НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.627

«До захисту допущено»
В.о. завідувача кафедри
_____ Едуард ЖАРІКОВ
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Методи та засоби кодування растрових зображень на основі подібності фрагментів»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІП-02мп
Портяний Іван Сергійович _____

Керівник:
доцент, к.т.н.
Олійник Юрій Олександрович _____

Рецензент:
професор кафедри ІСТ,
д.т.н., доцент
Корнага Ярослав Ігорович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Едуард ЖАРІКОВ

«___»_____2021р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Портяному Івану Сергійовичу

1. Тема дисертації «Методи та засоби кодування растрових зображень на основі подібності фрагментів», науковий керівник дисертації доцент, к.т.н. Олійник Юрій Олександрович затверджені наказом по університету від «27» жовтня 2021 р. № 3587-с

2. Термін подання студентом дисертації «6» грудня 2021 р.

3. Об’єкт дослідження – кодоване представлення растрового зображення

4. Предмет дослідження – методи та засоби кодування растрового зображення на основі подібності фрагментів

5. Перелік завдань, які потрібно розробити – аналіз наявних методів кодування растрових зображень та їх роботи; розробка методу кодування зображень на основі подібності фрагментів; створення програмної бібліотеки для реалізації розробленого методу; дослідження ефективності методу та засобу.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – 3 плакати

7. Орієнтовний перелік публікацій – одна публікація

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Графічний	доц. Ліщук К.І.		

9.Дата видачі завдання «30» вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Систематизація результатів огляду літератури	09.09.2020	
2	Порівняльний аналіз існуючих методів вирішення задачі	16.09.2020	
3	Формування постановки задачі	23.09.2020	
4	Розробка методу кодування	25.09.2020	
5	Проектування програмного забезпечення	19.10.2020	
6	Розробка програмного забезпечення для реалізації роботи методу	25.10.2020	
7	Виконання експериментальних досліджень	09.11.2020	
8	Оформлення пояснювальної записки	18.11.2021	
9	Подання дисертації на попередній захист	22.11.2021	
10	Подання дисертації на захист	6.12.2021	

Студент Іван ПОРТЯНИЙ

Науковий керівник Юрій ОЛІЙНИК

РЕФЕРАТ

Розмір пояснювальної записки – 89 аркушів, містить 27 ілюстрацій, 38 таблиць, 2 додатки.

Актуальність теми. У роботі розглянуто проблему в області кодування зображень та зменшення їх розміру для зберігання, показано основні особливості існуючих методів кодування зображень, їх переваги та недоліки. Виявлено потребу в розробці нового методу кодування зображень на основі подібності фрагментів.

Мета дослідження. Основною метою є зменшення об’єму даних для збереження зображення за рахунок використання методу кодування на основі наявної бази даних сегментів схожих зображень.

Об’єкт дослідження: кодоване представлення растрового зображення.

Предмет дослідження: методи та засоби кодування растрового зображення на основі подібності фрагментів.

Для реалізації поставленої мети **сформульовані наступні завдання**:

- проаналізувати наявні методи кодування та їх роботу;
- розробити метод кодування зображень на основі подібності фрагментів;
- створити програмну бібліотеку для реалізації розробленого методу;
- дослідити ефективність розробленого методу та програмного засобу.

Наукова новизна результатів магістерської дисертації полягає в тому, що запропоновано новий метод кодування растрових зображень для стиснення та зменшення розміру графічних даних, який використовує подібність фрагментів інших зображень. Результат досягнутий шляхом кодування вхідних зображень за допомогою фрагментів наявних у базі даних зображень.

Практичне значення отриманих результатів полягає в тому, що розроблено метод кодування зображень на основі подібності фрагментів, створено програмну бібліотеку, яка реалізує роботу методу. Розроблена система може бути використана організаціями, які зберігають велику кількість графічних даних і хочуть зекономити витрати на сховища даних.

Зв’язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського" в рамках теми «Методи та технології високопродуктивних обчислень та обробки надвеликих масивів даних». Державний реєстраційний номер 0117U000924.

Апробація. Наукові положення дисертації пройшли апробацію на першій Всеукраїнській науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2021). Секція кафедри інформатики та програмної інженерії. Матеріали конференції. – Київ. – 2021. 22–26 листопада 2021р. – С.198.

Публікації. Наукові положення дисертації опубліковані в:

Матеріали першої Всеукраїнської науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2021). Секція кафедри інформатики та програмної інженерії. Матеріали конференції. – Київ. – 2021. 22–26 листопада 2021р. – С.198.

Ключові слова: КОДУВАННЯ ЗОБРАЖЕНЬ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, СТИСНЕННЯ ЗОБРАЖЕНЬ, ДЕКОДУВАННЯ ЗОБРАЖЕНЬ.

ABSTRACT

Explanatory note size – 89 pages, contains 27 illustrations, 38 tables, 2 appendixes.

Topicality. Examines the problem of image encoding and reducing their size for storing. Work shows the main features of existing methods of image encoding, their advantages and disadvantages. Detected the need to develop a new method of encoding images based on fragments similarity.

The aim of the study. The main target is to reduce the amount of data to save the image by using an encoding method based on the existing database of segments of similar images.

Object of research: coded representation of a raster image.

Subject of research: methods and software of coding a raster image based on the similarity of fragments.

To achieve this goal, the **following tasks** were formulated:

- analyze the existing coding methods and their work;
- to develop a method of encoding images based on the similarity of fragments;
- to create a software library for the implementation of the developed method;
- to investigate the effectiveness of the developed method and software.

The scientific novelty of the results of the master's dissertation is that a new method of encoding raster images to compress and reduce the size of graphic data, which uses the similarity of fragments of other images. The result is achieved by encoding the input images using fragments of images available in the database.

The practical value of the obtained results is that a method of encoding images based on the similarity of fragments is developed and a software library for implementing the method is created. The developed system can be used by organizations that store large amounts of graphical data and want to save on data warehouse costs.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Informatics and Software Engineering of the National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky» within the theme "Methods and technologies of high-performance computing and processing of large data sets." State registration number 0117U000924.

Approbation. The scientific provisions of the dissertation were approbated at the first All- Ukrainian Scientific and Practical Conference of Young Scientists and Students "Software Engineering and Advanced Information Technologies" (SoftTech-2021). Section of the Department of Informatics and Software Engineering. Conference materials. - Kyiv. - 2021. November 22-26, 2021. - P.198.

Publications. The scientific provisions of the dissertation published in:

The first All- Ukrainian Scientific and Practical Conference of Young Scientists and Students "Software Engineering and Advanced Information Technologies" (SoftTech-2021). Section of the Department of Informatics and Software Engineering. Conference materials. - Kyiv. - 2021. November 22-26, 2021. - P.198.

Keywords: IMAGE ENCODING, CONVOLUTIONAL NEURAL NETWORKS, IMAGE COMPRESSION, IMAGE DECODING

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

kNN (k Nearest Neighbor) – алгоритм машинного навчання для класифікації.

LZW (Lempel-Ziv-Welch) – універсальний алгоритм стиснення даних

SSIM (structure similarity) – індекс структурної подібності

lzma (Lempel-Ziv-Markov chain-Algorithm) – алгоритм стиснення даних

ВСТУП

Технології та їх застосування у всіх сферах життя дуже стрімко розвиваються. Технології роботи із зображеннями не виключення, за останні роки було досягнуто значних успіхів у сфері комп'ютерного зору, глибокого навчання та, відповідно, нейронних мереж, які працюють з зображеннями. За допомогою нейронних мереж можна порівнювати зображення, доповнювати їх, виділяти ознаки. Завдяки такому розвитку значно розширюються можливості роботи з картинками.

У сучасному світі достатньо актуальною є проблема зберігання зображень. Адже графічні дані займають достатньо велику кількість дискового простору, при цьому користувачі інтернету завантажують все більше й більше картинок. Також з кожним роком відбувається розвиток фототехніки й якість зображень покращується, відповідно й розмір графічних даних зростає. Сховища даних соціальних мереж, месенджерів, файлообмінників та інших інтернет ресурсів кожного дня наповнюються десятками тисяч нових картинок. Тому постає питання щодо зменшення розміру графічних даних.

Загалом слід зазначити, що одним з найбільш важливих і визначальних аспектів як для зберігання, так і для передачі інформації є її стиснення. Описана вище проблема вирішується за допомогою кодування та стиснення зображень. За допомогою кодування досягається зменшення розміру графічної інформації, що дозволяє економити місце в сховищі, ресурси для передачі даних через мережу й, відповідно, кошти які витрачаються. Зважаючи на це актуальною є розробка методу та засобів кодування зображень.

У сучасному світі вже існують методи для стиснення графічної інформації. Наприклад, jpeg, webp, png та інші. Ці методи зазвичай використовують видалення надлишкової інформації на фото й працюють суто з самим зображенням, але жоден з методів не використовує фрагменти подібних зображень.

Головною ідеєю даної роботи є розробка методу та засобів для кодування зображень на основі визначення подібних фрагментів інших зображень.

Розроблюваний інструмент дозволить зменшити розмір графічного файлу за допомогою кодування.

Метою дослідження є зменшення об’єму даних для збереження зображень за рахунок використання методу кодування на основі наявної бази даних сегментів схожих зображень.

Щоб досягнути поставленої мети необхідно вирішити наступні **завдання**:

- проаналізувати наявні методи кодування та їх роботу;
- розробити метод кодування зображень на основі подібності фрагментів;
- створити програмну бібліотеку для реалізації розробленого методу;
- дослідити ефективність розробленого методу та засобу.

Об’єкт дослідження – це кодоване представлення растрового зображення.

Предметом дослідження є методи та засоби кодування растрового зображення на основі подібності фрагментів.

Науковою новизною є новий метод для кодування растрових зображень для стиснення та зменшення розміру графічних даних, який використовує подібність фрагментів інших зображень.

1 ОБЛАСТЬ ЗАСТОСУВАННЯ І ЗАГАЛЬНИЙ ОГЛЯД МЕТОДІВ КОДУВАННЯ ТА СТИСНЕННЯ ЗОБРАЖЕНЬ

1.1 Стиснення та кодування зображень

Зображення – це двовимірний сигнал, який обробляється зоровою системою людини. Зазвичай сигнали, якими представлено зображення, є аналоговими. Для для того, щоб зберігати, обробляти та передавати їх через комп'ютерні програми, ці сигнали потрібно перетворювати з аналогової форми в цифрову. Зазвичай зображення представляється двовимірним масивом пікселів. Існує два типи 2D зображень – растрова та векторна.

Растрові зображення це зображення, які складаються з дуже маленьких прямокутників – пікселів різного кольору. Кожен піксель має своє місце на зображенні, а також свій власний колір. Кожне зображення має фіксовану кількість пікселів. Якість зображення називають розширенням. Розширення визначається кількістю пікселів, з яких сформовано зображення. Чим більше пікселів розміщено на одиниці площі, тим вище розширення, а отже вищою є якість зображення. Основним недоліком растрових зображень є помітне погіршення якості при збільшенні масштабу.

Векторні ж зображення складаються з великої кількості окремих, масштабованих об'єктів (кривих та ліній), які визначаються за допомогою математичних рівнянь. Під час роботи з векторним зображенням, користувач має можливість задавати опорні точки і характер векторних кривих. Завдяки цьому векторні зображення зберігають дуже високу якість [12].

У даній роботі використовуються саме растрові зображення, так як майже всі зображення, з якими ми зустрічаємось в житті, є растровими, а робота з векторною графікою є достатньо складною й потребує спеціальних інструментів.

У необробленому вигляді зображення охоплюють велику кількість пам'яті, як оперативної, так і фізичної в сховищі даних. Наприклад, растрове зображення розміром 1024 на 1024 пікселі з глибиною кольору 24 біт буде займати 3 МБ. Зрозуміло, що передача і зберігання зображень в такому вигляді є достатньо

ресурсозатратним завданням. Тому задача представлення зображення в більш компактній формі (кодування та стиснення) є достатньо актуальною.

Для того щоб зменшити розмір даних, яке займає зображення, використовується кодування, за допомогою якого відбувається стиснення зображень. У результаті стиснення ми зменшуємо розмір даних для зберігання у сховищі, а також збільшуємо швидкість передачі зображень через мережу, відповідно зменшуються витрати на ресурси для роботи з зображеннями.

1.2 Аналіз і порівняння наявних методів

Алгоритми кодування та стиснення зображень можна розділити на два класи: стиснення без втрат і стиснення з втратами. У першому випадку після компресії інформація про зображення зберігається в повному обсязі, а в другому – частково втрачається.

Під час **стиснення без втрат** стиснуте зображення є таким самим як і оригінальне, після декодування якість зображення не погіршується. Стиснення без втрат можливе, завдяки тому, що у сигналах зображення є значна надмірність. Надмірність пропорційна величині кореляції між вибірками даних зображення. Наприклад, у зображенні зазвичай існує високий ступінь просторової кореляції між сусідніми пікселями [13].

У кодуванні без втрат дані декодованого зображення мають бути ідентичними до оригінального закодованого зображення як кількісно (чисельно), так і якісно (візуально). Але ця вимога доволі сильно обмежує ступінь стиснення, якого можна досягти. Зазвичай ступінь стиснення обмежується коефіцієнтом стиснення в 2 або 3. Такі методи використовуються у вузьких сферах, де є жорсткі вимоги до якості, наприклад зображення в медицині.

Прикладами методів стиснення без втрат є:

- метод Хафмана;
- групове стиснення (RLE – Run Length Encoding);
- метод LZW.

Методи **стиснення з втратами**, як зрозуміло з назви, призводять до невеликої втрати інформації й погіршення якості зображення. Втрачати інформацію нікому не подобається, але насправді певні типи файлів є настільки великими, що для зберігання всіх вихідних даних просто не вистачає місця, крім того, ці дані не завжди потрібні [13].

Сучасні фотоапарати можуть зафіксувати дуже велику кількість деталей на фотографії, але як виявилося, є багато деталей, які ми можемо видалити, тому що людське око просто їх не сприймає. Алгоритми стиснення з втратами спрямовані на пошук і видалення тих деталей, щоб люди навіть і не помітили б. У цих методах втрата інформації є прийнятною, адже видалення несуттєвої інформації з джерела даних може заощадити місце для зберігання.

Прикладами методів стиснення з втратами є:

- JPEG;
- стиснення на основі вейвлет-перетворень;
- фрактальне стиснення.

У таблиці 1.1 наведено порівняльну характеристику методів стиснення [14].

Таблиця 1.1 – Порівняльна характеристика методів стиснення

Методи стиснення з втратами	*Методи стиснення без втрат*
Частина даних оригінального зображення втрачається	Стиснення зображення відбувається без втрати даних
Є незворотніми процесами і якщо дані втрачено, то їх не можна повернути	Зворотні процеси, зображення можна повернути в оригінальний стан
Зменшують розмір файлу, при цьому погіршується якість	Зменшують розмір файлу, але якість зображення залишається незмінною

Продовження таблиці 1.1

Методи стиснення з втратами	*Методи стиснення без втрат*
Зменшення розміру зображення є значним. Розмір файлу можна зменшити приблизно на 60-70%.	Зменшення розміру зображення не є істотним.
Формати збережених файлів JPG, JPEG, GiF	Формати збережених файлів BMP, PNG, RAW

Розглянемо деякі методи стиснень з втратами та без і порівняємо їх.

Метод Хафмана

Цей метод винайдено у 1952 році, ще до того, як з'явився перший цифровий комп'ютер нашого часу, отримавши назву на честь його розробника. Алгоритм було засновано на припущенні, що певні значення сигналів можуть зустрічатись частіше інших. Так і є насправді, у цьому можна переконатись якщо проаналізувати гістограми зображень. Таке відкриття можна використовувати з метою стиснення зображень.

Головна ідея кодування в тому, що сигнали, які найчастіше зустрічаються в послідовності, отримують новий дуже маленький код, порівняно з іншими, а ті сигнали, що зустрічаються найрідше, отримують, навпаки, довгий код. Коли всю послідовність буде оброблено, то найчастіші символи будуть займати найменше місця, навіть менше ніж вони вже займали в оригінальній послідовності, а рідкісні – навпаки більше, але так як вони зустрічаються рідко, то їх розмір не має великого впливу на кінцевий результат.

Алгоритм заснований на такій структурі даних, як бінарне дерево, де кожен вузол має ключ та посилання на нащадків. Символи послідовності вхідного алфавіту утворюють перелік вільних вузлів. Кожен лист дерева має свою вагу, який може дорівнювати як і ймовірності, так і кількості входжень символу в очікуване

повідомлення. Структура дерева будується від листя до кореня, так, щоб символи з меншою частотою були подалі від кореня, ніж символи, що мають більшу частоту.

Щоб побудувати дерево, знаходять частоту входжень для кожного символу. Далі беруться 2 вільник вузла дерева, що мають найменшу вагу, для них створюється батьківський вузол, що має вагу їх підсумованих нащадків. Цей вузол додається у список з вільними вузлами, а його нащадки видаляються з списку частот символів. Для кожної дуги батьків, що виходить ставиться у відповідність 1, а іншій – 0. Побудова дерева продовжується до тих пір, поки у списку з вільними вузлами вже не залишиться жодного вільного вузла, який і буде коренем бінарного дерева. Далі для кожного з символів визначається шлях з бітами, що містяться при переміщенні гілками дерев. Отримана послідовність бітів, що записана в зворотньому порядку є кодом даного символа.

Розглянемо алгоритм на прикладі. Припустимо, що в нас є послідовність з таблицею частот.

Таблиця 1.2 – Таблиця частот

15	7	6	6	5
A	B	C	D	E

З неї видно, що найменша вага у D (6) та E (5). Тому вони приєднуються до нового вузла-батька з вагою 11. Далі вузли D, E видаляються з послідовності вільних вузлів. Вузол D – 0 до батька, E – 1. Робимо теж саме для вузлів B (7) та C (6), бо ця пара є новою парою з найменшими вагами у дереві. На їх основі створюється новий вузол з вагою 13, вузли B, C видаляються аналогічно. На наступному кроці вузлами з найменшою вагою є новоутворений вузол з пари B-C (13) та D-E (11). Створюється новий батько з вагою 24. Вузол B-C - 0 до новоствореного батька, D-E - 1. На останньому етапі залишається тільки два вузли A (15) та вузол B-C-D-E (24). Для них створюється батько, що буде коренем дерева з вагою 39. Усі видалені вільні вузли приєднуються до різних його гілок. На цьому етапі побудова бінарного дерева кодування є закінченим. Тому визначивши

послідовність бітів для кожного символа отримали наступну таблицю символів кода Хафмана.

Таблиця 1.3 – Таблиця символів кода Хафмана

A	0
B	100
C	101
D	110
E	111

Як бачимо сигнал А закодований найменшою кількістю бітів (100), у той час рідкісний символ Е – найбільшим (111). Жоден з отриманих кодів Хафмана не є префіксом іншого, тобто вони можуть бути декодовані під час читання їх з потоку.

Перевага цього алгоритму в тому, що стиснення даних відбувається без втрати інформації. Головним **недоліком** цього алгоритму є те, що для відновлення вмісту повідомлення потрібно мати таблицю частот, якою було зроблене кодування. Як результат, довжина стиснутого повідомлення буде збільшена на довжину таблиці частот, яка має бути надіслана перед даними. Це призводить до того, що збільшується розмір вихідного файлу, відповідно – зменшується стиснення.

Метод Хафмана багато хто використовує у ускладненій формі, коли кодується не тільки один сигнал, а послідовність сигналів. Найчастіше алгоритм використовується для запису графічних зображень у файли, а також він є компонентом алгоритму стиснення даних JPEG, про який йдеться далі [15].

Метод LZW

Цей метод винайдено у 1978 році. Він отримав назву на честь його трьох розробників Lempel, Ziv, Welch. Алгоритм кодує будь-які сигнали. Він подібний до метода Хафмана, але на відміну від нього, для кодування сигналів

використовуються коди однакових довжин, також додатково використовуються коди для послідовностей елементів, що часто зустрічаються.

Спочатку послідовно зчитуються символи вхідного потоку послідовності та відбувається перевірка на існування у новоствореній таблиці рядків такого рядка. Якщо умова дійсна, то зчитується наступний сигнал, інакше – в потік заноситься код попередньо знайденого рядка, цей рядок заноситься в таблицю й пошук відбувається знову.

Для кодування зображення спочатку створюється таблиця кольорів, що наявні у стиснутому зображенні. Це потрібно для того, щоб замість значення кольору пікселя використовувати індекс з таблиці. Аналогічно алгоритму Хафмана, найбільш часто зустрічають кольори з найменшими індексами, а рідші – розміщуються в кінці таблиці. Ця палітра кольорів з індексами повинна розміщуватись між заголовком та зображенням.

Розглянемо алгоритм на прикладі. Припустимо, що в нас є наступний вхідний потік даних:

123 145 201 4 119 89 243 245 59 11 206 145 201 4 243 245

Для нього будується таблиця з індексами для значень або послідовності значень.

Таблиця 1.4 – Таблиця індексів

Індекс	*Значення*
0000	0
...	...
0254	254
0255	255
...	...
4095	xxx xxx xxx

Після кодування потік даних виглядає так:

123 256 119 89 257 59 11 206 256 257

Перевага алгоритму – стискає дані без втрат, і для декодування не потрібно зберігати таблицю рядків у файл для розпакування, як у методі Хафмана. Метод адаптивний та використовує коди змінної довжини. Він також є компонентом алгоритмів стиснення даних JPEG. Головний **недолік** в тому, що алгоритм не проводить аналіз вхідних даних, тому не завжди оптимально визначає індекси.

Метод застосовується для стиснення даних під час запису на жорсткий диск комп'ютера у форматі ARJ, ZIP, GZ, застосовується для запису файлів у форматах TIFF і GIF. Краще працює на зображеннях, що містять однорідні, вільних від шуму ділянки кольорів [16].

Стиснення на основі вейвлет-перетворення

Вейвлети – це функції, що визначені на кінцевому інтервалі. Основна ідея вейвлет-перетворення полягає в представленні довільної функції $f(x)$ як лінійної комбінації набору таких вейвлетів або базисних функцій. Базисні функції отримують з одного прототипу вейвлета, який називається материнським вейвлетом, шляхом розширення (масштабування) і трансляції.

Мета вейвлет-перетворення полягає в тому, щоб змінити дані з часово-просторової області на часово-частотну область, що забезпечує кращі результати стиснення. Розглянемо цей метод покроково.

Перший крок це оцифрування зображення в сигнал, яки є рядком чисел. Оцифроване зображення характеризується рівнем інтенсивності або рівнем сірого кольору в діапазоні від 0 (чорний) до 255 (білий), а також кількістю пікселів на квадратний дюйм.

На другому кроці сигнал розкладається на послідовність вейвлет-коефіцієнтів w. У деяких сигналах багато вейвлет-коефіцієнтів близькі чи дорівнюють нулю. Ці коефіцієнти можна змінити так, щоб вся послідовність містила довгі рядки нулів, за допомогою порогового методу. Саме на третьому кроці завдяки цьому методу послідовність w перетворюється в нову w'.

На четвертому кроці використовується квантування для перетворення послідовності дійсних чисел w' в цілочисельну послідовність q. Найпростішим методом виконання цього перетворення є просто округлення до найближчого цілого числа. Інший варіант – це множення кожного числа з w' на певну константу k, після чого округляти отримане значення до найближчого цілого. На цьому кроці відбуваються втрати якості, так як квантування вносить похибку, адже перетворення w' в q не є оберненим.

А завдяки ентропійному кодуванню ці рядки можна зберігати та надсилати в електронному вигляді з у значно меншому розмірі.

На останньому п’ятому застосовується ентропійне кодування, в результаті якого q стискається в послідовність e. До цього дані не стискались. Перетворення здійснюється з використанням таблиці ентропійного кодування. Рядки нулів кодуються числами від 1 до 100 (і додатково 105, 106), а ненульові цілі числа кодуються від 101 до 104 і від 107 до 254. Ентропійне кодування розроблено таким чином, що числа, які найчастіше зустрічаються в q, будуть займати найменшу кількість даних у фінальній послідовності e [17].

На рисунках 1.1 - 1.3 можна побачити результат роботи розглянутого методу.

Рисунок 1.1 – Вхідне зображення

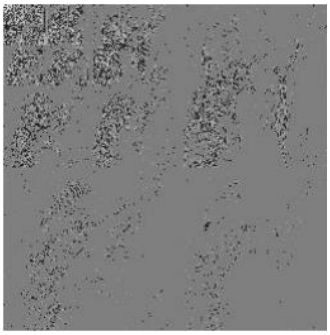


Рисунок 1.2 – Зображення після вейвлет-перетворення

Рисунок 1.3 – Зображення після вейвлет-перетворення

Метод JPEG

Найбільш відомим алгоритмом стиснення даних є JPEG, який розроблений організацією Joint Photographic Experts Group. Його розробили як ефективний метод зберігання 24 бітних зображень з великою глибиною кольору ще у 1991 році. Алгоритм допускає як стиснення без втрат, так і з ними.

Спочатку зображення конвертується з простору кольорів RGB в YCbCr за допомогою стандартної чіткості ITU-R BT.601 та ITU-R BT.709. У даному випадку Y – компонент яскравості, Cb та Cr є синьою та червоною кольороразностними компонентами. В цьому випадку Y не змінюється, щоб не змінити яскравість вхідних даних. YCbCr має велику надмірність, тому не є ефективним для зберігання та передачі зображень.

На наступному кроці значення каналів розбивається на блоки зі стороною у 8 пікселів. Після цього кожен з розбитих блоків перетворюється згідно дискретно-косинусного-перетворення. Тобто із зображення дістається все те, що не потрібно. Це застосовується для того, щоб зрозуміти описує блок складну структуру (візерунок, волосся), або монотонну частину (море, шпалери). На цьому етапі можливе стиснення у 64 рази, якщо 64 схожих за кольором блоки можна описати лише одним. Результуюча матриця коефіцієнтів 8x8 показує зміни величин компонентів. Якщо коефіцієнт високий – зображення має чіткі переходи кольору та яскравості, а якщо низький, то плавні зміни.

Після дискретно-косинусного-перетворення утворені коефіцієнти матриці квантуються. Це потрібно для створення можливості ще більшого стиснення зображень. Кожен з елементів ділиться на константе число, яке визначається за

наступним правилом: для більшого стиснення пам’яті – константа більше 1, для меншого – рівна 1.

Після цього етапу в матриці з’являється велика кількість нулів, які потрібно прибрати і послідовність нулів замінюється на всього 1 нуль з числом, що позначає кількість замінених нулів. На цьому етапі можна отримати стискання у 3 рази.

Останнім етапом є кодування коефіцієнтів методом Хафмана, який вже було розглянуто вище [18].

Розглянемо приклад роботи методу JPEG. На рисунку 1.4 зображено вхідне зображення формату *.bmp з розміром даних 263,2 кілобайти.

Рисунок 1.4 – Вхідне зображення

На рисунку 1.5 зображено вихідне зображення формату *.jpeg з розміром даних 63 кілобайти.

Рисунок 1.5 – Вихідне зображення

Перевагами алгоритму є те, що існує можливість керувати співвідношенням розміру до якості, задаючи ступінь стиснення. За допомогою алгоритму JPEG досягаються великі коефіцієнти стиснення при візуально високій якості зображення. Коефіцієнт стискання є опціональним, так як він задається

користувачем. Алгоритм застосовується і показує найкращі результати на зображеннях, які містять плавні переходи кольору та яскравості. Але є менш придатний для стиснення текстової, символьної графіки, а також креслень.

Недоліками алгоритму є те, що при високому коефіцієнті стиснення зображення може розділитись на окремі квадратні області (розміром 8x8), так як відбудуться дуже великі втрати в низьких частотах при квантуванні, після чого відновлення вихідних даних стає неможливим. Крім того, можливе виникнення ефекту Гіббса – ореоли на межах різких переходів кольорів. Також, оскільки це алгоритм стиснення з втратами, то подальший аналіз і обробка зображення майже неможливе.

Метод фрактального стиснення

Аналізуючи останні дослідження було знайдено метод фрактального стиснення. Новий алгоритм було запропоновано Майклом Барнслі та Аланом Слоуном на основі системи доменних та рангових блоків зображення, а також квадратних блоків, що покривали все зображення. Цей метод є одним з найкращих, бо має досить хороше співвідношення між якістю відновленим зображенням і глибиною стиснення.

Ідея методу стиснення заснована на пошуку подібних ділянок зображення, які дуже близькі між собою за розподілом інтенсивності пікселів та розрахунку коефіцієнтів. Результатом такого алгоритму є набір параметрів системи ітерованих функцій (Iterated Function System).

Головна проблема застосування IFS полягає в перетворенні простору цього зображення в деякий окремий елемент, тому що реальне зображення немає повної самоподібності. Тому необхідно вводити нові сегментовані IFS. Їх відмінністю є те, що команда діє не на все зображення, а лише на окрему частину. Тому перед цим потрібно кластеризувати ці ділянки зображення. Опис зображення відбувається шляхом серії стискаючих перетворень, що застосовуються на різних ділянках кластеризації. Після чого вихідні дані будуть дуже відрізнятися за розміром від реальних.

Також зображення буде легко відновити. Декомпресія відбувається через побудову атрактора за допомогою IFS, коефіцієнти яких було визначено під час компресії, тому від вибору початкового зображення нічого не зміниться. Отже, його можна взяти навіть звичайне чорне. Важливо і те, що декодування фрактальним методом не залежить від роздільної здатності.

Перевагами методу є те, що в порівнянні з іншими – цей алгоритм позбавляє таких недоліків, як втрата важливих деталей зображення. На сьогодні є багато удосконалень та оптимізацій методу. І не менш важливим є його широка сфера застосування. Метод має досить хороше співвідношення між якістю відновленим зображенням і глибиною стиснення. Наприклад для стискання зображення Коха було використано лише чотирма афінними перетвореннями, які були визначені лише за допомогою 24 коефіцієнтів [19].

Недоліком є складність, бо досить довго відбувається процес пошуку доменних блоків, адже потрібний повний перебір (потрібно порівняти два масиви).

Фрактальне стиснення можна розглядати, як найкращу альтернативу існуючим методам, заснованих на рядах Фур'є (яскравим прикладом є метод JPEG).

Для спрощення порівняння методів побудуємо таблицю з плюсами й мінусами кожного методу.

Таблиця 1.5 – Таблиця порівняння методів

	Метод Хафмана	LZW	Стиснення на основі вейвлет-перетворення	Фрактальне стиснення	JPEG
Збереження початкової якості	+	+	-	-	-
Відсутність потреби в збереженні допоміжних даних для декодування	-	-	+	+	+

Продовження таблиці 1.5

	Метод Хафмана	LZW	Стиснення на основі вейвлет-перетворення	Фрактальне стиснення	JPEG
Високий коефіцієнт стиснення	-	-	+	+	+
Можливість самостійно задавати коефіцієнт стиснення	-	-	-	-	+

З таблиці бачимо що в кожного метода є свої переваги та недоліки. Для різних задач можна використовувати відповідні методи. Але одним з найбільш використовуваних та найпоширеніших методів є JPEG.

Проаналізувавши існуючі методи, враховуючи сучасні дослідження, не вдалося знайти подібного методу, до того, що розглядається в даній науковій роботі.

1.3. Постановка задачі

Метою дисертації є зменшення об’єму даних для збереження зображення за рахунок використання методу кодування на основі наявної бази даних сегментів схожих зображень.

Для того, щоб досягнути поставленої мети необхідно вирішити наступні задачі:

- проаналізувати наявні методи кодування растрових зображень та їх роботу;
- розробити метод кодування зображень на основі подібності фрагментів;
- створити програмну бібліотеку для реалізації розробленого методу;
- дослідити ефективність розробленого методу та засобу.

Висновки до розділу

Розглянуто поняття зображення, типи зображень, різницю між растровими та векторними зображеннями, обґрунтовано вибір растрових зображень. Описано призначення кодування та стиснення зображень.

Також розглянуто типи стиснення: з втратами та без втрат. Для кожного з типів наведено приклади алгоритмів, серед яких були виділені ті, які найчастіше використовуються: метод Хафмана, LZW, JPEG.

Проведено огляд найпопулярніших методів, після чого виконано їх порівняльну характеристику. Виконано постановку задачі.

2 РОЗРОБКА МЕТОДУ КОДУВАННЯ ЗОБРАЖЕНЬ

2.1 Загальний опис методу

Метод кодування зображень на основі подібності фрагментів містить в собі два основних процеси:

1) Процес наповнення бази даних фрагментів.
2) Процес кодування зображення.

Кожен з процесів складається з окремих етапів. Деякі етапи будуть унікальними для процесу, а інші будуть зустрічатися в обох. Кодування зображення буде доступним тільки після наповнення бази даних фрагментів зображень.

Діаграма діяльності для обох процесів наведена у графічному матеріалі до дисертації.

2.1.1 Процес наповнення бази даних фрагментами

Перед кодуванням зображення, потрібно наповнити базу даних фрагментами інших зображень, за допомогою яких буде виконано кодування. Щоб досягти вищої якості закодованого зображення, для наповнення бази краще обирати зображення, які схожі на ті, що будуть кодуватись.

Процес містить наступні етапи:

1) Етап зчитування та підготовки даних.
2) Етап розділення зображення на фрагменти.
3) Етап обробки фрагментів та виділення ознак
4) Етап збереження фрагментів в базу

2.1.2 Процес кодування зображення

Після того, як було наповнено базу даних, можна переходити до кодування нових зображень. Частина етапів повторюється відповідно до попереднього. Вхідне зображення для кодування аналогічно потрібно зчитати та підготувати, після чого розділити його на бажану кількість фрагментів та виділити ознаки з

фрагментів. Це зумовлено тим, що вхідні зображення та їх фрагменти мають бути в тому ж форматі, що й наявні в базі.

Процес містить наступні етапи:

1) Етап зчитування та підготовки даних.
2) Етап розділення зображення на фрагменти.
3) Етап обробки фрагментів та виділення ознак.
4) Етап пошуку подібних фрагментів.
5) Етап формування та стиснення масиву ідентифікаторів.

Для процесу кодування зображення наведено діаграму послідовності у графічному матеріалі.

2.1.3 Вимоги до вхідних даних

Алгоритм отримує на вхід зображення, яке задовольняє наступні умови:

- формат зображення: *.jpg, *.jpeg, *.png [2];
- зображення повинно бути 24-бітним;
- зображення повинно бути растровим.

Розмірність зображення може бути будь-якою.

На рисунку 2.1 зображено приклад вхідного зображення, розмірністю 200 x 300 пікселів.

Рисунок 2.1 – Приклад вхідного зображення

2.1.3 Вихідні дані

Після кодування повертається набір байтів. Цей набір являється стиснутим масивом ідентифікаторів фрагментів з бази, якими закодовано вхідне зображення.

На рисунку 2.2 зображено приклад вихідних даних в байтах.

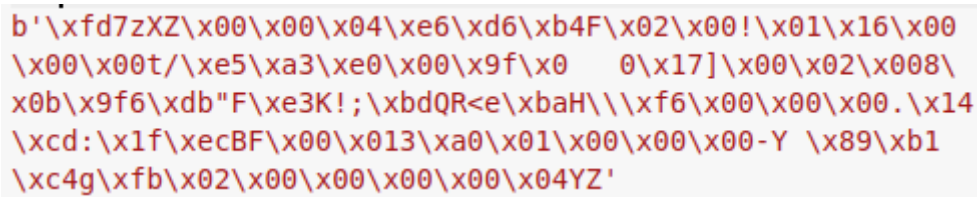


Рисунок 2.2 – Приклад вихідних даних в байтах

Приклад набору ідентифікаторів фрагментів, якими закодовано вхідне зображення, наведено на рисунку 2.3.

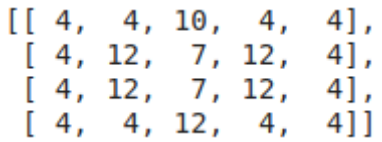


Рисунок 2.3 – Приклад набору ідентифікаторів фрагментів

2.2 Огляд етапів методу

2.2.1 Етап зчитування та підготовки даних

На цьому етапі отримується вхідне зображення, яке при зчитуванні перетворюється у масив, кожним елементом якого є цифрове представлення пікселя у форматі RGB. Далі, за потреби, виконується зміна розширення (висоти та/або ширини) зображення.

2.2.2 Етап розділення зображення на фрагменти

Під час цього етапу зображення розділяється на фрагменти. Для початку визначаються розміри фрагментів і, в залежності від цього, їх кількість. Після чого масив пікселів, отриманий при зчитуванні зображення, розділяється на задану кількість фрагментів відповідного розміру.

Під час процесу наповнення бази вхідне зображення розділяється на фрагменти декілька разів. Кожного разу задаються різні розміри фрагментів, від відносно невеликих (3 x 4, 5 x 6) до більших (16 x 24, 24 x 32). Це дозволяє отримати більшу кількість різноманітних фрагментів з одного зображення, що при кодуванні дасть можливість вибирати бажану деталізацію.

Під час процесу кодування зображення на цьому етапі є можливість обрати розміри та число фрагментів. Чим менший розмір фрагментів та більша їх кількість, тим кращою буде якість декодованого зображення, але й водночас більший розмір вихідних закодованих даних.

2.2.3 Етап обробки фрагментів та виділення ознак

На цьому етапі відбувається підготовка отриманих фрагментів, так як виділення ознак виконується за допомогою нейронної мережі, яка потребує відповідний формат вхідних даних [8].

Кожен з фрагментів перетворюється в тензор, потім змінюється розмір тензора на 224 x 224 x 3 і дані конвертуються в формат float32.

Після цього підготовлені дані фрагменту подаються на вхід нейронній мережі, яка виділяє ознаки фрагменту й на виході повертає масив чисел-ознак.

2.2.4 Етап збереження фрагментів у базу

На цьому етапі ми маємо набір фрагментів, їх характеристики та виділені ознаки. Ці дані зберігаються в базу даних. Запис містить наступний формат:

- ідентифікатор фрагменту в базі даних у цілочисельному форматі;
- фрагмент у вигляді масиву пікселів, які представлені у форматі RGB;
- розмір фрагменту;
- масив виділених ознак.

За допомогою ідентифікатору буде відбуватись кодування нових зображень. Сам фрагмент потрібно зберігати для того, щоб у подальшому декодувати зображення. Розмір знадобиться при фільтрації й відкиданні зайвих фрагментів. А за допомогою масиву виділених ознак буде відбуватись порівняння й визначення рівня подібності.

2.2.5 Етап пошуку подібних фрагментів

Цей етап є одним з головних в розробленому методі. Передбачається, що при переході на даний етап база даних фрагментів заповнена, а вхідне зображення розділене на фрагменти, з яких отримано ознаки.

Отже, вхідними даними етапу є масив з наборами ознак кожного фрагменту, отриманого із зображення, яке потрібно закодувати, та самі фрагменти.

На цьому етапі для фрагментів вхідного зображення знаходиться найбільш схожий фрагмент з бази, шляхом порівняння ознак. Спочатку визначається розмір фрагменту, після чого з бази вибираються всі фрагменти з таким же розміром. Далі для кожного вхідного фрагменту виконується пошук найбільш схожого й зберігається його ідентифікатор.

Схожість фрагментів визначається за допомогою порівняння ознак. У нашому випадку між наборами ознак знаходиться евклідова відстань й фрагмент з найменшим значенням відстані вважається найбільш схожим.

На виході з даного етапу отримується масив з ідентифікаторами схожих зображень.

2.2.6 Етап формування та стиснення масиву ідентифікаторів

Зазвичай розмір даних масиву ідентифікаторів уже буде набагато меншим за розмір даних вхідного зображення. Але на останньому етапі масив також стискається для того, щоб ще більше зменшити розмір вихідних даних закодованого зображення. Існує багато методів для стиснення масиву цілих чисел, вони будуть описані в наступному розділі дисертації.

Кінцевим результатом даного етапу є набір байтів – стиснений методом lzma (Lempel-Ziv-Markov chain-Algorithm) масив ідентифікаторів подібних фрагментів.

2.3 Декодування зображення

Декодування зображення є достатньо простим процесом. У закодованому вигляді зображення є набором байтів, процес його відновлення описано в наступних кроках:

1) Декодування байтів, в результаті якого отримується масив з ідентифікаторами фрагментів, якими закодовано зображення.
2) По кожному ідентифікатору з бази отримується фрагмент у форматі масиву пікселів.
3) Всі фрагменти поєднуються в один набір пікселів у потрібному порядку.
4) Числове представлення зображення переводиться у графічне.

На рисунку 2.4 зображено приклад вхідного зображення, яке кодувалось, а на рисунку 2.5 – приклад декодованого зображення.

Рисунок 2.4 – Вхідне зображення

Рисунок 2.5 – Декодоване зображення

Висновки до розділу

У даному розділі дисертації було наведено детальний покроковий опис роботи методу кодування зображень, формат і приклад вхідних та вихідних даних.

Розглянуто два процеси, які містяться в методі: наповнення бази та безпосередньо кодування, показано їх схематичне зображення. Кожен з процесів складається з набору етапів, які можуть повторюватись у кожному з них. Для всіх етапів наведено їх детальний огляд та принцип роботи.

Також описано декодування, під час якого з набору закодованих байтів отримується зображення.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вимоги до програмного забезпечення

Відповідно до завдання даної роботи розроблене програмне забезпечення для методу кодування зображень повинно задовольняти наступні функціональні вимоги:

- зчитування та обробка вхідного зображення;
- розділення зображення на фрагменти;
- відновлення суцільного зображення з набору фрагментів;
- виділення ознак з фрагменту;
- пошук подібного фрагменту;
- стиснення набору ідентифікаторів фрагментів;
- кодування зображення;
- декодування зображення;
- зчитування та запис даних про фрагменти в базу даних.

3.2 Огляд обраних засобів розробки

Для розробки програмного продукту було використано наступні засоби та технології розробки:

- мова програмування Python;
- середовища розробки PyCharm та Google Colab;
- відкрита програмна бібліотека для роботи з нейронними мережами tensorflow;
- бібліотеки для роботи із зображенням opencv і scikit-image;
- бібліотека для роботи з масивами даних numpy;
- база даних MongoDB та pymongo – бібліотека для роботи з відповідною базою;
- технологія для керування версіями git.

Нижче розглянемо обрані технології та наведемо обґрунтування вибору.

Python

Python – високорівнева мова програмування, яка за останні роки стала однією з найбільш швидкозростаючих мов. Це потужний інструмент, який підтримує багато парадигм програмування: імперативну, структурну, процедурну, функціональну, а також об’єктно-орієнтовану.

Синтаксис мови є дуже зрозумілим і інтуїтивним, завдяки цьому програми, написані на Python, легко сприймати та читати. Простота синтаксису дозволяє швидко розібратись з мовою та забезпечити високу продуктивність програмістів.

Python це інтерпретована мова, тобто до запуску код є звичайним текстовим файлом, а його переведення на машинний відбувається під час виконання програми. Також Python підтримується основними операційними системами. А за допомогою автоматичного збору сміття, уникаються витоки пам’яті.

Недоліком обраної мови програмування є відносно невелика швидкість виконання програм, у порівнянні з C та C++.

Так як Python є однією з основних мов програмування в машинному та глибокому навчанні, то однією з головних причин вибору Python для даної наукової дисертації є саме широкий вибір засобів для роботи з машинним навчанням, нейронними мережами та зображеннями. Більше того, у обраної мови висока продуктивність при обробці великих масивів даних, що стане в нагоді при роботі з наборами фрагментів [11].

У якості середовищ розробки було використано PyCharm та Google Colab.

PyCharm є зручним для розробки, містить велику кількість вбудованих плагінів, підказок та перевірок коду, що значно полегшує та пришвидшує розробку.

Google Colab це хмарний сервіс, який направлений на спрощення досліджень в області машинного та глибокого навчання, який базується на середовищі Jupyter Notebook. Дуже зручним є те, що Colab використовується в браузері й немає потреби встановлювати додаткове програмне забезпечення на персональний комп’ютер. Використання Google Colab, дозволяє отримати віддалений доступ до машини з підключеною відеокартою, що сильно спрощує роботу з великими

обрахунками. У даній дисертації Colab використовується для проведення експериментів [20].

Tensorflow

Tensorflow – це фреймворк з відкритим вихідним кодом, який був розроблений дослідниками Google і використовується для проектування, створення та вивчення моделей глибокого навчання, машинного навчання та нейромереж.

Кожне обчислення в TensorFlow представляється як граф потоку даних. У нього є два елементи: який представляє одиниці обчислень та той, що представляє одиниці даних [4].

Важливими особливостями даного фреймворку є:

- функціонал для легкого визначення, оптимізації та обрахунків математичних виразів за допомогою тензорів, які являють собою багатовимірні масиви;
- підтримка програмування глибоких нейронних мереж та методів машинного навчання;
- функції обрахунків з різними наборами даних, які можна масштабувати;
- фреймворк використовує обрахунки на графічному процесорі;
- фреймворк включає в себе функцію оптимізації пам’яті й використовуваних даних.

У даній роботі tensorflow використовується для обробки фрагментів та перетворення їх в тензори. Далі з отриманих тензорів за допомогою цього фреймворку виділяється список ознак фрагменту [9].

Scikit-image і opencv

Scikit-image і opencv – це бібліотеки з відкритим вихідним кодом, що використовуються для комп’ютерного бачення та призначені для аналізу, класифікації та обробки зображень [6]. Містять велику кількість функцій для роботи із зображеннями, наприклад:

- імпорт і перегляд зображення;
- кадрування;

- зміна розміру зображення;
- поворот, перевод в градації сірого;
- розмиття та згладжування;
- метрики порівняння зображення.

У дисертації ці бібліотеки використовуються для зчитування, первинної обробки та підготовки зображень, а також для порівняння якості.

NumPy

Бібліотека numpy надає загальні числові й математичні у вигляді швидких пре-скомпільованих функції, які об’єднано у високорівневі пакети. За допомогою цієї бібліотеки додається підтримка великих багатовимірних масивів та матриць і забезпечується функціонал, який можна порівняти з функціоналом MatLab [6].

Усі зображення та фрагменти у програмній реалізації являтимуть собою numpy-масиви даних.

MongoDB

У якості бази даних для збереження фрагментів і інформації про них було обрано MongoDB. Це документо-орієнтована, NoSQL база даних, яка забезпечує високу масштабованість та продуктивність. MongoDB використовує колекції та документи, замість таблиць та записів, які є типових для реляційних баз даних, таких як MySQL, PostgreSQL.

Колекція являє собою групу документів, які не обов’язково повинні мати однакову структуру, а навпаки можуть містити складну й різну структуру даних. Сам документ можна представити як набір ключів та значень [21].

Для роботи з базою даних було обрано бібліотеку pymongo – офіційний драйвер MongoDB для Python, який рекомендують використовувати розробники MongoDB. Ця бібліотека надає потрібний функціонал для взаємодії з базою.

3.3 Архітектура ПЗ

Програмний продукт являє собою бібліотеку з набором класів та методів, які реалізують роботу з зображеннями та кодування й декодування.

При написанні програмного забезпечення було використано функціональну та об’єктно-орієнтовану парадигми програмування.

Об’єктно-орієнтовану парадигму використано для написання класів, які реалізують функціонал роботи із зображеннями та функціонал кодування і декодування, а також для класів зображення та фрагменту.

Функціональна парадигма застосовується при реалізації утиліт для роботи з зображеннями.

У таблиці 3.1 наведено перелік створених класів та опис їх функцій.

Таблиця 3.1 – Опис класів

Клас	*Опис*
ImageManager	Клас, який відповідає за підготовку зображення, розділення на фрагменти та, навпаки, відновлення зображення з набору фрагментів.
BaseImageEncoder	Базовий клас для роботи з кодуванням і декодуванням зображення. Реалізує метод для підготовки зображення до виділення ознак і метод виділення ознак. Визначає набір основних методів, які повинні бути реалізовані в класах, що наслідують базовий.
ImageEncoder	Клас, який реалізує кодування та декодування зображення, використовуючи базу зображень, які зберігаються локально, не використовує базу даних. Використовується для проведення експериментів.
DBImageEncoder	Клас, який реалізує кодування та декодування зображення, використовуючи базу даних фрагментів зображень.
Image	Клас, який використовується для полегшення роботи з зображенням. Реалізує представлення зображення в коді.
Fragment	Клас, який використовується для представлення фрагменту зображення.

Для відображення зв’язків між описаними вище класами було побудовано діаграму класів, яку наведено у графічному матеріалі.

У таблиці 3.2 наведено опис атрибутів класів.

Таблиця 3.2 – Опис атрибутів класів

Клас	*Атрибут*	*Тип*	*Опис*
ImageManager	scale_percent	int	Відсоток зменшення розміру вхідного зображення
	height	int	Висота для зміни розміру зображення
	width	int	Ширина для зміни розміру зображення
BaseImageEncoder	tf_model	tensorflow.keras.Model	Модель нейронної мережі для виділення ознак
ImageEncoder	tf_model	tensorflow.keras.Model	Модель нейронної мережі для виділення ознак
	trees_int	int	Глибина побудови дерева
DBImageEncoder	tf_model	tf.keras.Model	Модель нейронної мережі для виділення ознак
	trees_int	int	Глибина побудови дерева
	db	pymongo.collection.Collection	Об’єкт колекції з фрагментами в MongoDB

Продовження таблиці 3.2

Клас	*Атрибут*	*Тип*	*Опис*
Image	img	numpy.array	Зображення у форматі масиву пікселів
	name	str	Назва зображення
Fragment	id	int	Ідентифікатор фрагменту
	data	numpy.array	Представлення фрагменту, як масиву пікселів
	size	tuple	Розмір фрагменту
	features	numpy.array	Масив ознак фрагменту

Опис методів класів наведено в таблиці 3.3.

Таблиця 3.3 – Опис методів класів

Клас	*Метод*	*Опис*
ImageManager	prepare_img()	Підготовка зображення
	split_fragments()	Розділення зображення на фрагменти вказаного розміру
	get_image_from_fragments()	Відновлення суцільного зображення з набору фрагментів
BaseImageEncoder	prepare_img_cnn()	Перетворення зображення на тензор
	get_image_feature_vectors()	Виділення ознак із зображення

Продовження таблиці 3.3

Клас	*Метод*	*Опис*
ImageEncoder	prepare_fragments_and_get_features()	Підготовка фрагментів та виділення з них ознак
	build_tree()	Побудова дерева для пошуку подібних фрагментів
	find_most_similar_fragment_cnn()	Пошук подібного фрагменту
	get_new_fragments_indices()	Отримання ідентифікаторів нових фрагментів для кодування
	encode()	Кодування зображення
	decode()	Декодування зображення
	get_new_fragments()	Пошук подібних фрагментів
DBImageEncoder	prepare_fragments_and_get_features()	Підготовка фрагментів та виділення з них ознак
	build_tree()	Побудова дерева для пошуку подібних фрагментів
	find_most_similar_fragment_cnn()	Пошук подібного фрагменту в базі даних
	get_new_fragments_indices()	Отримання ідентифікаторів нових фрагментів для кодування
	encode()	Кодування зображення фрагментами з бази даних

Продовження таблиці 3.3

Клас	*Метод*	*Опис*
DBImageEncoder	decode()	Декодування зображення
	get_new_fragments()	Пошук подібних фрагментів в базі даних
	prepare_and_save_to_db()	Підготовка фрагменту, виділення ознак та збереження в базу
	save_fragment_to_db()	Збереження даних фрагменту в базу
Image	get_size()	Отримання розміру зображення
Fragment	get_height()	Отримання висоти фрагменту
	get_width()	Отримання ширини фрагменту

Також програма містить модуль utils.py, де зберігаються утиліти, опис яких наведено в таблиці 3.4.

Таблиця 3.4 – Опис методів модулю utils.py

Метод	*Опис*
blur_image()	Розмиття зображення
ssim_metric()	Обрахунок метрики схожості зображень SSIM

Структуру документу в колекції даних бази MongoDB описано в таблиці 3.5.

Таблиця 3.5 – Структура документу

Назва поля	*Тип*	*Опис*
_id	ObjectId	Ідентифікатор документу в колекції
id	Integer	Цілочисельний ідентифікатор фрагменту
data	Array	Масив пікселів фрагменту
size	Array	Розмір фрагменту
features	Array	Масив ознак фрагменту

3.4 Реалізація виділення ознак та пошуку подібних фрагментів

Двома найважливішими етапами методу є етап виділення ознак із зображення та пошук подібних фрагментів.

Виділення ознак з фрагментів реалізовано за допомогою згорткових нейронних мереж [7]. Існує велика кількість вже нетренованих мереж. На порталі TensorFlow Hub є широкий вибір згорткових нейронних мереж з різними характеристиками. Для даної роботи було обрано модель ResNet V2 50 [22], яку було натреновано на датасеті ImageNet [23]. Обґрунтування вибору моделі описано в частині 4.

Перед тим як виділяти ознаки, фрагмент потрібно підготувати – перевести в тензори. Реалізація підготовки показана на рисунку 3.1.

```
@staticmethod
def prepare_img_cnn(img: np.array) -> tf.Tensor:
    img = tf.image.resize_with_pad(img, 224, 224)
    img = tf.image.convert_image_dtype(img, tf.float32)[tf.newaxis, ...]

    return img
```

Рисунок 3.1 – Підготовка фрагменту до виділення ознак

Далі за допомогою обраної моделі нейронної мережі з фрагменту виділяються ознаки. Реалізацію цього процесу зображено на рисунку 3.2.

```
def get_image_feature_vectors(self, img: np.array):
    features = self.tf_model(img)
    return np.squeeze(features)
```

Рисунок 3.2 – Реалізація виділення ознак

Далі виконується етап пошуку подібних фрагментів. Це один з найбільш ресурсоємних процесів методу, так як для кожного набору ознак розділених фрагментів потрібно знайти найбільш схожий з великої кількості існуючих в базі даних фрагментів.

Перебір кожного наявного фрагменту й порівняння буде займати дуже багато часу та ресурсів, тому для оптимізації було використано бібліотеку annoy та метод K-Nearest Neighbors [10].

Ознаки виділених з вхідного зображення фрагментів та фрагментів з бази даних об’єднуються в один набір, з якого будується дерево за допомогою бібліотеки annoy. Annoy (Approximate Nearest Neighbors Oh Yeah) – це бібліотека, написана розробниками Spotify, для пошуку точок у просторі, які є близькими до заданої. Реалізація побудови дерева за допомогою обраної бібліотеки зображена на рисунку 3.3.

```
def build_tree(
    self, prepared_fragments_features, prepared_test_fragments_features
):
    dims = prepared_test_fragments_features.shape[1]
    trees_int = self.trees_int
    tree = AnnoyIndex(dims, metric='euclidean')

    i = 0
    for target_fragment in np.concatenate((
        prepared_test_fragments_features, prepared_fragments_features
    )):
        tree.add_item(i, target_fragment)
        i += 1
    tree.build(trees_int)
    return tree
```

Рисунок 3.3 – Побудова дерева

На побудованому дереві annoy дозволяє ефективно використовувати метод машинного навчання K-Nearest Neighbors (KNN) для пошуку подібних фрагментів. Для обрахунку подібності ознак фрагментів всередині дерева використовується евклідова відстань.

Висновки до розділу

У цьому розділі наведено список функціональних вимог до програмного забезпечення, який було сформовано на основі поставлених задач.

Описано засоби й технології розробки, які було обрано: мова програмування Python, tensorflow, opencv, scikit-image та базу даних MongoDB. Описаний набір технологій є одним з найзручніших для роботи із зображеннями та нейронними мережами. Наведено діаграму класів та опис їх атрибутів і методів.

Окремо детально розглянуто реалізацію виділення ознак та пошуку подібних фрагментів, так як вони є одними з найважливіших та ресурсоємних етапів.

4 ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА РОЗРОБЛЕНОГО МЕТОДУ КОДУВАННЯ

4.1 Критерії для оцінки ефективності

Для оцінки ефективності методу, результатів експериментів та порівняння зображень було обрано наступні критерії:

- розмір закодованого зображення;
- відсоток стиснення зображення;
- якість декодованого зображення;
- час, який витрачається на кодування зображення.

Якість декодованого зображення буде визначатись подібністю вхідного та декодованого зображення, яка вимірюватиметься за домогою індексу структурної подібності SSIM (Structural similarity index measure). SSIM було створено для точного визначення різниці двох зображень. Значення індексу знаходиться в межах від -1 до 1. Значення індексу, що дорівнює 1, означає що порівнювані зображення є однаковими [24].

4.2 Набір даних для екпериментів

Для проведення експериментів було обрано набір зображень квітів 102 Flowers Diff Species DataSet [1]. Цей набір містить зображення квітів, що відносяться до 102 різних категорій. Зображення були отримані шляхом пошуку в Інтернеті і фотографування. Для кожної категорії є мінімум 40 зображень з різних ракурсів та з різним освітленням. Дані узято з ресурсу Kaggle[1]. Приклади даних датасету зображено на рисунку 4.1.

Рисунок 4.1 – Приклади зображень з датасету

Для проведення експериментів було обрано один клас квітів, який представлений ста картинками. Приклади картинок зображено на рисунку 4.2.

Рисунок 4.2 – Приклади з обраного набору зображень

Для спрощення розміри всіх зображень було зведено до одного стандартного – 200 x 300 пікселів.

В якості тестового зображення, яке буде кодуватись, було обрано картинку, що зображена на рисунку 4.3.

Рисунок 4.3 – Тестове зображення

Тестове зображення має розмірність 200 x 300 пікселів та розмір даних 180 128 байт.

4.3 Вибір моделі нейронної мережі

У розроблюнованому методі важливу роль відіграє нейронна мережа, за допомогою якої виділяються ознаки із зображення. На порталі TensorFlow Hub [25] міститься велика кількість нетренованих моделей. Для вибору конкретної моделі було проведено експерименти.

Тестове зображення було розділено на 2000 фрагментів розміром 5 x 6 пікселів. У якості бази фрагментів було взято 20 000 фрагментів. У таблиці 4.1 наведено результати кодування тестового зображення з використанням моделей з різними архітектурами.

Таблиця 4.1 – Порівняння моделей

Модель	*Час кодування, секунд*	*Вихідний розмір, байт*	*Відсоток стиснення, %*	*Подібність, SSIM*
MobileNet V2	33.03	3829	97.87	0.53
Inception V3	50.46	3777	97.9	0.84
NASNet-A (mobile)	79.47	3773	97.9	0.76
ResNet V1 50	39.65	3745	97.92	0.82
ResNet V2 50	40.376	3733	97.93	0.84

На рисунках 4.4 – 4.5 зображено графік з результатами експериментів (на графіку метрика подібності SSIM помножена на коефіцієнт 100 для масштабування).

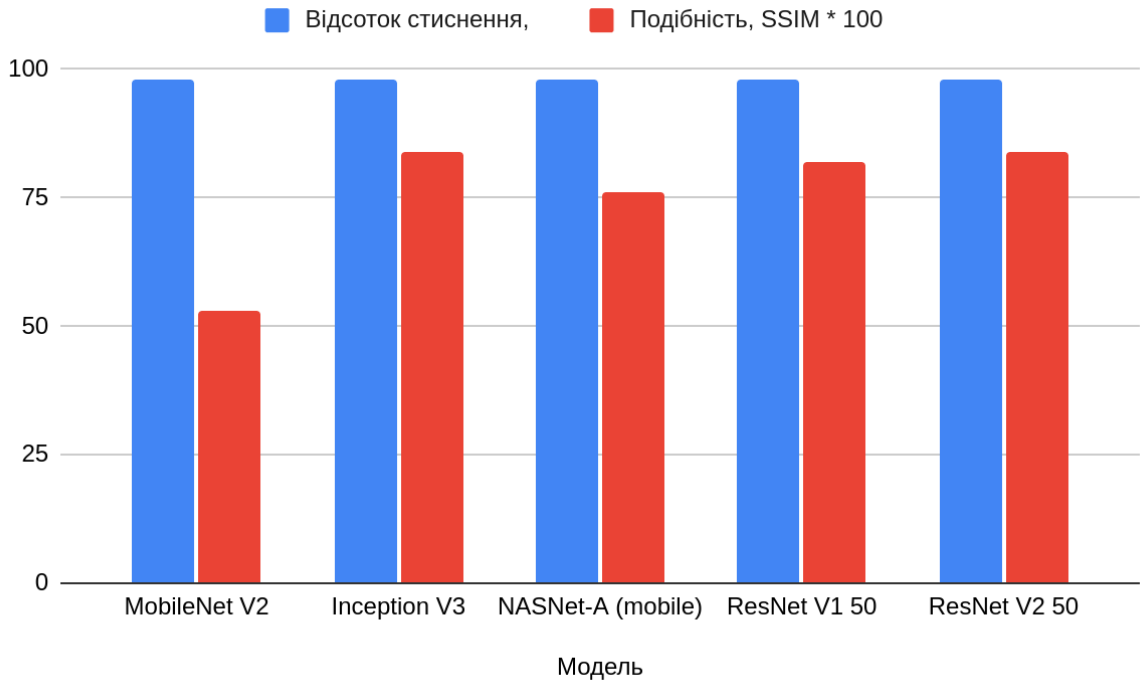


Рисунок 4.4 – Графік рівня стиснення та подібності

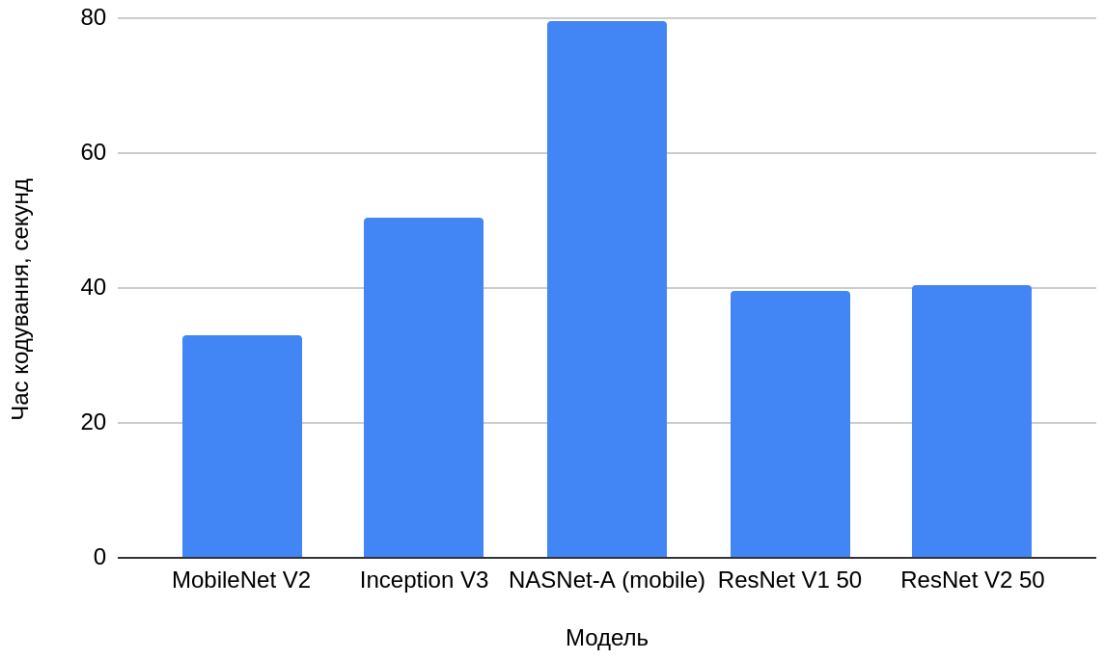


Рисунок 4.5 – Графік часу кодування

З результатів експерименту бачимо, що відсоток стиснення у всіх алгоритмів майже однаковий, тому найкращу модель було обрано за рівнем подібності та часом кодування. Найвища швидкість у моделі MobileNet V2, але рівень подібності в неї дуже низький у порівнянні з іншими. NASNet-A (mobile) потребує значно більше часу на кодування у порівнянні з іншими й має нижчий рівень подібності.

ResNet V1 50 та ResNet V2 50 мають дуже близькі показники й хоча перша працює незначно швидше, та друга має краще показник подібності. ResNet V2 50 і Inception V3 мають однакові значення SSIM, але перша працює швидше, тому для алгоритму було обрано модель ResNet V2 50.

4.4 Вибір алгоритму стиснення масиву ідентифікаторів

Останнім етапом методу кодування зображень є стиснення масиву ідентифікаторів подібних фрагментів [26]. У таблиці 4.2 наведено порівняння алгоритмів стиснення масиву ідентифікаторів.

Таблиця 4.2 – Порівняння алгоритмів стиснення масиву ідентифікаторів

Алгоритм	*Розмір масиву*	*Розмір після стиснення*
lzma	16 128	3829
zlib	16 128	5239
gzip	16 128	5186
bz2	16 128	4246

З результатів бачимо, що найкращим методом є lzma.

4.5 Залежність якості та часу кодування від кількості фрагментів

Важливу роль в розробленому методі кодування є кількість подібних фрагментів у базі даних. У роботі було проведено дослідження залежності якості декодованого зображення та часу кодування від кількості фрагментів, серед яких виконується пошук подібних.

У таблиці 4.3 наведено результати дослідження.

Таблиця 4.3 – Результати експериментів

Кількість фрагментів	*Час кодування, секунд*	*Вихідний розмір, байт*	*Відсоток стиснення, %*	*Подібність, SSIM*
20 000	15.55	3733	97.93	0.84
50 000	27.56	4069	97.74	0.86
100 000	44.67	4677	97.40	0.88
150 000	63.36	4905	97.28	0.89
200 000	82.83	5005	97.22	0.89

Результат декодування зображення при кількості фрагментів 20 000 та 200 000 зображено на рисунку 4.6 та рисунку 4.7 відповідно.

Рисунок 4.6 – Декодоване зображення при кількості 20 000 фрагментів

Рисунок 4.7 – Декодоване зображення при кількості 200 000 фрагментів

З рисунків вище можна побачити, що деталізація зображення краща у експерименті з більшою кількістю фрагментів, що підтверджується метрикою SSIM.

На рисунку 4.8 зображено графік залежності часу кодування від кількості фрагментів у базі.

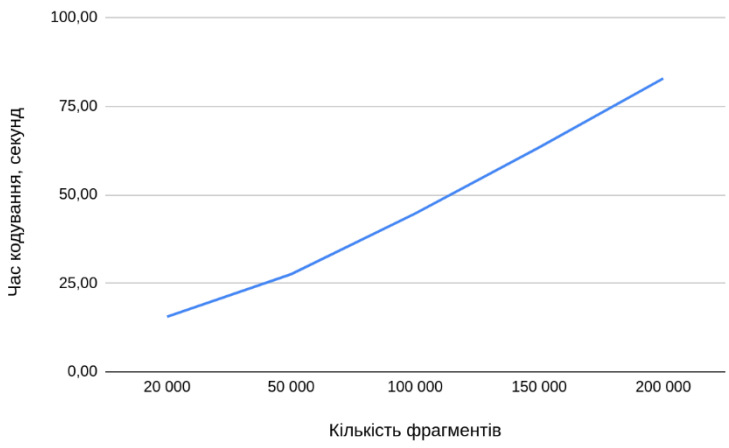


Рисунок 4.8 – Графік залежності часу кодування від кількості фрагментів

На рисунку 4.9 зображено графік залежності стиснення та подібності від кількості фрагментів.

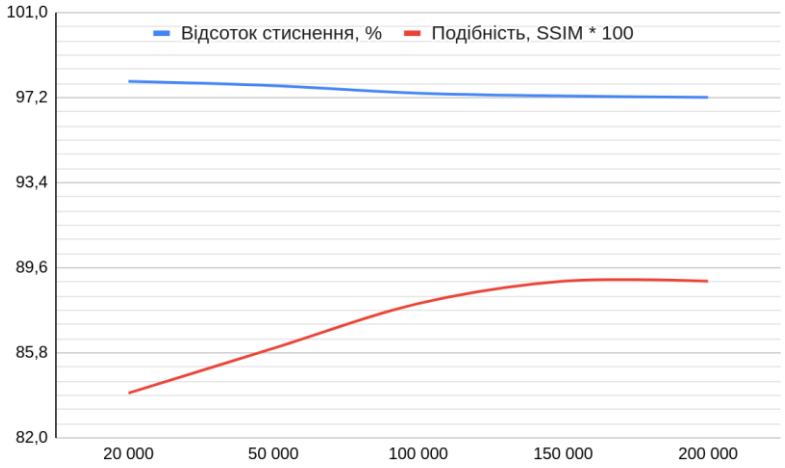


Рисунок 4.9 – Графік залежності стиснення та подібності від кількості фрагментів

З результатів експерименту бачимо, що зі збільшенням кількості фрагментів у базі час кодування стрімко зростає, відсоток стиснення незначно зменшується, а подібність зображення поступово збільшується. Варто зазначити, що після 150 000 фрагментів значення відсотку стиснення та подібності майже не змінюються, а от час кодування зростає.

Отже, при збільшенні кількості фрагментів у базі даних покращується якість декодованого зображення та в незначній мірі погіршується відсоток стиснення. Збільшення кількості фрагментів є ефективним до певного числа, після якого всі метрики не сильно змінюються, а час роботи методу кодування зростає.

4.6 Залежність критеріїв від кількості фрагментів вхідного зображення

Якість зображення, час кодування та розмір закодованих даних сильно залежать від того, на скільки фрагментів буде розділено вхідне зображення. Для підтвердження цього було проведено експеримент. У таблиці 4.4 наведено результати дослідження.

Таблиця 4.4 – Результати експериментів

Кількість фрагментів	*Час кодування, секунд*	*Вихідний розмір, байт*	*Відсоток стиснення, %*	*Подібність, SSIM*
96	3.52	333	99.82	0.53
120	4.55	381	99.79	0.55
150	5.65	437	99.76	0.58
200	7.75	537	99.70	0.62
500	20.34	1141	99.37	0.75
1000	43.50	2457	98.64	0.81
2000	96.61	5005	97.22	0.89

З таблиці бачимо, що при збільшенні кількості фрагментів збільшується час кодування, розмір закодованих даних, але також збільшується й подібність.

4.7 Залежність якості зображення від кількості не замінених фрагментів

Розроблений метод дозволяє збільшувати якість декодованого зображення за рахунок збереження початкових фрагментів вхідного зображення, якщо найбільш подібний до нього фрагмент має низьку подібність.

Межа подібності задається коефіцієнтом SSIM. Якщо коефіцієнт подібності знайденого в базі фрагменту є меншим за заданий, то фрагмент вхідного зображення зберігається в базу даних й не буде замінений в декодованому зображенні. Таким чином можна регулювати мінімальний рівень якості зображення.

У таблиці 4.5 показано залежність якості зображення та кількості не замінених фрагментів від межі коефіцієнту схожості. Вхідне зображення для експерименту було розділено на 2000 фрагментів.

Таблиця 4.5 – Залежність якості зображення та кількості не замінених фрагментів від межі

Межа коефіцієнту схожості	*Відсоток замінених фрагментів, %*	*Подібність, SSIM*
-1	0	0.886
0	0.15	0.887
0.4	2.5	0.903
0.6	6.05	0.918
0.7	9.95	0.929
0.75	13.0	0.935
0.8	16.85	0.943

На рисунках 4.10 та 4.11 показано декодовані зображення з межою коефіцієнту схожості 0.4 та 0.7 відповідно.

Рисунок 4.10 – Декодоване зображення з межею коефіцієнту схожості 0.4

Рисунок 4.11 – Декодоване зображення з межою коефіцієнту схожості 0.7

З результатів можна зробити висновок, що для досягнення коефіцієнту подібності SSIM більшого за 0.9 достатньо зберегти всього 2.5% фрагментів вхідного зображення. А збереження 10% вхідних фрагментів підвищить коефіцієнт подібності на 5%.

4.8 Порівняння розробленого методу з JPG і JPEG

Виконаємо порівняння результатів розробленого методу з найбільш популярними методами стиснення зображень JPG та JPEG. У таблиці 4.6 наведено результати порівняння. Вхідне зображення у форматі PNG показано на рисунку 4.12. Зображення має розмір 300 x 300 пікселі, а розмір даних 175 248 байт.

Рисунок 4.12 – Вхідне зображення у форматі PNG

Таблиця 4.6 – Порівняння з методами JPG та JPEG

Метод	*Вихідний розмір, байт*	*Відсоток стиснення, %*	*Подібність, SSIM*
Розроблений метод	11 613	93.37	0.85
JPG	40 890	76.67	0.98
JPEG	41 832	76.13	0.98

На рисунку 4.13 наведено зображення, яке стиснуте за допомогою методу JPG.

Рисунок 4.13 – Зображення, стиснуте методом JPG

На рисунку 4.14 наведено зображення, яке стиснуте за допомогою методу, який запропоновано в даній роботі.

Рисунок 4.14 – Зображення, стиснуте запропонованим методом

В результаті експериментів приходим до висновку , що розроблений метод має високий коефіцієнт подібності, хоча він і є меншим ніж у JPG та JPEG. Але запропонований у даній роботі метод має більший відсоток стиснення. Абсолютне значення розміру декодованого зображення є в 3.5 рази меншим за вихідний розмір даних JPG та в 3.6 рази меншим за вихідний розмір даних JPEG.

Висновки до розділу

У цьому розділі описано критерії, за якими проводиться оцінка ефективності розробленого методу: розмір закодованого зображення, відсоток стиснення зображення, якість декодованого зображення та час, який витрачається на кодування зображення.

Наведено дослідження моделей нейронних мереж і алгоритмів стиснення даних. В якості моделі для виділення ознак було обрано ResNet V2 50, а в якості алгоритму для стиснення масиву ідентифікаторів – lzma.

Проведено низку експериментів для перевірки роботи методу, в результаті яких було визначено, що якість зображення покращується при:

- збільшенні кількості фрагментів у базі даних, при цьому розмір закодованих даних майже не збільшується;
- збільшенні кількості фрагментів, на які розділяється вхідне зображення, але розмір закодованих даних збільшується;

- встановленні мінімальної межі подібності фрагментів, але при цьому потрібно зберігати вхідні фрагменти в базу даних.

Виконано порівняння методу з JPG і JPEG. Було визначено, що JPG і JPEG мають кращий показник подібності декодованих зображень – 0.98, проти 0.85 в розробленого методу, але запропонований метод краще стискає дані – 93.37% стиснення проти 76.67% та 76.13% у JPG та JPEG відповідно.

5 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проєкту

Призначенням проєкту є реалізація програмного забезпечення для кодування зображення використовуючи запропонований метод. Сутністю розробки є створення програмної бібліотеки, яка дозволить виконувати кодування зображення на основі подібності фрагментів.

Цільовою аудиторією в першу чергу є організації, що зберігають у своїх сховищах багато схожих зображень і яким необхідно зменшити витрати ресурсів на збереження нових зображень. Основною вигодою використання цього продукту є зменшення витрат на ресурси для збереження графічних даних.

У таблиці 5.1 наведено опис ідеї.

Таблиця 5.1 – Опис ідеї стартап-проєкту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користучача*
Програмне забезпечення для реалізації методу кодування зображень на основі подібності фрагментів	Впровадження програмного забезпечення в системи, в яких зберігається велика кількість зображень	Зменшення витрат на ресурси для збереження графічних даних

Для отримання цілісного уявлення про зміст ідеї, а також базові можливі потенційні ринки збуту, в межах яких потрібно шукати групи клієнтів, які б використовували даний продукт, опишемо переваги нашого продукту в таблиці 5.2.

Таблиця 5.2 – Переваги продукту

№	*Техніко-економічні характерис-тики ідеї*	*Продукція конкурентів*		*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій метод	*Формат JPEG*			
1.	Стиснення більше ніж на 90%	Наявна	Відсутня			S
2.	Високий коефіцієнт подібності декодованих зображень	Наявна	Наявна		N	
3.	Декодування з додаткових даних	Наявна	Відсутня	W		
4.	Регулювання якості декодованого зображення	Наявна	Наявна		N	

Як видно з наведеної вище таблиці, можна зазначити, що проєкт має свої переваги перед конкурентами, а саме – високий коефіцієнт стиснення даних. Недоліком запропонованої ідеї є потреба в додаткових даних для кодування зображень, але враховуючи потенційних користувачів це не повинно бути проблемою, так як вони вже матимуть базу зображень.

5.2 Технічний аудит ідеї проєкту

Таблиця 5.3 – Технологічна здійсненність ідеї проєкту

№	*Ідея проєкту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1	Стиснення більше ніж на 90%	Система розроблена на мові програмування Python 3.9	Наявна	Доступна
		Згорткові нейронна мережа ResNet V2 50	Наявна	Доступна
		Алгоритм стиснення lzma	Наявна	Доступна
2	Високий коефіцієнт подібності декодованих зображень	Бібліотека annoy	Наявна	Доступна
		Алгоритм машинного навчання kNN	Наявна	Доступна
3	Декодування з додаткових даних	База даних MongoDB	Наявна	Доступна
4	Регулювання якості декодованого зображення	Метрика подібності зображення SSIM	Наявна	Доступна

Висновок: технологічна реалізація продукту – можлива. Обрана технологія реалізації ідеї проєкту: технологія Python, згорткові нейронні мережі, kNN, метрику SSIM та базу даних MongoDB.

5.3 Аналіз ринкових можливостей стартап-проєкту

5.3.1 Аналіз попиту

У таблиці 5.4 наведено попередню характеристику ринку.

Таблиця 5.4 – Попередня характеристика ринку

№	*Показники стану ринку (найменування)*	*Характеристика*
1	Кількість головних гравців, од	1
2	Загальний обсяг продаж, грн/ум.од	Невідомо
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі або по ринку, %	Невідома

Висновок: враховуючи невелику кількість головних гравців по ринку, зростаючу динаміку ринку можна зробити висновок, що на даний момент, ринок для входження стартап-продукту є привабливим.

5.3.2 Визначення потенційних груп клієнтів

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проєкту

№	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
1	Зменшення розміру даних для збереження зображень	організації, яким потрібно зберігати багато графічної інформації	готові заплатити високу ціну, за продукт, що вирішує їх проблеми	Високий коефіцієнт стиснення зображень, регулювання якості декодованого зображення, масштабованість

Продовження таблиці 5.5

№	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
2	Отримання якісних декодованих зображень	користувачі, які завантажують зображення	необхідність у отриманні якісних зображень	Завантаження зображень в високій якості

5.3.3 Аналіз ринкового середовища

Проведемо аналіз ринкового середовища: складемо таблиці факторів, що сприяють ринковому впровадженню проєкту (таблиця 5.6), та факторів, що йому перешкоджають (таблиця 5.7). Фактори в таблиці подані в порядку зменшення значущості.

Таблиця 5.6 – Фактори загроз

№	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1	Команда розробки організацій	Складніший процес кодування та декодування зображень	Написання детальної документації
2	Відмова у співпраці	Організація відмовляється від співробітництва	Отримання коментарів щодо причини відмови, пропонування компромісу, введення змін

Таблиця 5.7 – Фактори можливостей

№	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1	Науково-технічний	Вдосконалення інформаційної системи	Впровадження в роботу
2	Інноваційний	Новий метод кодування зображень	Надання нових рішень у сфері
3	Економічний	Підтримка інновацій у виробництві	Підвищення / Пониження ціни на послугу

5.3.4 Аналіз пропозиції

Визначимо загальні риси конкуренції на ринку необхідні для аналізу пропозиції.

Таблиця 5.8 – Ступеневий аналіз конкуренції

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*
1. Тип конкуренції - чиста	В галузі є достатньо гравців, кожний з яких має свої сильні та слабкі сторони	Покращення існуючих послуг та впровадження інновацій
2. За рівнем конкурентної боротьби - міжнародний	Компанії конкуренти з різних країн	Максимальна локалізація продукту

Продовження таблиці 5.8

Особливості конкурентного середовища	*В чому проявляється дана характеристика*	*Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)*
3. За галузевою ознакою - глобальна	Продукт може використовуватись в будь-яких галузях	Інтеграція потреб різних галузей
4. Конкуренція за видами товарів: товарно-видова	Конкуренція між видами ПП, їх особливостями.	Вдосконалити ПП, враховуючи недоліки конкурентів
5. За характером конкурентних переваг - нецінова	Вдосконалення технології створення ПП, щоб собівартість була нижчою	Удосконалення моделі.
6. За інтенсивністю - марочна	Бренд присутній, але його роль незначна	Популяризація власної розробки шляхом участі в конференціях, семінарах

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
Складові аналізу	Метод збереження JPEG	Невідомі	-	Контроль якості продукту	Наявність функціоналу для вищого коефіцієнту стиснення зображень

Продовження таблиці 5.9

	Прямі конкуренти в галузі	*Потенційні конкуренти*	*Постачальники*	*Клієнти*	*Товари-замінники*
Висновки	Досить інтенсивна конкурентна боротьба з вже закріпленими на ринку гравцями	Дані відсутні	-	Клієнти диктують умови роботи на ринку: високий коефіцієнт стиснення, висока якість декодованого зображення	Необхідно випускати ПЗ краще, ніж у конкурентів.

Висновок: З проведеного аналіз у конкуренції на ринку нами було виявлено, що існує можливість виходу на ринок. Як стратегію на початку роботи можна обрати напрямок роботи на встановлення зв’язків з існуючими замовниками послуг.

5.3.6 Обґрунтування факторів конкурентоспроможності

Користуючись характеристиками ідей проєкту та вимогами споживачів визначимо перелік факторів конкурентоспроможності(таблиця 5.10).

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)*
1	Масштабовність	Масштабованість продукту дозволяє використовувати його в роботі з великими даними

Продовження таблиці 5.10

№	*Фактор конкурентоспроможності*	*Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проєктів значущим)*
2	Високий коефіцієнт стиснення даних	Порівняно високий коефіцієнт стиснення даних дозволяє економити місце в сховищі та витрати на ресурси.
3	Якість декодованих зображень	Висока подібність декодованих зображень дозволяє надавати якісні зображення для клієнтів.
4	Орієнтованість на кінцевого споживача	Індивідуальний підхід до кожного клієнта з метою мінімізувати його витрати.

5.3.7 Аналіз сильних та слабких сторін проєкту

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін

№ п/п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з CoreNLP*						
			–3	*–2*	*–1*	*0*	*+1*	*+2*	*+3*
1	Масштабовність	15			+				
2	Високий коефіцієнт стиснення даних	19		+					
3	Якість декодованих зображень	14					+		
4	Орієнтованість на кінцевого споживача	15					+		

5.3.8 SWOT-аналіз

Завершальним етапом ринкового аналізу можливостей реалізації проєкту є складання SWOT (матриця аналізу сили та слабкості), аналізу загроз (проблем) та можливостей (таблиця 5.12) на основі обраних ринкових загроз та можливостей, а також сильні та слабкі сторони (таблиця 5.11).

Перелік загроз та ринкових можливостей базується на аналізі загроз та факторів маркетингового середовища. Ринкові загрози та ринкові можливості - це наслідки (очікувані результати) впливу факторів і, навпаки, вони ще не реалізовані на ринку та мають певну ймовірність реалізації. Наприклад: падіння доходів потенційних споживачів - фактор загрози, на основі якого можна скласти прогноз підвищення значення цінового фактора у виборі товару і, отже, - цінової конкуренції (а це є - ринкова загроза).

Таблиця 5.12 – SWOT- аналіз стартап-проєкту

Сильні сторони (S): Високий коефіцієнт стиснення даних, масштабованість, якість декодованих зображень	Слабкі сторони (W): потреба в додаткових даних для кодування
Можливості (O): Новий метод кодування зображень	Загрози (T): відмова у співпраці

5.3.9 Альтернативи поведінки

На основі SWOT-аналізу ми розробимо альтернативи поведінці на ринку (перелік заходів) для виведення на ринок стартового проєкту та приблизний оптимальний час їх реалізації на ринку, враховуючи потенційних проєктів-конкурентів, які можна вивести на ринок. Визначені альтернативи аналізуються з точки зору часу та ймовірності отримання ресурсів.

Таблиця 5.13 – Альтернативи ринкової поведінки

№	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
1	Безкоштовне розповсюдження створеного продукту	Дуже висока	1 рік

Продовження таблиці 5.13

№	*Альтернатива (орієнтовний комплекс заходів) ринкової поведінки*	*Ймовірність отримання ресурсів*	*Строки реалізації*
2	Створення програмного продукту з подальшим розповсюдженням за певну оплату	Висока	1.5-3 роки

Отже, логічно обрати першу альтернативу та згодом розробити розширений функціонал, якщо підхід буде мати попит на ринку.

5.4 Розроблення ринкової стратегії проєкту

5.4.1 Опис цільових груп потенційних користувачів

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1	Організації, яким необхідно зберігати великі об’єми графічної інформації	Висока, оскільки продукт зменшить їх витрати на сховища даних	Високий	Середня, небагато компаній мають дуже великі сховища зображень	Просто, оскільки попит на продукт є

Продовження таблиці 5.14

№	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
2	Користувачі персональних комп'ютерів	Помірна	Помірний	Висока	Складно, оскільки мало користувачів ПК зможуть сформувати базу зображень
Які цільові групи обрано: 1					

Відповідно до проведеного аналізу можна зробити висновок, що підходящою цільовою групою для розповсюдження даного програмного продукту є організації, яким необхідно зберігати великі об'єми графічної інформації. Відповідно до стратегії охоплення ринку збуту товару обрано стратегію масового маркетингу.

5.4.2 Базова стратегія розвитку

Щоб почати роботу у вибраних сегментах ринку варто спочатку сформувати базову стратегію розвитку.

Таблиця 5.15 – Визначення базової стратегії розвитку

№	*Обрана альтернатива розвитку проєкту*	*Стратегія охоплення ринку*	*Ключові конкурентоспроможні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку**
1	Стратегія спеціалізації	Налагодження зв’язків з клієнтами, індивідуальна модифікація ПЗ під. потреби)	Висока якість та точність, ухил на довготривалі стосунки	Стратегія диференціації

Базовою стратегією оберемо стратегію диференціації – орієнтування на потреби користувача. Альтернативною до неї (у разі провалу) буде обрано стратегію спеціалізації – налаштування під окремий цільовий сегмент.

5.4.3 Вибір стратегії конкурентної поведінки

Наступним логічним кроком є обрання базової стратегії розвитку конкурентної поведінки.

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

№	*Чи є проєкт «першо-прохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки**
1	Ні	70% шукати нових клієнтів, 30% — переконувати інших клієнтів скористатись саме нашими послугами	Основна ціль компанії це розробка нового функціоналу для кодування зображень.	Стратегія заняття конкурентної ніші

5.4.4 Стратегія позиціонування

На основі потреб споживачів від вибраних сегментів до постачальника (стартап-компанії) та товару, а також обраної базової стратегії розвитку та стратегії конкурентної поведінки розробляється стратегія позиціонування, яка полягає у формуванні ринкової позиції (група асоціацій) за якою користувачі матимуть змогу безпомилково визначити бренд / проєкт.

Таблиця 5.17 – Визначення стратегії позиціонування

№	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проєкту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)*
1	Масштабов-ність	Стратегія диференціації	Можливість обробки великих даних	Відчуття надійності продукту

Продовження таблиці 5.17

№	*Вимоги до товару цільової аудиторії*	*Базова стратегія розвитку*	*Ключові конкурентоспроможні позиції власного стартап-проєкту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проєкту (три ключових)*
2	Високий коефіцієнт стиснення даних	Стратегія диференціації	Можливість економити ресурси й витрати для зберігання зображень	Практичність, економія витрат
3	Якість декодованих зображень	Стратегія диференціації	Продукт дозволяє отримати декодоване зображення у високій якості	Відчуття якості продукту

Відповідно до проведеного аналізу можна зробити висновок, що стартап-компанія вибирає як базову стратегію розвитку – стратегію диференціації, як базову стратегію конкурентної поведінки – стратегію заняття конкурентної ніші.

5.5 Розроблення маркетингової програми стартап-проєкту

5.5.1 Маркетингова концепція товару

Першим кроком є формування маркетингової концепції товару, який отримує споживач. Для цього нам потрібно узагальнити результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити*
1	Високий коефіцієнт стиснення зображень	Стиснення зображень за допомогою розробленого методу, що дозволяє економити місце в сховищі та витрати на підтримку сховища.	Більший коефіцієнт стиснення даних.
2	Якість декодованих зображень	Висока якість декодованого зображення.	Висока якість при завантаженні зображення зі сховища.

5.5.2 Маркетингова модель товару

Розробимо трирівневу маркетингову модель товару: уточнення ідеї товару та / або послуги, її фізичних складових, характеристик процесу її постачання.

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	*Сутність та складові*		
I. Товар за задумом	Програмне забезпечення для нового методу кодування зображень на основі подібності фрагментів.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Якість 2. Рівень стиснення даних 3. Безвідмовність 4. Собівартість	Нм Нм М М	Тл Тл Тх Вр
	Якість: стандарти - PEP 8 Тестування - unit test/ integration test		

Продовження таблиці 5.19

Рівні товару	*Сутність та складові*
II. Товар у реальному виконанні	Пакування програмна бібліотека
	Марка:
III. Товар із підкріпленням	До продажу - безкоштовна ліцензія сроком на 1 місяць
	Після продажу - безкоштовні оновлення та модифікація продукту. Допомога з налаштуванням роботи за окрему плату.
За рахунок чого потенційний товар буде захищено від копіювання: Патенту на продукт, унікальності деяких деталей та комерційної таємниці.	

5.5.3 Визначення меж встановлення ціни

Наступним кроком є визначення цінових меж, якими слід керуватися при визначенні ціни потенційного продукту (кінцева ціна визначається під час фінансово-економічного аналізу проєкту), що передбачає аналіз ціни на подібні або замінні товари, а також аналіз доходу цільової групи споживачів. Аналіз проводиться за допомогою експертного методу.

Таблиця 5.20 – Визначення меж встановлення ціни

№	*Рівень цін на товари-замінники*	*Рівень цін на товари-аналоги*	*Рівень доходів цільової групи споживачів*	*Верхня та нижня межі встановлення ціни на товар/послугу*
1	Відсутні	Відсутні	від 50 000 грн/місяць	Від 5 000 грн на місяць і вище. Верхня ціна договірна і залежить від потреб та обсягу використання.

5.5.4 Оптимальна система збуту

Наступним кроком є визначення оптимальної системи продажів, за якою приймається рішення: здійснювати власні продажі або залучати сторонніх посередників (власна або задіяна система продажів); вибір та обґрунтування оптимальної глибини каналу збуту; вибір та обґрунтування типу посередників.

Таблиця 5.21 – Формування системи збуту

№	*Специфіка закупівельної поведінки цільових клієнтів*	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
1	Пробний період, купівля ліцензії на використання	Контакти зі споживачами, формування попиту, досліди в області маркетингу	Канал нульового рівня (виробник безпосередньо продає товару клієнту)	Ліцензія на використання

5.5.5 Розроблення стратегії маркетингових комунікацій

Останньою складовою маркетингової програми є розробка концепції маркетингової комунікації, заснованої на заздалегідь обраній основі позиціонування, на визначених специфікаціях поведінки клієнтів.

Таблиця 5.22 – Концепція маркетингових комунікацій

№	*Специфіка поведінки цільових клієнтів*	*Канали комунікацій, якими користуються цільові клієнти*	*Ключові позиції, обрані для позиціонування*	*Завдання рекламного повідомлення*	*Концепція рекламного звернення*
1	Орієнтація на якісний застосунок	Соцмережі, пошта, конференції, сповіщення	Комплексний підхід	Повідомлення має переконати клієнта, що продукт дозволить економити йому кошти	“Новий підхід до збереження зображень”

Як результат було створено ринкову (маркетингову) програму, що включає в себе визначення ключових переваг концепції потенційного товару, опис моделі товару, визначення меж встановлення ціни, формування системи збуту та концепцію маркетингових комунікацій.

Висновки до розділу

У даному розділі описано стратегії та підходи з розробки стартап-проєкту, визначено наявність попиту, динаміку та рентабельність роботи ринку. Було визначено, що існує можливість ринкової комерціалізації проєкту.

Після розгляду потенційних груп клієнтів, бар’єри входження та загальний стан конкурентоспроможності проєкту було встановлено, що проєкт є перспективним.

Було розглянуто та обрано альтернативу впровадження стартап-проєкту та доведено доцільність імплементації проєкту.

ВИСНОВКИ

Дослідження присвячене пошуку нових шляхів зменшення розміру зображень для їх збереження, зокрема за допомогою методів кодування. Було запропоновано новий метод для кодування растрових зображень на основі подібності фрагментів.

На основі даних, які було отримано в процесі аналізу, сформульовано задачу розробки нового методу кодування зображень на основі подібності фрагментів, створити програмну бібліотеку для реалізації методу та дослідити ефективність.

Для розробки програмного забезпечення було використано мову програмування Python, бібліотеку для роботи з нейронними мережами tensorflow, бібліотеки для роботи із зображеннями opencv та scikit-image, а також базу даних MongoDB. Для проведення досліджень та експериментів було використано хмарний сервіс Google Colab.

Розроблений метод кодування растрових зображень розділяє вхідне зображення на фрагменти, для яких відбувається пошук подібних фрагментів з бази даних. Після чого вхідним фрагментам ставиться у відповідність ідентифікатор подібного фрагменту з бази, а далі набір ідентифікаторів додатково стискається.

Метод містить два процеси: наповнення бази даних фрагментами та безпосередньо процес кодування зображення. Кожен з процесів складається з етапів, які на початку є однаковими.

Виконано розробку програмної бібліотеки, яка реалізує запропонований метод. Вона містить функціонал як для роботи з базою даних, так і для роботи з набором зображень, які завантажені у файлову систему. Також реалізовано функціонал для декодування зображень.

Було проведено ряд експериментів для перевірки ефективності методу та порівняння з популярними методами кодування JPG і JPEG. В результаті експериментів було визначено від яких умов залежить якість та розмір декодованого зображення: кількість наявних фрагментів у базі даних, кількість фрагментів на які розділяється вхідне зображення та кількість не замінених

фрагментів. У порівнянні з стандартами JPG і JPEG виявлено невелике зменшення метрики SSIM, але розмір стиснених даних є меншим у більше ніж 3 рази.

Проведено маркетинговий аналіз стартап-проєкту. Наведено опис ідеї, її технологічний аудит, аналіз ринкових можливостей запуску проєкту. Також було розроблено ринкову й маркетингову стратегію стартап-проєкту.

Результати роботи над магістерською дисертацією апробовано на першій всеукраїнській науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології»(SoftTech-2021).

До наукової новизни одержаних у магістерській дисертації результатів відноситься новий метод для кодування растрових зображень для стиснення та зменшення розміру графічних даних, який використовує подібність фрагментів інших зображень.

Практична значимість одержаних результатів полягає у тому, що запропонований метод та розроблена програмна бібліотека дозволяють зменшити розмір даних растрових зображень за допомогою кодування зображення, що зменшує витрати на ресурси для зберігання графічних даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) 102 Flowers Diff Species DataSet [Електронний ресурс] / Naik Sheil. – 2016. – Режим доступу до ресурсу: https://www.kaggle.com/lenine/flower-102diffspecies-dataset.
2) Milano J. Compressed Image File Formats: JPEG, PNG, GIF, XBM, BMP / John Milano., 1999. – 288 с.
3) Burger W. Principles of Digital Image Processing: Advanced Methods (Undergraduate Topics in Computer Science) / Wilhelm Burger. – London: Springer, 2013. – 382 с.
4) Thakar V. Deep Learning with Python and OpenCV: A beginner's guide to perform smart image processing techniques using TensorFlow and Keras / Viral Thakar.
5) Розроблення стартап-проєкту [Електронний ресурс] : Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ : НТУУ «КПІ», 2016. – 28 с.
6) Pajankar A. Python 3 Image Processing: Learn Image Processing with Python 3, NumPy, Matplotlib, and Scikit-image / Ashwin Pajankar. – New Delhi: BPB PUBLICATIONS, 2019. – 187 с.
7) G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets.,” Neural Comput., vol. 18, no. 7, pp. 1527–54, 2006.[38] Y. Bengio, Learning Deep Architectures for AI, vol. 2, no. 1. 2009.
8) Rohan T. Convolutional Networks for everyone [Електронний ресурс] / Rohan Thomas. – 2018. – Режим доступу до ресурсу: https://medium.com/@rohanthomas.me/convolutional-networks-for-everyone-1d0699de1a9d
9) Erdem Isbilen. Image Similarity Detection in Action with Tensorflow 2.0 [Електронний ресурс] / Erdem Isbilen. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/image-similarity-detection-in-action-with-tensorflow-2-0-b8d9a78b2509.

10) Nikhil B. Fundamentals of Deep Learning: Designing Next-Generation Machine Intelligence Algorithms / Buduma Nikhil., 2017. – 277 с.

11) Лутц М. Изучаем Python / Марк Лутц. – Санкт-Петербург: Диалектика, 2019. – 832 с.

12) Jackson W. Digital Illustration Fundamentals: Vector, Raster, WaveForm, NewMedia with DICF, DAEF and ASNMF / Wallace Jackson., 2015. – 186 с.

13) Various Image Compression Techniques: Lossy and Lossless [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.researchgate.net/publication/303319054_Various_Image_Compression_Techniques_Lossy_and_Lossless.

14) Image Optimization, Compression Techniques [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://medium.com/plus-marketing/image-optimization-compression-techniques-b88de8e71b09.

15) Sehrawat P. Data Compression and Decompression using Huffman coding / Preeti Sehrawat., 2019. – 56 с.

16) LZW Data Compression [Електронний ресурс] // American Journal of Engineering Research (AJER). – 2014. – Режим доступу до ресурсу: https://www.ajer.org/papers/v3(2)/C0322226.pdf.

17) Image Compression by Wavelet Transform. [Електронний ресурс]. – 2001. – Режим доступу до ресурсу: https://dc.etsu.edu/cgi/viewcontent.cgi?article=1108&context=etd.

18) Acharya T. JPEG2000 Standard for Image Compression: Concepts, Algorithms and VLSI Architectures / T. Acharya, P. Tsai., 2005. – 296 с.

19) Ситник А. Ю. Обробка графічних зображень засобами фрактальної геометрії : 122 / Ситник Аким Юрійович – Київ, 2021.

20) Google Colab [Електронний ресурс] – Режим доступу до ресурсу: https://colab.research.google.com/.

21) Bradshaw S. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage / Shannon Bradshaw., 2019. – 514 с

22) Feature vectors of images with ResNet V2 50 [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://tfhub.dev/google/imagenet/resnet_v2_50/feature_vector/5.

23) ImageNet dataset [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://image-net.org/.

24) Datta P. All about Structural Similarity Index (SSIM) [Електронний ресурс] / Pranjal Datta. – 2020. – Режим доступу до ресурсу: https://medium.com/srm-mic/all-about-structural-similarity-index-ssim-theory-code-in-pytorch-6551b455541e.

25) TensorFlow Hub [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/hub.

26) Heinz M. All The Ways to Compress and Archive Files in Python [Електронний ресурс] / Martin Heinz. – 2021. – Режим доступу до ресурсу: https://towardsdatascience.com/all-the-ways-to-compress-and-archive-files-in-python-e8076ccedb4b.

ДОДАТОК А ПРОГРАМНИЙ КОД

```
import lzma

import cv2
import time
import glob

import numpy as np
import typing as t
import tensorflow as tf

from annoy import AnnoyIndex
from skimage.util.shape import view_as_blocks

from utils import ssim_metric

TF_MODEL_PATH = 'tf_models/resnet_v2_50/'


class Image:
    def __init__(self, img: np.array, name: str):
        self.img: np.array = img
        self.name: str = name

    def get_size(self):
        return self.img.shape


class Fragment:
    def __init__(
        self,
        _id: int,
        data: np.array,
        size: t.Tuple[int, int],
        features: np.array
    ):
        self.id = _id
        self.data = data
        self.size = size
        self.features = features

    def get_height(self):
        return self.size[0]

    def get_width(self):
```

```
        return self.size[1]


class ImageManager:
    def __init__(self, scale_percent=100, height=200, width=300):
        self.scale_percent = scale_percent
        self.height = height
        self.width = width

    @staticmethod
    def prepare_img(
        img: np.array,
        scale_percent: t.Optional[int] = 40,
        height: t.Optional[int] = None,
        width: t.Optional[int] = None
    ):
        if not (width and height):
            width = int(img.shape[1] * scale_percent / 100)
            height = int(img.shape[0] * scale_percent / 100)
        dsize = (width, height)

        img_resized = cv2.resize(img, dsize, interpolation=cv2.INTER_AREA)

        return img_resized

    @staticmethod
    def split_fragments(
        img: np.array, fragment_height: int, fragment_width: int
    ):
        fragments_2d = view_as_blocks(
            img, (fragment_height, fragment_width, 3)
        )
        n, m, fragment_height, fragment_width, depth = (
            fragments_2d.squeeze().shape
        )
        return fragments_2d.squeeze().reshape(
            (n * m, fragment_height, fragment_width, depth)
        )

    @staticmethod
    def get_image_from_fragments(fragments: np.array, row_len):
        rows = []
        row = []
        for i in range(1, len(fragments) + 1):
            row.append(fragments[i - 1])
            if i % row_len == 0:
```

```
                rows.append(np.hstack(row))
                row = []

        return np.vstack(rows)


class BaseImageEncoder:
    tf_model = tf.saved_model.load(TF_MODEL_PATH)

    @staticmethod
    def prepare_img_cnn(img: np.array) -> tf.Tensor:
        # Resizes to 224 x 224 x 3  tensor
        img = tf.image.resize_with_pad(img, 224, 224)
        # Converts the data type of uint8 to float32 by adding a new axis
        # img becomes 1 x 224 x 224 x 3 tensor with data type of float32
        # This is required for the mobilenet model we are using
        img = tf.image.convert_image_dtype(img, tf.float32)[tf.newaxis, ...]

        return img

    def get_image_feature_vectors(self, img: np.array):
        # Calculate the image feature vector of the img
        features = self.tf_model(img)
        # Remove single-dimensional entries from the 'features' array
        return np.squeeze(features)

    def encode(self, *args, **kwargs):
        pass

    def decode(self, *args, **kwargs):
        pass


class ImageEncoder(BaseImageEncoder):

    def __init__(
        self, trees_int: int = 10,
        tf_model=tf.saved_model.load(TF_MODEL_PATH)
    ):
        self.trees_int = trees_int
        self.tf_model = tf_model
        print(type(tf_model))

    def prepare_fragments_and_get_features(
        self, fragments: np.array, name: str
    ) -> np.array:
        features_folder = 'features'
```

```
    npy_filename = f'{name}.npy'
    file_path = f'{features_folder}/{npy_filename}'

    print(f'npy_filename: {npy_filename}')
    files = glob.glob(features_folder + '/*.npy')
    print(f'files: {files}')
    if file_path in files:
        print(f'load features from: {file_path}')
        return np.load(file_path)

    prepared_fragments_features = []

    i = 1
    total = len(fragments)
    start = time.time()
    total_time = 0

    for fragment in fragments:
        if i % 100 == 0:
            print(f'{i} / {total}')
            temp_time = time.time() - start
            total_time += temp_time
            print(f'time: {temp_time}')
            start = time.time()
        i += 1

        prepared_fragment = self.prepare_img_cnn(fragment)

        prepared_fragments_features.append(
            self.get_image_feature_vectors(prepared_fragment)
        )

    print(f'total time: {total_time}')
    prepared_fragments_features = np.array(prepared_fragments_features)
    np.save(file_path, prepared_fragments_features)
    return prepared_fragments_features

def build_tree(
    self, prepared_fragments_features, prepared_test_fragments_features
):
    dims = prepared_test_fragments_features.shape[1]
    trees_int = self.trees_int
    tree = AnnoyIndex(dims, metric='euclidean')

    i = 0
    for target_fragment in np.concatenate((
```

```
            prepared_test_fragments_features, prepared_fragments_features
        )):
            tree.add_item(i, target_fragment)
            i += 1
        tree.build(trees_int)
        return tree

    @staticmethod
    def find_most_similar_fragment_cnn(
        target_fragment_features_index, total_target, tree
    ):
        n_nearest_neighbors = total_target + 1
        not_include_indices = list(range(total_target))

        nearest_neighbors_indices = tree.get_nns_by_item(
            target_fragment_features_index, n_nearest_neighbors
        )

        for index in nearest_neighbors_indices:
            if index not in not_include_indices:
                nearest_neighbor_index = index
                break

        return nearest_neighbor_index - total_target

    def get_new_fragments_indices(
        self,
        prepared_test_fragments_features,
        prepared_fragments_features
    ):
        new_fragments_indices = []

        print('--- building tree ---')
        tree = self.build_tree(
            prepared_fragments_features, prepared_test_fragments_features
        )
        print('--- tree builded ---')
        print()
        print('---- getting new indices ----')

        total = len(prepared_test_fragments_features)
        i = 1
        start = time.time()
        total_time = 0

        for target_fragment_index in range(total):
```

```
            similar_fragment_index = self.find_most_similar_fragment_cnn(
                target_fragment_index, total, tree
            )
            if i % 100 == 0:
                print(f'{i} / {total}')
                temp_time = time.time() - start
                total_time += temp_time
                print(temp_time)
                start = time.time()

            i += 1

            new_fragments_indices.append(similar_fragment_index)

        print(f'total_time: {total_time}')

        return new_fragments_indices

    def encode(self, img_test, test_fragments, fragments) -> bytes:
        prepared_fragments_features = (
            self.prepare_fragments_and_get_features(fragments, name='train')
        )
        prepared_test_fragments_features = (
            self.prepare_fragments_and_get_features(test_fragments,
name='test')
        )

        new_fragments_indices = self.get_new_fragments_indices(
            prepared_test_fragments_features, prepared_fragments_features
        )
        new_fragments_indices = np.array(new_fragments_indices)

        img_test_height, img_test_width = img_test.shape[0], img_test.shape[1]
        fragment_height, fragment_width, _ = test_fragments[0].shape
        shape_to_resize = (
            int(img_test_height / fragment_height),
            int(img_test_width / fragment_width)
        )
        new_fragments_indices_reshaped = new_fragments_indices.reshape(
            shape_to_resize
        )
        print(f'new_fragments_indices_reshaped:
{new_fragments_indices_reshaped.shape}')

        return lzma.compress(new_fragments_indices_reshaped)
```

```
    def decode(self, compressed_indexes, fragments, test_fragments, manager:
ImageManager):
        indexes = lzma.decompress(compressed_indexes)
        indexes: np.array = np.frombuffer(indexes, dtype=np.dtype('int64'))
        row_len = 50
        fragments_for_decoding = [fragments[i] for i in indexes]

        final_fragments_cnn = []
        replaced_count = 0

        for i in range(len(test_fragments)):
            fragment = test_fragments[i]
            new_fragment = fragments_for_decoding[i]

            metric = ssim_metric(
                np.vstack(fragment), np.vstack(new_fragment)
            )
            if metric < 0.4:
                replaced_count += 1
                final_fragments_cnn.append(fragment)
            else:
                final_fragments_cnn.append(new_fragment)

        print(f'replaced_count: {replaced_count}')

        return manager.get_image_from_fragments(
            final_fragments_cnn, row_len=row_len
        )

    def get_new_fragments(
        self,
        test_fragments,
        fragments,
        prepared_test_fragments_features,
        prepared_fragments_features,
        similarity_threshold=0.6
    ):
        new_fragments_indices_cnn = self.get_new_fragments_indices(
            prepared_test_fragments_features,
            prepared_fragments_features,
        )

        new_fragments_cnn = np.array(
            [fragments[index] for index in new_fragments_indices_cnn]
        )
```

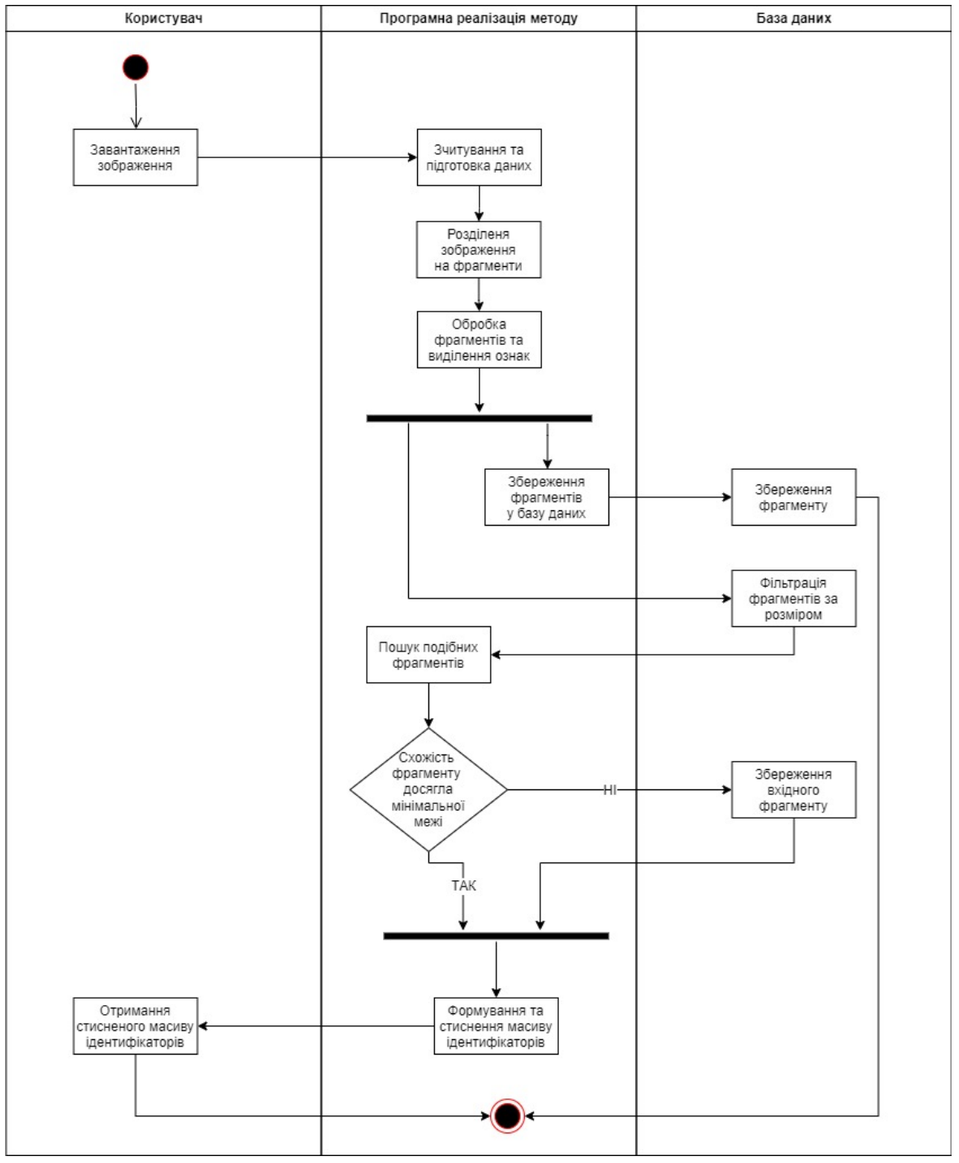
```
        final_fragments_cnn = []
        replaced_count = 0

        for i in range(len(test_fragments)):
            fragment = test_fragments[i]
            new_fragment = new_fragments_cnn[i]

            metric = self.ssim_metric(
                np.vstack(fragment), np.vstack(new_fragment)
            )
            if metric < similarity_threshold:
                replaced_count += 1
                final_fragments_cnn.append(fragment)
            else:
                final_fragments_cnn.append(new_fragment)

        print(f'replaced_count: {replaced_count}')

        return np.array(final_fragments_cnn)
```

```
import cv2
import numpy as np

from skimage import metrics

ssim = metrics.structural_similarity


def blur_image(img: np.array, blur_percent: int = 2) -> np.array:
    return cv2.blur(img, (blur_percent, blur_percent))


def ssim_metric(img_1: np.array, img_2: np.array) -> float:
    return ssim(img_1, img_2, multichannel=True)

def main() -> None:
    cv2.startWindowThread()

    manager = ImageManager()
    encoder = ImageEncoder()

    # ======== LOAD DATA ===========
    height = 200
    width = 300

    img_train_name = 'data/image_train.jpg'
    img_train = cv2.imread(img_train_name)
```

```
    img_train = manager.prepare_img(img_train, height=height, width=width)

    img_test_name = 'data/test_red.jpg'
    img_test = cv2.imread(img_test_name)
    img_test = manager.prepare_img(img_test, height=height, width=width)

    fragment_height = 5
    fragment_width = 6
    fragments = split_fragments(img_train, fragment_height, fragment_width)
    test_fragments = split_fragments(img_test, fragment_height, fragment_width)

    encoded_data = encoder.encode(img_test, test_fragments, fragments)
    decoded_img = encoder.decode(encoded_data, fragments, test_fragments,
manager)

    cv2.imshow('decoded_img', decoded_img)
    cv2.waitKey(0)

    score = ssim_metric(img_test, decoded_img)
    print(f'score: {score}')

    cv2.destroyAllWindows()
```

ДОДАТОК Б ГРАФІЧНІ МАТЕРІАЛИ

Діаграма діяльності для процесів наповнення бази даних та кодування зображень



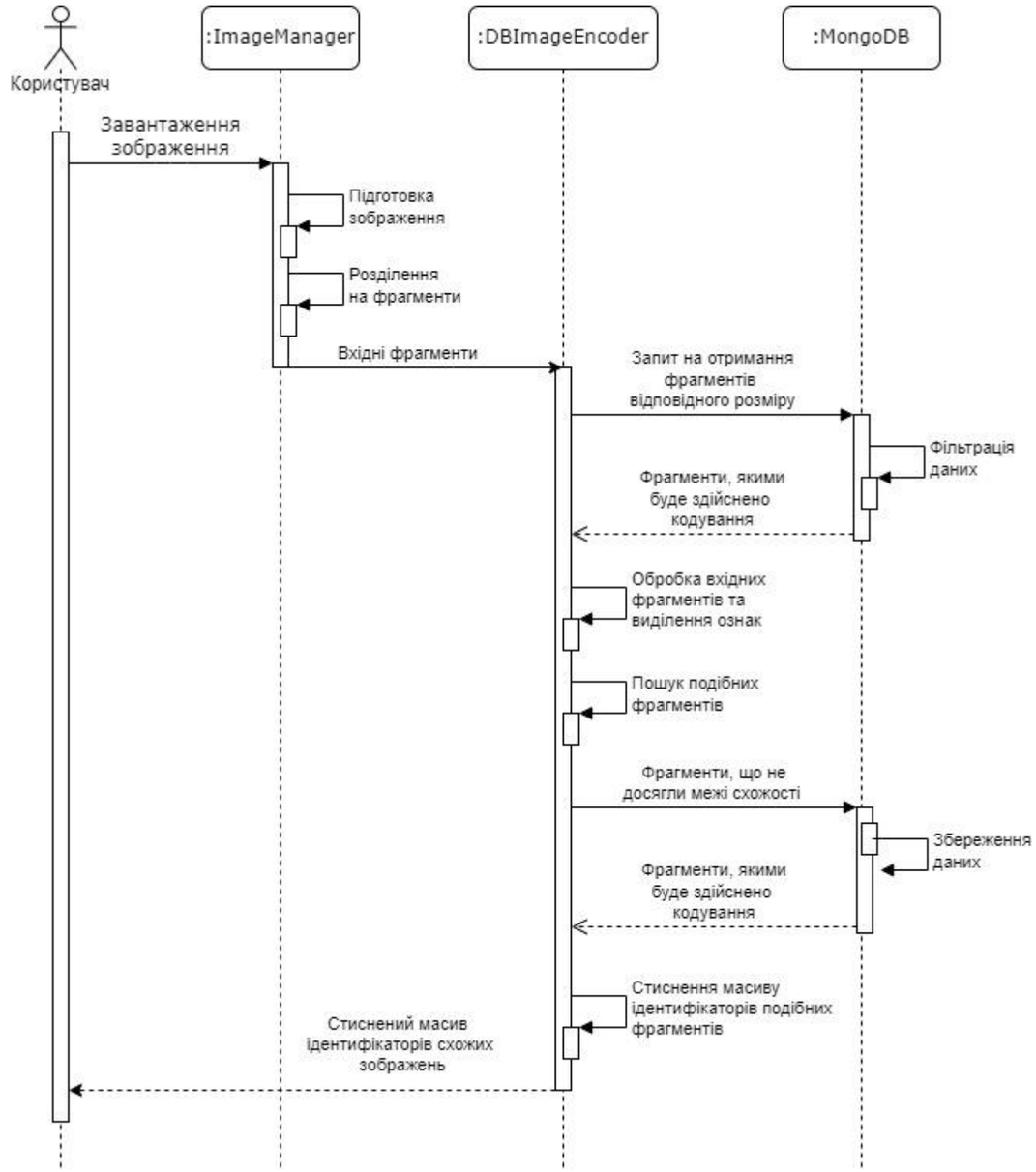
Демонстраційний плакат до магістерської дисертації

Діаграма діяльності для процесів наповнення бази даних та кодування зображення

Виконав студент гр. ІП-02мп Портяний І.С.

Керівник Олійник Ю.О.

Діаграма послідовності для процесу кодування зображення



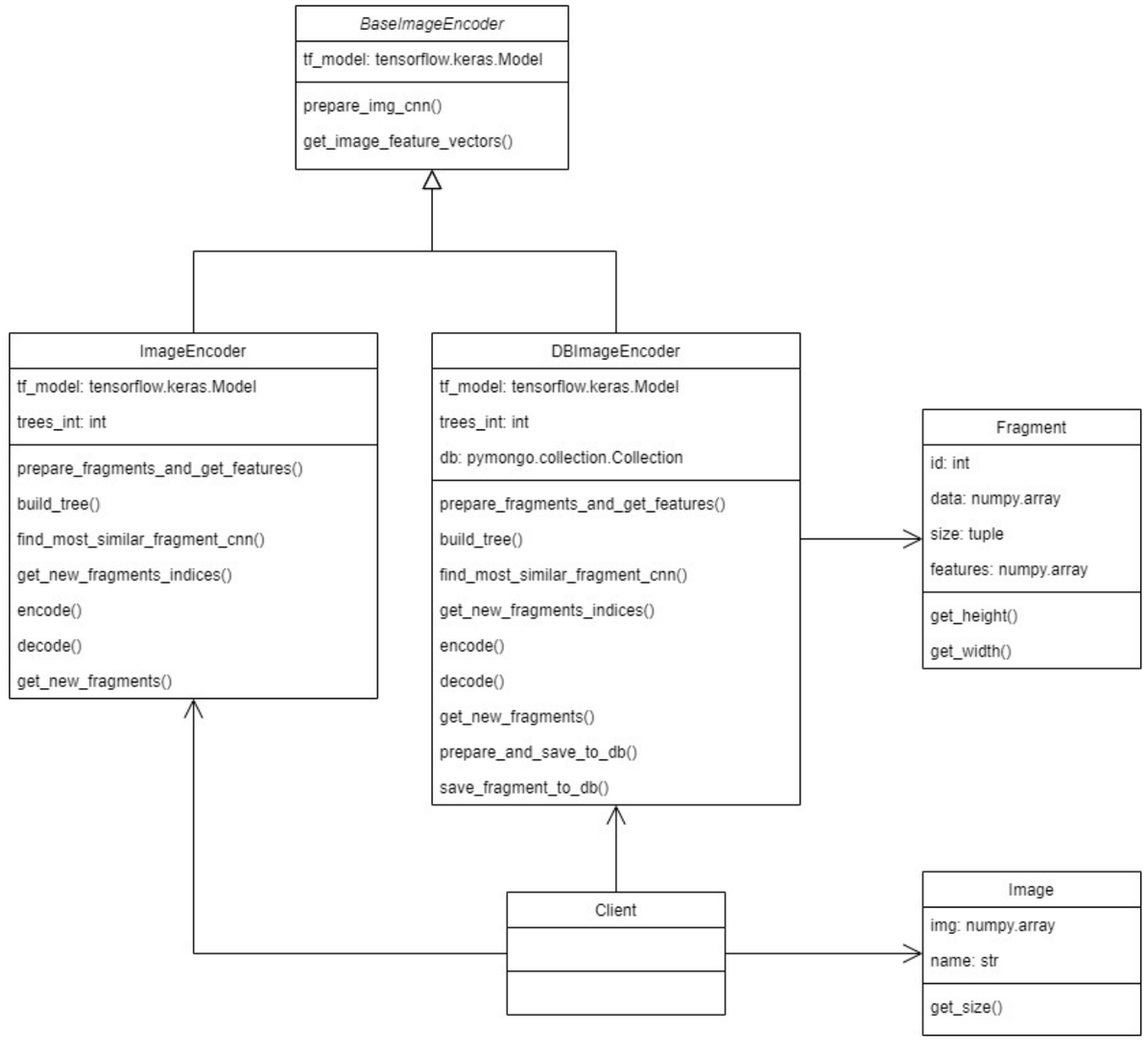
Демонстраційний плакат до магістерської дисертації

Діаграма послідовності для процесу кодування зображення

Виконав студент гр. ІП-02мп Портяний І.С.

Керівник Олійник Ю.О.

Діаграма класів



Демонстраційний плакат до магістерської дисертації
Діаграма класів

Виконав студент гр. ІП-02мп Портяний І.С.

Керівник Олійник Ю.О.

UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1009420317

Дата проверки:
30.11.2021 07:57:44 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
30.11.2021 07:58:23 EET

ID пользователя:
76913

Название файла: **ІП-02мп_Портяний_ПЗ_перевірка_плагіат**

Количество страниц: **46** Количество слов: **8416** Количество символов: **61695** Размер файла: **109.55 KB** ID файла: **1009437609**

1.76%
Совпадения

Наибольшее совпадение: **0.78%** с источником из Библиотеки (ID файла: **1008290499**)

0.33% Источники из Интернета	11	Страница 48
1.54% Источники из Библиотеки	28	Страница 48

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников