Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістреський)

Дослідження моделей підвищення якості цифрових зображень

(тема)

Виконав:
студент 6 курсу, групи СШІм-22-1
Батарін М.Д.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма Системи штучного інтелекту
(повна назва спеціальності)
Керівник проф. Смеляков К.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістреський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«» _____ 20 __ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Батаріну Микиті Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження моделей підвищення якості цифрових зображень

затверджена наказом університету від 01 квітня 2024 р. №260Ст
2. Термін подання студентом роботи до екзаменаційної комісії 19 травня 2023 р.
3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі
2) Опис проведених теоретичних досліджень
3) Опис системи, що пропонується
4) Опис проведених експериментальних досліджень

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	01.04.2024	виконано
2	Аналіз предметної галузі	10.04.2024	виконано
3	Огляд існуючих метрик оцінювання якості	15.04.2024	виконано
4	Аналіз підходів вирішення задачі детекції	22.04.2024	виконано
5	Експериментальне моделювання та навчання	05.05.2024	виконано
	моделі		
6	Написання пояснювальної записки	23.05.2024	виконано
7	Підготовка презентації та доповіді	29.05.2024	виконано
8	Захист перед ЕК	14.06.2024	виконано

Дата видачі завдання 1 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Смеляков К.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 62 с., 14 рис., 1 табл., 1 дод., 26 джерел.

ГЛИБИННА НЕЙРОННА МЕРЕЖА, НЕЙРОННІ МЕРЕЖІ, ПОКРАЩЕННЯ ЯКОСТІ, ЦИФРОВІ ЗОБРАЖЕННЯ.

Об’єкт дослідження – завдання покращення якості цифрових зображень.

Предмет дослідження – застосування глибоких нейронних мереж для підвищення якості цифрових зображень.

Мета роботи – порівняти різні моделі для покращення якості цифрових зображень та обрати найефективнішу для подальшої розробки системи. Ця система буде включати в себе розробку архітектури нейронної мережі, її навчання та реалізацію для подальшого використання.

Методи дослідження – теоретичний (аналіз літературних джерел), експериментальний (програмна реалізація нейронних мереж та їх навчання), порівняльний (порівняння ефективності різних готових моделей). Розробка базується на Python технологіях з використанням фреймворку TensorFlow та інших бібліотек для роботи з нейронними мережами, таких як PyTorch, Keras тощо.

У результаті виконання дослідження було порівняно декілька моделей для підвищення якості зображень, а саме: SRGAN, ESRGAN, Deep Image Prior, VDSR . Для цього були обрані певні критерії для оцінки ефективності кожної з них, які дозволили здійснити об’єктивне порівняння. На основі результатів порівняння було визначено найефективнішу модель. Після вибору найкращої моделі був розроблений користувацький інтерфейс, який дозволяє зручно застосовувати обрану модель для покращення якості зображень. Усі ці процеси були реалізовані з використанням мови програмування Python та суміжних бібліотек.

ABSTRACT

Master’s thesis contains: 62 pp., 14 fig., 1 tabl., 1 ann., 26 references.

DEEP NEURAL NETWORK, DIGITAL IMAGES, NEURAL NETWORKS, QUALITY IMPROVEMENT.

The object of research is the task of improving the quality of digital images.

The subject of research is the application of deep neural networks to improve the quality of digital images.

The purpose of the study is to compare different models for improving the quality of digital images and choose the most effective one for further development of the system. This system will include the development of a neural network architecture, its training and implementation for further use.

Research methods: theoretical (analysis of literature), experimental (software implementation of neural networks and their training), comparative (comparison of the effectiveness of different ready-made models). The development is based on Python technologies using the TensorFlow framework and other libraries for working with neural networks, such as PyTorch, Keras, etc.

As a result of the study, several models for improving image quality were compared, namely: SRGAN, ESRGAN, Deep Image Prior, VDSR. For this purpose, certain criteria were selected to evaluate the effectiveness of each of them, which allowed for an objective comparison. Based on the comparison results, the most effective model was identified. After selecting the best model, a user interface was developed that allows the selected model to be conveniently applied to improve image quality. All these processes were implemented using the Python programming language and related libraries.

ЗМІСТ

Вступ 7
1 Моделі підвищення якості зображень 8
1.1 Теоретичний аналіз поняття «підвищення якості» зображень 8
1.2 Принцип роботи моделей підвищення якості зображень 12
1.3 Формування завдань роботи 15
2 Порівняльний аналіз моделей 18
2.1 Оцінка ефективності обраних моделей 18
2.1.1 SRGAN 18
2.1.2 ESRGAN 24
2.1.3 VDSR 26
2.1.4 Deep Image Prior 28
2.2 Формування критеріїв порівняння моделей 30
2.3 Аналіз результатів та вибір найефективнішої моделі 33
3 Програмна реалізація моделі 37
3.1 Вибір інструментів програмної реалізації 37
3.2 Проектування та реалізація користувацького інтерфейсу 39
3.3 Тестування, налагодження та аналіз результатів роботи інтерфейсу 39
Висновки 49
Перелік джерел посилання 50
Додаток А Код програми 53
Додаток Б Відомість кваліфікаційної роботи 62

ВСТУП

В епоху цифрових технологій попит на високоякісні зображення на різних платформах постійно зростає. Від соціальних мереж до веб-сайтів електронної комерції потреба в чітких, детальних і візуально привабливих зображеннях стала першочерговою. Це призвело до сплеску досліджень і розробок, спрямованих на підвищення якості цифрових зображень за допомогою складних алгоритмів і моделей.

Моделі покращення якості цифрових зображень відіграють ключову роль у підвищенні роздільної здатності, різкості та загальної візуальної достовірності зображень. Ці моделі використовують передові технології, такі як надвисока роздільна здатність, зменшення шуму та відновлення зображень, щоб підняти якість цифрових зображень до нових висот. Використовуючи передові технології, такі як глибоке навчання та обробка зображень, дослідники постійно вдосконалюють ці моделі, щоб досягти виняткових результатів. Значення покращення якості зображень виходить за рамки естетики: воно безпосередньо впливає на користувацький досвід, сприйняття бренду та зрозумілість інформації. Покращення зображень товарів для інтернет-магазинів, оптимізація медичних зображень для діагностики чи вдосконалення супутникових знімків для дослідницьких цілей - застосування цих моделей є різноманітним і далекосяжним.

Ця наукова робота має на меті дослідити ефективність різноманітних моделей, виокремлюючи низку методологічних підходів, починаючи від класичних алгоритмів і закінчуючи новітніми нейронними мережами. Завдяки ретельному вивченню, дослідження намагається з’ясувати притаманні їм сильні сторони, обмеження та нові тенденції, що формують постійний пошук шляхів покращення візуальної точності цифрових зображень.

1 МОДЕЛІ ПІДВИЩЕННЯ ЯКОСТІ ЗОБРАЖЕНЬ

1.1 Теоретичний аналіз поняття «підвищення якості» зображень

Поняття «покращення якості зображення» стосується набору методів, спрямованих на покращення візуального сприйняття зображення. Це, по суті, дисципліна обробки зображень, яка використовує різні алгоритми і методи для усунення недоліків зображення і створення більш візуально приємного та інформативного представлення. Дуже важливо визнати суб'єктивну природу якості зображення. Хоча деякі аспекти, такі як зменшення шуму або різкість, можна об'єктивно виміряти, на загальну «якість» покращення зображення часто впливає людське сприйняття і потреби конкретного застосування.

Це розмаїття відносних вимог і вподобань створює виклик для розробників алгоритмів покращення якості зображення, оскільки їм потрібно збалансувати різноманітні аспекти, щоб задовольнити потреби різних кінцевих користувачів [1]. Такі аспекти, як кольори, контрастність та розмір деталізації, можуть варіювати в залежності від контексту застосування.

Наприклад, у медичній графіці або при судово-медичних дослідженнях важливо, щоб покращене зображення точно передавало мікроструктури і дозволяло точну ідентифікацію патологічних змін. З іншого боку, у сфері медіа та маркетингу вирішальним може бути естетичний аспект, такий як привабливість та емоційна складова зображення. Тому врахування суб'єктивного сприйняття і вимог різних категорій користувачів є ключовим аспектом розвитку ефективних алгоритмів покращення якості зображення.

Існують різні фактори, які можуть погіршити якість зображення, що вимагає застосування методів покращення [2]:

– шум. Шум в зображенні може бути викликаний різними причинами, такими як електронний шум в області низької освітленості, артефакти стиснення при збереженні зображення або випадкові зовнішні фактори, такі як вітер, які впливають на фізичні елементи зйомки. Шум може маскувати дрібні деталі та впливати на загальну чіткість зображення, тому його покращення важливо для поліпшення якості (рисунок 1.1, а);

– розмиття. Розмиття зображення виникає внаслідок руху камери, об’єкта зйомки або неправильного фокусування. Це призводить до втрати різкості та деталізації, особливо в областях, де рух є значним. Методи покращення розмиття включають у себе відновлення різкості та видалення артефактів розмиття (рисунок 1.1, б);

– низька контрастність. Зображення з низькою контрастністю має обмежену різницю між світлими і темними ділянками, що призводить до втрати глибини і реалізму. Підвищення контрастності допомагає покращити візуальний ефект і дозволяє краще виділяти деталі на зображенні (рисунок 1.1, в);

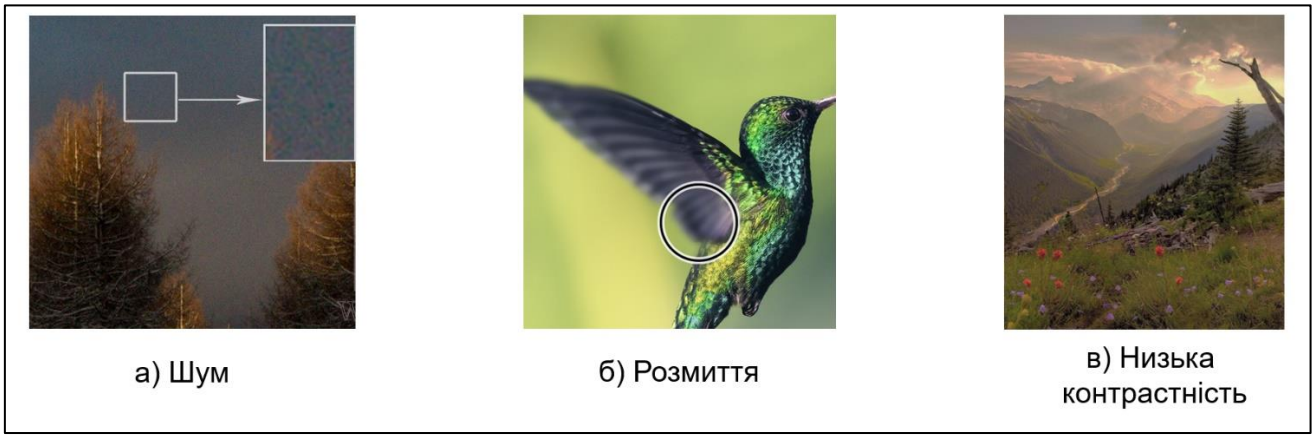

Рисунок 1.1 – Різні фактори погіршення якості зображення

– баланс кольорів. Неправильний баланс кольорів або переважання певних відтінків може призвести до натяків на не справжність зображення або відхилення від природних кольорів. Корекція кольорового балансу та

усунення артефактів кольору може значно покращити сприйняття зображення (рисунок 1.2, а);

– артефакти. Артефакти включають в себе різноманітні небажані елементи, такі як стиснення або обрізання артефактів під час зйомки чи обробки зображення. Ці артефакти можуть спотворювати зображення та знижувати його якість, тому їх усунення є важливим кроком у процесі покращення зображення (рисунок 1.2, б);

– низька роздільна здатність. Недостатня кількість пікселів у зображенні може призвести до втрати деталізації та реалізму, особливо при збільшенні масштабу або при виконанні робіт, які потребують високої роздільної здатності. Використання методів покращення роздільної здатності допомагає забезпечити більш точне відтворення деталей на зображенні (рисунок 1.2, в).

Рисунок 1.2 – Різні фактори погіршення якості зображення

При роботі з обробкою та покращенням зображень існують два основних підходи для оцінки їхньої якості: використання об’єктивних та суб’єктивних метрик. Обидва підходи мають свої переваги та обмеження і використовуються в залежності від конкретних вимог та обставин. В даному

контексті важливо розглянути кожен з цих підходів для кращого розуміння процесу оцінки якості оброблених зображень.

Об’єктивні метрики якості зображень є кількісними міркуваннями, що використовують різноманітні математичні моделі та алгоритми для оцінки рівня якості обробленого зображення. Один з найпоширеніших підходів , використання PSNR (Peak Signal-to-Noise Ratio) [3], який порівнює вхідне та оброблене зображення за допомогою різниці між сигналом і шумом. Інша важлива метрика, SSIM (Structural Similarity Index) [4], яка враховує структурні властивості зображення для визначення рівня схожості між оригінальним та обробленим зображеннями. Перевагою об’єктивних метрик є їхнє автоматичне вимірювання та повторюваність результатів. Вони дозволяють швидко оцінити якість зображення без прив’язки до суб’єктивного сприйняття. Однак вони можуть не враховувати тонкі нюанси людського сприйняття та не враховувати контекст використання зображення.

Суб’єктивні метрики якості зображень використовують людське сприйняття для оцінки якості. Це може бути здійснено за допомогою експертних оцінок або анкетування широкого кола користувачів. Експерти або учасники оцінюють якість зображення з урахуванням різних аспектів, таких як різкість, колір, контрастність та загальна естетика.

Перевагою суб’єктивних метрик є їхня здатність враховувати складні аспекти сприйняття, які не завжди можна врахувати за допомогою об’єктивних методів. Вони дозволяють отримати більш точне уявлення про те, як зображення сприймається кінцевим користувачем. Проте вони можуть займати багато часу та використовувати значну кількість ресурсів, оскільки вимагають участі людей у тестуванні, і оцінки можуть бути суб’єктивними через індивідуальні відмінності сприйняття.

Об’єктивні та суб’єктивні метрики якості зображень є важливими інструментами для оцінки ефективності методів обробки та покращення

зображень. Хоча кожен підхід має свої переваги та обмеження, їх комбінація може забезпечити більш комплексну оцінку якості зображення.

1.2 Принцип роботи моделей підвищення якості зображень

Моделі покращення зображень, особливо ті, що базуються на нейронних мережах, працюють за принципом навчання складних відображень від вхідних зображень до їхніх покращених версій. Ці моделі використовують архітектури глибокого навчання, такі як згорткові нейронні мережі (CNN), для вилучення та вивчення ієрархічних ознак зображень, які потім використовуються для покращення різних аспектів вхідних зображень.

Загальний робочий процес моделей покращення зображень включає наступні кроки [5]:

– вхідне зображення. Процес починається з вхідного зображення, яке потребує покращення. Це зображення може мати низьку роздільну здатність, погану контрастність, шум або інші недоліки;

– виділення елементів. Вхідне зображення подається в модель покращення зображення, яка зазвичай реалізується за допомогою архітектур глибокого навчання. Ці моделі складається з декількох шарів операцій згортки та об’єднання, які слугують для вилучення ієрархічних ознак із вхідного зображення. Ці ознаки охоплюють різні рівні деталізації та абстракції, починаючи від простих країв і текстур до більш складних візерунків і структур;

– функція відображення. Після того, як ознаки вилучені, вони проходять через додаткові шари нейронної мережі, які виконують функцію відображення. Ця функція відображення вчиться трансформувати витягнуті ознаки з вхідного простору у бажаний вихідний простір. Наприклад, у випадку надвисокої роздільної здатності функція відображення має на меті збільшити масштаб вхідного зображення з низькою роздільною здатністю

до версії з вищою роздільною здатністю. В інших завданнях, таких як згладжування або корекція кольору, функція відображення може зосередитися на зменшенні шуму або налаштуванні колірних каналів відповідно;

– покращення. Відображені функції використовуються для створення покращеної версії вхідного зображення. Цей процес покращення передбачає коригування різних атрибутів зображення, таких як різкість, роздільна здатність, контрастність, баланс кольорів і текстура, для отримання візуально привабливого та якісного результату. Для покращення деталей та загального вигляду зображення можуть бути використані такі методи, як деконволюція (відтворення вихідного зображення після проходження через згорткові шари нейронної мережі), апсемплінг (збільшення розміру зображення для покращення його деталізації та роздільної здатності) або механізми уваги (компоненти нейронних мереж, які визначають, на які частини вхідних даних слід звернути особливу увагу під час обробки);

– вихід. Нарешті, покращене зображення генерується як результат роботи моделі. Це вихідне зображення відображає покращення, зроблені моделлю покращення зображення, усуваючи недоліки, присутні у вихідному вхідному зображенні. Вихідне зображення може бути додатково оброблене або проаналізоване залежно від конкретного застосування.

Загалом, робочий процес моделей покращення зображень складається з низки етапів, на яких вхідне зображення проходить процеси вилучення, картографування та покращення для отримання покращеної версії оригінального зображення. Методи глибокого навчання відіграють вирішальну роль у вивченні складних відображень і закономірностей на основі даних, що дозволяє цим моделям досягати чудових результатів у різних завданнях покращення зображень.

Тепер варто розглянути принцип роботи конкретних моделей для кращого розуміння та можливості формування необхідних задач для проведення аналізу моделей.

SRGAN (Super-Resolution Generative Adversarial Network) модель, яка використовується для підвищення роздільної здатності зображень низької якості. Вона використовує дві основні частини: генератор і дискримінатор. Генератор відповідає за перетворення зображень низької роздільної здатності в зображення високої роздільної здатності. Він навчається створювати детальніші зображення, використовуючи навчальні дані. Дискримінатор служить для визначення того, наскільки реалістичними виглядають зображення, створені генератором. Він навчається розрізняти між реальними зображеннями високої роздільної здатності та зображеннями, створеними генератором.

Під час навчання генератора і дискримінатора працюють разом у змагальному режимі. Генератор намагається виглядати якомога більш реалістичним, створюючи зображення високої якості, тоді як дискримінатор намагається розрізнити між справжніми та синтезованими зображеннями. Цей процес триває до того моменту, поки генератор не стане достатньо добрим у виробленні реалістичних зображень. В результаті SRGAN може створювати високоякісні, деталізовані зображення з низької роздільною здатністю [6].

ESRGAN це покращена версія SRGAN, яка використовується для підвищення роздільної здатності зображень. Вона використовує архітектуру глибокого навчання GAN (генеративної змагальної мережі). Головна ідея ESRGAN полягає в тому, щоб навчити модель використовувати два види втрат: традиційні втрати на рівні пікселів і втрати сприйняття на високому рівні. Втрати сприйняття враховують високорівневі аспекти зображення, які важливі для людського сприйняття, за допомогою попередньо навчених 19 глибоких нейронних мереж [7]. Під час навчання ESRGAN модель намагається максимально відтворити не лише деталі зображення, а й його загальну візуальну якість, враховуючи особливості сприйняття людьми. Це дозволяє досягти більш реалістичних та природних результатів у підвищенні роздільної здатності зображень.

Deep Image Prior це архітектура мережі, яка використовує особливості вхідних даних, а не навчання на підтверджених даних, для вирішення завдань обробки зображень. На відміну від багатьох глибоких нейронних мереж, що вимагають великої кількості навчальних даних, Deep Image Prior використовує архітектурні особливості, щоб управляти процесом відновлення зображень. Замість того, щоб оптимізувати параметри мережі на основі даних, Deep Image Prior використовує саму мережу для генерації зображень, які відповідають вхідним даним [8]. Це відбувається шляхом випадкової ініціалізації параметрів мережі і їх оптимізації таким чином, щоб максимізувати подібність між вихідними даними та відновленими зображеннями.

VDSR мережа глибокого навчання, спеціально призначена для вирішення завдань збільшення яксоті зображень. Основна ідея полягає у використанні нейронної мережі з багатьма шарами для відновлення деталей у зображеннях високої роздільної здатності. VDSR використовує конволюційні шари для виявлення візуальних особливостей на різних рівнях складності та додавання глибини до мережі, що дозволяє вирішувати більш складні завдання покращення якості. Крім того, використовується метод зворотного розповсюдження помилок для навчання мережі на парах зображень низької та високої роздільної здатності [9].

1.3 Формування завдань роботи

У сфері покращення зображень існує багато моделей, що створює проблеми для фахівців, оскільки важко визначити найбільш ефективне рішення. Порівняння цих моделей має вирішальне значення, оскільки воно допомагає визначити, яка модель найкраще відповідає конкретним вимогам завдання чи програми. Оцінка та вибір найефективнішої моделі дозволяє забезпечити оптимальну продуктивність та результати в області покращення зображень.

Доречність порівняння моделей полягає в наступному:

– оцінка ефективності: порівняння дозволяє з’ясувати сильні та слабкі сторони кожної моделі у покращенні зображень;

– оптимізація: вибір найефективнішої моделі сприяє оптимізації розподілу ресурсів, зокрема, обчислювальних ресурсів та часу;

– забезпечення якості: порівняння допомагає забезпечити якість результатів, визначаючи модель, яка генерує найякісніші покращені зображення;

– економічна ефективність: вибір найефективнішої моделі може призвести до економії коштів за рахунок мінімізації потреби в додатковій обробці;

– інновації: порівняння моделей стимулює інновації та вдосконалення існуючих рішень у галузі покращення зображень.

Для виконання поставленої мети в роботі, необхідно реалізувати наступні завдання:

– провести дослідження ефективності моделей, зібрати дані про продуктивність кожної моделі, включаючи якість покращення зображень та обчислювальну ефективність;

– порівняти та проаналізувати результати дослідження, виділивши сильні та слабкі сторони кожної моделі;

– створити критерії оцінювання, для оцінки ефективності кожної моделі на основі якості зображення, швидкості обробки та використання ресурсів. Визначити чіткі та вимірювані показники для об’єктивної оцінки моделей;

– проаналізувати принцип роботи найбільш ефективної системи. Вивчити принцип роботи найбільш ефективної моделі, зосередившись на її архітектурі та методах оптимізації;

– розробити користувацький інтерфейс: створити зручний інтерфейс для взаємодії з обраною моделлю, що включає функції введення/виведення зображень та візуалізацію результатів;

– провести тестування та налагодження, для підтвердження його функціональності та надійності. Виявити та вирішити будь-які проблеми або помилки через систематичне тестування та оптимізацію.

Виконання цих завдань дозволить ефективно порівняти моделі покращення зображень, вибрати найбільш підходящий варіант та розробити надійне програмне рішення.

2 ПОРІВНЯЛЬНИЙ АНАЛІЗ МОДЕЛЕЙ

2.1 Оцінка ефективності обраних моделей

Перш ніж почати порівняння та формулювання критеріїв для оцінки обраних моделей, важливо зрозуміти, як працює кожна з них. Аналіз робочих механізмів, сильних і потенційних слабких сторін цих моделей на кожному етапі процесу є ключовим. Таке розуміння закладе основу для обґрунтованої оцінки, допомагаючи визначити конкретні аспекти, в яких одна модель може мати переваги або недоліки порівняно з іншими. Такий підхід забезпечує всебічну та об’єктивну оцінку ефективності кожної моделі у відповідних сферах застосування.

2.1.1 SRGAN

Для навчання моделі SRGAN на вході обирається набір даних з фотографіями низької роздільної здатності, а відповідні зображення високої роздільної здатності використовуються як цільовий вихід.

Таке налаштування необхідне для навчання компонента генератора SRGAN для ефективного масштабування зображень з низькою роздільною здатністю. Під час навчання генератор вчиться створювати зображення високої роздільної здатності з зображень низької роздільної здатності. Тим часом дискримінатор оцінює якість покращених зображень порівняно з оригінальними зображеннями високої роздільної здатності. Цей процес допомагає моделі навчитися генерувати більш реалістичні та якісні зображення надвисокої роздільної здатності.

Такий процес у моделі керується Генеративно-Змагальною Мережею (Generative Adversarial Network, GAN) [10] (рисунок 2.1). GAN являє собою клас алгоритмів машинного навчання, в якому дві мережі, генератор та дискримінатор, змагаються одна з одною, щоб відповідно генерувати дані,

які неможливо відрізнити від реальних, та розрізнити справжні дані від синтетичних.

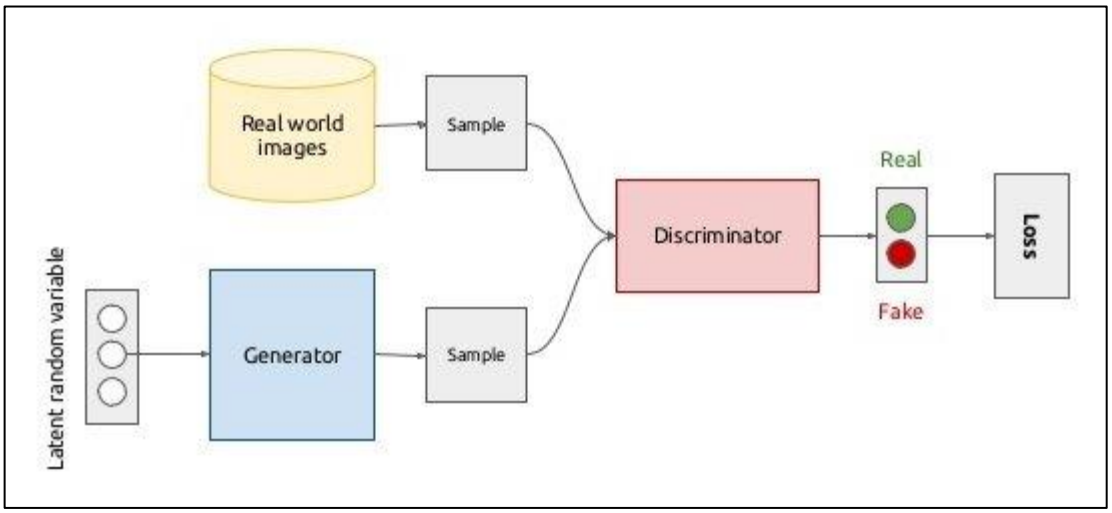

Рисунок 2.1 – Архітектура мережі GAN [11]

Основними компонентами в архітектурі GAN є:

– генератор (Generator): це нейронна мережа, яка генерує нові зразки даних, намагаючись імітувати реальні зображення. Вона отримує в якості вхідних даних вектор зі стохастичних (випадкових або непередбачуваних значень і виводить зображення;

– дискримінатор (Discriminator): це друга нейронна мережа, яка отримує зразки як від генератора, так і реальні зображення. Її завдання розрізнити справжні зображення від фальшивих (генерованих);

– справжні зображення (Real World Images): це датасет реальних зображень, на яких навчається дискримінатор. Він використовується для порівняння з зображеннями, створеними генератором;

– втрата (Loss): це метрика, яка вимірює, наскільки добре дискримінатор здатний розрізняти справжні зображення від фальшивих, та наскільки переконливі зображення, створені генератором. Ця втрата використовується для оновлення вагів обох мереж під час навчання.

Так як GAN це тільки частина моделі SRGAN варто розглянути і архітектуру самої моделі, вона зображена на рисунку 2.2.

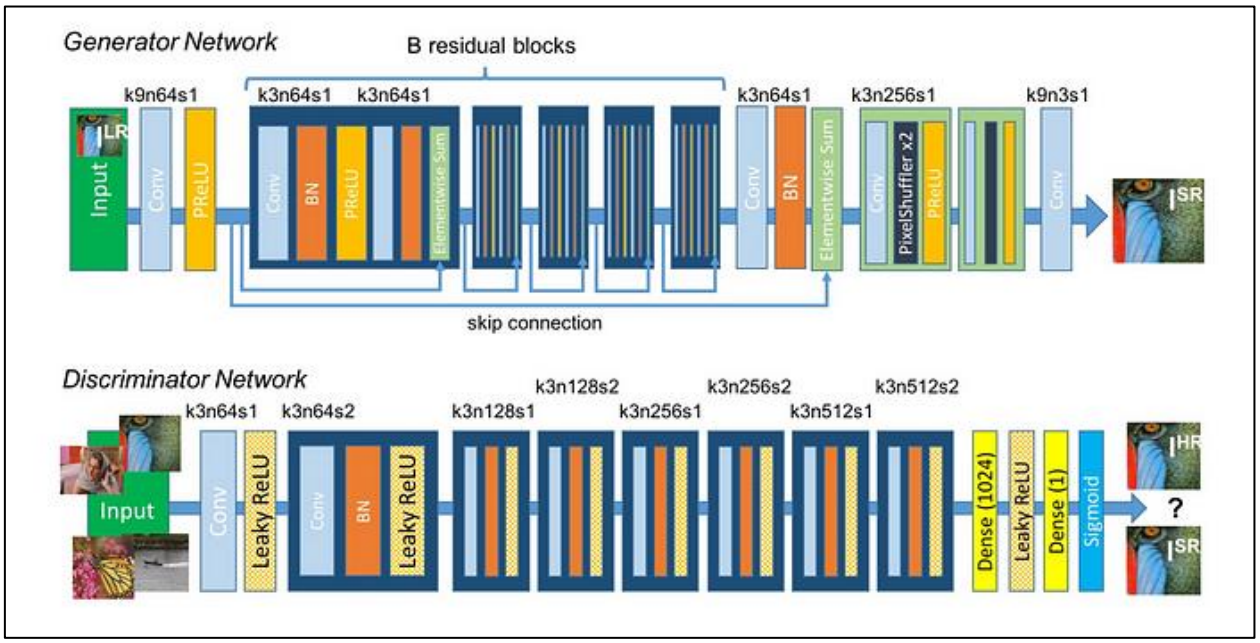

Рисунок 2.2 – Архітектура моделі SRGAN [12]

Тут генераторна мережа використовується для підвищення якості зображень з низькою роздільною здатністю (LR), таких як зображення з веб-камер або мобільних телефонів. У генераторній мережі використовуються такі елементи:

– вхід: зображення з низькою роздільною здатністю, наприклад, 100x100 пікселів;

– первинний конволюційний шар: це перший крок обробки зображення. Він використовує фільтри розміром 9x9 для виявлення особливостей зображення та генерує 64 різних характеристики для подальшої обробки;

– в блоки з відсталим з'єднанням: ці блоки допомагають мережі вивчити корисні представлення зображення на високому рівні, уникнувши втрати інформації. Кожен блок містить два конволюційних шари з пакетною нормалізацією та активацією PReLU (Parametric Rectified Linear Unit

параметризований випрямлений лінійний блок [13]. Це активаційна функція, яка використовується в нейронних мережах для додавання нелінійності в модель). Вони дозволяють зберегти деталі з вхідного зображення;

– пропускне з’єднання: це з’єднання передає вхідні дані безпосередньо до виходу блоку, зберігаючи деталі зображення;

– підвищення роздільної здатності (Upsample): зображення збільшується в розмірі, використовуючи конволюційні шари з більшим кроком, щоб отримати зображення (HR), наприклад, 400x400 пікселів.

Дискримінаторна мережа використовується для відмінення між справжніми зображеннями (HR) і зображеннями, що згенеровані генератором (ISR). Основні елементи дискримінаторної мережі включають:

– вхід: зразки зображень високої роздільної здатності (HR) та збільшені зображення (ISR);

– конволюційні шари: ці шари використовуються для виявлення характеристик зображення. Вони використовують фільтри різного розміру та кількості, щоб визначити, чи є зображення справжнім HR або згенерованим ISR. Leaky ReLU (функція, для додавання нелінійності в модель) та пакетна нормалізація (метод, який використовується для нормалізації входів нейронної мережі) допомагають у підвищенні ефективності;

– вищі шари (повнозв’язні шари): після декількох шарів конволюції характеристики зображення конвертуються в більш високорівневі представлення за допомогою повнозв’язних шарів;

– вихід: останній шар видає скалярну оцінку, яка показує, чи є зразок справжнім HR зображенням чи згенерованим ISR.

Генераторна та дискримінаторна мережі навчаються одночасно через змагальний процес, де генератор намагається створити все більш правдоподібні зображення, а дискримінатор намагається відрізнити

справжні HR зображення від згенерованих ISR. Це дозволяє покращувати якість згенерованих зображень.

Результатом тренування буде згенероване зображення (рисунок 2.3). На зображенні продемонстровано три версії одного і того ж кадру: починаючи з лівого зображення з низькою роздільною здатністю (Low Resolution), через згенероване зображення (Generated), створене моделлю SRGAN, до оригінального зображення (Original) з права.

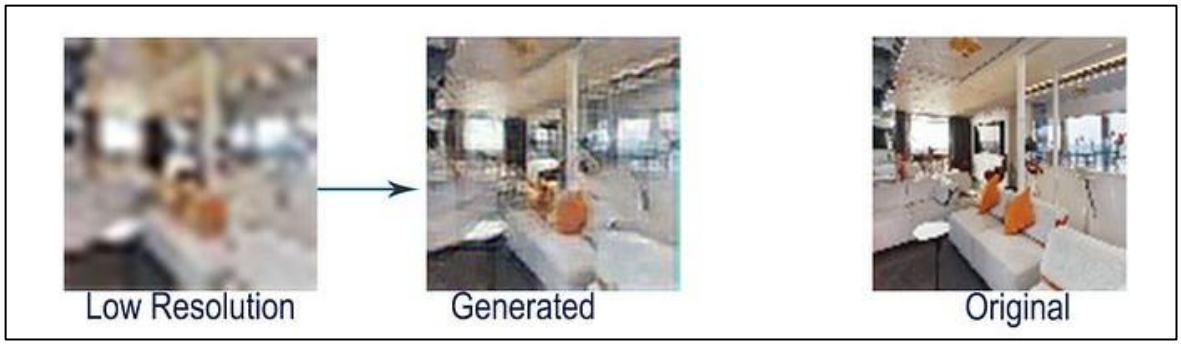

Рисунок 2.3 – Результат роботи моделі SRGAN

Модель була навчена лише на 25 000 зображеннях, що є відносно невеликим набором даних для завдань глибокого навчання, особливо у контексті супер-резолюції. Згенероване зображення виглядає значно чіткішим та деталізованим, ніж зображення з низькою роздільною здатністю, і покращення якості є очевидним. Однак, якщо придивитися до порівняння згенерованого та оригінального зображень, можна помітити деякі відмінності в деталях і текстурах, хоча загальне враження від згенерованого зображення досить близьке до оригіналу.

Судячи з цього прикладу, SRGAN здатна відновлювати багато деталей, навіть навчаючись на відносно обмеженому наборі даних. Це підкреслює ефективність генеративно-змагальних мереж у застосуванні до завдань супер-резолюції.

Наприклад результат зображений на рисунку 2.4. Зображення демонструє порівняння між високоякісним зображенням, згенерованим за допомогою моделі SRGAN з чотириразовим збільшенням роздільної здатності (4x SRGAN), та оригінальним зображенням. Модель була навчена на великому та різноманітному наборі даних з ImageNet із 350 тисяч зображень, можна оцінити наскільки добре модель виконує супер-резолюцію проаналізувавши даний приклад.

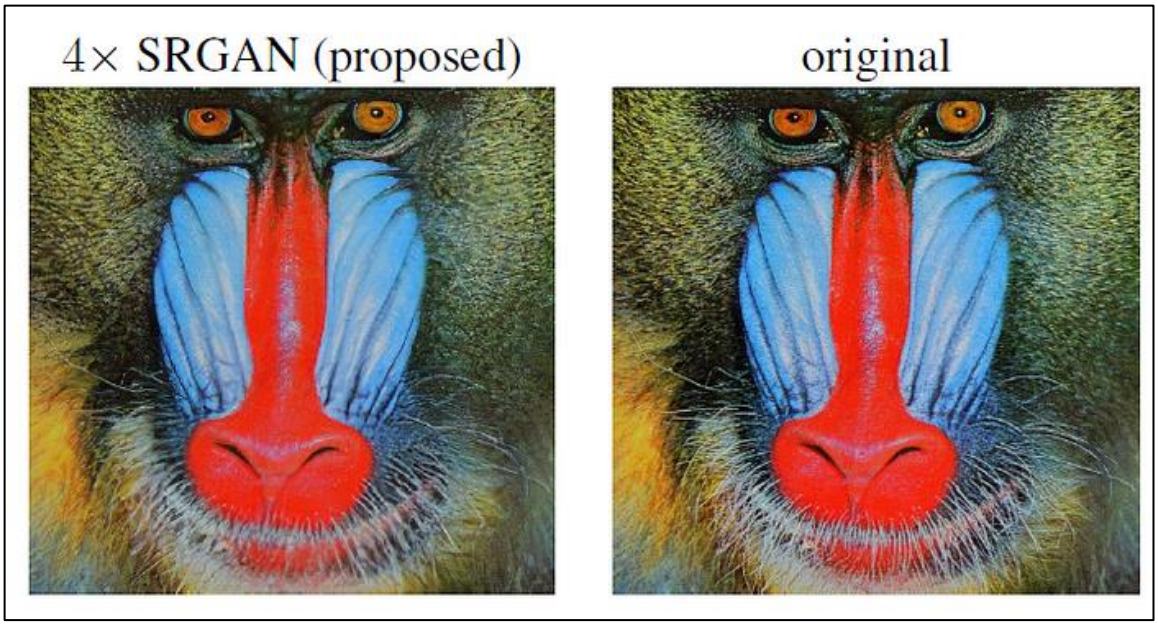

Рисунок 2.4 – Результат роботи моделі SRGAN

Зліва зображення, створене моделлю, яке візуально майже не відрізняється від оригінального зображення справа. Відновлення чіткості, текстур і кольорів заслуговує на увагу, особливо з огляду на те, що вихідне зображення мало в чотири рази менше пікселів. Це свідчить про те, що коли модель навчається на великій кількості якісних даних та виконується на потужному обчислювальному обладнанні, вона може досягати результатів, що дуже близькі до оригінальних зображень, навіть при значному збільшенні роздільної здатності.

2.1.2 ESRGAN

ESRGAN (Enhanced Super-Resolution Generative Adversarial Network) вдосконалена нейромережева архітектура, призначена для покращення роздільної здатності зображень. Вона є розвитком попередньої SRGAN. Як і SRGAN, ESRGAN складається з двох основних компонентів генератора та дискримінатора. ESRGAN запроваджує модифіковану структуру залишкових блоків без шарів пакетної нормалізації, що допомагає запобігти втраті гнучкості діапазону в мережі.

Ця модифікація базується на спостереженні, що пакетна нормалізація має тенденцію до внесення артефактів у процес надвисокої роздільної здатності. Функція втрат в ESRGAN поєднує в собі втрати контенту (для точного відтворення вихідного зображення), втрати, пов’язані з суперництвом (для реалістичності зображення), і втрати (які вимірюються за тим, наскільки схожі зображення з точки зору сприйняття, а не просто точності пікселів). Це допомагає генератору створювати більш привабливі текстури та деталі для людського ока. Процес навчання в ESRGAN складається з двох етапів. Спочатку генератор тренується з втратою середньоквадратичної помилки (MSE) для стабільності, а потім проводиться тонка настройка з урахуванням втрат, пов’язаних з суперництвом і сприйняттям. Цей підхід допомагає досягти як високої точності, так і вражаючої якості зображень [14], [15].

Щодо архітектури, ESRGAN використовує схожу за принципом роботи архітектуру, але глибшу ніж SRGAN, що дозволяє здійснювати більш складні перетворення зображень. Ця мережа складається з блоків залишкового з’єднання (RRDB) (рисунок 2.5), які допомагають ефективно передавати інформацію низької та високої частот через мережу.

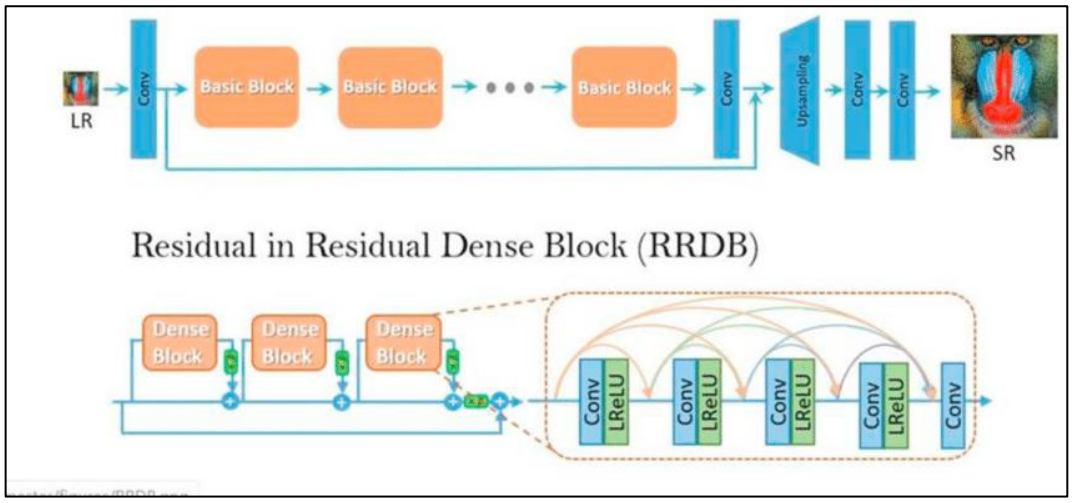

Рисунок 2.5 – Архітектура блоку Residual in Residual Dense Block (RRDB)

RRDB є інноваційною частиною ESRGAN, що допомагає покращувати роздільну здатність зображень із збереженням їх деталей. Ось детальний опис RRDB [16]:

– щільний блок (Dense Block): кожен щільний блок містить декілька конволюційних шарів (Conv), де кожен шар отримує вхідні дані від усіх попередніх шарів у блоку. Це створює щільну мережу зв’язків, де вхідні дані і характеристики з кожного шару передаються до наступних шарів, забезпечуючи багатий потік інформації;

– конволюційні шари з активацією ReLU: кожен конволюційний шар в щільному блоку застосовується з активаційною функцією ReLU. Функція ReLU допомагає моделі вчитися швидше і зменшує ймовірність проблеми зникнення градієнтів;

– залишкові з’єднання: в RRDB використовуються залишкові з’єднання між шарами, а також на вході та виході кожного блоку. Залишкові з’єднання дозволяють передавати інформацію безпосередньо через блок, що допомагає уникнути втрати інформації при проходженні через глибокі шари мережі;

– послідовність RRDB: У ESRGAN кілька RRDB послідовно з’єднані один за одним. Це створює глибоку архітектуру, яка може здійснювати

складні перетворення зображень і відновлювати деталі з низькорівневих на високорівневі представлення.

Така структура RRDB в ESRGAN дозволяє моделі ефективно обробляти інформацію та відтворювати зображення з високою роздільною здатністю з низькорівневого зображення. Це досягається завдяки використанню щільних та залишкових з'єднань для забезпечення потоку характеристик та запобігання їх втратам у глибокій мережі.

2.1.3 VDSR

VDSR (Very Deep Super Resolution), архітектура нейронної мережі, розроблена для завдання надвисокої роздільної здатності зображень. Основна мета надвисокої роздільної здатності, відновити зображення з високою роздільною здатністю на основі вхідних даних з низькою роздільною здатністю [17]. VDSR застосовує підхід глибокого навчання, використовуючи дуже глибоку згорткову нейронну мережу (CNN) для вивчення відповідності між зображеннями низької та високої роздільної здатності.

Використовуючи стратегію залишкового навчання, щоб зосередитися на вивченні залишків або відмінностей між зображеннями низької і високої роздільної здатності. Це базується на спостереженні, що зображення низької та високої роздільної здатності мають багато спільного, тому вивчення лише залишків може бути більш ефективним [18]. Архітектура зазвичай складається з 20 або більше згорткових шарів (рисунок 2.6), для надвисокої роздільної здатності. Завдяки більшій кількості шарів VDSR може вивчати більш складні зображення і досягати вищої точності при реконструкції зображень високої роздільної здатності.

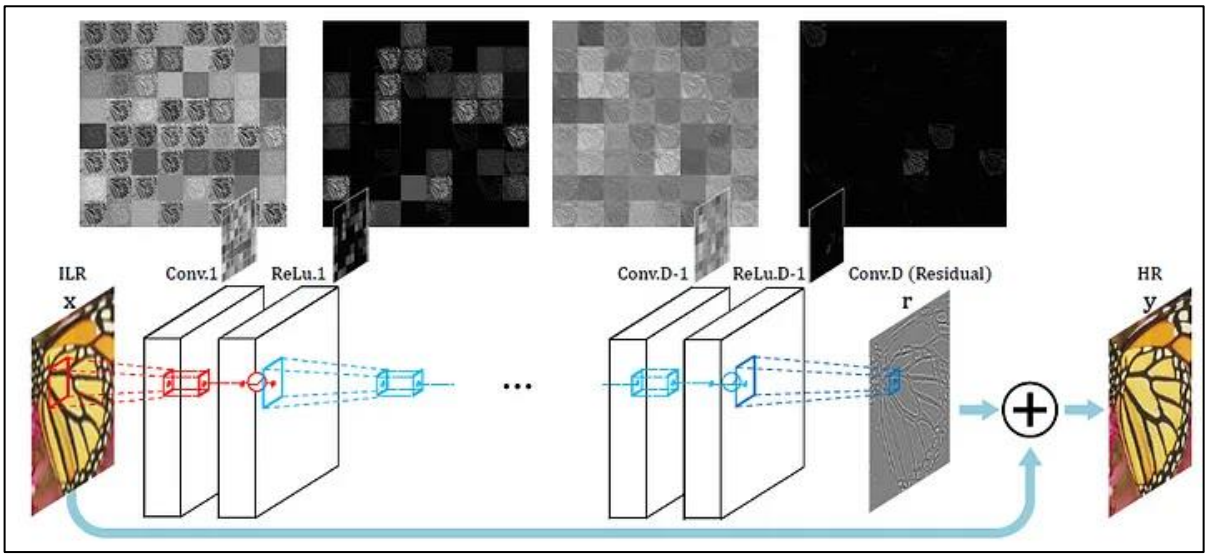

Рисунок 2.6 – Архітектура VDSR [19]

Архітектура містить наступні ключові елементи:

– ILR (Input Low Resolution) x: процес починається з вхідного зображення низької роздільної здатності;

– згортковий та ReLU шари: зображення потрапляє в нейронну мережу, проходячи через серію згорткових шарів (Conv), за якими слідують функції активації випрямлених лінійних одиниць (ReLU). Ці шари призначені для виділення ознак і вивчення ієрархії зображень з вхідних даних низької роздільної здатності;

– проміжні ознаки: коли зображення проходить через мережу, воно трансформується кожним шаром, отримуючи більш детальні ознаки на кожному кроці. Ілюстровані точки (...) вказують на багаторазове повторення шарів згортки та ReLU, що характерно для «дуже глибокої» природи VDSR;

– Conv-D (залишковий): наприкінці використовується шар згортки для виведення залишкового зображення «r». Залишкове зображення представляє різницю, яку потрібно вивчити, щоб перетворити зображення з низькою роздільною здатністю у зображення з високою роздільною здатністю;

– HR (High Resolution): останнім кроком в архітектурі є додавання залишкового зображення до вхідних даних низької роздільної здатності. Це важливий аспект підходу VDSR; він використовує цю стратегію навчання за залишковим принципом для уточнення деталей збільшеного зображення, ефективно створюючи вихідний результат «HR» з високою роздільною здатністю.

Зосереджуючись на вивченні залишків, а не всього зображення з високою роздільною здатністю, VDSR може ефективніше вивчати функцію масштабування, а також дає можливість застосовувати ту саму модель до кількох коефіцієнтів масштабування за допомогою однієї навченої мережі. Це може призвести до покращення якості зображення з дрібнішими деталями та точнішим відтворенням текстур.

2.1.4 Deep Image Prior

Deep Image Prior (DIP) це новаторська концепція в галузі комп’ютерного аналізу зображень та їх обробки. На відміну від традиційних методів глибокого навчання, які покладаються на великі набори даних для навчання, DIP використовує структуру, притаманну самому вхідному зображенню, для виконання різних завдань, таких як відновлення зображень, розфарбовування, знебарвлення та надвисока роздільна здатність. Розроблений у 2017 році, DIP являє собою зміну парадигми глибокого навчання, пропонуючи незалежний від даних підхід до реконструкції та покращення зображень [20]. В основі Deep Image Prior лежить архітектура згорткової нейронної мережі (CNN) (рисунок 2.7), яка зазвичай базується на структурах, подібних до автокодерів. Ключова інновація полягає у відсутності будь-якого явного набору даних для навчання. Замість цього мережа ініціалізується випадковими вагами та оптимізується для мінімізації обраної функції втрат по відношенню до самого вхідного зображення.

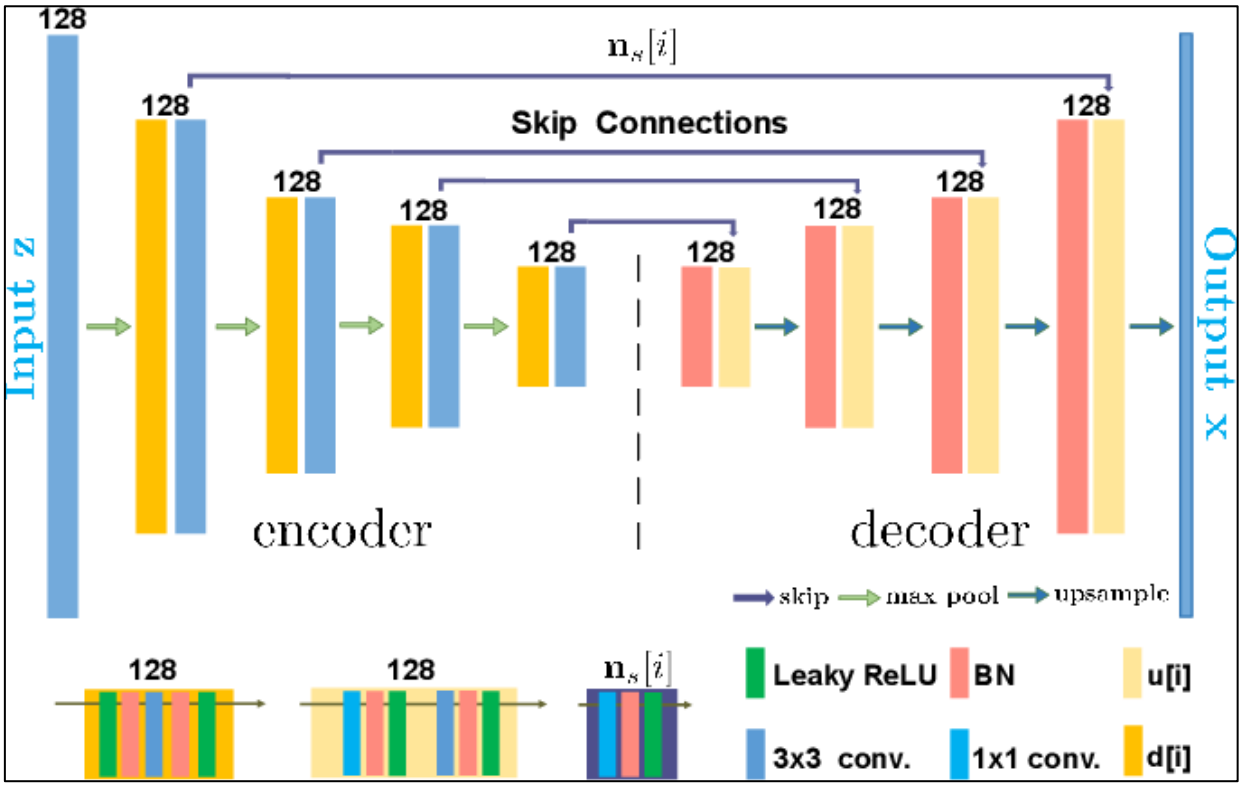

Рисунок 2.7 – Архітектура DIP [21]

Основними компонентами в цій архітектурі є:

– кодер (Encoder): ця частина мережі, складається з послідовності згорткових блоків, кожен з яких містить шари згортки 3x3, за якими йдуть шари максимального об’єднання або згорткові шари для зменшення розмірів простору. Шари згортки можуть містити активаційну функцію, наприклад Leaky ReLU, і шар нормалізації, наприклад Batch Normalization (BN), які допомагають стабілізувати навчання та покращити екстракцію ознак;

– декодер (Decoder): відповідна частина мережі відновлює просторовий розмір вхідного зображення через шари з підвищенням вибірки, які можуть бути реалізовані через транспоновані згортки або методи інтерполяції. Кожен блок декодера також містить згорткові шари, які прагнуть відновити деталі зображення та його високу роздільну здатність;

– пропускні з’єднання (Skip Connections): з’єднання, які прямо передають ознаки з кодера до відповідного шару декодера, запобігають втраті інформації та дозволяють мережі зберегти тонкі деталі, що є особливо корисним при задачах відновлення зображення;

– функція втрат (Loss Function): функція втрат може варіюватися в залежності від завдання. Наприклад, MSE може використовуватись для задач, де важлива точність пікселів, а функції, які оцінюють сприйняття, можуть бути корисними для відновлення зображення, щоби воно було більш приємним для людського ока.

Підсумовуючи, Deep Image Prior являє собою революційний підхід до обробки зображень, пропонуючи незалежне від даних рішення для різних завдань. Використовуючи внутрішню інформацію, що міститься у вхідному зображенні, DIP відкриває нові шляхи для реконструкції, покращення та генерації зображень з перспективними застосуваннями в різних сферах.

2.2Формування критеріїв порівняння моделей

При роботі з моделями машинного навчання формування критеріїв для їх порівняння є важливим кроком, який суттєво впливає на ефективність і застосовність моделей до реальних проблем. Ці критерії, по суті, є набором метрик і показників, які дозволяють оцінювати і порівнювати продуктивність різних моделей у структурований і об'єктивний спосіб. Для порівняння та оцінки моделей були обрані наступні критерії:

– якість зображення: вимірюється такими показниками, як PSNR (пікове відношення сигнал/шум) і SSIM (індекс структурної подібності), що вказують на точність оригінальних зображень високої роздільної здатності;

– якість сприйняття: оцінюється людьми-спостерігачами або метриками, що базуються на сприйнятті, для оцінки візуальної привабливості зображень;

– здатність до узагальнення: здатність добре працювати на зображеннях, які відрізняються від навчального набору даних;

– стійкість до шуму: ефективність обробки зображень з різним рівнем шуму;

– час навчання: час, необхідний для навчання моделі до збіжності;

– час виведення: час, необхідний моделі для обробки зображення і виведення результату на етапі тестування;

– розмір моделі: кількість параметрів у моделі, яка впливає на використання пам'яті та обчислювальні вимоги;

– ефективність використання ресурсів: оцінка продуктивності моделі відносно використаних обчислювальних ресурсів і пам'яті;

– масштабованість: наскільки добре модель може масштабуватися до різних розмірів зображень і наборів даних;

– перенесення: легкість, з якою модель може бути адаптована або доопрацьована до суміжних завдань;

– легкість навчання: наскільки просто навчати модель, включаючи необхідність налаштування гіперпараметрів;

– ефективність даних: обсяг даних, необхідний для ефективного навчання моделі;

– швидкість збіжності: наскільки швидко продуктивність моделі покращується в процесі навчання;

– роздільна здатність на виході: максимальна роздільна здатність зображень, які може генерувати модель;

– стабільність: послідовність роботи моделі на різних прогонах або підмножинах даних;

– адаптивність до різних областей: наскільки добре модель працює в різних областях зображень (наприклад, медичні зображення, супутникові знімки);

– представлення характеристик: якість та інтерпретованість ознак, які вивчила модель;

– тонкий контроль: здатність налаштовувати і контролювати рівень деталізації або аспекти процесу генерації зображень;

– спільнота та підтримка: обсяг та якість документації, підтримка спільноти та простота використання моделі;

– здатність відновлення фото: оцінка того, наскільки ефективно модель може відновлювати деталі та якість зображення після деградації (наприклад, через стиснення або старіння);

– усунення артефактів: здатність моделі виявляти та виправляти типові артефакти в зображеннях, такі як розмиття, jpeg-квантування, або шум від датчика камери.

З врахуванням даних критерії було створено таблицю порівняння (таблиця 2.1) з оцінками від 1 до 5, де:

– 1 означає недостатньо (модель показує погані результати відносно даного критерію);

– 2 означає посередньо (модель має певні обмеження і не завжди відповідає очікуванням;

– 3 означає добре (модель виконує базові вимоги даного критерію);

– 4 означає дуже добре (модель переважно відповідає високим стандартам);

– 5 означає відмінно (модель є лідером у даній категорії і задає стандарти для інших);

Таблиця 2.1 – Порівняння моделей

Критерій	SRGAN	ESRGAN	VDSR	Deep Image Prior
1	2	3	4	5
Якість зображення	4	5	5	4
Якість сприйняття	4	5	2	3
Узагальнююча здатність	3	4	4	3
Стійкість до шуму	2	3	1	2
Час навчання	4	5	2	3
Час виведення	3	4	1	2

Продовження таблиці 2.1

1	2	3	4	5
Розмір моделі	4	5	1	2
Ефективність використання ресурсів	3	4	5	4
Масштабованість	3	4	5	3
Переносимість	2	3	4	2
Простота навчання	2	3	5	4
Ефективність використання даних	2	3	4	3
Швидкість зближення	3	4	5	4
Роздільна здатність вихідного сигналу	4	5	3	4
Стійкість	3	4	5	3
Адаптивність	3	4	2	2
Представлення ознак	4	5	2	3
Тонкий контроль	2	3	1	1
Спільнота та підтримка	4	5	3	2
Можливість відновлення фото	3	4	1	3
Видалення артефактів	3	4	1	2

2.3Аналіз результатів та вибір найефективнішої моделі

В попередньому розділі кожна модель була оцінена за кількома аспектами, починаючи від якості зображення і сприйняття і закінчуючи часом навчання та ефективністю використання ресурсів. Необхідно детальніше розглянути отримані оцінки моделей щоб обрати найефективнішу з них:

– якість зображення: ESRGAN і VDSR вирізняються найвищими показниками якості зображення, обидві системи отримали оцінку «5». Вони створюють візуально привабливі та високоякісні зображення, що є важливим для покращення зображень, завантажених користувачами;

– якість сприйняття: SRGAN та ESRGAN отримали найвищі бали за якість сприйняття, що свідчить про їхню здатність покращувати зображення

у приємний для сприйняття спосіб. Однак ESRGAN дещо випереджає SRGAN у цьому аспекті;

– здатність до узагальнення: SRGAN і VDSR демонструють хорошу узагальнюючу здатність, що свідчить про те, що вони можуть ефективно покращувати різноманітні зображення за межами навчальних даних. Це має вирішальне значення для забезпечення стабільної продуктивності на різних вхідних зображеннях;

– стійкість до шуму: хоча жодна з моделей не вирізняється стійкістю до шуму, SRGAN і VDSR демонструють дещо кращі показники порівняно з ESRGAN і Deep Image Prior. Це означає, що вони краще пристосовані для обробки вхідних зображень з дефектами шуму;

– час навчання: ESRGAN та SRGAN є найкращими з точки зору ефективності часу навчання, що дозволяє швидше навчати модель. Це вигідно для масштабування системи та зменшення обчислювальних витрат;

– час на виході: SRGAN і ESRGAN демонструють чудову ефективність у швидкій генерації вихідних зображень, що робить їх придатними для застосувань у реальному часі, де швидкість має вирішальне значення;

– розмір моделі: ESRGAN і SRGAN мають відносно більші розміри моделей у порівнянні з VDSR і Deep Image Prior. Хоча більший розмір моделі потенційно може призвести до кращої продуктивності, він також може створювати проблеми з розгортанням і споживанням ресурсів;

– ефективність використання ресурсів: VDSR виявляється найбільш ресурсоефективною моделлю, отримавши найвищий бал за цим критерієм. Вона забезпечує баланс між складністю моделі та обчислювальними ресурсами, що робить її придатною для розгортання в середовищах з обмеженими ресурсами;

– масштабованість: ESRGAN та VDSR отримали найвищі бали за масштабованість, що свідчить про їхню здатність ефективно масштабуватися для задоволення зростаючих потреб у робочому

навантаженні. Це має вирішальне значення для ефективної обробки великого обсягу завантажених користувачем зображень;

– переносимість: Deep Image Prior і SRGAN демонструють вищу портативність завдяки меншим розмірам моделей і потенційно простішій архітектурі. Це полегшує їх розгортання на різних платформах і середовищах;

– простота навчання: Deep Image Prior виділяється як найбільш зручна для користувача модель, отримавши найвищий бал за простотою навчання. Це свідчить про те, що розробникам і користувачам відносно легше зрозуміти і впровадити її;

– ефективність використання даних: ESRGAN і SRGAN демонструють хороші результати в ефективному використанні наявних даних для навчання, що свідчить про їхню здатність витягувати значущу інформацію з обмежених наборів даних;

– швидкість збіжності: SRGAN та ESRGAN демонструють швидшу швидкість збіжності, що означає швидке досягнення оптимальної продуктивності під час навчання;

– роздільна здатність вихідного сигналу: ESRGAN і VDSR досягають вищої роздільної здатності вихідного сигналу, що призводить до більш чітких і детальних покращених зображень;

– стабільність: SRGAN і ESRGAN демонструють більшу стабільність своєї роботи при різних вхідних умовах і наборах даних;

– адаптивність: SRGAN і VDSR демонструють кращу адаптивність, що свідчить про їхню здатність пристосовуватися до різноманітних завдань і сценаріїв покращення зображень;

– представлення особливостей: ESRGAN і VDSR демонструють високу точність представлення об'єктів на покращених зображеннях, зберігаючи важливі деталі і структури;

– точний контроль: ESRGAN і SRGAN пропонують кращий тонкий контроль над процесом покращення, дозволяючи користувачам налаштовувати параметри для отримання індивідуальних результатів;

– спільнота та підтримка: SRGAN і ESRGAN мають потужну спільноту і широку підтримку, що може бути корисним для усунення помилок і постійного вдосконалення;

– можливість відновлення фотографій і видалення артефактів: ESRGAN і SRGAN демонструють чудову продуктивність у відновленні фотографій і видаленні артефактів, покращуючи загальну якість вихідних зображень.

Враховуючи комплексний аналіз за різними критеріями, ESRGAN виявляється найефективнішою моделлю для покращення зображень. Модель вирізняється високою якістю зображення і сприйняття, здатністю до узагальнення, ефективністю навчання і адаптивністю, а також пропонує тонкий контроль і потужну підтримку спільноти. Надійна продуктивність ESRGAN у багатьох аспектах робить його добре придатним для покращення зображень із шумами, артефактами та іншими недоліками.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ МОДЕЛІ

3.1 Вибір інструментів програмної реалізації

Вибір правильних інструментів реалізації програмного забезпечення має вирішальне значення для успіху будь-якого проекту. Ці інструменти слугують фундаментом, на якому розробники створюють додатки та системи. Правильно підібрані інструменти спрощують процеси розробки, роблячи їх більш ефективними. Це призводить до скорочення циклів розробки та швидшої доставки продуктів кінцевим користувачам. Також потрібно враховувати й те шо, інструменти повинні бути сумісними з вимогами проекту та один з одним, щоб забезпечити безперебійну інтеграцію та роботу.

Для програмної реалізації поставленої задачі було обрано середовище програмування Visual Studio Code (VS Code). VS Code є легким порівняно з іншими IDE, що робить його швидким і чуйним навіть на машинах з низьким рівнем продуктивності. Широкий ринок розширень дозволяє розробникам налаштовувати своє середовище за допомогою інструментів і функцій, пристосованих до їхніх конкретних потреб. VS Code без проблем працює на Windows, macOS та Linux, забезпечуючи однаковий досвід розробки на різних операційних системах. Зручний інтерфейс у поєднанні з такими функціями, як IntelliSense для завершення коду та підтримки налагодження, підвищує продуктивність та покращує досвід розробників [22].

Поєднання мов програмування Python, JavaScript (JS), CSS та HTML у фреймворку Flask є чудовим вибором для реалізації простої, але ефективної веб-сторінки для використання нейронної мережі ERSGAN з наступних причин:

– Python відомий своєю простотою і читабельністю, що робить його ідеальним вибором для бекенд-розробки. Його великі бібліотеки та фреймворки, такі як Flask, спрощують завдання веб-розробки;

– JS необхідний для написання сценаріїв на стороні клієнта, що дозволяє створювати динамічні та інтерактивні елементи веб-сторінок. Такі бібліотеки, як TensorFlow.js, можна використовувати для роботи нейронних мереж на стороні клієнта;

– CSS і HTML: CSS надає можливості для стилізації, в той час як HTML структурує веб-контент. Обидві мови є фундаментальними для створення візуально привабливих і добре структурованих веб-сторінок;

– Flask це легкий і гнучкий фреймворк для створення веб-додатків на Python. Його простота і зручність у використанні роблять його чудовим вибором для малих і середніх проектів, таких як реалізація веб-інтерфейсу для ERSGAN [23].

Використовуючи ці мови та фреймворки в поєднанні з VS Code як середовищем розробки, можливо ефективно створити зручну веб-сторінку для використання нейронної мережі ERSGAN, надаючи користувачам інтуїтивно зрозумілий та зручний інтерфейс та користувацький досвід.

Окрім основних мов та середовища розробки, реалізація веб-сторінки для покращення зображень за допомогою нейронної мережі ERSGAN передбачає інтеграцію спеціальних бібліотек. Коротко розглянемо кожну з них та їхні функції на сторінці:

– у реалізації сторінки Flask слугує основою веб-сервера, обробляючи HTTP-запити та відповіді, маршрутизацію та рендеринг HTML-шаблонів;

– OpenCV (Open Source Computer Vision Library), бібліотека програмних функцій, головним чином спрямованих обробку зображень у реальному часі. OpenCV використовується для задач обробки зображень, таких як читання, декодування та запис зображень;

– NumPy потужна бібліотека для чисельних обчислень на мові Python. Вона забезпечує підтримку великих багатовимірних масивів і матриць, а

також набір математичних функцій для ефективної роботи з цими масивами. NumPy використовується для обробки даних зображень у форматі масиву, що полегшує різні операції з обробки зображень;

– Realesrgan_ncnn_py: ця бібліотека реалізує алгоритм ESRGAN) з використанням фреймворку NCNN (NVIDIA CUDA Neural Network). Ця бібліотека використовується для виконання процесу покращення зображень за допомогою нейронної мережі ERSGAN;

– модуль os у мові Python надає можливість використовувати функціональність, залежну від операційної системи, він використовується для обробки файлових операцій, таких як створення каталогів та керування шляхами до файлів.

Використовуючи ці спеціальні бібліотеки у веб-фреймворку Flask, розробники можуть легко інтегрувати можливості обробки зображень на основі алгоритму ESRGAN у веб-сторінку, дозволяючи користувачам покращувати свої зображення легко та ефективно.

3.2 Проектування та реалізація користувацького інтерфейсу

Для подальшого проектування та програмної реалізації, спочатку необхідно зрозуміти яким саме чином буде реалізована дана сторінка. Щоб зменшити кількість навантаження та зробити все максимально просто та ефективно запропонована структура зображена на рисунку 3.1.

В даній структурі Flask є головним компонентом, який управляє веб-запитами та відповідями. Технології фронтенду (HTML/CSS/JavaScript), які використовуються для створення користувацького інтерфейсу веб-додатку. Вони відповідають за відображення веб-сторінки в браузері користувача, стилізацію контенту (CSS) та інтерактивність (JavaScript). Realesrgan_ncnn_py та OpenCV/NumPy: OpenCV та NumPy є популярними бібліотеками для обробки зображень і використовуються для підготовки зображень до та після покращення ESRGAN.

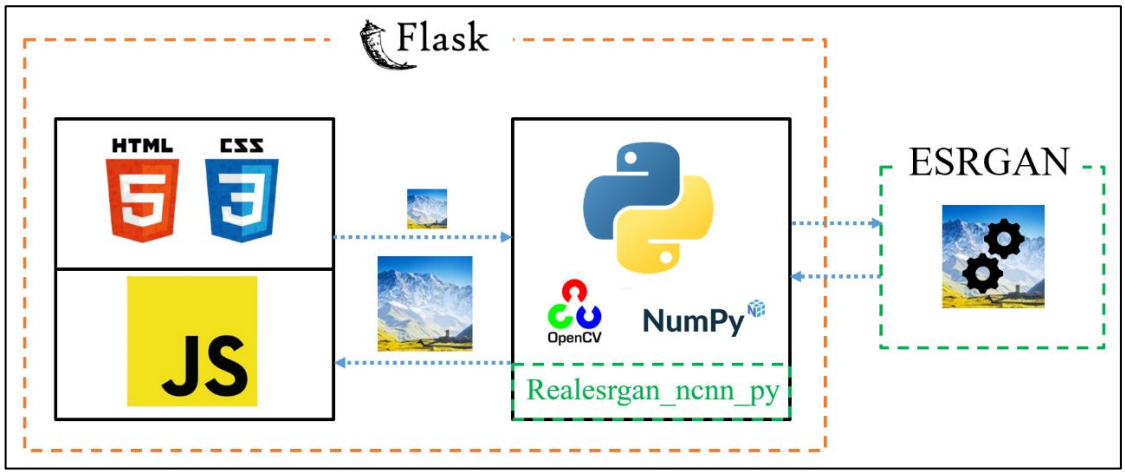

Рисунок 3.1 – Структура веб-сторінки

Realesrgan_ncnn_py бібліотека яка дозволяє легко та зручно використовувати модель ESRGAN. Користувацький інтерфейс передає зображення на сервер де це зображення за допомогою ESRGAN обробляється та передається назад до користувача.

Щодо програмної реалізації, розглянемо детально лістинг серверної частини веб-сторінки (лістинг 3.1).

Лістинг 3.1 – Серверна частина сторінки

```
from flask import Flask, request, send_file, render_template
import cv2
import numpy as np
from realesrgan_ncnn_py import Realesrgan
import os
app = Flask(__name__, static_folder='static', template_folder='templates')
realesrgan = Realesrgan(gpuid=0, tta_mode = False, tilesize= 0, model=4)
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return 'No file part', 400
    file = request.files['file']
    if file.filename == '':
        return 'No selected file', 400
```

Продовження лістингу 3.1

```
        if file:
            img    =    cv2.imdecode(np.frombuffer(file.read(),
np.uint8), cv2.IMREAD_COLOR)
            output_image = realesrgan.process_cv2(img)
            output_path = 'static/output.jpg'
            cv2.imwrite(output_path, output_image)
            return                        send_file(output_path,
mimetype='image/jpeg',                       as_attachment=True,
download_name='enhanced.jpg')
    if __name__ == '__main__':
        app.run(debug=True)
```

Спочатку відбувається імпорт необхідних модулів,з бібліотеки flask використовується для створення веб-додатку. request і send_file також імпортуються з flask і використовуються для обробки HTTP-запитів та відправлення файлів у відповіді. render_template дозволяє генерувати HTML-сторінки з шаблонів. cv2 з бібліотеки opencv-python використовується для обробки зображень. numpy бібліотека для обробки масивів. Realesrgan з пакета realesrgan_ncnn_py клас, що реалізує покращення зображень.

Далі йде створення екземпляра додатку Flask, об’єкт app створюється з використанням Flask, вказуючи папки для статичних файлів (static_folder='static') та шаблонів (template_folder='templates'). Після чого ініціалізація моделі ESRGAN, Об’єкт realesrgan створюється з параметрами для покращення зображень, де gpuid=0 вказує на використання першої GPU, а model=4 вибирає певну вже навчену модель ESRGAN.

Маршрутизація веб-додатку, @app.route('/') вказує на головну сторінку веб-додатку, яка відображатиме HTML-шаблон index.html у відповідь на запит GET. @app.route('/upload', methods=['POST']) визначає маршрут для завантаження файлів. Коли веб-форма відправляє дані методом POST на цей маршрут, скрипт обробляє вхідний файл.

Основна функція тут upload_file, вона виконує такий алгоритм дій:

– перевіряє, чи файл було завантажено з запитом;

– декодує зображення з вхідних даних файлу;

– використовує realesrgan для покращення зображення;

– зберігає оброблене зображення у папці static під іменем output.jpg;

– відсилає оброблене зображення як файл у відповіді HTTP.

Після чого відбувається запуск веб-сторінки, якщо цей файл є головним скриптом, веб-додаток запускається з активованим режимом налагодження (debug=True).

Наступним розглянемо лістинг сторони користувача (лістинг 3.2).

Лістинг 3.2 – Клієнтська частина сторінки

```
    document.getElementById('uploadButton').addEventListener('
click', function() {
        document.getElementById('imageInput').click();
    });

    document.getElementById('processButton').addEventListener(
'click', function() {
        var                    imageInput                    =
document.getElementById('imageInput');
        var                  originalImage                   =
document.getElementById('originalImage');
        var                   enhancedImage                  =
document.getElementById('enhancedImage');
        var                 enhancedLoading                  =
document.getElementById('enhancedLoading');
        var               enhancedLoadingText                =
document.getElementById('enhancedLoadingText');
        var loadingMessages = [
            "Покращуємо зображення...",
            "Працюємо над кожним пікселем...",
            "ERSGAN аналізує структуру зображення...",
            "Створюємо деталізовану карту текстур...",
            "Виправляємо артефакти та шуми...",
            "Використовуємо глибинне навчання для досягнення
найкращого результату...",
            "Кожен елемент зображення опрацьовується окремо для
максимальної точності...",
            "ERSGAN уточнює контраст та колір для кращої якості
зображення...",
            "Цей процес може тривати до декількох хвилин...",
            "Майже готово..."
        ];
        var messageIndex = 0;
        function updateLoadingMessage() {
            enhancedLoadingText.innerText                    =
loadingMessages[messageIndex++ % loadingMessages.length];
        }
```

Продовження лістингу 3.2

```
        if (imageInput.files && imageInput.files[0]) {
            var formData = new FormData();
            formData.append('file', imageInput.files[0]);

            var reader = new FileReader();
            reader.onload = function(e) {
                originalImage.src = e.target.result;
                originalImage.classList.remove('hidden');
            };
            reader.readAsDataURL(imageInput.files[0]);
            enhancedLoading.style.display = 'block';
            enhancedLoadingText.style.display = 'block';
            enhancedImage.classList.add('hidden');
            var                messageInterval                =
setInterval(updateLoadingMessage, 5000);
            fetch('/upload', {
                method: 'POST',
                body: formData
            })
            .then(response => {
                if (!response.ok) {
                    throw new Error('Network response was not
ok ' + response.statusText);
                }
                return response.blob();
            })
            .then(blob => {
                var url = window.URL.createObjectURL(blob);
                enhancedImage.src = url;
                enhancedImage.classList.remove('hidden');
                enhancedImage.title    =    "Натисніть,    щоб
відкрити";

                enhancedImage.onclick = function() {
                    window.open(url, '_blank');
                };
                enhancedLoading.style.display = 'none';
                enhancedLoadingText.style.display = 'none';
                clearInterval(messageInterval);
            })
            .catch(error => {
                console.error('There has been a problem with
your fetch operation:', error);
                enhancedLoading.style.display = 'none';
                enhancedLoadingText.style.display = 'none';
                clearInterval(messageInterval);
            });
        } else {
            alert('Please select an image.');
        }
    });
```

В цьому коді спрацьовування кнопки «Обрати зображення», при натисканні на кнопку з id='uploadButton', автоматично активується клік на прихований елемент вводу файлу (<input type="file">), що дозволяє користувачу вибрати зображення зі свого пристрою. Далі йде обробка події «Почати обробку», коли користувач натискає кнопку з id='processButton', виконується функція, що ініціює процес обробки зображення. Створюються змінні для керування елементами на сторінці: вхідним полем для файлу, місцем для оригінального зображення, місцем для обробленого зображення, індикатором завантаження та текстом повідомлень процесу обробки. Визначається масив loadingMessages з текстами повідомлень, які будуть почергово відображатися користувачу під час завантаження та обробки зображення. Це відбувається завдяки функції updateLoadingMessage, вона оновлює текст повідомлення процесу обробки згідно з масивом повідомлень. Далі перевіряється, чи користувач вибрав файл. Якщо так, то виконується низка дій, створення FormData, обраний файл додається до об’єкту FormData, який пізніше буде відправлений на сервер.

Читання та відображення оригінального зображення, використовуючи FileReader, оригінальне зображення відображається на сторінці безпосередньо після вибору. Анімація завантаження та текстові повідомлення стають видимими, а місце для обробленого зображення приховується. Інтервалом налаштовується оновлення тексту повідомлення кожні 5 секунд.

Після чого йде запит fetch до сервера, відправляється POST-запит на сервер з об’єктом FormData, що містить вибране зображення. Отримавши відповідь, відбувається перевірка на успішність відповіді та створюється URL для обробленого зображення. Потім оброблене зображення показується на сторінці. До обробленого зображення додається title з текстом підказки і обробник подій onclick, який відкриває зображення в новому вікні браузера.

Після завантаження зображення, анімація завантаження та текстові повідомлення приховуються, а інтервал оновлення повідомлень очищується. У разі помилок під час fetch-запиту, індикатори завантаження приховуються, і виводиться повідомлення про помилку в консоль. Якщо файл не було вибрано, користувачеві показується відповідне попередження.

3.3 Тестування, налагодження та аналіз результатів роботи інтерфейсу

Щоб реалізувати поставлені задачі було розроблено користувацький інтерфейс для роботи з веб-сторінкою (рисунок 3.2).

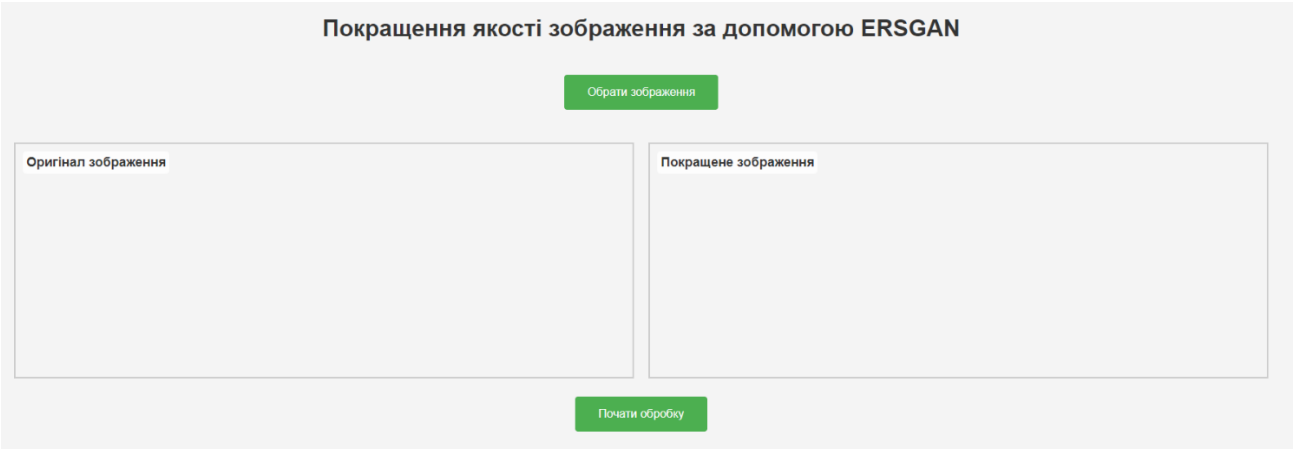

Рисунок 3.2 – Дизайн сторінки

На сторінці зображені дві чітко розділені секції для «Оригінального зображення» та «Покращеного зображення», кожна має свою рамку, яка допомагає візуально відділити зображення одне від одного. Кнопки «Обрати зображення» та «Почати обробку» виділяються своїм зеленим кольором, що робить їх легко помітними та доступними для користувача. Така простота, без відволікаючих елементів дозволяє користувачам легко розуміти, що від них очікується. Кнопки для вибору та обробки зображення є зрозумілими та очевидними у використанні, не вимагаючи додаткових інструкцій. Після проведення тестування даних елементів виявлено, що

кнопки «Обрати зображення» та «Почати обробку» функціонують правильно, реагуючи на клік користувача та виконуючи відповідні дії. Також було перевірено, що секції «Оригінального зображення» та «Покращеного зображення» відокремлені чіткою рамкою, що сприяє візуальному розмежуванню між ними. Користувачам було легко розуміти, як користуватися цими елементами без необхідності додаткових інструкцій, що підтверджує успішне пройдене тестування та відповідність елементів очікуваним результатам.

На сторінці також реалізована функція завантаження процесу обробки (рисунок 3.3).

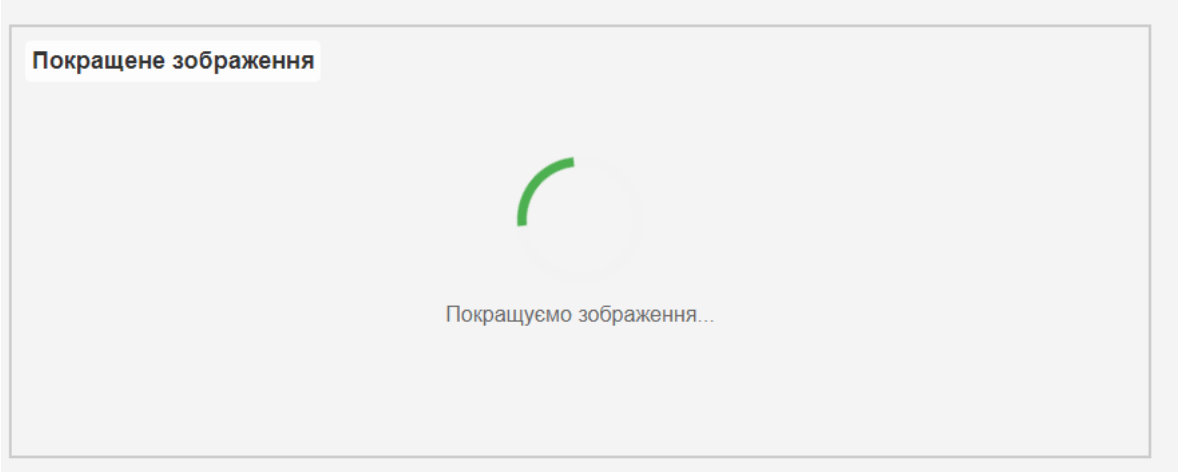

Рисунок 3.3 – Демонстрація завантаження процесу обробки

Тут відображаються текстові підказки користувачеві про процес обробки його зображення. Завантаження та текст до них показуються тільки тоді коли це потрібно, коли користувач обирає фото та натискає на кнопку початку процесу обробки. Тому цей функціонал працює так як і задумано.

Після того як оброблене зображення з’явиться та при умові що користувач наведе курсор на нього появиться підказка «Натисніть щоб відкрити» (рисунок 3.4).

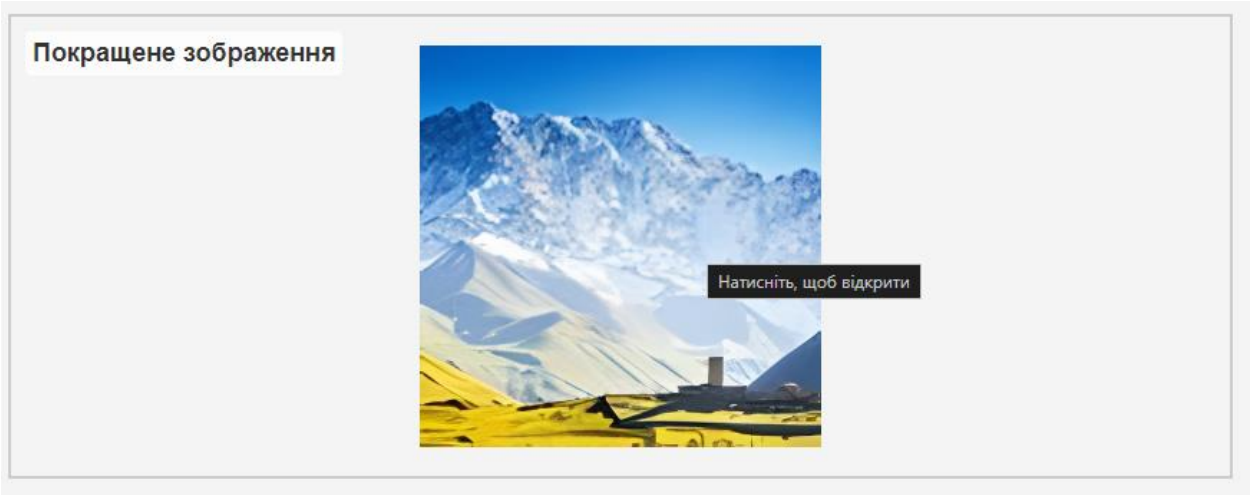

Рисунок 3.4 – Демонстрація підказки

Коли користувач натисне на оброблене зображення воно відкриється у новому максимальному розширенні у новій вкладці браузера. Таким чином користувач зможе детальніше переглянути оброблене зображення та при необхідності завантажити його на свій пристрій. Проведене тестування показало що даний функціонал також працює належним чином, підказка хоч і з маленькою затримкою але з’являється, під час натискання на область обробленого фото воно відкривається в новому вікні.

Якщо ж користувач натисне на кнопку «Почати обробку» але не вибере зображення, він отримає повідомлення про необхідність зробити це (рисунок 3.5).

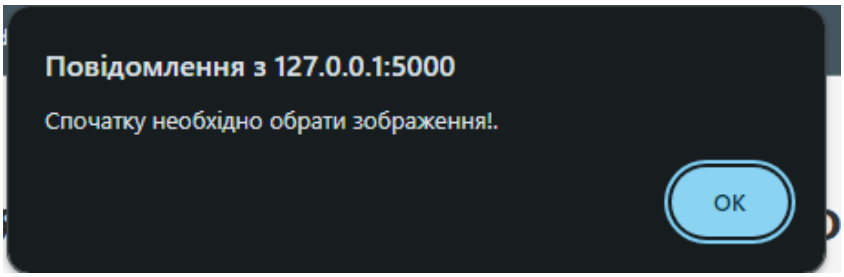

Рисунок 3.5 – Повідомлення про необхідність вибору зображення

Тестування показало що повідомлення про помилку з’являється тільки тоді коли користувач виконує певні умови, тобто не обирає зображення. Тестування цього аспекту веб-сторінки також пройдено успішно. Враховуючи те що всі очікуванні функції працюють як належне, зображення покращуються, всі необхідні кнопки та повідомлення працюють так як цього і очікується, тестування можна вважати пройденим.

При оцінці результатів покращення зображень за допомогою моделі ERSGAN важливо усвідомлювати, що цей процес базується на доповненні візуальної інформації на основі патернів, засвоєних моделлю під час навчання. Такий підхід може призвести до того, що результати не завжди відповідають очікуванням користувачів, особливо у випадках, коли зображення мають складні структури або специфічні особливості, не враховані в навчальному наборі даних. Крім того, варто враховувати, що модель ERSGAN навчена на певному обсязі різноманітних зображень, але вона все одно відображає певні стилі та особливості, характерні для цього набору даних. Це може вплинути на те, як саме модель реагує на конкретне зображення та його покращення. Такий фактор може відтворити непередбачувані результати, які можуть бути відчутними для користувача.

Отже, при оцінці результатів покращення зображень за допомогою моделі ERSGAN важливо мати на увазі не лише технічні аспекти алгоритму, але й його обмеження та особливості, що можуть вплинути на кінцевий результат. У висновку можна зазначити, що незважаючи на вищезгадані обмеження, модель ERSGAN ефективно виконує своє завдання та дійсно покращує якість зображень, збільшуючи їх фактичну роздільну здатність. Це є вагомим та значним результатом, особливо у контексті розробки та вдосконалення методів обробки зображень.

ВИСНОВКИ

У ході виконання даної роботи було проведено дослідження ефективності моделей підвищення якості зображень, зібрано та проаналізовано дані про їх продуктивність, включаючи якість покращення зображень та обчислювальну ефективність. За допомогою стандартизованих наборів даних та метрик продуктивності були оцінені моделі, а результати порівняні та проаналізовані з метою виділення сильних та слабких сторін кожної з них. Далі були розроблені критерії оцінювання ефективності кожної моделі на основі якості зображення, швидкості обробки та використання ресурсів, що дозволило об’єктивно оцінити їх продуктивність. Аналіз принципу роботи найбільш ефективної моделі допоміг виокремити ключові особливості, що призводять до найкращої ефективності.

Надалі був розроблений користувацький інтерфейс у вигляді веб-сторінки, який забезпечує зручність взаємодії з обраною моделлю, включаючи функції введення/виведення зображень та візуалізацію результатів. Через систематичне тестування та налагодження було підтверджено функціональність та надійність розробленого продукту, виявлені та вирішені всі виявлені проблеми та помилки.

В результаті отримано глибокий і детальний порівняльний аналіз моделей підвищення якості зображень, розроблені критерії їх оцінки та користувацький інтерфейс для взаємодії з обраною моделлю.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Штучний інтелект для покращення якості фото. *Evergreens*. 2020. URL: https://evergreens.com.ua/ua/articles/image-enhancement-solutions.html (дата звернення: 05.05.2024).

2. Deep Degradation Prior for Low-quality Image Classification. *University of Science and Technology of China*. 2023. URL: https://openaccess.thecvf.com/content_CVPR_2020/papers/Wang_Deep_Degradation_Prior_for_Low-Quality_Image_Classification_CVPR_2020_paper.pdf (дата звернення: 05.05.2024).

3. Peak Signal-to-Noise Ratio as an Image Quality Metric. *NI*. 2023. URL: https://www.ni.com/en/shop/data-acquisition-and-control/add-ons-for-data-acquisition-and-control/what-is-vision-development-module/peak-signal-to-noise-ratio-as-an-image-quality-metric.html (дата звернення: 05.05.2024).

4. All about Structural Similarity Index (SSIM): Theory + Code in PyTorch. *Medium*. 2020. URL: https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e (дата звернення: 05.05.2024).

5. Штучний інтелект для покращення якості фото. *Evergreens*. 2020. URL: https://evergreens.com.ua/ua/articles/image-enhancement-solutions.html (дата звернення: 05.05.2024).

6. SRGAN: Super Resolution Generative Adversarial Networks. *PapersPace*. 2020. URL: https://blog.paperspace.com/super-resolution-generative-adversarial-networks/ (дата звернення: 05.05.2024).

7. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks. *Esrgan*. 2022. URL: https://esrgan.readthedocs.io/en/latest/ (дата звернення: 05.05.2024).

8. Deep Image Prior. *Openaccess*. URL: https://openaccess.thecvf.com/content_cvpr_2018/Ulyanov_Deep_Image_Prior_2018_paper.pdf (дата звернення: 05.05.2024).

9. VDSR in Tensorflow. *Modelzoo*. 2020. URL: https://modelzoo.co/model/vdsr (дата звернення: 05.05.2024).

10. A Gentle Introduction to Generative Adversarial Networks (GANs). *MachineLearningMastery*. 2019. URL: https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/ (дата звернення: 05.05.2024).

11. Synthetic Data Generation Using CTGAN. *Medium*. 2022. URL: https://medium.com/@rahul.anb24/synthetic-data-generation-using-ctgan-ecb7b1f4687b (дата звернення: 05.05.2024).

12. Felix P. G. Synthetic Data Generation Using CTGAN. *MATEC Web of Conferences*. 2022. URL: https://medium.com/@rahul.anb24/synthetic-data-generation-using-ctgan-ecb7b1f4687b (дата звернення: 05.05.2024).

13. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *Computer Vision and Pattern Recognition*. 2015. URL: https://arxiv.org/abs/1502.01852v1 (дата звернення: 05.05.2024).

14. ESRGAN. *Github*. 2018. URL: https://github.com/xinntao/ESRGAN (дата звернення: 05.05.2024).

15. ESRGAN: Enhanced Super-Resolution Generative Adversarial Networks / Wang X. et al. *ECCV 2018*. 2022. URL: https://arxiv.org/abs/1809.00219 (дата звернення: 05.05.2024).

16. Deep Residual Dense Network for Single Image Super-Resolution. 2021. URL: https://www.mdpi.com/2079-9292/10/5/555 (дата звернення: 05.05.2024).

17. VDSR-PyTorch. *Github*. 2019. URL: https://github.com/Lornatang/VDSR-PyTorch (дата звернення: 05.05.2024).

18. Image Super-Resolution Using Very Deep Convolutional Network / Kim J. et al. Department of ECE, ASRI, Seoul National University. 2016. URL: https://openaccess.thecvf.com/content_cvpr_2016/papers/Kim_Accurate_Image_Super-Resolution_CVPR_2016_paper.pdf (дата звернення: 05.05.2024).

19. Review: VDSR (Super Resolution). *TowardsDataScience*. 2018. URL: https://towardsdatascience.com/review-vdsr-super-resolution-f8050d49362f (дата звернення: 05.05.2024).

20. Ulyanov D. deep-image-prior. 2018. URL: https://github.com/DmitryUlyanov/deep-image-prior (дата звернення: 05.05.2024).

21. Restoration using Total Variation Regularized Deep Image Prior / Liu J. et al. Department of Electrical and Systems Engineering, Washington University in St. Louis, MO 63130, USA. 2020. URL: https://www.semanticscholar.org/reader/750fce296d4337aa69cf6c15911032ab1afde297 (дата звернення: 05.05.2024).

22. VisualStudio. *Microsoft*. 2024. URL: https://code.visualstudio.com/ (дата звернення: 05.05.2024).

23. Welcome to Flask's documentation. *FLASK*. 2010. URL: https://flask.palletsprojects.com/en/3.0.x/ (дата звернення: 05.05.2024).

24. OpenCV. 2024. URL: https://opencv.org/ (дата звернення: 05.05.2024).

25. NumPy. 2024. URL: https://numpy.org/ (дата звернення: 05.05.2024).

26. realsr-ncnn-vulkan-python 1.0.6. *PyPi*. 2022. URL: https://pypi.org/project/realsr-ncnn-vulkan-python/ (дата звернення: 05.05.2024).

ДОДАТОК А

Код програми

Основний лістинг веб-сторінки:

app.py:

```
from flask import Flask, request, send_file, render_template
import cv2
import numpy as np
from realesrgan_ncnn_py import Realesrgan
import os
app = Flask(__name__, static_folder='static',
template_folder='templates')
realesrgan = Realesrgan(gpuid=0, tta_mode = False, tilesize=
0, model=4)
@app.route('/')
def index():
    return render_template('index.html')
@app.route('/upload', methods=['POST'])
def upload_file():
    if 'file' not in request.files:
        return 'No file part', 400
    file = request.files['file']
    if file.filename == '':
        return 'No selected file', 400
    if file:
        img = cv2.imdecode(np.frombuffer(file.read(),
np.uint8), cv2.IMREAD_COLOR)
        output_image = realesrgan.process_cv2(img)
        output_path = 'static/output.jpg'
        cv2.imwrite(output_path, output_image)
        return send_file(output_path, mimetype='image/jpeg',
as_attachment=True, download_name='enhanced.jpg')
if __name__ == '__main__':
```

```
    app.run(debug=True)
```

index.html:

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <title>ERSGAN</title>

    <link rel="stylesheet" href="/static/style.css">

</head>

<body>

    <div class="container">

        <h1>Покращення якості зображення за допомогою
ERSGAN</h1>

        <div class="file-upload-wrapper">

            <button type="button" id="uploadButton"
class="upload-button">Обрати зображення</button>

            <input type="file" id="imageInput"
accept="image/*" class="hidden">

        </div>

        <div id="loading" class="hidden"></div>

        <div class="image-display">

            <div class="image-frame">

                <h2>Оригінал зображення</h2>

                <img id="originalImage" src="#" alt="Original
Image" class="hidden">

            </div>

            <div class="image-frame">

                <h2>Покращене зображення</h2>

                <div class="loading-container hidden">

                    <div id="enhancedLoading" class="loading-
spinner"></div>
```

```
                    <div id="enhancedLoadingText"
class="loading-text">Покращуємо зображення...</div>
                </div>
                <img id="enhancedImage" src="#" alt="Enhanced
Image" class="hidden">
            </div>
        </div>
        <button id="processButton">Почати обробку</button>
    </div>
    <script src="/static/script.js"></script>
</body>
</html>
```

style.css:

```
body {
    font-family: 'Arial', sans-serif;
    text-align: center;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
}
.container {
    max-width: 90%;
    margin: 20px auto;
    padding: 20px;
}
h1 {
    color: #333;
    margin-bottom: 20px;
}
button {
    background-color: #4CAF50;
    border: none;
    color: white;
```

```
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 25px 2px;
    cursor: pointer;
    border-radius: 4px;
}
button:hover{
    background-color: #367d38;
}
.upload-button {
    background-color: #4CAF50;
    border: none;
    color: white;
    padding: 15px 32px;
    text-align: center;
    text-decoration: none;
    display: inline-block;
    font-size: 16px;
    margin: 25px 2px;
    cursor: pointer;
    border-radius: 4px;
}
.upload-button:hover{
    background-color: #367d38;
}
input[type="file"] {
    margin: 20px 0;
}
.image-display {
```

```
    display: flex;
    justify-content: space-around;
    flex-wrap: wrap;
    gap: 20px;
    margin-top: 20px;
}
@media (min-width: 768px) {
    .image-display {
        flex-wrap: nowrap;
    }
}
.image-frame {
    border: 2px solid #cccccc;
    padding: 10px;
    flex: 1;
    min-height: 300px;
    display: flex;
    justify-content: center;
    align-items: center;
    position: relative;
}
h2 {
    position: absolute;
    top: 10px;
    left: 10px;
    margin: 0;
    padding: 5px;
    background: rgba(255,255,255,0.8);
    border-radius: 4px;
    font-size: 18px;
    color: #333;
}
```

```
img {
    max-width: 100%;
    max-height: 280px;
    vertical-align: middle;
}
@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}
.loading-spinner {
    border: 7px solid #f3f3f3;
    border-top: 7px solid #4CAF50;;
    border-radius: 50%;
    width: 80px;
    height: 80px;
    animation: spin 2s linear infinite;
    display: none;
}
.hidden {
    display: none;
}
.loading-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
}
.loading-text {
    margin-top: 15px;
    color: #666;
    display: none;
}
```

script.js:

```
document.getElementById('uploadButton').addEventListener('clic
k', function() {
    document.getElementById('imageInput').click();
});


document.getElementById('processButton').addEventListener('cli
ck', function() {
    var imageInput = document.getElementById('imageInput');
    var originalImage =
document.getElementById('originalImage');
    var enhancedImage =
document.getElementById('enhancedImage');
    var enhancedLoading =
document.getElementById('enhancedLoading');
    var enhancedLoadingText =
document.getElementById('enhancedLoadingText');
    var loadingMessages = [
        "Покращуємо зображення...",
        "Працюємо над кожним пікселем...",
        "ERSGAN аналізує структуру зображення...",
        "Створюємо деталізовану карту текстур...",
        "Виправляємо артефакти та шуми...",
        "Використовуємо глибинне навчання для досягнення
найкращого результату...",
        "Кожен елемент зображення опрацьовується окремо для
максимальної точності...",
        "ERSGAN уточнює контраст та колір для кращої якості
зображення...",
        "Цей процес може тривати до декількох хвилин...",
        "Майже готово..."
    ];
    var messageIndex = 0;
    function updateLoadingMessage() {
```