МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Харківський національний університет імені В.Н.Каразіна
Факультет математики і інформатики
Кафедра теоретичної та прикладної інформатики

Кваліфікаційна робота

бакалавр

на тему «Алгоритми відновлення спотворених зображень»

Виконав: студент 4 курсу, групи МФ-41
спеціальність 122 «Комп'ютерні науки»
освітньо-професійна програма
«Інформатика» Дубина А.М.

Керівник: доц. Доля П.Г.

Харків – 2024 року

ЗМІСТ

ВСТУП

Актуальність теми. Відновлення спотворених зображень є важливою проблемою в галузі комп'ютерної обробки зображень. Ця проблема має широкий спектр застосувань, від реставрації старих фотографій та покращення якості цифрових зображень до медичних зображень та супутникової фотозйомки. У даній дипломній роботі розглянуто різні алгоритми відновлення зображень, зокрема, сучасні методи, що базуються на глибоких нейронних мережах. Крім того, розроблено інтерфейс для найефективнішого алгоритму, що дозволяє користувачам легко застосовувати цей алгоритм до своїх зображень. Актуальність обраної теми обумовлена зростаючою потребою у високоякісному відновленні зображень у різних сферах життя, включаючи мистецтво, медицину, науки про землю та інші. Сучасні методи, що базуються на глибоких нейронних мережах, відкривають нові можливості для відновлення зображень, які були недоступні раніше. Аналіз вітчизняної та зарубіжної наукової літератури показує, що проблема відновлення спотворених зображень є актуальною та має велике значення для розвитку відповідної галузі науки.

Метою даної роботи є дослідження та порівняння різних алгоритмів відновлення спотворених зображень, а також розробка інтерфейсу для найефективнішого з них. Для досягнення цієї мети було поставлено наступні завдання:

1. Вивчити сучасні методи відновлення зображень на основі глибоких нейронних мереж;
2. Проаналізувати ефективність обраних методів на різних наборах зображень;
3. Розробити інтерфейс для найефективнішого алгоритму, що дозволяє зручно використовувати його на практиці;
4. Здійснити порівняльний аналіз результатів роботи різних алгоритмів;

У процесі дослідження використовувалися наступні методи:

- Аналіз літератури для вивчення існуючих підходів та алгоритмів відновлення зображень;
- Експериментальний метод для тестування ефективності різних алгоритмів на реальних даних;
- Порівняльний аналіз для оцінки результатів роботи різних алгоритмів та вибору найефективнішого;
- Розробка програмного забезпечення для створення інтерфейсу, що дозволяє користувачам застосовувати обраний алгоритм до своїх зображень.

Практичне значення результатів полягає у можливості використання розробленого інтерфейсу для відновлення зображень у різних сферах, включаючи реставрацію старих фотографій, покращення якості цифрових зображень, медичних зображень та супутникової фотозйомки. Розроблений інтерфейс може бути впроваджений у відповідних організаціях та використовуватися для практичних завдань відновлення зображень.

У першому розділі подано огляд існуючих методів відновлення зображень. У другому розділі проведено порівняльний аналіз ефективності обраних методів. У третьому розділі описано розробку інтерфейсу для найефективнішого алгоритму та його тестування.

РОЗДІЛ 1 АЛГОРИТМИ ТА МЕТОДИ ВІДНОВЛЕННЯ ЯКОСТІ ЗОБРАЖЕНЬ

1.1 Класифікація методів відновлення зображень

Відновлення зображень є важливим завданням у сфері обробки зображень, що має на меті покращення якості зображень, які зазнали певних спотворень. За довгу історію розвитку методів відновлення зображень було розроблено велику кількість різноманітних підходів, які можна класифікувати за різними ознаками. Перші методи відновлення зображень виникли на основі математичних і статистичних методів обробки сигналів. Вони включали прості фільтруючі методи, такі як середнє арифметичне, медіанне фільтрування та інтерполяція. Згодом, з розвитком комп’ютерних технологій, з'явилися більш складні алгоритми, що дозволяли враховувати різні аспекти спотворень і відновлювати зображення з високою точністю [1-3].

Методи відновлення зображень можна класифікувати на кілька основних категорій: традиційні методи, методи на основі регуляризації, методи на основі глибоких нейронних мереж та гібридні методи. Фільтраційні методи, ці методи включають різноманітні типи фільтрів, які використовуються для усунення шумів і зменшення спотворень у зображеннях. До них належать лінійні фільтри, такі як Гауссів фільтр, та нелінійні фільтри, як-от медіанний фільтр. Фільтраційні методи є ефективними для зменшення шумів, але можуть призводити до втрати деталей у зображенні. Інтерполяційні методи, інтерполяція використовується для відновлення втрачених або спотворених пікселів зображення. Ці методи застосовуються, зокрема, при масштабуванні зображень і відновленні втрачених даних. Регуляризація є підходом, що дозволяє вирішувати некоректно поставлені задачі, вводячи додаткову інформацію або припущення про шукане рішення. Регуляризаційні методи дозволяють відновлювати зображення з високою якістю [4].

Глибокі нейронні мережі (ГНМ). Останніми роками значного поширення набули методи відновлення зображень на основі ГНМ. Ці методи використовують архітектури нейронних мереж, такі як конволюційні нейронні мережі (CNN), для навчання на великих наборах даних зображень. Прикладами є методи, що базуються на автоенкодерах, GAN (Generative Adversarial Networks) та інші сучасні архітектури. ГНМ дозволяють відновлювати зображення з високою деталізацією та точністю [5].

Щодо гібридних методів то вони комбінують різні підходи для досягнення кращих результатів. Наприклад, можна поєднувати фільтраційні методи з методами на основі ГНМ для покращення відновлення зображень. Інший підхід полягає у використанні регуляризації разом з глибокими нейронними мережами для досягнення кращої стабільності та точності.

Загалом, методи відновлення зображень можна класифікувати за кількома основними ознаками:

- За типом використовуваних алгоритмів: фільтраційні, інтерполяційні, регуляризаційні, на основі ГНМ, гібридні;
- За метою відновлення: усунення шумів, відновлення втрачених даних, покращення роздільної здатності, реставрація старих зображень;
- За обсягом вхідних даних: методи, що використовують лише одне зображення, та методи, що використовують множину зображень або відео.

Класифікація методів відновлення зображень допомагає систематизувати існуючі підходи та вибрати найбільш відповідний метод для конкретного завдання.

Це дозволяє покращити якість відновлення та підвищити ефективність обробки зображень у різних застосуваннях. Завдяки класифікації можна визначити сильні та слабкі сторони кожного методу, що сприяє обґрунтованому вибору і точнішому налаштуванню алгоритмів.

1.2 Традиційні методи відновлення зображень

Традиційні методи відновлення зображень включають широкий спектр алгоритмів, які використовуються для покращення якості зображень, усунення шуму та відновлення втрачених деталей. Ці методи поділяються на кілька основних категорій:

1. Фільтраційні методи використовуються для зменшення шуму та згладжування зображень. Серед них виділяють [6]:

1.1 Лінійні фільтри:

- Гауссовий фільтр: використовує нормальний розподіл для згладжування зображення, зменшуючи високочастотний шум;
- Фільтр середнього: замінює кожен піксель середнім значенням сусідніх пікселів, що допомагає згладити зображення та зменшити шум.

1.2 Нелінійні фільтри:

- Медіанний фільтр: замінює кожен піксель медіанним значенням сусідніх пікселів, ефективно видаляючи імпульсний шум;
- Білінійний фільтр: враховує вагові коефіцієнти сусідніх пікселів для згладжування, зберігаючи при цьому більше деталей.

2. Інтерполяційні методи застосовуються для відновлення втрачених пікселів. Основні методи включають:

2.1 Метод найближчого сусіда: використовує значення найближчого пікселя для заповнення відсутніх даних, що є швидким, але може створювати блокові артефакти;

2.2 Білайнейна інтерполяція: враховує значення сусідніх пікселів для заповнення пропусків, забезпечуючи плавні переходи між пікселями;

2.3 Бікубічна інтерполяція: використовує значення найближчих 16 пікселів, забезпечуючи більш плавні і природні переходи порівняно з білайнейною інтерполяцією [7].

3. Регуляризаційні методи вводять додаткові припущення для вирішення задач відновлення [8]. Серед них:

3.1 Метод Тихонова: застосовує регуляризаційний член для стабілізації рішення, зменшуючи ефект шуму;

3.2 Метод мінімізації загальної варіації (TV): зменшує шум, зберігаючи при цьому контури і різкі зміни в зображенні, що є важливим для відновлення деталей.

4. Методи на основі інтегральних перетворень, використовують частотний аналіз для відновлення зображень. Основні методи включають:

4.1 Перетворення Фур'є: аналізує зображення в частотній області, дозволяючи видалити високочастотний шум;

4.2 Вейвлет-перетворення: розкладає зображення на різні компоненти, забезпечуючи можливість одночасного аналізу і згладжування на різних масштабах.

Ці традиційні методи відіграють важливу роль у покращенні якості зображень, забезпечуючи ефективні підходи до обробки та відновлення візуальної інформації. Вони слугують основою для більш складних сучасних технологій, які використовуються сьогодні.

1.3. Сучасні методи на основі глибоких нейронних мереж

Сучасні методи відновлення зображень на основі ГНМ значно покращили можливості обробки та відновлення візуальної інформації. Ці методи дозволяють ефективно усувати шуми, відновлювати втрачені деталі та покращувати загальну якість зображень. Розглянемо найпопулярніші алгоритми та методи у цій сфері.

SRCNN (Super-Resolution Convolutional Neural Network) є одним із перших успішних підходів для покращення роздільної здатності зображень за допомогою ГНМ. Він складається з трьох основних шарів(рис. 1.1)[9].

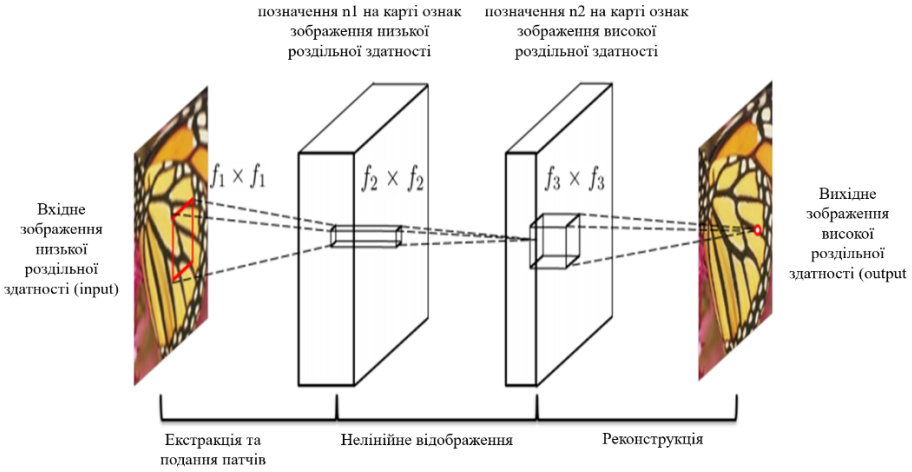

Рисунок 1.1 — Архітектура SRCNN [10]

1. Екстракція та подання патчів: виконує попередню обробку для виділення основних характеристик зображення;
2. Нелінійне відображення: створює нелінійні комбінації ознак для поліпшення якості зображення;
3. Реконструкція зображення: генерує зображення високої роздільної здатності на основі оброблених ознак.

Інший сучасний метод, DAE (Denoising Autoencoders) використовуються для усунення шуму з зображень. Вони навчаються реконструювати чисті зображення з зашумлених даних [11]. Архітектура складається з двох основних частин (рис. 1.2):

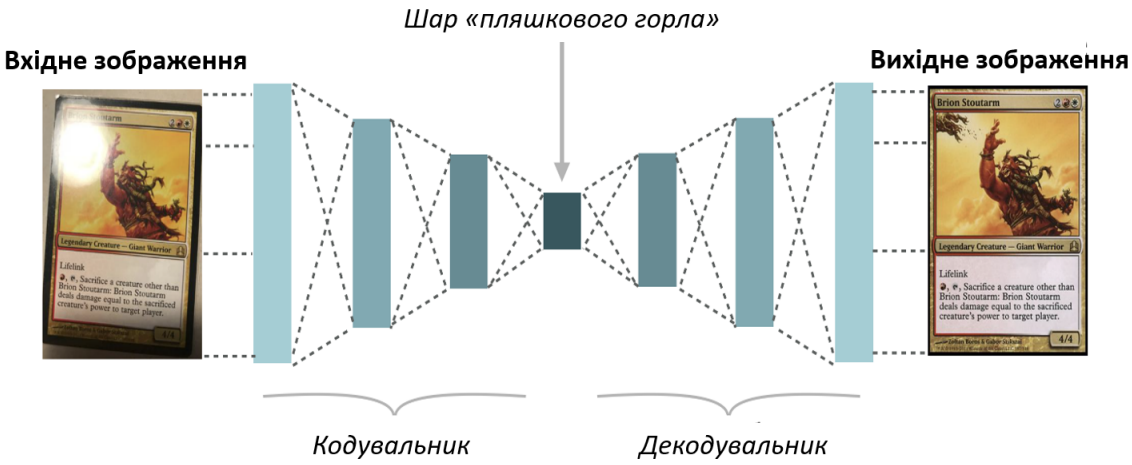

Рисунок 1.2 — Архітектура DAE [12]

1. Кодувальник: стискає вхідне зображення до меншого простору ознак;
2. Декодувальник: відновлює зображення з простору ознак, усуваючи шуми.

Метод Old Photo Restoration via Deep Latent Space Translation (OPRvDlST) представляє сучасний підхід до відновлення старих фотографій. Він використовує складну архітектуру глибоких нейронних мереж (ГНМ) для відновлення пошкоджених зображень, враховуючи різні аспекти, такі як тріщини, подряпини, знебарвлення тощо. Основні кроки цього методу включають:

- Передобробка зображень: Виділення регіонів, що потребують відновлення. Це включає ідентифікацію пошкоджених ділянок зображення, таких як тріщини, подряпини та знебарвлені області, які потребують подальшої обробки;
- Трансляція у прихованому просторі: Використання латентного простору для моделювання та відновлення деталей, що були втрачені або пошкоджені. Латентний простір дозволяє генерувати нові деталі, які гармонійно поєднуються з оригінальними частинами зображення;
- Післяобробка: Об’єднання відновлених регіонів з оригінальним зображенням для досягнення цілісності та природності результату. Це включає інтеграцію новостворених деталей з наявними частинами зображення, забезпечуючи узгодженість і реалістичність кінцевого результату.

Метод OPRvDlST застосовує генеративні мережі для створення реалістичних деталей, що відповідають оригінальним зображенням. Архітектура включає декілька основних компонентів [13]:

- Генеративні змагальні мережі (GANs): Основна частина архітектури, що складається з двох мереж – генератора та дискримінатора. Генератор створює відновлені деталі, а дискримінатор

оцінює їхню реалістичність, допомагаючи генератору покращувати якість;

- Автоенкодери: Використовуються для зменшення розмірності вхідних даних і витягнення важливих ознак. Автоенкодери навчаються відновлювати оригінальні зображення з пошкоджених вхідних даних;
- Латентний простір: Латентний простір забезпечує перетворення вхідних даних у зменшену репрезентацію, де втрачені деталі можуть бути відновлені. Використання латентного простору дозволяє моделювати складні залежності між різними частинами зображення;
- Спеціалізовані втрати: Використання спеціалізованих функцій втрат для покращення якості та узгодженості відновлених зображень. Ці функції включають комбіновані втрати для оцінки якості зображення та стилю, що дозволяє досягти більш природного вигляду відновлених областей.

Метод OPRvDlST був успішно реалізований та продемонстрований у проекті «Bringing Old Photos Back to Life» від Microsoft [13]. Відповідно до документації проекту на GitHub, цей метод застосовується для відновлення старих, пошкоджених фотографій, включаючи тріщини, подряпини та інші дефекти. Додаткова інформація про метод та результати його застосування доступна у науковій статті на arXiv [14]. У статті описано деталі архітектури, процес навчання та експериментальні результати, що демонструють ефективність методу.

Метод Deep Image Prior (DIP) використовує властивості нейронних мереж для відновлення зображень без попереднього навчання на великих наборах даних. Ідея полягає в тому, що архітектура глибокої нейронної мережі сама по собі здатна навчатися відновлювати зображення з високою якістю. DIP включає наступні основні компоненти[15]:

1. Навчання зображення на самому собі: Мережа налаштовується на відновлення вхідного зображення без використання зовнішніх даних. Це

дозволяє моделювати структуру та деталі зображення, використовуючи лише його вміст;

2. Регуляризація: Забезпечує згладжування та усунення шумів під час відновлення. Регуляризація допомагає уникнути перенавчання та забезпечує стабільність відновлених зображень;

На рисунку нижче (рис. 1.3) показано процес оновлення та навчання мережі DIP для відновлення зображення.

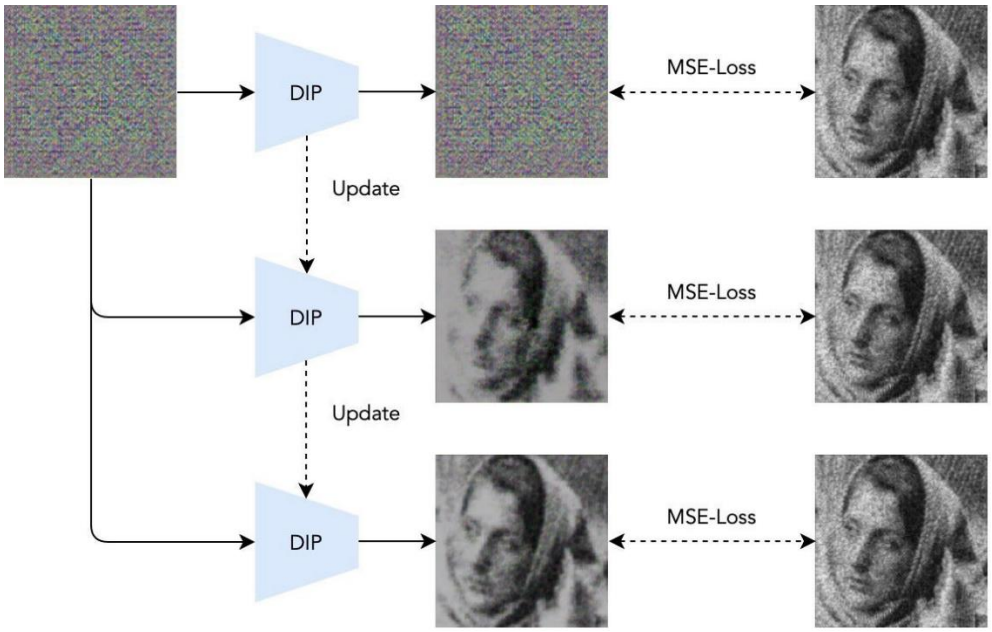

Рисунок 1.3 — Процес оновлення та навчання мережі DIP[16]

На цьому рисунку показано, як вхідний шумовий сигнал проходить через мережу DIP, яка оновлюється на основі мінімізації середньоквадратичної похибки (MSE-Loss) між вихідним сигналом мережі та цільовим зображенням. Процес повторюється кілька разів, поки мережа не відновить зображення до прийнятної якості.

Ці сучасні методи на основі глибоких нейронних мереж значно підвищили можливості обробки зображень, забезпечуючи високу точність та ефективність у відновленні візуальних даних.

РОЗДІЛ 2 ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ SRCNN, DAE, OPRvDlST та DIP

2.1. Критерії оцінки якості відновлення зображень

Для порівняння алгоритмів відновлення зображень, використовуються різноманітні критерії, які дозволяють оцінити їх ефективність та якість відновлених зображень. Основні критерії оцінки наведено нижче.

Піксельний сигнал-шум (PSNR) є основним критерієм для вимірювання якості відновлених зображень [17]. Він розраховується як логарифмічне відношення максимальної можливої потужності сигналу до потужності шуму, який впливає на якість зображення:

$$PSNR = 10log_{10}(\frac{MAX^2}{MSE}) \quad (1)$$

де:

MAX — максимальне значення пікселя (наприклад, 255 для 8-бітних зображень);

MSE — середньоквадратична похибка між відновленим та оригінальним зображенням.

Вищі значення PSNR вказують на кращу якість відновленого зображення.

Структурна подібність (SSIM) є метрикою, яка оцінює зорове сприйняття подібності між відновленим та оригінальним зображенням [18]. Вона враховує яскравість, контрастність та структуру зображень:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)} \quad (2)$$

де:

μ_x і μ_y — середні значення інтенсивності пікселів для зображень *x* та *y* відповідно;

σ_x^2 і σ_y^2 — дисперсії;

σ_{xy} — відношення зміни між зображеннями x та y, показує як зміни інтенсивності пікселів в одному зображенні пов'язані зі змінами інтенсивності пікселів в іншому зображенні;

C_1 і C_2 — константи для стабілізації розрахунків.

Значення SSIM варіюється від -1 до 1, де 1 означає ідентичність. SSIM=1 означає, що зображення x і y є ідентичними. Це вказує на те, що середні значення, дисперсії та відношення інтенсивності пікселів обох зображень повністю співпадають. SSIM=0 вказує на відсутність подібності між зображеннями. SSIM<0 (рідко використовується) може вказувати на протилежні структурні зміни між зображеннями. SSIM оцінює подібність між двома зображеннями на основі яскравості, контрастності та структури, і максимальне значення 1 показує, що всі ці аспекти співпадають повністю, що означає ідентичність зображень.

Візуальне порівняння є суб’єктивним, але важливим критерієм для оцінки якості відновлених зображень. Експерти оцінюють зображення на основі таких аспектів, як чіткість, відсутність артефактів, збереження дрібних деталей та природність відновлених областей. Візуальне порівняння дозволяє врахувати особливості, які можуть не бути виявленими за допомогою автоматичних метрик. Наприклад, навіть якщо PSNR та SSIM показують високі значення, зображення може містити дрібні артефакти або виглядати неприродно для людського ока. Тому експертна оцінка є невід'ємною частиною повного аналізу якості відновлення.

Час обробки є важливим критерієм, особливо для практичних застосувань, де швидкість виконання алгоритму може бути критичною. Час обробки залежить від складності алгоритму, його оптимізації та апаратних

ресурсів. Коротший час обробки при збереженні високої якості відновлення є значною перевагою. Наприклад, алгоритми на основі ГНМ можуть вимагати значних обчислювальних ресурсів, що збільшує час обробки, тоді як більш прості методи можуть забезпечити швидше виконання, але з меншою якістю.

Ефективність усунення шуму оцінюється за здатністю алгоритму видаляти різні типи шуму (наприклад, гауссів шум, імпульсний шум) при збереженні важливих деталей зображення. Це особливо важливо для алгоритмів, призначених для роботи з сильно зашумленими зображеннями. Наприклад, алгоритми на основі фільтраційних методів можуть ефективно усувати шум, але водночас можуть згладжувати дрібні деталі. Сучасні методи, такі як ГНМ, здатні краще зберігати структуру та текстуру зображень під час видалення шуму.

Алгоритми відновлення зображень повинні бути стійкими до різних типів спотворень, таких як розмиття, компресійні артефакти та інші дефекти. Оцінка стійкості до спотворень допомагає визначити універсальність та надійність алгоритму в різних умовах. Наприклад, методи, що використовують регуляризацію, можуть бути більш стійкими до різних типів спотворень, оскільки вони включають додаткові припущення про структуру зображення.

Деякі сучасні методи, такі як DIP, здатні відновлювати зображення без попереднього навчання на великих наборах даних. Це важлива властивість для застосувань, де доступ до великих навчальних вибірок обмежений. Оцінка алгоритмів за цим критерієм дозволяє визначити їхню гнучкість та універсальність. Наприклад, алгоритми, які не потребують попереднього навчання, можуть бути використані в умовах, коли кожне зображення є унікальним і не може бути частиною великої тренувальної вибірки.

Ці критерії оцінки якості відновлення зображень дозволяють здійснити комплексний аналіз алгоритмів та вибрати найбільш ефективний підхід для конкретного завдання. Врахування всіх аспектів дозволяє забезпечити

високоякісне відновлення зображень та підвищити продуктивність систем обробки візуальної інформації.

2.2 Порівняння алгоритмів та аналіз отриманих результатів

На основі обраних критеріїв та алгоритмів була створена таблиця порівняння (див. табл. 2.1).

Таблиця 2.1

Порівняння алгоритмів відновлення фото

Критерії	**SRCNN**	**DAE**	**OPRvDlST**	**DIP**
Піксельний сигнал-шум (PSNR)	30	28	35	29
Структурна подібність (SSIM)	0.85	0.80	0.90	0.82
Візуальне порівняння	Добре	Задовільно	Відмінно	Добре
Час обробки	Середній	Середній	Високий	Середній
Ефективність усунення шуму	Висока	Висока	Дуже висока	Висока
Стійкість до різних типів спотворень	Середня	Середня	Висока	Середня
Можливість навчання на малих наборах даних	Низька	Низька	Середня	Висока

На основі наведених критеріїв оцінки якості відновлення зображень та порівняльного аналізу алгоритмів, можна зробити наступні висновки, OPRvDlST показує найвищі результати за більшістю критеріїв. Він демонструє найвищі значення PSNR і SSIM, що вказує на високу якість відновлених зображень. Крім того, цей метод отримав найкращі результати за візуальним порівнянням, що підкреслює його здатність створювати реалістичні та деталізовані зображення. SRCNN та DIP також показують хороші результати, особливо в аспектах ефективності усунення шуму та якості відновлення. DIP має перевагу в можливості навчання на малих наборах даних, що робить його більш універсальним у застосуванні. DAE відстає за більшістю критеріїв порівняно з іншими методами, проте все ще демонструє високу ефективність у видаленні шуму.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРФЕЙСУ

3.1. Вибір та обґрунтування інструментів розробки

Для розробки веб-застосунку, який дозволяє користувачам відновлювати старі фотографії, було обрано наступні технології: JavaScript, HTML та CSS для фронтенду, а також Python для бекенду. Кожна з цих технологій має свої переваги та обґрунтування для використання у цьому проєкті. HTML є основною мовою розмітки для створення веб-сторінок. Вона забезпечує структуру веб-документів, дозволяючи визначати різні елементи сторінки, такі як заголовки, абзаци, зображення та форми. HTML є основою будь-якого веб-застосунку, забезпечуючи семантичну структуру та взаємодію користувача з веб-інтерфейсом. CSS використовується для стилізації HTML-документів. Вона дозволяє налаштовувати зовнішній вигляд веб-сторінок, включаючи кольори, шрифти, макети та адаптивність до різних екранів. CSS забезпечує привабливий та зручний інтерфейс користувача, що є важливим для покращення взаємодії з застосунком. JavaScript є мовою програмування, яка дозволяє додавати інтерактивність до веб-сторінок. Вона використовується для динамічного оновлення контенту, обробки подій, валідації форм та багато іншого. У цьому проєкті JavaScript використовується для інтеграції з API, обробки зображень на стороні клієнта та забезпечення зручного користувацького досвіду.

Python є популярною мовою програмування, відомою своєю простотою та зручністю для розробників. Вона широко використовується у веб-розробці завдяки своїм потужним бібліотекам та фреймворкам, таким як Flask та Django. У цьому проєкті Python використовується для створення бекенду, який обробляє запити від фронтенду, взаємодіє з API для відновлення фотографій та забезпечує необхідну логіку застосунку. Для відновлення старих фотографій використовується API від бібліотеки Replicate. Це API надає доступ до реалізації методу OPRvDlST.

HTML та CSS забезпечують стандартизовану структуру та стилізацію веб-сторінок, що робить їх базовими технологіями для створення веб-інтерфейсів. JavaScript дозволяє створювати динамічний та інтерактивний користувацький досвід, що є критичним для взаємодії користувача з веб-застосунком. Python забезпечує зручність розробки та потужні можливості для обробки запитів та взаємодії з API, роблячи його ідеальним вибором для бекенду. API Replicate надає доступ до передової технології відновлення зображень, що дозволяє інтегрувати високоякісні алгоритми відновлення фотографій у веб-застосунок [19].

Використання цих технологій забезпечує створення ефективного, зручного та функціонального веб-застосунку для відновлення старих фотографій, що відповідає сучасним вимогам до якості та користувацького досвіду.

3.2 Опис лістингу інтерфейсу

На початку відбувається Імпорт необхідних бібліотек. Flask використовується для створення веб-застосунку, а secure_filename з werkzeug для безпечного збереження файлів. Функція pred_img імпортується з модуля rop:

```
from flask import Flask, flash, request, redirect, url_for, render_template
from werkzeug.utils import secure_filename
from rop import pred_img
```

Далі встановлюються параметри завантаження файлів: директорія для збереження завантажених зображень (UPLOAD_FOLDER) та допустимі розширення файлів (ALLOWED_EXTENSIONS):

```
UPLOAD_FOLDER = 'src/img'
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}
```

Створюється екземпляр Flask додатку. Конфігуруються параметри додатку, включаючи директорію для збереження файлів та максимальний розмір файлу (16 МБ):

```
app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['MAX_CONTENT_LENGTH'] = 16 * 1000 * 1000
```

Визначається маршрут для головної сторінки. При зверненні до кореневого маршруту ("/") рендериться HTML шаблон index.html:

```
@app.route("/")
def home():
    return render_template("index.html")
```

Функція allowed_file перевіряє, чи має файл допустиме розширення. Це допомагає обмежити типи файлів, які можуть бути завантажені:

```
def allowed_file(filename):
    return '.' in filename and \
           filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
```

Наступним визначається маршрут для обробки POST-запитів на завантаження файлів. Код перевіряє, чи є файл у запиті, чи має він допустиме розширення, і зберігає його у визначеній директорії. Після цього викликається функція pred_img для обробки зображення, а результати рендеряться на сторінці index.html:

```
@app.route('/', methods=['POST'])
def upload_file():
    if request.method == 'POST':
        if 'file' not in request.files:
            flash('No file part')
            return redirect(request.url)
        file = request.files['file']
        if file.filename == '':
            return redirect(request.url)
        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            full_filename = "." + url_for("static",
filename="img/"+filename)
            file.save(full_filename)
            pred_img_url = pred_img(full_filename)
            print(full_filename)
            return render_template("index.html", filename=filename,
rest_img_url=pred_img_url)
```

Потім починає виконуватись код з іншого файлу проєкту. Запуск Flask додатку у режимі налагодження:

```
if __name__ == "__main__":
    app.run(debug=True)
```

Імпортуються бібліотеки dotenv для завантаження змінних середовища та replicate для взаємодії з API для відновлення старих фотографій. load_dotenv() завантажує змінні середовища з файлу .env:

```
from dotenv import load_dotenv
load_dotenv()
import replicate
```

Функція pred_img викликає API Replicate для обробки зображення, відправляючи зображення як вхідні дані. Повертає URL відновленого зображення:

```
def pred_img(filename):
    output = replicate.run(
        "microsoft/bringing-old-photos-back-to-
life:c75db81db6cbd809d93cc3b7e7a088a351a3349c9fa02b6d393e35e0d51ba799",
        input={"image": open(filename, "rb")}
    )
    print(output)
    return output
```

На цьому і закінчується робота серверу. Наступним в роботу вступає клієнтська сторона. HTML-розмітка визначає основну структуру сторінки. У <head> розміщено стилі, які налаштовують зовнішній вигляд елементів сторінки, таких як текст, зображення та кнопки. div з класом container забезпечує центрування та обмеження ширини контенту. Заголовок (<h2>) описує мету сторінки:

```
<div class="container">
    <h2>Відновлення Спотворених фотографій</h2>
```

div з класом image-container містить блоки для відображення оригінального та відновленого зображень. Умовні блоки {% if filename %} та {% if rest_img_url %} використовуються для перевірки наявності файлів, які потрібно відобразити:

```
<div class="image-container">
    {% if filename %}
```

```
                    <div>
                        <p>Оригінал</p>
                        <img                          src="{{url_for('static',
filename='img/'+filename)}}" alt="Original Image">
                    </div>
                {% endif %}
                {% if rest_img_url %}
                    <div>
                        <p>Відновлення</p>
                        <img src="{{rest_img_url}}" alt="Restored Image">
                    </div>
                {% endif %}
            </div>
```

div з класом button-container містить форму завантаження файлу. Форма (<form>) використовує метод POST для відправки файлу на сервер. Поле введення файлу (<input type="file">) дозволяє користувачам вибирати файл, а кнопка (<button type="submit">) відправляє форму:

```
            <div class="button-container">
                <form method="post" action="/" enctype="multipart/form-data"
id="uploadForm">
                    <p>
                        <input  type="file"  name="file"  autocomplete="off"
required>
                    </p>
                    <p>
                        <button type="submit">Відновити</button>
                    </p>
                </form>
            </div>
```

div з класом loading містить анімацію завантаження та повідомлення про статус. Вона відображається під час обробки файлу:

```
            <div class="loading" id="loading">
                <img
src="https://cdn.dribbble.com/users/493970/screenshots/1691794/progress.gif"
alt="Loading">
                <div class="status-messages" id="statusMessages"></div>
            </div>
        </div>
```

JavaScript-код обробляє подію відправки форми. Коли користувач натискає кнопку відправки, анімація завантаження (div з класом loading) стає видимою. Масив повідомлень messages містить різні повідомлення про статус, які відображаються кожні 2 секунди, поки триває обробка зображення:

```
            document.getElementById('uploadForm').addEventListener('submit',
function(event) {
                document.getElementById('loading').style.display = 'block';
                let messages = [
```

```
            "Завантаження файлу...",
            "Аналіз фотографії...",
            "Майже готово...",
            "Відновлюємо фото...",
            "Обробка зображення...",
            "Завершуємо відновлення...",
            "Процес відновлення може тривати до 5 хвилин..."
        ];
        let statusDiv = document.getElementById('statusMessages');
        let index = 0;
        statusDiv.innerText = messages[index];
        let interval = setInterval(function() {
            index = (index + 1) % messages.length;
            statusDiv.innerText = messages[index];
        }, 2000);
        event.target.submit();
    });
```

Форма відправляється автоматично після початку показу повідомлень. Щоб візуалізувати алгоритм виконання коду можна використати блок схему. (рис. 3.1).

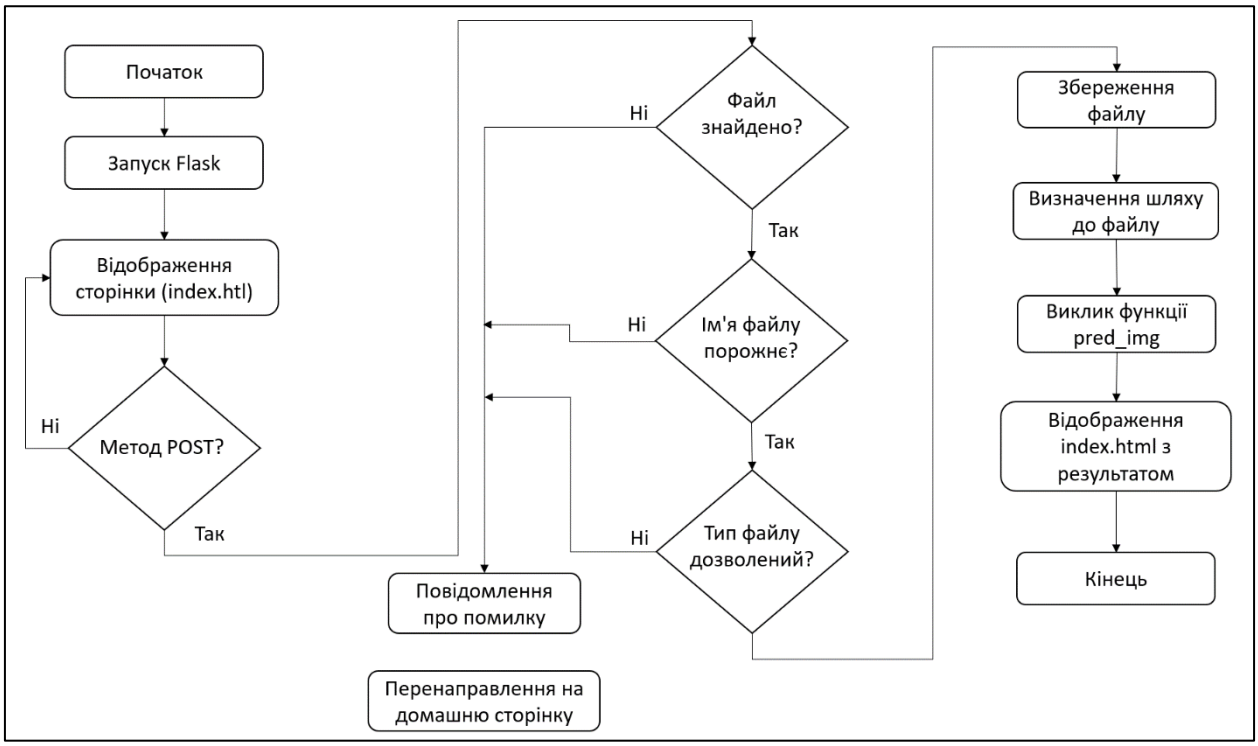

Рисунок 3.1 — Блок схема виконання коду

На блок схемі видно алгоритм виконання коду. Вона ілюструє основні кроки обробки запиту завантаження файлу, перевірки його валідності, збереження і передбачення результату. Вона показує, як система реагує на різні сценарії.

3.3. Тестування та демонстрація роботи системи

Тестування та демонстрація роботи системи є критично важливими етапами розробки будь-якого програмного забезпечення. Вони дозволяють переконатися, що система працює коректно, відповідає вимогам та забезпечує задоволення користувацьких потреб. Тестування також допомагає виявити можливі помилки та недоліки, які можуть бути виправлені перед випуском продукту. Демонстрація роботи системи надає користувачам можливість ознайомитися з її функціональністю та переконатися у її ефективності.

Першим, що побачить користувач, буде простий інтерфейс із заголовком «Відновлення Спотворених фотографій» (рис. 3.2).

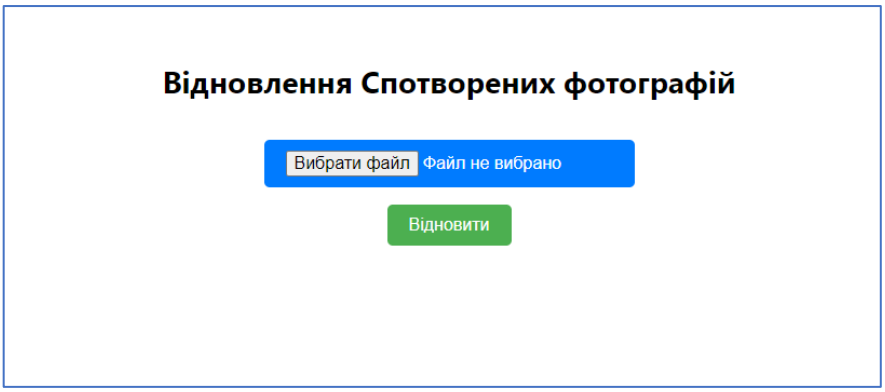

Рисунок 3.2 — Інтерфейс програми

Цей екран містить заголовок, кнопку вибору файлу яка дозволяє користувачам вибрати файл зі свого комп’ютера. Та кнопку відновлення яка відправляє вибраний файл на сервер для обробки. Простий дизайн цього інтерфейсу є великою перевагою, оскільки він забезпечує зрозумілість та легкість у використанні для всіх користувачів. Мінімалістичний підхід допомагає користувачам швидко орієнтуватися та розпочинати процес відновлення фотографій без зайвих кроків.

Коли користувач вибирає файл та натискає кнопку «Відновити», з’являється анімація завантаження та текстові повідомлення (рис. 3.3).

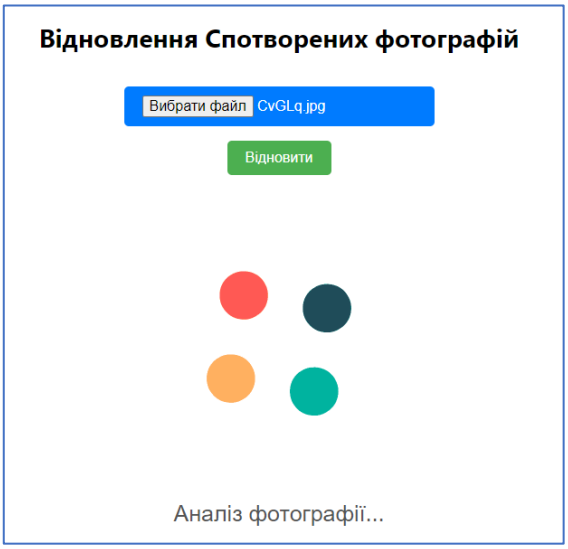

Рисунок 3.3 — Інтерфейс програми

Анімація завантаження: візуально інформує користувача, що процес обробки розпочато. Текстові повідомлення: змінюються під час процесу обробки, інформуючи користувача про поточний стан (наприклад, «Завантаження файлу...», «Аналіз фотографії...», «Майже готово...»).

Це забезпечує користувачів розумінням, що програма працює і не зависла, що покращує користувацький досвід і запобігає непорозумінням.

Після завершення обробки користувач отримує результат (рис. 3.4).

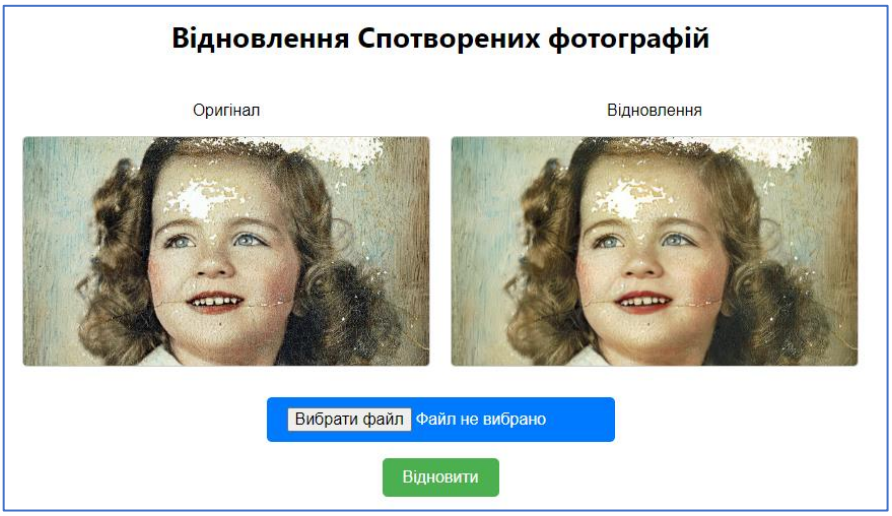

Рисунок 3.4 — Результат відновлення

Відображення оригінального зображення: показує початкове зображення до обробки. Відображення відновленого зображення: показує результат обробки зображення. Цей фінальний екран дозволяє користувачам порівняти оригінал та відновлене зображення, що демонструє ефективність роботи алгоритму.

Зображення та функціональні елементи інтерфейсу працюють згідно з очікуваннями, що дозволяє вважати тестування успішним. Користувачі можуть легко взаємодіяти із системою, отримувати зворотній зв’язок про стан процесу обробки та бачити результати роботи алгоритму. Це підтверджує, що система відповідає вимогам та готова до використання.

ВИСНОВКИ

У даній роботі було проведено комплексне дослідження та реалізацію системи для відновлення спотворених фотографій з використанням сучасних методів на основі глибоких нейронних мереж. Основні етапи виконаної роботи включали: Було детально розглянуто різні підходи до відновлення зображень, включаючи традиційні методи (фільтраційні та інтерполяційні) та сучасні методи на основі глибоких нейронних мереж, такі як SRCNN, Denoising Autoencoders (DAE), Old Photo Restoration via Deep Latent Space Translation та Deep Image Prior (DIP). Було обґрунтовано вибір технологій для фронтенду (HTML, CSS, JavaScript) та бекенду (Python, Flask). Для відновлення фотографій було інтегровано API бібліотеки Replicate.

Створено простий та інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам легко завантажувати фотографії, спостерігати за процесом обробки та отримувати результати. Інтерфейс забезпечує високу зручність використання завдяки мінімалістичному дизайну та інтерактивним елементам. Проведено тестування системи, яке підтвердило коректність її роботи та відповідність функціональності заданим вимогам. Користувачі можуть завантажувати фотографії, бачити процес обробки у вигляді анімації та отримувати відновлені зображення.

Результатом виконання роботи є програма, яка дозволяє відновлювати старі та спотворені фотографії за допомогою сучасних алгоритмів на основі глибоких нейронних мереж. Програма забезпечує високу якість відновлення, зручність використання та інтерактивний інтерфейс, що робить її доступною для широкого кола користувачів. У подальшому програма може вдосконалюватися шляхом оптимізації алгоритмів та використання різних API для відновлення фотографій. Це дозволить підвищити ефективність роботи, покращити якість відновлених зображень та розширити функціональність системи, забезпечуючи ще кращий користувацький досвід.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Методи обробки зображень: Класичні методи обробки зображень [Електронний ресурс] // Posibnyk. – 2023. – Режим доступу до ресурсу: web.posibnyky.vntu.edu.ua/fksa/2kvetnyj_komp\'yuterne_modelyuvannya_system_procesiv/t2/2..htm.
2. Regularization in Image Restoration and Reconstruction [Електронний ресурс] // Researchgate. – 2005. – Режим доступу до ресурсу: https://www.researchgate.net/publication/247379368_Regularization_in_Image_Restoration_and_Reconstruction.
3. A method for remote sensing image restoration based on the system degradation model [Електронний ресурс] // Results in Physics. – 2024. – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/pii/S2211379723010550.
4. Development of Methods for Image Filtering in Noise [Електронний ресурс] // ResearchGate. – 2022. – Режим доступу до ресурсу: https://www.researchgate.net/publication/368598573_Development_of_Methods_for_Image_Filtering_in_Noise_and_their_Implementation_for_a_Web_Service.
5. How Faster R-CNNs work part3(Machine Learning 2023) [Електронний ресурс] // Medium. – 2023. – Режим доступу до ресурсу: https://medium.com/@monocosmo77/how-faster-r-cnns-work-part3-machine-learning-2023-575715db0a16.
6. Image Filtering Techniques in Image Processing — Part 1 [Електронний ресурс] // Medium. – 2023. – Режим доступу до ресурсу: https://medium.com/@henriquevedoveli/image-filtering-techniques-in-image-processing-part-1-d03362fc73b7.
7. Метод відновлення зображення на основі математичного апарату штучних нейронних мереж удосконаленого методу ієрархічної інтерполяції [Електронний ресурс] // jiks. – 2018. – Режим доступу до ресурсу: http://jiks.kart.edu.ua/article/view/146695.
8. Regularization Method [Електронний ресурс] // ScienceDirect. – 2003. – Режим доступу до ресурсу: https://www.sciencedirect.com/topics/engineering/regularization-method.
9. Review: SRCNN (Super Resolution) [Електронний ресурс] // Medium. – 2018. – Режим доступу до ресурсу: https://medium.com/coinmonks/review-srcnn-super-resolution-3cb3a4f67a7c.
10. SRCNN Paper Summary & Implementation [Електронний ресурс] // Medium. – 2021. – Режим доступу до ресурсу: https://medium.com/analytics-vidhya/srcnn-paper-summary-implementation-ad5cea22a90e.
11. Denoising Autoencoders (DAE) — How To Use Neural Networks to Clean Up Your Data [Електронний ресурс] // owardsdatascience. – 2022. – Режим доступу

до ресурсу: https://towardsdatascience.com/denoising-autoencoders-dae-how-to-use-neural-networks-to-clean-up-your-data-cd9c19bc6915.
12. Denoising Autoencoders (DAEs) [Електронний ресурс] // opengenus. – 2024. – Режим доступу до ресурсу: https://iq.https://iq.opengenus.org/denoising-autoencoders/.org/denoising-autoencoders/.
13. Bringing-Old-Photos-Back-to-Life [Електронний ресурс] // microsoft. – 2020. – Режим доступу до ресурсу: https://github.com/microsoft/Bringing-Old-Photos-Back-to-Life.
14. Bringing Old Photos Back to Life [Електронний ресурс] // arxiv. – 2020. – Режим доступу до ресурсу: https://arxiv.org/abs/2004.09484.
15. Deep Image Prior [Електронний ресурс] // 2024. – Режим доступу до ресурсу: https://openaccess.thecvf.com/content_cvpr_2018/papers/Ulyanov_Deep_Image_Prior_CVPR_2018_paper.pdf.
16. Deep Image Prior in PyTorch [Електронний ресурс] // Medium. – 2022. – Режим доступу до ресурсу: https://towardsdatascience.com/deep-image-prior-in-pytorch-e6edf666a480.
17. What is the formula for calculating PSNR from two images? [Електронний ресурс] // Quora. – 2024. – Режим доступу до ресурсу: https://www.quora.com/What-is-the-formula-for-calculating-PSNR-from-two-images.
18. Structural similarity (SSIM) index for measuring image quality [Електронний ресурс] // mathworks. – 2024. – Режим доступу до ресурсу: https://www.mathworks.com/help/images/ref/ssim.html.
19. Run AI with an API [Електронний ресурс] // Replicate. – 2024. – Режим доступу до ресурсу: https://replicate.com/.