КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра математичної інформатики

Кваліфікаційна робота

на здобуття освітнього рівня бакалавра

за спеціальністю 122 Комп'ютерні науки

на тему:

АЛГОРИТМИ СТИСНЕННЯ ЗОБРАЖЕННЯ З ВИКОРИСТАННЯМ КЛАСТЕРИЗАЦІЇ

Виконав студент 4-го курсу

Владислав ГАСЛО _____ (підпис)

Науковий керівник:

професор, кандидат фіз.-мат. наук

Ірина ВЕРГУНОВА _____ (підпис)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____ (підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри математичної інформатики

«____» _____ 2023 р.,

протокол № ___

Завідувач кафедри

В. М. Терещенко _____ (підпис)

Київ – 2023

РЕФЕРАТ

Обсяг роботи 44 сторінки, 11 ілюстрацій, 14 джерел посилань.

ЧАТ-БОТ, ОБРОБКА ФОТО, СТИСНЕННЯ ЗОБРАЖЕННЯ, КЛАСТЕРИЗАЦІЯ, КВАНТУВАННЯ, КОЛІР, ЦЕНТРОЇД, ВАГА ЗОБРАЖЕННЯ, ПАЛІТРА.

Об’єктом роботи є процес стиснення зображення за допомогою зменшення кольорів через кластеризацію. Предметом роботи є чат-бот в месенджері Telegram для стиснення зображення.

Метою роботи є створення та розробка Telegram-бота для зменшення ваги зображень.

Інструмент розроблення: безкоштовний, вільно поширюваний редактор коду Visual Studio Code та модульне інтегроване середовище Eclipse, мови програмування Python та Java, хостинг Google Cloud, бібліотеки Scikit-image, NumPy, Python-telegram-bot.

Результат роботи: досліджено різні методи стиснення зображення через кластеризацію, визначено переваги та недоліки кожного методу, розроблено чат-бот, який буде виконувати алгоритми стиснення через зменшення кольорів у зображенні відповідно до параметрів користувача.

Розроблений чат-бот може використовуватися у різних напрямках роботи із зображеннями. Також буде корисним для стиснення зображень на пристроях, що мають доступ до месенджеру Telegram, коли надходить велика кількість зображень і їх потрібно зберігати.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

IDE – Integrated Design Environment, інтегроване середовище розробки;

PNG – Portable Network Graphics, портативна мережева графіка;

JPG – Joint Photographic Experts Group, об’єднана експертна група з фотографій;

TIFF – Tagged Image File Format, формат файлу зображень із тегами;

HEIF – High Efficiency Image File, високоефективний файл зображення;

RGB – Red, Green, Blue, червоний, зелений, синій;

ПАРКС – Паралельні Асинхронні Рекурсивно Керовані Системи.

ВСТУП

Оцінка сучасного стану об’єкта розробки. Кількість даних та їхній розмір постійно зростає. Збільшення даних стосується також і зображень, і окрім кількості, ростуть їхні вага та розмір. Зберігати велику кількість важких зображень вимагає багато дискового простору та значного часу на опрацювання.

Для ефективної роботи потрібно їх обробляти, наприклад, зменшуючи вагу файлів зображень. Цього можна досягнути різними способами. Зображення містять багато різних кольорів, але корисну інформацію можна отримати і маючи невелику їхню кількість. Щоб обрати колір, яким можна замінити якнайбільше кольорів зображення, вони розбиваються на групи і серед них обирається відповідний представник. Досліджувалися різні методи обрання таких кольорів та було впроваджено різноманітні алгоритми.

Актуальність роботи та підстави для її виконання. Так як вага файлів зображень збільшується, а використання додаткових носіїв інформації не завжди є доречним, то необхідно зменшувати вагу зображень. Запропоновано різні алгоритми для стиснення зображень та представлено їхнє використання через чат-бот. Можливо виконувати різні алгоритми просто надаючи зображення та бажані параметри і отримувати результат у вигляді стисненого зображення. Це дозволить отримати зображення меншої ваги із рівнем втрат, який задає користувач, тому можна стискати зображення, і обирати підходящий результат.

Мета й завдання роботи. Метою кваліфікаційної роботи є створення чат-боту для стиснення зображення з використанням кластеризації. Щоб досягти мети, були поставлені такі завдання:

- Дослідити алгоритми стиснення з використанням кластеризації.
- Реалізувати досліджені алгоритми.
- Реалізувати один з алгоритмів за допомогою системи ПАРКС.
- Розробити чат-бот для використання алгоритмів.

Об'єкт, методи й засоби розроблення. Об’єктом є чат-бот для стиснення зображення в месенджері Telegram. Для його створення були досліджені

властивості алгоритмів стиснення зображення з використанням кластеризації та створення чат-ботів. Реалізовувалися різні алгоритми для схожих форматів вхідних та вихідних даних, а потім вони були об’єднані за допомогою чат-боту. Для розробки використовувалися редактор коду Visual Studio Code та середовище розробки Eclipse для мов програмування Python та Java відповідно.

Можливі сфери застосування. Розроблений чат-бот може використовуватися у різних напрямках роботи із зображеннями. Його роль стає вагомою, коли вага зображення є важливішим фактором за різноманітність кольорів у ньому.

РОЗДІЛ 1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Стиснення зображення

Стиснення зображення – це процес зменшення розміру файлу зображення, зберігаючи при цьому прийнятну якість зображення. Зображення можуть бути стиснені з метою економії простору на диску, зниження часу передачі через мережу або для підвищення швидкодії обробки зображень.

Існує два основних типи стиснення зображень: стиснення без втрат та стиснення із втратами.

При стисненні зображень можна налаштувати рівень компресії або якості, щоб визначити, наскільки значно буде зменшено розмір файлу. Вибір оптимального рівня компресії залежить від конкретних потреб, а також від балансу між розміром файлу та якістю зображення.

1.1.1 Стиснення зображення без втрат

Безвтратне стиснення, є методом зменшення розміру файлу зображення, при якому не втрачається жодна інформація або якість зображення. Це означає, що після стиснення зображення можна відновити вихідну якість без будь-яких втрат або артефактів.

Деякі популярні формати безвтратного стиснення зображень включають:

– PNG є форматом зображень, який використовує безвтратне стиснення. Він використовує алгоритми для стиснення растрових зображень. PNG підтримує прозорість та інші додаткові можливості.

– TIFF є форматом безвтратного стиснення, який широко використовується в професійній фотографії та графічному дизайні. Він підтримує безліч параметрів, форматів кольору та може зберігати різноманітні метадані.

Перевагою безвтратного стиснення є те, що вихідне зображення можна відновити точно таким самим, як і перед стисненням. Однак, безвтратне стиснення зазвичай не досягає такої великої стисненості, як стиснення із втратами, тому розмір файлу може бути більшим в порівнянні з втратними методами.

1.1.2 Стиснення зображення із втратами

Стиснення зображень із втратами є одним з основних методів стиснення, який дозволяє значно зменшити розмір файлу зображення за рахунок видалення некритичних деталей та зміни точності представлення кольорів.

Популярні формати втратного стиснення зображень включають:

– JPEG є одним з найбільш використовуваних форматів для стиснення зображень із втратами. Він розкладає зображення на частотні компоненти та розділяє їх, видаляючи менш важливі деталі. Далі використовується алгоритм стиснення для подальшого зменшення розміру файлу.

– WebP є форматом зображень, розробленим Google, який підтримує втратне стиснення. Він використовує алгоритм стиснення, який застосовує різні методи, такі як прогресивне завантаження та просторова передача, щоб забезпечити високу стисненість при прийнятній якості зображення.

– HEIF є відносно новим форматом зображень, який підтримує втратне стиснення. Він використовує різні технології стиснення, такі як кодування по блоках та прогресивне завантаження, щоб забезпечити високу стисненість зображення.

Втратне стиснення зображень дозволяє досягти великого зменшення розміру файлу, але це може призвести до помітної втрати якості, особливо при високому ступені стиснення. При виборі рівня стиснення важливо знайти баланс між розміром файлу та збереженням достатньої якості. При роботі з втратними форматами важливо зберігати оригінальну копію зображення, оскільки втрачена якість не може бути повністю відновлена після стиснення.

1.2 Види стиснення із втратами

Існує кілька видів втратного стиснення, які використовуються для зменшення розміру файлу зображення за рахунок втрати певної якості. Ось декілька поширених видів втратного стиснення:

– Квантування є важливою частиною багатьох методів втратного стиснення. Воно використовується для обмеження точності представлення значень пікселів у зображенні. Вибір рівнів квантування впливає на якість та розмір стисненого зображення. Вищі рівні квантування призводять до більшої втрати якості та більшого стиснення.

– Стиснення на основі дискретної косинусної трансформації (DCT) для розкладу зображення на фрейми або блоки частот. Значення коефіцієнтів DCT квантується з використанням різних рівнів квантування, що призводить до втрати деякої інформації. Ці квантовані значення потім стискаються для подальшого зберігання.

– Стиснення засноване на хвильових перетвореннях використовує перетворення для розкладу зображення на набір хвильових компонентів різних масштабів та частот. Значення хвильових коефіцієнтів квантується, і тільки найбільш значущі коефіцієнти зберігаються, тоді як менш значущі коефіцієнти відкидаються. Цей процес також веде до втрати деякої інформації.

– Фрактальне стиснення є методом втратного стиснення зображень, який використовує фрактальну геометрію для подання та стиснення зображень. Ідея полягає в тому, що зображення розглядається як самоподібна структура, де окремі частини можуть бути подані як масштабовані версії інших частин. Однак, воно вимагає значних обчислювальних ресурсів для ефективного пошуку подібностей у зображенні та кодування цих подібностей.

Ці види втратного стиснення можуть застосовуватись окремо або комбіновано в різних методах стиснення зображень. Кожен метод має свої особливості та принципи роботи для досягнення оптимального балансу між якістю та розміром файлу.

1.3 Квантування

Квантування використовується для скорочення обсягу даних, збереження просторової та колірної роздільної здатності зображення за рахунок втрати деякої точності.

У контексті стиснення зображень, квантування означає зменшення числа можливих значень для пікселів або деяких компонент кольору в зображенні. Це досягається шляхом розділення діапазону можливих значень на кілька рівнів та приведення кожного значення до найближчого представника вибраного рівня.

Процес квантування може бути лінійним або нелінійним, в залежності від того, які рівні використовуються. Зазвичай використовуються рівні розташовані рівномірно або з певним розподілом, але також можуть використовуватися адаптивні рівні для забезпечення кращої точності для важливих ділянок зображення.

Після квантування значення пікселів зменшуються до представлення квантованих рівнів, що призводить до втрати деякої інформації і деталей в зображенні. Чим більше рівнів квантування використовуються, тим більша втрата якості зображення, але й менший обсяг файлу зображення.

Квантування є важливим етапом в різних методах стиснення зображень, таких як JPEG, де використовуються різні алгоритми квантування для різних компонент кольору з метою досягнення високого ступеня стиснення з втратою.

1.3.1 Уніформне квантування

Уніформне квантування є одним з основних методів квантування, який використовується в стисненні зображень. У цьому методі інтервал можливих значень розділяється на рівномірні підінтервали. Тобто для кольорових зображень кожний кольоровий канал ділиться на рівні частини. Кожний кольоровий канал кожного пікселю відносить його до певного регіону. Потім

знаходиться репрезентуючий колір регіону через середнє значення його елементів.

Далі по кожному пікселю у зображенні замінюються кольори відповідно репрезентативними з їхнього регіону.

1.3.2 Медіановий зріз

Ключовою ідеєю медіанового зрізу є рекурсивне сортування пікселів за кольоровими каналами та поділ їх по медіані.

Алгоритм можна записати у наступному узагальненому вигляді:

1. Занести всі пікселі в один список.
2. Знайти кольоровий канал із найбільшим розкидом.
3. Сортувати пікселі за значеннями кольорів у обраному каналі.
4. Знайти медіану і поділити регіон за цим пікселем.
5. Повторювати процес для утворених списків поки не буде досягнуто задану кількість кольорів.
6. У кінцевому списку знаходимо середні значення для кольорів та замінюємо ними кольори початкового зображення.

Оскільки з кожною ітерацією кількість списків подвоюється, то кількість результуючих кольорів буде степенем двійки.

1.3.3 Метод K-середніх

Метод К-середніх є одним з поширених алгоритмів кластеризації, який також може бути використаний для стиснення зображень. Також він є алгоритмом машинного навчання без учителя. Він допомагає групувати пікселі зображення в задану кількість кластерів залежно від їхньої подібності. Так як в кольорових зображеннях три кольорові канали, а саме червоний, зелений та синій, то будемо застосовувати кластеризацію для трьохвимірного простору, щоб знайти репрезентуючі кольори.

Спочатку задається кількість кластерів, вона ж буде відповідати кількості репрезентуючих кольорів для вихідного зображення.

Відстань обчислюється за допомогою Евклідової відстані в просторі кольорів:

$$d = \sqrt{(r_c - r)^2 + (g_c - g)^2 + (b_c - b)^2},$$

де r_c, g_c, b_c – кольори центроїда за різними кольоровими каналами;

r, g, b – кольори пікселя, з яким працюємо.

Алгоритм наведемо у наступному вигляді:

1. Випадковим чином обираються K початкових центрів кластерів.
2. Кожен піксель зображення призначається до кластеру за найменшою відстанню до його центру.
3. Після приналежності до кластеру кожного пікселя, обчислюються нові центри кластерів, шляхом обчислення середнього значення кольорів пікселів у кожному кластері.
4. Кроки 3 і 4 повторюються ітеративно, доки не досягнутий критерій зупинки, такий як збіжність або задана кількість ітерацій.
5. Після досягнення критерію зупинки кожен піксель зображення замінюється значенням центру кластеру, до якого він належить.

Результатом методу K-середніх є стиснене зображення, в якому кожен піксель замінений значенням центру відповідного кластеру.

1.3.4 Векторне квантування

Векторне квантування є ефективним методом для стиснення зображень. Воно базується на розділенні зображення на блоки і заміні цих блоків на найближчі вектори з кодової книги.

Алгоритм можна подати у такому вигляді:

1. Створення або вибір кодової книги, яка містить набір векторів. Ці вектори можуть бути отримані, наприклад, шляхом кластеризації пікселів

тренувального набору зображень і вибору центрів кластерів в якості векторів кодової книги.

2. Зображення розбивається на блоки фіксованого розміру, який задає користувач.
3. Кожен блок зображення порівнюється з векторами в кодовій книзі, і обирається найближчий вектор. Блок замінюється цим найближчим вектором.
4. Замість збереження всіх значень пікселів, зберігаються лише коди або індекси векторів кодової книги, які були використані для заміни блоків зображення.
5. Під час відтворення стислого зображення, блоки відновлюються, замінюючи їх збереженими векторами з кодової книги.

Векторне квантування дозволяє досягти значного стиснення зображень, зберігаючи важливу інформацію про зображення у вигляді кодів або індексів векторів.

1.3.5 Метод дерева октантів

Метод дерева октантів базується на структурі дерева. Дерево октантів є деревом, що має вісім дітей на вершині. Листки дерева не мають дітей, проте містять інформацію про колір та кількість пікселів з таким кольором.

Для додавання кольору до дерева необхідно кожен колір пікселя записати бінарно. Потім утворюється значення вершини наступного рівня через біти на поточній позиції. Ці значення будуть у двійковому записі від 000 до 111, тобто від 0 до 7 у десятковому записі, що відповідає кількості вершин дітей.

Наведемо приклад додавання кольору до дерева, нехай піксель матиме значення (90, 113, 157), у бінарному вигляді (01011010, 01110001, 10011101). На рис. 1.1 показано знаходження наступної вершини.

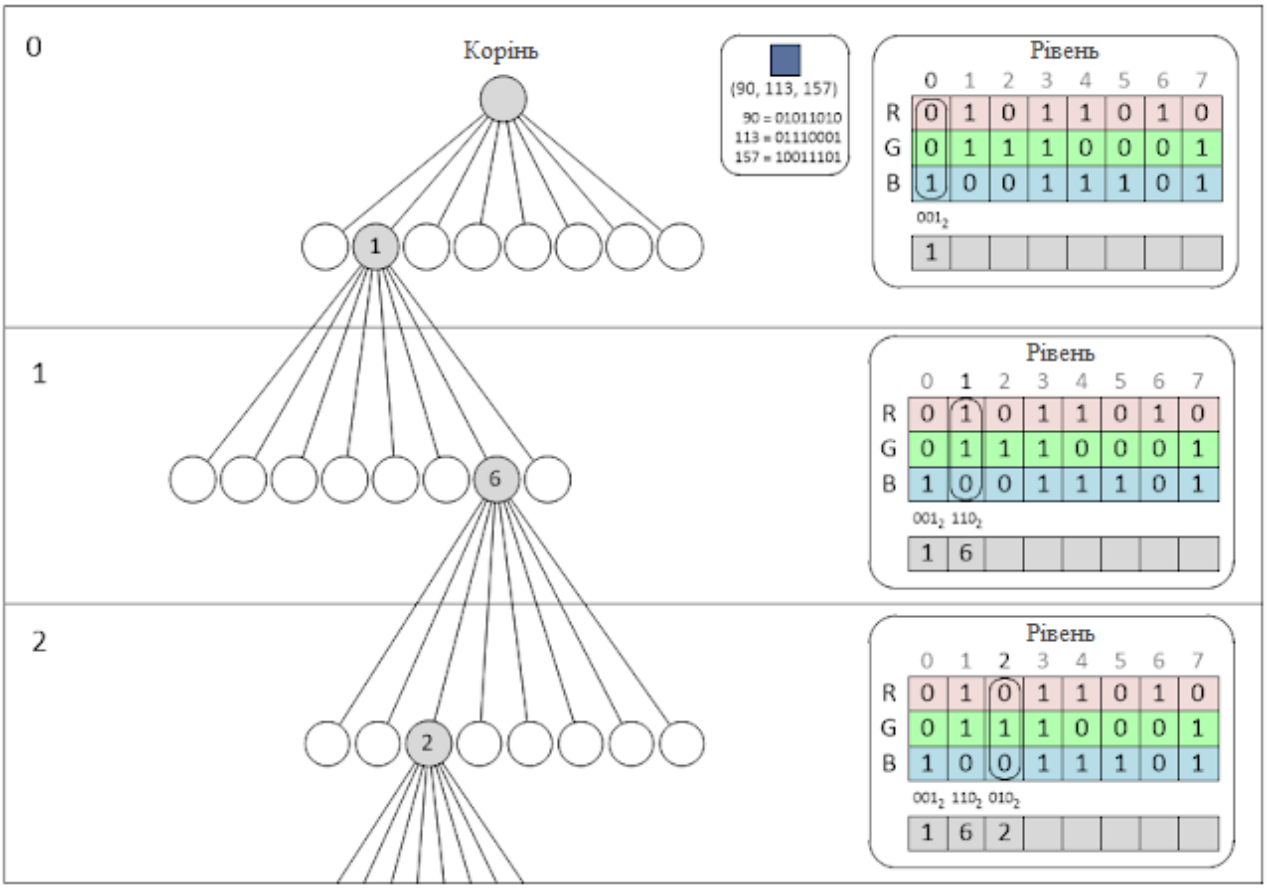

Рисунок 1.1 – Визначення вершини в дереві октантів

На рис. 1.2 показані індекси для всіх вершин.

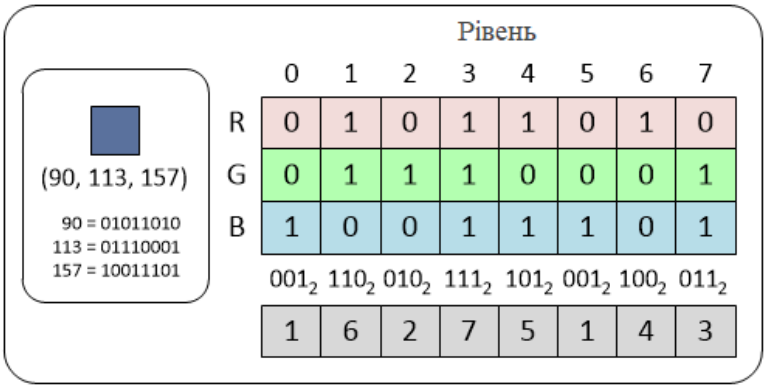

Рисунок 1.2 – Індекси в дереві октантів

На рис. 1.3 зображене повне дерево октантів для даного прикладу.

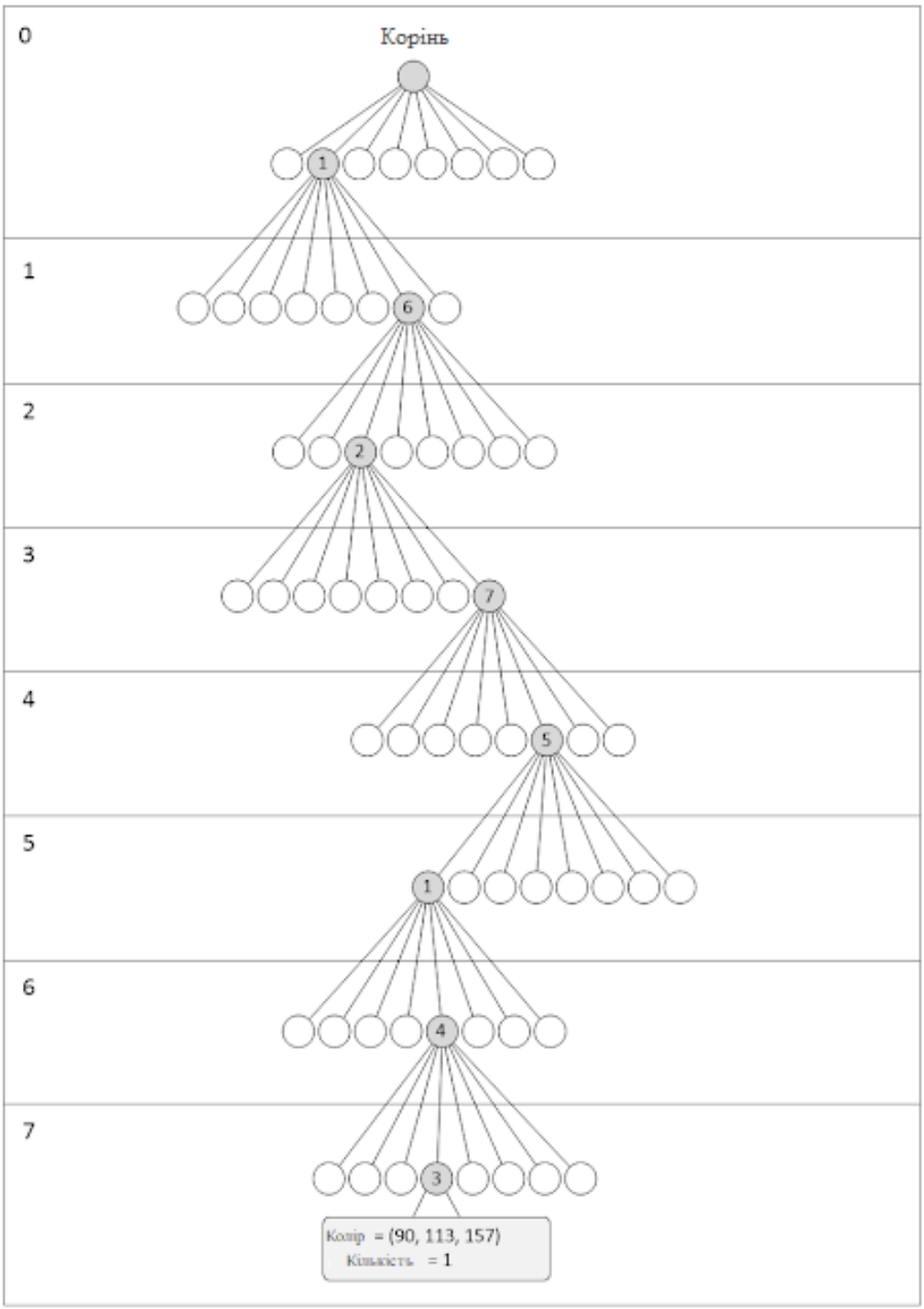

Рисунок 1.3 – Повне дерево октантів

На рис. 1.4 показано як змінюється дерево, коли до нього додають такий самий колір.

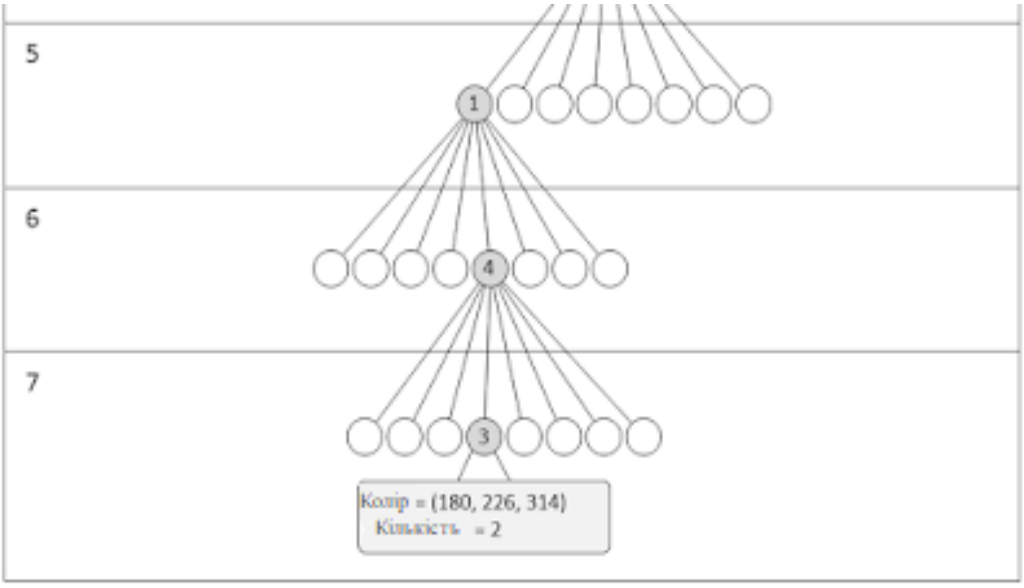

Рисунок 1.4 – Два однакових кольори в дереві октантів

Для зменшення кількості кольорів необхідно провести редукцію листків. Для цього додамо значення кольорів та кількості таких кольорів у батьківську вершину, цим самим зробимо її листком. Так продовжується до досягнення заданої кількості репрезентуючих кольорів. Далі залишається зробити палітру через середні значення кольорів у листках. Для всіх кольорових каналів суми кольорів діляться на їхні кількості, так отримуємо палітру.

Потім замінюються кольори вхідного зображення з використанням створеної палітри.

1.4 Порівняння алгоритмів

Кожен алгоритм можна ефективно використовувати в певних випадках. Так як деякі алгоритми мають точність, як параметр, а інші ні, то порівнювати їх за фактичними показниками часу виконання буде складно. Натомість наведемо приклади, коли алгоритми будуть ефективно працювати і коли будуть давати посередній результат.

Уніформне квантування легко реалізувати та не потребує складних операцій, проте у випадках, коли у зображенні переважають відтінки одного кольору, а інші кольори майже не використовуються, результат буде посередній.

Медіановий зріз також відносно простий алгоритм, але як і в попередньому випадку, якщо на зображенні переважають відтінки одного кольору, але найбільший розкид утворюється за іншим кольоровим каналом, то можливе обрання невідповідних репрезентуючих кольорів.

Метод К-середніх здатний створювати необхідну кількість результуючих кольорів та зупинятися після заданої кількості ітерацій, що дозволяє гнучко налаштовувати результати. Проте цей алгоритм складніший і потребує більше ресурсів. Для отримання гарних результатів необхідно багато разів проходити по зображенню, яке може бути досить великим. Також алгоритм залежить від початкових центроїдів, які обираються випадковим чином, що може призвести до втрати деталей в результуючому зображенні.

Векторне квантування оперує блоками, тому виконання відбувається відразу для певної сукупності пікселів. Присутня здатність контролювати точність та розміри блоку. Алгоритм є складнішим та з необхідністю правильного вибору параметрів. Блоки можуть між собою не пов’язуватися і результат не буде сприйматися як цілісний.

Метод дерева октантів здійснює ефективне збереження даних із високою швидкістю обробки. Також алгоритм дозволяє значно зменшити кількість кольорів із збереженням відповідної якості. Натомість алгоритм потребує багато пам’яті для збереження всієї інформації.

РОЗДІЛ 2 РЕАЛІЗАЦІЯ АЛГОРИТМІВ

Для реалізації алгоритмів необхідно обрати мову програмування, на якій вони будуть реалізовані, та бібліотеки з готовими рішеннями певних задач для полегшення написання.

2.1 Обрання мови програмування

Для написання алгоритмів буде використовуватися мова програмування Python.

Python – це високорівнева інтерпретована мова програмування з динамічною типізацією.

Плюси мови програмування Python:

– Простота та читабельність коду: Python має простий і зрозумілий синтаксис, що полегшує розробку та підтримку коду. Читабельність коду сприяє легкому розумінню програмного коду.

– Велика кількість бібліотек та модулів: Python має велику і активну спільноту розробників, що призводить до наявності великої кількості сторонніх бібліотек та модулів. Це дозволяє розробникам швидко вирішувати різноманітні задачі без необхідності писати код з нуля.

– Платформонезалежність: Python підтримується на багатьох операційних системах, таких як Windows, macOS, Linux і багатьох інших. Це дозволяє розробникам використовувати Python для розробки програм для різних платформ.

– Широке застосування: Python використовується в багатьох сферах, включаючи веб-розробку, наукові дослідження, аналіз даних, штучний інтелект, машинне навчання та багато іншого. Його універсальність дозволяє розробникам використовувати Python у різних проектах.

– Інтерактивність: Python підтримує інтерактивну роботу з інтерпретатором, що дозволяє розробникам експериментувати з кодом, тестувати фрагменти коду та швидко отримувати результати.

Мінуси мови програмування Python:

– Швидкодія: Python є інтерпретованою мовою програмування, що робить його менш ефективним в порівнянні з компільованими мовами, такими як C++ або Java. В деяких випадках, коли потрібна висока швидкодія, Python може бути менш підходящим варіантом.

– Пам'ять: Python може бути більш вимогливим до пам'яті порівняно з іншими мовами програмування. Це означає, що великі програми, особливо ті, які працюють з великими обсягами даних, можуть вимагати більше ресурсів для виконання.

– Обмеження в мобільній розробці: Хоча Python підтримується на платформах мобільних пристроїв, таких як Android та iOS, він не є основним вибором для мобільної розробки. Це означає, що інші мови програмування, такі як Java або Swift, можуть бути більш підходящими для розробки мобільних додатків.

– GIL (Global Interpreter Lock): Python має GIL, який обмежує виконання одновідповідних потоків Python на одному ядрі процесора. Це може стати перешкодою для ефективного використання багатопотокового програмування на багатоядерних системах.

Попри недоліки Python залишається зручною для користування мовою програмування.

2.2 Обрання середовища розробки

Середовищем розробки було обрано Visual Studio Code.

Visual Studio Code (VS Code) – це редактор вихідного коду, оптимізований для створення та налагодження сучасних хмарних та веб-програм.

Переваги Visual Studio Code:

– Кросплатформеність: VS Code підтримується на різних операційних системах, таких як Windows, macOS та Linux. Це дає можливість розробникам працювати в улюбленій операційній системі без необхідності переключатися на інший редактор.

– Розширюваність: VS Code має масивну екосистему розширень, яка дозволяє розширити його функціональність та адаптувати його до конкретних потреб розробки. Ви можете встановлювати розширення для підтримки різних мов програмування, інструментів, тем оформлення та багато іншого.

– Легкість використання та налаштування: VS Code має інтуїтивний і простий інтерфейс користувача, що полегшує його використання. Він також надає багато можливостей для налаштування, таких як налаштування тем оформлення, шрифтів, клавіатурних скорочень та інших параметрів.

– Розширена функціональність: VS Code має багатий набір функцій, таких як автодоповнення коду, підсвічування синтаксису, налагоджувач, контроль версій, вбудований термінал та багато іншого. Він також підтримує інтеграцію з різними інструментами розробки, такими як Git, Docker і розповсюджені системи контролю версій.

– Спільнота та підтримка: VS Code має велику та активну спільноту розробників, яка надає підтримку, навчальні матеріали та розширення. Ви можете знайти багато ресурсів та плагінів, які полегшують процес розробки та покращують продуктивність.

Недоліки Visual Studio Code:

– Вимоги до ресурсів: VS Code може вимагати значних ресурсів системи, особливо при роботі з великими проектами або використанні багатьох розширень. Це може призвести до збільшення використання пам'яті та потужності процесора.

– Не так потужний як повноцінні IDE: Хоча VS Code надає багато функцій та розширень, він все ж не є повноцінним IDE. Для деяких великих та складних проектів може бути більш підходящим використання інших IDE, які надають більш широкий набір інструментів та можливостей.

– Відсутність вбудованої підтримки деяких мов програмування: Хоча VS Code підтримує багато мов програмування, деякі менш популярні або екзотичні мови можуть мати обмежену або відсутню підтримку.

– Залежність від розширень сторонніх розробників: Використання розширень сторонніх розробників може бути потрібним для підтримки конкретних функцій або мов програмування. Однак, залежність від розширень може створювати проблеми з підтримкою та сумісністю в майбутньому.

Незважаючи на деякі недоліки, Visual Studio Code є потужним та популярним інструментом розробки з багатьма перевагами та можливостями, які роблять процес програмування зручним та продуктивним.

2.3 Бібліотеки для роботи алгоритмів

Для того, щоб виконувати алгоритми стиснення зображення, необхідно їх зчитати та перевести у матричний вигляд. Для кольорових зображень отримуватимемо трьохвимірний масив, елементами якого будуть значення кольорів кожного пікселя. Зовнішні шари відповідатимуть позиції пікселя у зображенні, а внутрішній матиме кортеж із трьох кольорів відповідно. Це допоможе зробити бібліотека Scikit-image для Python.

Так як зображення буде представлятись як масив, то для зручності використовуватиметься бібліотека NumPy, так як вона містить багато високорівневих математичних функцій для роботи з масивами.

2.3.1 Scikit-image

Scikit-image є бібліотекою Python, спеціалізованою на обробку та аналіз зображень.

Переваги scikit-image:

– Легкість використання: scikit-image має зрозуміле та просте API, що полегшує роботу з обробкою зображень. Вона надає високорівневі функції та

інструменти, що дозволяють швидко реалізувати різні завдання обробки зображень.

– Багатофункціональність: scikit-image містить велику кількість функцій та алгоритмів для обробки зображень, включаючи фільтрацію, сегментацію, розпізнавання об'єктів, вимірювання та багато іншого. Це дозволяє легко виконувати різноманітні завдання обробки зображень без необхідності писати багато власного коду.

– Інтеграція з іншими бібліотеками: scikit-image взаємодіє з іншими потужними бібліотеками Python, такими як NumPy, SciPy та matplotlib. Це дозволяє використовувати її разом з цими бібліотеками для обробки, аналізу та візуалізації зображень.

– Документація та підтримка: scikit-image має докладну документацію, яка надає приклади використання, пояснення алгоритмів та іншу корисну інформацію. Також існує активна спільнота розробників, яка надає підтримку, відповідає на запитання та приймає участь у вдосконаленні бібліотеки.

Недоліки scikit-image:

– Відсутність деяких спеціалізованих функцій: scikit-image не містить всіх можливих алгоритмів та функцій для обробки зображень, які можуть бути доступні в інших спеціалізованих бібліотеках. Для деяких конкретних завдань може бути необхідно шукати альтернативні рішення.

– Обмежена підтримка відео обробки: scikit-image зосереджена переважно на обробці статичних зображень, і вона має обмежену функціональність для роботи з відео. Для відео обробки можуть бути більш спеціалізовані альтернативи.

– Швидкодія: scikit-image, як і багато інших бібліотек Python, може мати обмеження у швидкодії порівняно зі спеціалізованими бібліотеками, написаними на низькорівневих мовах програмування, таких як C++.

Загалом, scikit-image є потужною та зручною бібліотекою для обробки зображень в середовищі Python. Вона має багато переваг та надає зручні інструменти для багатьох завдань обробки зображень.

2.3.2 NumPy

Numpy (Numerical Python) є потужною бібліотекою для обробки числових даних у Python.

Переваги numpy:

– Швидкодія: numpy використовує оптимізовані операції з масивами, що дозволяє прискорити обчислення. Вона реалізована на мові програмування C, що дозволяє використовувати низькорівневі оптимізації та використовувати векторизацію для швидкого виконання операцій з масивами.

– Масштабованість: numpy дозволяє працювати з багатовимірними масивами даних, включаючи вектори, матриці та більш складні структури даних. Вона надає потужні операції для маніпуляцій та обчислень з такими масивами.

– Універсальні функції: numpy має велику кількість універсальних функцій, які дозволяють виконувати різні операції на масивах, такі як обчислення математичних функцій, статистичних операцій, операцій лінійної алгебри та багато інших. Це полегшує роботу з числовими даними та обчисленнями.

– Інтеграція з іншими бібліотеками: numpy добре інтегрується з іншими популярними бібліотеками для наукових обчислень у Python, такими як SciPy, pandas та matplotlib. Це дозволяє створювати потужні та комплексні розрахункові рішення, використовуючи ці бібліотеки разом з numpy.

Недоліки numpy:

– Використання пам'яті: numpy може вимагати значну кількість пам'яті для збереження масивів даних, особливо для великих масштабних обчислень. Це може стати проблемою при роботі з об'ємними даними або обчисленнями, які вимагають великої точності.

– Відсутня гнучкість типів даних: numpy використовує фіксовані типи даних для масивів, що може призвести до обмежень при роботі з даними різних

типів. Наприклад, він не дозволяє зберігати рядки різної довжини в одному масиві.

– Складність навчання: для новачків в програмуванні чи тих, хто ще не мав досвіду роботи з масивами та операціями лінійної алгебри, вивчення numpy може бути викликом. Іноді складність синтаксису та концепцій може стати перешкодою для швидкого освоєння.

– Відсутність підтримки інших мов програмування: numpy є бібліотекою, спеціалізованою на Python, і не надає прямої підтримки для інших мов програмування. Це може бути проблемою, якщо вам потрібно інтегрувати numpy з програмами, написаними на інших мовах програмування.

Загалом, numpy є потужною та широко використовуваною бібліотекою для числових обчислень у Python. Вона має багато переваг, зокрема швидкодію та багатофункціональність.

2.4 Написання алгоритмів

Після налаштування середовища розробки йде написання коду алгоритмів. Кожен алгоритм винесений в окремий файл. Постійно використовується бібліотека NumPy і зображення обробляються в матричному вигляді. Результатом алгоритмів також будуть матриці. У деяких алгоритмах ще використовуються додаткові вбудовані бібліотеки Python.

2.4.1 UniformQuantization.py

Файл містить дві функції, а саме запуск алгоритму через Uniform_quantization() та допоміжна get_region_index(). Фіксується кількість можливих значень для кожного кольорового каналу, а саме 255. На вхід основної функції подаються зображення та кількість регіонів, на яку буде поділений кожний кольоровий канал. Допоміжна функція буде відносити колір, що опрацьовується до відповідної ділянки, повертаючи її індекс.

Далі, проходячи по всіх пікселях, значення кольорів заносяться у масив відповідного кольору і з обрахованим індексом. Для цих масивів знаходиться середнє, що і буде репрезентуючим. Потім копіюється вхідне зображення і в ньому замінюються значення кольорів на репрезентуючі. На вихід подається перетворене зображення.

2.4.2 MedianCut.py

У файлі є функції Median_cut(), median_cut_quantize() та split_into_buckets(). На вхід Median_cut() подається зображення та глибина поділу. Зображення перетворюється у масив, елементами якого є масиви із інформацією про значення кольорів та їхнє розташування. Глибина відповідає за кількість поділів, і так як кожен масив буде ділитись на дві частини, то результуюча кількість кольорів буде дорівнювати двійці в степені глибини.

Утворений масив передається функції split_into_buckets(), де шукається найбільший розкид значень. По цьому кольоровому каналі відбувається сортування та береться медіана. Потім викликається ця ж сама функція для масивів, розділених медіаною, та зменшується показник глибини.

Коли досягнуто заданої глибини, на утвореному масиві береться середнє значення кольорів, та ними замінюються пікселі вхідного зображення.

2.4.3 KMeans.py

Для алгоритму використовується вбудована бібліотека random. Файл містить 5 функцій. На вхід подається зображення, кількість кластерів та максимальна кількість ітерацій. Кількість кластерів відповідає за кількість результуючих кольорів. Максимальна кількість ітерацій є одним із факторів зупинки, якщо центроїди продовжують рухатися при досягненні її, то репрезентуючими обираються значення на поточній ітерації.

Спочатку знаходиться палітра через get_centroids(). За допомогою random випадковим чином обираються початкові центроїди у функції initialize_K_centroids(). Далі пікселі відносяться до центроїда через найменшу Евклідову відстань у find_closest_centroids(). Центроїди перераховуються у compute_means(). Так продовжується поки не досягнутий фактор зупинки. Потім записуються пікселі у відповідність до утворених кольорів через кольори вхідного зображення.

2.4.4 VectorQuantization.py

Для алгоритму використовувалися словники із вбудованої бібліотеки defaultdict. На вхід подаються зображення, розмір кодової книжки, поріг точності та вікно проходу. Розмір кодової книжки відповідатиме кількості, за яку результуючих ділянок має бути більше, так як при додаванні кодових векторів, вони будуть подвоюватися. Тому кількість кодових векторів у кодовій книжці буде найближчим значенням до степеня двійки, більшого чи рівного за вказаний параметр. Так як будуть опрацьовуватися не окремі пікселі, а цілі блоки, то на виході буде зображення складене із таких блоків. Поріг є фактором зупинки, коли спотворення стає меншим за значення порогу, то кодові вектори вважаються знайденими. Параметри вікна проходу відповідатимуть розмірам блоку, яким буде опрацьовуватися зображення. У ньому вказуються довжина, ширина та розмірність кольору. Для кольорових зображень, третім параметром буде три.

Так як зображення буде опрацьовуватись по блоках, то для поділу буде вилучено рядки і стовпчики, в які блок не поміститься. Далі зображення розбивається на такі блоки функцією generate_training(). Потім генерується кодова книжка через generate_codebook() використовуючи split_codebook() для збільшення кількості кодових векторів. На рис. 2.1 показані функції, які використовуються для обчислення необхідних параметрів.

```python
def avg_all_vectors(vecs, dim=None, size=None):
    size = size or len(vecs)
    nvec = np.array(vecs)
    nvec = nvec / size
    navg = np.sum(nvec, axis=0)
    return navg.tolist()

def new_codevector(c, e):
    nc = np.array(c)
    return (nc * (1.0 + e)).tolist()

def initial_avg_distortion(c0, data, size=None):
    size = size or _size_data
    nc = np.array(c0)
    nd = np.array(data)
    f = np.sum(((nc-nd)**2)/size)
    return f

def avg_codevector_dist(c_list, data, size=None):
    size = size or _size_data
    nc = np.array(c_list)
    nd = np.array(data)
    f = np.sum(((nc-nd)**2)/size)
    return f
```

```python
def get_mse(a, b):
    na = np.array(a)
    nb = np.array(b)
    return np.sum((na-nb)**2)

def generate_training(img, block):
    train_vec = []
    x = block[0]
    y = block[1]
    for i in range(0, img.shape[0], y):
        for j in range(0, img.shape[1], x):
            train_vec.append(img[i:i + y, j:j + x].reshape((x * y * block[2])))
    return (np.array(train_vec))

def distance(a, b):
    return np.mean((np.subtract(a, b) ** 2))

def closest_match(src, cb):
    c = np.zeros((cb.shape[0],))
    for i in range(0, cb.shape[0]):
        c[i] = distance(src, cb[i])
    minimum = np.argmin(c, axis=0)
    return minimum
```

Рисунок 2.1 – Додаткові функції векторного квантування

Відбувається пошук відповідності блоків із зображення із кодовою книжкою і присвоєння їхніх індексів в encode_image(). На останок за індексами вставляються кольори зображення у decode_image().

2.4.5 OCTree.py

У файлі реалізовано три класи: Color, OctreeNode та OctreeQuantizer. Color містить значення кольорів. OctreeNode має інформацію про вершину та методи для роботи з нею. OctreeQuantizer має інформацію про дерево та методи, щоб отримати палітру.

Спочатку проходяться всі пікселі і до дерева додаються всі кольори через методи add_color() у відповідних класах. Далі робиться палітра репрезентуючих кольорів методом make_palette() через редукцію. Маючи палітру на копії вхідного зображення замінюються кольори і отримується результуюче зображення.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ АЛГОРИТМУ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ ПАРКС

Алгоритми стиснення зображення повинні робити багато операцій на великій кількості даних. Розмірності зображень стають все більшими, а працювати необхідно з кожним пікселем, тому витрачається все більше часу на опрацювання. Якщо все робити послідовно, то і часу буде затрачено відповідно, але можливі різні оптимізації.

Одною з таких є розподілені обчислення. Розділяючи зображення на частини та опрацьовуючи їх окремо можна досягнути швидшого виконання. Для реалізації паралельних обчислень було обрано технологію ПАРКС, через поєднання змістовної наукової бази та сучасних інформаційних технологій. Вони дозволять об’єднати розрізнені обчислювальні ресурси та використати їх для вирішення складних обчислювальних задач.

3.1 Обрання мови програмування для технології ПАРКС

Технологія ПАРКС реалізована на декількох мовах програмування, а саме: Python, Java, C#. Інша частина роботи реалізована на Python, проте ПАРКС на Python використовує технології map-reduce та запускається через веб-інтерфейс. ПАРКС на Java має всі аспекти реалізовані безпосередньо. Присутній керуючий простір із алгоритмічними модулями, точками та каналами. Запускатися програма буде через термінал та результати виводити відразу у середовище. Це зручно для поставленої задачі, тому була обрана мова програмування Java.

Також можна вказати на такі переваги мови програмування Java:

– Переносимість: Java розроблена з урахуванням переносимості, що означає, що програми, написані на Java, можуть працювати на різних платформах, таких як Windows, macOS, Linux та інші, без потреби перекомпілювати їх для кожної платформи.

– Об'єктно-орієнтована природа: Java базується на об'єктно-орієнтованому підході, що дозволяє створювати модульні, повторно використовувані та легко зрозумілі програми. Об'єктно-орієнтований підхід дозволяє розробникам організовувати код у вигляді класів та об'єктів, що сприяє полегшенню розробки та підтримці програмного забезпечення.

– Велика екосистема та бібліотеки: Java має велику та активну спільноту розробників, що призводить до наявності великої кількості сторонніх бібліотек, фреймворків та інструментів. Це дозволяє розробникам використовувати готові рішення для швидкого розробки програм, зменшуючи час та зусилля, необхідні для розробки від нуля.

– Висока продуктивність: Java відома своєю високою швидкодією та ефективністю виконання. Код, написаний на Java, компілюється в байт-код, який потім виконується на віртуальній машині Java (JVM). Це дозволяє досягти високої продуктивності програм та ефективного використання ресурсів.

Недоліки мови програмування Java:

– Великий обсяг коду: Java відома своїм великим обсягом коду, що може призводити до більшої кількості написаного коду порівняно з іншими мовами програмування. Це може впливати на продуктивність розробки та читабельність коду.

– Висока вибагливість до ресурсів: Запуск програм на віртуальній машині Java (JVM) може вимагати більшого обсягу оперативної пам'яті та процесорної потужності порівняно з іншими мовами програмування. Це може бути обмеженням для вбудованих систем або пристроїв з обмеженими ресурсами.

– Відсутність підтримки низькорівневого програмування: Java, в основному, спрямована на високорівневу розробку програмного забезпечення, тому вона не підходить для завдань, які вимагають прямого доступу до апаратних ресурсів або низькорівневого програмування.

– Повільний запуск програм: Запуск програм на віртуальній машині Java може займати більше часу порівняно з виконанням нативних програм. Це може бути проблемою для деяких вимогливих до продуктивності додатків.

3.2 Обрання середовища розробки для технології ПАРКС

Так як код буде писатися на Java, то відповідно було обране середовище розробки, в якому буде зручно працювати, а саме IDE Eclipse.

Переваги IDE Eclipse:

– Багатофункціональність: Eclipse є повноцінним інтегрованим середовищем розробки, яке підтримує широкий спектр мов програмування, включаючи Java, C/C++, Python, PHP та інші. Воно надає розробникам доступ до багатьох інструментів, функцій рефакторингу, налагоджувача та плагінів, що полегшують розробку програмного забезпечення.

– Кросплатформовість: Eclipse підтримується на різних операційних системах, включаючи Windows, macOS і Linux. Це дозволяє розробникам працювати відповідно до своїх вподобань та використовувати одну і ту саму IDE на різних платформах.

– Розширюваність: Eclipse має потужну систему плагінів, яка дозволяє розробникам розширювати функціональність IDE за допомогою сторонніх плагінів. Це дозволяє налаштовувати робоче середовище для власних потреб та використовувати різноманітні інструменти, бібліотеки та фреймворки.

– Спільнота та підтримка: Eclipse має велику спільноту розробників, що означає, що ви можете знайти багато документації, ресурсів та форумів підтримки для вирішення проблем та отримання порад щодо розробки.

Недоліки IDE Eclipse:

– Великий обсяг ресурсів: Eclipse працює з великими обсягами ресурсів. Воно може бути важким для менш потужних комп'ютерів або проектів з обмеженими ресурсами.

– Складність налаштування: Хоча Eclipse є потужним інструментом, його також може бути складно налаштувати. Конфігурація проектів та додаткових інструментів може вимагати певного часу та зусиль.

– Швидкість роботи: Деякі розробники вказують на те, що Eclipse може бути повільним у порівнянні з іншими IDE. Відкриття проектів та виконання деяких операцій може займати більше часу.

– Великий обсяг вивчення: Оскільки Eclipse є потужним інструментом з багатьма функціями, він також має великий обсяг вивчення. Для новачків вивчення всіх можливостей та опцій може бути викликом.

3.3 Написання алгоритму за допомогою технології ПАРКС

Так як інша частина роботи написана на Python, то запуск алгоритму відбуватиметься через термінал, де буде виконуватись Makefile. На Java написано два файли, один для роботи із вхідним і вихідним зображенням та керуючим простором, інший для розподіленого опрацювання.

Для реалізації був обраний алгоритм K-середніх. В основному файлі підключаються бібліотеки parcs та для роботи з графікою, зображеннями та масивами, у алгоритмічному файлі підключається бібліотека parcs та випадкової генерації. Із середовища зчитується зображення та перетворюється на масив кольорів. Кольори опрацьовуються у двійковому вигляді із зсувами.

Зображення ділиться навпіл. Створюється керуючий простір із двома точками та каналами, яким передається половина зображення. Далі паралельно обраховуються центроїди на кожній частині, на одній такій визначається 16 центроїдів. Всього результатом буде 32 кольори, які повертаються каналами після обрахунку. На рис. 3.1 показано керуючий простір.

```
point p1 = info.createPoint();
channel c1 = p1.createChannel();
p1.execute("KMeans");
c1.write(rgb1);

point p2 = info.createPoint();
channel c2 = p2.createChannel();
p2.execute("KMeans");
c2.write(rgb2);

System.out.println("Waiting for result...");
int[] k_values1 = (int[]) c1.readObject();
int[] k_values2 = (int[]) c2.readObject();
int[] k_values = new int[k_values1.length + k_values2.length];
```

Рисунок 3.1 – Керуючий простір

Маючи результуючі кольори, замінюються кольори вхідного зображення. Створюється файл зображення та поміщається в середовище, де його зчитає операція на Python.

РОЗДІЛ 4 РЕАЛІЗАЦІЯ ЧАТ-БОТА

Для зручного використання написаних алгоритмів необхідно створити інтерфейс. Багато різних комунікацій відбувається за допомогою месенджерів, тому у них є значна кількість користувачів. З цих міркувань оберемо месенджер Telegram. В ньому можливо створювати чат-боти, якими буде нескладно користуватися. Розробимо один такий та налаштуємо виконання алгоритмів на ньому.

4.1 BotFather

Спочатку необхідно створити сам чат-бот. Для цього в Telegram необхідно знайти бота BotFather. Далі почати діалог командою /start, що надасть список команд, інструкції та посилання на посібник користувача. Далі записується команда на створення нового бота та згідно з інструкціями вказується його назва та посилання. Після цього бот видає токен, який можна використовувати вже для власного бота.

4.2 Обрання бібліотеки для роботи з Telegram

Далі необхідно використовуючи отриманий токен написати самого бота. Для цього необхідно обрати бібліотеку роботи з Telegram на Python. Було обрано Python-telegram-bot.

Переваги Python-telegram-bot:

– Простота використання: Python-telegram-bot надає простий та зрозумілий інтерфейс для створення та керування телеграм-ботами. Це робить його легким у використанні для початківців та досвідчених розробників.

– Багатофункціональність: Python-telegram-bot надає багато функцій та можливостей для створення різноманітних ботів. Ви можете використовувати його для створення текстових ботів, ботів з клавіатурою, ботів для обробки

мультимедіа, використовувати API Telegram для отримання даних, керувати груповими чатами та багато іншого.

– Багата документація та активна спільнота: Python-telegram-bot має добре документований API та активну спільноту розробників. Це означає, що ви можете знайти багато ресурсів, прикладів та підтримки в разі потреби.

– Підтримка асинхронності: Python-telegram-bot підтримує асинхронну розробку з використанням бібліотеки asyncio, що дозволяє створювати ботів зі здатністю обробляти багато запитів одночасно та ефективно використовувати ресурси.

Недоліки Python-telegram-bot:

– Обмежена швидкодія: Python-telegram-bot може бути повільнішим у порівнянні з деякими іншими мовами програмування, особливо при обробці великого потоку запитів або великих обсягів даних. Це може бути проблемою для додатків, які вимагають високої швидкодії обробки.

– Відсутність вбудованого сервера: Python-telegram-bot не має вбудованого сервера, тому вам потрібно буде налаштувати власний веб-сервер для розгортання та запуску бота. Це може бути викликом для тих, хто не має досвіду роботи з веб-серверами.

– Обмеження API Telegram: Python-telegram-bot підтримує API Telegram і обмеження, які накладені самим Telegram. Наприклад, є обмеження на кількість повідомлень, які можна відправити за один раз, а також обмеження на кількість запитів до API. Ці обмеження можуть вплинути на функціональність та продуктивність бота.

4.3 Написання бота

Необхідно написати бота, який би запускав алгоритми та виводив результати в Telegram. Код до бота написаний у файлі Main.py. Спочатку помістимо токен у файл конфігурації та підключимо його до нашого основного

файлу. Окрім бібліотеки для роботи з Telegram також знадобиться scikit-image. Також підключимо файли з алгоритмами.

Далі потрібно написати обробники повідомлень. Команда /start буде вітати користувача та починати переписку з ним (рис. 4.1).

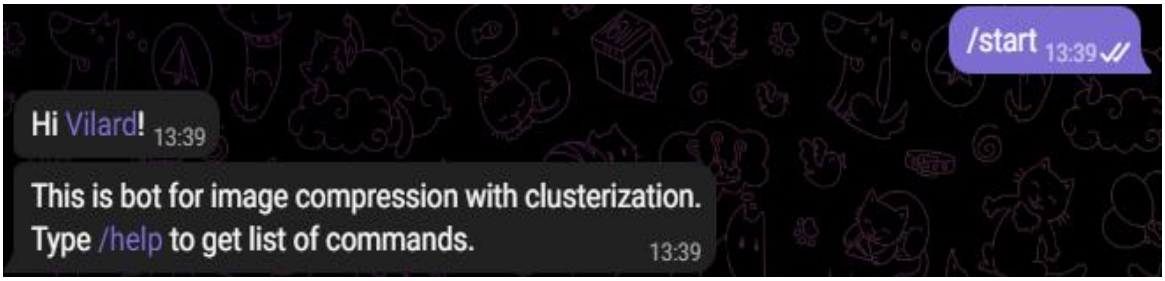

Рисунок 4.1 – Команда /start

Наступна запропонована команда /help, яка покаже список доступних команд та їхній опис (рис. 4.2).

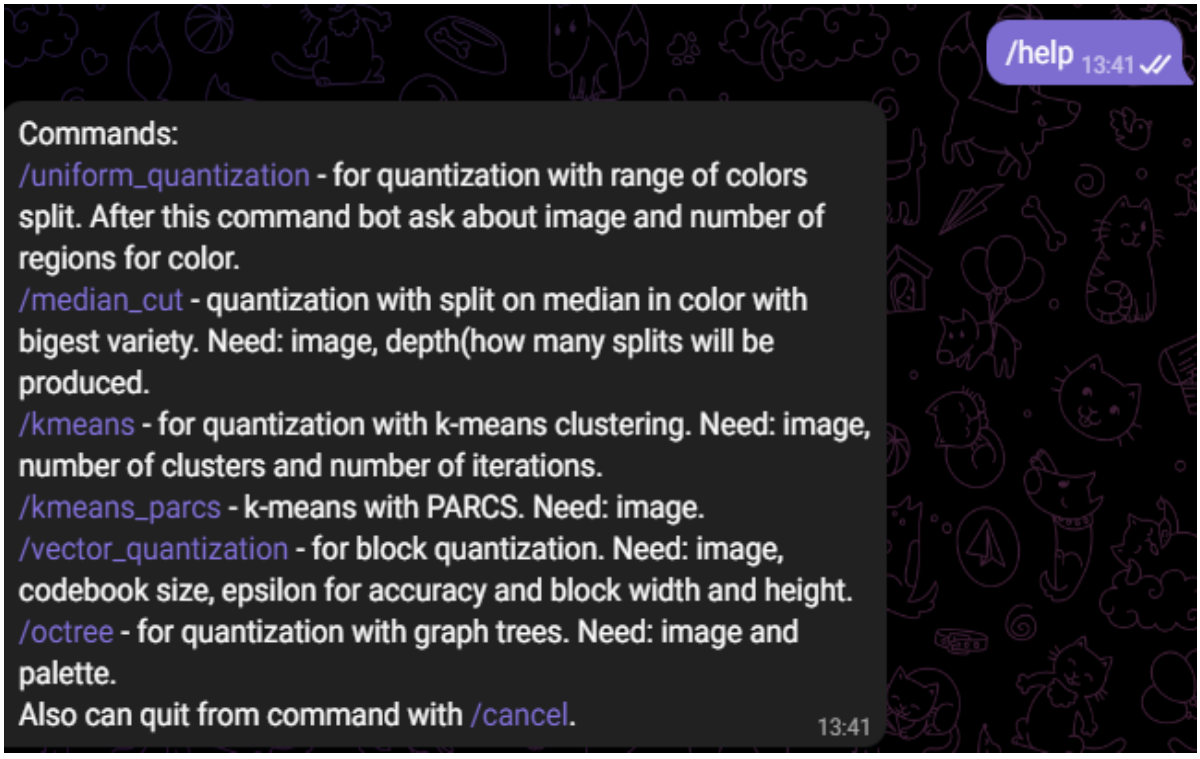

Рисунок 4.2 – Команда /help

Для наступних команд необхідно запускати оброблювачі діалогів, оскільки будемо мати по декілька запитів. Приклад оброблювача діалогів показано на рис. 4.3.

```
vq_handler = ConversationHandler(
    entry_points=[CommandHandler("vector_quantization", vector_quantization)],
    states={
        PHOTO: [MessageHandler(filters.PHOTO, photo), CommandHandler("cancel", cancel), MessageHandler(filters.ALL, send_photo)],
        CODEBOOK_SIZE: [MessageHandler(filters.Regex("^[0-9]+$"), codebook_size), CommandHandler("cancel", cancel), MessageHandler(filters.ALL, send_integer)],
        EPSILON: [MessageHandler(filters.Regex("^0.[0-9]+$"), epsilon), CommandHandler("cancel", cancel), MessageHandler(filters.ALL, send_float)],
        BLOCK_WIDTH: [MessageHandler(filters.Regex("^[0-9]+$"), block_width), CommandHandler("cancel", cancel), MessageHandler(filters.ALL, send_integer)],
        BLOCK_HEIGHT: [MessageHandler(filters.Regex("^[0-9]+$"), block_height), CommandHandler("cancel", cancel), MessageHandler(filters.ALL, send_integer)],
    },
    fallbacks=[CommandHandler("cancel", cancel)],
)
application.add_handler(vq_handler)
```

Рисунок 4.3 – Приклад оброблювача діалогів

Якщо була обрана команда стиснення зображення, але потім змінена думка, то з команди можна вийти за допомогою команди /cancel.

При введені некоректних даних, бот буде повідомляти, що необхідно зробити (рисунок 4.4).

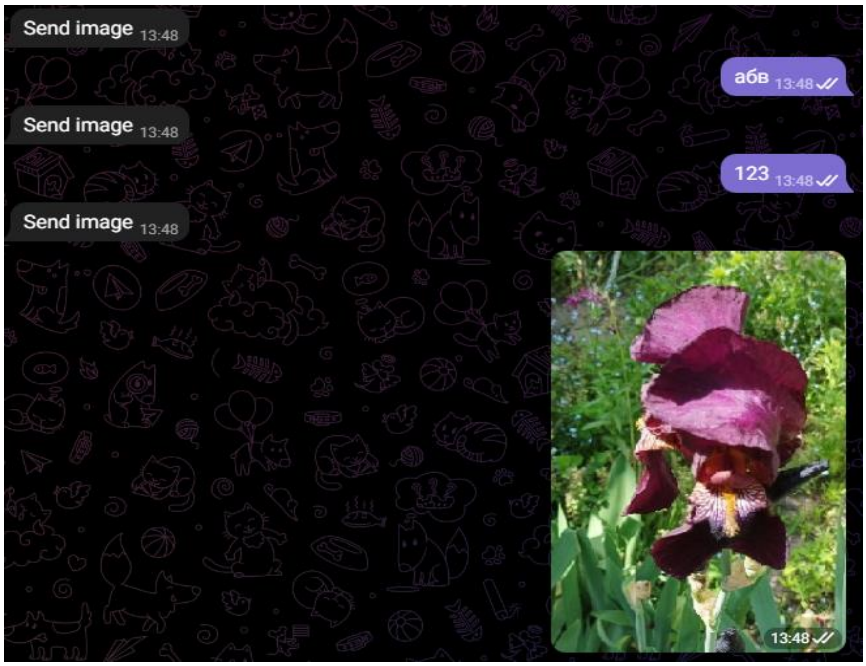

Рисунок 4.4 – Некоректний ввід

Приклад роботи бота зображено на рис. 4.5.

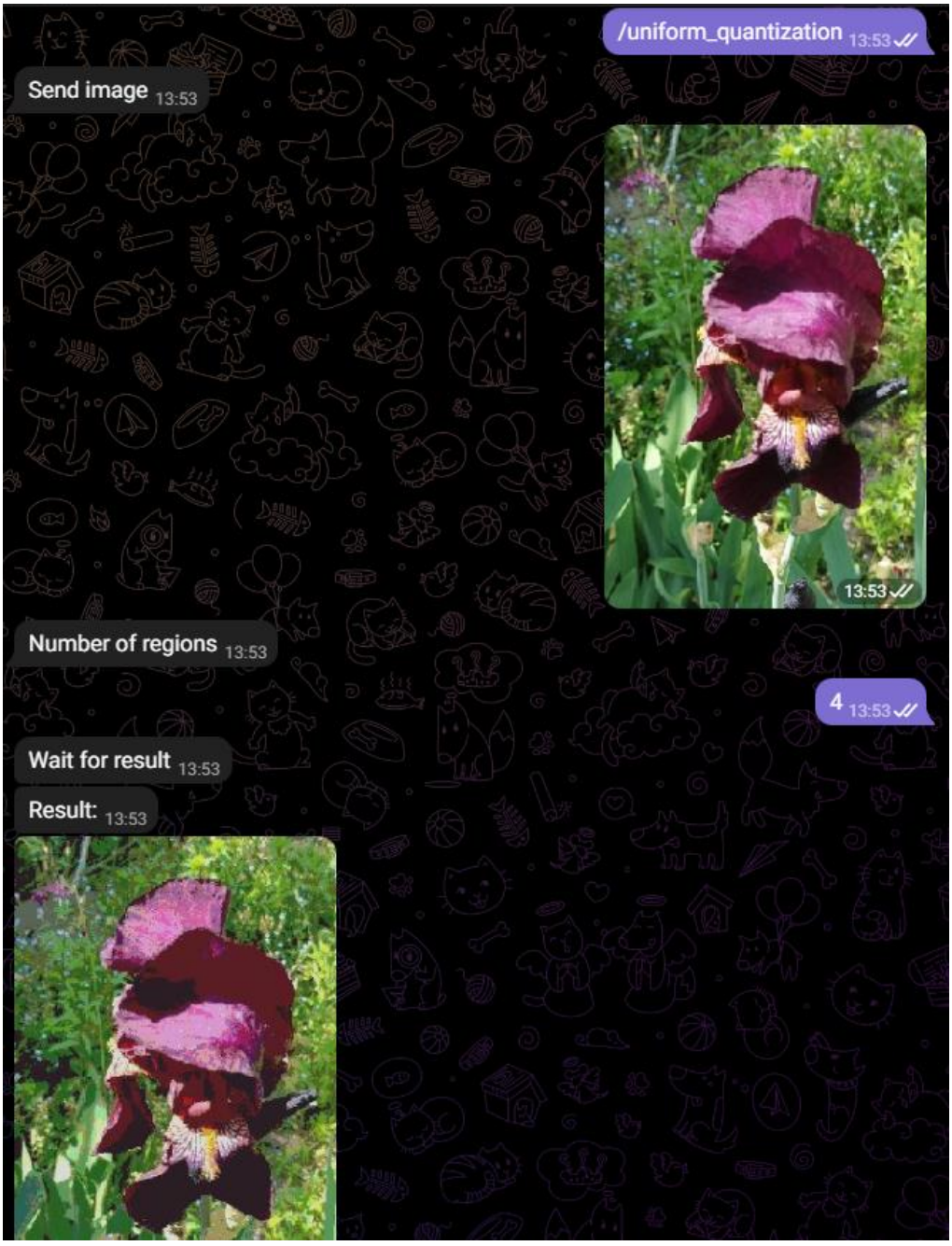

Рисунок 4.5 – Приклад роботи бота

4.4 Хостинг

Для того, щоб користувач постійно мав доступ до бота, необхідно його десь захостити. Для цього був обраний Google cloud.

Переваги Google Cloud:

– Масштабованість: Google Cloud надає гнучкі можливості масштабування, що дозволяє збільшувати або зменшувати обсяг ресурсів залежно від потреб. Ви можете легко розширювати свої ресурси в межах хмари Google, щоб задовольнити потреби проекту.

– Широкий спектр послуг: Google Cloud пропонує багато різноманітних послуг, включаючи обчислення, зберігання даних, бази даних, штучний інтелект, аналітику, машинне навчання та інше. Це дозволяє використовувати різні інструменти та сервіси, щоб відповідати специфічним потребам проекту.

– Надійність та продуктивність: Google має потужну та розподілену інфраструктуру, що забезпечує високу надійність та продуктивність послуг Google Cloud. Вони забезпечують широку географічну наявність дата-центрів, резервне копіювання даних та механізми автоматичного масштабування, що допомагають забезпечити безперебійну роботу додатків.

– Інтеграція з іншими сервісами Google: Google Cloud легко інтегрується з іншими популярними сервісами Google, такими як Google Analytics, Google Ads, Google Drive тощо. Це дозволяє використовувати дані та функціонал Google Cloud в поєднанні з іншими інструментами Google для оптимізації.

Недоліки Google Cloud:

– Складність: Google Cloud може бути складним у використанні, особливо для новачків. Конфігурація та налаштування послуг можуть вимагати додаткових зусиль та знань. Потрібно мати певний рівень технічної експертизи для ефективного використання всіх можливостей Google Cloud.

– Вартість: Хоча Google Cloud пропонує різні плани тарифів, використання послуг може бути пов'язане зі значними витратами. Необхідно враховувати вартість послуг та розраховувати бюджет проекту перед вибором Google Cloud.

– Залежність від мережі: Для доступу до послуг Google Cloud потрібне постійне та стабільне з'єднання з Інтернетом. В разі проблем з мережевим з'єднанням можуть виникати перебої у доступі до послуг та даних.

Для використання всіх функцій необхідно створити проект, на якому віртуальні машини, що знаходяться у хмарному сервісі. При створенні обираються місця розташування та потужності віртуальних машин. Після цього потрібно встановити необхідні компоненти. Для боту та алгоритмів достатньо буде однієї віртуальної машини, але для технології ПАРКС знадобиться ще 3. Одна виступатиме запускаючим сервером, а інші дві проводитимуть обчислення.

Для їхньої роботи необхідно їх запустити через відповідні команди. Для запуску всього бота необхідно залити весь код на GitHub та звідти в середовище Google cloud. Потім командами запускається бот і ним можна користуватися.

ВИСНОВКИ

З часом розміри зображень, а отже і їхня вага, зростають. На зображеннях присутня велика кількість різних кольорів, проте не всі вони є важливими для отримання корисної інформації. Щоб зберігати велику кількість зображень і при цьому не витрачати багато пам’яті, можна стискувати зображення. Можливим рішенням буде кластеризація кольорів та квантування для отримання репрезентуючих кольорів замість багатьох подібних.

В результаті виконання роботи було створено чат-бот для стиснення зображення з використанням кластеризації. Було досліджено та реалізовано різні алгоритми та наведено їхні сильні та слабкі сторони. Кожний алгоритм добре працює для рівномірно розподілених кольорів, але у випадку шумів може видавати спотворений результат. Уніформне квантування є алгоритмом, який легко написати, але він буде погано працювати для зображень, які переважно в одному кольорі. Медіановий зріз також легко реалізувати, але при найбільшому розкиді в кольоровому каналі, де не найбільше різних кольорових значень, можуть виникнути зміщені репрезентуючі кольори. Метод К-середніх можна гнучко налаштовувати, але він вимагає багато ресурсів для виконання та залежить від випадкових початкових центрів. Векторне квантування працює з блоками, тому відразу опрацьовується певна множина пікселів, проте при заміні зображення блоками, вони можуть слабо між собою поєднуватися. Метод дерева октантів має високу швидкість обробки та може сильно зменшувати кількість кольорів із збереженням якості, натомість потребує багато пам’яті.

Було також реалізовано розширення алгоритму за допомогою паралельних обчислень технології ПАРКС. Щоб була можливість скористатися ботом в будь-який момент, він був запущений на хостингу.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дерев'янченко О. В. ПАРКС-JAVA система для паралельних обчислень на комп’ютерних мережах / О. В. Дерев'янченко. – Київ, 2011. – 60 с.
2. Digital image processing [Електронний ресурс] – Режим доступу до ресурсу: https://cv.cs.nthu.edu.tw/upload/courses/18/uploads/Chapter3_2012.pdf.
3. Pattern Recognition Letters. – 2004. – №25. – С. 1025–1043.
4. Reduce the number of colors of an image using Uniform Quantization [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://muthu.co/reduce-the-number-of-colors-of-an-image-using-uniform-quantization/.
5. Color Quantization [Електронний ресурс]. – 2008. – Режим доступу до ресурсу: https://tpgit.github.io/UnOfficialLeptDocs/leptonica/color-quantization.html#modified-median-cut-implementation.
6. Arvo J. Graphics Gems II (Graphics Gems - IBM) / James Arvo., 1991. – 672 с.
7. Reducing the number of colors of an image using Median Cut algorithm [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://muthu.co/reducing-the-number-of-colors-of-an-image-using-median-cut-algorithm/.
8. What Is Image Compression? [Електронний ресурс]. – 2018. – Режим доступу до ресурсу: https://www.keycdn.com/support/what-is-image-compression.
9. Image Compression with K-means Clustering [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://medium.com/codex/image-compression-with-k-means-clustering-48e989055729.
10. Reduce the number of colors of an image using K-Means Clustering [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://muthu.co/reduce-the-number-of-colors-of-an-image-using-k-means-clustering/.

11. Python and Java Implementations for Linde-Buzo-Gray / Generalized Lloyd Algorithm [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://mkonrad.net/projects/gen_lloyd.html.
12. Magnenat-Thalmann N. New Trends in Computer Graphics / N. Magnenat-Thalmann, D. Thalmann. – Berlin: Springer Berlin, Heidelberg, 1988. – 682 с.
13. Octree color quantizer in Python [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: http://delimitry.blogspot.com/2016/02/octree-color-quantizer-in-python.html.
14. Chowdhury A. Deploy Telegram Bot on Google Cloud Platform [Електронний ресурс] / Anirban Chowdhury. – 2021. – Режим доступу до ресурсу: https://programmingforgood.medium.com/deploy-telegram-bot-on-google-cloud-platform-74f1f531f65e.

ДОДАТОК А

Код програмної реалізації:

https://github.com/VilardCool/Diploma

Посилання на бота:

https://t.me/VilardsQuantizationBot