Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ЗАСТОСУНКУ ЗБІЛЬШЕННЯ РАСТРОВИХ ЗОБРАЖЕНЬ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-1
Овсянніков Д. О.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір В. П.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Овсяннікову Дмитру Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка застосунку збільшення растрових зображень

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 01 червня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом OpenCV.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів інтерполяції зображення та аналіз їх можливостей.
2. Предметне порівняння білінійного біквадратичного і бікубічного методів інтерполяції.
3.Розробка застосунку для наочного і предметного порівняння методів інтерполяції.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Мотивація, постановка задачі, теоретичне дослідження, експериментальні дослідження, отримані результати і висновки.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів і бібліотек	15.04.25-20.04.25	
5	Розробка методу	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-22.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____проф. Машталір В. П._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 89 с., 4 табл., 29 рис., 1 дод., 42 джерело.

БІЛІНІЙНА ІНТЕРПОЛЯЦІЯ, БІКУБІЧНА ІНТЕРПОЛЯЦІЯ, БІКВАДРАТИЧНА ІНТЕРПОЛЯЦІЯ, МАСШТАБУВАННЯ ЗОБРАЖЕНЬ, РАСТРОВІ ЗОБРАЖЕННЯ, МЕТРИКИ ЯКОСТІ, ЦИФРОВА ОБРОБКА ЗОБРАЖЕНЬ.

Об’єктом роботи є растрові зображення, що підлягають масштабуванню з використанням методів інтерполяції.

Метою роботи є розробка програмного застосунку для масштабування растрових зображень із використанням білінійної, бікубічної та біквадратичної інтерполяції, а також порівняльний аналіз їхньої ефективності за кількісними метриками та візуальними характеристиками на різних типах зображень.

Досліджено принципи масштабування растрових зображень, методи інтерполяції, метрики якості. Реалізовано програмний застосунок із підтримкою візуалізації результатів.

Актуальність роботи полягає у потребі масштабування растрових зображень у сферах, де класичні методи інтерполяції є затребуваними через їх простоту, а також у необхідності їхнього порівняльного аналізу в різних сценаріях.

BILINEAR INTERPOLATION, BICUBIC INTERPOLATION, BIQUADRATIC INTERPOLATION, IMAGE SCALING, RASTER IMAGES, QUALITY METRICS, DIGITAL IMAGE PROCESSING.

The object of the work is raster images in PNG format, including different types such as medical images, landscapes, and program/game logos, subjected to scaling.

The aim of the work is to develop a software application for scaling raster images in PNG format using bilinear, bicubic, and biquadratic interpolation, and to compare their effectiveness via quantitative metrics and visual characteristics on different types of images.

The principles of raster image scaling, interpolation methods, and quality metrics were studied. A software application with asynchronous processing and result visualization was implemented.

The relevance of the work lies in the need for raster image scaling in spheres where classical interpolation methods remain in demand due to simplicity, as well as in the need for their comparative analysis in various scenarios.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 7
Вступ 9
1 Аналіз методів збільшення растрових зображень шляхом інтерполяції 10
1.1 Предметна область цифрової обробки зображень 10
1.1.1 Растрова графіка: визначення та характеристики 11
1.1.2 Роздільна здатність зображення 12
1.1.3 Масштабування растрових зображень: основна проблема збільшення 14
1.1.4 Піраміди зображень та їх роль у масштабуванні 16
1.2 Методи інтерполяції для збільшення зображень 18
1.2.1 Загальні принципи інтерполяції в обробці зображень 19
1.2.2 Білінійна інтерполяція: аналіз та порівняння 22
1.2.3 Бікубічна інтерполяція: аналіз та порівняння 23
1.2.4 Біквадратична інтерполяція: аналіз та порівняння 25
1.2.5 Огляд інших методів та порівняння 27
1.3 Постановка задачі 28
2 Архітектура системи збільшення растрових зображень 30
2.1 Загальний опис системи 30
2.2 Архітектура фронтенду 31
2.2.1 Технологічний стек 31
2.2.2 UML-діаграма фронтенду 32
2.2.3 Опис компонентів 34
2.3 Архітектура бекенду 35
2.3.1 Технологічний стек 35
2.3.2 UML-діаграма взаємодії 36
2.3.3 Опис API та WebSocket 37
2.4 Логіка обробки зображень 39
2.4.1 Методи інтерполяції 39

2.4.2 Процес обробки 42
2.4.3 Формули метрик 43
2.5 Аналіз продуктивності 48
2.5.1 Логіка аналізу 48
2.5.2 Приклад результатів 49
3 Комп’ютерна модель масштабування зображень 50
3.1 Обґрунтування вибору середовища програмної реалізації 50
3.2 Інструкція користувача 55
3.3 Тестування розробленої мод елі 60
3.4 Експеременталь ні дослідження 68
Висновки 74
Перелік Джерел посилання 76
Додаток А Тестові зображення 80

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

dpi – dots per inch (одиниця вимірювання роздільної здатності для друку)

ppi – pixels per inch (одиниця вимірювання роздільної здатності для екранів)

PSNR – Peak Signal-to-Noise Ratio (метрика оцінки якості зображень)

SSIM – Structural Similarity Index Measure (метрика оцінки якості зображень)

MSE – Mean Squared Error (метрика оцінки якості зображень)

C0 – неперервність функції (нульового порядку), математичний термін, що означає відсутність розривів у функції

C1 – неперервність перших похідних (першого порядку), математичний термін, що означає гладкість функції

CNN – Convolutional Neural Network (тип нейронної мережі для обробки зображень)

GAN – Generative Adversarial Network (тип нейронної мережі для генерації даних)

SRGAN – Super-Resolution Generative Adversarial Network (генеративно-змагальна мережа для супер-роздільності)

NEDI – New Edge-Directed Interpolation (нова інтерполяція, спрямована на краї)

DCCI – Directional Cubic Convolution Interpolation (спрямована кубічна згорткова інтерполяція)

OpenCV – Open Source Computer Vision Library (бібліотека комп’ютерного зору з відкритим кодом)

ГІС – геоінформаційна система

TIFF – Формат файлу для зберігання растрових зображень, який підтримує високу якість і гнучкість. TIFF дозволяє зберігати зображення без втрат (lossless) або зі стисненням, підтримує кілька шарів, альфа-канали

PNG – Формат растрових зображень, який підтримує стиснення без втрат (lossless), прозорість (альфа-канали) та широкий діапазон кольорів. PNG розроблений як заміна GIF і частково TIFF для вебвикористання

ВСТУП

Цифрова обробка зображень є важливою галуззю комп’ютерної науки, що має широке застосування – від розваг, мультимедіа та ігор до медичної діагностики, систем безпеки й наукових досліджень. Поширення цифрових пристроїв, таких як камери та екрани високої роздільної здатності, зробило маніпуляцію зображеннями повсякденною потребою.

Одна з ключових операцій – масштабування, зокрема збільшення зображень, необхідне для розгляду дрібних деталей, адаптації до екранів 4K/8K, підготовки матеріалів для друку чи використання в задачах комп’ютерного зору. Проте збільшення растрових зображень, що складаються з дискретних пікселів, викликає проблеми. Просте дублювання пікселів призводить до деградації якості, що проявляється у розмитті, зубчастості країв, втраті чіткості деталей і спотворенні текстур.

Для вирішення цих недоліків застосовуються методи інтерполяції, які оцінюють значення нових пікселів на основі вихідного зображення. У цій роботі досліджуються три класичні методи: білінійна, бікубічна та біквадратична інтерполяція. Ці методи є базовими і часто використовуються в обробці зображень, але їхній порівняльний аналіз, особливо з урахуванням біквадратичного методу, залишається актуальним.

Актуальність роботи полягає в розробці застосунку для збільшення растрових зображень із використанням білінійного, бікубічного та біквадратичного методів інтерполяції, а також у проведенні аналізу їхньої ефективності, якості результатів і продуктивності в різних сценаріях.

1 АНАЛІЗ МЕТОДІВ ЗБІЛЬШЕННЯ РАСТРОВИХ ЗОБРАЖЕНЬ ШЛЯХОМ ІНТЕРПОЛЯЦІЇ

1.1 Предметна область цифрової обробки зображень

Цифрова обробка зображень є міждисциплінарною галуззю, що об’єднує комп’ютерні науки, математику та інженерію для аналізу, трансформації та вдосконалення візуальних даних [1]. Вона лежить в основі численних застосунків, від цифрової фотографії, вебдизайну та комп’ютерних ігор до медичної діагностики [2], супутникових знімків і систем безпеки [3]. У цій роботі центральним завданням є масштабування растрових зображень у форматі PNG – процес, що вимагає глибокого розуміння їхньої піксельної структури, роздільної здатності та методів обробки. Растрові зображення, через свою фіксовану природу, створюють унікальні виклики при збільшенні, що обумовлює необхідність інтерполяційних алгоритмів для збереження візуальної якості та досягнення компромісу між якістю й обчислювальною швидкістю.

Теоретична основа роботи включає аналіз методів інтерполяції, визначаючи ключові поняття: растрову графіку, роздільну здатність, принципи масштабування та багатомасштабний аналіз через піраміди зображень [4]. Ці концепції є критичними для розуміння обмежень і можливостей алгоритмів, таких як білінійна, бікубічна та біквадратична інтерполяція, які досліджуються в роботі.

Історично цифрова обробка зображень зародилася в 1960-х роках, коли комп’ютери почали обробляти супутникові знімки для космічних місій NASA. Наприклад, зображення поверхні Місяця вимагали масштабування для аналізу деталей, що стало поштовхом до розвитку інтерполяційних методів. Сьогодні ця галузь охоплює комп’ютерний зір, глибоке навчання та обробку в реальному часі, застосовуючись у задачах, як реставрація старих фотографій, адаптивний рендеринг у вебінтерфейсах і аналіз МРТ-знімків [2]. У

комп’ютерних іграх масштабування зображень є ключовим для згладжування текстур у реальному часі, де прості методи, як метод найближчого сусіда та білінійна інтерполяція, використовуються для швидкості, хоча це може призводити до втрати чіткості. Ця робота також шукає компроміс між якістю результату та швидкістю обробки, що є актуальним для локальних систем із обмеженими ресурсами.

Растрові зображення домінують завдяки своїй здатності відтворювати складні сцени, але їхня піксельна структура ускладнює масштабування [5]. Піраміди зображень дозволяють аналізувати дані на різних рівнях деталізації, що є основою для тестування інтерполяційних методів [6].

Сучасні тенденції в цифровій обробці зображень також включають використання нейронних мереж для супер-роздільності, що дозволяє генерувати деталі, яких немає у вихідному зображенні. Наприклад, такі методи застосовуються в реставрації архівних матеріалів, де старі фотографії отримують нове життя завдяки алгоритмам, що «домальовують» втрачені елементи. Проте для локальних систем із обмеженими ресурсами класичні методи інтерполяції залишаються незамінними через їхню ефективність і меншу залежність від обчислювальної потужності.

1.1.1 Растрова графіка: визначення та характеристики

Растрова графіка, також відома як точкова графіка, є способом представлення цифрових зображень за допомогою двовимірної сітки або матриці окремих елементів, що називаються пікселями [7]. Кожен піксель у цій сітці має свої унікальні координати та певний атрибут, найчастіше колір або рівень яскравості. Таким чином, растрове зображення можна уявити як мозаїку, складену з великої кількості маленьких однорідних за кольором прямокутних або квадратних елементів.

Фундаментальною характеристикою растрової графіки є її залежність від роздільної здатності. Якість та деталізація растрового зображення безпосередньо визначаються загальною кількістю пікселів, з яких воно складається [8]. На відміну від векторної графіки, де зображення описуються математичними формулами (лініями, кривими, фігурами), растрові зображення мають фіксовану піксельну структуру.

Саме ця фіксована структура піксельної сітки лежить в основі проблеми масштабування [9]. Коли потрібно збільшити розмір растрового зображення, наприклад, подвоїти його ширину та висоту, потрібно також зробити нову, більшу сітку пікселів. У цій новій сітці з’являються позиції, для яких у вихідному зображенні не було відповідних пікселів. Просте «розтягування» пікселів призводить до втрати якості. Отже, процес збільшення растрового зображення неминуче вимагає створення нової піксельної інформації для заповнення цих «прогалин». Це завдання вирішується за допомогою алгоритмів інтерполяції, які намагаються обчислити найбільш правдоподібні значення для нових пікселів на основі існуючих [10]. Вибір конкретного алгоритму інтерполяції безпосередньо впливає на візуальну якість кінцевого, збільшеного зображення.

Піксельна структура растрових зображень також впливає на їхню поведінку при різних типах трансформацій, наприклад, повороті чи деформації. У таких випадках також застосовуються методи інтерполяції, щоб уникнути спотворень, таких як аліасинг. Це робить растрову графіку універсальним, але водночас складним об’єктом для обробки, особливо коли йдеться про збереження деталей при зміні розміру.

1.1.2 Роздільна здатність зображення

Роздільна здатність є ключовим параметром, що характеризує якість та деталізацію растрового зображення [11]. Вона визначає щільність пікселів на

одиницю фізичної довжини. Зазвичай роздільну здатність вимірюють у пікселях на дюйм (pixels per inch) для зображень, призначених для відображення на екранах, або в точках на дюйм (dots per inch) для зображень, що готуються до друку. Чим вище значення ppi або dpi, тим більше пікселів припадає на один дюйм, і, відповідно, тим вищою є деталізація, різкість та загальна якість зображення [12].

Важливо розуміти, що інтерпретація роздільної здатності залежить від контексту використання зображення. Роздільна здатність екрана монітора чи дисплея мобільного пристрою, що вимірюється кількістю пікселів по горизонталі та вертикалі (наприклад, Full HD: 1920×1080 пікселів), визначає максимальну чіткість, з якою може бути відображено зображення. Якщо піксельні розміри зображення співпадають із роздільною здатністю монітора, воно заповнить весь екран при 100% масштабі. У поліграфії роздільна здатність зображення (в dpi) визначає якість друкованого відбитка. Для якісного друку, наприклад, журналів чи брошур, часто вимагається роздільна здатність 300 dpi [13].

Слід розрізняти піксельні «розміри» (загальна кількість пікселів по ширині та висоті) та «роздільну здатність» (кількість пікселів на дюйм) [14]. Зміна лише метаданих роздільної здатності у файлі зображення без зміни загальної кількості пікселів (тобто без передискретизації або resampling) не змінює самі дані зображення, а лише впливає на його фізичний розмір при друці або відображенні [7].

Зв’язок між роздільною здатністю та необхідністю масштабування стає очевидним, коли зображення з низькою роздільною здатністю потрібно відобразити або надрукувати у великому фізичному розмірі [8]. У такому випадку окремі пікселі стають помітними, що призводить до ефекту «пікселізації» або блоковості. Щоб покращити сприйману деталізацію при збільшенні фізичного розміру, необхідно збільшити загальну кількість пікселів зображення [9]. Саме ця потреба в збільшенні піксельних розмірів для

досягнення бажаної якості при заданому фізичному розмірі обумовлює застосування методів інтерполяції [10].

Сучасні екрани з високою роздільною здатністю, як Retina-дисплеї, ще більше підкреслюють важливість правильного масштабування. Наприклад, зображення, оптимізоване для звичайного екрану (72 ppi), на Retina-дисплеї (218 ppi) виглядатиме розмитим, якщо не застосувати інтерполяцію для адаптації до вищої щільності пікселів. Це показує, як роздільна здатність впливає не лише на друк, а й на цифрове відображення.

1.1.3 Масштабування растрових зображень: основна проблема збільшення

Масштабування растрового зображення, зокрема його збільшення, означає зміну його піксельних розмірів – збільшення кількості пікселів по ширині та/або висоті [11]. Оскільки вихідне зображення містить лише обмежену, фіксовану кількість пікселів, процес збільшення створює нові позиції в розширеній піксельній сітці, для яких значення кольору чи яскравості невідомі. Наприклад, при збільшенні зображення в 4 рази, на місці одного вихідного пікселя з’являються чотири нових, і лише один із них може успадкувати значення вихідного пікселя напряму. Значення решти потрібно обчислити [12].

Найпростіші методи збільшення, такі як метод найближчого сусіда, просто копіюють значення найближчого вихідного пікселя у нові позиції. Це призводить до появи грубих візуальних артефактів, зокрема блоковості та вираженого «східчастого» ефекту на діагональних лініях та кривих [13]. Хоча такий підхід зберігає вихідні деталі без розмиття, результат часто є візуально незадовільним. Наприклад, збільшення медичного зображення (МРТ-знімка) з 256×256 до 1024×1024 пікселів методом найближчого сусіда призведе до помітної зубчастості контурів, що ускладнить діагностику [2].

Для отримання більш гладких результатів використовуються алгоритми інтерполяції [7]. Вони призначені для оцінки значень нових пікселів на основі відомих сусідніх пікселів із вихідного зображення. Мета інтерполяції – створити плавні переходи між пікселями та зберегти якомога більше візуальної інформації, мінімізуючи артефакти, такі як шум, розмиття чи блочна структура [8]. Однак сам процес масштабування растрових форматів супроводжується труднощами [9].

Масштабування шляхом інтерполяції – це не просто перерозподіл існуючих пікселів, а процес оцінки інформації, яка не була явно присутня у вихідному зображенні нижчої роздільної здатності [10]. Вихідні пікселі надають відомі дані, а алгоритм інтерполяції виступає як метод для обчислення значень у невідомих точках нової сітки [11].

Різні методи інтерполяції є різними стратегіями для цієї оцінки: «кращий» алгоритм робить більш правдоподібні припущення про відсутні значення, що призводить до меншої кількості артефактів та кращого збереження деталізації [12]. Проте жоден алгоритм не може ідеально відновити деталі, які були втрачені через низьку роздільну здатність вихідного зображення – це фундаментальне обмеження процесу збільшення [13].

Основна проблема масштабування полягає в неможливості створення нових деталей, яких немає у вихідному зображенні [8]. Наприклад, у медичній візуалізації розмиття контурів на МРТ-знімку через білінійну інтерполяцію може ускладнити діагностику, а ореоли від бікубічної інтерполяції – спотворити форму об’єктів [6]. У комп’ютерних іграх масштабування текстур методом найближчого сусіда призводить до зубчастості, а білінійна інтерполяція розмиває деталі, погіршуючи візуальний досвід [7]. Основні артефакти: розмиття через усереднення (білінійна інтерполяція); зубчастість на лініях (найближчий сусід); ореоли навколо контрастних країв (бікубічна); муар при масштабуванні регулярних текстур [8, 9]. Складніші методи зменшують артефакти, але підвищують обчислювальну складність [5]. Ці

артефакти залежать від типу зображення: текст стає нечітким через зубчастість, а пейзажі страждають від розмиття.

Складність масштабування також залежить від типу зображення. Наприклад, зображення з великою кількістю дрібних деталей, як текст чи складні текстури, потребують особливо обережного підходу, адже навіть невелике розмиття чи зубчастість можуть зробити текст нечитабельним або текстуру неприродною. У таких випадках вибір методу інтерполяції стає ще більш критичним, адже він визначає, чи залишиться зображення функціональним після збільшення.

1.1.4 Піраміди зображень та їх роль у масштабуванні

Представлення зображення на різних рівнях деталізації є фундаментальною концепцією в цифровій обробці зображень [8]. Одним із класичних підходів до створення багатомасштабних представлень є побудова пірамід зображень [9]. Піраміда зображень – це послідовність копій одного зображення, кожна з яких має нижчу роздільну здатність, ніж попередня [10]. Це дозволяє аналізувати зображення як на рівні дрібних деталей (верхні рівні піраміди), так і на рівні загальної структури (нижні рівні) [11].

Для завдань, пов’язаних зі зменшенням зображень та аналізом на різних масштабах, часто використовується Гауссова піраміда (рис. 1.1) [16]. Процес її побудови є ітеративним і включає два основні кроки на кожному рівні:

– до зображення застосовується фільтр низьких частот, зазвичай Гауссовський, який прибирає шум та різкі переходи [13]. Цей крок запобігає ефекту аліасингу – появи небажаних візерунків чи спотворень, що виникають при дискретизації сигналів із високими частотами (дрібними деталями) [7];

– після згладжування розмір зображення зменшується, зазвичай удвічі по кожній осі (ширині та висоті) [8]. Наприклад, із блоку 2×2 пікселів

вихідного зображення залишається один піксель у зменшеному зображенні [9].

Ці кроки повторюються для кожного наступного рівня піраміди, починаючи з вихідного зображення (рівень 0) [10]. Кожен наступний рівень Гуссової піраміди є зменшеною та згладженою версією попереднього [11].

Візуально Гауссова піраміда представлена як стопка зображень, де найширше (вихідне) зображення – внизу, а кожне наступне вище є меншим і більш розмитим, формуючи структуру, схожу на піраміду (рис. 1.1) [15].

На рисунку показано, як із вихідного зображення шляхом послідовного згладжування та зменшення вдвічі формуються наступні рівні піраміди (рівень 1, рівень 2 тощо), кожен із яких має вдвічі менші розміри порівняно з попереднім [16].

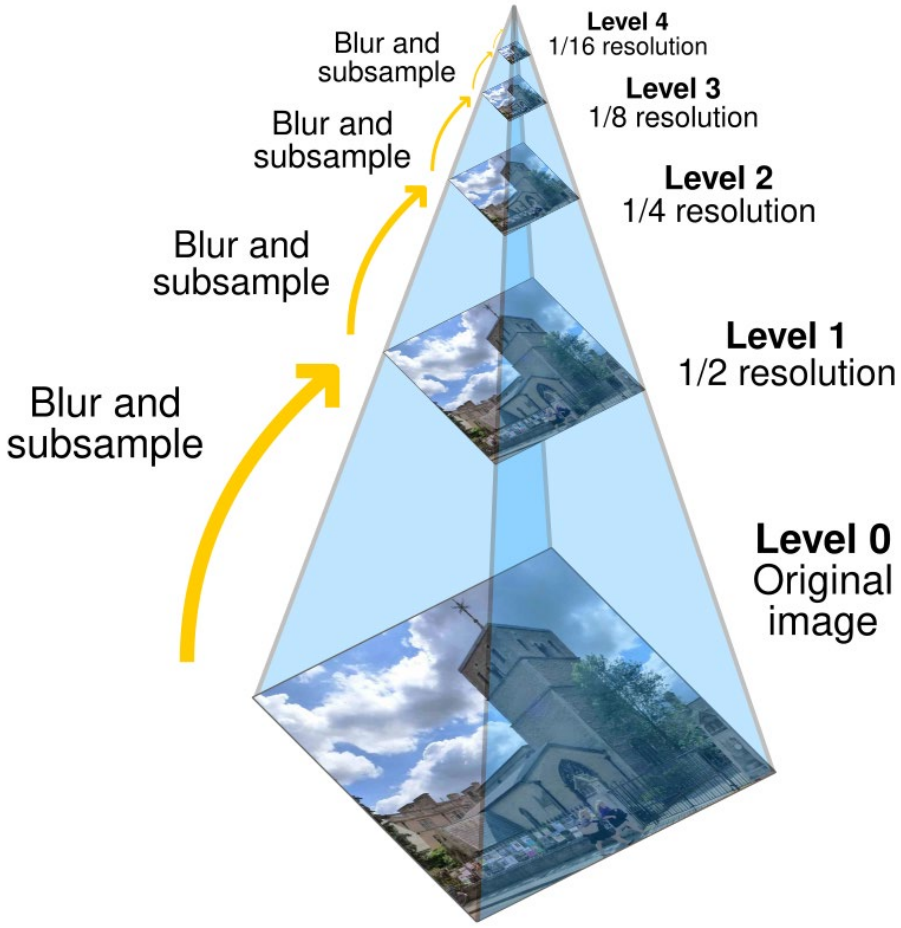


Рисунок 1.1 – Концепція Гауссової піраміди зображень [16]

Гауссові піраміди мають важливе значення в обробці зображень не тільки для зменшення, але й для завдань збільшення та аналізу на різних масштабах [13]. Наприклад, вони використовуються для швидкого пошуку об’єктів: пошук великих об’єктів проводиться на нижчих рівнях піраміди, що значно швидше, ніж на вихідному зображенні [17]. Концепція багатомасштабності, закладена в пірамідах, використовується в складних алгоритмах для кращого відновлення деталей при збільшенні [7].

Розуміння принципів побудови Гауссових пірамід, зокрема згладжування Гауссовим фільтром та зменшення розміру вдвічі, є важливим для методів обробки зображень, які працюють із різними рівнями деталізації та масштабу [8]. Піраміди зображень також знаходять застосування в сучасних алгоритмах комп’ютерного зору, наприклад, у задачах розпізнавання об’єктів. Такі бібліотеки, як OpenCV, використовують піраміди для оптимізації пошуку об’єктів різного розміру, що дозволяє швидше обробляти великі зображення, зберігаючи високу точність.

1.2 Методи інтерполяції для збільшення зображень

Інтерполяція є ключовим інструментом для вирішення проблеми збільшення растрових зображень, дозволяючи оцінювати значення нових пікселів у розширеній сітці на основі вихідних даних [12]. У цифровій обробці зображень, де якість і продуктивність є критичними, вибір методу інтерполяції визначає баланс між візуальною чіткістю та обчислювальною ефективністю [13].

Збільшення растрових зображень вимагає створення нових пікселів, коли вихідне зображення не містить достатньо даних [7]. Наприклад, масштабування зображення 100×100 пікселів до 400×400 створює 160 000 нових пікселів, для яких потрібно оцінити значення кольору чи яскравості. Інтерполяція забезпечує правдоподібну оцінку, використовуючи сусідні

пікселі, але різні методи дають різні результати: від швидких, але розмитих (білінійна) до якісних, але повільніших (бікубічна) [8]. Такі методи часто застосовуються в задачах сегментації зображень, де важлива просторова структура даних [19]

У комп'ютерних іграх, де рендеринг текстур у реальному часі потребує швидкості, білінійна інтерполяція є стандартом, хоча вона може розмити деталі, як у текстурах ландшафту чи персонажів [20]. Ця робота пропонує біквадратичний метод, який оптимізує компроміс між якістю та продуктивністю, що особливо цінно для локальних систем із обмеженими ресурсами.

Історично інтерполяція в обробці зображень розвинулася разом із комп'ютерною графікою [9]. У 1970-х роках прості методи, як найближчий сусід, використовувалися в перших растрових дисплеях, але в 1980-х роках бікубічна інтерполяція стала стандартом завдяки роботам Катмулла та Рома. Сьогодні методи інтерполяції застосовуються в програмах, як Adobe Photoshop і графічних движках ігор (Unity, Unreal Engine), де швидкість і якість є ключовими [15]. Сучасні графічні редактори, такі як GIMP чи Affinity Photo, також активно використовують ці методи, пропонуючи користувачам вибір між швидкістю та якістю залежно від завдання. Наприклад, для швидкого попереднього перегляду зображення в вебдизайні білінійна інтерполяція залишається популярною, тоді як для підготовки до друку професійні фотографи частіше обирають бікубічну через її здатність зберігати деталі.

1.2.1 Загальні принципи інтерполяції в обробці зображень

З математичної точки зору, інтерполяція – це процес знаходження наближених значень деякої функції для проміжних значень аргументу, якщо відомі значення цієї функції в заданому наборі точок [10]. У контексті цифрової обробки зображень, «функція» представляє собою розподіл

інтенсивності або значень колірних компонентів пікселів по площині зображення, а «вузлами» є центри пікселів вихідного зображення з їх відомими значеннями [11].

Метою інтерполяції при збільшенні зображення є створення умовно неперервного представлення зображення на основі його дискретних піксельних даних [12]. Це дозволяє обчислити значення функції (тобто колір або яскравість) у нових точках, що відповідають центрам пікселів у збільшеній сітці [13]. Чим точніше інтерполяційний метод може оцінити ці проміжні значення, тим якіснішим буде результат збільшення [20].

Для виконання інтерполяції можуть використовуватися різноманітні математичні підходи [7]. До основних належать:

– лінійна інтерполяція – є одним з найпростіших методів інтерполяції, процес полягає в простому з'єднанні відомих точок прямими лініями [8];

– поліноміальна інтерполяція – використання поліномів (многочленів) певного ступеня (наприклад, за допомогою формул Лагранжа або Ньютона) для апроксимації функції між вузлами [9];

– сплайн-інтерполяція – використання кусково-поліноміальних функцій (сплайнів), які забезпечують гладкість не тільки самої функції, а й її похідних у вузлах [10];

– інтерполяція за допомогою спеціалізованих функцій – це наприклад, використання функції sinc (Кардинальний сінус) або функцій Гауса, які мають специфічні властивості у частотній області [11]. Хоча sinc є ідеальним фільтром у частотній області, що теоретично забезпечує ідеальну інтерполяцію, на практиці вона обмежена через обчислювальну складність [12].

При масштабуванні двовимірних зображень ці одновимірні методи інтерполяції узагальнюються на дві координати [13]. Популярні бібліотеки для обробки зображень, такі як OpenCV, надають реалізації кількох стандартних методів інтерполяції, серед яких: інтерполяція методом найближчого сусіда (INTER_NEAREST), білінійна (INTER_LINEAR), бікубічна (INTER_CUBIC),

передискретизація з урахуванням площі (INTER_AREA) та інтерполяція Ланцоша (INTER_LANCZOS4) [15].

Вибір методу залежить від вимог до якості та доступних обчислювальних ресурсів [20]. Варто зазначити, що сучасні бібліотеки, як OpenCV, також дозволяють комбінувати методи інтерполяції з іншими техніками обробки, наприклад, із попередньою фільтрацією зображення для зменшення шуму. Це може покращити результат інтерполяції, особливо для зображень із високим рівнем шуму, таких як старі скановані документи. У таблиці 1.1 представлено короткий огляд методів.

Таблиця 1.1 – Порівняльна характеристика методів інтерполяції

Характеристика (Characteristic)	**Білінійна (Bilinear)**	**Бікубічна (Bicubic)**	**Біквадратична (Biquadratic)**
Розмір околу (Neighborhood)	2×2	4×4	3×3
Математична основа (Basis)	Лінійна (Linear)	Кубічна (Cubic)	Квадратична (Quadratic)
Обчисл. складність (Complexity)	Низька (Low)	Середня (Medium)	Низька/Середня (Low/Med)
Типова якість (Typical Quality)	Гладка, розмита (Smooth, Blurry)	Різка, детальна (Sharp, Detailed)	Проміжна (Intermediate)
Неперервність (Continuity)	C0	C1	C0
Основні артефакти (Artifacts)	Розмиття, зубчастість (Blur, Jaggedness)	Ореоли (Halos)	Розмиття Дискретність (Blur, Discontinuities)

1.2.2 Білінійна інтерполяція: аналіз та порівняння

Білінійна інтерполяція є узагальненням одновимірної лінійної інтерполяції на випадок функції двох змінних [7]. При збільшенні зображення цей метод обчислює значення нового пікселя шляхом взяття зваженого середнього значення чотирьох найближчих до нього пікселів (окіл 2×2) з вихідного зображення [8]. Процес можна описати як послідовне виконання лінійної інтерполяції: спочатку вздовж однієї осі (наприклад, горизонтальної) для двох пар сусідніх пікселів, а потім ще раз вздовж іншої осі (вертикальної) між отриманими проміжними значеннями [20].

Вага кожного з чотирьох сусідніх пікселів у середньому визначається його відстанню до точки, в якій обчислюється нове значення: чим ближче піксель, тим більша його вага [9]. Якщо точка нового пікселя знаходиться точно посередині між чотирма вихідними пікселями, її значення буде простим середнім арифметичним їх значень [10].

Переваги білінійної інтерполяції:

– відносна простота та швидкість. Алгоритм є обчислювально менш складним порівняно з методами вищого порядку, такими як бікубічна інтерполяція, що робить його привабливим для застосувань, де швидкість є критичною [15];

– згладжування. У порівнянні з методом найближчого сусіда, білінійна інтерполяція дає значно більш гладкі результати, ефективно усуваючи блоковість та різкі переходи між пікселями [20].

Недоліки білінійної інтерполяції:

– розмиття зображення. Головним недоліком є тенденція до згладжування не тільки небажаних артефактів, але й корисних деталей та країв зображення, що призводить до втрати чіткості та загального розмиття [21];

– залишкова зубчастість. Хоча значно краще, ніж у методі найближчого сусіда, діагональні лінії та криві все ще можуть виглядати дещо зубчастими або нерівними [15];

– неперервність C0. З математичної точки зору, поверхня, що генерується білінійною інтерполяцією, є неперервною (C0), але її перші похідні (градієнти) мають розриви на межах між вихідними пікселями, що візуально сприймається як недостатня гладкість переходів [21].

Білінійна інтерполяція посідає місце базового методу згладжувальної інтерполяції [11]. Вона є першим кроком у покращенні якості порівняно з найпростішим методом найближчого сусіда, оскільки враховує інформацію від найближчих чотирьох пікселів і застосовує найпростішу форму усереднення у двох вимірах [12]. Її відносна швидкість робить її доцільною для попереднього перегляду або застосувань у реальному часі, наприклад, у комп'ютерних іграх [20]. Однак схильність до розмиття чітко демонструє компроміс: швидкість та базова гладкість досягаються ціною втрати чіткості [15]. Білінійна інтерполяція також часто використовується в задачах обробки відео, де потрібно швидко масштабувати кадри для адаптації до різних роздільностей екрану. Наприклад, при стрімінгу відео на платформах, як YouTube, білінійна інтерполяція може застосовуватися для швидкого масштабування кадрів у реальному часі, хоча це іноді призводить до помітного розмиття динамічних сцен.

1.2.3 Бікубічна інтерполяція: аналіз та порівняння

Бікубічна інтерполяція є більш складним та якіснішим методом збільшення зображень порівняно з білінійною [21]. Вона враховує ширший контекст вихідного зображення, використовуючи для обчислення значення нового пікселя інформацію від 16 найближчих сусідніх пікселів (окіл 4×4) [22]. Замість лінійних функцій, бікубічна інтерполяція використовує поліноми третього ступеня (кубічні) для апроксимації поверхні інтенсивності зображення між вихідними пікселями [21]. Аналогічно до білінійної

інтерполяції, вага кожного з 16 пікселів залежить від його відстані до точки інтерполяції – ближчі пікселі мають більший вплив [22].

Математично бікубічна інтерполяція часто базується на використанні кубічних сплайнів або інших типів кубічних функцій з певними властивостями гладкості [21]. Існують різні варіації кубічної інтерполяції, такі як методи Катмулла-Рома (Catmull-Rom) або Мітчелла-Нетравалі (Mitchell-Netravali), які використовують різні кубічні фільтри і можуть давати дещо відмінні результати за різкістю та гладкістю [22].

Переваги бікубічної інтерполяції:

– вища чіткість. У порівнянні з білінійною інтерполяцією, бікубічна значно краще зберігає деталі та різкість країв зображення, що призводить до візуально чіткіших результатів [21];

– краща гладкість. Завдяки використанню кубічних поліномів, метод забезпечує неперервність не тільки самої інтерпольованої поверхні, але і її перших похідних (неперервність C1), що призводить до більш плавних переходів тонів та менш помітної зубчастості на кривих та діагональних лініях [22];

– оптимальний компроміс. Бікубічна інтерполяція часто розглядається як вдалий баланс між якістю результату та обчислювальною складністю, тому вона є стандартним методом у багатьох професійних графічних редакторах (наприклад, Adobe Photoshop) [21].

Недоліки бікубічної інтерполяції:

– вища обчислювальна складність. Обробка околу 4×4 та використання кубічних функцій вимагають значно більше обчислень, ніж білінійна інтерполяція, що робить її повільнішою [22];

– артефакти «ореолу». Через природу кубічних функцій, які можуть «перевищувати» або «занижувати» значення поблизу різких перепадів інтенсивності, бікубічна інтерполяція може створювати артефакти у вигляді світлих або темних контурів («ореолів») навколо контрастних країв [21].

Бікубічна інтерполяція утвердилася як «золотий стандарт» якості серед класичних методів інтерполяції [22]. Використання ширшого контексту (окіл 4×4) та складної математичної моделі (кубічні поліноми) дозволяє їй краще моделювати плавні криві та переходи в зображенні, що транслюється у візуально вищу якість – більшу чіткість та меншу зубчастість порівняно з білінійною інтерполяцією [21]. Артефакти «ореолу», характерні для бікубічної інтерполяції, можуть бути особливо проблемними при обробці зображень із високим контрастом, наприклад, у медичних знімках, де чіткі межі між тканинами є критично важливими. У таких випадках іноді застосовують додаткові методи корекції, щоб зменшити вплив ореолів і зберегти точність діагностики.

1.2.4 Біквадратична інтерполяція: аналіз та порівняння

Біквадратична інтерполяція є методом, заснованим на поліноміальній апроксимації, але використовує квадратичні функції у двох вимірах [18]. Зазвичай для обчислення значення нового пікселя вона спирається на 9 найближчих пікселів вихідного зображення (окіл 3×3) [19]. Процес інтерполяції, аналогічно до інших двовимірних методів, може виконуватися поетапно: спочатку квадратичні функції підганяються під значення пікселів у кожному з трьох рядків околу 3×3, потім ці функції обчислюються в потрібній горизонтальній координаті (X), і нарешті, фінальна квадратична функція підганяється під три отримані проміжні значення вздовж вертикальної осі (Y) для обчислення кінцевого значення [18].

Математичною основою методу є квадратичні поліноми [19]. Важливою особливістю біквадратичної інтерполяції є її менша поширеність у програмному забезпеченні та бібліотеках для обробки зображень порівняно з білінійною та бікубічною [20].

Переваги:

– проміжне положення. Використання околу 3×3 та квадратичних функцій забезпечує компроміс між простотою білінійної (2×2, лінійна) та складністю бікубічної (4×4, кубічна) інтерполяції, враховуючи більше контексту, ніж білінійна (9 пікселів проти 4) [18].

Недоліки та причини рідкісного використання:

– неперервність C0. Подібно до білінійної, біквадратична інтерполяція забезпечує лише неперервність самої функції (C0), але не її перших похідних, що обмежує гладкість переходів порівняно з бікубічною (C1) [19];

– складність vs якість. Хоча біквадратична інтерполяція складніша за білінійну (окіл 3×3 проти 2×2), вона може не давати суттєвого покращення візуальної якості, оскільки їй бракує гладкості вищого порядку, яку забезпечує бікубічна [20, 22];

– артефакти зсуву вікна. Як метод, що використовує поліноми парного ступеня, біквадратична інтерполяція схильна до появи невеликих стрибків в інтерпольованих значеннях при переході вікна 3×3 між сусідніми областями зображення [21]. Це особливо помітно при обробці зображень із регулярними візерунками, де можуть виникати ефекти муару [22];

– асиметрія ваг. Використання трьох точок для квадратичної інтерполяції є менш «природним» для сітки, ніж використання двох (лінійна) або чотирьох (кубічна) точок, що може впливати на якість [20].

Біквадратична інтерполяція займає невигідну позицію: вона збільшує обчислювальну складність порівняно з білінійною, але не забезпечує ключової переваги бікубічної – гладкості C1 [19]. Додатковим недоліком є потенційна схильність до специфічних артефактів, пов’язаних зі зсувом вікна для поліномів парного ступеня [21]. Ця комбінація факторів пояснює, чому метод не набув популярності в інструментах обробки зображень [20]. Незважаючи на обмежене використання, біквадратична інтерполяція може бути корисною в специфічних сценаріях, наприклад, для обробки зображень із середнім рівнем деталізації, де потрібен баланс між швидкістю та якістю, але без

надмірної обчислювальної складності бікубічного методу. У таких випадках вона може слугувати проміжним рішенням, хоча й потребує додаткової оптимізації для практичного застосування.

1.2.5 Огляд інших методів та порівняння

Білінійна, бікубічна та біквадратична інтерполяція належать до класичних поліноміальних методів [17]. Однак існують й інші підходи до масштабування зображень. Метод найближчого сусіда – найпростіший і найшвидший, але дає низьку якість із вираженою блоковістю [20]. Наприклад, при масштабуванні текстур у комп'ютерних іграх цей метод призводить до помітної зубчастості, що погіршує візуальний досвід [22] Методи на основі сплайнів чи функції sinc, як інтерполяція Ланцоша, забезпечують вищу якість, зберігаючи деталі, але потребують більше обчислень [17]. При обробці зображень із регулярними текстурами такі методи можуть викликати ефект муару [23, 24, 25]. Алгоритми, орієнтовані на збереження країв, такі як NEDI чи DCCI, зменшують зубчастість на контурах, що корисно для зображень із чіткими межами [26, 27]. Сучасні методи супер-роздільності на основі нейронних мереж, наприклад, SRGAN, можуть генерувати деталі, яких немає у вихідному зображенні, але вони вимагають значних ресурсів і даних для навчання [18].

Якість методів оцінюється за допомогою метрик, таких як PSNR (пікове відношення сигналу до шуму) та SSIM (індекс структурної подібності), а також візуально [28, 29, 30]. Класичні методи, попри появу нових технологій, залишаються важливою основою, адже вони простіші й часто слугують відправною точкою для оцінки складніших алгоритмів. Комбінація класичних методів із нейронними мережами, наприклад, використання бікубічної інтерполяції перед застосуванням нейронних моделей, може покращити

результати супер-роздільності, зменшуючи обчислювальне навантаження [31].

Такі комбіновані підходи часто застосовуються в задачах реального часу, наприклад, у комп'ютерних іграх, де потрібна швидка обробка текстур із мінімальними артефактами, такими як розмиття чи зубчастість [32]. У задачах обробки відео подібні методи дозволяють адаптувати кадри до різних роздільностей без значних втрат якості, хоча динамічні сцени можуть вимагати додаткових алгоритмів для зменшення розмиття чи спотворень [33]. У медичній візуалізації, де важливі чіткі межі між тканинами, комбінація класичних методів із нейронними мережами допомагає зменшувати артефакти, такі як ореоли, зберігаючи точність діагностики, що є критично важливим для таких застосувань [34]. Для зображень із середнім рівнем деталізації, наприклад, у фотографіях природи чи портретах, ці методи забезпечують баланс між швидкістю та якістю, хоча часто потребують додаткової оптимізації для уникнення ефектів муару чи надмірного згладжування [35]. Деякі методи дозволяють досягти кращої якості за рахунок попередньої обробки зображень для видалення шуму, що особливо корисно для зображень із низькою якістю, таких як старі скановані фотографії чи знімки з низькою роздільністю [36]. Такі комбінації виявилися ефективними в задачах обробки супутникових знімків, де потрібна висока деталізація для аналізу ландшафтів чи об'єктів [37].

1.3 Постановка задачі

Таким чином, розробка вебзастосунку з можливістю збільшення зображень вище описаними методами інтерполяції є актуальною темою. Об'єктом роботи є растрові зображення у форматі PNG, що підлягають масштабуванню з використанням методів інтерполяції.

Метою роботи є розробка програмного застосунку для масштабування растрових зображень із використанням білінійної, бікубічної та біквадратичної інтерполяції, а також порівняльний аналіз їхньої ефективності за кількісними метриками та візуальними характеристиками на різних типах зображень.

Для досягнення мети необхідно вирішити такі завдання:

– проаналізувати існуючі методи інтерполяції зображень та їхні обмеження з точки зору якості й обчислювальної складності;

– розробити алгоритми білінійної, бікубічної та біквадратичної інтерполяції, оптимізувавши їх для локальних систем із обмеженими ресурсами;

– реалізувати програмний застосунок для масштабування зображень із підтримкою асинхронної обробки, обчислення метрик якості (PSNR, SSIM, MSE, Gradient Difference) і візуалізації результатів;

– провести експериментальний аналіз ефективності методів інтерполяції на наборі з 10 зображень, що включають пейзажі, текст, їжу, та інші види зображень, із масштабуванням на різні коефіцієнти;

– оцінити візуальні артефакти (розмиття, ореоли, зубчастість) для кожного методу та зробити висновки щодо їхньої застосовності в різних сценаріях.

2 АРХІТЕКТУРА СИСТЕМИ ЗБІЛЬШЕННЯ РАСТРОВИХ ЗОБРАЖЕНЬ

2.1 Загальний опис системи

Система, розроблена в рамках цієї роботи, є інноваційним інструментом для масштабування растрових зображень, що дозволяє користувачам досліджувати та порівнювати результати різних методів інтерполяції: білінійної, бікубічної та біквадратичної. Її основне призначення – забезпечити зручний і ефективний спосіб збільшення зображень із збереженням їхньої якості, а також надати детальний аналіз результатів через кількісні метрики та візуальні порівняння.

Основні функції системи включають:

– завантаження зображень у формат PNG;

– вибір коефіцієнта масштабування, що дозволяє користувачу гнучко налаштовувати ступінь збільшення (наприклад, 1.4x, 2x, 4x);

– асинхронна обробка зображень, яка забезпечує швидке виконання ресурсоємних обчислень;

– обчислення метрик якості, таких як пікове відношення сигнал/шум (PSNR), індекс структурної подібності (SSIM), середньоквадратична помилка (MSE) та різниця градієнтів (Gradient Difference);

– візуалізація результатів через інтерактивний інтерфейс, що включає збільшені зображення, таблиці метрик і графіки для порівняння.

Користувач взаємодіє з системою через сучасний вебінтерфейс, який поєднує простоту використання з естетичним дизайном. Інтерфейс дозволяє легко завантажувати зображення, налаштовувати параметри та отримувати результати в реальному часі. Завдяки асинхронній обробці користувач може відстежувати прогрес виконання завдання, що робить взаємодію з системою динамічною та комфортною [38].

2.2 Архітектура фронтенду

2.2.1 Технологічний стек

Фронтенд системи є ключовим елементом, що забезпечує зручну взаємодію користувача з усіма функціями програми, від завантаження зображень до аналізу результатів масштабування. Для його реалізації обрано бібліотеку React, яка є стандартом для сучасних вебдодатків завдяки своїй модульній архітектурі та ефективному управлінню інтерфейсом [39]. React дозволяє розбити інтерфейс на незалежні компоненти, кожен із яких відповідає за конкретну функцію, наприклад, завантаження файлу чи відображення графіків. Це забезпечує чітку організацію коду, полегшує тестування та дозволяє легко додавати нові функції без значних змін у структурі.

Стилізація інтерфейсу виконана за допомогою Tailwind CSS – утилітарного фреймворку, який дозволяє швидко створювати адаптивний і естетичний дизайн [39]. Tailwind використовує набір класів для стилізації безпосередньо в розмітці, що зменшує обсяг CSS-коду та забезпечує консистентність дизайну. Завдяки цьому інтерфейс виглядає сучасно та коректно відображається на пристроях із різними розмірами екранів, від великих моніторів до смартфонів.

Для візуалізації метрик якості, таких як пікове відношення сигнал/шум (PSNR), індекс структурної подібності (SSIM), середньоквадратична помилка (MSE) та різниця градієнтів (Gradient Difference), використовується бібліотека Chart.js. Вона дозволяє створювати інтерактивні гістограми, які допомагають користувачу швидко оцінити ефективність методів інтерполяції. Графіки підтримують спливаючі підказки з точними значеннями метрик, що полегшує аналіз результатів

Компонентний підхід у React забезпечує низку переваг:

– модульніть. Кожен компонент є ізольованим, що дозволяє повторно використовувати його в різних частинах програми;

– масштабованість. Нові функції чи компоненти можна додавати без зміни існуючої архітектури;

– легкість підтримки. Завдяки чіткому розподілу відповідальностей зміни в одному компоненті не впливають на інші;

– ефективність. React використовує віртуальний DOM, який мінімізує прямі маніпуляції з реальним DOM, що підвищує швидкість оновлення інтерфейсу [33].

Використання Vite як інструменту для збирання проєкту забезпечує швидке завантаження сторінок і зручний процес розробки. Поєднання React, Tailwind CSS і Chart.js створює потужну основу для фронтенду, яка відповідає сучасним стандартам веброзробки та забезпечує високу якість користувацького досвіду.

2.2.2 UML-діаграма фронтенду

Для опису динамічної взаємодії компонентів під час обробки зображення використовується UML-діаграма послідовності (рис. 2.1). Ця діаграма є оптимальним вибором, оскільки чітко ілюструє послідовність викликів між компонентами фронтенду та бекендом, підкреслюючи асинхронний характер обробки.

На рисунку 2.1 зображено UML-діаграму послідовності, яка описує взаємодію між користувачем (User), компонентами фронтенду (App, ImageUploader, Controls, Loader, ResultCard, MetricsTable, MetricChart, ImageModal, ErrorMessage) та бекендом (FastAPI, WebSocket). Процес охоплює основні етапи: завантаження зображення, налаштування масштабування, запуск завдання, відстеження прогресу, відображення результатів, перегляд результатів і обробку помилок.

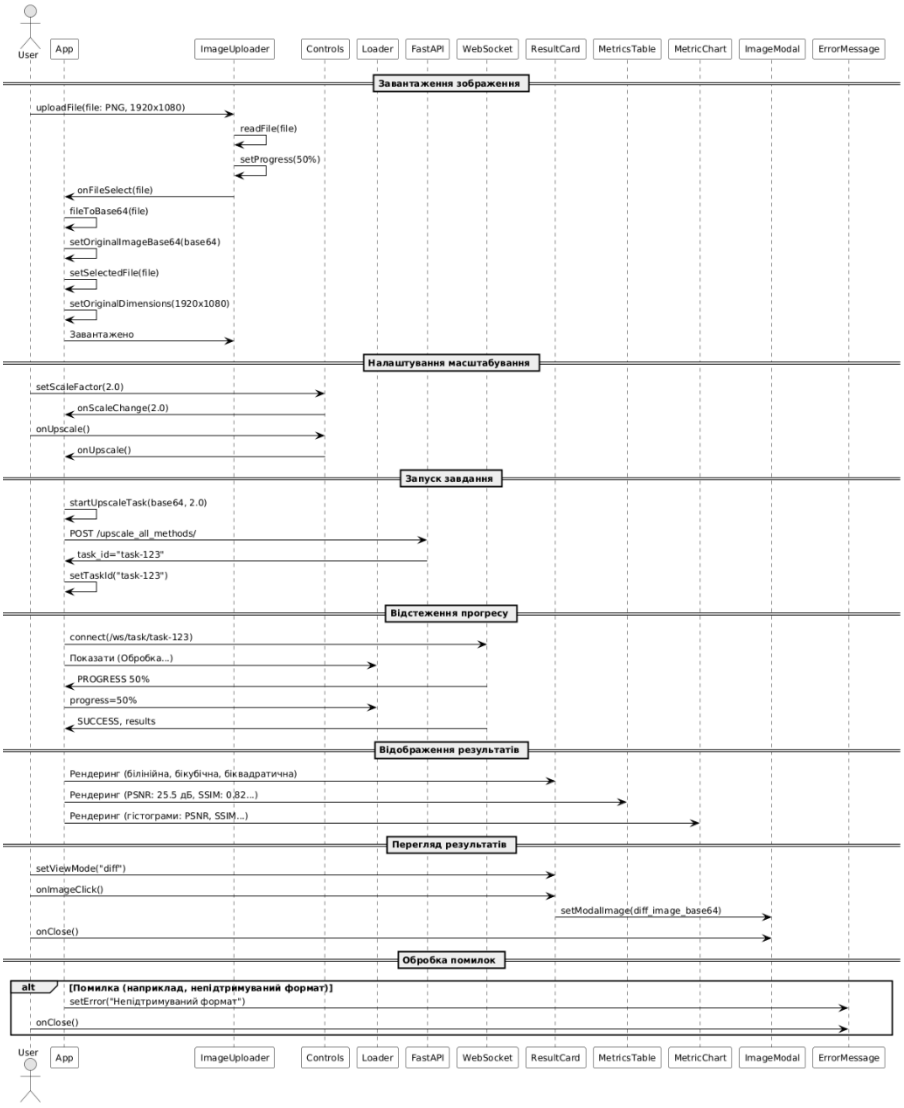


Рисунок 2.1 – UML-діаграма компонентів для фронтенд-архітектури системи масштабування зображень

Користувач починає із завантаження зображення через ImageUploader, який перевіряє файл (наприклад, PNG 1920×1080) і передає його до App через onFileSelect. У Controls користувач обирає коефіцієнт масштабування (наприклад, 2.0) і запускає обробку через onUpscale, що ініціалізує POST-запит до FastAPI для створення завдання (task_id=«task-123»). App підключається до WebSocket для відстеження прогресу, відображаючи Loader із повідомленням «Обробка...» і прогресом (наприклад, 50%). Після завершення обробки App рендерить результати через ResultCard (збільшені зображення, карти різниць, гістограми), MetricsTable (метрики PSNR, SSIM тощо) і MetricChart

(гістограми). Користувач може переглянути деталі через ImageModal або закрити помилки через ErrorMessage, якщо вони виникають (наприклад, «Непідтримуваний формат»).

2.2.3 Опис компонентів

Фронтенд системи складається з кількох ключових компонентів, кожен із яких виконує визначену функцію та взаємодіє з іншими відповідно до діаграми послідовності (рис 2.1):

– App – Головний компонент, який координує роботу всіх інших компонентів. Він управляє станом (наприклад, originalImageBase64, scaleFactor, taskId) і викликає запити до бекенду через FastAPI та WebSocket. У діаграмі послідовності App є центральним елементом, що обробляє події від ImageUploader і Controls, відстежує прогрес і передає результати до компонентів відображення;

– ImageUploader відповідає за завантаження зображень через drag-and-drop або діалог вибору файлу. Перевіряє формат (PNG) і розмір (макс. 5 МБ), конвертує файл у base64 і передає його до App через onFileSelect. У діаграмі послідовності цей компонент є початковою точкою взаємодії користувача із системою;

– Controls дозволяє користувачу налаштувати коефіцієнт масштабування (наприклад, 2.0) через числове поле і запустити обробку кнопкою «Збільшити». Передає події onScaleChange і onUpscale до App. У діаграмі послідовності Controls відповідає за етап налаштування масштабування;

– Loader відображає прогрес обробки (наприклад, «Обробка 50%...») із прогрес-баром під час асинхронної роботи. Отримує дані від App через WebSocket. У діаграмі послідовності Loader активується на етапі відстеження прогресу;

– ResultCard відображає результати масштабування для кожного методу інтерполяції (білінійна, бікубічна, біквадратична). Містить збільшене зображення, карту різниць, гістограму помилок і метрики (PSNR, SSIM, MSE, Gradient Difference). Дозволяє перемикати режими перегляду і завантажувати результати. У діаграмі послідовності ResultCard рендериться після завершення обробки;

– MetricsTable формує таблицю для порівняння метрик якості (PSNR, SSIM, MSE, Gradient Difference, час обробки) усіх методів. Підтримує сортування і виділення найкращих значень. У діаграмі послідовності відображається разом із ResultCard на етапі відображення результатів;

– MetricChart створює інтерактивні гістограми для візуалізації метрик (PSNR, SSIM, MSE, Gradient Difference, час обробки) із спливаючими підказками. У діаграмі послідовності рендериться разом із ResultCard і MetricsTable;

– ImageModal відображає модальне вікно для детального перегляду зображень (збільшених, карт різниць, гістограм). Активується через onImageClick у ResultCard. У діаграмі послідовності з’являється на етапі перегляду результатів;

– ErrorMessage показує повідомлення про помилки (наприклад, «Непідтримуваний формат») із можливістю закриття. У діаграмі послідовності відображається в умовному блоці обробки помилок.

2.3 Архітектура бекенду

2.3.1 Технологічний стек

Бекенд системи є її «серцем», відповідаючи за обробку зображень, виконання обчислень і повернення результатів клієнту. Для реалізації бекенду обрано фреймворк FastAPI, який вирізняється високою продуктивністю та підтримкою асинхронного програмування [38]. FastAPI дозволяє створювати

ефективні API-ендпоінти, які швидко обробляють запити, що критично важливо для роботи з ресурсоємними завданнями масштабування.

Для виконання самих обчислень використовується Celery – система для асинхронної обробки завдань [38]. Celery дозволяє виносити тривалі операції, такі як інтерполяція зображень, у фонові процеси, звільняючи основний сервер для обробки нових запитів. Це забезпечує високу швидкість роботи навіть при одночасному обслуговуванні кількох користувачів та допомагає оброблювати великі зображення [41].

Redis відіграє подвійну роль: він виступає брокером повідомлень для Celery, координуючи виконання завдань, і сховищем для збереження стану завдань та результатів. Redis вибрано через його швидкість і простоту роботи з даними в пам’яті, що ідеально підходить для асинхронних систем [41].

Для обробки зображень використовується бібліотека OpenCV, яка надає реалізації методів інтерполяції (INTER_LINEAR для білінійної, INTER_CUBIC для бікубічної). Метрики якості, такі як PSNR і SSIM, обчислюються за допомогою бібліотеки Scikit-image, яка забезпечує точні та оптимізовані алгоритми для аналізу зображень [41].

Такий технологічний стек дозволяє системі бути швидкою, масштабованою та надійною. FastAPI забезпечує ефективну взаємодію з клієнтом, Celery і Redis дозволяють обробляти ресурсоємні завдання без затримок, а OpenCV і Scikit-image гарантують високу якість обчислень [38].

2.3.2 UML-діаграма взаємодії

Діаграма (рис. 2.2) відображає процес обробки зображення: фронтенд надсилає зображення і масштаб через POST-запит до FastAPI, який перевіряє дані та створює завдання в Celery. Redis зберігає стан завдання (INITIATED). Фронтенд підключається до WebSocket за task_id, отримуючи оновлення прогресу (наприклад, 50%). Celery викликає ImageProcessor для декодування,

зменшення, масштабування (всі методи інтерполяції), обчислення метрик і генерації гістограм. Результати зберігаються в Redis, а стан оновлюється (PROGRESS, SUCCESS). FastAPI надсилає результати через WebSocket. Запит GET /average_times/ повертає статистику (середні часи, MSE), обчислену ImageProcessor.

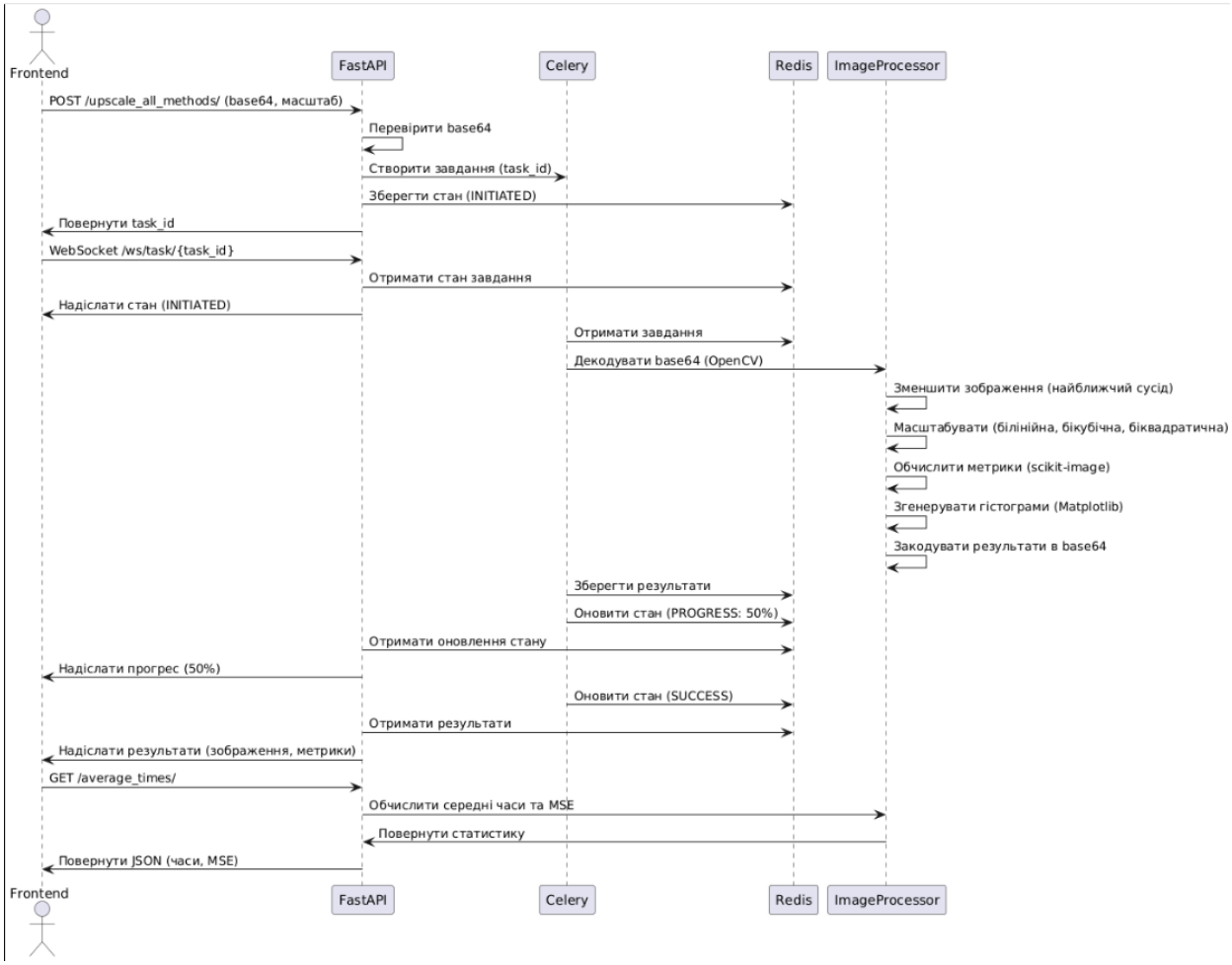


Рисунок 2.2 – UML-діаграма компонентів для бекенд-архітектури системи масштабування зображень

2.3.3 Опис API та WebSocket

Бекенд системи надає кілька ключових точок взаємодії з клієнтом через HTTP API та WebSocket:

POST /upscale_all_methods/. Цей ендпоінт є основним для запуску обробки зображень. Клієнт надсилає зображення у форматі base64 разом із

бажаним коефіцієнтом масштабування. FastAPI перевіряє коректність даних (наприклад, чи є base64 валідним зображенням), після чого створює завдання в Celery для обробки зображення всіма методами інтерполяції. Ендпоінт повертає унікальний ідентифікатор завдання (task_id), який використовується для подальшого відстеження прогресу. У разі помилки, наприклад, якщо зображення пошкоджене або має некоректний формат, користувач отримує детальне повідомлення про проблему, що полегшує діагностику.

GET /average_times/. Цей ендпоінт повертає статистичні дані про середні часи обробки та середню MSE для кожного методу інтерполяції. Дані обчислюються на основі логів попередніх обробок, що зберігаються в текстових файлах. Скрипт calculate_avg_times.py аналізує ці логи, обмежуючи вибірку першими 10 зображеннями для кожного методу, щоб забезпечити репрезентативність без надмірного навантаження. Результати повертаються у форматі JSON і відображаються в інтерфейсі, дозволяючи користувачу оцінити продуктивність і якість методів перед запуском обробки.

WebSocket /ws/task/{task_id}. WebSocket-з'єднання є ключовим для забезпечення інтерактивності системи. Після запуску завдання клієнт підключається до WebSocket за отриманим task_id і отримує повідомлення про поточний стан обробки. Система надсилає оновлення в реальному часі, включаючи відсоток виконання (наприклад, 50% прогресу), статус завдання (INITIATED, PROGRESS, SUCCESS, FAILURE) і, після завершення, повні результати обробки, такі як збільшені зображення та метрики. Це дозволяє відображати користувачу індикатор прогресу та уникати необхідності періодичних HTTP-запитів для перевірки стану. WebSocket також обробляє можливі помилки, повідомляючи клієнта про збої в обробці. У разі недоступності Redis (наприклад, через збій сервера) FastAPI повертає помилку 503 Service Unavailable із повідомленням «Не вдалося підключитися до брокера завдань», а клієнт отримує повідомлення через WebSocket про необхідність повторного запуску завдання.

2.4 Логіка обробки зображень

2.4.1 Методи інтерполяції

Система підтримує три методи інтерполяції для масштабування зображень: білінійну, бікубічну та біквадратичну. Кожен метод має унікальні характеристики, які впливають на якість і швидкість обробки. Нижче наведено їхній опис, принципи роботи та математичні основи.

Білінійна інтерполяція є одним із найпростіших і найшвидших методів масштабування. Вона використовує значення чотирьох сусідніх пікселів (2×2) для обчислення значення нового пікселя в збільшеному зображенні. Метод виконує лінійну інтерполяцію спочатку в горизонтальному напрямку, а потім у вертикальному.

Для пікселя з координатами (x, y), розташованого між пікселями $I(x_0, y_0)$, $I(x_1, y_0)$, $I(x_0, y_1)$, $I(x_1, y_1)$, значення обчислюється як:

$$I(x,y) = (1-\alpha)(1-\beta)I(x_0,y_0) + \alpha(1-\beta)I(x_1,y_0) + (1-\alpha)\beta I(x_0,y_1) + \alpha\beta I(x_1,y_1), \quad (2.1)$$

де $\alpha = x - x_0$ – дробова відстань по осі x;

$\beta = y - y_0$ – дробова відстань по осі y.

Цей метод ефективний для реального часу, наприклад, у комп'ютерних іграх для згладжування текстур, але призводить до розмиття, особливо при масштабуванні зображень із різкими контурами, як текст чи логотипи.

Метод реалізований через функцію cv2.resize бібліотеки OpenCV із параметром interpolation=cv2.INTER_LINEAR. OpenCV забезпечує високу продуктивність завдяки оптимізації на рівні C++, що робить метод ідеальним для реального часу. Функція bilinear_interpolation приймає зображення (масив NumPy) і коефіцієнт масштабування, повертаючи збільшене зображення. Наприклад, масштабування зображення 1920×1080 до 3840×2160 пікселів обробляється за секунди на сучасних ПК.

Бікубічна інтерполяція є складнішим методом, який використовує 16 сусідніх пікселів (4×4) для обчислення значення нового пікселя. Вона застосовує кубічні сплайни, що забезпечують більш плавні переходи та кращу збереженість деталей порівняно з білінійною інтерполяцією.

Для пікселя (x, y) значення обчислюється як:

$$I(x,y) = \sum_{i=-1}^{2} \sum_{j=-1}^{2} I(x_i, y_i)\, \omega(\alpha - \mathrm{i})\omega(\beta - \mathrm{i}), \tag{2.2}$$

де $\alpha = x\ -\ x_0, \beta = y\ -\ y_1$ – дробові відстані;

$\omega(t)$ – вагова функція кубічного сплайну:

$$\omega(t) = \begin{cases} \frac{3}{2}|t|^3 - \frac{5}{2}|t|^2 + 1, & 0 \le |t| < 1 \\ -\frac{1}{2}|t|^3 + \frac{5}{2}|t|^2 - 4|t| + 2, & 1 < |t| < 2. \\ 0, & |t| \ge 2 \end{cases} \tag{2.3}$$

Цей метод ідеально підходить для поліграфії чи медичної візуалізації, але його обчислювальна складність обмежує використання в реальному часі, наприклад, у іграх.

Метод реалізований через cv2.resize із параметром interpolation=cv2.INTER_CUBIC. OpenCV оптимізує обчислення, але обробка 16 пікселів робить метод повільнішим за білінійний. Функція bicubic_interpolation масштабує зображення, наприклад, із 1000×1000 до 4000×4000 пікселів, що займає більше часу, ніж білінійна, але забезпечує чіткість.

Біквадратична інтерполяція, розроблена в рамках цієї роботи, є компромісом між білінійною та бікубічною інтерполяцією за якістю та швидкістю. Вона використовує 9 сусідніх пікселів (3×3) і базується на двовимірних сплайнах другого порядку, що забезпечують плавні переходи та помірну обчислювальну складність.

Для пікселя (x,y) значення обчислюється як:

$$I(x,y) = \sum_{i=0}^{2} \sum_{j=0}^{2} c_{ij} B_i(x - x_0) B_j(y - y_0), \quad (2.4)$$

де $B_i(t)$, $B_j(t)$ – базисні квадратичні функції (B-сплайни другого порядку), c_{ij} – коефіцієнти, отримані з значень пікселів у околі 3×3.

Базисна функція B(t) для квадратичного сплайну визначається як:

$$B(t) = \begin{cases} \frac{1}{2}t^2 - t + \frac{1}{2}, & 0 \le |t| < 1 \\ -\frac{1}{2}t^2 + t - \frac{1}{2}, & 1 \le |t| < 2 \\ 0 & |t| \ge 2 \end{cases} . \quad (2.5)$$

Цей метод оптимізовано для локальних систем, забезпечуючи високу якість для пейзажів чи портретів із помірними ресурсами, що робить його привабливим для вебдодатків і легких редакторів зображень.

Метод (рис. 2.4) реалізований через функцію biquadratic_interpolation яка обробляє кожен канал RGB окремо, використовуючи RectBivariateSpline із SciPy з параметрами kx=2, ky=2. Для одноканальних зображень (градації сірого) система автоматично адаптує обробку, застосовуючи інтерполяцію до єдиного каналу, що зменшує час обчислень.

```
try:
    target_height, target_width = validate_image(image, scale_factor, f"Біквадратична інтерполяція {'(зменшення)' if scale_factor < 1 else '(збільшення)'}")

    height, width, channels = image.shape
    x = np.arange(width)
    y = np.arange(height)
    x_new = np.linspace(0, width - 1, target_width)
    y_new = np.linspace(0, height - 1, target_height)

    # Створюємо масив для масштабованого зображення
    scaled = np.zeros((target_height, target_width, channels), dtype=np.float32)

    # Інтерполюємо кожен канал окремо
    for channel in range(channels):
        # Використовуємо біквадратичний сплайн (kx=2, ky=2)
        spline = RectBivariateSpline(y, x, image[:, :, channel], kx=2, ky=2)
        scaled[:, :, channel] = spline(y_new, x_new)

    # Обрізаємо значення до діапазону [0, 255] і конвертуємо в uint8
    scaled = np.clip(scaled, 0, 255).astype(np.uint8)
    return scaled
except Exception as e:
    logger.error(f"Помилка в біквадратичній інтерполяції: {str(e)}")
    raise
```

Рисунок 2.3 – Реалізація біквадратичної інтерполяції в коді

2.4.2 Процес обробки

Логіка обробки зображень є основою функціональності системи й реалізується в асинхронних завданнях Celery. Процес складається з кількох етапів, кожен із яких ретельно оптимізований для забезпечення якості та швидкості:

– завантажене зображення, передане у форматі base64, декодується в масив NumPy за допомогою бібліотеки OpenCV. Це дозволяє отримати растрове зображення у форматі RGB, придатному для подальшої обробки. На цьому етапі перевіряється цілісність даних, щоб уникнути роботи з пошкодженими файлами;

– для того щоб еталоном у нас виступало оригінальне зображення, проводиться обратна інтерполяція – зображення зменшується методом найближчого сусіда. Коефіцієнт зменшення обернений до заданого коефіцієнта масштабування: наприклад, при масштабуванні 2x зображення зменшується вдвічі по ширині та висоті після цього проводиться пряма інтерполяція;

– зменшене зображення масштабується назад до оригінального розміру за допомогою трьох методів інтерполяції: білінійної, бікубічної та біквадратичної. Кожен метод реалізується через відповідні функції OpenCV, які забезпечують точне виконання алгоритмів. Наприклад, білінійна інтерполяція враховує 4 сусідні пікселі, бікубічна – 16, а біквадратична – 9;

– після масштабування для кожного методу обчислюються метрики якості шляхом порівняння отриманого зображення з оригіналом. Використовуються PSNR, SSIM, MSE і Gradient Difference, які дозволяють кількісно оцінити точність відтворення деталей, структурну подібність і чіткість країв;

– усі результати – збільшені зображення, зображення різниці, гістограми помилок і метрики – кодуються в base64 для передачі клієнту. Результати тимчасово зберігаються в Redis, щоб забезпечити швидкий доступ

через WebSocket. Гістограми помилок генеруються за допомогою Matplotlib, що дозволяє візуально оцінити розподіл помилок для кожного методу.

Цей процес є повністю автоматизованим і виконується у фоновому режимі, що дозволяє користувачу продовжувати взаємодію з інтерфейсом, не чекаючи завершення обчислень.

2.4.3 Формули метрик

Оцінка якості масштабування зображень є критично важливою для порівняння ефективності методів інтерполяції (білінійної, бікубічної та біквадратичної), оскільки вона дозволяє кількісно визначити, наскільки отримане зображення схоже на оригінал. У системі використовуються чотири метрики: PSNR (Peak Signal-to-Noise Ratio), SSIM (Structural Similarity Index), MSE (Mean Squared Error) і Gradient Difference. Ці метрики враховують різні аспекти якості – від загального рівня шуму до структурної подібності та чіткості країв, що робить їх універсальними для застосунків у комп’ютерних іграх, вебдизайні, поліграфії та медичній візуалізації. Кожна метрика має математичну основу, що забезпечує об’єктивність оцінки, а їхні результати використовуються для порівняння методів інтерполяції в системі.

Метрики якості відіграють ключову роль у виборі оптимального методу залежно від задачі. Наприклад, у комп’ютерних іграх, де рендеринг текстур у реальному часі вимагає швидкості, нижчий PSNR може бути прийнятним, якщо візуальні артефакти не критичні для сприйняття (наприклад, у фонових елементах, як трава). У медичній візуалізації, навпаки, висока структурна подібність (SSIM) є пріоритетом, щоб зберегти дрібні деталі на рентгенівських знімках. Система обчислює ці метрики автоматично, порівнюючи масштабоване зображення з оригіналом, що дозволяє користувачу об’єктивно оцінити методи. Біквадратна інтерполяція, розроблена в цій роботі, демонструє компроміс на специфічних зображеннях між якістю (високий

SSIM) і швидкістю (нижча обчислювальна складність порівняно з бікубічною), що ідеально для локальних систем.

Історично метрики, як PSNR і MSE, з’явилися в 1960-х роках для оцінки сигналів, а SSIM був запропонований у 2004 році для кращого врахування сприйняття людським оком [33]. Gradient Difference додає аналіз чіткості країв, що важливо для оцінки контурів у зображеннях.

Нижче у таблиці 2.4 наведений короткий опис метрик якості.

Таблиця 2.4 – Огляд метрик якості

Метрика	**Аспект оцінки**	**Діапазон значень**	**Застосування**
PSNR	Рівень шуму	0–∞ дБ (вище – краще)	Ігри, вебдизайн
SSIM	Структурна подібність (контраст, яскравість,структура)	0–1 (1 – ідеальна)	Медична візуалізація, поліграфія
MSE	Середньоквадратична помилка	0–∞ (нижче – краще)	Загальна оцінка
Gradient Difference	Чіткість країв	0–∞ (нижче – краще)	Аналіз контурів

PSNR(Peak Signal-to-Noise Ratio) є однією з найпоширеніших метрик для оцінки якості зображень, вимірюючи відношення максимального сигналу до шуму, спричиненого помилками масштабування. Вона базується на MSE і виражається в децибелах (дБ), що дозволяє легко інтерпретувати результати [39].

MSE(Mean Squared Error) вимірює середньоквадратичну помилку між оригінальним I і масштабованим K зображенням:

Формула для MSE. Спочатку обчислюється MSE між оригінальним зображенням I і масштабованим K:

$$MSE = \frac{1}{HW}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1}[I(i,j) - K(i,j)^2], \quad (2.6)$$

де, H і W – висота і ширина зображення, $I(i,j)$ і $K(i,j)$ – значення пікселів, потім PSNR:

$$PSNR = 10\log_{10}(\frac{MAX^2}{MSE}). \quad (2.7)$$

де, MAX – максимальне значення пікселя (255 для 8-бітних зображень).

Наприклад, PSNR 30 дБ вказує на хорошу якість, тоді як значення нижче 20 дБ свідчить про значні спотворення.

MSE використовується для загальної оцінки помилок, наприклад, у вебдизайні для оцінки якості зображень. У переваги цього виду оцінювання, можна внести низку складність обчислення, але існує і обмеження, що MSE враховує людське сприйняття.

PSNR широко використовується в комп'ютерних іграх для оцінки якості текстур після масштабування. Наприклад, білінійна інтерполяція може дати PSNR приблизно 28 дБ для текстур трави, що прийнятно для швидкості, але біквадратна досягає близ 30 дБ, покращуючи сприйняття. У вебдизайні PSNR допомагає оцінити якість зображень у галереях.

Перевагою PSNR є простота її обчислення, оскільки вона базується на середньоквадратичній похибці (MSE), що робить її зручною для швидкого аналізу. Крім того, інтерпретація значень PSNR є інтуїтивно зрозумілою: вищі показники вказують на менший рівень спотворень і кращу якість зображення. Проте метрика має певні обмеження, які необхідно враховувати. Зокрема, PSNR не бере до уваги особливості людського сприйняття, такі як чутливість до структурних змін, що робить її менш релевантною для оцінки візуальної якості порівняно з іншими метриками. Також PSNR виявляє високу чутливість до малих помилок у гладких областях зображення, що може призводити до некоректної оцінки якості в таких випадках.

SSIM (Structural Similarity Index) є метрикою, що оцінює структурну подібність між оригінальним і масштабованим зображенням, враховуючи сприйняття людським оком. Вона аналізує яскравість, контраст і структуру, що робить її більш релевантною для оцінки якості, ніж PSNR [39].

Формула SSIM між двома зображеннями *x* і *y* обчислюється для кожного пікселя з використанням ковзного вікна:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y+C_1)(2\sigma_{xy}+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)}, \quad (2.8)$$

де, μ_x, μ_y – середні значення яскравості в околі;

σ_x, σ_y – стандартні відхилення;

σ_{xy} – коваріація;

C_1, C_2 – константи для стабільності (залежні від діапазону пікселів).

Результат усереднюється по всьому зображенню, даючи значення від 0 до 1 (1 – ідеальна подібність).

SSIM ідеально підходить для медичної візуалізації, де збереження структури (наприклад, контурів пухлин на МРТ) є критичним. Наприклад, бікубічна інтерполяція може досягти приблизно 0.95, тоді як білінійна – лише приблизно 0.85, що помітно на дрібних деталях. У поліграфії SSIM допомагає оцінити якість друкованих зображень.

Переваги:

– сприйняття. Враховує особливості людського зору;

– чутливість. Ефективна для оцінки структурних змін.

Обмеження:

– складність. Обчислювально складніша за PSNR;

– локальність. Залежить від розміру ковзного вікна.

MSE і Gradient Difference доповнюють PSNR і SSIM, надаючи додаткові аспекти оцінки якості масштабування.

$$MSE = \frac{1}{HW}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1}[I(i,j) - K(i,j)^2], \quad (2.9)$$

де, H і W – висота і ширина зображення, $I(i,j)$ і $K(i,j)$ – значення пікселів.

Нижче значення вказує на меншу помилку. Наприклад, білінійна інтерполяція може дати MSE приблизно 500 для пейзажного зображення, тоді як бікубічна – приблизно 300, що свідчить про кращу якість.

Gradient Difference. Ця метрика оцінює чіткість країв, обчислюючи різницю градієнтів між оригіналом і масштабованим зображенням. Градієнт для пікселя обчислюється через оператори Собеля:

$$G_x = Sobel_x(I), \quad (2.10)$$

$$G_y = Sobel_y(I), \quad (2.11)$$

$$Gradient\ Difference = \frac{1}{HW}\sum_{i=0}^{H-1}\sum_{j=0}^{W-1}\sqrt{(G_x(I) - G_x(k))^2 + (G_y(I) - G_y(k))^2}. \quad (2.12)$$

Оператор Собеля є класичним інструментом у цифровій обробці зображень, запропонованим у 1968 році Ірвіном Собелем і Гері Фельдманом [34]. Він використовує згортку з двома ядрами 3x3 для обчислення градієнтів у горизонтальному (G_x) і вертикальному (G_y) напрямках:

$$Sobel_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad (2.13)$$

$$Sobel_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \quad (2.14)$$

Ці ядра підкреслюють зміни інтенсивності пікселів, що дозволяє ефективно виявляти краї [34]. У комп'ютерних іграх оператор Собеля застосовується для виділення контурів персонажів перед масштабуванням, що допомагає оцінити, як методи інтерполяції зберігають чіткість. Нижче значення Gradient Difference вказує на кращу збереженість країв, що важливо для тексту чи логотипів.

Gradient Difference корисна в іграх для оцінки чіткості текстур персонажів, де бікубічна інтерполяція забезпечує кращі результати, ніж білінійна. До переваг можна віднести ефективність для контурів, але цей метод вимагає додаткових обчислень.

2.5 Аналіз продуктивності

2.5.1 Логіка аналізу

Аналіз продуктивності системи здійснюється за допомогою скрипта, який обробляє логи обробки зображень, збережені в текстових файлах у директорії static/results. Логи містять інформацію про час виконання та MSE для кожного методу інтерполяції. Скрипт обчислює середні значення цих показників, обмежуючи аналіз першими 10 зображеннями для кожного методу. Таке обмеження дозволяє отримати репрезентативну статистику без надмірного навантаження на систему, що особливо важливо для локального виконання.

Процес аналізу включає зчитування логів, групування даних за методами інтерполяції та обчислення середніх значень часу обробки та MSE. Результати форматуються у вигляді JSON-об'єкта, який повертається через API-ендпоінт GET /average_times/. Ці дані відображаються в інтерфейсі у вигляді таблиці, що дозволяє користувачу оцінити, який метод є найшвидшим або забезпечує найменшу помилку. Обмеження в 10 зображень обґрунтовано компромісом між точністю статистики та швидкістю аналізу, оскільки обробка великої кількості зображень може займати значний час.

2.5.2 Приклад результатів

Нижче на рисунку 2.4 наведено приклад результатів аналізу продуктивності.

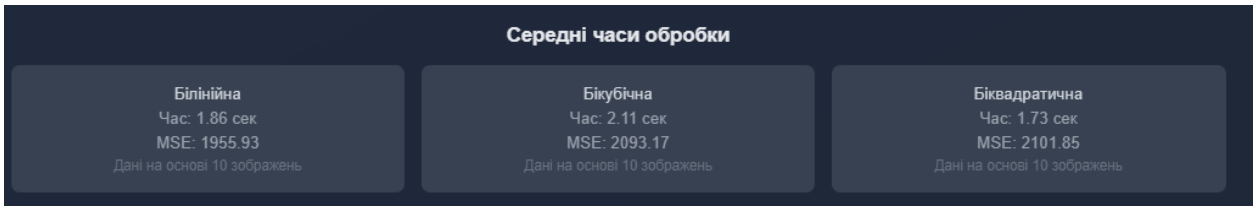


Рисунок 2.4 – Приклад результатів аналізу продуктивності

Проведений детальний аналіз отриманих даних, зібраних на основі обробки десяти окремих зображень, відкриває широкі можливості для оцінки ключових характеристик, таких як час виконання алгоритму та середньоквадратична помилка, з метою визначення загальних тенденцій, закономірностей і напрямків ефективного застосування конкретного методу інтерполяції в різних обчислювальних задачах. Запропонована реалізація біквадратичної інтерполяції, відповідно до результатів проведеного дослідження, може бути розглянута в окремих випадках як компромісний варіант, що забезпечує розумний баланс між продуктивністю, точністю та обчислювальною ефективністю, особливо для систем із обмеженими апаратними ресурсами або для сценаріїв, де найвищим пріоритетом є досягнення мінімального часу виконання обробки даних при збереженні прийнятного рівня якості результатів.

3 КОМП'ЮТЕРНА МОДЕЛЬ МАСШТАБУВАННЯ ЗОБРАЖЕНЬ

3.1 Обґрунтування вибору середовища програмної реалізації

Вибір середовища реалізації є ключовим етапом розробки, оскільки він визначає продуктивність, зручність розробки та ефективність системи масштабування зображень. Для реалізації обрано мову програмування Python із набором бібліотек: OpenCV, NumPy, SciPy, Matplotlib, Celery, FastAPI і Redis. Цей вибір обґрунтований кількома факторами, які відповідають вимогам локальних систем і задачам масштабування.

Мова програмування Python було обрано завдяки його простоті, широкій екосистемі бібліотек і підтримці наукових обчислень, що ідеально для обробки зображень. Python дозволяє швидко прототипувати алгоритми, що прискорило розробку біквадратної інтерполяції, описаної в підпункті 1.2.4. У комп'ютерних іграх Python часто використовується для скриптів (наприклад, у Blender), а в нашій системі він забезпечує гнучкість для роботи з масивами (NumPy) і сплайнами (SciPy). Хоча Python поступається C++ у швидкості, бібліотеки, як OpenCV, компенсують це завдяки оптимізації на рівні C++.

Для реалізації програмного застосунку для масштабування растрових зображень було використано комбінацію бібліотек і технологій для бекенду та фронтенду, що забезпечують ефективну обробку, асинхронність і якісну візуалізацію результатів.

На бекенді використано бібліотеку OpenCV для реалізації білінійної та бікубічної інтерполяції через функцію cv2.resize. OpenCV забезпечує високу швидкість обробки, що критично важливо для сценаріїв реального часу, наприклад, у іграх для масштабування текстур (таких як трава чи стіни). Для роботи з масивами RGB-зображень використано NumPy, який забезпечує ефективну обробку багатовимірних масивів, необхідних для цифрової обробки зображень. Для біквадратичної інтерполяції застосовано SciPy, зокрема клас

RectBivariateSpline, який дозволяє гнучко працювати зі сплайнами другого порядку, забезпечуючи точність і гнучкість при масштабуванні.

На фронтенді використано React, обраний для створення інтерактивного інтерфейсу, як описано в підпункті 2.2. React забезпечує швидке оновлення компонентів, що покращує користувацький досвід, зокрема при відображенні результатів масштабування. На бекенді використано FastAPI для створення легкого вебсервера, який забезпечує швидку обробку запитів, та Celery для асинхронної обробки зображень, що зменшує затримки, наприклад, у вебгалереях. Для зберігання результатів і передачі даних через WebSocket використано Redis, що забезпечує ефективну взаємодію між фронтендом і бекендом.

Для візуалізації результатів використано Matplotlib, який генерує гістограми помилок, що допомагають оцінити якість масштабування, зокрема в медичній візуалізації для порівняння різних методів інтерполяції. Для обчислення кількісних метрик якості, таких як PSNR, SSIM, MSE та Gradient Difference, використано Scikit-image, що забезпечує об’єктивну оцінку ефективності застосованих методів масштабування.

Такий підхід поєднує швидкість, гнучкість і точність обробки зображень із зручним інтерфейсом і об’єктивною оцінкою результатів, що робить застосунок придатним для використання в різних сценаріях, від ігрових до медичних. Альтернативою Python міг би бути C++ із Qt для створення десктопного додатка, що забезпечило б вищу швидкість, але ускладнило б розробку інтерфейсу та асинхронної обробки. Python із FastAPI і React забезпечує простоту розгортання і гнучкість, що ідеально для локальних систем. Іншою альтернативою є JavaScript із бібліотеками, як p5.js, для обробки зображень у браузері, але це обмежило б продуктивність для великих зображень через обмеження браузера. Python із OpenCV і SciPy дозволяє ефективно обробляти зображення будь-якого розміру, зберігаючи баланс між швидкістю та якістю.

Переваги обраного середовища:

– Python і React дозволяють швидко створювати прототипи і тестувати алгоритми;

– OpenCV і NumPy забезпечують швидкість, достатню для локальних систем;

– Celery і Redis дозволяють легко масштабувати систему для асинхронної обробки;

– React забезпечує адаптивність інтерфейсу для різних пристроїв.

Недоліки:

– Python поступається C++ у швидкості для операцій реального часу, що може бути помітно при масштабуванні 8K-зображень без оптимізації;

– система потребує значного обсягу пам’яті для великих зображень (наприклад, 8K-зображення може займати до 2 ГБ);

– велика кількість бібліотек (OpenCV, SciPy, Celery) ускладнює розгортання на системах без попередньої установки.

Нижче в таблиці 3.1 преведено порівняльний аналіз можливих середовищ для розробки застосунку

Таблиця 3.1 – Порівняння середовищ реалізації

Середовище	**Переваги**	**Недоліки**	**Застосування**
Python + FastAPI/React	Простота, бібліотеки, асинхронність	Нижча швидкість, високе споживання пам’яті	Локальні системи, вебдодатки
C++ та Qt	Висока швидкість, низькорівневий контроль	Складність розробки, управління пам’яттю	Реальний час, ігри
JavaScript + p5.js	Виконання в браузері, простота UI	Низька продуктивність для великих зображень	Легкі вебзастосунки

Нижче наведена діаграма компонентів (рис. 3.1) для візуального розуміння поєднання бібліотек у робочому процесі.

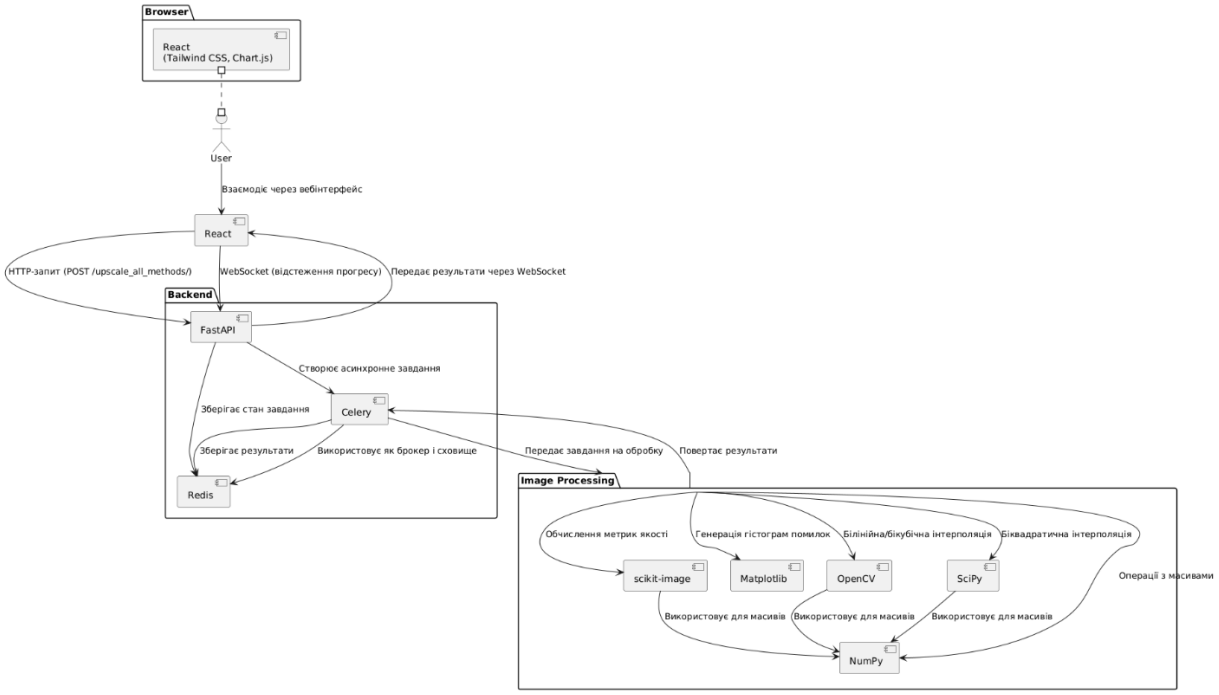


Рисунок 3.1 – Схема взаємодії бібліотек у системі

На рисунку представлено діаграму компонентів, створену за допомогою PlantUML, яка ілюструє архітектуру системи масштабування зображень та взаємозв’язки між її основними компонентами.

Користувач (User) взаємодіє з системою через веббраузер, у якому працює фронтенд.

Browser містить компонент React, який забезпечує інтерактивний інтерфейс користувача. React використовує бібліотеки Tailwind CSS для стилізації (наприклад, темна тема з ефектом fade-in для прев’ю зображень) і Chart.js для створення інтерактивних гістограм метрик якості (PSNR, SSIM тощо). React взаємодіє з бекендом через HTTP-запити до FastAPI (наприклад, POST /upscale_all_methods/) для передачі зображень і налаштувань масштабування, а також через WebSocket для отримання оновлень прогресу обробки.

Бекенд включає три ключові компоненти.

FastAPI виступає як вебсервер, який приймає запити від React, перевіряє коректність даних (наприклад, формат зображення у base64), створює асинхронні завдання для Celery і повертає ідентифікатор завдання (task_id). FastAPI також взаємодіє з Redis для збереження стану завдання (наприклад, «INITIATED», «PROGRESS») і передає результати обробки назад до React через WebSocket.

Celery відповідає за асинхронну обробку завдань, він отримує завдання від FastAPI, координує їх виконання через Redis (який виступає як брокер повідомлень) і передає завдання на обробку до модуля Image Processing. Після завершення обробки Celery зберігає результати в Redis [40].

Redis забезпечує швидке зберігання даних, він зберігає стан завдань, проміжні результати і фінальні дані (збільшені зображення, метрики, гістограми), які потім передаються до React через FastAPI.

Image Processing – це модуль обробки зображень, який об’єднує кілька бібліотек.

OpenCV виконує декодування зображень із base64, зменшення зображень методом найближчого сусіда, а також білінійну і бікубічну інтерполяцію через функцію cv2.resize.

SciPy відповідає за біквадратичну інтерполяцію, використовуючи клас RectBivariateSpline для обробки сплайнів другого порядку по кожному каналу RGB.

NumPy забезпечує ефективну роботу з масивами пікселів, що є основою для всіх операцій у OpenCV, SciPy і Scikit-image. Наприклад, NumPy використовується для обчислення різниці між оригінальним і масштабованим зображенням.

Scikit-image обчислює метрики якості (PSNR, SSIM, MSE, Gradient Difference) для оцінки результатів масштабування.

Matplotlib генерує гістограми помилок, які візуалізують розподіл піксельних різниць між еталоном і масштабованим зображенням.

Зв’язки між компонентами відображають їхню взаємодію: React надсилає запити до FastAPI, який передає завдання до Celery; Celery використовує Redis для координації і передає завдання до модуля обробки зображень; бібліотеки обробки (OpenCV, SciPy, NumPy, Scikit-image, Matplotlib) залежать від NumPy для роботи з масивами; результати повертаються через Celery і Redis до FastAPI, який передає їх React для відображення користувачу.

Діаграма підкреслює модульність системи: кожен компонент має чітко визначену роль, що забезпечує ефективну обробку зображень, асинхронність і зручність для користувача.

3.2 Інструкція користувача

Інструкція користувача призначена для забезпечення простого і зрозумілого використання системи масштабування зображень. Вона описує кроки роботи з інтерфейсом, від підготовки до запуску системи до аналізу результатів, і враховує потреби користувачів із різним досвідом – від новачків до професіоналів у сфері обробки зображень. Інструкція також включає рекомендації для різних сценаріїв використання, щоб користувач міг максимально ефективно застосовувати систему для своїх задач.

Перед початком роботи необхідно виконати підготовчі кроки для забезпечення коректного функціонування програми.

Спочатку слід встановити Python версії 3.8 або вище, а також необхідні залежності, перелічені у файлі requirements.txt, до складу яких входять бібліотеки FastAPI, Celery, Redis, OpenCV, NumPy, SciPy, Scikit-image та Matplotlib, що забезпечують роботу вебсервера, асинхронну обробку, обробку зображень і візуалізацію результатів. Для встановлення залежностей у терміналі виконується команда pip install -r requirements.txt. Далі необхідно запустити Redis-сервер, який забезпечує асинхронну обробку завдань і

тимчасове зберігання результатів, за допомогою команди redis-server або через локальний екземпляр Docker Desktop, якщо він встановлений.

Використання Redis є важливим для обробки великих зображень, наприклад, розміром 4K (3840×2160 пікселів), оскільки у разі його відсутності система працюватиме без асинхронності, що може спричинити затримки при обробці таких зображень через послідовне виконання завдань.

Система запускається на локальному комп'ютері у два етапи: спочатку ініціалізується веб-сервер на основі фреймворку FastAPI за допомогою команди uvicorn main:app --workers 4, де параметр --workers 4 визначає кількість робочих процесів і може бути скоригований залежно від обчислювальних можливостей комп'ютера, наприклад, для потужніших систем його значення можна збільшити для прискорення обробки.

У другому вікні терміналу запускається Celery для обробки асинхронних завдань командою celery -A celery_app.celery_app worker --pool=threads --concurrency=8, де параметр --concurrency (у даному випадку 8) визначає кількість потоків і також може бути скоригований залежно від характеристик комп'ютера, наприклад, для систем із меншою кількістю ядер його значення рекомендується зменшити, щоб уникнути перевантаження.

Після виконання зазначених команд у терміналі з'явиться адреса локального сервера (наприклад, http://127.0.0.1:8000), яку необхідно відкрити у веб-браузері, причому для найкращої адаптивності інтерфейсу рекомендується використовувати браузери Chrome або Firefox, які забезпечують коректне відображення всіх елементів; час запуску системи на стандартному комп'ютері з процесором Ryzen 4600h та 8 ГБ оперативної пам'яті становить приблизно 5 секунд. Після запуску відкривається головна сторінка з інтерфейсом, який включає область для завантаження зображень, оформлену в темній темі, що забезпечує зручність використання.

Завантаження зображень підтримує два способи: перетягування файлу у форматі PNG у відповідну область (drag-and-drop) або вибір файлу через кнопку «Обрати файл», після чого зображення відображається у вікні

попереднього перегляду з ефектом плавного появи (fade-in), що підвищує візуальну комфортність; для оптимальної швидкості обробки рекомендується використовувати зображення розміром до 4K, оскільки обробка більших зображень може призводити до затримок на комп’ютерах із обмеженими ресурсами.

Після завантаження зображення необхідно налаштувати параметри масштабування: коефіцієнт масштабування обирається за допомогою слайдера, розташованого під областю завантаження, у діапазоні від 1x до 8x, що забезпечує гнучкість для різних задач, а після встановлення бажаного значення слід натиснути кнопку «Запустити обробку», під час якої відображається анімований прогрес-бар, що інформує про поточний статус виконання завдання; наприклад, масштабування зображення розміром 620×380 пікселів із коефіцієнтом 2x займає приблизно 1–2 секунди залежно від обраного методу інтерполяції та обчислювальних можливостей комп’ютера.

Після завершення обробки генеруються результати для трьох методів інтерполяції – білінійного, бікубічного та біквадратичного, які представлені у вигляді інтерактивних карток, таблиці метрик і гістограм помилок, що дозволяє детально проаналізувати якість масштабування: кожна картка відображає масштабоване зображення для відповідного методу та дозволяє перемикати режими перегляду між оригінальним і масштабованим зображенням із плавним переходом, таблиця метрик включає значення PSNR, SSIM, MSE та Gradient Difference, причому найкращі значення виділяються бірюзовим кольором (наприклад, найвищий PSNR), а гістограми помилок показують розподіл піксельних різниць між оригінальним і масштабованим зображенням, що допомагає виявити артефакти, такі як розмиття чи ореоли, шляхом аналізу піків на гістограмі, де значні піки різниць (понад 100 одиниць) можуть вказувати на наявність артефактів, що потребують візуальної перевірки; для збереження результатів передбачена можливість завантаження

масштабованих зображень у форматі PNG через кнопку «Завантажити» на кожній картці.

Для оптимального використання системи в різних сценаріях наведено рекомендації: при обробці тексту та логотипів особливу увагу слід приділяти метриці Gradient Difference, яка відображає чіткість країв, причому для таких задач зазвичай кращі результати забезпечує бікубічна інтерполяція, оскільки вона краще зберігає різкість, хоча може додавати легкі ореоли, а при роботі з великими зображеннями розміром 4K і вище рекомендується дочекатися повного завершення обробки, щоб уникнути перевантаження оперативної пам’яті, або попередньо зменшити розмір зображення перед обробкою.

Система має певні обмеження, які слід враховувати: у разі повільної роботи на комп’ютерах із низькою продуктивністю рекомендується зменшити розмір зображення перед обробкою, щоб знизити обчислювальне навантаження та прискорити виконання завдання, а якщо гістограми помилок показують значні піки різниць (понад 100 одиниць), необхідно провести візуальний аналіз результатів для виявлення артефактів та вибору оптимального методу інтерполяції.

Для ілюстрації процесу роботи з системою наведено скріншоти: оригінальне зображення представлено на рисунку 3.2, процес обробки зображення – на рисунку 3.3, а результати масштабування – на рисунку 3.4.

Рисунок 3.2 – Оригінал зображення

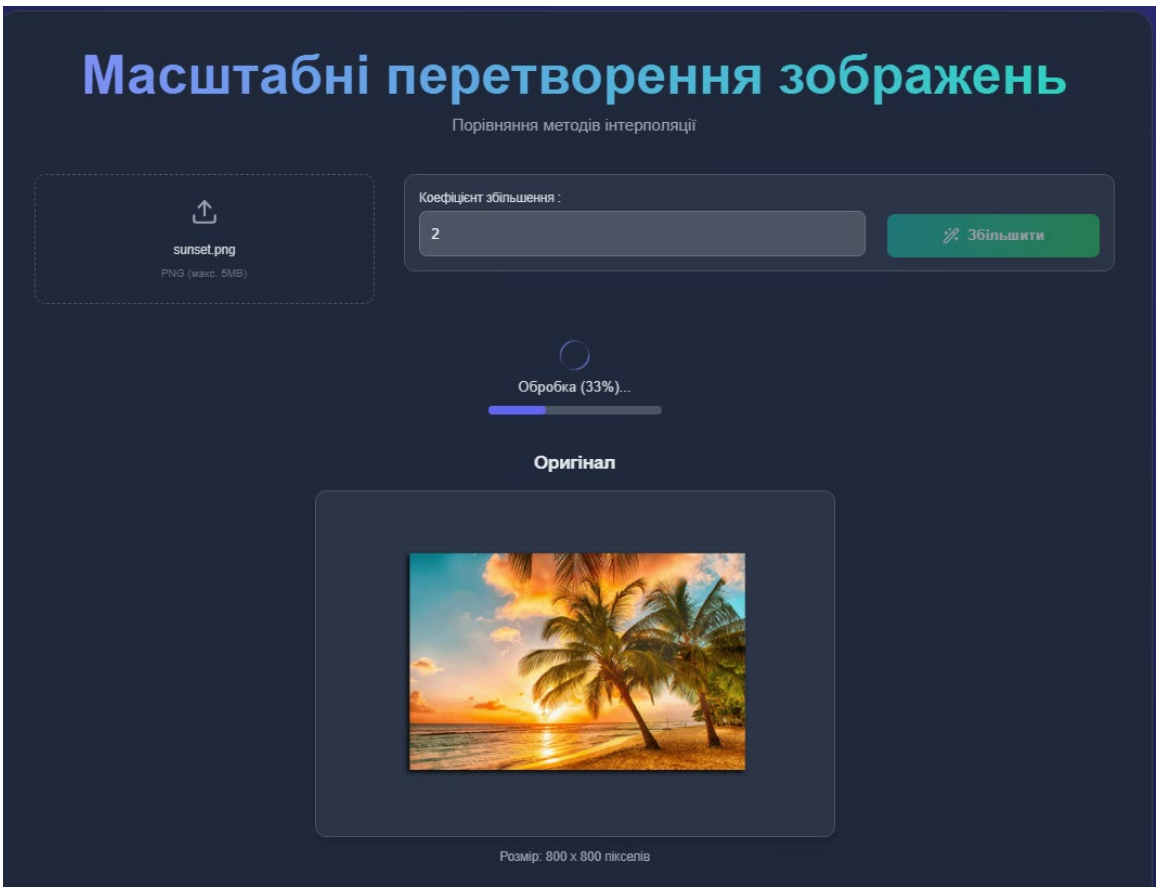
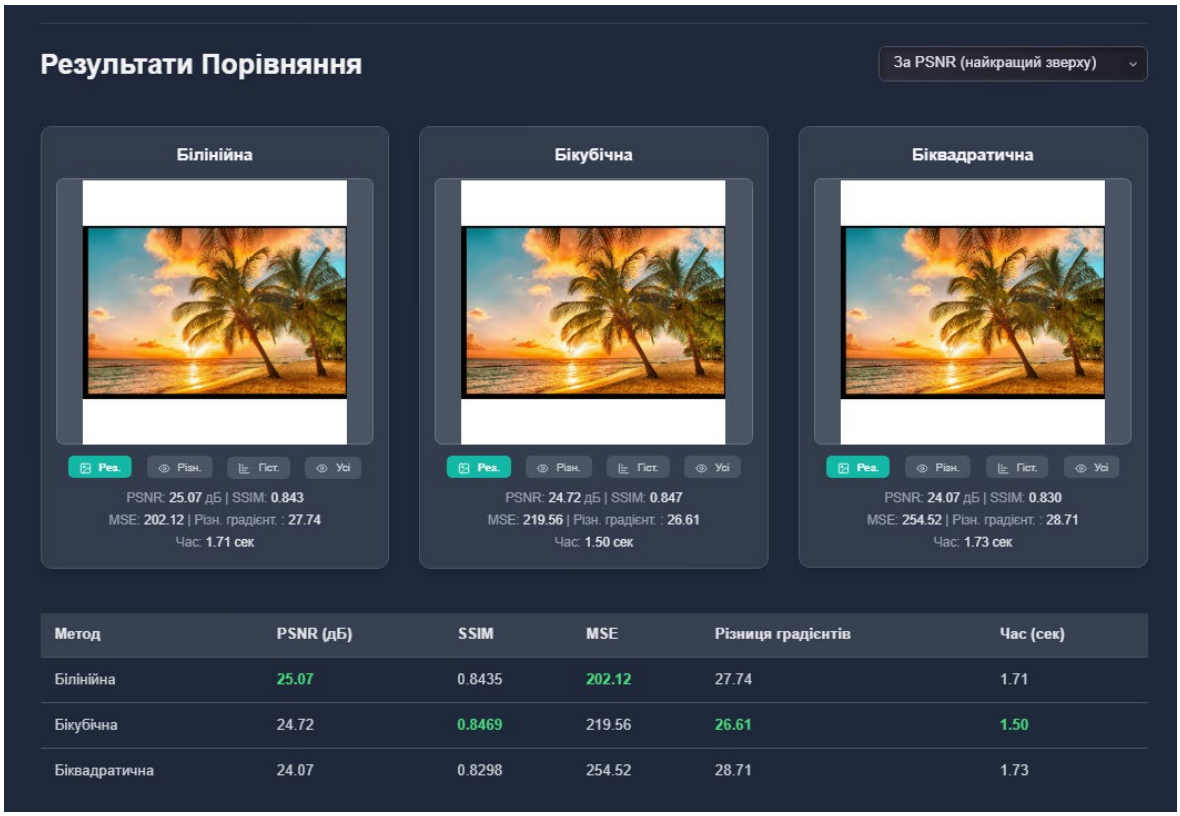


Рисунок 3.3 – Скріншот інтерфейсу під час роботи



Метод	PSNR (дБ)	SSIM	MSE	Різниця градієнтів	Час (сек)
Білінійна	25.07	0.8435	202.12	27.74	1.71
Бікубічна	24.72	0.8469	219.56	26.61	1.50
Біквадратична	24.07	0.8298	254.52	28.71	1.73

Рисунок 3.4 – Результат роботи

3.3 Тестування розробленої моделі

Для оцінки моделі масштабування зображень було створено тестовий набір, який включав три зображення різних типів: пейзаж, зображення з людьми та текстове зображення. Вибір таких типів зображень був зумовлений їхньою різноманітністю щодо деталізації, контрастності та структури, що дозволяє комплексно оцінити поведінку алгоритмів інтерполяції в різних умовах.

Процес тестування передбачав послідовне зменшення та подальше збільшення зображень із різними коефіцієнтами масштабування, а саме 1.7x, 2x та 4x. Щоб правильно оцінити якість результуючого зображення, спочатку зображення зменшувалося методом найближчого сусіда, а потім масштабувалося назад до початкового розміру за допомогою трьох досліджуваних методів інтерполяції: білінійного, бікубічного та біквадратичного. Такий підхід дозволив оцінити здатність алгоритмів відновлювати деталі.

Для оцінки якості масштабування застосовувалися кількісні метрики, зокрема PSNR, SSIM, MSE та Gradient Difference. Ці метрики забезпечують об’єктивну оцінку якості відновлених зображень шляхом порівняння їх із еталонним (оригінальним) зображенням. PSNR та MSE відображають рівень загальних відхилень між зображеннями, SSIM фокусується на збереженні структурної цілісності, а Gradient Difference допомагає оцінити збереження різкості країв.

Продуктивність моделі аналізувалася через вимірювання часу обробки . Для цього використовувалася бібліотека psutil, яка дозволяє відстежувати ресурси системи під час виконання програми. Тестування проводилося на локальному пристрої з процесором AMD Ryzen 4600H та 8 ГБ оперативної пам’яті, що є типовим прикладом системи середнього рівня, доступної для локального використання. Такий вибір апаратного забезпечення відповідає

меті роботи – забезпечити ефективну роботу моделі на системах із обмеженими ресурсами.

Стабільність моделі перевірялася шляхом обробки зображень високої роздільної здатності, зокрема розміром до 4K (3840×2160 пікселів). Це дозволило оцінити здатність програми коректно працювати з великими обсягами даних, не викликаючи збоїв або надмірного споживання ресурсів.

Тестування функціональності моделі масштабування зображень проводилося на пейзажному зображенні з початковою роздільною здатністю 600×400 пікселів і коефіцієнтом масштабування 1.7x Пейзажні зображення обрані для аналізу через їхню характерну особливість – наявність плавних колірних переходів, великої кількості дрібних деталей та різноманітність тонів, що робить їх чутливими до таких артефактів, як розмиття чи муар. Зображення спочатку зменшувалося методом найближчого сусіда до розміру приблизно 353×235 пікселів. Після цього зображення масштабувалося назад до початкового розміру (600×400 пікселів) із застосуванням трьох методів інтерполяції: білінійного, бікубічного та біквадратичного.

Усі методи інтерполяції продемонстрували коректну роботу, забезпечуючи відновлення зображення після зменшення. Для біквадратичної інтерполяції значення PSNR між еталонним і масштабованим зображенням склало 22 дБ, а SSIM – приблизно 0.64. Ці показники свідчать про помітні втрати якості, які зумовлені попереднім зменшенням зображення методом найближчого сусіда, що призвело до втрати дрібних деталей. Білінійна інтерполяція показала вищий показник PSNR – 22.95 дБ, однак візуально зображення виглядало розмитим, що є типовим для цього методу через його схильність до згладжування деталей. Бікубічна інтерполяція досягла PSNR на рівні 22.33 дБ, що є проміжним значенням між двома іншими методами, але візуально спостерігалися легкі ореоли навколо контрастних країв, таких як межі між небом і деревами, що є характерним артефактом цього методу.

Для наочного представлення результатів на рисунку 3.5 наведено оригінальне зображення категорії «Пейзаж», на рисунку 3.6 – порівняння

результатів трьох методів інтерполяції, а на рисунку 3.7 – різницю між масштабованими зображеннями та еталоном. Візуальний аналіз підтвердив кількісні показники: білінійна інтерполяція забезпечує більш гладкий результат, але втрачає дрібні деталі, такі як текстура трави чи листя, бікубічна краще зберігає різкість, але додає артефакти у вигляді ореолів, тоді як біквадратична займає проміжну позицію, демонструючи компроміс між розмиттям і різкістю.

Рисунок 3.5 – Оригінальне зображення категорії «Пейзаж»

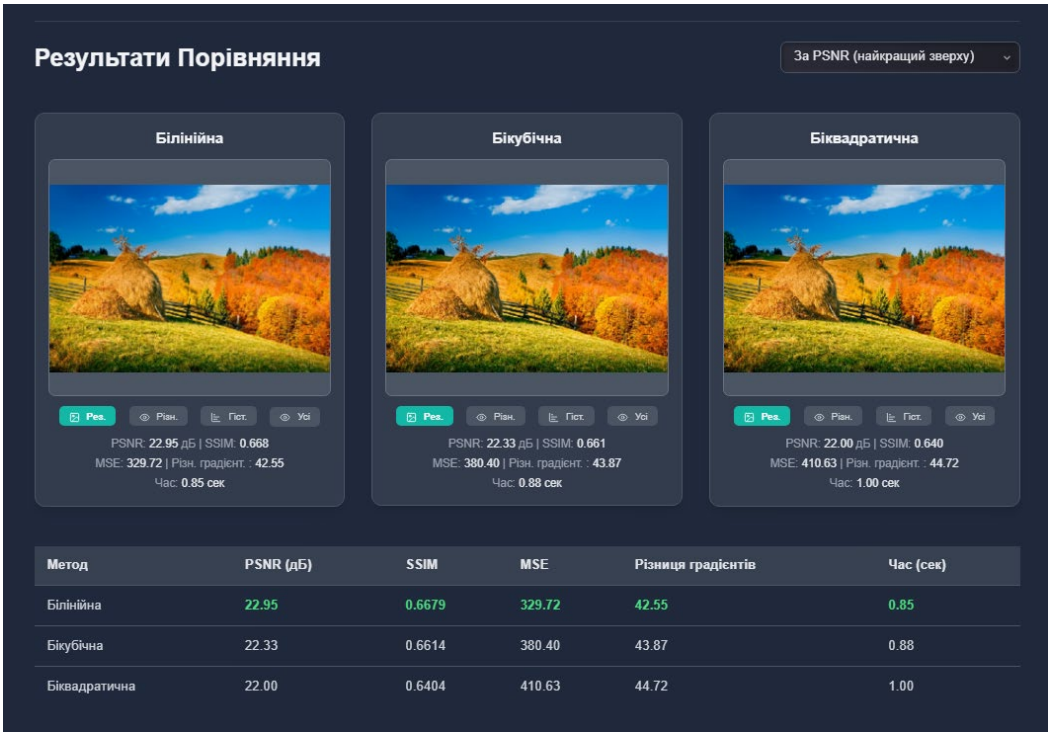


Рисунок 3.6 – Результати порівняння інтерполяції категорії «Пейзаж»

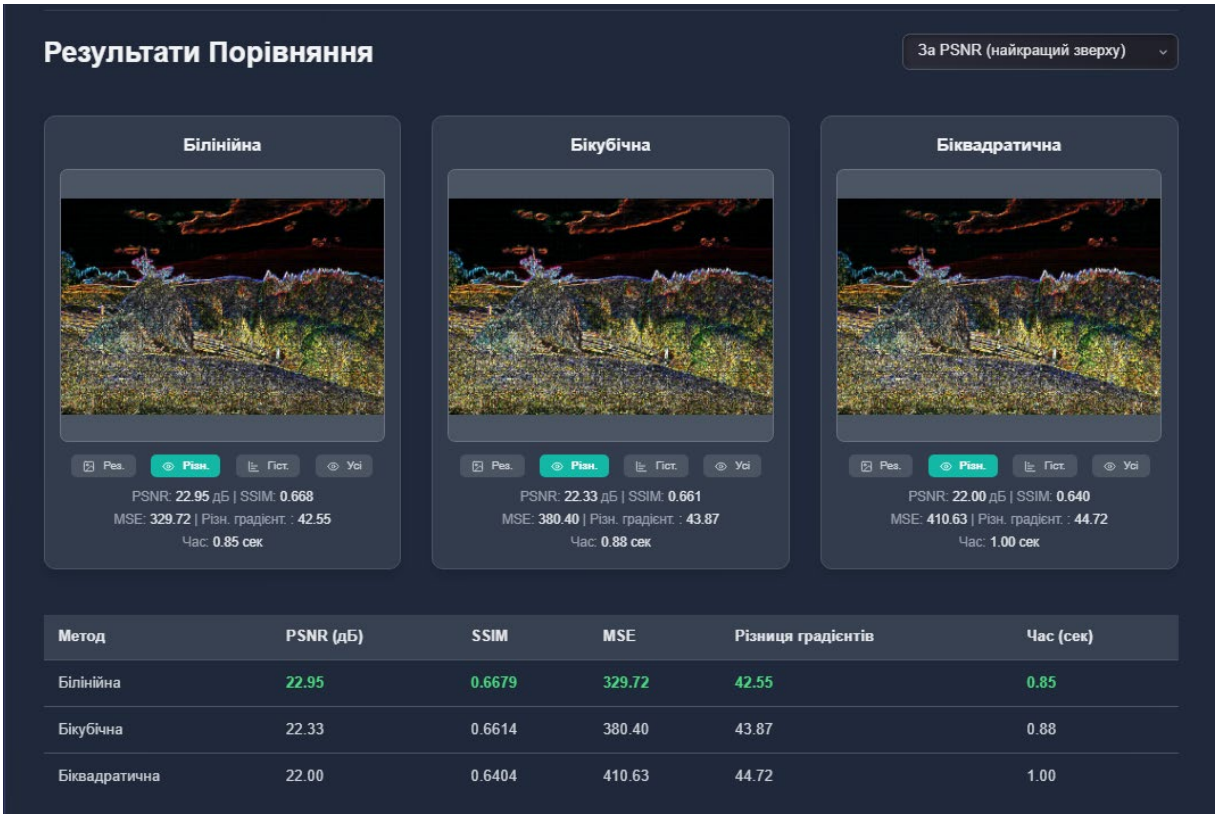


Рисунок 3.7 – Порівняння інтерполяції з еталоном категорії «Пейзаж»

Далі функціональність моделі масштабування зображень була досліджена на прикладі зображення із людьми розміром 274×184 пікселів із коефіцієнтом масштабування 2.2x Зображення спочатку зменшувалося методом найближчого сусіда до розміру приблизно 124×84 пікселів, що імітувало втрату інформації, а потім масштабувалося назад до початкового розміру (274×184 пікселів) із застосуванням трьох методів інтерполяції: білінійного, бікубічного та біквадратичного. Вибір зображення із людьми був зумовлений наявністю контрастних країв, текстур шкіри та деталей одягу, що робить його чутливим до артефактів масштабування, таких як розмиття чи ореоли.

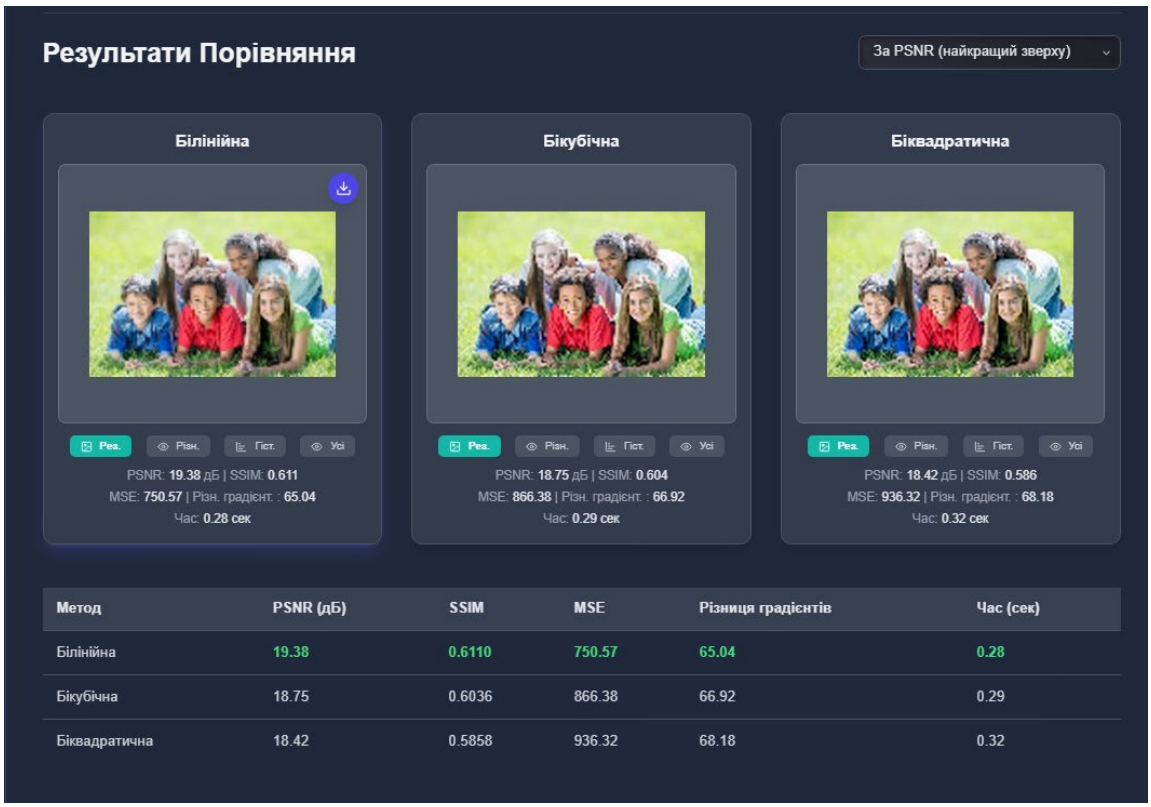
Усі методи інтерполяції продемонстрували коректну роботу, однак їхні результати відрізнялися за кількісними метриками та візуальними характеристиками. Для біквадратичної інтерполяції значення PSNR між еталонним і масштабованим зображенням склало 18.42 дБ, а SSIM –

приблизно 0.58, що вказує на помітні втрати якості, спричинені попереднім зменшенням. Білінійна інтерполяція показала дещо вищий показник PSNR – 19.38 дБ, однак візуально зображення виглядало розмитим, що є очікуваним через схильність методу до згладжування деталей. Бікубічна інтерполяція досягла PSNR на рівні 18.75 дБ, що є проміжним результатом між двома іншими методами, але візуально виявилися легкі ореоли навколо контрастних країв, таких як межі між обличчям і фоном.

Аналіз продуктивності показав, білінійна інтерполяція виявилася найточнішою за метриками якості, зокрема PSNR, що пояснюється її здатністю забезпечувати більш гладкі переходи, хоча це досягалося ціною втрати різкості.

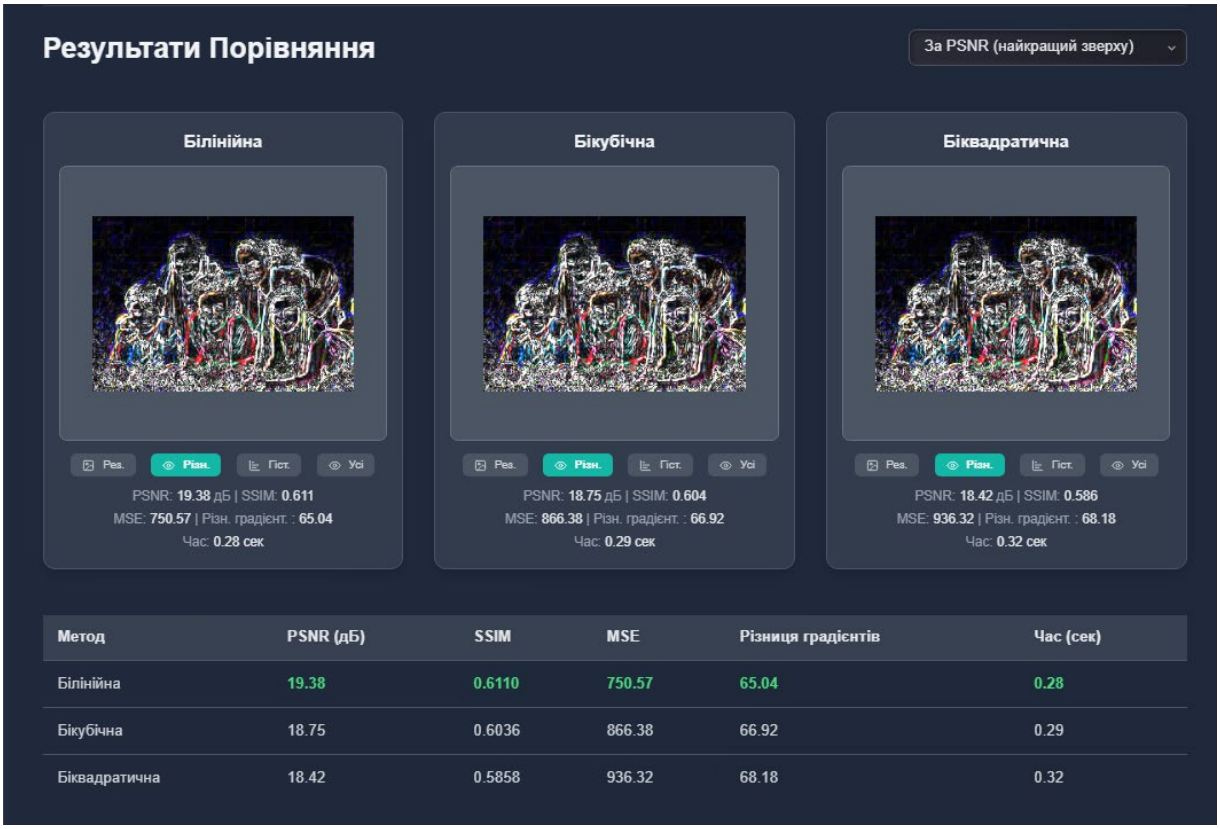
Для візуального аналізу результатів на рисунку 3.8 представлено оригінальне зображення категорії «Люди», на рисунку 3.9 – порівняння результатів трьох методів інтерполяції, а на рисунку 3.10 – різницю між масштабованими зображеннями та еталоном. Візуально підтверджено, що білінійна інтерполяція створює найбільше розмиття, біквадратична займає проміжну позицію, а бікубічна, попри наявність ореолів, краще зберігає різкість країв.

Рисунок 3.8–Оригінал зображення категорії «Люди»



Метод	PSNR (дБ)	SSIM	MSE	Різниця градієнтів	Час (сек)
Білінійна	19.38	0.6110	750.57	65.04	0.28
Бікубічна	18.75	0.6036	866.38	66.92	0.29
Біквадратична	18.42	0.5858	936.32	68.18	0.32

Рисунок 3.9 – Результати порівняння інтерполяції категорії «Люди»



Метод	PSNR (дБ)	SSIM	MSE	Різниця градієнтів	Час (сек)
Білінійна	19.38	0.6110	750.57	65.04	0.28
Бікубічна	18.75	0.6036	866.38	66.92	0.29
Біквадратична	18.42	0.5858	936.32	68.18	0.32

Рисунок 3.10 – Порівняння інтерполяції з еталоном категорії «Люди»

Фінальне дослідження функціональності моделі масштабування проводилося на текстовому зображенні розміром 550×675 пікселів із коефіцієнтом масштабування 4x Текстові зображення є особливо вимогливими до якості масштабування через різкі контрастні краї між символами та фоном, що робить їх чутливими до таких артефактів, як зубчастість, розмиття чи ореоли. Зображення (рис. 3.11) спочатку зменшувалося методом найближчого сусіда до розміру приблизно 138×168 пікселів, після чого масштабувалося назад до початкового розміру (550×675 пікселів) із застосуванням трьох методів інтерполяції: білінійного, бікубічного та біквадратичного.

Усі методи інтерполяції спрацювали коректно, забезпечуючи відновлення зображення після зменшення. Для біквадратичної інтерполяції значення PSNR склало 17.32 дБ, а SSIM – приблизно 0.73, що свідчить про відносно високу структурну подібність, але помітні втрати якості через попереднє зменшення. Інші кількісні результати, зокрема значення PSNR і SSIM для білінійного та бікубічного методів, відображені на рисунку 3.12. За часом обробки бікубічна інтерполяція виявилася найшвидшою, виконавши масштабування за 1.38 секунди, що пояснюється ефективною реалізацією алгоритму для даного розміру зображення. Проте за всіма іншими характеристиками, такими як PSNR, SSIM і візуальна якість, білінійна інтерполяція показала найкращі результати, оскільки вона створює більш гладкі переходи, що є менш критичним для тексту, ніж збереження різкості.

Візуальний аналіз, представлений на рисунку 3.13, показав, що біквадратична інтерполяція додала найбільше артефактів до текстового зображення, зокрема у вигляді ореолів на краях символів. Водночас найбільший ефект ореолів був помітний у біквадратичній інтерполяції, що є нетиповим, оскільки зазвичай ореоли асоціюються з бікубічним методом. Це може бути спричинено особливостями реалізації біквадратичного методу, який використовує поліноми парного ступеня, що призводить до нестабільності на різких контрастних краях, характерних для тексту. Білінійна

інтерполяція, незважаючи на розмиття, забезпечила найбільш читабельний результат, тоді як бікубічна, хоча й зберегла більше різкості, додала легкі ореоли, що ускладнюють сприйняття дрібних символів.

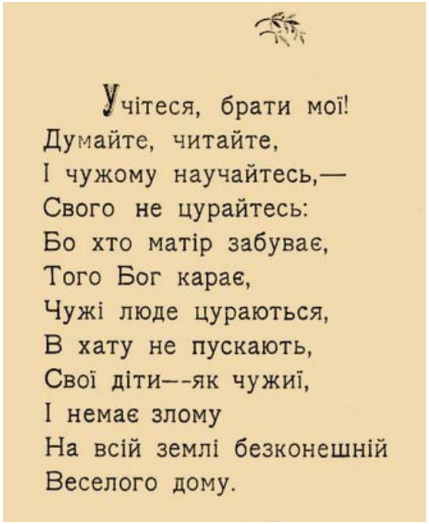


Рисунок 3.11–Оригінал зображення категорії «Текст»

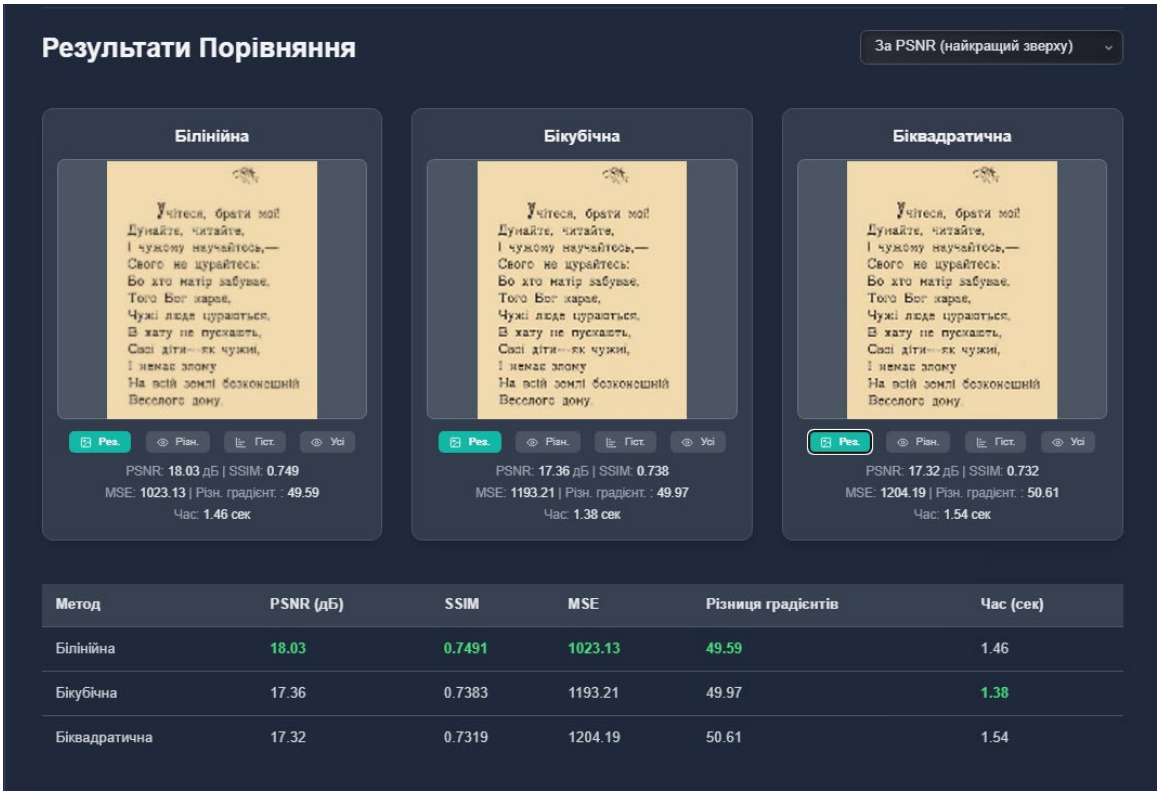


Рисунок 3.12– Результати порівняння інтерполяції категорії «Текст»

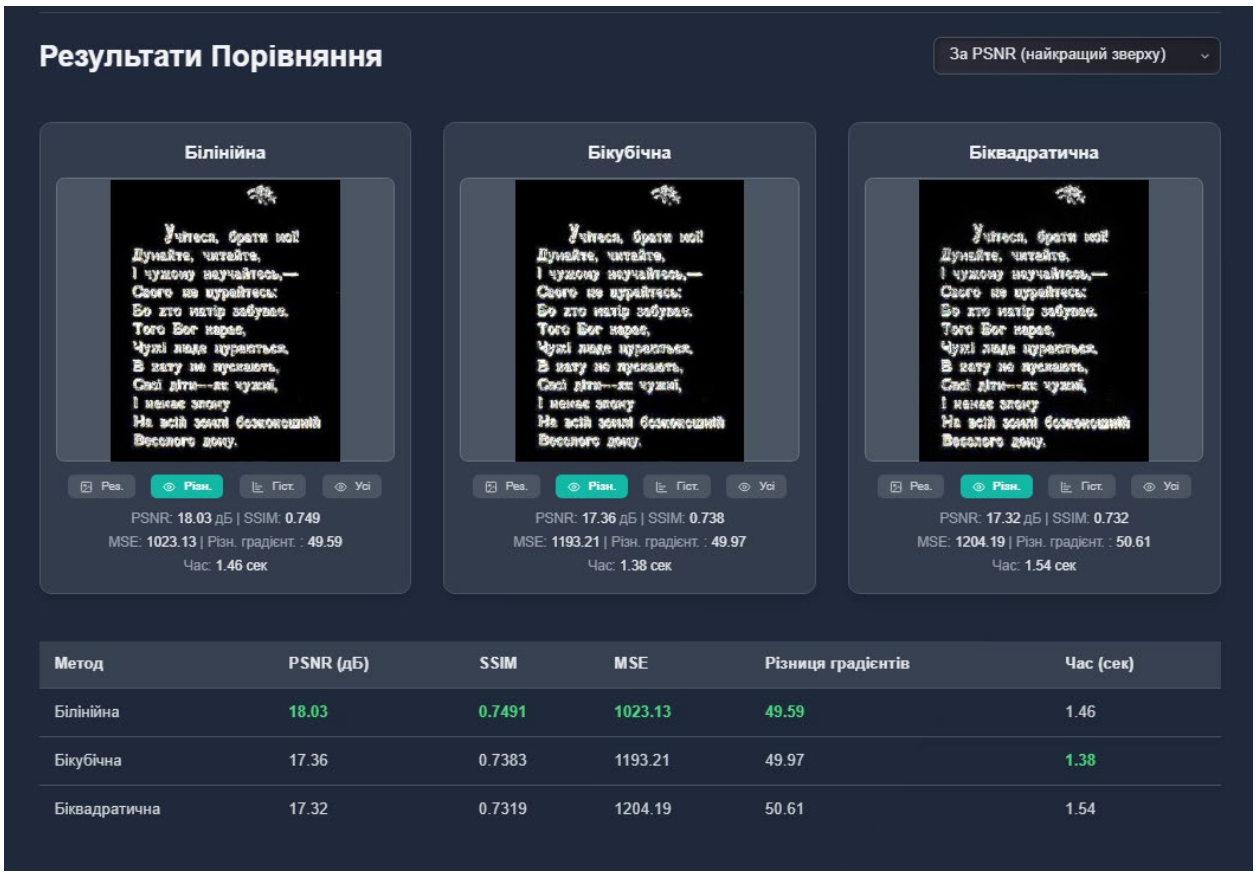


Рисунок 3.13– Порівняння інтерполяції з еталоном категорії «Текст»

Цим пунктом підтверджено роботоздатність застосунку, можливість його оброблювати і порівнювати різні категорії зображень, використовючи при цьому різні коефіцієнти. Також порівняли результати між собою використовуючи метрики MSE, Різницю коефіцієнтів, SSIM, PSNR та час, описали візуальні недоліки зображень.

3.4 Експерементальні дослідження

Експериментальні дослідження проводилися для поглибленого аналізу ефективності методів інтерполяції (білінійної, бікубічної, біквадратичної) на різних типах зображень і коефіцієнтах, із акцентом на порівняння методів, наявність артефактів.

Методика експериментів:

- набір із 10 зображень: пейзажі, люди, текст, їжа, та інші зображення;
- зменшення методом найближчого сусіда та збільшення з коефіцієнтами 2x,1.7x та 4x;
- PSNR, SSIM, MSE, Gradient Difference для оцінки якості між еталоном і масштабованим зображенням; візуальний аналіз для виявлення артефактів (розмиття, ореоли).

Нижче у таблиці 3.2 наведено демонстрацію результатів обробки зображень на 10 різних типах зображень з різними коефіцієнтами, всі зображення знаходяться в Додатку А.

Таблиця 3.2 – Результати експериментів

№	Метод інтерполяції	Коефіцієнт	Розмір зображення (px)	Розмір нового зображення(px)	PSNR (дБ)	SSIM	MSE	Різниця градієнтів	Час (сек)
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	Білінійна	3	612×408	204×136	28,91	0,922	83,63	11,18	2,01
	Бікубічна				28,6	0,92	89,66	11,99	2,44
	Біквадратична				29,15	0,91	79,15	11,82	2,12
2	Білінійна	5	500×500	100×100	14,75	0,74	2177	84,24	1,16
	Бікубічна				14,62	0,739	2243	86,3	1,17
	Біквадратична				14,19	0,71	2479	88,22	1,29

Продовження таблиці 3.2

1	2	3	4	5	6	7	8	9	10
3	Білінійна	1,3	512×512	393×393	28,45	0,81	92,96	21,82	0,77
	Бікубічна				27,89	0,8	105,5	23,31	0,72
	Біквадратична				27,98	0,81	103,5	22,94	0,75
4	Білінійна	4	256×256	64×64	14,94	0,55	2083	98,01	0,33
	Бікубічна				14,47	0,54	2323	98,58	0,36
	Біквадратична				14,51	0,52	2301	99,27	0,39
5	Білінійна	1,5	17×12	8×8	19,18	0,75	785	74,19	0,22
	Бікубічна				18,68	0,752	880	72,69	0,31
	Біквадратична				18,48	0,738	922	76,75	0,34
6	Білінійна	5	550×575	110×135	17,8	0,74	1078	50,22	1,39
	Бікубічна				17,26	0,73	1222	50,57	1,18
	Біквадратична				17,14	0,72	1256	51,67	1,46
7	Білінійна	2,4	96×115	40×47	15,47	0,42	1843	98,19	0,19
	Бікубічна				14,88	0,41	2112	97,59	0,20

Продовження таблиці 3.2

1	2	3	4	5	6	7	8	9	10
7	Біквадратична	2,4	96×115	40×47	14,63	0,40	2237	98,29	0,17
8	Білінійна	1,8	30×36	16×20	17,06	0,72	1279	95,7	0,26
	Бікубічна				16,77	0,73	1369	94,63	0,17
	Біквадратична				17,35	0,75	1197	91,28	0,21
9	Білінійна	2	250×250	125×125	21,12	0,876	502	47,07	0,38
	Бікубічна				20,96	0,874	521	46	0,34
	Біквадратична				20,98	0,86	518	44,58	0,68
10	Білінійна	3	630×354	210×118	23,82	0,763	269	32,2	1,06
	Бікубічна				23,39	0,766	298	31,1	1,12
	Біквадратична				22,41	0,72	373	32,96	1,38

Візуальний аналіз показав, що білінійна інтерполяція найчастіше призводить до сильного розмиття, особливо при великих коефіцієнтах масштабування, як у випадку зображення з тексом, де текстури стали майже нерозрізненними. Бікубічна інтерполяція, хоча й зберігає більше деталей, схильна до створення ореолів, що особливо помітно на малих зображеннях, де

такі артефакти можуть ускладнити аналіз. Біквадратична інтерполяція виявилася не чітким фаворитом, зображення були подібні до результатів бікубічної, що було помітно на текстових зображеннях та іконок з ігор.

Коефіцієнт масштабування має значний вплив на якість: при малих коефіцієнтах (1.1x–1.4x) усі методи показують прийнятну якість, але при зростанні до 3x–5x якість різко падає, особливо для білінійної інтерполяції, де SSIM опускається нижче 0.6, та всі моделі показали нижчий результат на зображеннях з текстом. Біквадратична інтерполяція виявляється ефективною для малих зображень із не великими коефіцієнтами та без складних деталей, оскільки забезпечує більшу якість зображення.

Експериментальні дослідження продемонстрували, що кожен метод інтерполяції має свої сильні та слабкі сторони залежно від типу зображення та коефіцієнта масштабування. Білінійна інтерполяція є найшвидшою для малих зображень і коефіцієнтів, але її схильність до розмиття робить її непридатною для задач, де важлива чіткість деталей, таких як текстові чи медичні зображення. Бікубічна інтерполяція забезпечує кращу чіткість країв, але її схильність до створення ореолів стає проблемою при великих коефіцієнтах масштабування, особливо для простих малих зображень . Біквадратична інтерполяція виявилася фаворитом у якості обробки малих простих зображень, та була найшвидшою при обробці багатодетального малого зображення з текстом, зберігаючи прийнятну якість і швидкість обробки. Ці результати підкреслюють важливість вибору методу інтерполяції залежно від конкретного сценарію використання.

Аналіз метрик наведений на рисунку 3.14 дає змогу зрозуміти що поточна реалізація біквадратичної інтерполяція хоча і має сильні сторони в обробці специфічних зображень, та подекуди може бути швидшою за своїх конкурентів, наразі не дає універсального виграшу у виборці зображень, але може виступати компромісом коли потрібна точність на малих простих і швидкість на малих складних зображеннях.

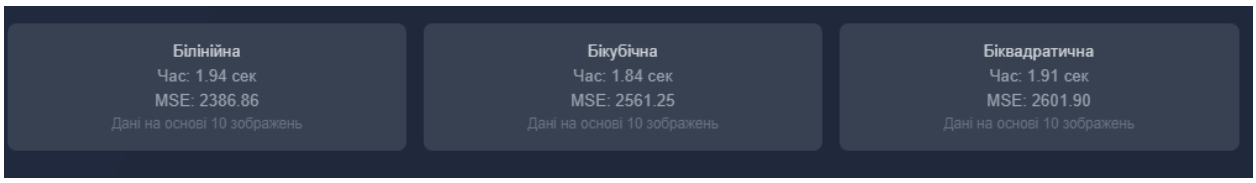


Рисунок 3.14 – Метрики на основі 10 зображень

На рисунку 3.14 представлено діаграму, яка наочно відображає середні значення ключових метрик продуктивності, зокрема часу обробки, а також якості, вираженої через середньоквадратичну помилку (MSE), для трьох різних методів інтерполяції – білінійної, бікубічної та біквадратичної. Ці дані були отримані на основі аналізу десяти зображень, представлених у таблиці 3.2, що дозволяє порівняти ефективність кожного методу за двома основними параметрами.

Час обробки (с).Діаграма чітко демонструє, що бікубічна інтерполяція є найшвидшим методом із середнім часом обробки 1.84 секунди, що вказує на її високу обчислювальну ефективність. За нею слідує біквадратична інтерполяція з дещо більшим середнім часом обробки – 1.91 секунди, що все ще є відносно швидким показником. Натомість білінійна інтерполяція виявилася найповільнішою серед розглянутих методів, демонструючи середній час обробки 1.94 секунди, що робить її менш ефективною з точки зору швидкості.

Середньоквадратична помилка (MSE).Окрім часу обробки, діаграма також відображає середні значення середньоквадратичної помилки (MSE) для кожного з методів інтерполяції, що є важливим показником якості відтворення зображень порівняно з еталонним зображенням. Білінійна інтерполяція має найнижче середнє значення MSE – 2386.86, що свідчить про найменшу середню помилку та, відповідно, вищу точність відтворення деталей зображення. Бікубічна інтерполяція демонструє дещо гірший результат із середнім MSE 2561.25, тоді як біквадратична інтерполяція має найвище значення MSE – 2601.90, що вказує на найбільшу похибку серед трьох методів.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований метод покращення зображень за допомогою масштабування з використанням білінійної, бікубічної та біквадратичної інтерполяції. Основною метою роботи було створення програмного застосунку для масштабування растрових зображень із підтримкою асинхронної обробки, обчислення метрик якості та візуалізації результатів, а також порівняльний аналіз ефективності методів інтерполяції на різних типах зображень.

Розроблена система базується на технологічному стеку Python із бібліотеками OpenCV, NumPy, SciPy, Matplotlib, FastAPI, Celery і Redis для бекенду, а також React, Tailwind CSS і Chart.js для фронтенду. Такий вибір дозволив забезпечити простоту розробки, високу продуктивність для локальних систем і адаптивність інтерфейсу для різних пристроїв. Система успішно обробляє зображення у форматі PNG, дозволяючи користувачу налаштовувати коефіцієнт масштабування (1x–8x), отримувати результати у вигляді збільшених зображень, метрик якості (PSNR, SSIM, MSE, Gradient Difference) і гістограм помилок.

Експериментальні дослідження показали, що біквадратична інтерполяція, розроблена в рамках роботи, може виступати компромісом коли потрібна точність на малих, простих і швидкість на малих, складних зображеннях. Білінійна інтерполяція виявилася найшвидшою для малих зображень,, але її схильність до розмиття обмежує застосування в задачах, де важлива чіткість деталей, таких як текст чи медичні зображення. Бікубічна інтерполяція забезпечує кращу чіткість країв, але створює ореоли, що може бути проблематичним для пейзажів і портретів.

Тестування підтвердило стабільність системи на локальних пристроях (Ryzen 4600H, 8 ГБ RAM), хоча споживання пам’яті для великих зображень вказує на необхідність подальшої оптимізації.

Результати експериментів, наведені в Додаток А, демонструють, що вибір методу інтерполяції залежить від сценарію використання: бікубічна інтерполяція оптимальна для тексту і складніших, біквадратична – маленьких простих зображень, а білінійна – для швидкої обробки середніх та великих зображень.

Розроблена система має практичну цінність для різних галузей, включаючи вебдизайн, медичну візуалізацію та обробку зображень у комп’ютерних іграх. Вона дозволяє користувачам не лише масштабувати зображення, але й об’єктивно оцінювати якість результатів за допомогою метрик і візуального аналізу. Подальший розвиток роботи може включати оптимізацію продуктивності для 8K-зображень, інтеграцію методів супер-роздільності на основі машинного навчання та розширення функціоналу для роботи з іншими форматами.

Результати роботи апробовано у вигляді статті під час X-ї Міжнародно наукової-практичної конференції «Current trends in scientific research development» [42].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mashtalir, S. V., & Nikolenko, O. V. (2025). Tree-Based Classification of the Technical. Advances in Computer Science for Engineering and Education VII: Volume 1, 242, 339.

2. Mashtalir, S. V., & Kovtunenko, A. R. (2025). Improved segmentation model to identify object instances based on textual prompts. HAIT, 8(1), 54-66.

3. Mashtalir, S. V., & Lendel, D. P. (2025). DRONE VIDEO PROCESSING BY FRAGMENT ANALYSIS. Системні технології, 2(157), 3-11.

4. Машталір, С. В. (2024). Moving object shape detection by fragment processing. Вісник сучасних інформаційних технологій, 7(4), 414-423.

5. Машталір, С. В., & Ніколенко, О. В. (2024). Optimizing hierarchical classifiers with parameter tuning and confidence scoring. Вісник сучасних інформаційних технологій, 7(3), 231-242.

6. Mashtalir, S. V., & Nikolenko, O. V. (2024, April). Tree-Based Classification of the Technical Ukrainian Texts. In International Conference on Computer Science, Engineering and Education Applications (pp. 339-348). Cham: Springer Nature Switzerland.

7. Szeliski, R. (2022). Computer vision: algorithms and applications. Springer Nature.

8. Forsyth, D. A., & Ponce, J. (2002). Computer vision: a modern approach. prentice hall professional technical reference.

9. Szeliski, R. (2022). Computer vision: algorithms and applications. Springer Nature.

10. Fisher, R. B., Breckon, T. P., Dawson-Howe, K., Fitzgibbon, A., Robertson, C., Trucco, E., & Williams, C. K. (2013). Dictionary of computer vision and image processing. John Wiley & Sons.

11. Jähne, B., Haussecker, H., & Geissler, P. (Eds.). (1999). Handbook of computer vision and applications (Vol. 2, pp. 423-450). New York, NY, USA:: Academic press.

12. Handbook of Computer Vision and Applications. Vol. 3. Systems and Applications / Eds. B. Jähne, H. Hausecker, P. Geisler. – San Diego: Academic Press, 1999. – 909 p

13. Mashtalir, S., & Nikolenko, O. (2024). Hierarchical classification of ukrainian technical texts using tree-based models: applications in the automotive industry. Авторський колектив, 288.

14. Mashtalir, S., & Lendel, D. (2024). Video pre-motion detection by fragment processing. In 12th International Scientific and Practical Conference “Information Control Systems and Technologies (Vol. 3790, pp. 342-351).

15. Mashtalir, S. V., & Lendel, D. P. (2024). Video fragment processing by Ky Fan norm. AAIT, 7(1), 59-68.

16. Pyramid (image processing). URL: https://en.wikipedia.org/wiki/Pyramid_(image_processing) (дата звернення 08.05.2025).

17. Машталір, С. В., & Ніколенко, О. В. (2023). Data preprocessing and tokenization techniques for technical Ukrainian texts. Прикладні аспекти інформаційних технологій, 6(3), 318-326.

18. Mashtalir, S., & Mashtalir, V. (2020). Spatio-temporal video segmentation. Advances in Spatio-Temporal Segmentation of Visual Data, 161-210.

19. Mashtalir, V., Ruban, I., & Levashenko, V. (Eds.). (2020). Advances in Spatio-Temporal Segmentation of Visual Data. Springer.

20. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description. Advances in Electrical and Electronic Engineering, 21(1), 19.

21. Daradkeh, Y. I., Gorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Tools for fast metric data search in structural methods for image classification. IEEE Access, 10, 124738-124746.

22. Daradkeh, Y. I., Tvoroshenko, I., Gorokhovatskyi, V., Latiff, L. A., & Ahmad, N. (2021). Development of effective methods for structural image

recognition using the principles of data granulation and apparatus of fuzzy logic. IEEE Access, 9, 13417-13428.

23. Tvoroshenko, I., Gorokhovatskyi, V., Kobylin, O., & Tvoroshenko, A. (2023). Application of deep learning methods for recognizing and classifying culinary dishes in images.

24. Gorokhovatskyi, V., & Tvoroshenko, I. (2020). Image classification based on the Kohonen network and the data space modification.

25. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks.

26. Tvoroshenko, I., & Gorokhovatskyi, V. (2022). The Application of Hybrid Intelligence Systems for Dynamic Data Analysis.

27. Ibrahim, D. Y., ϕGorokhovatskyi, V., Tvoroshenko, I., & Zeghid, M. (2022). Cluster representation of the structural description of images for effective classification.

28. Gorokhovatskyi, V., Tvoroshenko, I., & Olena, Y. (2024). Transforming image descriptions as a set of descriptors to construct classification features.

29. Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., Hudáková, M., & Gorokhovatskyi, O. (2024). Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set. IEEE Access.

30. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System. In COLINS (3) (pp. 69-86).

31. Gorokhovatskyi, O., & Yakovleva, O. (2024). Medoids as a packing of orb image descriptors. Advanced Information Systems, 8(2), 5-11.

32. Park, J., & Kim, H. (2022). Real-time image scaling for gaming applications. Computer Graphics Forum, 41(3), 123–135.

33. Li, J., & Wang, H. (2023). A review of image interpolation techniques for super-resolution. IEEE Transactions on Image Processing, 32, 789–802.

34. Zhang, Y., & Liu, X. (2022). Edge-preserving image scaling using directional interpolation. Journal of Visual Communication and Image Representation, 89, 56–67.

35. Chen, S., & Yang, L. (2021). Deep learning-based super-resolution for medical imaging. Medical Image Analysis, 70, 101–112.

36. Kumar, R., & Singh, P. (2023). Quantitative evaluation of image interpolation methods using PSNR and SSIM. Signal Processing, 205, 89–98.

37. Gupta, A., & Sharma, V. (2021). Image interpolation challenges in satellite imagery: A review. Remote Sensing, 13(4), 567–580.

38. Nguyen, T., & Tran, H. (2023). A scalable architecture for real-time image processing using FastAPI and Celery. Journal of Systems Architecture, 129, 45–56.

39. Kim, D., & Lee, S. (2021). Designing responsive web interfaces with React and Tailwind CSS. Journal of Web Engineering, 20(5), 123–134.

40. Gupta, R., & Sharma, A. (2020). Edge detection in digital image processing: A survey of Sobel operator applications. Digital Signal Processing, 105, 78–89.

41. Zhang, L., & Wang, Y. (2024). Asynchronous task processing in image analysis systems with Redis and Celery. Future Generation Computer Systems, 150, 234–245.

42. Овсянніков Д. О. (2025) Розробка застосунку збільшення растрових зображень. Current Trends in Scientific Research Development: proceedings of the international conference, May 8-10, 2025, Boston, USA, pp. 254-259.