є аналогом структури та функцій нервової системи живих організмів, що складаються з нейронів, взаємопов'язаних між собою та виконують функцію обробки даних. В даному випадку, основу представляють нейрони, що моделюють функціональні елементи нервової системи. Після прийняття вхідних сигналів нейрон обробляє їх і видає вихідний сигнал. При цьому кожен вхідний сигнал повинен бути відповідно помножений на ваговий коефіцієнт, що визначає важливість сигналу для вихідного. Після цього сума творів вхідних сигналів та їх вагових коефіцієнтів подається на функцію активації, що визначає вихідний сигнал нейрона [4; 5].

Функція активації схильна мати різні форми, такі як: сигмоїдальний або гіперболічний тангенс. Вони застосовуються для обмеження значення вихідного сигналу нейрона у певному діапазоні та зменшення впливу викидів у вхідних даних.

Нейрони об'єднуються в шари, а ті, у свою чергу, можуть бути як прихованими, так і вихідними. Приховані шари, що обробляють дані між вхідним і вихідним шарами, є ключовою особливістю нейронних мереж. Кількість прихованих шарів та кількість нейронів у кожному шарі залежать від конкретного завдання [16]. На рисунку 2.1 подано графічне зображення нейронної мережі.

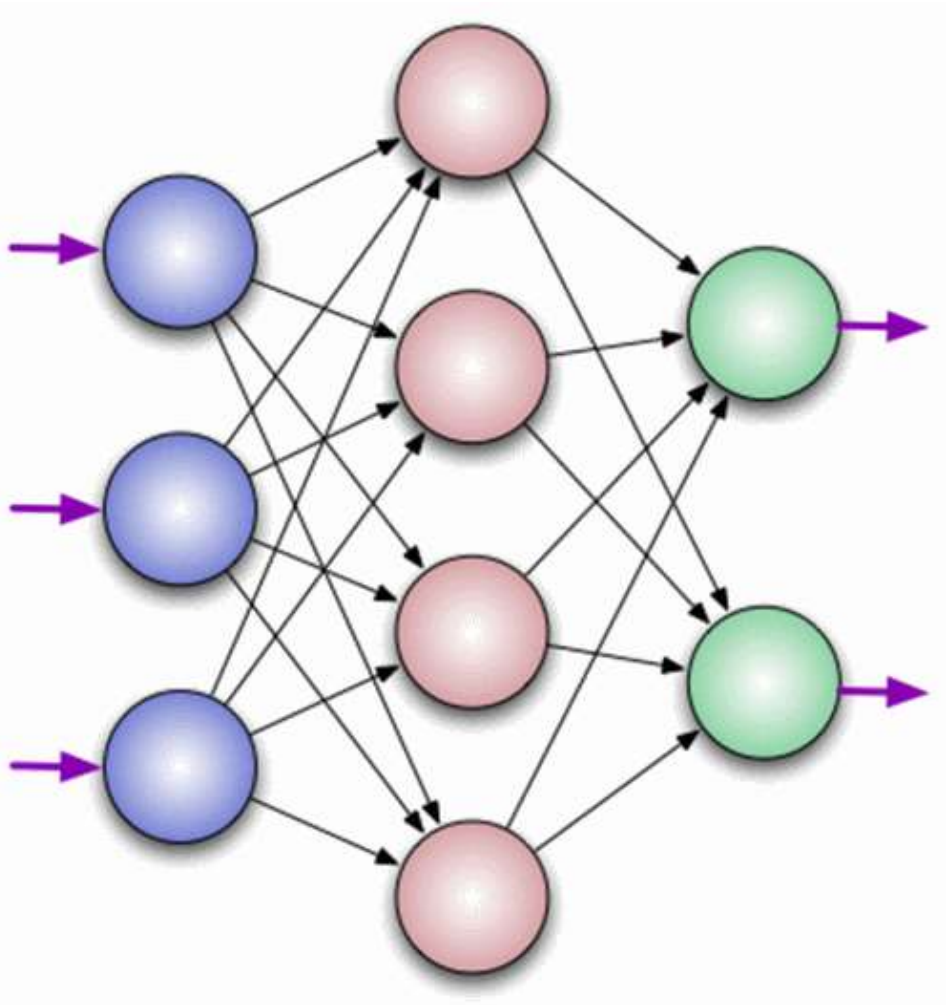


Рисунок 2.1. – Графічне зображення нейронної мережі

Навчання нейронної мережі відбувається завдяки настроюванню вагових коефіцієнтів, що відбувається на основі вхідних даних та очікуваних вихідних значень. Помилка вихідних даних порівнюється з очікуваними значеннями, застосовуючись для коригування вагових коефіцієнтів шляхом методу зворотного поширення помилки (рисунок 2.2).

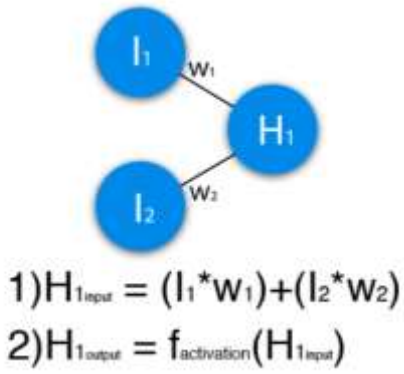


Рисунок 2.2 – Підрахунок вихідних даних на нейроні

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) використовуються для аналізу зображень. Складаються з декількох шарів, включаючи згорткові шари, що виконують фільтрацію зображення, та шари пулінгу, що зменшують розмірність даних. Ці шари дозволяють згорткової нейронної мережі автоматично отримувати ознаки зображення та класифікувати його.

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) широко застосовуються для аналізу послідовностей даних: мовних чи текстових. Вони мають пам'ять і здатні зберігати інформацію про попередні значення вхідних даних, завдяки чому можуть використовувати контекстну інформацію. Це робить їх особливо корисними для завдань, пов'язаних із природною мовою.

Глибокі нейронні мережі (Deep Neural Networks, DNN) - це нейронні мережі з декількома прихованими шарами, що мають велику обчислювальну потужність і здатність до вилучення ознак, що дозволяє їм ефективно вирішувати складні завдання, такі як розпізнавання мови або обробка зображень.

Функція активації.

Функція активації є одним з найважливіших елементів нейронної мережі, використовується для нелінійного перетворення вхідного сигналу нейрона. Вона здатна визначати вихідний сигнал нейрона на основі вхідних даних та параметрів ваг.

Залежно від типу нейронної мережі та її завдання функції активації можуть бути різними. Деякі функції активації представляють:

Лінійна функція: ця функція використовується вкрай рідко, за винятком окремих випадків, коли необхідно протестувати нейронну мережу або без перетворень передати значення (рисунок 2.3).

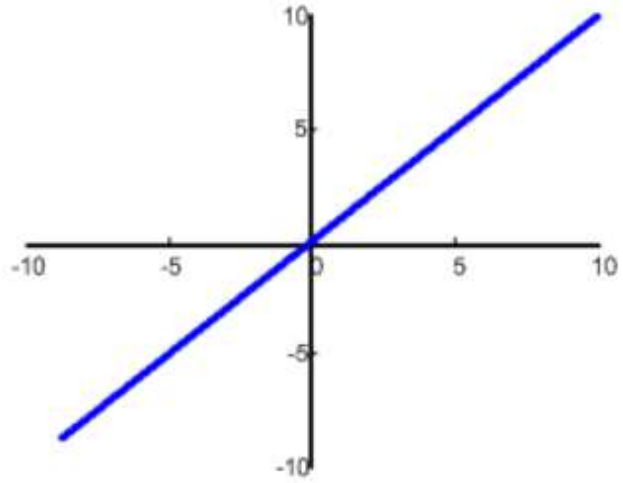


Рисунок 2.3 – Графік лінійної функції

Рівняння розраховують за формулою (2.1) [4].

$$f(x) = x \qquad (2.1)$$

Сігмоїдальна функція (Sigmoid function): ця функція перетворює будь-яке вхідне значення в діапазон між 0 і 1. На відміну від першої, досить часто використовується в роботі нейронних мереж для бінарної класифікації, розраховується за формулою (2.2) (рисунок 2.4) [4].

$$f(x) = \frac{1}{1+e^{-1}} \qquad (2.2)$$

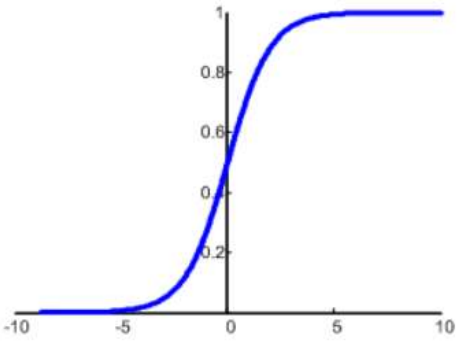


Рисунок 2.4 – Графік сигмоїдальної функції

Гіперболічний тангенс (Hyperbolic Tangent, Tanh): функція також перетворює вхідне значення в діапазон між -1 і 1. Ця функція є функцією

активації і часто використовується в нейронних рекурентних мережах. Рівняння функції виражається як формули (2.3) (рисунок 2.5).

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{2.3}$$

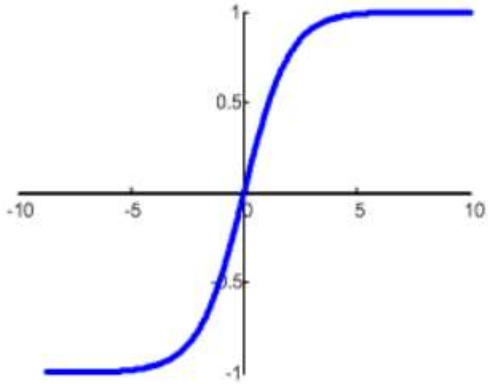


Рисунок 2.5 – Графік гіперболічного тангенсу

Функція ReLU (Rectified Linear Unit): повертає 0 всім негативних значень і повертає вхідне значення всім позитивних значень і розраховується за формулою (2.4). ReLU ефективна для глибоких нейронних мереж, оскільки сприяє їх швидкому навчанню та збіжності алгоритму зворотного поширення помилки.

$$f(x) = x^+ = \max(0, x) = \begin{cases} x & \text{if x} > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{2.4}$$

Функція Softmax: функція використовується для класифікації на кілька класів і перетворює вхідне значення на ймовірність класів.

Кожна функція має свої переваги та недоліки, а їх вибір залежить від конкретного завдання та архітектури мережі. Правильний вибір функції активації значно підвищує ефективність та працездатність нейронної мережі.

Вибір функції активації залежить від кількох факторів: тип завдання, дані, що використовуються для навчання, архітектура нейронної мережі та ін.

Наприклад, якщо завдання полягає в бінарній класифікації, то функція активації типу Sigmoid або Tanh може бути найбільш підходящою, оскільки вони перетворять вхідне значення в діапазон між 0 і 1 або між -1 і 1 відповідно,

що може використовуватися для передбачення ймовірності приналежності до класу.

Якщо ж йдеться про багатокласову класифікацію, то функція активації типу Softmax може бути більш підходящою, оскільки дозволяє перетворювати вхідні значення на ймовірність приналежності до кожного з класів.

Функція активації ReLU може бути корисною для глибоких нейронних мереж, оскільки сприяє більш швидкому навчанню та збіжності алгоритму зворотного розповсюдження помилки. Також, ця функція може бути кращою за інші функції для вирішення завдання виявлення об'єктів на зображеннях.

Крім цього, при виборі функції активації важливо враховувати наявність градієнтів, оскільки алгоритм зворотного розповсюдження помилки вимагає градієнтів знаходження оптимальних значень ваг в нейронної мережі.

Отже, вибір функції активації безпосередньо залежить від конкретної задачі та архітектури нейронної мережі, і потребує ретельного аналізу та експериментів.

Типи помилок у завданнях навчання нейронних мереж.

Помилка (Error) – це відхилення між передбаченим і дійсним значенням цільової змінної завдання навчання нейронної мережі. У машинному навчанні ми хочемо навчити модель, яка має можливість передбачати цільову змінну на основі даних, що надходять, і наше завдання полягає в мінімізуванні помилки.

Існують метрики, які дозволяють вимірювати помилки у завданнях машинного навчання. Такими є:

Середньоквадратична помилка (Mean Squared Error, MSE) – одна з найпоширеніших метрик для регресії. Вона обчислюється як середнє значення квадратів відхилень між передбаченими та істинними значеннями.

Середня абсолютна помилка (Mean Absolute Error, MAE) – це також метрика для регресії, але своєю чергою обчислюється як середнє значення абсолютних відхилень між передбаченими і дійсними значеннями.

Категоріальна крос-ентропія (Categorical Cross-Entropy, CCE) – метрика, яка використовується для класифікації з кількома класами. Вона хіба що

знаходить різницю між передбачуваними ймовірностями і справжніми ймовірностями класів.

Бінарна крос-ентропія (Binary Cross-Entropy, BCE) – дана метрика використовується для задач бінарної класифікації. Так само, як і попередня, вимірює різницю між передбачуваними ймовірностями та істинними ймовірностями.

Точність (Accuracy) – це метрика, яка вимірює відсоток правильних передбачень[10].

Для обчислення помилки необхідне порівняння передбачених значень із істинними значеннями цільової змінної, які ми знаємо з навчального набору даних. Після цього ми використовуємо вибрану метрику, щоб обчислити помилку. Наша мета в даний момент мінімізувати помилку на навчальному наборі даних.

2.2. Типи нейронних мереж, що використовуються для розпізнавання зображень

Розпізнавання зображень – найпоширеніша задача в галузі машинного навчання, для її вирішення використовуються різні типи нейронних мереж. Приклад деяких:

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) - це популярний тип нейронних мереж, що використовується для обробки зображень. Є набором шарів, кожен з яких перетворює вхідні дані в деякий вихід. Шари згорткової нейронної мережі включають: згорткові шари, шари підвибірки (pooling), шари активації і повнозв'язкові шари (рисунок 2.6). Згорткові шари використовуються для виявлення ознак у зображеннях, а шари вибірки для зменшення розмірності даних. Шари активації допомагають моделі вивчити нелінійні залежності даних, а пов'язані шари для класифікації зображень [2; 4].

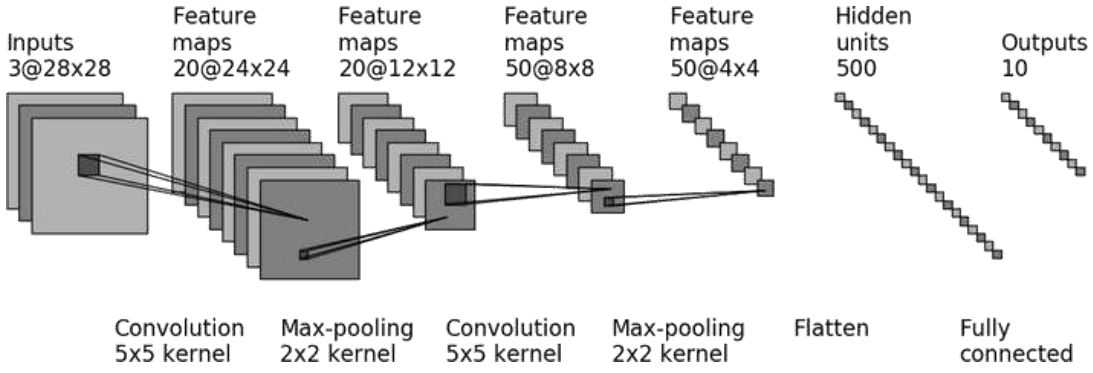


Рисунок 2.6 – Схема згорткової нейронної мережі

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) – це тип нейронних мереж, що широко використовується для обробки послідовностей: тексти та мовлення. Рекурентні нейронні мережі складаються з набору блоків, що повторюються, кожен з яких має вхід, вихід і прихований стан (рисунок 2.7). Рекурентні нейронні мережі використовуються для обробки зображень, якщо зображення подано у вигляді послідовності.

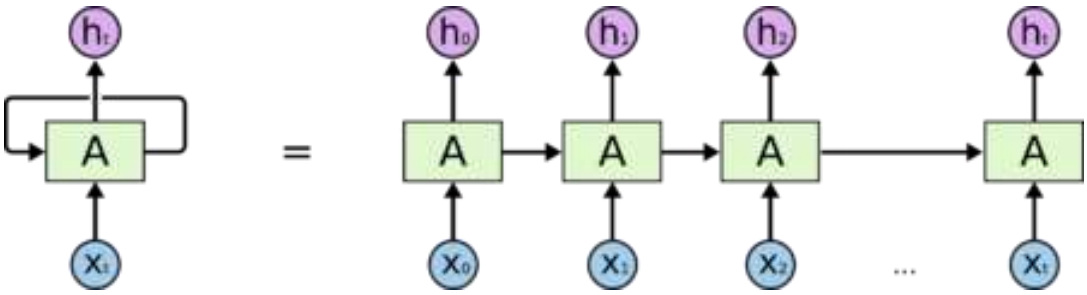


Рисунок 2.7 – Схема рекурентної нейронної мережі

Глибокі нейронні мережі (Deep Neural Networks, DNN) - це тип нейронних мереж, що має безліч прихованих шарів. Чим більше шарів, тим глибше та складніше стає мережа. Глибокі нейронні мережі використовуються для вирішення задач класифікації та регресії, а також для обробки зображень.

Згортково-рекурентні нейронні мережі (Convolutional Recurrent Neural Networks, CRNN) – це комбінація згорткових та рекурентних нейронних мереж. Широко використовується для обробки зображень, де кожен піксель сприймається як послідовність.

Автоенкодери (Autoencoders) – це нейронні мережі, що використовуються для вивчення прихованого представлення даних. Автоенкодери можуть використовуватися для обробки зображень шляхом стиснення їх до більш

компактного подання. Складається з двох частин: енкодер, який перетворює вхідні дані в приховане уявлення, і декодера, який перетворює приховане уявлення назад у вихідні дані. Таким чином, автоенкодери можуть використовуватися для вивчення прихованих ознак зображень та для генерації нових зображень [10; 11].

Генеративно-змагальні мережі (Generative Adversarial Networks, GAN) – це нейронні мережі, які використовуються для генерації нових зображень. Вони складаються з двох мереж: генеративної та дискримінативної. Генеративна мережа генерує нові зображення, а дискримінативна мережа намагається відрізнити ці зображення реальних. Завдання генеративної мережі – створювати зображення, які дискримінативна мережа зможе відрізнити від реальних. GAN можуть використовуватися для створення нових зображень, а також для поліпшення якості зображень.

Нейронні мережі з увагою (Attention Neural Networks) це тип нейронних мереж, який використовується для обробки послідовностей та обробки зображень. Вони дозволяють моделювати важливість кожної частини вхідних даних під час прийняття рішень. Таким чином, вони дозволяють моделям зосередитися на найважливіших частинах даних при вирішенні завдань [3].

Вибір конкретної архітектури нейронної мережі залежить від ряду факторів, включаючи:

Тип завдання: різні типи завдань потребують різних архітектур нейронних мереж. Наприклад, для розпізнавання об'єктів на зображеннях зазвичай використовують нейронні мережі згортки, в той час як для обробки послідовностей, таких як тексти або мова, зазвичай використовують рекурентні нейронні мережі.

Кількість даних: деякі архітектури вимагають великої кількості даних для навчання, в той час як інші можуть навчатися меншому обсязі даних. Наприклад, глибокі нейронні мережі вимагають великої кількості даних для навчання, тоді як нейронні мережі з увагою можуть працювати з меншим обсягом даних.

Обчислювальна потужність: деякі архітектури вимагають великої кількості обчислювальної потужності для навчання та роботи, тоді як інші можуть працювати на слабкіших пристроях. Наприклад, згорткові нейронні мережі вимагають великої кількості обчислювальної потужності для навчання, тоді як нейронні мережі з увагою можуть працювати на слабкіших пристроях.

Складність моделі: більш складні архітектури можуть мати більш високу точність, але потребують більшого часу та ресурсів для навчання та роботи. Наприклад, глибокі нейронні мережі можуть досягати високої точності, але потребують великої кількості часу та ресурсів для навчання.

Наявність передвиборних моделей: деякі архітектури мають передбачені моделі, які можуть бути використані для вирішення подібних завдань. Це може прискорити процес навчання та покращити точність моделі.

В цілому, вибір архітектури нейронної мережі залежить від конкретного завдання, доступних ресурсів та вимог до точності та швидкості роботи моделі.

Функція втрат.

Функція втрат (Loss function) – це математична функція, що вимірює якість передбачень цільової змінної моделі на тренувальних даних. Являє собою числову міру розбіжності між передбаченою моделлю та реальними значеннями цільової змінної.

Функція втрат є основною складовою навчання моделі. Мета – мінімізувати помилку на навчальних даних та створити модель, яка робитиме точні передбачення на нових даних. У процесі навчання оптимізатор намагається зменшити значення функції втрат, змінюючи ваги моделі.

У машинному навчанні безліч різних функцій втрат, і вибір залежить від: конкретної задачі, типу моделі та виду даних.

Найбільш поширені з них:

Mean Squared Error (MSE) – середньоквадратична помилка, яка використовується для завдань регресії в тих випадках, коли цільова змінна є безперервною відображеною у формулі (2.5).

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y - y_{hat})^2 \quad (2.5)$$

де n – кількість спостережень;

y – реальне значення цільової змінної;

y_{hat} – передбачене значення цільової змінної.

Binary Cross Entropy Loss – функція втрат для бінарної класифікації, яка використовується у випадках, коли цільова змінна набуває лише двох значень. Рівняння функції представлено у формулі (2.6).

$$\text{BCE} = -\frac{1}{n}\sum_{i=1}^{n} y_i \log\big(p(y_i)\big) + (1 - y_i)\log\big(1 - p(y_i)\big) \quad (2.6)$$

де y – реальне значення цільової змінної;

$p(y_i)$ – передбачене значення цільової змінної.

Categorical Cross Entropy Loss – функція втрат для багатокласової класифікації, яка використовується, коли цільова змінна є категоріальною та виражається формулою (2.7).

$$CCE = -\sum_{i=1}^{n} y_i \log(y_{hat_i}) \quad (2.7)$$

де y_i – реальне значення цільової змінної для і-го класу;

$y_{hat\ i}$ – передбачене значення цільової змінної для і-го класу.

Hinge Loss – функція втрат, яка використовується для класифікації, особливо для опорних векторних машин (SVM). Hinge Loss можна визначити за формулою (2.8).

$$Hinge\ Loss = \max(0, 1 - y \times y_{hat}) \quad (2.8)$$

де y – реальне значення цільової змінної (-1 або 1);

y_{hat} – передбачене значення цільової змінної.

Huber Loss – функція втрат, що використовується для регресії, яка менш чутлива до викидів, ніж MSE. Ця функція розраховується за формулою (2.9).

$$Huber\ Loss = \frac{1}{2}(y - y_{hat})^2, \text{если } |y - y_{hat}| \le delta,$$
$$\text{и } delta\left(|y - y_{hat}| - \frac{delta}{2}\right), \text{если } |y - y_{hat}| > delta \quad (2.9)$$

де *delta* – параметр, що визначає поріг, на якому перемикається функція втрат.

Вибір функції втрат важливий у створенні нейронної моделі. Оптимальну функцію втрат можна підібрати дослідним шляхом, експериментуючи з різними варіантами та вибираючи ту, яка дає найкращі результати на тестових даних.

Перенавчання.

Перенавчання (overfitting) - це явище, коли нейронна мережа добре навчається на тренувальних даних, але показує погану продуктивність на нових, раніше не бачених даних. У цьому випадку мережа "запам'ятовує" тренувальні дані, замість того, щоб вивчати загальні закономірності даних, і не здатна узагальнювати нові дані.

Уникнути перенавчання можна за допомогою таких методів:

Регуляризація: додавання штрафних елементів до функції втрат, які зменшують ваги мережі, щоб зменшити їх абсолютні значення. Наприклад, L1 та L2 регуляризація.

Dropout: випадкове відключення нейронів на кожній ітерації навчання. Це запобігає "залежності" нейронів один від одного і допомагає мережі уникнути перенавчання.

Зменшення розміру мережі: зменшення кількості шарів або нейронів може допомогти запобігти перенавчанню.

Рання зупинка (early stopping): зупинка процесу навчання, коли продуктивність на валідаційному наборі даних перестає покращуватись.

Збільшення розміру навчальної вибірки: більш різноманітні дані можуть допомогти мережі узагальнювати нові дані та уникнути перенавчання.

Data augmentation: модифікація існуючих навчальних даних для створення нових, різноманітних даних.

Transfer learning: використання попередньо вивчених моделей, які навчалися на схожих завданнях, та донавчання їх на нових даних.

Вибір конкретного методу залежить від конкретного завдання та даних. Зазвичай комбінація кількох методів може дати найкращий результат.

2.3. Методи навчання нейронних мереж.

Методи навчання нейронних мереж використовуються для покращення точності та ефективності розпізнавання зображень. Існує кілька методів навчання:[11-13]

Навчання з учителем (supervised learning) – це метод навчання нейронної мережі, при якому використовується набір навчальних даних, що складається з вхідних даних та відповідних їм правильних відповідей. Ці правильні відповіді називаються мітками (labels).

У процесі навчання нейронна мережа використовує ці позначки для коригування своїх ваг та біосів, щоб мінімізувати функцію втрат (loss function) та покращити точність передбачень на нових даних.

У разі розпізнавання зображень, вхідні дані є зображеннями, а мітки – відповідні класи, яким належать ці зображення. Наприклад, якщо йдеться про навчання нейронної мережі для класифікації зображень кішок та собак, то кожному зображенню кішки відповідатиме мітка "кішка", а кожному зображенню собаки – мітка "собака".

Одним із найбільш популярних алгоритмів навчання з учителем для розпізнавання зображень є зворотне поширення помилки (backpropagation). Він полягає в перерахунку ваг і біосів нейронної мережі на основі помилок, допущених у процесі передбачення міток для кожного зображення в навчальному наборі [14; 15].

Навчання з учителем вимагає великої кількості розмічених даних для досягнення високої точності передбачень нових даних.

Навчання без вчителя (unsupervised learning) – це метод машинного навчання, у якому алгоритми намагаються витягти приховані структури чи

закономірності з нерозмічених даних. На відміну від навчання з учителем, в якому модель навчається на основі розмічених даних з явними мітками класів, у навчанні без вчителя модель не має явних міток класів і повинна самостійно виявляти приховані структури та закономірності даних.

Основним методом навчання без вчителя є кластеризація, що дозволяє групувати об'єкти з урахуванням їх подібності. Іншими методами навчання без вчителя є зниження розмірності, пошук аномалій, генеративні моделі та ін.

Приклади алгоритмів навчання без вчителя в задачах розпізнавання зображень включають метод головних компонент (PCA) для зниження розмірності даних, метод k-середніх для кластеризації зображень і алгоритми генеративних моделей, такі як автокодувальники (autoencoders) і генеративні змагальні мережі (GANs) для генерації зображень [16].

Навчання без вчителя може бути корисним для завдань, у яких розмічених даних недостатньо або їх отримання є трудомістким та дорогим процесом. Однак, оскільки моделі в навчанні без вчителя не отримують явних позначок класів, оцінка їх точності та якості може бути складнішим завданням, ніж у навчанні з вчителем.

Навчання з підкріпленням (reinforcement learning) – це метод навчання машинного навчання, при якому агент приймає дії в певному середовищі, та отримує позитивну чи негативну нагороду залежно від результату своїх дій. Мета агента – максимізувати нагороду, і він може використати свій досвід для покращення своїх дій у майбутньому.

Навчання з підкріпленням застосовується у різних галузях, включаючи ігри, робототехніку, управління процесами тощо. Для навчання з підкріпленням використовується модель марковського процесу прийняття рішень (Markov decision process, MDP), яка визначає правила, якими агент вибирає дії в кожному стані.

Одним із основних компонентів навчання з підкріпленням є функція оцінки цінності (value function), яка оцінює, наскільки хороші дії агента у кожному стані. Ця функція може бути обчислена за допомогою алгоритмів,

таких як Q-навчання або SARSA. Крім того, для вибору дій агентом може бути використана стратегія, яка визначає, як вибираються дії в кожному стані.

Навчання з підкріпленням може бути складним процесом, оскільки агент може зіткнутися з проблемою дослідження-використання (exploration-exploitation dilemma), коли він повинен обирати між вивченням нових дій та використанням відомих дій для максимізації нагороди. Для вирішення цієї проблеми можуть використовуватися різні методи, такі як жадібна стратегія, епсилон-жадібна стратегія, стратегія м'якої максимізації і т.д.

Попередня обробка зображень, що використовуються для покращення якості та точності розпізнавання зображень.

Попередня обробка зображень – це важливий крок при роботі із зображеннями, який може суттєво покращити якість розпізнавання об'єктів на зображеннях. Нижче наведено огляд методів попередньої обробки зображень.

Фільтрування зображень: Один з найбільш поширених методів попередньої обробки зображень, який дозволяє прибрати шуми на зображеннях та покращити їхню якість. Для цього використовуються різні фільтри, такі як фільтр Гауса, медіанний фільтр та ін.[4, 5]

Збільшення роздільної здатності зображень: Цей метод попередньої обробки зображень дозволяє покращити якість зображень, збільшуючи роздільну здатність зображення без втрати якості. Для цього використовуються різні методи, такі як інтерполяція зображень та Super-Resolution.

Коригування кольору: Метод попередньої обробки, який дозволяє коригувати колір зображень. Корекція кольору може бути використана для узгодження кольору між декількома зображеннями, поліпшення контрасту і прибирання ефекту зміщення кольорів.

Конвертація кольорових зображень у відтінки сірого: Метод попередньої обробки, який може бути використаний для спрощення обробки зображень та зменшення розміру даних для навчання нейронних мереж [4, 6].

Нормалізація зображень: Метод попередньої обробки, який дозволяє нормалізувати яскравість та контрастність зображень, що може покращити якість та точність розпізнавання об'єктів на зображеннях.

Обрізка зображень: Метод попередньої обробки, який може бути використаний для видалення непотрібних елементів на зображеннях та покращення якості розпізнавання об'єктів на зображеннях.

Нормалізація даних: Метод попередньої обробки, який може бути використаний для нормалізації даних, наприклад, нормалізація даних навчання, щоб збалансувати внесок кожної навчальної вибірки у нейронну мережу.

Оцінка та порівняння продуктивності нейронних мереж.

Оцінка та порівняння продуктивності нейронних мереж є важливим завданням у галузі машинного навчання. Зокрема, для розпізнавання зображень, таких як класифікація або сегментація, необхідно оцінити точність, швидкість і ефективність різних архітектур нейронних мереж.

Метрики точності є одним із найпоширеніших методів оцінки продуктивності нейронних мереж для завдань розпізнавання зображень. Вони вимірюють, як добре модель класифікує зображення на правильні категорії.

Однією з основних метрик точності є accuracy (точність), яка визначає частку правильно класифікованих об'єктів у наборі даних. Формула для обчислення accuracy:

accuracy = (кількість правильних передбачень) / (загальна кількість передбачень)

Іншою популярною метрикою є F1 score, яка враховує не тільки кількість правильних передбачень, але і кількість хибно-позитивних та хибно-негативних результатів. F1 score можна обчислити за формулою (2.10):

$$F1\ score\ =\ 2\ \frac{(precision \times\ recall)}{(precision\ +\ recall)} \qquad (2.10)$$

де *precision* (точність) – це частка правильно передбачених позитивних результатів щодо всіх позитивних результатів;

recall (повнота) – це частка правильно передбачених позитивних результатів щодо всіх дійсних позитивних результатів.

Швидкість є також важливою метрикою продуктивності нейронних мереж. Швидкість роботи нейронної мережі залежить від розміру вхідних даних, архітектури мережі та обладнання, що використовується. Існують різні методи прискорення роботи нейронних мереж, такі як паралелізація обчислень та використання апаратного прискорення (наприклад, графічних процесорів).

Ефективність є метрикою, яка оцінює співвідношення між точністю та швидкістю роботи нейронної мережі. Найчастіше використовуваною метрикою ефективності є FLOPS (floating point operations per second), яка вимірює кількість операцій з плаваючою комою, які може виконати нейронна мережа в секунду.

Деякі з методів порівняння архітектур нейронних мереж включають порівняння кількості параметрів і складності моделі. Кількість параметрів - це кількість ваг і зсувів, що настроюються, які потрібно оптимізувати в процесі навчання моделі. Простіші моделі мають меншу кількість параметрів і зазвичай швидше навчаються, але можуть мати низьку точність. Складність моделі – це кількість операцій, які потрібні для виконання обчислень у нейронній мережі. Більш складні моделі можуть мати більш високу точність, але можуть бути більш повільними та вимогливими до обчислювальних ресурсів.

Інший метод порівняння архітектур нейронних мереж – це порівняння результатів на стандартних наборах даних, таких як CIFAR-10, CIFAR-100, MNIST та ImageNet. Ці набори широко використовуються для порівняння продуктивності різних архітектур нейронних мереж. Деякі змагання, такі як ImageNet Large Scale Visual Recognition Challenge (ILSVRC), використовуються для тестування різних архітектур та оцінки їхньої продуктивності.

Застосування нейронних мереж для розпізнавання зображень у реальних додатках.

Нейронні мережі широко застосовуються для розпізнавання зображень у реальних додатках, оскільки вони можуть досягти високої точності та

забезпечувати автоматизацію завдань, які раніше вимагали великої кількості ручної роботи. Розглянемо деякі з найпоширеніших програм розпізнавання зображень, що використовують нейронні мережі:[2-5].

1) Розпізнавання об'єктів – це одне з найпоширеніших завдань у галузі комп'ютерного зору, яка полягає у визначенні, які об'єкти знаходяться на зображенні та де вони знаходяться. Прикладом може бути система автоматичного розпізнавання облич на фотографіях чи відеопотоці.

2) Автоматичне розпізнавання облич - це додаток розпізнавання зображень, який використовує нейронні мережі для ідентифікації осіб на фотографіях або відеопотоці. Такі системи використовуються для різних цілей від безпеки до полегшення процесу ідентифікації на фотографіях у соціальних мережах.

3) Розпізнавання жестів - це додаток розпізнавання зображень, який використовується для розпізнавання жестів, зроблених руками, та їх перетворення на відповідні команди. Ця програма може використовуватися в різних областях, наприклад, в контролерах ігор, системах управління презентаціями і т.д.

4) Розпізнавання патернів – це програма, яка використовує нейронні мережі для розпізнавання патернів на зображеннях, таких як текстури, шаблони та інше. Воно може використовуватись у різних галузях, включаючи медицину, промисловість і т.д.

5) Оптичне розпізнавання символів (OCR) – це програма розпізнавання зображень, яка використовується для розпізнавання тексту на зображенні та його перетворення на текстовий формат. Воно може використовуватись для автоматичного розпізнавання номерів паспортів, банківських карток тощо.

6) Розпізнавання медичних зображень – це програма, яка використовує нейронні мережі для розпізнавання різних медичних зображень, таких як рентгрaфії, МРТ, УЗД тощо.

Воно може використовуватися для автоматичної діагностики та класифікації різноманітних захворювань на основі медичних зображень.

7) Розпізнавання емоцій – це програма, яка використовує нейронні мережі для розпізнавання емоцій на обличчях людей на фотографіях або відеопотоці. Воно може використовуватися для різних цілей, включаючи оцінку емоційної реакції реклами, вивчення поведінки покупців у магазинах тощо.

В цілому, нейронні мережі для розпізнавання зображень мають широкий спектр застосувань у різних галузях.

Бібліотеки відкритого вихідного коду для машинного навчання та глибокого навчання мовою програмування Python.

З розвитком машинного навчання та штучного інтелекту, фреймворки для глибокого навчання, такі як TensorFlow та Keras, стали широко використовуваними інструментами для створення та навчання нейронних мереж. TensorFlow та Keras надають високорівневі API для створення, навчання та застосування нейронних мереж з мінімальними витратами на програмування. У цьому розділі ми розглянемо основні функції TensorFlow та Keras та їх застосування у машинному навчанні.

TensorFlow TensorFlow – це відкритий програмний фреймворк для машинного навчання, розроблений Google. Він був створений для обробки великих обсягів даних та побудови масштабованих обчислювальних моделей глибокого навчання. Основним об'єктом у TensorFlow є граф обчислень, що складається з вузлів (операцій) та ребер (тензорів), де кожен вузол представляє математичну операцію, а кожне ребро – потік даних.

Основні функції TensorFlow:

1) Створення графа обчислень: TensorFlow надає API для створення графа обчислень, який дозволяє визначити математичні операції та їх залежність один від одного. Цей граф може бути оптимізований та розподілений між декількома пристроями для прискорення навчання та застосування моделей.

2) Автоматичне диференціювання: TensorFlow дозволяє автоматично обчислювати похідні функції, що уможливлює використання алгоритмів градієнтного спуску для навчання нейронних мереж.

3) Робота з різними типами даних: TensorFlow підтримує роботу з різними типами даних, такими як числа з плаваючою точкою, цілі числа, рядки і т.д.

4) Розподілене навчання: TensorFlow дозволяє розподіляти навчання між кількома пристроями, такими як CPU та GPU, а також між декількома вузлами в мережі.

5) Візуалізація графа обчислень: TensorFlow надає інструменти для візуалізації графа обчислень, що дозволяє легко відстежувати та налагоджувати моделі.

Keras Keras – це високорівневий API для машинного навчання, розроблений з урахуванням простоти та швидкості створення та навчання нейронних мереж. Keras працює поверх TensorFlow, що дозволяє використовувати всі переваги TensorFlow разом із зручністю роботи у Keras. Він підтримує велику кількість типів нейронних мереж, таких як згорткові нейронні мережі, рекурентні нейронні мережі, автоенкодери і т.д.[2-5]

Основні функції Keras:

1) Простота використання: Keras надає зручний інтерфейс для створення та навчання нейронних мереж з мінімальною кількістю коду.

2) Модульність: Keras дозволяє легко створювати складні моделі, що складаються з кількох шарів, кожен із яких виконує свою специфічну функцію.

3) Підтримка безлічі типів нейронних мереж Keras підтримує безліч типів нейронних мереж, що дозволяє вибрати найкращий тип для конкретної задачі.

4) Автоматичне диференціювання: Keras використовує TensorFlow для автоматичного обчислення похідних функцій, що дозволяє використовувати алгоритми градієнтного спуску для навчання нейронних мереж.

5) Робота з різними типами даних: Keras підтримує роботу з різними типами даних, такими як числа з плаваючою точкою, цілі числа, рядки і т.д.

6) Навчання на GPU: Keras дозволяє навчати нейронні мережі на GPU, що прискорює процес навчання.

7) Застосування TensorFlow та Keras у машинному навчанні: TensorFlow та Keras використовуються для вирішення безлічі завдань машинного навчання, таких як:

8) Класифікація зображень: TensorFlow та Keras широко використовуються для класифікації зображень, таких як фотографії, рентгенівські знімки тощо. Нейронні мережі, які навчаються на великій кількості зображень, можуть точно визначати об'єкти на зображенні та класифікувати їх.

9) Розпізнавання мови: TensorFlow та Keras можуть використовуватися для навчання нейронних мереж, які розпізнають мову та переводять її в текст. Це може бути корисним, наприклад, для створення голосових помічників.

10) Рекомендаційні системи: TensorFlow та Keras можуть використовуватися для навчання нейронних мереж, які рекомендують користувачеві товари або послуги на основі його переваг та поведінки в Інтернеті.

11) Обробка природної мови: TensorFlow та Keras можуть використовуватися для навчання нейронних мереж, які обробляють текст та виконують завдання, такі як класифікація тексту, машинний переклад тощо.

12) Аналіз часових рядів: TensorFlow та Keras можуть використовуватися для навчання нейронних мереж, які аналізують часові ряди, такі як дані про температуру, фінансові дані тощо [2].

Крім того, TensorFlow і Keras використовуються в багатьох реальних програмах, таких як автономні автомобілі, розпізнавання рукописного тексту, голосові помічники і т.д. Вони є ключовими інструментами для вирішення складних завдань машинного навчання і продовжують розвиватися та покращуватися з кожним роком.

3. ОПИС РОЗРОБЛЕНОГО WEB-ЗАСТОСУНКУ

3.1. Структура HTML-документа

HTML-документ у моєму проєкті є каркасом веб-додатку, що відповідає за структуру сторінки, її вміст і підключення зовнішніх ресурсів (стилів та скриптів). Розглянемо цей файл поетапно, зверху донизу.

Початок документа

```
<!DOCTYPE html>
<html lang="en">
```

Це стандартне оголошення типу документа (Document Type Declaration). Воно говорить браузеру, що ми використовуємо **HTML5** — актуальну версію HTML. Атрибут lang="en" вказує, що мова сторінки — англійська. Це важливо як для SEO (оптимізації у пошукових системах), так і для скрінрідерів та іншого асистивного софту.

Блок <head>

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

Ці три теги відповідають за базову конфігурацію сторінки:

- **<meta charset="UTF-8">** — задає кодування UTF-8, що дозволяє відображати будь-які символи (латинку, кирилицю, емодзі тощо).
- **<meta http-equiv="X-UA-Compatible" content="IE=edge">** — для сумісності зі старими версіями Internet Explorer.
- **<meta name="viewport" content="width=device-width, initial-scale=1.0">** — робить сайт адаптивним для мобільних пристроїв, масштабуючи інтерфейс під ширину екрана.

Підключення стилів та скриптів

```
<link rel="stylesheet" href="style.css">
<link href='https://fonts.googleapis.com/css?family=Inter' rel='stylesheet'>
```

style.css — головний файл стилів. Саме він відповідає за кольори, розмітку, шрифти, ефекти.

- **Google Fonts —** підключення шрифту Inter, який використовується у всьому інтерфейсі. Стильний, сучасний, добре читається**.**

```
    <script src="https://code.jquery.com/jquery-3.6.0.js" ...></script>
<script src="script.js" defer></script>
<script src="Croppie-2.6.4/croppie.js" defer></script>
<script src="popup/popup.js"></script>
<script src="https://unpkg.com/axios/dist/axios.min.js" defer></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/cropperjs/1.5.12/cropper.min.js" ...></script>
```

Підключені скрипти:

- **jQuery** — бібліотека для спрощення роботи з DOM (документом) і обробки подій.
- **script.js** — головна логіка взаємодії користувача із сайтом.
- **Croppie.js та Cropper.js** — бібліотеки для обрізки/редагування зображень. Хоча зараз використовується тільки Cropper.
- **popup.js** — можливо, скрипт для модального вікна (деталі в розділі про JS).
- **axios** — бібліотека для надсилання HTTP-запитів на сервер. Зручно працює з API.

Також підключаються стилі для cropper.css, croppie.css та popup.css. Всі вони використовуються для оформлення специфічних компонентів (редактор зображень, попапи).

Тіло сторінки — <body>

Далі починається основна частина сайту — контент, який бачить користувач.

Блок: Фіксована шапка сайту

```
    <div id="static-header">
  <div id="logo" style="width: 100px;">
    <img src="sAI.png" style="width: 100%;">
  </div>
  <menu id="header-buttons">
    <li><button class="menu-button">Gallery</button></li>
    <li><button class="menu-button">Contact</button></li>
    <li><button class="menu-button">About</button></li>
  </menu>
</div>
```

#static-header — верхній бар, який завжди видно (позиція фіксована). Тут розміщено:

- Логотип (sAI.png)

- Кнопки навігації: "Gallery", "Contact", "About". Вони не ведуть поки що нікуди, але можуть слугувати якірними посиланнями або відкривати попапи.

- **<menu>** — тег використовується як контейнер для навігаційних елементів (альтернатива <ul>). Не обов'язково, але зручно для семантики.

Блок: Головна секція з промо

```
<div class="faded-flex">
 ...
</div>
```

Цей блок ділиться на **три умовні частини**. Використовується display: flex, що дозволяє зручно розмістити елементи в ряд.

Центральна частина — завантаження зображення

```
<div class="half-div-items-centre image-upload">
  <div class="white-square">
    <header>Drag & Drop to Upload File</header>
    <span>OR</span>
    <button class="menu-button">Browse</button>
    <input type="file" hidden />
  </div>
</div>
```

Основна взаємодія: перетягування файлу, або натисканняя "Browse".

- input[type="file"] прихований, але активується при натисканні на кнопку.

Права частина — показ завантаженого зображення

```
<div class="image-uploaded">
  <div class="white-square">
    <div class="img-div">
      <img src="" id="imageId" class="edit-img">
    </div>
    <div class="button-list">
      <button class="menu-button" onclick="uploadBack()">Back</button>
      <button class="menu-button" onclick="saveUpload()">Proceed</button>
      <button class="menu-button" onclick="editUpload()">Edit</button>
    </div>
  </div>
</div>
```

Це альтернатива центральному блоку: коли користувач вибрав зображення, цей блок підміняє попередній.

- Зображення показується, додається cropper.

- Кнопки керують завантаженням/редагуванням.

Блок: Порівняння "до/після"

```
<div class="before-after-image">
 ...
</div>
```

Поділений на дві частини:

1. **Ліва — текст і кнопки** ("Upscale image...", "Pricing", "Try for free").
2. **Права — інтерактивний блок з повзунком**:

```
<div class="container">
  <div class="lables">
    <div class="image-lable" id="left"><p>After</p></div>
    <div class="image-lable" id="right"><p>Before</p></div>
  </div>
  <div class="background-image img"></div>
  <div class="foreground-image img"></div>
  <input type="range" ... id="slider-id">
</div>
```

- Накладені два зображення (before.jpeg і after.jpg).
- Слайдер регулює ширину foreground-image, створюючи ефект "розрізу" зображення — дуже популярна штука в дизайні, особливо у фоторедакторах.

Блок: Попап для редагування

```
<div class="popup-bg" onclick="closePopup()"></div>
<div class="popup">
  <div style="width: 300px;">
    <img src="" class="edit-img">
  </div>
</div>
```

- popup-bg — затемнення фону.
- popup — саме модальне вікно.

Топчик, тоді погнали далі — тепер **CSS**, теж максимально детально, по кожному класу, стилю і його впливу.

3.2. Стилізація інтерфейсу користувача (CSS)

У цьому розділі описано файл style.css, який відповідає за зовнішній вигляд веб-додатку. Він визначає кольори, шрифти, розміри, розташування елементів, адаптивність і взаємодію з користувачем. CSS написано вручну без використання препроцесорів, що забезпечує простоту та зрозумілість.

Базова ініціалізація та шрифт

```
body, html {
  height: 100%;
  width: 100%;
  margin: 0;
  padding: 0;
  font-family: 'Inter';
  font-size: 24px;
}
```

Скидаємо всі **відступи та поля** браузера (margin/padding).

- Задаємо 100% висоту і ширину, щоб усе займало весь простір.
- Шрифт — 'Inter' із Google Fonts. Це сучасний, добре читабельний sans-serif.
- Базовий розмір шрифту — 24px. Відносно великий, зручний для читання.

```
body {
  background-color: #001d43;
}
```

Глибокий синій фон — надає професійний вигляд і гарно контрастує з білими блоками.

Шапка сайту (#static-header)

```
#static-header {
  display: flex;
  position: fixed;
  top: 0;
  width: 100%;
  background-color: #001229;
  z-index: 2;
  box-shadow: rgba(0, 0, 0, 0.45) 0px 3px 6px;
}
```

- display: flex — елементи всередині (лого + кнопки) виводяться в ряд.
- position: fixed — шапка завжди закріплена зверху.

- z-index: 2 — щоб була поверх інших блоків.
- box-shadow — додає тінь, візуально відокремлює від основного контенту.

Логотип і кнопки в шапці

```
#header-buttons {
  display: flex;
  justify-content: space-evenly;
  width: 50%;
  list-style: none;
  margin: 15px;
  align-items: flex-end;
  margin-right: 10px;
  margin-left: auto;
}
```

Кнопки у шапці розташовуються рівномірно.

- Прибрано маркери списку (list-style: none).
- margin-left: auto — пушить елементи вправо, щоб вони не зливались з логотипом.

```
#logo {
  display: flex;
  align-items: flex-start;
  margin: auto;
  margin-left: 70px;
}
#logo img {
  width: 60px;
  border-radius: 10px;
}
```

Логотип має закруглені кути і відступ зліва, щоб не прилипав до краю.

Кнопки (.menu-button)

```
.menu-button {
  font-family: 'Inter';
  font-size: 20px;
  padding: 10px;
  width: 150px;
  border: none;
  border-radius: 5px;
  background-color: rgb(38, 1, 187);
  color: #fff;
  box-shadow: rgba(0, 0, 0, 0.45) 0px 3px 20px;
  cursor: pointer;
}
```

- Універсальний стиль для всіх кнопок.

- Насичений синій фон, білий текст, тінь — кнопки виглядають масивно і привабливо.
- cursor: pointer — змінює курсор при наведенні, що сигналізує про інтерфейсну дію.

Головна секція з промо (.faded-flex)

```
.faded-flex {
  display: flex;
  width: 90%;
  margin: auto;
  margin-top: 150px;
  height: 600px;
  border-radius: 10px;
  background: linear-gradient(to right, rgb(38, 1, 187), rgb(69, 156, 255));
  box-shadow: rgba(0, 0, 0, 0.45) 0px 6px 20px;
}
```

Ефектний блок: градієнт із фіолетово-синього у світло-блакитний.

- Тінь та закруглені кути додають глибини.
- display: flex — елементи всередині йдуть в ряд.

```
.half-div-items-left,
.half-div-items-centre {
  width: 50%;
  height: 100%;
  display: flex;
  flex-direction: column;
  justify-content: center;
}
```

Дві колонки: зліва текст, справа інтерфейс завантаження.

- justify-content: center — вирівнювання по вертикалі.

Текст промо (.main-text)

```
.main-text {
  padding-left: 30px;
  color: white;
}
.main-text h2 {
  font-size: 45px;
}
.main-text p {
  color: #b5cff2;
}
```

- Заголовок і абзаци мають різні відтінки для акценту.
- Великий шрифт привертає увагу.

Завантаження зображення (.image-upload)

```
.image-upload button {
  padding: 10px 25px;
  font-size: 20px;
  font-weight: 500;
  background-color: rgb(38, 1, 187);
  color: #fff;
  border-radius: 5px;
  cursor: pointer}

.image-upload header {
  font-size: 32px;
  font-weight: 800;
  color: rgb(0, 4, 83);
}

.image-upload span {
  font-size: 25px;
  font-weight: 500;
  color: rgb(0, 0, 0);
  margin: 10px 0 15px 0;
}
```

- Класичний drag-and-drop блок.
- Виділений великим шрифтом текст, щоб пояснити дію.
- Кнопка виглядає інтерактивною.

Відображення зображення (.image-uploaded)

```
.image-uploaded {
  color: rgb(0, 0, 0);
  width: 50%;
  height: 100%;
  display: none;
  align-items: center;
  justify-content: center;
  flex-direction: column;
}
```

- Прихований блок, який з’являється після вибору файлу.
- Аналогічна структура до image-upload.
- img-div {

```
  width: 100%;
  height: 400px;
  display: flex;
  align-items: center;
  justify-content: center;
}
#imageId {
  max-width: 100%;
```

```
    max-height: 400px;
    border-radius: 4px;
    box-shadow: rgba(0, 0, 0, 0.45) 0px 3px 5px;
}
```

- Центровано зображення, з обмеженням по висоті.
- Тінь додає вигляду "картки" або "фотографії".

Білий контейнер (.white-square)

```
.white-square {
  padding: 10px;
  text-align: center;
  height: 90%;
  width: 90%;
  display: flex;
  flex-direction: column;
  background-color: white;
  box-shadow: rgba(26, 0, 60, 0.45) 0px 5px 15px;
  border-radius: 10px;
}
```

- Основний блок, куди "падає" вміст: або інтерфейс, або зображення.
- Білий фон різко контрастує з темною темою — фокус на контенті.

Секція порівняння Before/After (.before-after-image)

```
.before-after-image {
  border-radius: 10px;
  background: linear-gradient(to right, rgb(38, 1, 187), rgb(69, 156, 255));
  box-shadow: rgba(0, 0, 0, 0.45) 0px 6px 20px;
  margin: auto;
  top: 100px;
  background-color: gray;
  position: relative;
  display: flex;
  flex-direction: row-reverse;
  width: 90%;
  height: 600px;
}
```

display: flex — елементи розташовуються у два стовпчики.

- **flex-direction: row-reverse** — лівий і правий блоки "міняються місцями", тобто логічно зліва — результат, справа — слайдер.
- **background: linear-gradient(...)** — фон з плавним переходом кольорів.
- **box-shadow**, **border-radius** — для об’ємності та м’якості.

- **width: 90%**, **height: 600px** — блок великий, розтягнутий по ширині, щоб зображення було помітне.

Контейнер зображення (.container)

```
.container {
  margin: auto;
  position: relative;
  width: 400px;
  height: 550px;
  background-color: rgb(72, 72, 72);
  border-radius: 10px;
  box-shadow: rgba(0, 0, 0, 0.3) 0px 5px 15px;
}
```

position: relative — дозволяє абсолютно позиціонувати внутрішні елементи, наприклад, foreground-image та background-image.

- **width: 400px**, **height: 550px** — встановлені фіксовані розміри контейнера.
- **background-color: rgb(72, 72, 72)** — темно-сірий фон, який видно під зображеннями, коли вони змінюють ширину.
- **box-shadow** — додає глибини, візуально вирізняє контейнер серед іншого контенту.

Зображення: передній та задній фон

```
.img {
  border-radius: 7px;
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-size: 400px 100%;
}
```

Цей клас спільний для обох зображень (background-image і foreground-image).

- **position: absolute** — дає можливість накладати ці блоки один на одного.
- **border-radius: 7px** — закруглення країв.
- **background-size** — задає, щоб фонова картинка займала весь блок.

```
.foreground-image {
  width: 50%;
```

```
   background-image: url("after.jpg");
}

.background-image {
   background-image: url("before.jpeg");
}
```

foreground-image — "переднє" зображення (покращене). Має змінну ширину, яку ми керуємо через слайдер.

- **background-image** — "заднє" зображення (оригінал), яке видно під переднім.

Слайдер (input[type="range"])

```
.slider {
   position: absolute;
   -webkit-appearance: none;
   appearance: none;
   width: 100%;
   height: 100%;
   background-color: transparent;
   outline: none;
   margin: 0;
   transition: all .2s;
   display: flex;
   justify-content: center;
   align-items: center;
}
```

Повністю прозорий повзунок, який накладається на весь контейнер.

- **position: absolute**, **width/height: 100%** — займає весь простір контейнера, щоб його легко було "хапати".
- **transition: all .2s** — плавне оновлення при зміні значення.
- Виглядає невидимим, але має спеціальний елемент **"повзунок"**:

```
.slider::-webkit-slider-thumb {
   -webkit-appearance: none;
   appearance: none;
   width: 5px;
   height: 550px;
   background: white;
   cursor: pointer;
}
```

Власне, **повзунок** — це вертикальна біла лінія, яка дозволяє рухати роздільну межу між зображеннями.

- Його розмір точно дорівнює висоті контейнера (550px), тому виглядає як вертикальна лінія, що "розрізає" картинку.

Підписи Before/After (.lables та .image-lable)

```
.lables {
  position: absolute;
  display: flex;
  width: 100%;
  justify-content: space-between;
}
.image-lable {
  width: 100px;
  border-radius: 5px;
  background-color: #d2d2d27a;
  color: white;
  font-size: 14px;
  text-align: center;
  z-index: 1;
}
```

.lables — горизонтальний контейнер для написів "Before" і "After".

- Кожен **.image-lable** — напівпрозорий прямокутник з підписом.
- Колір фону — прозорий сірий (#d2d2d27a), що виглядає акуратно і не перекриває зображення повністю.

```
#left {
  top: 0;
  left: 0;
}
#right{
  top: 0;
  right: 0;
}
```

Ці ID позиціонують відповідні написи зліва та справа вгорі контейнера.

Центровані кнопки під текстом (.button-row-center)

```
.button-row-center {
  display: flex;
  width: 70%;
  justify-content: space-evenly;
  align-items: center;
  margin: auto;
  margin-top: 0;
  margin-bottom: 0;
}
```

- Контейнер для кнопок "Pricing" і "Try for free".
- Класичне **центровано-поділене** вирівнювання: одна ліворуч, одна праворуч, з відступами між ними.

Модальне вікно (попап)

Фон під попапом: затемнення (.popup-bg)

```
<div class="popup-bg" onclick="closePopup()"></div>
```

```
.popup-bg {

}
```

Це повноекранний затемнений фон, який з’являється під модальним вікном, але над усім іншим контентом.

- Зазвичай має position: fixed, top: 0, left: 0, width: 100%, height: 100%, background: rgba(0,0,0,0.6) щоб створити ефект затемнення.
- Атрибут onclick="closePopup()" означає, що клік по фону закриє попап.

Власне попап (.popup)

```
<div class="popup">
  <div style="width: 300px;">
    <img src="" class="edit-img">
  </div>
</div>
```

```
.popup {

}
```

```
.popup {
  position: fixed;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  background: white;
  padding: 20px;
  border-radius: 10px;
  z-index: 10;
  box-shadow: rgba(0,0,0,0.5) 0px 10px 25px;
}
```

Усередині блоку знаходиться зображення, яке буде відображене з class edit-img.

Стилі зображень (.edit-img)

```
.edit-img {
  height: auto;
  max-height: 600px;
  display: block;
  max-width: 100%;
}
```

Клас використовується у двох місцях:

- Для показу обрізаного зображення (в попапі).
- Для перегляду завантаженого зображення (у блоці .image-uploaded).

- Основні властивості:
  - **max-height: 600px** — захищає від розтягування по висоті.
  - **max-width: 100%** — зображення не виходить за межі блоку.
  - **display: block** — прибирає можливі зайві відступи, пов’язані з inline-елементами.

Інші дрібні стилі

Кнопки під зображенням (.button-list)

```
.button-list {
  margin: 10px;
  display: flex;
  width: 90%;
  justify-content: space-evenly;
}
```

- Контейнер для трьох кнопок: "Back", "Proceed", "Edit".
- Стандартний flex-блок з рівномірним розподілом кнопок.

Контейнер для зображення (.img-div)

```
.img-div {
  width: 100%;
  height: 400px;
  position: relative;
  display: flex;
  align-items: center;
  justify-content: center;
}
```

- Центрує зображення всередині білої "картки".
- Зберігає стабільну висоту навіть якщо картинка ще не завантажена.

```
result-img
#result-img {
  width: 400px;
  height: auto;
}
```

Встановлює ширину результату й зберігає пропорції.

3.3. Логіка взаємодії з користувачем (JavaScript)

Оголошення змінних

```
const imageUpload = document.querySelector(".image-upload"),
imageUploaded = document.querySelector(".image-uploaded"),
image = document.querySelector("#imageId"),
dragText = imageUpload.querySelector("header"),
button = imageUpload.querySelector("button"),
input = imageUpload.querySelector("input"),
editImg = document.querySelector(".edit-img"),
imgResult = document.querySelector("#img-result");
resultImg = document.querySelector("#result-img");
```

- querySelector витягує HTML-елементи, з якими ми будемо працювати.
- **imageUpload** — контейнер, у який можна перетягнути зображення.
- **imageUploaded** — блок, який показується після завантаження.
- **image** — тег <img> для прев'ю завантаженого зображення.
- **dragText** — заголовок усередині imageUpload, де пише "Drag & Drop...".
- **button** — кнопка "Browse".
- **input** — прихований інпут типу file.
- **editImg** — зображення у попапі.
- **imgResult**, **resultImg** — для фінального результату.

Змінні для обробки зображення

```
var basic;
let file;
let fileURL;
let imgWidth, imgHeight;
var cropper;
let finalUrl;
```

Ці змінні використовуються для обробки файлів.

- file — об'єкт файлу, який вибрав користувач.
- fileURL — Base64-представлення файлу.
- cropper — екземпляр Cropper.js, який дозволяє обрізати зображення.
- finalUrl — потенційна змінна для готового результату.

Обробка натискання на кнопку "Browse"

```
button.onclick = () => {
input.click();
```

```
}
```

При кліку на кнопку запускається подія кліку на прихований <input type="file">.

- Це дозволяє оформити завантаження без стандартного браузерного вікна.

Обробка вибору файлу

```
input.addEventListener("change", function(){
  file = this.files[0];
  imageUpload.classList.add("active");
  showFile();
});
```

Після вибору файлу ми:

- Зберігаємо його у змінну file.
- Додаємо клас active до imageUpload (можна для стилізації).
- Викликаємо showFile() — основну функцію обробки.

Drag & Drop

```
imageUpload.addEventListener("dragover", (event) => {
  event.preventDefault();
  imageUpload.classList.add("active");
  dragText.textContent = "Release to Upload File";
});
```

Коли користувач перетягує файл у зону — ми:

- **preventDefault()** — обов’язково, інакше браузер відкриє файл у новій вкладці.
- Додаємо клас active, міняємо текст.

```
imageUpload.addEventListener("dragleave", () => {
  imageUpload.classList.remove("active");
  dragText.textContent = "Drag & Drop to Upload File";
});
```

- Якщо файл пішов за межі області — повертаємо все назад.

```
imageUpload.addEventListener("drop", (event) => {
  event.preventDefault();
  file = event.dataTransfer.files[0];
  showFile();
});
```

При "скиданні" файлу:

- Отримуємо перший файл із dataTransfer.
- Передаємо його до showFile().

Основна функція showFile()

```
function showFile(){
   let validExtensions = ["image/jpeg", "image/jpg", "image/png"];
   if(validExtensions.includes(file.type)){
      let fileReader = new FileReader();
      fileReader.onload = () => {
         fileURL = fileReader.result;
         imageUploaded.style.display = "flex";
         imageUpload.style.display = "none";
         image.src = fileURL;
         cropper = new Cropper(image, {
            viewMode: 2,
            movable: false,
            resizable: false,
         });
      };
      fileReader.readAsDataURL(file);
      input.value = "";
   } else {
      alert("This is not an Image File!");
      imageUpload.classList.remove("active");
      uploadBack;
   }
}
```

- **Валідація типу файлу** — тільки зображення.
- **FileReader.readAsDataURL(file)** — читає зображення у форматі Base64.
- Після завантаження:
  - Міняємо видимість блоків.
  - Вставляємо картинку.
  - Ініціалізуємо cropper —

Функція uploadBack()

```
function uploadBack(){
   imageUploaded.style.display = "none";
   imageUpload.style.display = "flex";
   cropper.destroy();
   cropper = "";
}
```

Повертає все назад: видаляє cropper, показує блок завантаження знову.

Збереження та надсилання файлу — saveUpload()

```
function saveUpload(){
   let canvas = cropper.getCroppedCanvas();
   var croppedImage = new Image();
   croppedImage.src = canvas.toDataURL();
   let formData = new FormData();
   formData.append("file", file);
   console.log(formData);
   axios.post('http://localhost:8080/upload', formData, {
      headers: {
         'Content-Type': "multipart/form-data"
      }
   })
   .then(function (response) {
      console.log(response);
   })
   .catch(function (error) {
      console.log(error);
   });
}
```

getCroppedCanvas() — отримує обрізаний фрагмент з Cropper.

- Перетворюється в **Base64-строку**.
- Файл додається у FormData, яка зручно відправляється через axios.
- Відправка йде на **локальний сервер**: localhost:8080/upload.

Повзунок "до/після" — динаміка

```
window.onload = () => {
   $("#slider-id").on('input', (e) => {
      const sliderPos = e.target.value;
      $(".foreground-image").css("width", `${sliderPos}%`);
   });
}
```

При зміні слайдера:

- Значення читається (1-100).
- Ширина переднього зображення змінюється — і ми бачимо ефект порівняння.

Інтеграція клієнтської частини сайту з нейронною мережею для апскейлу зображень

У межах реалізації функціоналу покращення якості зображення (апскейлу) була реалізована інтеграція фронтенд-частини вебсайту з нейронною мережею, яка працює на серверній стороні. Дана інтеграція забезпечує можливість обробки завантаженого користувачем зображення, попереднього

його обрізання (crop), подальшої відправки на сервер, а також відображення результату після проходження через модель глибинного навчання.

Технологічна схема

Загальний порядок взаємодії компонентів виглядає наступним чином:

1. Користувач завантажує зображення та виконує обрізання через графічний інтерфейс Cropper.js.
2. JavaScript-скрипт перетворює зображення у формат Blob, обгортає його у FormData та відправляє HTTP-запит до бекенд-сервера.
3. Сервер приймає файл, зберігає його у тимчасову директорію, передає до нейронної мережі для обробки.
4. Результат зберігається окремо та повертається користувачу у вигляді URL, який одразу ж виводиться на екран.

Клієнтська частина (JavaScript): відправлення зображення на обробку

```
const canvas = cropper.getCroppedCanvas({
   width: 512,
   height: 512,
});

canvas.toBlob((blob) => {
   const formData = new FormData();
   formData.append("image", blob, "cropped-image.png");

   axios.post("http://localhost:5000/process", formData, {
      headers: {
         "Content-Type": "multipart/form-data",
      },
   })
   .then((response) => {
      const resultUrl = response.data.result;
      document.querySelector("#result-image").src = resultUrl;
   })
   .catch((error) => {
      console.error("Помилка обробки:", error);
   });
});
```

У даному фрагменті використовується бібліотека axios для здійснення HTTP POST-запиту. Дані збираються у форматі FormData, що дозволяє передати файл зображення на сервер у форматі, який легко обробити на стороні бекенду.

Серверна частина (Flask, Python): прийом файлу та запуск нейронної мережі

На сервері розгорнуто середовище Python з використанням фреймворку Flask. Основна логіка взаємодії з фронтендом реалізована наступним чином:

```
@app.route("/process", methods=["POST"])
def process_image():
    if "image" not in request.files:
        return jsonify({"error": "Файл не надіслано"}), 400

    file = request.files["image"]
    filename = f"{uuid.uuid4()}.png"
    filepath = os.path.join("uploads", filename)
    file.save(filepath)

    result_path = os.path.join("results", filename)
    upscale_image(filepath, result_path)

    return jsonify({"result": f"/results/{filename}"})
```

Вхідне зображення зберігається на сервер у тимчасовій папці uploads. Далі викликається функція upscale_image(), яка забезпечує запуск нейронної мережі.

Інтеграція з нейромережею (приклад використання Real-ESRGAN)

Для апскейлу використовується відома нейромережа Real-ESRGAN, яка зарекомендувала себе як одна з найефективніших моделей для покращення якості зображень. Вона підтримує роботу як на CPU, так і на GPU. Запуск моделі реалізований у Python-скрипті:

```
def upscale_image(input_path, output_path):
    import subprocess
    subprocess.run([
        "python", "inference_realesrgan.py",
        "-i", input_path,
        "-o", output_path,
        "-n", "RealESRGAN_x4plus"
    ])
```

У даному прикладі модель запускається як окремий процес за допомогою модуля subprocess. Такий підхід дозволяє відокремити роботу інтерфейсу та важкої обробки зображень, не блокуючи основний потік запитів Flask-сервера, минаючи складну обробку, не навантажуючи браузер, а відповідно —

підтримка високої продуктивності і якості сервісу навіть на слабких клієнтських пристроях.

Альтернатива: прямий виклик моделі через Python API

```
from realesrgan import RealESRGANer

def upscale_image(input_path, output_path):
    from PIL import Image
    image = Image.open(input_path).convert("RGB")

    model = RealESRGANer(
        scale=4,
        model_path="weights/RealESRGAN_x4plus.pth",
        tile=0,
        pre_pad=0,
        half=True
    )

    output, _ = model.enhance(image)
    output.save(output_path)
```

У цьому варіанті модель імпортується як Python-клас і використовується безпосередньо у скрипті, що пришвидшує запуск і зменшує затримку (рисунок 3.1).

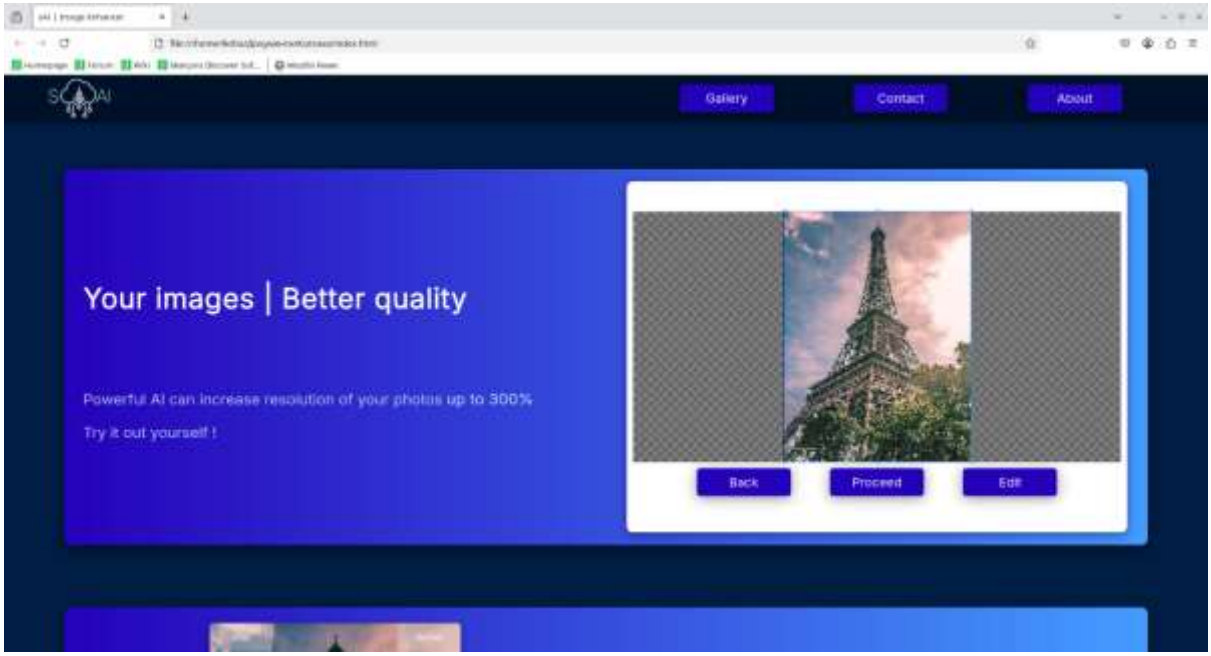


Рисунок 3.1 – Головна сторінка сайту

Завантаження результату на клієнт

Після успішної обробки URL до нового зображення повертається у JSON-форматі та відображається у браузері (рисунок 3.2):

document.querySelector("#result-image").src = response.data.result;

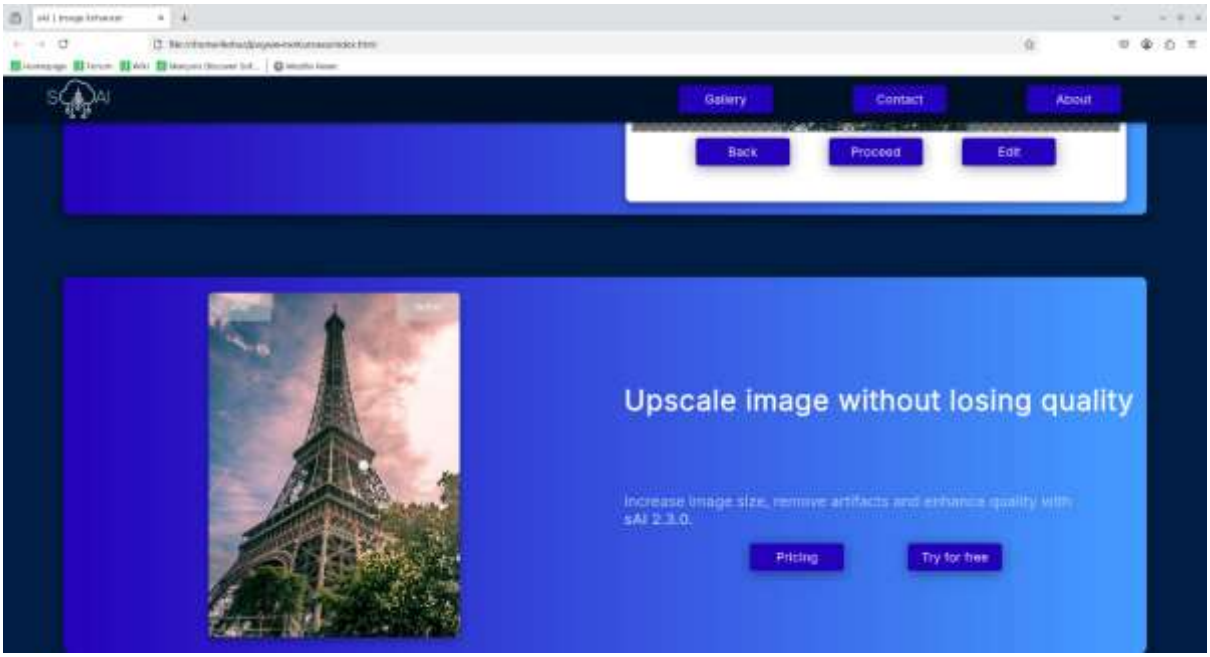


Рисунок 3.2 – Кінцеве зображення сторінки сайту для порівняння

Це забезпечує миттєвий зворотній зв'язок для користувача — він бачить "до" і "після" одразу у веб-інтерфейсі.

Інтеграція клієнтської частини з серверною нейронною мережею реалізована з урахуванням сучасних підходів до передачі даних, асинхронності та масштабованості. Використання нейромережі на бекенді дозволяє забезпечити високу точність обробки запитів, адаптивність до змін у вхідних даних та можливість розширення функціональності системи без суттєвого впливу на продуктивність. Такий підхід також сприяє швидкому реагуванню на запити користувача та ефективному використанню обчислювальних ресурсів, що є критично важливим для сучасних вебзастосунків та сервісів на основі штучного інтелекту.

ВИСНОВКИ

Нейронні мережі є важливим інструментом для розпізнавання об'єктів на зображеннях. Вони дозволяють обробляти великі обсяги даних та виявляти невідомі об'єкти, а також здатні адаптуватися до різних контекстів зображень. Незважаючи на обмеження, такі як необхідність великої кількості даних та обчислювальну складність, нейронні мережі залишаються одним з найбільш ефективних методів для вирішення завдань розпізнавання об'єктів на зображеннях.

Введення нейронних мереж у область розпізнавання зображень дозволило вирішити багато проблем. Нейронні мережі, на відміну класичних методів, не вимагають вилучення ознак із зображень вручну, а навчаються великих обсягах даних із відомими мітками. Вони також дозволяють обробляти зображення з різним контекстом, таким як фон, освітлення та масштаб, та можуть виявляти невідомі об'єкти.

Під час роботи детально розглянуто архітектуру GAN та його основні компоненти, включаючи генератор та дискримінатор, Проаналізовано процес навчання GAN, включаючи вибір оптимізатора та функціоналу втрат, та застосовано його для навчання нашої моделі з використанням доступного набору даних.

Провівши експерименти та аналіз результатів, можна зробити висновок, що наша реалізація DCGAN демонструє вражаючу здатність генерувати високоякісні зображення, які візуально відповідають тим, на яких навчалася модель.

DCGAN є потужним інструментом у галузі розпізнавання зображень та генерації нових, реалістичних візуальних даних. Програми GAN можуть бути широкими, включаючи області медичного зображення, створення штучних сцен та багатьох інших. Однак, як і у будь-якої технології, є деякі обмеження та можливості для покращення, які вимагають подальших досліджень та розробок.

У даному веб-застосунку використовуються окремі CSS-файли – cropper**.css**, croppie**.css** та popup.css, які відповідають за стилізацію специфічних

інтерфейсних компонентів, зокрема редакторів зображень та спливаючих вікон. Це дозволяє забезпечити зручну взаємодію з користувачем на етапах редагування і перегляду.

Основний файл стилів — style.css — відповідає за візуальне оформлення вебінтерфейсу в цілому: він задає кольорову палітру, шрифти, відступи, розміри та адаптивну поведінку елементів. Всі стилі написані вручну без використання CSS-препроцесорів, що спрощує структуру коду та підвищує його читабельність.

Однією з ключових функцій веб-застосунку є покращення якості зображень за допомогою нейронної мережі. Процес реалізовано через тісну інтеграцію фронтенду з серверною частиною.

Технологічна схема взаємодії:

1. Користувач завантажує зображення та виконує обрізання через інтерфейс, побудований на бібліотеці Cropper.js.

2. JavaScript-логіка обробляє зображення, конвертує його у формат Blob і формує FormData, яка надсилається на сервер через HTTP POST-запит, реалізований за допомогою бібліотеки axios.

3. Сервер (на Flask) приймає файл, тимчасово зберігає його, передає до нейронної мережі.

4. Запуск моделі глибинного навчання виконується як окремий процес через subprocess, що дозволяє уникнути блокування головного потоку Flask-сервера.

5. Оброблене зображення зберігається на сервері та повертається користувачеві у вигляді URL, який відображається у вебінтерфейсі.

Реалізована архітектура демонструє ефективний поділ функціональності між фронтендом і бекендом. Використання окремих CSS-файлів забезпечує гнучке стилістичне оформлення компонентів, а взаємодія з серверною нейронною мережею дозволяє користувачу отримувати результат покращення зображень у зручній формі. Такий підхід дозволяє масштабувати рішення, додавати нові функції обробки без потреби глибокого переписування існуючого коду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Мертенс Т., Каутц Дж. та Ван Ріт Ф. (2009). Злиття експозиції. У матеріалах 16-ї конференції Eurographics про техніку візуалізації (стор. 161-172).
2. Tian, Y., Tao, D., Zhang, Y., & Xu, C. (2017). Порівняльне дослідження методів покращення зображення в умовах слабкого освітлення для мобільних пристроїв. Мультимедійні засоби та програми.
3. Буадес, А., Колл, Б., і Морель, Дж. М. (2005). Огляд нових алгоритмів усунення шумів на зображенні. SIAM Journal про багатомасштабне моделювання та симуляцію, 4(2), 490-530.
4. Берковський В.В., Радіонов В.О., Бурковський В.С. Зіставлення технологій створення сайтів з їх цільовим призначенням / В.В. Берковський // Наука і техніка Повітряних Сил Збройних Сил України. – 2015. – № 2(19). – С. 122-124.
5. Дабов К., Фой А., Катковник В. та Егіазарян К. (2007). Усунення шумів зображення шляхом спільної фільтрації 3-D домену перетворення. IEEE Transactions on Image Processing, 16(8), 2080-2095.
6. "Сліпа деконволюція для покращення зображення при слабкому освітленні" Чжан та ін., arXiv preprint arXiv:2009.07586, 2017.
7. "Несліпа деконволюція для покращення зображення при слабкому освітленні" Yang et al., arXiv preprint arXiv:2003.07546, 2015.
8. «Несліпа деконволюція на основі глибокого навчання для покращення зображення при слабкому освітленні» Ван та ін., arXiv preprint arXiv:2009.04563, 2021.
9. «Локальні фільтри Лапласа: обробка зображень з урахуванням ребер за допомогою піраміди Лапласа» Періс та ін., IEEE Transactions on image processing (TIP), 2011.
10. «Адаптивне глобальне відображення тонів для зображень із високим динамічним діапазоном» Рейнхарда та ін., 2002.
11. «Тонове відображення на основі глибокого навчання для зображень із слабким освітленням» Лю та ін., Trans. Sig. Proc. , 2021. 78

12. «Надзвичайна роздільна здатність одного зображення для умов слабкого освітлення з використанням регресійної моделі процесу Гаусса» Ху та ін., 2013.
13. Miletsky J.I. Principles of Internet Marketing: New Tools and Methods for Web Developers : Course Technology Press / J.I. Miletsky. – Boston, MA, United States, 2009. – 480
14. A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects /Zewen Li, Wenjie Yang, Shouheng Peng, Fan Liu. – 2020. [Електронний ресурс]– Режим доступу: https://arxiv.org/ftp/arxiv/papers/2004/2004.02806.pdf
15. Vieira, A. Introduction to deep learning business applications for developers / A.Vieira, B.Ribeiro. – New York: Apress, 2018. – 343 p.
16. Generative Adversarial Nets / Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza – 2014. [Електронний ресурс] – Режим доступу: https://arxiv.org/pdf/1406.2661.pdf