МАГІСТЕРСЬКА РОБОТА

МР. ІПм - 33.00.00.000 ПЗ

Група ІПм-22-5

Кулик Роман

2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Кулик Роман Миколайович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

МАГІСТЕРСЬКА РОБОТА

Методи, моделі та алгоритми

відновлення зображень

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

Кулик Р.М.

(підпис, ініціали та прізвище здобувача освітнього ступеня)

Науковий керівник Яцишин Микола Миколайович, доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Допущено до захисту

В.о. завідувача кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Рецензент

(посада) (підпис) (дата) (ініціали та прізвище)

Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Івано-Франківськ – 2024

Івано-Франківський національний технічний університет нафти і газу

Інститут інформаційних технологій

Кафедра інженерії програмного забезпечення

Освітньо-кваліфікаційний рівень магістр

Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
В.о. зав. кафедрою **ІПЗ**
доц. В.В. Бандура
“ 04 ” вересня 2023 р.

ЗАВДАННЯ

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Кулику Роману Миколайовичу

(прізвище, ім’я, по-батькові)

1. Тема магістерської роботи **“**Методи, моделі та алгоритми відновлення зображень**”**
керівник проекту (роботи) Яцишин Микола Миколайович, доцент
затверджені наказом закладу вищої освіти від **“** 18 ” грудня 2023 р. № 738/7

2. Строк подання студентом проекту (роботи) 15 січня 2024 р.

3. Вихідні дані до проекту (роботи) Архітектура, формальний опис та алгоритми штучного інтелекту

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

1. Теоретичні засади відновлення зобразення
2. Методи та підходи оцінки якості відновлення зображень
3. Дослідження існуючих методів та програмного забезпечення для системи реконструкції зображень
4. Розробка алгоритму та програмна реалізація системи відновлення зображення

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1. Растрове зображення (рис. 1.1)
2. Приклад зменшення шуму (рис. 1.2)
3. Приклад підвищення різкості (рис. 1.3)
4. Приклад корекції кольору та контрасту (рис. 1.4)
5. Приклад усунення дефектів (рис. 1.5)

6. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата
Нормоконтроль	доц. к.т.н. Вовк Р. Б.	
Перевірка на плагіат	доц. к.т.н. Вовк Р. Б.	

7. Дата видачі завдання 04 вересня 2023 р.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури по темі відновлення зображення	20.09.2023	виконано
2	Аналіз та порівняння сучасних методів відновлення зображень	01.10.2023	виконано
3	Дослідження існуючого програмного забезпечення, які використовують реконсттрукцію зображення	20.10.2023	виконано
4	Розробка та аналіз алгоритму відновлення зображення	15.11.2023	виконано
5	Формулювання вимог та розробка алгоритму для відновлення зображення	03.12.2023	виконано
6	Програмна реалізація відновлення зображення	22.12.2023	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	15.01.2024	виконано

Студент – магістр _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

Магістерська робота: 90 с., 25 рис., 45 джерело, 3 додатків

Тема: Методи, моделі та алгоритми відновлення зображень.

Об’єкт дослідження: Сучасні методи та підходи відновлення зображень.

Мета роботи: Провести аналіз сучасних підходів до відновлення зображень. Розробити та впровадити інноваційні підходи до відновлення зображень.

Предмет дослідження: Детальний аналіз та оцінка сучасних методів і підходів до відновлення зображень, з особливим акцентом на розробку та впровадження інноваційних технік і алгоритмів у цій галузі. Включає вивчення різноманітних технологій та методик, таких як машинне навчання, штучний інтелект, комп'ютерний зір, а також традиційні підходи до обробки зображень.

Результати дослідження:

Проаналізовано низку існуючих програмних засобів та сучасні методи та підходи до відновлення зображень. Представлено новітні методи та підходи до відновлення зображень.

Висновок:

Робота демонструє, як методики та технології штучного інтелекту можуть бути використані для реставрації зображень, розширюючи горизонти їхнього подальшого розвитку та інтеграції з новітніми технологіями.

ВІДНОВЛЕННЯ ЗОБРАЖЕННЯ, ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, АНАЛІЗ ДАНИХ, РОЗПОДІЛЕНА АРХІТЕКТУРА, ВЕБ-СЕРВЕР.

ANNOTATION

Master's work: 90p., 25 fig., 45 sources, 3 appendices

Topic: Methods, models and algorithms for Image Restoration.

Object of research: Modern methods and approaches to image restoration.

Purpose: To conduct an analysis of modern approaches to image restoration. To develop and implement innovative approaches to image restoration.

Subject of research: Detailed analysis and evaluation of modern methods and approaches to image restoration, with a focus on the development and implementation of innovative techniques and algorithms in this field. This includes the study of various technologies and methodologies, such as machine learning, artificial intelligence, computer vision, as well as more traditional approaches to image processing.

Research results:

A range of existing software has been analyzed, as well as modern methods and approaches to image restoration. New methods and approaches to image restoration have been presented.

Conclusion:

The work demonstrates how techniques and artificial intelligence technologies can be used for image restoration, expanding the horizons for their further development and integration with cutting-edge technologies.

IMAGE RESTORATION, ARTIFICIAL INTELLIGENCE, MACHINE LEARNING, DATA ANALYSIS, DISTRIBUTED ARCHITECTURE, WEB SERVER.

ЗМІСТ

Стр.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI - Штучний інтелект (Artificial Intelligence)

API - Інтерфейс програмування додатків (Application Programming Interface)

CNN - Згорткова нейронна мережа (Convolutional Neural Network)

DnCNN - Глибокі згорткові нейронні мережі для денойзингу зображень (Deep Convolutional Neural Networks for Image Denoising)

ESRGAN - Покращена супер-роздільна здатність за допомогою генеративно-змагальних мереж (Enhanced Super-Resolution Generative Adversarial Networks)

HTTPS - Протокол безпечної передачі гіпертексту (Hypertext Transfer Protocol Secure)

MAE - Середня абсолютна помилка (Mean Absolute Error)

MSE - Середньоквадратична помилка (Mean Squared Error)

MVC - Модель-Вид-Контролер (Model-View-Controller)

PSNR - Відношення сигнал/шум пікового рівня (Peak Signal-to-Noise Ratio)

Redis - Віддалена система взаємодії зі структурами даних (Remote Dictionary Server)

Replicate API - API для реплікації функцій та даних

SRCNN - Згорткова нейронна мережа супер-роздільної здатності (Super-Resolution Convolutional Neural Network)

SSIM - Індекс структурної схожості (Structural Similarity Index)

VIF - Візуальна вірогідність інформації (Visual Information Fidelity)

ПЗ - програмне забезпечення

DDOS - Атака на відмову в обслуговуванні

ОС - операційна система

HTTPS - протокол безпечної передачі даних

RSA - криптографічний алгоритм з відкритим ключем

ВСТУП

Актуальність роботи

В останні роки з величезним розширенням сфери обробки зображень з'являється все більше завдань, пов'язаних з відновленням пошкоджених, зашумлених або неповних зображень. Це набуває особливої важливості в контексті зростаючих обсягів зображень з різних джерел, таких як медичні знімки, відеозаписи, зображення, отримані за допомогою різних сенсорів та камер, а також архівні фотографії.

Магістерська робота на тему "Методи, моделі та алгоритми відновлення зображень" спрямована на дослідження та аналіз широкого спектру інноваційних методів та підходів, які мають на меті поліпшення якості зображень через їх відновлення. Робота охоплює як теоретичні основи, так і практичні аспекти відновлення зображень за допомогою сучасних математичних моделей та алгоритмів.

Під час цього дослідження планується розгляд та порівняння ефективності різних методів відновлення зображень в різних умовах та ситуаціях, включаючи вплив шуму, зміни контрастності або пошкодження даних. Окрема увага буде приділена вивченню практичного застосування відновлення зображень у таких сферах, як медична діагностика, криміналістика, візуальне мистецтво та архітектура.

Робота має на меті не лише теоретичне обґрунтування різних методів відновлення зображень, але й їх практичне застосування у реальних умовах. Розглядувані моделі та алгоритми мають широкий спектр потенційних застосувань, включаючи медичну діагностику, візуальне спостереження, реставрацію старих архівних зображень та інше.

У рамках цієї магістерської роботи буде проведено аналіз сучасних підходів до відновлення зображень, порівняння їх ефективності та реалізація вибраних методів для вирішення конкретних завдань. Результати дослідження можуть сприяти

подальшому розвитку методів відновлення зображень та їх застосуванню у різних областях.

Порівняння розробленого рішення з відомими методами відновлення зображень

У рамках цієї роботи проведено аналіз ряду сучасних рішень у сфері відновлення зображень. Зокрема, увагу зосереджено на різноманітних методах та підходах, включаючи відновлення з використанням машинного навчання та обробки даних. Порівняно з традиційними рішеннями, розроблений у даній магістерській роботі підхід демонструє значну адаптивність, покращену ефективність та підвищений рівень безпеки даних під час роботи з користувацькими зображеннями.

На відміну від стандартних систем, які часто обмежені фіксованими алгоритмами, запропонований алгоритм активно використовує аналітику даних та методи машинного навчання для досягнення вищої якості та зручності відновлення зображень. Цей підхід не лише сприяє поліпшенню якості обробки та відновлення зображень, але й гарантує більш безпечний досвід користувача.

Таким чином, ця робота робить важливий внесок у розвиток веб-сервісів, демонструючи значні переваги інтеграції штучного інтелекту для підвищення їхньої ефективності та адаптивності.

Мета і задачі дослідження

Мета дослідження полягає у розробці та аналізі ефективних методів та підходів відновлення зображення. З особливим акцентом на розробці веб сервісу для взаємодії з користувачами. Це дослідження спрямоване на вивчення сучасних тенденцій та технологій. Також спрямоване на впровадження передових практик в відновлені зображень та області ШІ для підвищення якості.

Для досягнення цієї мети були визначені наступні задачі:

- Подання математичних моделей, що використовуються для опису пошкоджених зображень та методів їх відновлення. Розгляд моделей шуму, дефектів та інших артефактів
- Аналіз та порівняння сучасних методів відновлення зображень, включаючи фільтрацію, реконструкцію, та застосування машинного та глибинного навчання.
- Визначення метрик для оцінки якості відновлення, таких як PSNR, SSIM, MSE, MAE тощо.
- Використання об'єктивних та суб'єктивних методів оцінки якості відновлення.
- Дослідження існуючих підходів та програмного забезпечення для реконструкції зображень.
- Розробка алгоритму системи відновлення зображень, адаптованого до потреб та поведінки користувачів.
- Тестування а також оцінка ефективності системи на основі реальних сценаріїв використання.
- Аналіз потенційних проблем та викликів у сфері відновлення зображень та розробка рекомендацій щодо їх вирішення.

Робота націлена на застосування новітніх методик та використання можливостей штучного інтелекту з метою покращення ефективності та забезпечення безпеки процесів відновлення зображень.

Об’єктом дослідження: У цій роботі фокус зроблено на вивченні методів, моделей, алгоритмів та програмних рішень, які застосовуються для відновлення зображень. Особлива увага приділяється аспектам, що стосуються взаємодії з користувачами та ефективності процесу відновлення, зокрема з використанням можливостей штучного інтелекту для підвищення якості та безпеки обробки зображень.

Предмет дослідження: У центрі уваги даного дослідження знаходиться детальний аналіз та оцінка сучасних методів і підходів до відновлення зображень. Особлива увага приділяється розробці та впровадженню інноваційних технік та алгоритмів у цій сфері, охоплюючи широкий спектр технологій, включно з машинним навчанням, штучним інтелектом, комп'ютерним зором, а також залученням традиційних методів обробки зображень. Метою є не лише вивчення цих методів, але й адаптація та вдосконалення їх для конкретних потреб відновлення зображень у веб-сервісах, забезпечуючи при цьому високу якість та персоналізоване використання.

Методи дослідження

Це дослідження включає комплексний аналіз та розробку з метою досягнення визначених цілей. Застосовується систематичний підхід до оцінки існуючих методів відновлення зображень, який об'єднує детальний огляд академічної літератури, аналіз передових практик та вивчення актуальних трендів у сфері штучного інтелекту. Для перевірки теоретичних концепцій проводяться експериментальні розробки та тестування, що дозволяє оцінити їх практичну придатність у процесах відновлення зображень. Аналітичні методи застосовуються для оцінювання якості відновлених зображень та безпеки збереження зображення. Також використовуються кількісні методи аналізу для обробки даних, що допомагає виявити загальні тенденції та оцінити ефективність різноманітних методів.

Наукова новизна одержаних результатів

Наукова новизна цього дослідження виражається у розробці інноваційного методу відновлення зображень з використанням передових технологій штучного інтелекту. Розроблений алгоритм відновлення зображень є значним внеском у галузь застосування штучного інтелекту, сприяючи підвищенню якості зображень та оптимізації взаємодії між користувачами та веб-сервісами.

Ця магістерська робота вносить важливий вклад у сферу відновлення зображень, демонструючи новаторське використання штучного інтелекту у веб-додатках. Ключовою новинкою є інтеграція бліотеки Replicate AI для відновлення зображень у веб-сервіс, що надає користувачам можливість безпосередньо взаємодіяти з інтелектуальним сервісом та використовувати його потенціал для покращення якості та відновлення зображень.

Практичне значення одержаних результатів

Практичне значення цього дослідження полягає у можливості застосування розробленого алгоритму відновлення зображень у різноманітних застосуваннях. Використання цього алгоритму вносить суттєвий вклад у забезпечення вищої ефективності, безпеки обробки даних та поліпшення користувацького досвіду. Це важливо для досягнення успішної цифрової трансформації в різних сферах, включаючи, але не обмежуючись, медичною діагностикою, безпекою, цифровим мистецтвом та іншими галузями, де якість зображення має критичне значення.

Особистий внесок

Мій внесок в це дослідження полягає у проведенні глибокого аналізу існуючих методів відновлення зображень, з акцентом на використанні штучного інтелекту. Я успішно реалізував програмний алгоритм відновлення зображень з використанням штучного інтелекту, який може адаптуватися до користувацьких потреб, забезпечуючи більш інтерактивну та персоналізовану взаємодію. Також, моя робота була спрямована на забезпечення безпеки та конфіденційності даних при використанні штучного інтелекту в веб-сервісах, і я розробив рекомендації для покращення цих аспектів. Мої зусилля в цьому проекті включали як теоретичний аналіз, так і практичну реалізацію методів відновлення зображень, демонструючи успішну інтеграцію теорії та практики в галузі штучного інтелекту.

Структура магістерської роботи.

Магістерська робота викладена на 90 сторінках друкованого тексту, який складається з вступу, чотирьох розділів, висновків, списку використаних джерел (47 найменувань) та додатків. Робота містить 25 рисунків.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ЗАСАДИ ВІДНОВЛЕННЯ ЗОБРАЖЕННЯ

1.1 Основні поняття принципів та методів, що лежать в основі обробки та відновлення зображень

Обробка зображень - це галузь науки, що займається аналізом, модифікацією та відновленням цифрових зображень. Основна мета полягає в покращенні якості зображень, вилученні корисної інформації та підготовці їх для подальшого аналізу або використання в різних застосунках. Вивчення обробки зображень включає в себе наступні ключові поняття, принципи та методи:

1. Цифрове зображення - це зображення, що складається з дискретних пікселів, кожен з яких має певне значення яскравості або колір. Цифрові зображення зберігаються у вигляді матриці чисел, де кожне число відображає яскравість або колір пікселя (рис. 1.1).

Рис. 1.1 Растрове зображення

Основні характеристики цифрового зображення:

- Роздільна здатність (розширення): Це кількість пікселів у зображенні. Воно вимірюється у горизонтальному та вертикальному

напрямках (наприклад, 1920 x 1080), що визначає загальну кількість пікселів у зображенні та його чіткість.

- Глибина кольору або яскравість: Це кількість бітів, які використовуються для кожного пікселя зображення для кодування кольорів або рівнів яскравості. Це визначає можливу кількість різних кольорів або відтінків, які можуть бути відображені в кожній точці зображення.
- Формат зберігання: Це спосіб, яким зображення зберігається у цифровому форматі. Популярні формати включають JPEG, PNG, TIFF, GIF, BMP та RAW. Кожен з них має свої особливості, такі як рівень стиснення, підтримка прозорості, якість збереження метаданих тощо.
- Масштабування та обрізка: Це можливості зміни розміру зображення без втрати якості. Масштабування може бути здійснене збільшенням або зменшенням розміру зображення, тоді як обрізка дозволяє вирізати певні частини зображення.

Цифрові зображення знаходять широке застосування у всіх сферах життя: фотографія, медицина, дизайн, наука, реклама, мультимедіа та багато інших. Обробка цифрових зображень включає в себе такі процеси, як обробка зображень, реконструкція, відновлення, виявлення об'єктів, стиснення та аналіз зображень за допомогою спеціалізованих програм та алгоритмів обробки зображень.

2. Попередня обробка зображень - це стадія обробки, яка включає ряд операцій та методів для покращення якості зображення перед його подальшим аналізом, обробкою або використанням у різних застосунках. Цей етап підготовки зображення може включати наступні операції:

- Шумоподавлення: У цій операції застосовуються фільтри для зменшення шуму, який може бути присутній у зображенні. Різні фільтри, такі як медіанний, середньоквадратичний або фільтри за гаусовим розподілом, можуть використовуватися для зменшення різних видів шуму (рис. 1.2).

Рис. 1.2. Приклад зменшення шуму

- Підвищення різкості (шарпенінг): Це процес підвищення чіткості зображення шляхом підсилення контрастності або виділення деталей. Такий ефект може бути досягнутий за допомогою фільтрів, наприклад, фільтру підвищення різкості (рис. 1.3).

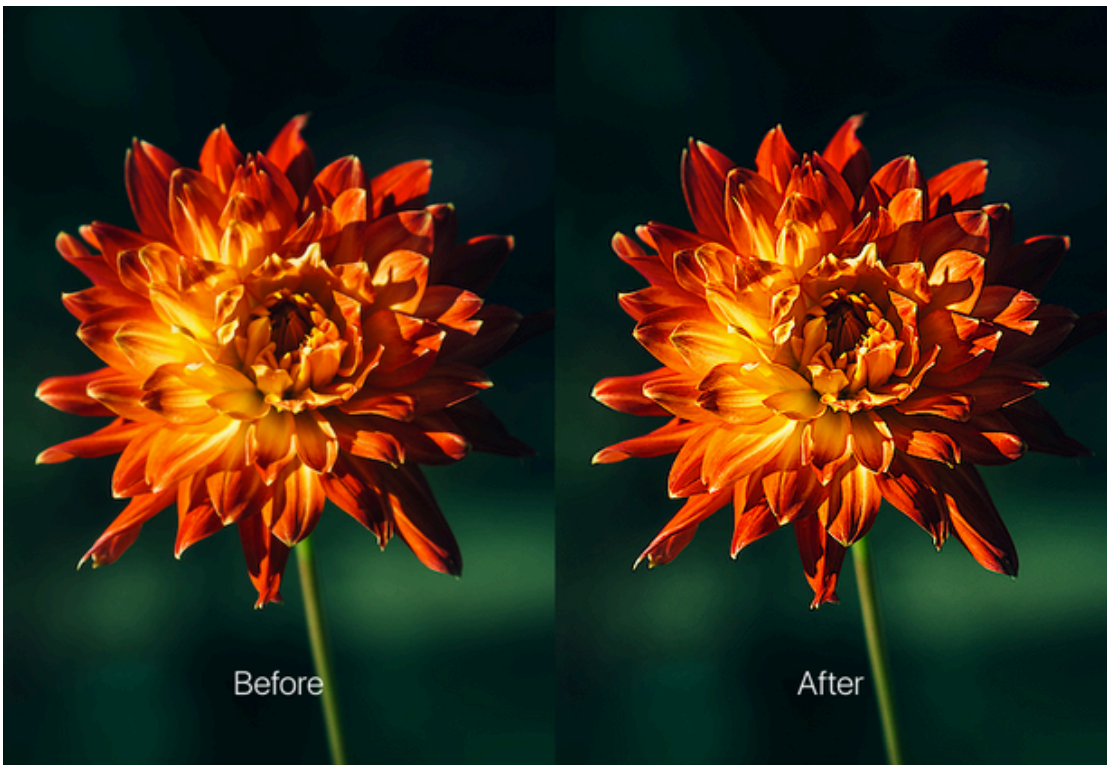

Рис. 1.3. Приклад підвищення різкості

- Корекція кольору та контрасту: Ця операція включає в себе налаштування кольору, яскравості, насиченості та інших параметрів зображення для покращення його візуальних характеристик (рис. 1.4).

Рис. 1.4. Приклад корекції кольору та контрасту

- Видалення артефактів та дефектів: Це включає усунення дефектів, таких як плями, променеві виблисків, тіні, відбиття тощо, які можуть з'явитися на зображенні через некоректне освітлення або неправильну обробку (рис. 1.5).

Рис. 1.5. Приклад усунення дефектів

- Кадрування та обрізка: Операції кадрування дозволяють вибирати певні частини зображення або обрізати непотрібні частини для покращення композиції чи усунення зайвої інформації (рис. 1.6).

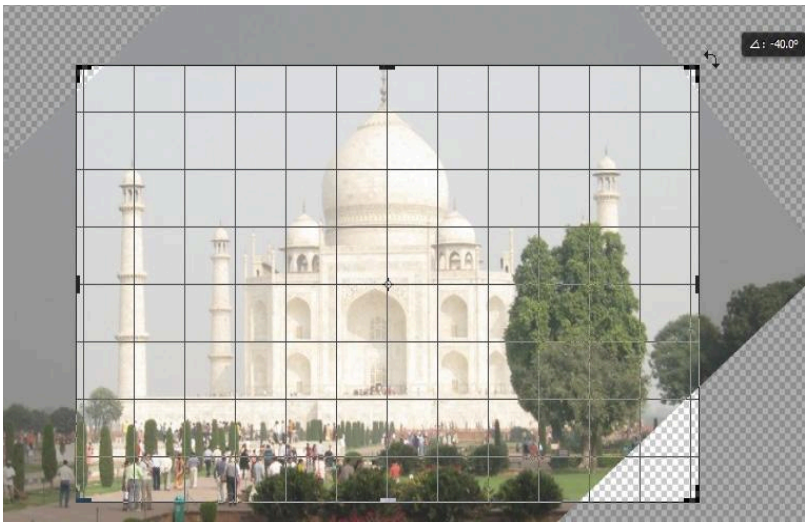

Рис. 1.6. Приклад кадрування

- Усунення спотворень та ретушування: Ця операція використовується для усунення будь-яких спотворень або дефектів, таких як розводи на лінзі камери, подряпини на фотографіях тощо (рис. 1.7).

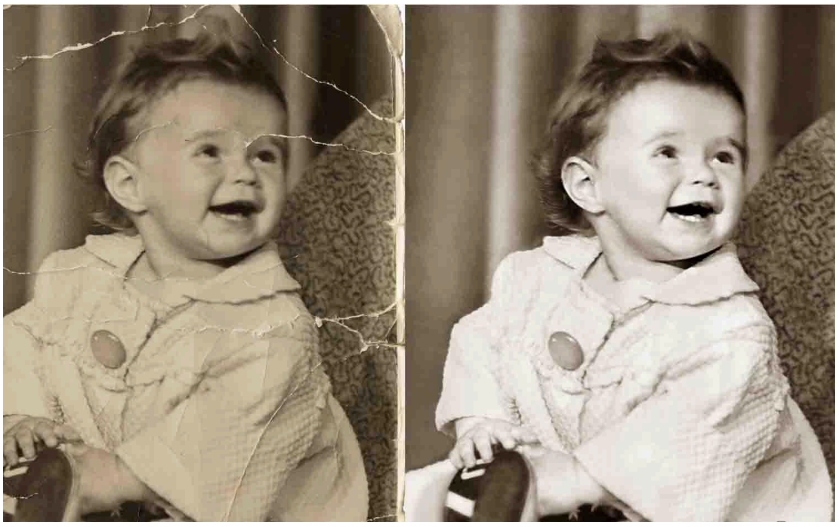

Рис. 1.7. Приклад усунення спотворень та дефектів

- Нормалізація та вирівнювання: Вона включає вирівнювання геометричних властивостей зображення, наприклад, виправлення перспективи, розмірів чи орієнтації.

Попередня обробка зображень важлива для підготовки зображення до подальшого аналізу, використання у веб-дизайні, друку, медицині, комп'ютерному зорі та в багатьох інших сферах. Вона допомагає покращити якість, чіткість та естетику зображення перед його подальшим використанням чи аналізом.

3. Сегментація - це процес розділення зображення на окремі частини або області з метою спрощення аналізу. Цей метод дозволяє виділити та визначити окремі області зображення, які мають схожі характеристики чи властивості, такі як кольори, текстури, контури чи інші візуальні особливості.

Існують різні методи сегментації зображень, серед яких основні включають:

- Порогова сегментація (Thresholding): Цей метод використовується для розділення зображення на різні області, використовуючи певний поріг яскравості або колірний поріг. Всі пікселі, що перевищують цей поріг, вважаються однією областю, в той час як інші пікселі належать до іншої області.
- Регіональна сегментація (Region-based Segmentation): Цей підхід використовує критерії схожості, такі як колір, текстура, яскравість тощо, для групування пікселів в одній області.
- Сегментація на основі країв (Edge-based Segmentation): Цей метод використовує виявлення контурів або країв об'єктів на зображенні для їх виділення в окремі сегменти.
- Кластеризація (Clustering): Використання алгоритмів кластеризації, таких як k-середніх або алгоритми машинного навчання, для групування схожих пікселів у класи чи кластери, що формують сегменти.
- Семантична сегментація (Semantic Segmentation): Це більш складний підхід, де кожен піксель на зображенні віднесений до певного

класу або категорії, такої як "людина", "автомобіль", "дерево" тощо, що дозволяє розуміти смислове значення кожного пікселя.

Сегментація має широкі застосування у багатьох галузях, таких як медицина (виділення органів на знімках MRI), комп'ютерний зір (розпізнавання об'єктів), автономні автомобілі (виявлення дорожніх знаків та перешкод) та багато інших. Цей процес допомагає аналізувати та розуміти зображення, робить його більш структурованим для подальшого використання в різних цілях.

4. Відновлення зображень - це процес відтворення чи відновлення пошкоджених, шумних або нечітких зображень. Він може використовувати різні методи, такі як фільтрація, реконструкція даних або використання статистичних моделей для відновлення втрачених даних.

5. Морфологічна обробка - це набір методів, які використовують математичні операції для аналізу та обробки геометричних структур на зображеннях, таких як розширення, звуження, відкриття та закриття.

6. Обробка в реальному часі - це обробка зображень, яка відбувається майже миттєво, зазвичай у відповідь на надходження нових даних. Це важливо для застосувань, де час є критичним фактором, наприклад, у медицині або автомобільній промисловості.

Ці принципи та методи є основою для розвитку різноманітних алгоритмів та моделей обробки зображень, які використовуються в широкому спектрі галузей, включаючи медицину, технології, розваги та багато інших.

1.2 Подання математичних моделей, що використовуються для опису пошкоджених зображень та методів їх відновлення. Розгляд моделей шуму, дефектів та інших артефактів

Математичні моделі відновлення зображень включають в себе різноманітні підходи та методи, спрямовані на опис та відновлення пошкоджених або забруднених зображень. Такі моделі базуються на математичних розрахунках та

статистичних аналізах для відновлення оригінальної інформації з пошкодженого або зашумленого вхідного зображення. Основні аспекти математичних моделей включають:

1. Моделювання шуму: Один із ключових аспектів моделювання відновлення зображень - це врахування шуму, який може бути присутнім в зображенні внаслідок різних факторів, таких як електронний шум у цифрових зображеннях, випадкові відхилення вимірювань у медичних зображеннях тощо. Моделі шуму допомагають визначити його характеристики та розробляти алгоритми для його фільтрації або видалення.

- Адитивний білий гаусівський шум (AWGN): Це один з найпоширеніших типів шуму у цифрових зображеннях. Математично це випадковий шум, який додається до кожного пікселя та має гауссівський розподіл. Моделі AWGN використовуються для розробки алгоритмів фільтрації шуму, таких як фільтри Гаусса чи медіанні фільтри.
- Пошкодження та шум у зображеннях медичного призначення: Медичні зображення часто піддаються різного роду шумам та артефактам під час процесу зйомки чи передачі даних. Математичні моделі для цих шумів дозволяють розробляти методи, які враховують специфіку медичних зображень при їх відновленні, наприклад, рентгенівських знімків чи зображень з магнітно-резонансної томографії (MRI).

2. Моделі дефектів та артефактів: Пошкодження зображень може включати різноманітні дефекти, такі як плями, розмиття, випадкові втрати даних, розриви чи інші артефакти, які можуть виникати під час зйомки, зберігання або передачі зображень. Математичні моделі дозволяють описати ці дефекти та розробити методи для їх відновлення чи усунення.

- Модель розмиття: Однією з основних проблем у зображеннях є розмиття. Математичні моделі розмиття описують характеристики цього процесу, дозволяючи розробляти алгоритми для відновлення чіткості та різкості зображень.

- Модель пошкоджень при стисненні зображень: Під час стиснення зображень артефакти можуть виникати через втрату даних або стиснення, що може призводити до втрати якості. Математичні моделі для цих артефактів допомагають розробляти методи відновлення зображень після стиснення.

3. Спектральні моделі: Вони використовують математичні методи для аналізу частот та спектрів зображення з метою відновлення втрачених або пошкоджених даних. Ці моделі дозволяють враховувати спектральні характеристики зображення під час його обробки та відновлення.

- Фур'є-аналіз: Використання трансформації Фур'є дозволяє аналізувати частотний склад зображень. Математичні моделі на основі фур'є-аналізу використовуються для виокремлення чи фільтрації шумів у спектральному представленні зображення.
- Утворення зображень за допомогою спектральних моделей: Ці моделі враховують спектральні характеристики зображення для відновлення втраченої інформації, що може бути корисно при відновленні змазаних чи пошкоджених даних.

4. Байєсівські моделі: Це моделі, які базуються на теорії ймовірностей та використовують байєсівський підхід для вирішення проблем відновлення зображень. Вони враховують апріорні знання та ймовірнісні розподіли для відновлення пошкоджених зображень.

5. Методи регуляризації: Вони використовуються для уникнення перенавчання та зменшення впливу шуму під час відновлення зображень. Методи регуляризації вводять обмеження чи штрафні функції для поліпшення якості відновленого зображення.

- Тіконовська регуляризація: Це метод регуляризації, що вводить обмеження на рішення задачі відновлення зображення, щоб уникнути перенавчання та зменшити вплив шумів. Вона використовується для

відновлення зображень за допомогою обмежень на їхні параметри чи структуру.

Ці математичні моделі відіграють ключову роль у розробці ефективних методів відновлення зображень, що дозволяють усувати шум, відновлювати пошкоджені ділянки та покращувати якість зображень у різних областях застосування, від медицини до технологій розпізнавання образів.

1.3 Аналіз та порівняння сучасних методів відновлення зображень, включаючи фільтрацію, реконструкцію, машинне навчання та глибинне навчання

Сучасні методи відновлення зображень представляють різноманітні підходи, які використовуються для відновлення пошкоджених, зашумлених або неповних зображень. Нижче розглянуто кожен метод та його основні напрямки.

Фільтрація є традиційним методом для відновлення зображень, спрямованим на видалення шуму та покращення якості зображення за допомогою різних фільтрів, таких як медіанний, Гауссівський, адаптивний тощо.

- Переваги: Простота реалізації, добре підходить для відновлення випадкового шуму.
- Недоліки: Обмежений у відновленні складних дефектів та деталей; може призводити до розмиття.

Реконструкція: Методи реконструкції використовуються для відновлення пропущених або пошкоджених частин зображення. Це може бути застосовано до медичних, астрономічних або археологічних зображень.

- Переваги: Ефективні для відновлення геометричної інформації, можуть бути використані у великих проектах обробки даних.
- Недоліки: Може бути складним у відновленні текстур та деталей, обчислювально витратний.

Машинне навчання: Методи машинного навчання використовуються для відновлення зображень, навчаючи моделі на великих наборах навчальних даних, які включають як вхідні, так і вихідні зображення.

- Переваги: Здатні відновлювати складні деталі і текстури; висока точність за наявності великої кількості даних для навчання.
- Недоліки: Потребує велику кількість обчислювальних ресурсів та даних для навчання.

Глибинне навчання: Глибинні нейронні мережі, зокрема згорткові нейронні мережі (CNN), використовуються для відновлення зображень. Вони можуть автоматично вивчати важливі функції та структури на зображеннях.

- Переваги: Досягає найкращих результатів у багатьох задачах відновлення зображень; здатність працювати з різними типами даних.
- Недоліки: Потребує велику кількість даних для навчання та обчислювальних ресурсів.

Порівняння сучасних методів відновлення зображень полягає у визначенні їхньої ефективності, швидкості та здатності до роботи з різноманітними типами пошкоджень чи шумів. Кожен з цих методів має свої переваги та недоліки, і вибір конкретного методу залежить від конкретної задачі та характеристик вхідних даних.

Розглянемо порівняльну оцінку різних підходів до відновлення зображень за різними критеріями:

1. Якість відновленого зображення:
   - Глибинне навчання: Зазвичай забезпечує найвищу якість відновлення, особливо відмінну від фільтрації та реконструкції. Глибокі нейронні мережі можуть відтворювати складні деталі та текстури.
   - Машинне навчання: Також здатне досягати високої якості, залежно від обсягу та якості навчальних даних. Методи на основі SVM, регресії тощо можуть давати гарні результати.
   - Реконструкція: Може відновлювати геометричну інформацію добре, але може втратити деталі та текстури.

- Фільтрація: Найменш висока якість, особливо для складних дефектів та деталей.

2. Швидкість обробки:
   - Фільтрація: Зазвичай найшвидший метод, оскільки використовує прості математичні операції.
   - Реконструкція: Швидкість може бути середньою, залежно від вимог задачі та складності алгоритму.
   - Машинне навчання: Вимагає значної обчислювальної потужності та може бути повільним на великих наборах даних.
   - Глибинне навчання: Зазвичай потребує великої кількості обчислювальних ресурсів, але може бути ефективним на спеціалізованому обладнанні, такому як GPU.
3. Витрати ресурсів:
   - Фільтрація: Має низькі витрати ресурсів, включаючи пам'ять та обчислювальну потужність.
   - Реконструкція: Вимагає середні ресурси в порівнянні з іншими методами.
   - Машинне навчання: Вимагає великої кількості пам'яті та обчислювальної потужності.
   - Глибинне навчання: Може бути найвимогливішим за ресурсами, особливо при тренуванні складних моделей.
4. Застосування в конкретних сферах:
   - Фільтрація: Застосовується в широкому спектрі завдань, де важлива швидкість та простота.
   - Реконструкція: Застосовується у сферах, де важлива відновлення геометричної інформації, таких як медицина або астрономія.
   - Машинне навчання: Застосовується в завданнях з багатьма даними та вимогами до точності, наприклад, у медичному зображення відновленні.

- Глибинне навчання: Використовується в багатьох сферах завдяки високій якості та здатності до автоматизації завдань.

Після аналізу цих критеріїв можна зробити наступні висновки:

- Глибинне навчання та машинне навчання найбільше підходять для завдань, де вимагається висока якість відновлення та точність.
- Фільтрація може бути вибором для задач з обмеженими ресурсами та вимогами до швидкості.
- Реконструкція може бути корисною для відновлення геометричних даних.
- Вибір методу повинен залежати від конкретних вимог та обмежень вашого проекту.

Додаткові аспекти, такі як: вибір оптимальних гіперпараметрів, робота з обмеженими ресурсами та застосування методів відновлення зображень у конкретних сферах, мають велике значення в дослідженні та практичному використанні цих методів. Давайте детальніше розглянемо кожен з цих аспектів:

1. Вибір оптимальних гіперпараметрів:
   - У багатьох методах відновлення зображень існують гіперпараметри, які потрібно налаштувати для досягнення найкращої ефективності. Це можуть бути параметри моделі (наприклад, глибини нейронних мереж, розмір фільтрів тощо), параметри оптимізації та інші.
   - Важливо провести експерименти та аналіз для визначення оптимальних значень гіперпараметрів, щоб отримати найкращі результати відновлення зображень.
2. Робота з обмеженими ресурсами:
   - У реальних умовах можуть існувати обмежені обчислювальні та пам'яткові ресурси. Важливо розглянути можливості оптимізації алгоритмів для роботи з обмеженими ресурсами.
   - Можливість використовувати легковажні моделі, удосконалені алгоритми обчислення або апаратне прискорення (наприклад, за

допомогою GPU) може бути важливою для практичного застосування методів відновлення зображень у реальних задачах.

3. Застосування в конкретних сферах:

- Методи відновлення зображень мають широкий спектр застосувань у різних сферах. Наприклад:
    - Медицина: Відновлення медичних зображень може включати в себе відновлення рентгенівських знімків, МРТ або снімків КТ для покращення діагностики.
    - Астрономія: Відновлення астрономічних зображень може допомогти у виявленні та аналізі об'єктів на небосхилі, зменшенні шуму та покращенні якості знімків.
    - Комп'ютерний зір: У сфері комп'ютерного зору методи відновлення зображень можуть бути використані для розпізнавання об'єктів, відновлення пошкоджених або розмитих зображень тощо.
- Для кожної конкретної сфери важливо розглянути специфічні вимоги, завдання та джерела даних, і відповідно до цього налаштувати методи відновлення.

Звісно, що в кожній зі згаданих сфер можуть бути унікальні виклики та особливості, і доцільно проводити дослідження та експерименти для адаптації методів відновлення до конкретних умов. Додаткові аспекти також включають управління витратами, забезпечення конфіденційності та етичні питання, які можуть виникати при використанні цих методів у реальних додатках.

1.4 Висновки до розділу

У першому розділі магістерської роботи було розглянуто теоретичні основи систем відновлення зображень, які становлять фундамент для подальшого розроблення практичної частини дослідження. Аналіз наукових праць, технічних

звітів та існуючих рішень у галузі відновлення зображень дозволив визначити ключові концепції, методи та технології, що застосовуються у цій сфері.

Основна увага була приділена класифікації та аналізу різних типів пошкоджень зображень, таких як шум, розмитість, втрата кольору, і як ці пошкодження впливають на якість зображень. Також були розглянуті різноманітні методи та алгоритми, які використовуються для виправлення цих дефектів, включаючи фільтрацію, використання нейронних мереж, та інші методи глибокого навчання.

У розділі особливо підкреслювалася роль штучного інтелекту та машинного навчання у сучасних системах відновлення зображень. Аналіз сучасних підходів до використання конволюційних нейронних мереж, генеративно-змагальних мереж та інших алгоритмів глибокого навчання підтвердив їхню ефективність у рішенні складних завдань обробки зображень.

Детальний огляд літератури та аналіз існуючих досліджень виявив основні напрямки розвитку та виклики, що стоять перед галуззю відновлення зображень. Зокрема, було виявлено потребу в подальшому удосконаленні алгоритмів обробки для підвищення точності та швидкості відновлення, а також у розширенні можливостей систем з метою їх застосування у різних областях.

У підсумку, перший розділ заклав міцний теоретичний фундамент для подальшої практичної розробки системи відновлення зображень. Розуміння основних принципів, методів та проблем, що існують у цій області, є необхідним для успішного вирішення поставлених завдань у наступних розділах роботи.

РОЗДІЛ 2
МЕТОДИ ТА ПІДХОДИ ОЦІНКИ ЯКОСТІ ВІДНОВЛЕННЯ ЗОБРАЖЕНЬ

2.1 Визначення метрик для оцінки якості відновлення (PSNR, SSIM, MSE, MAE тощо)

Метрики оцінки якості відновлення зображень, такі як PSNR, SSIM, MSE та MAE, використовуються для вимірювання рівня відновлення зображення порівняно з оригіналом. Ось їх основні визначення:

1. PSNR (Peak Signal-to-Noise Ratio): Ця метрика вимірює співвідношення між максимально можливою потужністю сигналу (якість зображення) та впливом шуму, що спотворює його представлення. Вищий PSNR зазвичай означає кращу якість. Вона вимірюється в децибелах (дБ)

$$\text{PSNR}=10\cdot log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{1.1}$$

де MAX - максимальне можливе значення пікселя в зображенні. Наприклад, для 8-бітного зображення це 255. MSE - середньоквадратична помилка між оригінальним та відновленим зображеннями. Високий PSNR зазвичай означає меншу помилку та кращу якість відновлення. PSNR більше використовується для оцінки якості стиснених зображень.

2. SSIM (Structural Similarity Index): Ця метрика оцінює видимі зміни між двома порівняльними зображеннями з точки зору яскравості, контрасту та структури. SSIM вимірюється від 0 до 1, де 1 означає ідеальну схожість. Формула складна, охоплює порівняння яскравості, контрасту та структури між двома зображеннями. Включає середні значення, дисперсії та коваріацію між двома

зображеннями. SSIM більш чутлива до видимих змін, ніж PSNR. Індекс близький до 1 вказує на високу ступінь схожості між зображеннями.

3. MSE (Mean Squared Error): Вона вимірює середньоквадратичну помилку між двома зображеннями. MSE обчислює середнє значення квадратів різниць між відповідними пікселями двох зображень. Нижчий MSE означає меншу помилку, що є кращим.

$$\text{MSE}=\frac{1}{min}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j)-K(i,j)]^2 \quad (1.2)$$

де I(i,j) - піксель оригінального зображення. K(i,j) - піксель відновленого зображення. m,n - розміри зображення. MSE обчислює середнє значення квадратів різниць між пікселями. Нижчий MSE свідчить про меншу помилку.

4. MAE (Mean Absolute Error): Ця метрика подібна до MSE, але замість використання квадратів різниць, MAE вимірює середнє абсолютне значення цих різниць. Як і MSE, нижчий MAE вказує на вищу якість відновлення зображення.

$$\text{MAE} - \frac{1}{min}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}|I(i,j)-K(i,j)| \quad (1.3)$$

подібні до MSE, але замість квадрату різниці використовується абсолютне значення різниці.

Ці метрики використовуються для аналізу якості різних методів обробки зображень, включаючи стиснення, відновлення, посилення та інші операції з обробки зображень.

2.2 Використання об'єктивних та суб'єктивних методів оцінки якості

Оцінка якості зображення за допомогою об'єктивних і суб'єктивних методів є ключовою частиною багатьох сфер, де важливим є розуміння та підтримка високої якості візуальної інформації. Ось декілька причин, чому це важливо:

1. Розвиток та оцінка технологій обробки зображень: Ці методи використовуються для оцінки різних технологій обробки зображень, таких як стиснення, шумоподавлення, підвищення роздільної здатності тощо. Вони дозволяють розробникам розуміти, наскільки добре їх алгоритми працюють у реальних умовах.
2. Забезпечення якості у мультимедіа: У галузях, таких як кіноіндустрія, телебачення та онлайн-відео, якість зображення є критично важливою. Тут оцінка якості допомагає забезпечити, що контент відповідає певним стандартам перед його розповсюдженням.
3. Поліпшення користувацького досвіду: З розширенням цифрових медіа та інтерактивних додатків, якість зображення стає ключовим фактором у забезпеченні задовільного користувацького досвіду. Наприклад, у відеоіграх та віртуальній реальності висока якість зображення є важливою для імерсійного досвіду.
4. Наукові дослідження: В медицині, астрономії, супутникових знімках та інших галузях науки, точна оцінка якості зображення необхідна для точної інтерпретації візуальних даних.
5. Нормативні та стандартизовані питання: Оцінка якості зображення також важлива для розробки та впровадження стандартів у галузі зображень та відео.

Використання як об'єктивних, так і суб'єктивних методів дозволяє отримати повне уявлення про якість зображення. Об'єктивні методи забезпечують стандартизовані, вимірювані показники, в той час як суб'єктивні методи дозволяють зрозуміти, як людське сприйняття впливає на оцінку якості.

Тож, об'єктивні методи використовують математичні та статистичні моделі для вимірювання якості зображення. Вони засновані на чітко визначених алгоритмах і не залежать від людського сприйняття. Ось декілька прикладів:

1. PSNR (Peak Signal-to-Noise Ratio): Як вже згадувалось, це один з найпоширеніших методів для оцінки якості зображення, особливо у контексті стиснення (рис. 2.1)

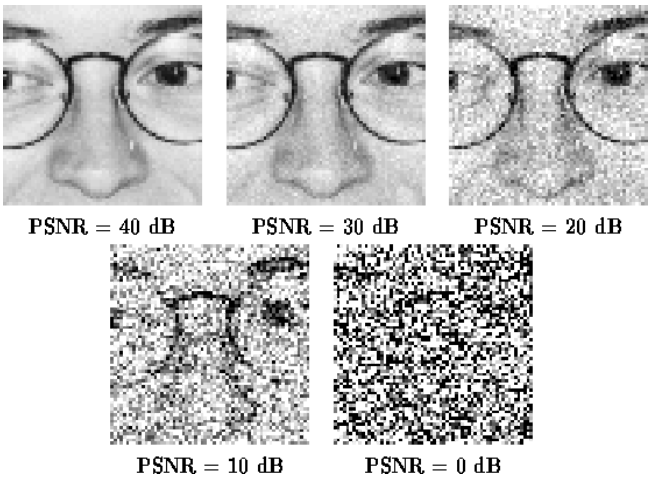

Рис. 2.1. Приклад зменшення PSNR та втрата якості

2. SSIM (Structural Similarity Index): Ця метрика оцінює видимі зміни у структурі, яскравості та контрасті зображення (рис. 2.2)

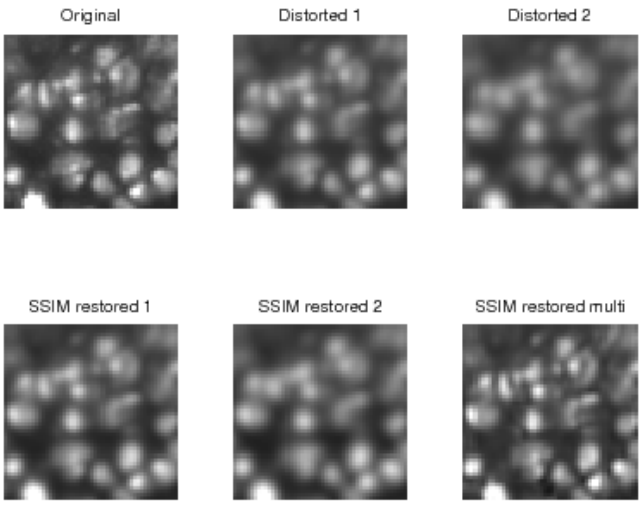

Рис. 2.2. Приклад застосування метрики SSIM

3. VIF (Visual Information Fidelity): Ця метрика намагається імітувати сприйняття якості зображення людським зором, враховуючи втрату інформації. Нижче наведено приклад застосування метрики з такими значеннями:

a - Reference VIF=1.0;

b - Contrast enhanced. VIF=1.10;

c - Blurred VIF=0.07;

d - JPEG compressed VIF=0.10 (рис. 2.3)

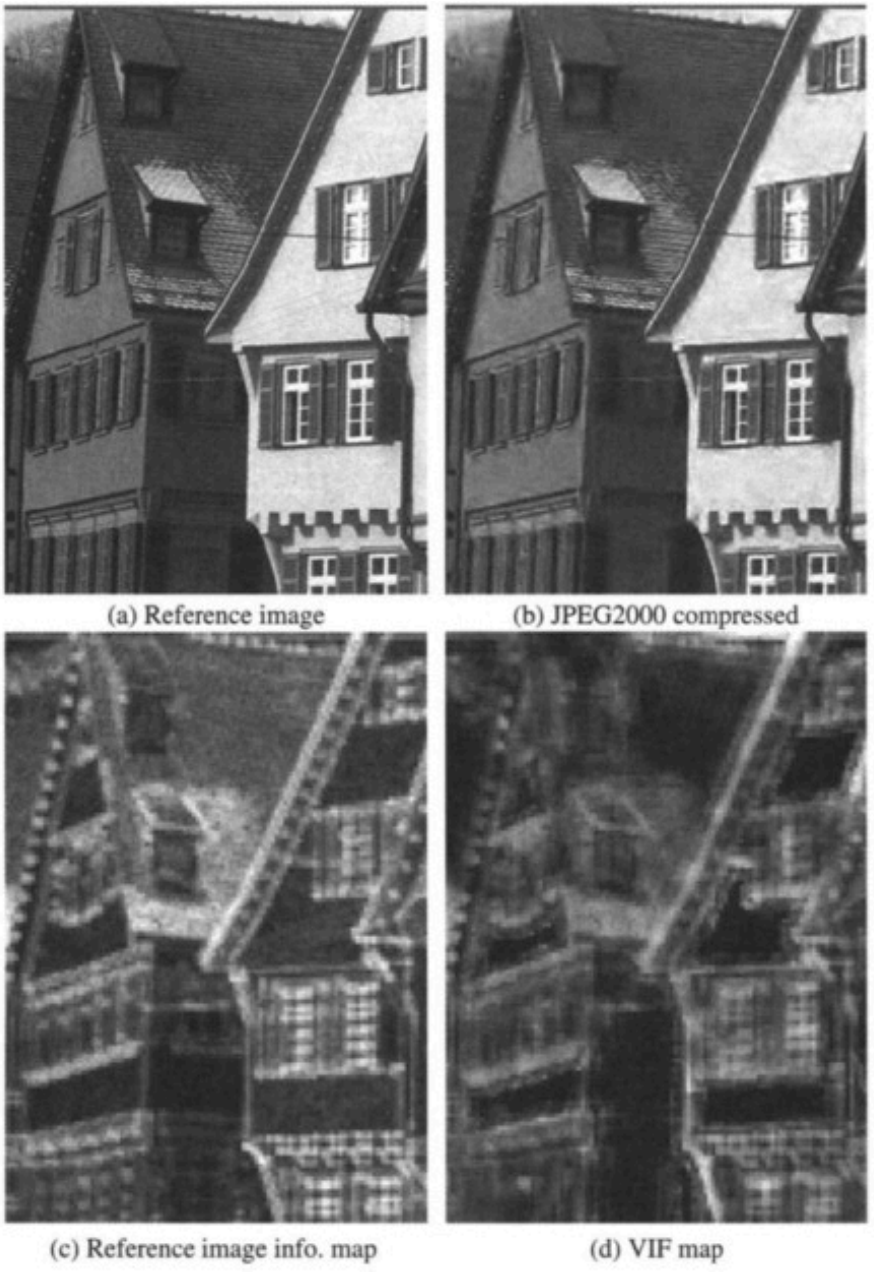

Рис. 2.3. Приклад застосування метрики VIF

Суб'єктивні методи оцінки залежать від людського сприйняття та враження. Вони включають:

1. Суб'єктивні тестування: Люди оцінюють якість зображення, використовуючи різні шкали, наприклад, від "погано" до "відмінно". Це можуть бути організовані тестування з великою кількістю учасників.
2. MOS (Mean Opinion Score): Це середнє значення всіх оцінок, наданих учасниками суб'єктивних тестів. Ця шкала часто використовується для оцінки загальної якості зображення або відео.

Порівняння об'єктивних та суб'єктивних методів оцінки якості зображення демонструє унікальні переваги та обмеження кожного підходу.

Об'єктивні Методи:

- Точність: Забезпечують консистентність і повторюваність, але можуть не повністю враховувати всі аспекти людського сприйняття якості.
- Ресурси: Ці методи можуть бути автоматизовані та вимагають менше часу та ресурсів для виконання.
- Застосування: Широко використовуються в промисловості для стандартизації та оцінки технологій обробки зображень. Часто застосовуються у тестуванні, розробці продуктів та наукових дослідженнях.
- Об'єктивність: Відсутність суб'єктивних впливів забезпечує чіткість та однозначність результатів.

Суб'єктивні Методи:

- Точність: Краще відображають людське сприйняття якості зображення, включаючи естетичні та емоційні аспекти.
- Ресурси: Вимагають значної участі людей і часу для проведення тестів, що робить їх більш ресурсоємними.

- Застосування: Ідеальні для визначення уподобань користувачів і оцінки візуального досвіду в реальних умовах. Використовуються для валідації та удосконалення об'єктивних методів.
- Суб'єктивність: Результати можуть варіюватися в залежності від індивідуальних переваг, культурних відмінностей та особистого досвіду учасників.

Отож, можна зробити такі висновки:

- Взаємодоповнюваність: Обидва підходи часто використовуються разом для отримання більш повного розуміння якості зображення. Об'єктивні методи надають кількісні дані, в той час як суб'єктивні методи допомагають зрозуміти, як ці дані впливають на сприйняття людиною.
- Контекст застосування: Вибір між об'єктивними та суб'єктивними методами часто залежить від конкретного застосування, цілей оцінки та доступних ресурсів.

2.3 Розгляд підходів до порівняльної оцінки ефективності різних методів відновлення

Порівняння об'єктивних та суб'єктивних методів оцінки якості зображення демонструє унікальні переваги та обмеження кожного підходу.

Порівняльна оцінка ефективності різних методів відновлення зображень включає декілька ключових підходів. Вони забезпечують об'єктивне порівняння між різними алгоритмами та техніками, що дозволяє визначити, які методи краще підходять для певних задач. Ось основні підходи:

1. Використання стандартизованих датасетів
   - Підхід: Використання загальновизнаних тестових наборів даних (датасетів) з різними типами зображень.
   - Переваги: Це дозволяє порівнювати різні методи в однакових умовах.

- Приклади датасетів: Set5, Set14, BSDS500 для задач супер-роздільної здатності або денойзингу.

2. Об'єктивні метрики оцінки якості
   - Підхід: Використання об'єктивних метрик, таких як PSNR, SSIM, MSE для кількісної оцінки ефективності.
   - Переваги: Забезпечують точні та повторювані результати.
   - Застосування: Ці метрики використовуються для оцінки якості відновлених зображень порівняно з оригінальними.
3. Суб'єктивна оцінка якості
   - Підхід: Залучення людей для оцінки якості відновлення зображень (наприклад, через Mean Opinion Score (MOS)).
   - Переваги: Враховують людське сприйняття якості, що може бути важливим для деяких застосувань.
   - Застосування: Особливо корисно в галузях, де сприйняття якості зображення відіграє ключову роль (наприклад, фотографія, кіно).
4. Порівняння часу обробки та ресурсоємності
   - Підхід: Оцінка швидкості обробки та потреб у обчислювальних ресурсах різних методів.
   - Переваги: Важливо для розуміння практичної застосовності методів в реальному часі та на обмежених платформах.
   - Застосування: Особливо актуально для мобільних додатків, веб-сервісів та вбудованих систем.

При порівняльній оцінці ефективності різних методів відновлення зображень, фахівці зазвичай звертаються до комбінації об'єктивних та суб'єктивних методів аналізу. Використання стандартизованих датасетів дозволяє порівнювати методи в однакових умовах, надаючи однорідну базу для тестування. Об'єктивні метрики, такі як PSNR, SSIM, або MSE, забезпечують кількісні дані про якість відновлених зображень, дозволяючи оцінити технічну точність методів. Водночас, суб'єктивна оцінка за допомогою людських спостерігачів, таких як використання шкали Mean

Opinion Score, дає змогу зрозуміти, наскільки добре відновлені зображення сприймаються людським оком, особливо у контекстах, де важлива естетика або специфічне візуальне сприйняття.

Крім якості зображення, важливим аспектом є також аналіз швидкості обробки та ресурсоємності методів. Це особливо актуально для застосувань у реальному часі або на пристроях з обмеженими обчислювальними ресурсами, таких як мобільні пристрої або вбудовані системи. Також важливо оцінювати універсальність методів, тобто їхню здатність ефективно працювати з різноманітними типами зображень та в умовах різних спотворень. Це дає змогу визначити, наскільки гнучкими та адаптивними є методи, і чи можуть вони бути застосовані в широкому спектрі ситуацій.

2.4 Висновки до розділу

У Розділі 2 було розглянуто важливість та методологію оцінки якості відновлення зображень. Перш за все, було описано ключові метрики якості, такі як PSNR, SSIM, MSE та MAE, які забезпечують об'єктивну оцінку якості відновлення. Особлива увага була приділена розумінню того, як ці метрики вимірюють різні аспекти зображення, такі як контраст, яскравість та структура.

Далі було обговорено, як об'єктивні та суб'єктивні методи оцінки співвідносяться та взаємодоповнюють один одного. Підкреслено, що об'єктивні метрики не завжди відображають людське сприйняття якості, в той час як суб'єктивні оцінки можуть дати глибше розуміння сприйняття якості зображень кінцевими користувачами.

Розділ також підкреслив важливість порівняльної оцінки різних методів відновлення зображень. Окреслено, що використання стандартизованих датасетів, різних метрик якості та суб'єктивних оцінок може надати цінну інформацію про ефективність та надійність різних підходів до відновлення зображень. Особливу

увагу було приділено також швидкості обробки та вимогам до обчислювальних ресурсів, що є ключовими для реального застосування відновлених зображень.

У підсумку, цей розділ наголосив на складності та багатогранності задачі оцінки якості відновлення зображень, демонструючи потребу в комплексному підході, який враховує як технічні аспекти, так і сприйняття користувачів.

РОЗДІЛ 3
ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ СИСТЕМ РЕКОНСТРУКЦІЇ ЗОБРАЖЕНЬ

3.1 Алгоритми та програмні реалізації систем відновлення зображень

Існуючі алгоритми та програмні реалізації систем відновлення зображень охоплюють широкий спектр методів, які вирішують різні завдання, такі як підвищення роздільної здатності, денойзинг (усунення шуму), та реставрація зображень.

3.1.1 Алгоритми супер-роздільної здатності

Ці алгоритми мають на меті підвищити роздільну здатність зображень, зберігаючи при цьому їхню якість.

- SRCNN (Super-Resolution Convolutional Neural Network): Використовує конволюційні нейронні мережі для покращення роздільної здатності.
- ESRGAN (Enhanced Super-Resolution Generative Adversarial Networks): Використовує генеративно-змагальні мережі для створення високоякісних високороздільних зображень.

Створення повноцінної реалізації алгоритмів супер-роздільної здатності, таких як SRCNN або ESRGAN, вимагає значного обсягу коду та використання спеціалізованих бібліотек глибокого навчання. Однак, нижче наведено базовий приклад реалізації SRCNN за допомогою Python та фреймворку глибокого навчання, наприклад, TensorFlow. Звертаю увагу, що для повноцінної роботи такого коду необхідний встановлений TensorFlow та набір даних для тренування.

Ось базовий приклад реалізації SRCNN:

```
import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Conv2D
```

```
def srcnn_model():
    input_shape = (None, None, 1)
    # Припустимо, зображення в градаціях сірого
    inputs = Input(shape=input_shape)
    # Перший шар: витягування ознак
    x = Conv2D(64, (9, 9), activation='relu',
padding='same')(inputs)
    # Другий шар: активація
    x = Conv2D(32, (1, 1), activation='relu', padding='same')(x)
    # Третій шар: реконструкція зображення
    x = Conv2D(1, (5, 5), padding='same')(x)
    model = Model(inputs=inputs, outputs=x)
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model
# Створення моделі SRCNN
srcnn = srcnn_model()
srcnn.summary()
```

Цей код створює базову архітектуру SRCNN з трьома конволюційними шарами. Перший шар використовує ядро 9x9 для витягування ознак, другий шар з ядром 1x1 діє як активаційний шар, а третій шар з ядром 5x5 виконує реконструкцію зображення.

Щодо ESRGAN, його реалізація значно складніша через використання генеративно-змагальних мереж, і часто вимагає більш складних підходів та більшого обсягу коду. Реалізація ESRGAN, як правило, здійснюється в рамках великих проектів з обробки зображень і займає значно більше місця та ресурсів, ніж може бути забезпечено в рамках цього відповіді. Зазвичай, для реалізації ESRGAN використовуються такі фреймворки, як TensorFlow або PyTorch, і це включає створення двох окремих мереж - генеративної та дискримінативної, які навчаються у змагальному режимі.

3.1.2 Алгоритми денойзингу зображень

Ці методи спрямовані на усунення шуму з зображень, зберігаючи при цьому важливі деталі.

- DnCNN: Глибокі нейронні мережі для видалення шуму з зображень.

- Non-Local Means: Традиційний метод, який використовує не локальні середні для видалення шуму.

Створення базової реалізації DnCNN (Deep Convolutional Neural Networks for Image Denoising) можна зробити за допомогою Python та фреймворку глибокого навчання, такого як TensorFlow. Ось приклад простої реалізації DnCNN:

```
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, Activation
from tensorflow.keras.models import Model
def DnCNN():
      input_shape = (None, None, 1)
      # Припустимо, зображення в градаціях сірого
      inputs = Input(shape=input_shape)
      # Перший шар: Conv + ReLU
      x = Conv2D(64, (3, 3), padding='same')(inputs)
      x = Activation('relu')(x)
      # Середні шари: Conv + BatchNorm + ReLU
      for _ in range(15):
      x = Conv2D(64, (3, 3), padding='same')(x)
      x = tf.keras.layers.BatchNormalization()(x)
      x = Activation('relu')(x)
      # Останній шар: Conv
      x = Conv2D(1, (3, 3), padding='same')(x)
model = Model(inputs=inputs, outputs=x)
model.compile(optimizer='adam', loss='mean_squared_error')
return model
# Створення моделі DnCNN
dncnn = DnCNN()
dncnn.summary()
```

Цей код створює DnCNN модель з використанням TensorFlow. Модель має один вхідний шар, багато середніх шарів, кожен з яких складається з Convolutional Layer, Batch Normalization та ReLU активації, та один вихідний шар. Оптимізатором виступає Adam, а функцією втрат - середньоквадратична помилка (mean squared error).

Ця модель є лише базовим прикладом і призначена для ілюстрації структури DnCNN. Для реального застосування вам потрібно буде тренувати модель на великій кількості зображень із шумом та без нього, щоб вона ефективно видаляє шум.

Щодо Non-Local Means, цей алгоритм можна легко реалізувати за допомогою бібліотеки OpenCV:

```
python
import cv2
def non_local_means_denoising(image, h=10, window_size=21, search_size=21):
    """
    Функція для денойзингу зображення за допомогою алгоритму Non-Local Means.
    :param image: Вхідне зображення у форматі NumPy array.
    :param h: Параметр фільтрації, що контролює силу денойзингу.
    :param window_size: Розмір вікна для обчислення ваг.
    :param search_size: Розмір вікна пошуку для порівняння патчів.
    :return: Зображення після денойзингу.
    """
    denoised_image = cv2.fastNlMeansDenoising(image, None, h, window_size, search_size)
    return denoised_image
# Припустимо, ми маємо зображення, завантажене як NumPy масив
# image = cv2.imread('path_to_image.jpg', cv2.IMREAD_GRAYSCALE)
# Денойзинг зображення
# denoised_image = non_local_means_denoising(image)
# Щоб відобразити зображення, можна використовувати cv2.imshow('Denoised Image', denoised_image)
# та cv2.waitKey(0) для очікування натискання клавіші перед закриттям вікна
```

Цей код демонструє використання алгоритму Non-Local Means для денойзингу зображень з допомогою OpenCV. Він ефективний для усунення шуму, при цьому зберігаючи важливі деталі зображення.

3.1.3 Алгоритми реставрації зображень

Ці алгоритми використовуються для відновлення деталей та якості пошкоджених або старих зображень.

- Inpainting Algorithms: Для заповнення відсутніх або пошкоджених частин зображення.

- Deep Learning-Based Methods: Глибокі нейронні мережі, що використовуються для відновлення та поліпшення якості зображень.

Алгоритми реставрації зображень, зокрема інпейнтинг, можна реалізувати за допомогою OpenCV. Один з найпростіших методів інпейнтингу - це техніка, яка використовує алгоритми, засновані на телеграфному рівнянні. OpenCV надає функцію `cv2.inpaint`, яка може використовуватися для таких цілей. Нижче наведено приклад простого інпейнтингу зображення:

```
python
import cv2
import numpy as np
def inpaint_image(image, mask, inpaint_radius=3,
inpaint_method=cv2.INPAINT_TELEA):
    """
    Функція для реставрації зображення за допомогою інпейнтингу.
    :param image: Пошкоджене зображення у форматі NumPy array.
    :param mask: Маска пошкоджених областей (білі пікселі вказують на
пошкоджені області).
    :param inpaint_radius: Радіус інпейнтингу.
    :param inpaint_method: Метод інпейнтингу, наприклад
cv2.INPAINT_TELEA або cv2.INPAINT_NS.
    :return: Відновлене зображення.
    """
    inpainted_image = cv2.inpaint(image, mask, inpaint_radius,
inpaint_method)
    return inpainted_image
# Завантаження зображення
# image = cv2.imread('path_to_damaged_image.jpg')
# mask = cv2.imread('path_to_mask_image.jpg', cv2.IMREAD_GRAYSCALE)
# Реставрація зображення
# restored_image = inpaint_image(image, mask)
# cv2.imshow('Restored Image', restored_image)
# cv2.waitKey(0)
# cv2.destroyAllWindows()
```

Цей код призначений для демонстрації використання функції інпейнтингу в OpenCV. Вам потрібно буде забезпечити власні шляхи до зображення та маски, що представляє області, які потребують реставрації.

Реалізація глибокого навчання для реставрації зображень, зокрема використання глибоких нейронних мереж, є значно складнішою і зазвичай вимагає роботи з фреймворками глибокого навчання, такими як TensorFlow або PyTorch. Нижче наведено базовий приклад створення моделі глибокого навчання для реставрації зображень. Цей приклад не є повністю функціональним і призначений для демонстрації структури моделі:

```
python
import tensorflow as tf
from tensorflow.keras.layers import Input, Conv2D, Activation
from tensorflow.keras.models import Model
def image_restoration_model():
    input_shape = (None, None, 3)
     # Припустимо, кольорові зображення
    inputs = Input(shape=input_shape)
    # Перший шар: Conv + ReLU
    x = Conv2D(64, (3, 3), padding='same')(inputs)
    x = Activation('relu')(x)
    # Середні шари: Conv + ReLU
    for _ in range(5):
        x = Conv2D(64, (3, 3), padding='same')(x)
        x = Activation('relu')(x)
    # Останній шар: Conv
    x = Conv2D(3, (3, 3), padding='same')(x)
    model = Model(inputs=inputs, outputs=x)
    model.compile(optimizer='adam', loss='mean_squared_error')
    return model
# Створення моделі для реставрації зображень
restoration_model = image_restoration_model()
restoration_model.summary()
```

Ця модель є лише ілюстрацією та потребує додаткових налаштувань та тренування на великій кількості даних для ефективного відновлення зображень. Модель використовує декілька конволюційних шарів із активацією ReLU та вихідний шар, який видає зображення з трьома кольоровими каналами.

3.2 Програмні реалізації

Програмні реалізації цих алгоритмів відновлення зображень зазвичай базуються на сучасних фреймворках глибокого навчання та обробки зображень.

1. Фреймворки глибокого навчання:
    - TensorFlow та PyTorch: Ці фреймворки широко використовуються для розробки та навчання моделей глибокого навчання, що включають SRCNN, ESRGAN, DnCNN.
    - Keras: Високорівневий інтерфейс для TensorFlow, що полегшує розробку та експериментування з нейронними мережами.
2. Бібліотеки обробки зображень:
    - OpenCV: Популярна бібліотека для обробки зображень, яка включає традиційні методи, такі як Non-Local Means.
    - Scikit-image: Бібліотека обробки зображень для Python, яка надає доступ до широкого спектру алгоритмів.
3. Інструменти для Реставрації Зображень:
    - Спеціалізоване програмне забезпечення, яке інтегрує різні алгоритми для реставрації та відновлення зображень, часто з використанням методів глибокого навчання.

Існуючі алгоритми та програмні реалізації систем відновлення зображень охоплюють широкий спектр методів від традиційних до передових технологій, заснованих на глибокому навчанні. Вони відіграють ключову роль у багатьох галузях, де важлива якість візуальної інформації, і постійно розвиваються, пропонуючи все більш точні та ефективні рішення для відновлення та поліпшення зображень

Таким чином, інтегрований підхід, який поєднує об'єктивні метрики, суб'єктивні оцінки та аналіз продуктивності, є ключовим для всебічного розуміння та порівняння ефективності різних методів відновлення зображень.

3.3 Висновки до розділу

У даному розділі було проведено глибокий аналіз існуючих методів та програмного забезпечення, які використовуються у сфері відновлення зображень. Починаючи з розгляду алгоритмів супер-роздільної здатності, таких як SRCNN та ESRGAN, було демонстровано, як сучасні методи глибокого навчання здатні значно покращити якість зображень. Ці технології відкривають нові можливості у відновленні деталей та текстур у зображеннях із низькою роздільною здатністю.

Далі розділ зосереджується на алгоритмах денойзингу, таких як DnCNN та Non-Local Means, підкреслюючи їхню здатність усувати шум із зображень, зберігаючи при цьому їхні ключові деталі. Це особливо важливо у контексті збереження аутентичності зображень після обробки.

Також було розглянуто алгоритми реставрації зображень, включно з інпейнтингом та методами, що базуються на глибокому навчанні. Ці методи виявилися ефективними у відновленні пошкоджених або старих зображень, заповнюючи прогалини та відновлюючи втрачені елементи.

Програмні реалізації цих методів, особливо використання фреймворків глибокого навчання та обробки зображень, таких як TensorFlow, PyTorch та OpenCV, відіграють ключову роль у втіленні теоретичних алгоритмів у практичні інструменти. Це вказує на важливість інтегрованого підходу, який об'єднує різноманітні техніки та інструменти для досягнення оптимальних результатів.

Висновок розділу підкреслює, що постійний розвиток та вдосконалення методів відновлення зображень відкриває нові можливості для покращення якості візуальної інформації у різних сферах застосування, від медицини до цифрового мистецтва. Сучасні програмні рішення дозволяють ефективно вирішувати складні задачі відновлення зображень, підвищуючи їхню цінність та доступність.

РОЗДІЛ 4
РОЗРОБКА АЛГОРИТМУ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВІДНОВЛЕННЯ ЗОБРАЖЕННЯ

4.1 Вимоги до відновлення зображень для їхньої ефективної роботи

Проблема відновлення зображень стосується викликів і складнощів, які виникають під час спроб відновити, покращити або модифікувати зображення, особливо якщо вони пошкоджені, розмиті, мають низьку роздільну здатність або страждають від інших візуальних недоліків. Основні аспекти цієї проблеми включають:

- Відновлення роздільної здатності: Підвищення якості зображень з низькою роздільною здатністю, що вимагає складних алгоритмів для відтворення деталей, яких може не бути в оригінальному зображенні.
- Видалення шуму: Зниження або усунення візуального шуму, який може бути викликаний різними факторами, включаючи погане освітлення, низьку якість камери, або передачу даних.
- Відновлення пошкоджених зображень: Реставрація зображень, що постраждали від фізичних пошкоджень, старіння, водних знаків або інших форм втручання.
- Корекція кольору: Виправлення або налаштування кольорового балансу, яскравості і контрастності, особливо для старих або блідих зображень.
- Етичні та правові питання: Відновлення зображень може порушувати приватність або авторські права, особливо якщо воно використовується без дозволу власників або для маніпуляцій.
- Обмеження технологій: Незважаючи на стрімкий розвиток технологій обробки зображень, деякі завдання все ще залишаються надто складними або часом неможливими для автоматизованого вирішення.

Ці виклики залучають широкий спектр технологій, включаючи машинне навчання, комп'ютерний зір, цифрову обробку зображень, а також вимагають глибокого розуміння візуальної естетики та контенту зображень.

Інтерфейс користувача повинен бути інтуїтивним для досягнення якомога кращої продуктивності роботи системи. Користувач повинен бути ознайомлений з інструкціями і головними вимогами до зображення. Після передачі зображення веб-сервісу очікування на результат повинне мати якнайменшу тривалість, а отже метод відновлення зображення повинен працювати якісно та швидко, але й давати змогу клієнту вносити корективи. Також повинна бути реалізована перевірка зображень.

4.2 Вимоги до програмного рішення та запропонована архітектура проектованої системи

4.2.1 Основні вимоги до програмного рішення

Для вирішення проблеми якнайефективніше було розглянуто комплексний підхід для досягнення високого рівня технічних вимог, а також вибору сервісу штучного інтелекту для забезпечення гнучкості та високої швидкодії. Оскільки веб додаток отримуватиме всю важливу інформацію з віддаленого веб-сервісу доступу за допомогою ключів доступу користувача має здійснюватись саме серверною стороною. Передача даних від клієнта до сервера здійснюватиметься за допомогою протоколу передачі даних HTTPS, що забезпечує захист від прослуховування, а дані відновлюваного зображення користувача будуть додатково зашифровані для збереження анонімності.

Для надійного зберігання зображення користувача слід використати віддалений простір. А також враховуючи масштабованість - це рішення повинно забезпечувати високу доступність даних, гнучкість управління ресурсами та зручність інтеграції з іншими компонентами системи з якою буде взаємодіяти лише веб-сервіс. При реалізації відновлення зображення користувача слушним буде

використання стороннього продукту, який був розроблений спеціалізовано для вирішення цієї задачі через складність реалізації алгоритмів відновлення зображення та високі вимоги до апаратного забезпечення зумовлене складністю обчислювальних операцій та високими вимоги до швидкодії системи. Це також допоможе перенести обов’язки обробки зображення з сервера, що вирішує проблему з необхідністю використання та обслуговування накопичувачів пам’яті та підвищує рівень безпеки, адже навіть при отриманні несанкціонованого доступу до сервера, зловмисники не отримають доступу до зображень користувачів. Оскільки усі готові для використання сервіси, які мають ці властивості є платними, було обрано найбільш ефективний та економічно вигідний варіант, який не накладатиме обмежень при потенційному рості системи.

Розширемо більш детальніше по пунктах:

1. Технічні Вимоги
    a. Висока Обчислювальна Потужність:
        - Штучний інтелект, особливо глибокі нейронні мережі, вимагає значних обчислювальних ресурсів для ефективного аналізу та обробки зображень.
    b. Оптимізовані Алгоритми Штучного Інтелекту:
        - Важливо використовувати ефективні та оптимізовані алгоритми МІ, які можуть швидко обробляти великі об'єми даних без втрати якості відновлення зображень.
    c. Надійне Зберігання Даних:
        - Потрібні масштабовані та безпечні рішення для зберігання зображень та оброблених даних.
    d. Ефективність Фронтенду та Бекенду:
        - Швидкі та відповідні веб-технології, що забезпечують гладке взаємодію користувача з інтерфейсом та швидку передачу даних між клієнтом та сервером.
2. Вимоги до Штучного Інтелекту

a. Точність та Надійність:
   - Моделі штучного інтелекту мають бути високоточними для забезпечення точного відновлення зображень, особливо у складних випадках, таких як сильне пошкодження або шум.

b. Навчання на Різноманітних Даних:
   - Для досягнення високої ефективності, моделі MI повинні бути навчені на великих та різноманітних наборах даних, щоб можна було впоратися з широким спектром сценаріїв відновлення зображень.

c. Часова Ефективність:
   - Моделі повинні бути здатні обробляти зображення швидко, забезпечуючи ефективне використання часу та ресурсів.

3. Інтерфейс Користувача та Доступність

a. Інтуїтивний Користувацький Інтерфейс:
   - Простий та зрозумілий інтерфейс, який дозволяє користувачам легко завантажувати та отримувати оброблені зображення.

b. Адаптивність та Сумісність:
   - Інтерфейс має бути адаптивним та сумісним з різними пристроями та браузерами

4.2.2 Архітектура рішення запропонованої програмної системи

Враховуючи всі вимоги, була висунута пропозиція створити систему з такою архітектурою:

1. Веб додаток написаний мовою програмного коду було обрано PHP, JS, HTML, CSS.
2. Використано фреймворк Laravel.
3. Використано Alpine.js
4. Використано Tailwind CSS

5. Використано Redis для обмеження швидкості
6. Місце збереження зображення було обрано DigitalOcean Space;
7. Бібліотека Replicate AI для Image restoration.

Схема взаємодії компонентів подана на рис. 4.1

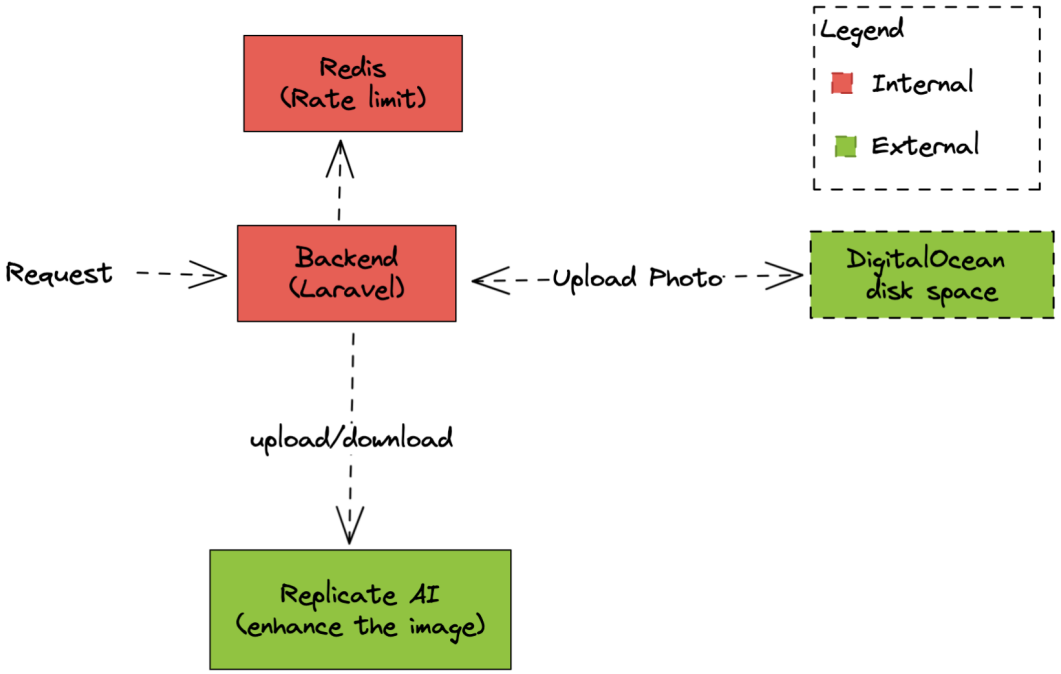

Рис. 4.1. Схема взаємодії компонентів системи

Програма була розроблена з метою відновлення та оптимізації зображень через веб-інтерфейс, надаючи користувачам можливість поліпшення якості, виправлення пошкоджень, або просто оновлення старих зображень до сучасних стандартів. Основна ідея полягає в тому, щоб надати користувачам простий та ефективний інструмент, який може бути використаний без спеціальних знань у галузі обробки зображень.

З урахуванням такого розподілу компонентів системи, в кожному з них з'являються унікальні обов’язки. Оскільки користувач взаємодіє напряму з клієнтським веб сайтом то графічний інтерфейс повинен побудований ергономічно і з урахуванням різноманітних формфакторів. А також врахувати використання в

сучасних браузерах з чуйним до розмірів екрану та рекомендацій щодо дизайну для оптимального відображення.

Основним фокусом програми є забезпечення високої якості обробки зображень з одночасним забезпеченням простоти використання для кінцевого користувача. Це досягається через інтуїтивно зрозумілий інтерфейс, швидке оброблення запитів та надійне зберігання даних. Такий підхід дозволяє користувачам з мінімальними технічними знаннями ефективно використовувати систему для відновлення та оптимізації зображень.

Ця програма також відображає сучасні тенденції у розробці веб-додатків, такі як використання мікросервісної архітектури, реактивного дизайну, та інтеграції з зовнішніми API для розширення функціональності.

Розглянемо більш детальніше використані технології.

Laravel: Laravel був обраний як основний фреймворк через його високу продуктивність, надійність та легкість використання. Цей фреймворк надає розробникам потужний набір інструментів, що дозволяють створювати розширені веб-додатки з урахуванням сучасних вимог безпеки та функціональності. Він дозволяє легко масштабувати додаток та впроваджувати складні функції, такі як обробка зображень та управління даними.

Alpine.js: Alpine.js використовується для створення інтерактивного користувацького інтерфейсу. Ця легка бібліотека дозволяє додавати динаміку на клієнтській стороні без потреби у важких JavaScript-фреймворках. Alpine.js є ідеальним вибором для реалізації простих, але ефективних динамічних елементів інтерфейсу.

Tailwind CSS: Tailwind CSS дозволяє швидко та ефективно створювати адаптивний дизайн з мінімальними зусиллями. Цей утилітарний фреймворк для стилізації надає гнучкість для створення унікального та сучасного дизайну, одночасно забезпечуючи консистентність та легкість утримання коду.

Redis: Використання Redis для обмеження швидкості запитів (rate limiting) є ключовим для забезпечення стабільності та безпеки додатку. Redis ефективно

управляє великим потоком запитів, запобігаючи перевантаженню сервера та можливим DDoS-атакам, а також забезпечує швидкий доступ до даних завдяки своїм високошвидкісним можливостям.

DigitalOcean Spaces: Вибір DigitalOcean Spaces для зберігання зображень ґрунтується на його високій надійності та масштабованості. Це рішення забезпечує високу доступність даних, гнучкість управління ресурсами та зручність інтеграції з іншими компонентами системи.

Якщо додаток призначений для широкого використання, важливо, щоб інструкції були докладними, а при виникненні помилок користувачам надавалася інформація про причину помилки та можливі шляхи її виправлення. Однією з обов'язкових функцій програми є можливість завантаження зображень та отримання відновленого результату. Додатково, для поліпшення користувацького досвіду та зменшення навантаження на сервер, рекомендується впровадження миттєвої валідації зображень в реальному часі.

Веб-сайт виступає як посередник у взаємодії між клієнтом та сервісом відновлення зображень. Оскільки отримані користувачем зображення будуть шифруватися, необхідно встановити потужне обладнання та оптимізовані веб-двигуни та балансери навантаження для забезпечення ефективної роботи всієї системи. Керування доступом до використовуваних сервісів буде здійснюватися за допомогою токенів, виданих сервером.

Для зберігання зображень обрано DigitalOcean Spaces через його надійність, високу доступність та масштабованість. Це дозволяє безпечно зберігати велику кількість зображень та забезпечувати швидкий доступ до них.

Відновлення зображення є критично важливим компонентом нашої системи, адже воно лежить в основі забезпечення високої якості кінцевих результатів. Для забезпечення гладкої інтеграції цього процесу з веб-сервером, важливо використовувати сервіси, побудовані за парадигмою RESTful. Це дозволяє забезпечити масштабованість, гнучкість та легкість у взаємодії з іншими компонентами системи.

Replicate API, який ми використовуємо для розпізнавання та відновлення зображень, є оптимальним вибором для нашої задачі. Він не лише забезпечує ефективну обробку та відновлення зображень, але й є відокремленою системою в хмарі, що додає додаткову гнучкість та масштабованість. Використання хмарного сервісу як Replicate API дає нам переваги у вигляді зменшення навантаження на наші внутрішні ресурси та забезпечення високої доступності та надійності обробки зображень.

4.3 Розробка алгоритму роботи системи

Оскільки система може використовуватись багатьма користувачами то алгоритм може бути адаптований або розширений залежно від специфіки вашої системи та потреб користувачів. Важливо забезпечити гнучкість та ефективність на кожному етапі, щоб оптимізувати процес відновлення зображень.

Розглянемо наступни алгорим для нашої системи.

1. Прийом Зображення

Крок 1.1: Отримання зображення від користувача.

- Користувач завантажує зображення через веб-інтерфейс.
- Система перевіряє формат та розмір зображення, щоб забезпечити сумісність.

2. Передача Зображення

Крок 2.1: Відправлення зображення на сервер.

- Зображення передається на бекенд-сервер через безпечний канал зв'язку.
- Опціонально: зображення тимчасово зберігається у хмарі (наприклад, на DigitalOcean Spaces).

3. Обробка Зображення

Крок 3.1: Аналіз зображення.

- Визначення пошкоджень або дефектів на зображенні (шум, розмитість, втрата деталей тощо).

Крок 3.2: Вибір методу відновлення.

- Залежно від типу та ступеня пошкоджень вибирається найбільш підходящий метод відновлення.

4. Застосування Алгоритмів Штучного Інтелекту

Крок 4.1: Застосування МІ для відновлення.

- Використання нейронних мереж для відновлення якості, зменшення шуму, підвищення роздільної здатності тощо.

Крок 4.2: Адаптація результатів.

- Налаштування параметрів обробки залежно від потреб користувача та характеристик зображення.

5. Зберігання та Відправлення Результату

Крок 5.1: Зберігання обробленого зображення.

- Зображення зберігається на сервері або у хмарному сховищі.

Крок 5.2: Надсилання результату користувачу.

- Користувач отримує доступ до відновленого зображення через веб-інтерфейс або через посилання для завантаження.

6. Зворотний Зв'язок і Оцінка

Крок 6.1: Збір зворотного зв'язку від користувача.

- Користувачі можуть оцінити якість відновлення та надати відгуки.

Крок 6.2: Аналіз та вдосконалення.

- Використання зворотного зв'язку для покращення алгоритмів та сервісу загалом.

4.4 Програмна реалізація системи відновлення зображення

4.4.1 Реєстрація на Replicate та отримання ключа доступу

Для можливості використовувати Replicate API необхідно отримати API ключ, який можна отримати створивши підписку на необхідний сервіс:

1. Перехід на сайт Replicate (рис. 4.2)

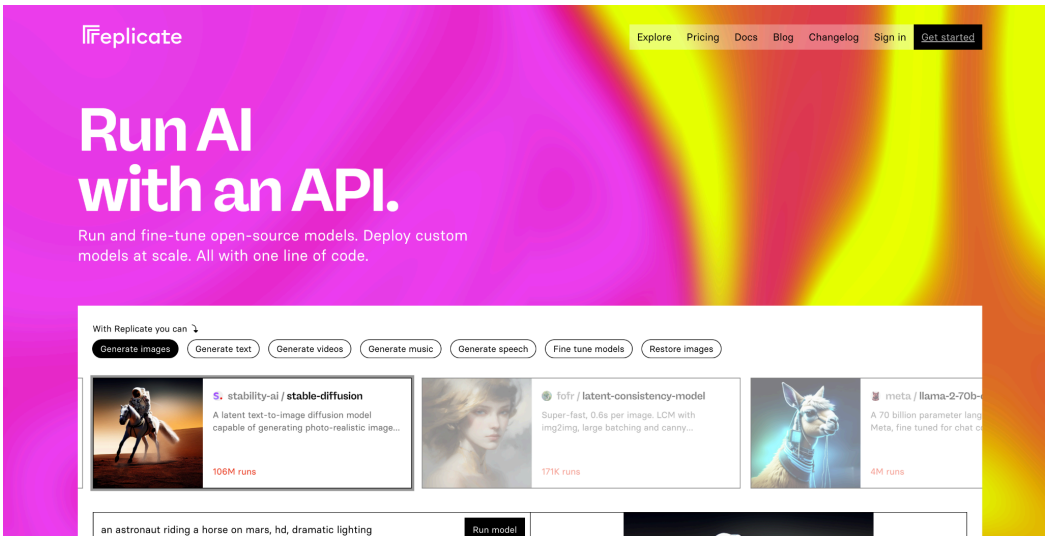

Рис. 4.2. Головна сторінка Replicate

2. Потрібно авторизуватись або створити новий обліковий запис для можливості створення підписки. Для входу в систему було обрано обліковий запис GitHub (рис. 4.3)

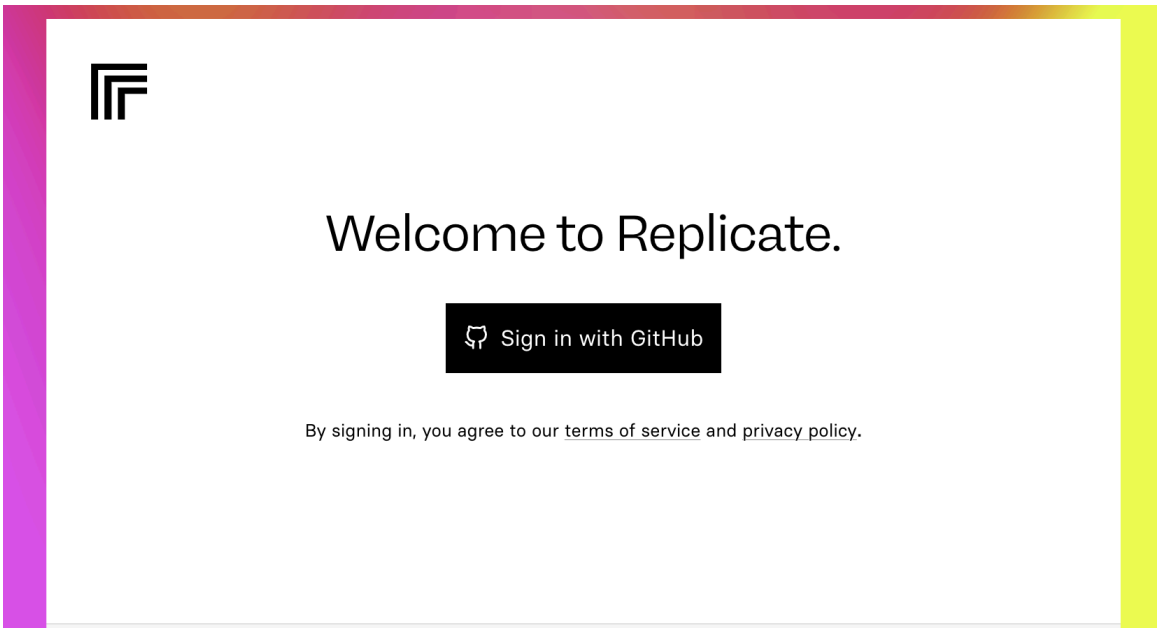

Рис. 4.3. Створення підписки передбачає вхід в обліковий запис

3. Після завершення входу в обліковий запис відбувається переадресація на головну сторінку з інформацією про Replicate (рис. 4.4).

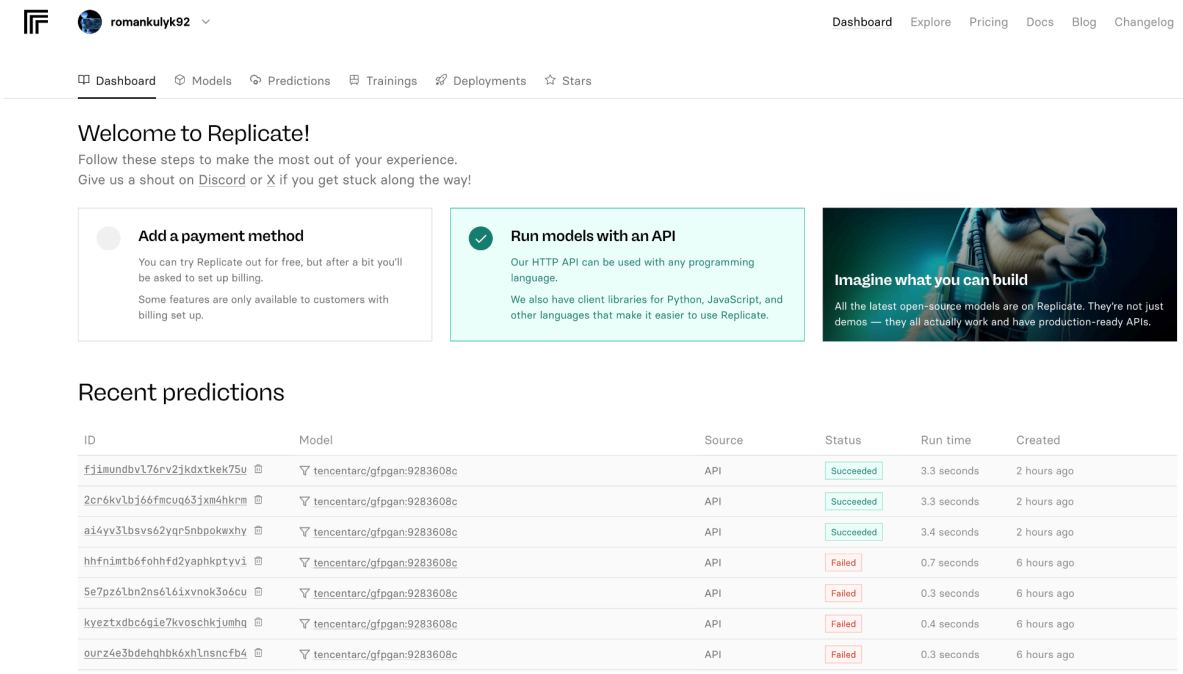

Рис. 4.4. Головна сторінка після завершення входу в Replicate

4. Для того щоб добавити новий ключ доступу потрібно перейти в Налаштування акаунту користувача. На першій вкладці API tokens ми маємо змогу добавини новий ключ доступу (рис. 4.5) та (рис. 4.6)

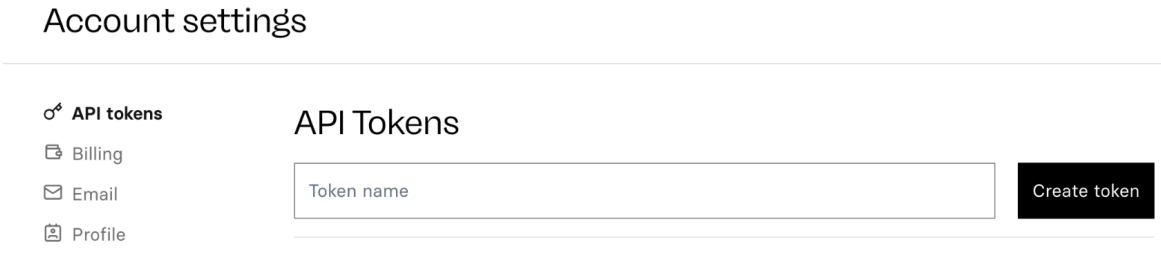

Рис. 4.5. Форма створення ключа доступу до методів Replicate API

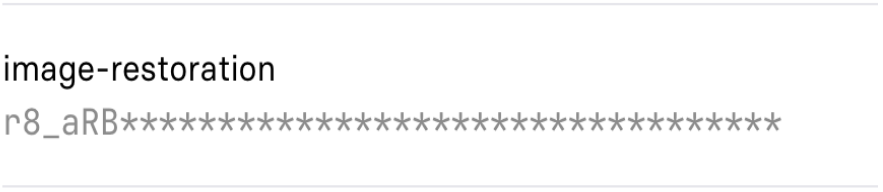

Рис. 4.6. Результат створення підписки та ключа доступу до Replicate API

4.4.2 Створення веб-сервера системи

Розробка веб-сервера є ключовим елементом у архітектурі системи відновлення зображень. Такий сервер виступає як центральний вузол інтеракції: через нього користувачі можуть надсилати запити на обробку зображень та отримувати результати. Основні функції веб-сервера охоплюють:

- Обробка HTTP-запитів: Забезпечує прийом та обробку запитів від користувачів.
- Управління даними користувачів: Включає зберігання та обробку користувацької інформації.
- Інтеграція з модулями відновлення зображень: Забезпечує взаємодію сервера зі спеціалізованими алгоритмами і технологіями відновлення зображень.
- Відправлення відповідей користувачам: Повертає результати обробки користувачам.

Для реалізації сервера було обрано мову програмування PHP через її простий синтаксис, оптимізацію під багатоядерні системи, високу швидкість обробки інструкцій, а також зручність інтеграції зі сторонніми бібліотеками.

Додатково, для ефективної організації вихідного коду, використовується фреймворк Laravel. Laravel надає потужні інструменти для маршрутизації та дотримується архітектури MVC (Model-View-Controller), що значно спрощує розробку та підтримку складних веб-додатків. Ця архітектура сприяє чіткому розділенню логіки обробки даних (Model), інтерфейсу користувача (View) та управління даними (Controller), забезпечуючи високий рівень масштабованості та легкість управління проектом.

Для розробки та запуску програмного коду на PHP із використанням фреймворку Laravel вам потрібно встановити декілька компонентів та забезпечити належне середовище розробки. Ось основні кроки:

1. Встановлення PHP: Laravel вимагає PHP певної версії, тому перш за все вам потрібно встановити PHP. Зазвичай, для останніх версій Laravel потрібна PHP версії 7.3 або вище.
2. Веб-сервер: На вибір є кілька веб-серверів, наприклад Apache або Nginx. Вибір залежить від ваших переваг та вимог проекту.
3. Система управління базами даних (СУБД): Laravel підтримує кілька СУБД, таких як MySQL, PostgreSQL, SQLite та інші. Виберіть ту, яка найкраще відповідає вашим потребам.
4. Composer: Це інструмент для управління залежностями в PHP, який дозволяє легко встановлювати та оновлювати бібліотеки та пакети.
5. Laravel: Після встановлення PHP і Composer, ви можете встановити Laravel за допомогою Composer. Це можна зробити за допомогою команди composer create-project --prefer-dist laravel/laravel ім'я_проекту.
6. Artisan: Це командний рядок Laravel, який допомагає створювати шаблони коду, бази даних, та керувати іншими аспектами додатків.
7. Розробницькі інструменти: Редактор коду або інтегроване середовище розробки (IDE) є необхідними для написання та відлагодження коду. Популярними виборами є Visual Studio Code, PHPStorm та інші.
8. Git: Система контролю версій, яка дозволить вам керувати версіями вашого коду та співпрацювати з іншими розробниками.
9. Додаткові пакети та інструменти: Залежно від ваших потреб, ви можете встановити додаткові пакети та інструменти, такі як Laravel Mix для компіляції асетів, пакети для автентифікації користувачів, інструменти для тестування тощо.

Переконайтеся, що ваше середовище відповідає всім вимогам Laravel, перевіривши офіційну документацію Laravel перед початком розробки.

Після завершення встановлення усіх компонентів потрібно зробити кілька наступних кроків для налаштування та початку розробки вашого веб-додатку наш проект. Розпочнемо з першого і головного кроку, а саме з конфігурації середовища.

Для цього нам потрібно створити .env файл та налаштувати його наступним цей файл відповідно до наших потреб. Конфігураційні дані файлу .env наведено в лістингу 4.1. А також для генерації ключа додатку нам потрібно виконати команду `php artisan key:generate` (рис. 4.7) для створення унікального ключа додатку, який використовується для шифрування сесій та інших зашифрованих даних.

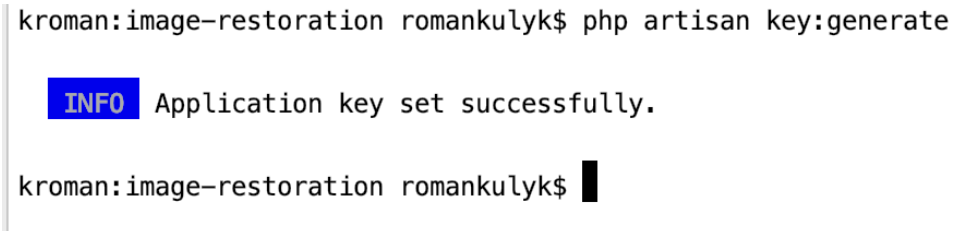

Рис. 4.7. Виконання команди генерації ключа додатку

Лістинг 4.1.

```
APP_NAME=RestoreImage
APP_ENV=local
APP_KEY=base64:absYpFVQVjf/E9qpdrKd92t118xvxh17hCuWMLy3xjo=
APP_DEBUG=true
APP_URL=http://restore.image
LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug
DB_CONNECTION=mysql
DB_HOST=app_host
DB_PORT=app_port
DB_DATABASE=restore_image_db
DB_USERNAME=root
DB_PASSWORD=
BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120
FILESYSTEM_DISK=digitalocean
DIGITALOCEAN_SPACES_KEY=DO0***
DIGITALOCEAN_SPACES_SECRET=A5H****
DIGITALOCEAN_SPACES_ENDPOINT=https://restoreimage.ams3.digitaloceanspa
ces.com
DIGITALOCEAN_SPACES_REGION=AMS3
```

```
DIGITALOCEAN_SPACES_BUCKET=restoreimage
DIGITALOCEAN_SPACES_URL=https://restoreimage.ams3.digitaloceanspaces.com
REPLICATE_API_TOKEN=r8_***
MEMCACHED_HOST=app_memcahed_host
REDIS_HOST=app_redis_host
REDIS_PASSWORD=null
REDIS_PORT=app_redis_port
MAIL_MAILER=smtp
MAIL_HOST=mailpit
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS="info@restoreimage.com"
MAIL_FROM_NAME="${APP_NAME}"
```

Основний файл програми лише виконує ініціалізацію основних сервісів та налаштувань програми перед її запуском на веб-сервері. Код файлу index.php наведено в лістингу 4.2.

Лістинг 4.2.

```
<?php

use Illuminate\Contracts\Http\Kernel;
use Illuminate\Http\Request;
define('LARAVEL_START', microtime(true));
/*
|--------------------------------------------------------------------------
| Check If The Application Is Under Maintenance
|--------------------------------------------------------------------------
|
| If the application is in maintenance / demo mode via the "down" command
| we will load this file so that any pre-rendered content can be shown
| instead of starting the framework, which could cause an exception.
|
*/
if (file_exists($maintenance = __DIR__.'/../storage/framework/maintenance.php')) {
    require $maintenance;
}
```

```
/*
|--------------------------------------------------------------------------
-
| Register The Auto Loader
|--------------------------------------------------------------------------
-
|
| Composer provides a convenient, automatically generated class loader
for
| this application. We just need to utilize it! We'll simply require
it
| into the script here so we don't need to manually load our classes.
|
*/
require __DIR__.'/../vendor/autoload.php';
/*
|--------------------------------------------------------------------------
-
| Run The Application
|--------------------------------------------------------------------------
-
|
| Once we have the application, we can handle the incoming request
using
| the application's HTTP kernel. Then, we will send the response back
| to this client's browser, allowing them to enjoy our application.
|
*/
$app = require_once __DIR__.'/../bootstrap/app.php';
$kernel = $app->make(Kernel::class);
$response = $kernel->handle(
   $request = Request::capture()
)->send();
$kernel->terminate($request, $response);
```

Для відображення сторінок веб-додатку потрібно налаштувати маршрути. В нашому проекті було створено два маршрути, це маршрут для головної сторінки а також маршрут POST запиту для завантаження зображення. Ці маршрути налаштовано у файлі routes/web.php. Код файлу представлено на лістингу 4.3.

Лістинг 4.3.

```
<?php
use App\Http\Controllers\UploadController;
use Illuminate\Support\Facades\Route;
```

```
Route::get('/', function () {
    return view('home');
});
Route::post('/upload', [UploadController::class, '__invoke']);
```

Щоб відобразити головну сторінку було створено шаблон вигляду. В якому ми створили структуру нашого додатку, підключили потрібні нами скрипти та стилі а також розроблені відповідна функція для завантаження зображення. Код файлу представлено на лістингу 4.4.

Лістинг 4.4.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Image Restoration App</title>
    <link
href="https://unpkg.com/tailwindcss@2.2.16/dist/tailwind.min.css"
rel="stylesheet">
    <script defer
src="https://cdn.jsdelivr.net/npm/alpinejs@3.x.x/dist/cdn.min.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/axios/1.3.4/axios.min.js"

integrity="sha512-LUKzDoJKOLqnxGWWIBM4lzRBlxcva2ZTztO8bTcWPmDSpkErWx0bSP4pd
sjNH8kiHAUPaT06UXcb+vOEZH+HpQ=="
            crossorigin="anonymous"
referrerpolicy="no-referrer"></script>
</head>
<body class="bg-gray-100">
    <div class="max-w-xl mx-auto py-12" x-data="uploadForm()">
        <h1 class="text-3xl font-bold mb-4 text-center">Restore
Photos</h1>
        <p class="text-gray-700 mb-6 text-center">Restore and enhance
your old photos with our AI model.</p>
        <form class="bg-white rounded-lg shadow-lg p-6 relative"
@submit.prevent="submitForm">
            <div x-show="loading" class="absolute top-0 left-0 w-full
h-full bg-white flex flex-col items-center justify-center">
                @include('components.spinner')
                <span x-show="uploadProgress < 100"
x-text="uploadProgress + '%'"></span>
                <span x-show="uploadProgress == 100">
                    Hold on, we are enhancing your photo...
```

```
                    </span>
                </div>
                <div class="flex max-w-lg justify-center rounded-md
border-2 border-dashed border-gray-300 px-6 pt-5 pb-6">
                    <div class="space-y-1 text-center">
                        <svg class="mx-auto h-12 w-12 text-gray-400"
stroke="currentColor" fill="none" viewBox="0 0 48 48" aria-hidden="true">
                            <path d="M28 8H12a4 4 0 00-4 4v20m32-12v8m0
0v8a4 4 0 01-4 4H12a4 4 0 01-4-4v-4m32-4l-3.172-3.172a4 4 0 00-5.656 0L28
28M8 32l9.172-9.172a4 4 0 015.656 0L28 28m0 0l4 4m4-24h8m-4-4v8m-12 4h.02"
stroke-width="2" stroke-linecap="round" stroke-linejoin="round"></path>
                        </svg>
                        <div class="flex text-sm text-gray-600"
x-show="!loading">
                            <label for="file-upload" class="relative
cursor-pointer rounded-md bg-white font-medium text-indigo-600
focus-within:outline-none focus-within:ring-2 focus-within:ring-indigo-500
focus-within:ring-offset-2 hover:text-indigo-500">
                                <span>Upload a file</span>
                                <input
                                        class="hidden shadow
appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight
focus:outline-none focus:shadow-outline"
                                        id="file-upload"
                                        type="file"
                                        x-on:change="(formData.file =
Object.values($event.target.files)[0]) && submitForm()"
                                >
                            </label>
                            <p class="pl-1">or drag and drop</p>
                        </div>
                        <p class="text-xs text-gray-500">PNG, JPG up to
2MB</p>
                    </div>
                </div>
                <div class="mb-4 text-red-900 rounded"
x-show="formErrors['file']" x-text="formErrors['file'][0]"></div>
            </form>

            <div id="result" class="my-4 bg-white rounded-lg shadow-lg p-6"
:class="{ 'hidden': !result}">
                <div class="flex items-center justify-between mb-4">
                    <h2 class="text-xl font-bold">Restored Photo</h2>
                    <a
                            class="border p-2 border-blue-500 text-blue-500
font-bold rounded focus:outline-none focus:shadow-outline"
                            :href="result"
                            download
                    >
```

```
                        Download
                    </a>
                </div>
                <img :src="result" alt="Restored Photo" class="mb-4">
            </div>
        </div>
        <script>
            function uploadForm() {
                return {
                    loading: false,
                    formData: {
                        email: "",
                        file: "",
                    },
                    result: null,
                    error: null,
                    formErrors: {},
                    uploadProgress: 0,
                    submitForm() {
                        this.loading = true;
                        const data = new FormData()
                        Object.keys(this.formData).map((key, index) => {
                            data.append(key, this.formData[key])
                        });
                        this.error = null;
                        this.formErrors = {};
                        this.result = null;
                        axios.post('/upload', data, {
                            onUploadProgress: (event) => {
                                this.uploadProgress =
Math.round((event.loaded / event.total) * 100);
                            }
                        }).then(response => {
                            this.result = response.data.result;
                        }).catch(error => {
                            this.error = error.response.data.message;
                            if (error.response.status === 422) {
                                this.formErrors =
error.response.data.errors;
                            }
                        }).finally(() => {
                            this.loading = false;
                        })
                    }
                }
            }
        </script>
    </body>
    </html>
```

Якщо детальніше розглянути даний код, то ми побачимо що у нас є форма для завантаження зображення. Форма місить подію яка в свою чергу після її виклику спрацьовує наступним чином, а саме:

- змінює статус змінної loading
- отримує вибране зображення
- відправляє запит на маршрут “/upload” для завантаження зображення та відновлення зображення
- після обробки запиту буде отримав відповідь і відповідно видасть певний результат на екрані

Роботу з Replicate API було виділено та реалізовано у файлі Replicate.php, розташованому у каталозі app/Http/Integrations/Api/. На лістингу 4.5 показано, як оголошено структуру Replicate, яка включає в себе посилання на сервер Replicate API та ключ підписки, що буде включатися в кожен запит. Також визначено стандартні заголовки.

Лістинг 4.5.

```
<?php
namespace App\Http\Integrations\Api;
use Saloon\Http\Connector;
use Saloon\Traits\Plugins\AcceptsJson;
class Replicate extends Connector
{
   use AcceptsJson;
   public function __construct(
       protected string $apiToken,
   ){
       $this->withTokenAuth($this->apiToken, 'Token');
   }
   public function resolveBaseUrl(): string
   {
       return 'https://api.replicate.com/v1';
   }
   protected function defaultHeaders(): array
   {
       return [
           "Content-Type" => "application/json",
```

```
            ];
        }
    }
```

Також було реалізовано наступні класи для роботи з запитами до Replicate API в файлі Predictions.php та FetchPrediction.php який розміщений в папці app/Http/Integrations/Replicate/Requests. На лістингу 4.6 показано код файлу Predictions.php а також на 4.7 показано код файлу FetchPrediction.php.

Лістинг 4.6.

```
<?php
namespace App\Http\Integrations\Replicate\Requests;
use App\Http\Integrations\Replicate\DTOs\PredictionRecord;
use Saloon\Contracts\Body\HasBody;
use Saloon\Contracts\Response;
use Saloon\Enums\Method;
use Saloon\Http\Request;
use Saloon\Traits\Body\HasJsonBody;
class Predictions extends Request implements HasBody
{
    use HasJsonBody;
    protected Method $method = Method::POST;
    public function __construct(private string $imageUrl)
    {
        //
    }
    public function resolveEndpoint(): string
    {
        return '/predictions';
    }
    protected function defaultBody(): array
    {
        return [
            'version' =>
'9283608cc6b7be6b65a8e44983db012355fde4132009bf99d976b2f0896856a3',
            'input' => [
                'version' => 'v1.4',
                'scale' => 2,
                'img' => $this->imageUrl,
            ]
        ];
    }
    public function createDtoFromResponse(Response $response):
PredictionRecord
```

```
    {
        return PredictionRecord::fromResponse($response);
    }
}
```

Лістинг 4.7.

```
<?php
namespace App\Http\Integrations\Replicate\Requests;
use App\Http\Integrations\Replicate\DTOs\Result;
use Saloon\Contracts\Response;
use Saloon\Enums\Method;
use Saloon\Http\Request;
class FetchPrediction extends Request
{
    protected Method $method = Method::GET;

    public function __construct(private string $imageId)
    {
        //
    }
    public function resolveEndpoint(): string
    {
        return "/predictions/{$this->imageId}";
    }
    public function createDtoFromResponse(Response $response): Result
    {
        return Result::fromResponse($response);
    }
}
```

Давайте розглянемо створений контроллер UploadController для обробки запиту за маршрутом “/upload”. Даний контролер повініть призначений одній дії тому ми визначили один __invok метод у контролері. В даному методі ми проводимо завантаження файлу та отримання посилання на нього після чого допомогою функції Pipeline, надаємо план задач які потрібно виконати і в результаті якого ми отримуємо посилання на відновлене зображення. В план дій входять наступні функції:

- initiatePrediction - функція ініціювання передбачення;
- checkCompleteness - функція перевірки повноти виконання відновлення;
- uploadRestored - функція завантаження відновленого зображення;

Код файлу представлено на лістингу 4.8.

Лістинг 4.8.

```
<?php
namespace App\Http\Controllers;
use App\Http\Integrations\Api\Replicate;
use App\Http\Integrations\Replicate\Requests\FetchPrediction;
use App\Http\Integrations\Replicate\Requests\Predictions;
use App\Http\Requests\UploadRequest;
use App\Services\ImageUploader;
use Illuminate\Support\Facades\Pipeline;
class UploadController extends Controller
{
   public function __construct(private readonly Replicate $replicate,
private readonly ImageUploader $uploader)
   {
       //
   }
   public function __invoke(UploadRequest $request)
   {
       $imageUrl =
$this->uploader->uploadAndGetUrl($request->file('file'));

       $outputImageUrl = Pipeline::send($imageUrl)
           ->through([
               $this->initiatePrediction(...),
               $this->checkCompleteness(...),
               $this->uploadRestored(...),
           ])
           ->then(fn($imageUrl) => $imageUrl);
       return [
           'result' => $outputImageUrl,
       ];
   }
   private function initiatePrediction(string $imageUrl, $next)
   {
       $predict = new Predictions($imageUrl);
       $record = $this->replicate->send($predict)->dtoOrFail();
       return $next($record->imageId);
   }
   private function checkCompleteness(string $imageId, $next)
   {
       $fetch = new FetchPrediction($imageId);

       do {
           sleep(1);
           $restoredImage =
$this->replicate->send($fetch)->dtoOrFail();
```

```
            } while ($restoredImage->processing());

            return $next($restoredImage->output);
        }
        private function uploadRestored(string $imageUrl, $next)
        {
            return $next($this->uploader->fetchAndUpload($imageUrl,
'restored'));
        }
    }
```

4.5 Виконання тестування розробленого програмного продукту з використанням тестових даних.

Для перевірки правильності роботи розробленої системи було проведено тестування основних сценаріїв використання. При переході за посиланням на веб-сайт, користувачу відображається головна сторінка сайту (див. рис. 4.8).

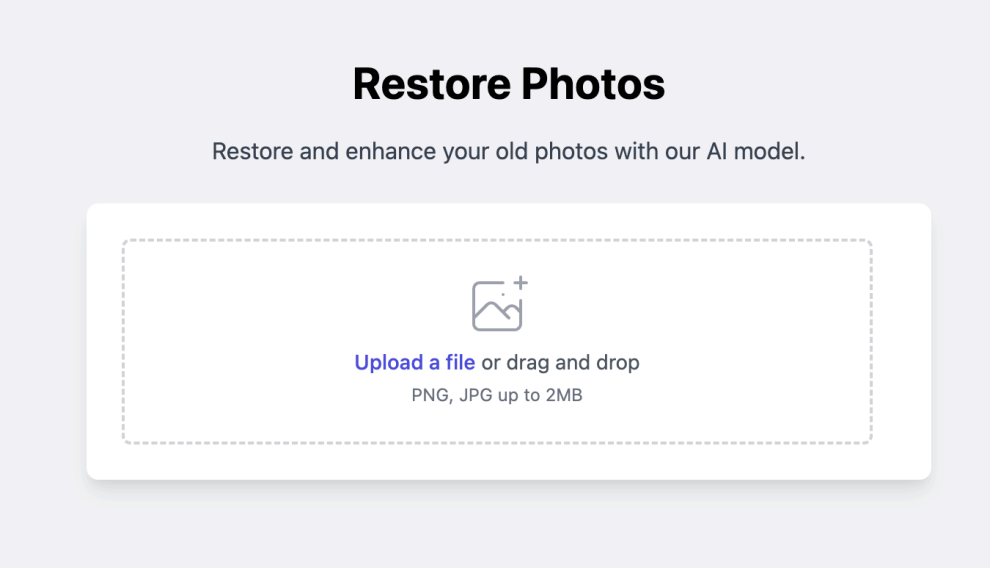

Рис. 4.8. Вигляд головної сторінки

Для можливості відновлення зображення на сайті потрібно натиснути “Upload a file” або перетягнути зображення на вказану область на екрані. Після того як користував вибрав зображення або перетягнув його на вказану область почнеться виконання дії. Після цього здійснюється перевірка типу та розміру вибраного

зображення. Якщо все добре то розпочинається процес завантаження зображення та буде представлено анімацію в вигляді завантаження зображення з прогресом завантаження (рис. 4.9).

Рис. 4.9. Вигляд сторінки з прогресом завантаження зображення

Після завершення завантаження відбувається процес відновлення зображення і буде представлено анімацію в вигляді “спінера” та напису про процес відновлення зображення. Вигляд сторінки з процесом відновлення зображення наведено на рис. 4.10.

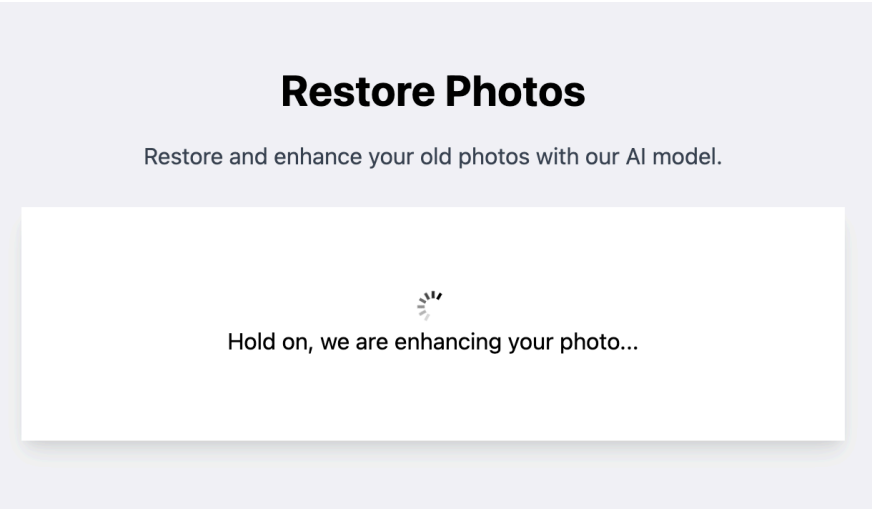

Рис. 4.10. Вигляд сторінки з процесом відновлення зображення

Після успішного відновлення зображення здійснюєтьс представлення цього зображення на головній сторінці нище форми завантаження зображення. Вигляд сторінки успішного відновлення зображення з можливістю його завантаження проілюстровано на рис. 4.11.

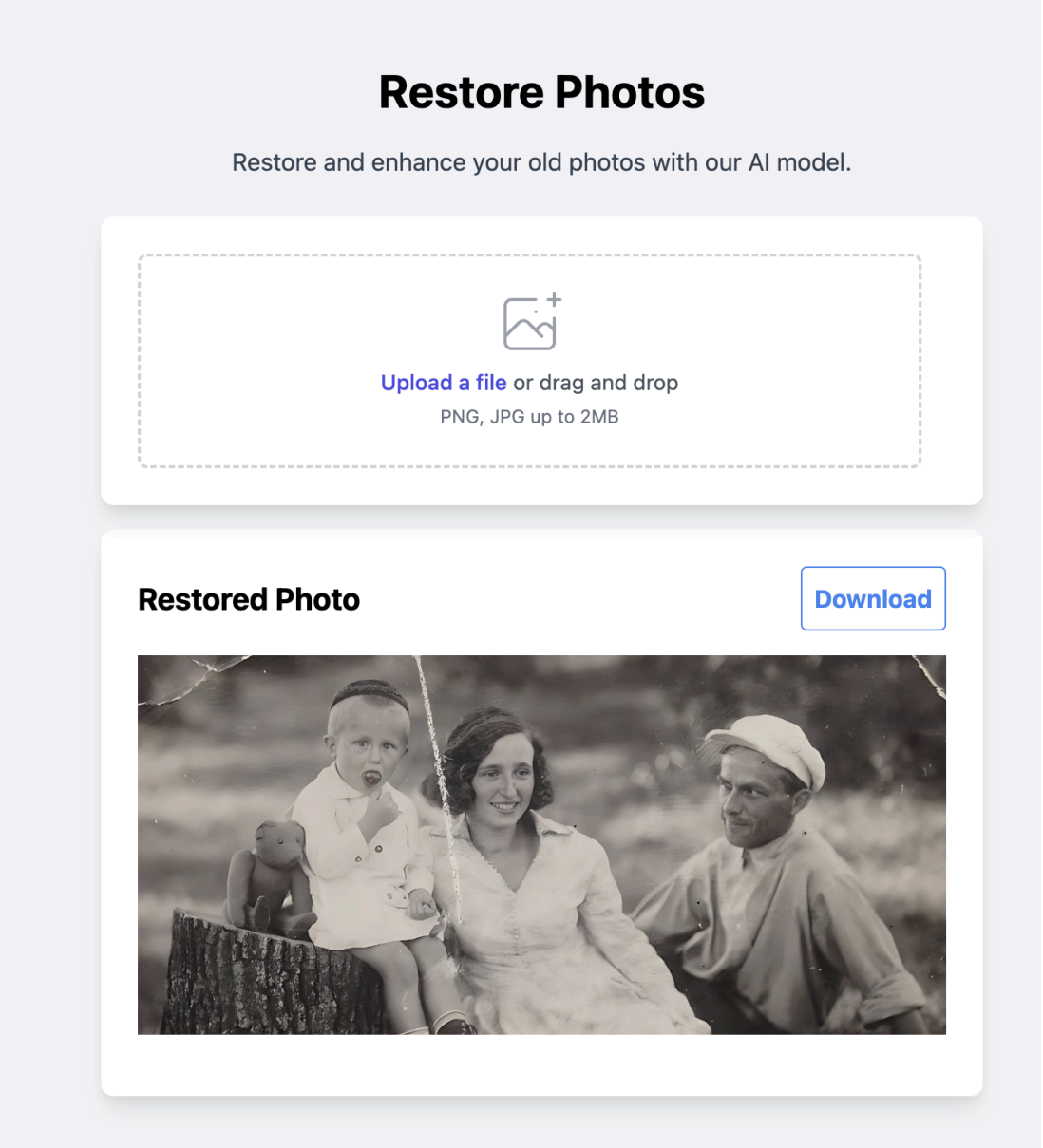

Рис. 4.11. Вигляд сторінки успішного відновлення зображення

Було прийнято рішення зробити тестування негативного сценарію завантаження зображень в систему і звірення результату її роботи з очікуваним. Для цього було здійснено спробу завантаження зображення не типу png чи jpeg. Після

обробки запиту було отримано відмову в завантажені з вказаною причиною невідповідності наданого файлу до вказаних в типів зображень у формі. Вигляд результату операції наведено на рис. 4.12. Також було здійснено спробу завантаження зображення більшого розмір. Після обробки запиту було отримано відмову в завантажені з вказаною причиною невідповідності наданого файлу до вказаного максимального розміру зображення у формі. Вигляд результату операції наведено на рисунку 4.13.

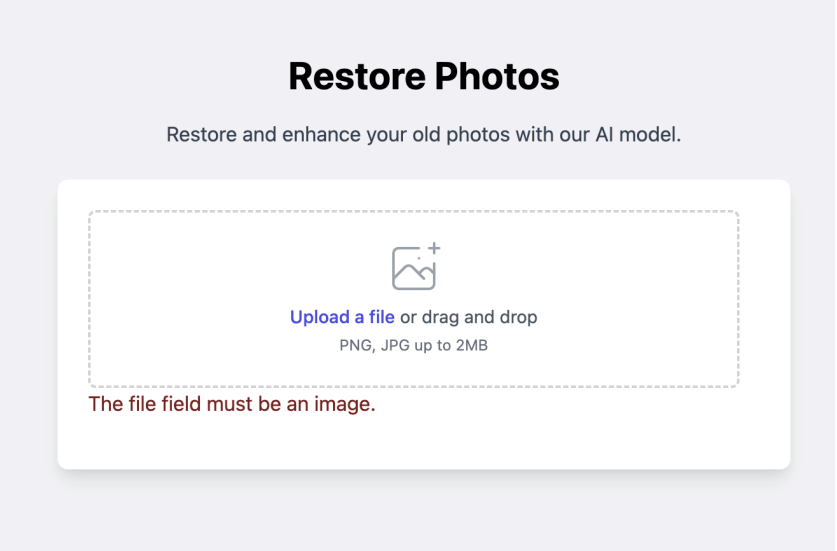

Рис. 4.12. Спроба здійснення відновлення зображення іншого типу

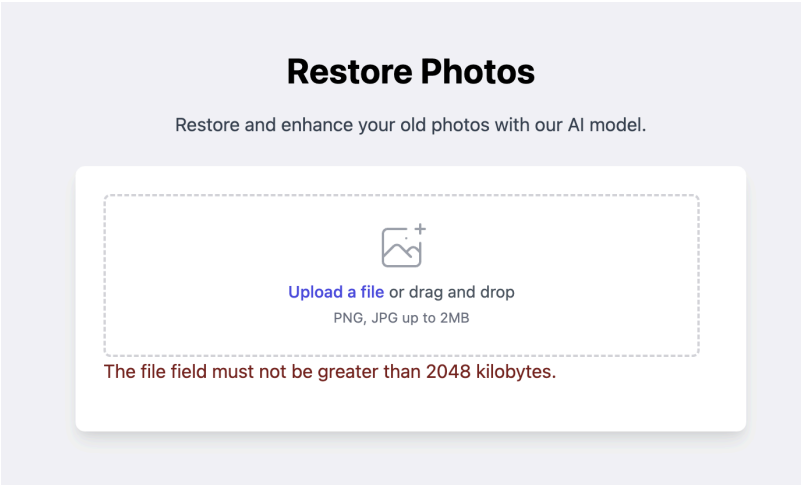

Рис. 4.13. Спроба здійснення відновлення зображення більшого розміру

Після перевірки негативних сценарії системи було здійснено тест на відновлення зображення з коректним розміром та типом зображення. Для цього було повторено попередній сценарій за винятком використання зображення обличчя користувача, яке відповідає зареєстрованому в обліковому записі. Після виконання цієї операції клієнтський додаток отримав позитивний результат, тому здійснив перехід на головну сторінку з виведенням даних про успішність відновлення зображення та даним зображенням. Зображення для відновлення наведено на рис. 4.14 та результат виконання цієї операції наведено на рис. 4.15.

Рис. 4.14. Зображення для відновлення

Рис. 4.15. Результат виконання відновлення зображення

4.6 Висновки до розділу

У четвертому розділі магістерської роботи було зосереджено увагу на детальній розробці алгоритму та програмної реалізації системи відновлення зображень. Цей етап роботи виявився ключовим у досягненні цілей дослідження, оскільки саме тут були втілені розроблені теоретичні концепції та методики.

Архітектура системи була спроектована з урахуванням сучасних підходів у розробці веб-додатків, включаючи використання фреймворків Laravel, Alpine.js, Tailwind CSS та інтеграцію з хмарними сервісами, такими як DigitalOcean Spaces. Такий вибір технологій забезпечив не тільки високу продуктивність та масштабованість системи, але й її гнучкість та адаптивність до різноманітних потреб користувачів.

Розроблений алгоритм відновлення зображень ефективно вирішує завдання по видаленню шуму, підвищенню роздільної здатності, корекції кольору та інших аспектів оптимізації зображень. Важливою особливістю системи є її здатність автоматично обирати оптимальні параметри обробки, що базується на характеристиках кожного конкретного зображення.

Програмна реалізація включала створення інтуїтивно зрозумілого користувацького інтерфейсу, що забезпечує простоту використання системи навіть для недосвідчених користувачів. Ефективність системи була додатково підтверджена через ретельне тестування, яке включало перевірку різних сценаріїв використання та обробки зображень.

У підсумку, розроблена система відновлення зображень демонструє високу точність, швидкість обробки та здатність адаптуватися до різних типів зображень і пошкоджень. Це робить її ефективним інструментом не тільки для індивідуального використання, але й для професійного застосування у сфері обробки зображень. Таким чином, четвертий розділ підтверджує успішність вибраного підходу та вносить вагомий вклад у досягнення загальних цілей магістерського дослідження.

ВИСНОВКИ

У даній магістерській роботі було розглянуто глибокий аналіз проблематики відновлення зображень із застосуванням сучасних методів обробки зображень та машинного навчання. Основною метою дослідження було розробка ефективного алгоритму та програмної системи, здатних відновлювати зображення високої якості, усуваючи шум, підвищуючи роздільну здатність та виправляючи інші візуальні дефекти.

У роботі було використано цілий ряд передових технік, включаючи глибокі нейронні мережі, генеративно-змагальні мережі та інші методи глибокого навчання для досягнення високої якості результатів відновлення. Особливу увагу було приділено аналізу таких метрик як PSNR, SSIM та MSE, що є критично важливими для оцінки ефективності алгоритмів відновлення зображень.

Розроблена програмна система демонструє високу ефективність у відновленні зображень, пропонуючи користувачам інтуїтивно зрозумілий інтерфейс і швидку обробку запитів. Використання хмарних технологій та сучасних веб-фреймворків, таких як Laravel, Alpine.js, і Tailwind CSS, забезпечило високу продуктивність та масштабованість системи.

Тестування системи показало її здатність ефективно відновлювати зображення з різноманітними типами пошкоджень, що підтверджує потенціал запропонованого підходу у вирішенні широкого спектру завдань відновлення зображень. Це відкриває шлях для подальшого вдосконалення та адаптації системи під різні практичні потреби у галузі обробки зображень.

У цілому, результати дослідження та розробки підкреслюють значення інтеграції передових технологій обробки зображень і машинного навчання для створення потужних інструментів відновлення зображень. Ця робота не лише сприяє науковому прогресу у сфері комп'ютерного зору, але й пропонує практичні рішення для широкого кола застосувань, від особистого використання до професійних додатків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Gonzalez, R. C., & Woods, R. E. Digital Image Processing. – 3rd ed. – Upper Saddle River, NJ : Prentice Hall, 2008. – 954 p.
2. Burger, W., & Burge, M. J. Digital image processing: an algorithmic introduction using Java. – London: Springer, 2016. – 811 p.
3. Jain, A. K. Fundamentals of digital image processing. – Upper Saddle River, NJ: Prentice Hall, 1989. – 569 p.
4. Sonka, M., Hlavac, V., & Boyle, R. Image processing, analysis, and machine vision. – 4th ed. – Boston, MA: Cengage Learning, 2014. – 912 p.
5. Pratt, W. K. Digital image processing: PIKS inside. – 4th ed. – Hoboken, NJ: Wiley-Interscience, 2007. – 808 p.
6. Wang, Z., Bovik, A. C., Sheikh, H. R., & Simoncelli, E. P. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing, 13(4), 2004, pp. 600-612.
7. Huynh-Thu, Q., & Ghanbari, M. The accuracy of PSNR in predicting video quality for different video scenes and frame rates. Telecommunication Systems, 49(1), 2012, pp. 35-48.
8. Zuiderveld, K. Contrast limited adaptive histogram equalization. Graphics gems IV, San Diego: Academic Press Professional, Inc., 1994, pp. 474-485.
9. He, K., Sun, J., & Tang, X. Single image haze removal using dark channel prior. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(12), 2011, pp. 2341-2353.
10. Dong, C., Loy, C. C., He, K., & Tang, X. Image super-resolution using deep convolutional networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38(2), 2016, pp. 295-307.
11. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. Generative adversarial nets. In Advances in neural information processing systems, 2014, pp. 2672-2680.

12. Krizhevsky, A., Sutskever, I., & Hinton, G. E. ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems, 2012, pp. 1097-1105.
13. LeCun, Y., Bengio, Y., & Hinton, G. Deep learning. Nature, 521(7553), 2015, pp. 436-444.
14. Simonyan, K., & Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations, 2015.
15. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1-9.
16. Zhang, K., Zuo, W., Chen, Y., Meng, D., & Zhang, L. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. IEEE Transactions on Image Processing, 26(7), 2017, pp. 3142-3155.
17. Buades, A., Coll, B., & Morel, J. M. A non-local algorithm for image denoising. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005, pp. 60-65.
18. Rudin, L. I., Osher, S., & Fatemi, E. Nonlinear total variation based noise removal algorithms. Physica D: Nonlinear Phenomena, 60(1-4), 1992, pp. 259-268.
19. Chambolle, A. An algorithm for total variation minimization and applications. Journal of Mathematical Imaging and Vision, 20(1), 2004, pp. 89-97.
20. Chan, T. F., & Shen, J. Mathematical models for local nontexture inpaintings. SIAM Journal on Applied Mathematics, 62(3), 2001, pp. 1019-1043.
21. Bertalmio, M., Sapiro, G., Caselles, V., & Ballester, C. Image inpainting. In Proceedings of the 27th annual conference on Computer graphics and interactive techniques, 2000, pp. 417-424.
22. Efros, A. A., & Leung, T. K. Texture synthesis by non-parametric sampling. In Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999, pp. 1033-1038.

23. Levin, A., Zomet, A., Peleg, S., & Weiss, Y. Seamless image stitching in the gradient domain. In European conference on computer vision, 2004, pp. 377-389.
24. Glassner, A. S. (Ed.). Graphics gems. – Boston, MA: Academic Press Professional, Inc., 1990. – 658 p.
25. Oppenheim, A. V., & Schafer, R. W. Digital signal processing. – Upper Saddle River, NJ: Prentice Hall, 1975. – 585 p.
26. Duda, R. O., Hart, P. E., & Stork, D. G. Pattern classification. – 2nd ed. – New York, NY: Wiley-Interscience, 2000. – 654 p.
27. Marr, D., & Hildreth, E. Theory of edge detection. Proceedings of the Royal Society of London. Series B. Biological Sciences, 207(1167), 1980, pp. 187-217.
28. Lowe, D. G. Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 2004, pp. 91-110.
29. Dalal, N., & Triggs, B. Histograms of oriented gradients for human detection. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005, pp. 886-893.
30. Viola, P., & Jones, M. J. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 2001, pp. 511-518.
31. Haralick, R. M., & Shapiro, L. G. Computer and robot vision. – Volume I. – Reading, MA: Addison-Wesley, 1992. – 672 p.
32. Szeliski, R. Computer vision: algorithms and applications. – London: Springer, 2010. – 812 p.
33. Forsyth, D. A., & Ponce, J. Computer vision: a modern approach. – 2nd ed. – Upper Saddle River, NJ: Prentice Hall, 2011. – 793 p.
34. Hartley, R., & Zisserman, A. Multiple view geometry in computer vision. – 2nd ed. – Cambridge, UK: Cambridge University Press, 2003. – 655 p.
35. Trucco, E., & Verri, A. Introductory techniques for 3-D computer vision. – Upper Saddle River, NJ: Prentice Hall, 1998. – 314 p.

36. Shapiro, L. G., & Stockman, G. C. Computer vision. – Upper Saddle River, NJ: Prentice Hall, 2001. – 608 p.
37. Horn, B. K. P. Robot vision. – Cambridge, MA: MIT Press, 1986. – 509 p.
38. Ballard, D. H., & Brown, C. M. Computer vision. – Englewood Cliffs, NJ: Prentice Hall, 1982. – 523 p.
39. Bovik, A. C. (Ed.). Handbook of image and video processing. – 2nd ed. – San Diego, CA: Academic Press, 2005. – 1160 p.
40. Wang, Z. et al. "Image quality assessment: From error visibility to structural similarity." IEEE Transactions on Image Processing, vol. 13, no. 4, 2004, pp. 600-612.
41. Blau, Y. and Michaeli, T. "The Perception-Distortion Tradeoff." 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 6228-6237.
42. Zhou, W. et al. "Learning Rich Features for Image Manipulation Detection." CVPR, 2018.
43. Dong, C. et al. "Image Super-Resolution Using Deep Convolutional Networks." IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 38, no. 2, 2016, pp. 295-307.
44. Ledig, C. et al. "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 105-114.
45. Goodfellow, I. J. et al. "Generative Adversarial Nets." NIPS, 2014.

ДОДАТКИ

Додаток А. Програмний код index.php

```
<?php
use Illuminate\Contracts\Http\Kernel;
use Illuminate\Http\Request;
define('LARAVEL_START', microtime(true));
/*
|--------------------------------------------------------------------------
| Check If The Application Is Under Maintenance
|--------------------------------------------------------------------------
|
| If the application is in maintenance / demo mode via the "down" command
| we will load this file so that any pre-rendered content can be shown
| instead of starting the framework, which could cause an exception.
|
*/
if (file_exists($maintenance = __DIR__.'/../storage/framework/maintenance.php')) {
    require $maintenance;
}
/*
|--------------------------------------------------------------------------
| Register The Auto Loader
|--------------------------------------------------------------------------
|
| Composer provides a convenient, automatically generated class loader for
| this application. We just need to utilize it! We'll simply require it
| into the script here so we don't need to manually load our classes.
|
*/
require __DIR__.'/../vendor/autoload.php';
/*
|--------------------------------------------------------------------------
| Run The Application
|--------------------------------------------------------------------------
|
|Once we have the application, we can handle the incoming request using
| the application's HTTP kernel. Then, we will send the response back
| to this client's browser, allowing them to enjoy our application.
```

```
|
*/
$app = require_once __DIR__.'/../bootstrap/app.php';
$kernel = $app->make(Kernel::class);
$response = $kernel->handle(
   $request = Request::capture()
)->send();
$kernel->terminate($request, $response);
```

Додаток Б. Програмний код home.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>Image Restoration App</title>
   <link
href="https://unpkg.com/tailwindcss@2.2.16/dist/tailwind.min.css"
rel="stylesheet">
   <script defer
src="https://cdn.jsdelivr.net/npm/alpinejs@3.x.x/dist/cdn.min.js"></script>
   <script
src="https://cdnjs.cloudflare.com/ajax/libs/axios/1.3.4/axios.min.js"

integrity="sha512-LUKzDoJKOLqnxGWWIBM4lzRBlxcva2ZTztO8bTcWPmDSpkErWx0bSP4pd
sjNH8kiHAUPaT06UXcb+vOEZH+HpQ=="
            crossorigin="anonymous"
referrerpolicy="no-referrer"></script>
</head>
<body class="bg-gray-100">
   <div class="max-w-xl mx-auto py-12" x-data="uploadForm()">
       <h1 class="text-3xl font-bold mb-4 text-center">Restore
Photos</h1>
       <p class="text-gray-700 mb-6 text-center">Restore and enhance
your old photos with our AI model.</p>
       <form class="bg-white rounded-lg shadow-lg p-6 relative"
@submit.prevent="submitForm">
           <div x-show="loading" class="absolute top-0 left-0 w-full
h-full bg-white flex flex-col items-center justify-center">
               @include('components.spinner')
               <span x-show="uploadProgress < 100"
x-text="uploadProgress + '%'"></span>
               <span x-show="uploadProgress == 100">
                   Hold on, we are enhancing your photo...
               </span>
           </div>
           <div class="flex max-w-lg justify-center rounded-md
border-2 border-dashed border-gray-300 px-6 pt-5 pb-6">
```

```
                    <div class="space-y-1 text-center">
                        <svg class="mx-auto h-12 w-12 text-gray-400"
stroke="currentColor" fill="none" viewBox="0 0 48 48" aria-hidden="true">
                            <path d="M28 8H12a4 4 0 00-4 4v20m32-12v8m0
0v8a4 4 0 01-4 4H12a4 4 0 01-4-4v-4m32-4l-3.172-3.172a4 4 0 00-5.656 0L28
28M8 32l9.172-9.172a4 4 0 015.656 0L28 28m0 0l4 4m4-24h8m-4-4v8m-12 4h.02"
stroke-width="2" stroke-linecap="round" stroke-linejoin="round"></path>
                        </svg>
                        <div class="flex text-sm text-gray-600"
x-show="!loading">
                            <label for="file-upload" class="relative
cursor-pointer rounded-md bg-white font-medium text-indigo-600
focus-within:outline-none focus-within:ring-2 focus-within:ring-indigo-500
focus-within:ring-offset-2 hover:text-indigo-500">
                                <span>Upload a file</span>
                                <input
                                        class="hidden shadow
appearance-none border rounded w-full py-2 px-3 text-gray-700 leading-tight
focus:outline-none focus:shadow-outline"
                                        id="file-upload"
                                        type="file"
                                        x-on:change="(formData.file =
Object.values($event.target.files)[0]) && submitForm()"
                                >
                            </label>
                            <p class="pl-1">or drag and drop</p>
                        </div>
                        <p class="text-xs text-gray-500">PNG, JPG up to
2MB</p>
                    </div>
                </div>
                <div class="mb-4 text-red-900 rounded"
x-show="formErrors['file']" x-text="formErrors['file'][0]"></div>
            </form>

            <div id="result" class="my-4 bg-white rounded-lg shadow-lg p-6"
:class="{ 'hidden': !result}">
                <div class="flex items-center justify-between mb-4">
                    <h2 class="text-xl font-bold">Restored Photo</h2>
                    <a
                            class="border p-2 border-blue-500 text-blue-500
font-bold rounded focus:outline-none focus:shadow-outline"
                            :href="result"
                            download
                    >
                        Download
                    </a>
                </div>
                <img :src="result" alt="Restored Photo" class="mb-4">
```

```
            </div>
        </div>
        <script>
            function uploadForm() {
                return {
                    loading: false,
                    formData: {
                        email: "",
                        file: "",
                    },
                    result: null,
                    error: null,
                    formErrors: {},
                    uploadProgress: 0,
                    submitForm() {
                        this.loading = true;
                        const data = new FormData()
                        Object.keys(this.formData).map((key, index) => {
                            data.append(key, this.formData[key])
                        });
                        this.error = null;
                        this.formErrors = {};
                        this.result = null;
                        axios.post('/upload', data, {
                            onUploadProgress: (event) => {
                                this.uploadProgress =
Math.round((event.loaded / event.total) * 100);
                            }
                        }).then(response => {
                            this.result = response.data.result;
                        }).catch(error => {
                            this.error = error.response.data.message;
                            if (error.response.status === 422) {
                                this.formErrors =
error.response.data.errors;
                            }
                        }).finally(() => {
                            this.loading = false;
                        })
                    }
                }
            }
        </script>
    </body>
    </html>
```

Додаток В. Програмний код UploadController.php

```
<?php

namespace App\Http\Controllers;

use App\Http\Integrations\Api\Replicate;
use App\Http\Integrations\Replicate\Requests\FetchPrediction;
use App\Http\Integrations\Replicate\Requests\Predictions;
use App\Http\Requests\UploadRequest;
use App\Services\ImageUploader;
use Illuminate\Support\Facades\Pipeline;

class UploadController extends Controller
{
    public function __construct(private readonly Replicate $replicate,
private readonly ImageUploader $uploader)
    {
        //
    }

    public function __invoke(UploadRequest $request)
    {
        $imageUrl =
$this->uploader->uploadAndGetUrl($request->file('file'));

        $outputImageUrl = Pipeline::send($imageUrl)
            ->through([
                $this->initiatePrediction(...),
                $this->checkCompleteness(...),
                $this->uploadRestored(...),
            ])
            ->then(fn($imageUrl) => $imageUrl);
        return [
            'result' => $outputImageUrl,
        ];
    }
    private function initiatePrediction(string $imageUrl, $next)
    {
        $predict = new Predictions($imageUrl);
        $record = $this->replicate->send($predict)->dtoOrFail();
        return $next($record->imageId);
    }

    private function checkCompleteness(string $imageId, $next)
    {
        $fetch = new FetchPrediction($imageId);
```

```
            do {
                sleep(1);
                $restoredImage =
$this->replicate->send($fetch)->dtoOrFail();
            } while ($restoredImage->processing());

            return $next($restoredImage->output);
        }

        private function uploadRestored(string $imageUrl, $next)
        {
            return $next($this->uploader->fetchAndUpload($imageUrl,
'restored'));
        }
    }
```