Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук (або центр післядипломної освіти, або навчально-науковий центр заочної форми навчання)
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження можливостей інтеграції машинного навчання для оптимізації доступу до даних в гібридному сховищі зображень
(тема)

Виконав:
студент (ка) 2 курсу, групи ІПЗм-22-3

Панасенко Д.П.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-наукова

Керівник доц. Кириченко І.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти другий (магістерський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми освітньо-наукова програма
Освітня програма Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ:
зав. каф. ПІ, к.т.н., проф.
Дудар З. В.
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Панасенко Даніілу Павловичу
(прізвище, ім’я, по батькові)

1. Тема роботи «Дослідження можливостей інтеграції машинного навчання для оптимізації доступу до даних в гібридному сховищі зображень»
Затверджена наказом по університету від 29.03.2024р. № 250 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 09.06.2024
3. Вихідні дані до роботи Інформація щодо нейронних мереж, галузі машинного навчання, інформація щодо гібридного сховища зображень мова програмування Python, бібліотеки Python, MongoDB, ElasticSearch, середовища розробки JupyterLab
4. Перелік питань, що потрібно опрацювати в роботі
Аналіз сховищ зображень, аналіз гібридних підходів, аналіз підходів машинного навчання, написання програмних рішень, проведення експериментів та аналіз отриманих результатів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі та постановка задачі	01.02 – 14.02.24	*виконано*
2	Аналіз та вибір сховищ для дослідження	15.02 – 28.02.24	*виконано*
3	Проектування рішень	01.03 – 14.03.24	*виконано*
4	Планування експериментів	15.03 – 31.03.24	*виконано*
5	Програмна реалізація для проведення дослідження	01.04 – 14.04.24	*виконано*
6	Експериментальні дослідження	14.04 – 30.04.24	*виконано*
7	Аналіз результатів експериментальних досліджень	01.05 – 14.05.24	*виконано*
8	Написання та оформлення статті / тез доповіді	15.05 – 27.05.24	*виконано*
9	Підготовка пояснювальної записки	01.04 – 27.05.24	*виконано*
10	Підготовка презентації та доповіді	27.05 – 03.06.24	*виконано*
11	Нормоконтроль	03.06 – 06.06.24	*виконано*
12	Рецензування	08.06 – 10.06.24	*виконано*
13	Занесення диплома в електронний архів	10.06.2024	*виконано*
14	Попередній захист	10.06.2024	*виконано*
15	Допуск до захисту у зав. кафедри	12.06.2024	*виконано*

Дата видачі завдання 20 січня 2024р.

Студент (ка) _____ Панасенко Д.П.
(підпис)

Керівник роботи _____ доц. Кириченко І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка містить: 65 с., 5 рис., 2 табл., 21 джерело.

ГІБРИДНЕ СХОВИЩЕ ЗОБРАЖЕНЬ, ДОСТУП ДО ДАНИХ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, ELASTICSEARCH, MONGO DB, PYTHON

Об’єктом дослідження є доступ до даних в гібридних сховищах зображень.

Метою роботи є проведення практичного дослідження можливостей інтеграції машинного навчання для оптимізації доступу до даних в гібридному сховищі зображень.

Методами розробки та проектування є аналіз проблемної області дослідження, вибір MongoDB та ElasticSearch в якості гібридного сховища зображень для проведення, та використання бібліотек Python для реалізації моделей машинного навчання.

У результаті кваліфікаційної роботи було проведено теоретичне дослідження предметної галузі, розроблено програму для доступу до даних в гібридному сховищі зображень, та проведено експерименти.

HYBRID IMAGE STORAGE, DATA ACCESS, MACHINE LEARNING, NEURAL NETWORK, ELASTICSEARCH, MONGO DB, PYTHON

The object of the study is data access in hybrid image storages.

The purpose of the work is to conduct a practical study of the possibilities of integrating machine learning to optimize data access in a hybrid image storage.

The methods of development and design include the analysis of the problem domain, the selection of MongoDB and Elasticsearch as a hybrid image storage for conducting the study, and the use of Python libraries to implement machine learning models.

As a result of the qualification work, a theoretical study of the subject area was conducted, a program for data access in a hybrid image storage was developed, and experiments were conducted.

Заява щодо самостійного виконання кваліфікаційної роботи та можливості її публікації в електронному архіві відкритого доступу ElArKhNURE.

Я, Панасенко Даніїл Павлович, студент(ка) гр. ІПЗм-22-3, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження можливостей інтеграції машинного навчання для оптимізації доступу до даних в гібридному сховищі зображень», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений(на) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – База Даних

ML – Machine Learning

IDE – Integrated Development Environment

GUI – Graphical User Interface

HTTP – Hyper Text Transfer Protocol

ВСТУП

В епоху великого обсягу візуальних даних та зростаючої потреби в їх ефективному управлінні та зберіганні, гібридні сховища зображень стають стратегічно важливим компонентом сучасних інформаційних систем. Однак, навіть при використанні таких сховищ, виникають виклики щодо оптимізації доступу до даних, особливо при великих обсягах та розподілених ресурсах.

Машинне навчання, як ключова галузь інформаційних технологій, виступає як потужний інструмент для розв'язання складних завдань оптимізації та автоматизації в області обробки зображень та управління даними. Інтеграція методів машинного навчання в гібридні сховища зображень може вирішити проблеми швидкості доступу, адаптації до змінних потреб користувачів та ефективного використання ресурсів. Цим обумовлюється актуальність даного дослідження

Ця кваліфікаційна робота присвячена вивченню та дослідженню можливостей використання машинного навчання для оптимізації доступу до даних у гібридних сховищах зображень. Шляхом вивчення архітектур, алгоритмів та підходів машинного навчання, ми сподіваємося визначити ефективні стратегії для поліпшення управління великими обсягами візуальних даних в гібридних середовищах. Дана робота націлена на розкриття нових можливостей та розвиток практичних рішень для вдосконалення ефективності гібридних сховищ зображень через використання машинного навчання.

Метою роботи є проведення практичного дослідження можливостей інтеграції машинного навчання для оптимізації доступу до даних в гібридному сховищі зображень.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- зробити аналіз гібридних сховищ зображень та методів машинного навчання;
- визначити задаче для проведення експерименту;

- вибрати гібридне сховище зображень і метод машинного навчання для проведення експерименту;
- спроектувати програму для проведення експерименту;
- розробити програму для проведення експерименту;
- виміряти результати і зробити аналіз;
- надати рекомендації щодо використання машинного навчання для доступа до даних в гібридному сховищу зображень.

Об’єктом дослідження є доступ до даних в гібридних сховищах зображень.

Предметом дослідження є гібридні сховища та методи машинного навчання

Методами дослідження є аналіз проблемної області, вибір гібридного сховища зображень для проведення, та використання бібліотек Python для реалізації моделей машинного навчання.

Отримані результати дослідження можуть бути використані в процесі прийняття рішення щодо інтеграції машинного навчання для оптимізації доступу до даних в гібридному сховищі зображень. Також можливе подальше продовження дослідження в рамках інших методів машинного навчання чи на прикладі інших гібридних сховищ.

Результати даної кваліфікаційної роботи було опубліковано в журналі «Біоніка інтелекту», який включено до Переліку наукових фахових видань України, категорія «Б», технічні науки (затверджено наказом МОНУ від 02.07.2020 № 886):

- Використання машинного навчання для оптимізації доступу до даних в гібридному сховищі зображень / І. Кириченко, Г. Терещенко, Д. Панасенко// Біоніка інтелекту. – Харків: ХНУРЕ. – 2023. – № 1 (99). – с. 59-67 [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Загальний огляд предметної галузі дослідження

В сучасному світі обсяг даних постійно зростає і значну частину цих даних становлять зображення. Важливість ефективного зберігання, управління та доступу до візуальних даних важко переоцінити, оскільки ці задачі виникають у різних галузях, включаючи медицину, безпеку, комерцію, наукові дослідження та багато інших.

Зберігання та управління зображеннями є важливим завданням, яке потребує ефективних рішень. Звичайні реляційні бази даних часто не забезпечують достатньої продуктивності для зберігання та обробки великих обсягів візуальних даних. Для таких задач підходять нові підходи, такі як NoSQL бази даних, об'єктні сховища та гібридні сховища.

Гібридні сховища поєднують різні типи сховищ, надаючи можливість зберігати дані в оптимальному форматі в залежності від їх характеристик та вимог до доступу. Наприклад, метадані зображень можуть зберігатися в реляційних або документних базах даних, тоді як самі зображення можуть зберігатися в об'єктних сховищах. Це забезпечує ефективне управління та доступ до даних [2].

Машинне навчання значно змінює підходи до обробки візуальних даних. ML моделі можуть використовуватися для автоматичного витягання ознак зображень, класифікації, розпізнавання об'єктів, покращення якості зображень та багато іншого. Використання попередньо навчених моделей дозволяє значно знизити час та ресурси, необхідні для навчання, забезпечуючи при цьому високу точність і ефективність.

Гібридні сховища даних у поєднанні з методами машинного навчання надають потужний інструмент для ефективного управління великими обсягами візуальних даних.

Таким чином, сучасні підходи до зберігання та обробки зображень базуються на використанні гібридних сховищ та методів машинного навчання, що забезпечує ефективне управління візуальними даними, підвищує швидкість та точність доступу до них, а також дозволяє вирішувати складні завдання в різних галузях.

1.2 Огляд гібридних сховищ зображень

Гібридні сховища даних представляють собою комбінацію різних типів сховищ, які працюють разом для забезпечення оптимальної продуктивності, масштабованості та гнучкості [2]. Вони поєднують переваги реляційних баз даних (SQL) та нереляційних баз даних (NoSQL) для задоволення різних потреб у зберіганні та обробці даних.

Основна ідея гібридного сховища полягає в тому, щоб використовувати найкращі властивості кожного типу сховища для вирішення конкретних завдань. Наприклад, реляційні бази даних відмінно підходять для транзакційних систем, де важлива цілісність та консистентність даних, тоді як NoSQL бази даних, такі як MongoDB, забезпечують високу продуктивність та масштабованість для зберігання великих обсягів даних, особливо неструктурованих або напівструктурованих.

Основні особливості використання гібридних сховищ зображення.

Гнучкість у виборі технологій: Гібридні сховища дозволяють використовувати різні бази даних для різних типів даних. Це означає, що можна вибрати найкращий інструмент для кожної конкретної задачі.

Масштабованість: Використання NoSQL баз даних, таких як MongoDB, дозволяє легко масштабувати сховище для обробки великих обсягів даних. Це особливо важливо для додатків, які мають справу з великими обсягами неструктурованих даних.

Висока продуктивність: Гібридні сховища забезпечують високу продуктивність за рахунок використання NoSQL баз даних для швидкого доступу до великих обсягів даних та реляційних баз даних для забезпечення цілісності транзакцій.

Забезпечення цілісності даних: Реляційні бази даних, такі як PostgreSQL, або MySQL, забезпечують цілісність даних та підтримку складних транзакцій, що є важливим для критично важливих додатків.

Зниження витрат: Використання гібридних сховищ дозволяє знизити витрати за рахунок оптимізації використання ресурсів. Наприклад, часто використовувані дані можуть зберігатися у швидкодоступних NoSQL базах даних, тоді як менш

часто використовувані дані можуть зберігатися у дешевших реляційних базах даних.

Проблеми, які вирішують гібридні сховища.

Управління різноманітними типами даних: Гібридні сховища дозволяють ефективно управляти як структурованими, так і неструктурованими даними. Це особливо корисно для додатків, які обробляють різноманітні типи даних, такі як текстові документи, зображення, відео та інші мультимедійні дані.

Швидкий доступ до даних: Використання NoSQL баз даних дозволяє забезпечити швидкий доступ до великих обсягів даних, що є критично важливим для додатків з високими вимогами до продуктивності.

Масштабованість: Гібридні сховища легко масштабуються, що дозволяє забезпечити безперебійний доступ до даних навіть при збільшенні обсягів даних.

Забезпечення безпеки даних: Реляційні бази даних забезпечують високий рівень безпеки даних, що є важливим для додатків, які обробляють конфіденційну інформацію.

Загалом, гібридні сховища забезпечують гнучкість, продуктивність та масштабованість, що робить їх ідеальним рішенням для багатьох сучасних завдань, пов'язаних з обробкою та зберіганням даних. Гібридні сховища дозволяють ефективно управляти різноманітними типами даних та забезпечувати швидкий доступ до них, що є критично важливим у сучасному світі великих даних.

1.3 Застосування машинного навчання

Машинне навчання стає дедалі важливішим інструментом у різних сферах, де зберігаються та обробляються великі обсяги зображень. Його застосування в поєднанні з гібридними сховищами зображень дозволяє вирішувати низку складних задач, що вимагають автоматизації та підвищеної точності.

Ось кілька прикладів задач, які можуть бути ефективно вирішені за допомогою машинного навчання в контексті гібридних сховищ зображень:

– автоматична класифікація зображень: машинне навчання може бути використане для автоматичної класифікації зображень у великій базі даних [3];

– аналіз і обробка зображень: алгоритми машинного навчання здатні автоматично аналізувати зображення для виявлення певних об'єктів або особливостей;

– сегментація зображень: у сфері обробки зображень важливо вміти сегментувати зображення, тобто розділяти його на значущі частини, машинне навчання дозволяє автоматично виділяти об'єкти на зображеннях;

– пошук схожих зображень: це одна з найпоширеніших задач, яка може бути вирішена за допомогою машинного навчання – система, що використовує алгоритми машинного навчання, здатна швидко знайти схожі зображення в базі даних на основі їхніх ознак;

– анотація та розпізнавання зображень: машинне навчання може автоматично додавати анотації до зображень, розпізнаючи об'єкти або сцени і це спрощує процес організації та пошуку зображень у великих базах даних.

Використання машинного навчання в контексті гібридних сховищ зображень дозволяє значно покращити ефективність та точність обробки візуальних даних.

1.4 Формування проблеми

Розібравшись в особливостях доступу до даних в гібридних сховищах зображень і проблемами які вони вирішують, для експериментальної інтеграції машинного навчання було вирішено взяти показову задачу, яка вирішується завдяки машинному навчанню. Однією з найактуальніших задач в цій області є пошук схожих зображень [4]. Ця задача полягає в тому, щоб знайти велику кількість зображень, схожих на задане, серед інших зображень в гібридному сховищі зображень. Пошук схожих зображень є дуже практичною задачею, вирішення якої може принести значну користь у різних сферах.

Проблема, яка вирішується в даній роботі, полягає в оптимізації процесу пошуку схожих зображень у гібридному сховищі.

Основна мета цього дослідження – визначити, які методи машинного навчання можуть бути інтегровані в процес пошуку для підвищення його ефективності. Важливо також оцінити, як саме використання гібридного сховища

впливає на ефективність рішення даної задачі, а також які переваги застосування такого підходу.

1.5 Постановка задачі

Виходячи з усього перерахованого вище, в рамках дослідження необхідно вирішити наступні завдання:

- розглянути існуючі методи та технології управління зображеннями;
- оцінити потреби та вимоги до системи зберігання та пошуку зображень;
- розглянути та оцінити різні методи машинного навчання для пошуку схожих зображень [4];
- розробити архітектуру системи для зберігання та пошуку схожих зображень;
- розробити програму для пошуку схожих зображень в гібридному сховищі зображень використовуючи методи машинного навчання;
- провести експериментальні дослідження для оцінки продуктивності системи;
- проаналізувати результати експериментів та надати рекомендації.

2 ОПИС ПРИЙНЯТИХ ПРОЄКТНИХ РІШЕНЬ

2.1 Вибір підходу до інтеграції машинного навчання

Вибір підходу до інтеграції машинного навчання є критично важливим етапом данного дослідження, оскільки саме методи машинного навчання визначають ефективність і точність пошуку схожих зображень у гібридному сховищі. Існує кілька популярних методів і моделей, які можуть вирішувати цю задачу [4].

Серед найбільш поширених підходів для пошуку схожих зображень можна виділити наступні методи машинного навчання.

Автокодувальники (Autoencoders) [5]: ці нейронні мережі можуть навчитися стискати дані до латентного простору меншої розмірності, зберігаючи важливу інформацію. Вони часто використовуються для вилучення ознак і можуть бути застосовані для порівняння схожості зображень. Автокодувальники можуть працювати з неструктурованими даними і вилучати значущі ознаки, але вони можуть бути менш точними у порівнянні з більш складними моделями, такими як CNN або ResNet.

Сверточні нейронні мережі (Convolutional Neural Networks, CNNs) [6]: CNN широко використовуються для аналізу зображень. Вони можуть вилучати складні ознаки з зображень, що робить їх ідеальними для задач класифікації та пошуку схожих зображень. CNN здатні вилучати складні та значущі ознаки зображень, що робить їх високоефективними для задач класифікації, але навчання CNN може бути ресурсомістким та вимагати великої кількості даних.

Мережі глибинного навчання для вилучення ознак [7] (Deep Feature Extraction): Цей підхід полягає у використанні попередньо навчених моделей, таких як VGG, Inception або ResNet, для вилучення ознак зображень, які потім використовуються для порівняння схожості [8]. Використання попередньо навчених моделей дозволяє знизити витрати на обчислювальні ресурси та час на навчання, а також забезпечує високу точність результатів, але можуть виникати проблеми з адаптацією до специфічних завдань або даних.

Зібрані дані про методи машинного навчання для пошуку схожих зображень, було зведено в порівняльну таблицю (див. таб. 1).

Таблиця 1 – Порівняння методів машинного навчання (таблицю створено самостійно)

Метод	Переваги	Недоліки
Автокодувальники	Працюють з неструктурованими даними, вилучають значущі ознаки	Менш точні
CNN	Вилучають складні та значущі ознаки, висока ефективність	Ресурсомісткі, потребують великої кількості даних
Глибинне вилучення ознак	Знижені витрати на обчислення та навчання, використання попередньо навчених моделей, висока точність результатів	Можливі проблеми з адаптацією до специфічних завдань чи даних

Виходячи з проведеного аналізу було вирішено для інтеграції машинного навчання в систему пошуку схожих зображень обрати метод глибинного вилучення ознак, так як він найбільш підходить для проведення даного дослідження, не вимагає витрат на навчання, через використання попередньо навчених моделей, а також надає високу точність результатів.

Серед цих методів ми обрали модель ResNet50 (Residual Networks) [9]. ResNet50 є однією з найпопулярніших моделей для вилучення ознак завдяки своїй високій точності. Модель ResNet50 вже попередньо навчена на великому датасеті зображень, що дозволяє використовувати її для вилучення ознак без додаткового навчання, знижуючи витрати на обчислювальні ресурси. Крім того, ResNet50 забезпечує високу точність результатів, що робить її ідеальним вибором для нашого дослідження.

ResNet50 – це глибока нейронна мережа, яка складається з 50 шарів і використовує концепцію "залишкових блоків" (residual blocks) для подолання проблеми затухання градієнтів у глибоких мережах. Основна ідея залишкових блоків полягає в тому, що кожен блок не намагається вивчити точну карту відображення вхідних даних до вихідних, а замість цього вивчає різницю (залишок) між вхідними та вихідними даними.

Коли зображення подається на вхід ResNet50, воно проходить через кілька згорткових шарів, шарів підвибірки (pooling layers) і повнозв'язаних шарів. На кожному етапі мережа вилучає все більш абстрактні ознаки, що дозволяє моделі будувати багатошарове представлення вхідного зображення. На виході ResNet50 ми отримуємо вектор ознак (feature vector), який можна використовувати для порівняння схожості з іншими зображеннями.

Для порівняння схожості між векторами ознак ми використовуємо метрику косинусної подібності [10]. Косинусна подібність вимірює кут між двома векторами у векторному просторі і визначає, наскільки ці вектори схожі між собою. Воно обчислюється за формулою 2.1:

$$similarity = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \tag{2.1}$$

де A і B – два вектори.

Косинусна подібність набуває значень від -1 до 1, де 1 означає, що вектори ідеально вирівняні (максимальна схожість), 0 означає, що вони ортогональні (немає схожості), а -1 означає, що вони ідеально протилежні. Для задачі пошуку схожих зображень ми зазвичай розглядаємо значення косинусної подібності від 0 до 1.

2.2 Вибір гібридного сховища зображень

Виходячи з проведеного аналізу, визначеної задачі і обраного методу машинного навчання, визначимо що функціональністю гібридного сховища зображень має бути: збереження зображень та пошук за вектором ознак. Вибір БД

для виконання поставленої задачі проведемо за даною необхідною функціональністю.

Виділимо головні вимоги до збереження зображень:

– підтримка великих обсягів даних;

– гнучка схема зберігання для різних форматів даних (зображення, метадані);

– висока продуктивність при записі і читанні даних;

– масштабованість і надійність.

Виходячи з вимог до збереження зображень, БД, які відповідають цим вимогам, можуть бути такі БД, як: MongoDB, Cassandra та Couchbase [11]. В якості БД для збереження зображень для виконання дослідження було обрано MongoDB, так як вона являється найбільш популярною, з великою спільнотою користувачів.

Виділимо головні вимоги до пошуку за векторами ознак:

– швидка індексація та пошук за векторними ознаками;

– підтримка складних запитів і агрегацій;

– висока продуктивність при обробці великих обсягів даних;

– масштабованість і надійність.

Виходячи з вимог до пошуку за векторами ознак, БД, які відповідають цим вимогам можуть бути, такі БД, як: Elasticsearch та Faiss [12]. В якості БД для здійснення пошуку за векторами ознак для виконання дослідження було обрано Elasticsearch. Elasticsearch є оптимальним вибором для пошуку за векторами ознак завдяки своїй високій швидкості пошуку, швидкій індексації, підтримці складних запитів та агрегацій, а також розподіленій архітектурі. Крім того, Elasticsearch має велику та активну спільноту, що забезпечує підтримку та розвиток технології.

2.3 Опис архітектури рішення

Програмне рішення базується на використанні двох головних компонентів. Перша компонента – це розгорнутий Docker Compose [13], який відповідає за контейнеризацію гібридного сховища зображень. Друга компонента це Python –

додаток у JupyterLab, який використовується для виконання основного коду даного дослідження.

В якості гібридного сховища зображень було обрано використання комбінації двох баз даних – MongoDB та ElasticSearch.

MongoDB – це документо-орієнтована база даних, яка забезпечує ефективне зберігання та доступ до великих обсягів даних. Це найбільш ефективний спосіб зберігання зображень у базі даних. Кожне зображення зберігається у форматі BSON разом з інформацією про його метадані, такі як назва файлу, розмір та формат [14].

Так як задача даного дослідження вимагає пошук схожих зображень за допомогою векторних подібностей, було обрано ElasticSearch. ElasticSearch – це найкращий інструмент для швидкого пошуку векторів ознак зображень, так як в цій БД присутня індексація та Elasticsearch підтримує потужні функції, що робить його ідеальним вибором для виконання даного завдання [15].

Docker Compose використовується для управління контейнерами, що забезпечують роботу MongoDB та Elasticsearch. Це дозволяє легко розгортати та масштабувати систему, забезпечуючи ізоляцію середовища виконання для кожного компонента.

Python-додаток складається з кількох функціональних модулів, які забезпечують виконання різних функцій.

Модуль завантаження зображень: відповідає за завантаження зображень у гібридне сховище даних. Він обробляє завантажені зображення та зберігає їх у MongoDB і ElasticSearch.

MongoDB-клієнт: забезпечує взаємодію з базою даних MongoDB для зберігання та отримання зображень та їх метаданих.

ElasticSearch-клієнт: Відповідає за індексацію та пошук векторів ознак зображень у Elasticsearch.

ML–модель: Використовується для генерації векторів ознак зображень. Було обрано модель ResNet50, яка завантажується з попередньо навченими вагами ImageNet [16]. За допомогою даної моделі відбувається порівняння схожості зображень.

Дана архітектура представлена у вигляді діаграми компонентів (див. рис. 1).

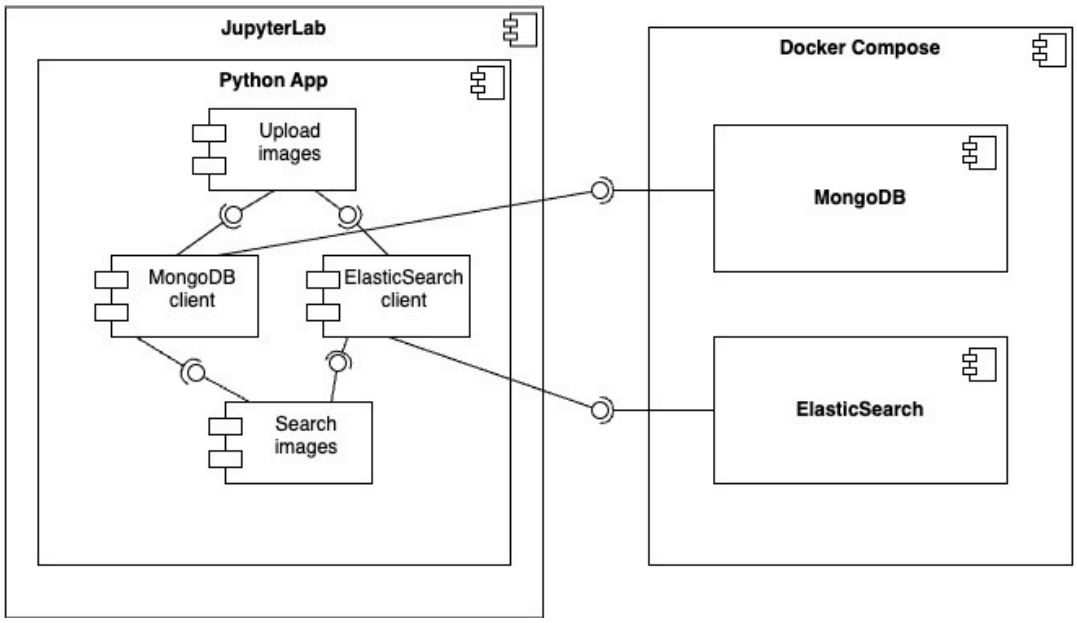

Рисунок 1 – Діаграма компонентів програмної реалізації дослідження (рисунок створено самостійно)

Ця архітектура забезпечить ефективну обробку, зберігання та пошук зображень, використовуючи переваги обраних нами сучасних технологій та інструментів.

2.4 Вибір технологій та інструментів

Для успішної реалізації проекту було обрано широкий спектр технологій та інструментів, які забезпечують ефективну розробку, тестування та проведення якісного дослідження. У цьому розділі буде детально розглянуто кожну з обраних технологій та пояснені причини їх вибору.

Основною мовою програмування для проведення дослідження, було обрано мову програмування Python. Python відомий своєю простотою та читабельністю коду, що робить його дуже зручним інструментом. Крім того, Python має велику кількість бібліотек для роботи з даними та машинного навчання, що допоможе виконати поставлену задачу.

Для розробки були використані наступні бібліотеки.

Tensorflow Keras [17]. Для реалізації та тренування моделей машинного навчання використовується бібліотека Keras з бекендом TensorFlow [18]. Це дозволило використати модель нейронної мережі, таку як ResNet50, яка використовується для визначення векторів ознак зображень. Використання попередньо натренованої моделі ResNet50 значно спростило процес машинного навчання для обробки зображень та підвищило точність результатів даного дослідження.

Pandas та NumPy [19]. Використання даних бібліотек допомагає дуже зручно обробляти і маніпулювати даними, багатовимірними масивами і числовими обчисленнями.

PyMongo та elasticsearch-py. Дані бібліотеки використовувались у якості клієнтів для забезпечення доступу до даних у сховища даних.

Matplotlib [19]. Для візуалізації результатів та аналізу даних обрана бібліотека Matplotlib. Вона дозволяє створювати графіки для візуального представлення результатів експериментів, а також виводити зображення у вікно виводу IDE.

Sklearn [20]. З даної бібліотеки використовувалась функція cosine_similarity для визначення косинусної подібності векторів.

Розробка виконувалась в веб IDE – JupyterLab, що є інтерактивним середовищем для роботи з Python. JupyterLab дозволяє виконувати код, переглядати результати та візуалізувати дані у реальному часі, що значно спрощує процес дослідження та тестування моделей машинного навчання. Це середовище є особливо корисним для роботи з даними та побудови експериментів.

Для реалізації поставленої задачі також необхідно створити гібридне сховище зображень. В якості гібридного сховища зображень використовується комбінація MongoDB та ElasticSearch.

MongoDB – це документо-орієнтована база даних, яка забезпечує високу гнучкість та масштабованість. MongoDB дозволяє зберігати великі обсяги даних у форматі JSON та легко їх обробляти. MongoDB в даному дослідженні використовується для зберігання зображень та пов'язаних з ними метаданих.

ElasticSearch – це пошуковий рушій з відкритим вихідним кодом, який спеціалізується на повнотекстовому пошуку та аналізі великих обсягів даних у реальному часі. ElasticSearch дозволяє швидко проводити пошук схожих зображень за допомогою методів машинного навчання та підтримує індексацію та пошук векторних даних.

Kibana використовувалась в якості веб GUI для роботи з ElasticSearch.

MongoDB Compass використовувався в якості десктоп GUI для роботи з MongoDB.

Для забезпечення зручного розгортання та управління інфраструктурою проекту був обран Docker. Docker дозволяє створювати контейнери, які ізолюють середовище виконання додатків, забезпечуючи їхню портативність та легкість розгортання. Docker Compose був використаний для спрощення управління багатосервісною архітектурою, що включає MongoDB, Elasticsearch та Kibana.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Опис функціональних модулів

Архітектура програмної реалізації включає кілька основних функціональних модулів, які забезпечують роботу зі сховищами зображень, їх обробку та пошук зображень. Кожен з цих модулів відіграє важливу роль у забезпеченні швидкої та ефективної роботи системи.

Інфраструктура для проведення дослідження побудована за допомогою Docker Compose (див. Додаток Б), що дозволяє легко розгорнути необхідні сервіси, такі як MongoDB та Elasticsearch, які будуть виступати в якості гібридного сховища зображень. Використання Docker забезпечує ізольоване середовище для роботи сервісів, полегшуючи процес їх налаштування та обслуговування [13].

Docker Compose створює два сервіси: MongoDB для зберігання зображень та їх метаданих, Elasticsearch для індексації та пошуку ознак зображень, а також Kibana в якості GUI для зручної роботи з ElasticSearch.

Головні функціональні модулі будуть відповідати за інтеграцію з MongoDB, ElasticSearch, реалізацію і інтеграцію моделі машинного навчання, а також тестування і виконання експериментів.

3.1.1 Інтеграція з MongoDB

Інтеграція з MongoDB [14] включає завантаження зображень, збереження метаданих та ознак, а також отримання зображень за їх ідентифікаторами.

Для інтеграції з MongoDB в якості клієнта було використано бібліотеку pymongo.

Підключення до MongoDB:

```
mongo_client = pymongo.MongoClient("mongodb://localhost:27017/")
mongo_db = mongo_client["image_database"]
mongo_collection = mongo_db["images"]
```

Функція upload_image_mongo відповідає за завантаження зображень у базу даних MongoDB. Вона відкриває зображення, читає його дані та зберігає їх у MongoDB разом із метаданими та векторами ознак.

Код функції завантаження зображення в MongoDB:

```
def upload_image_mongo(image_path, features):
    with open(image_path, "rb") as image_file:
        image_data = image_file.read()
        image = Image.open(io.BytesIO(image_data))
        metadata = {
            "filename": os.path.basename(image_path),
            "size": image.size,
            "format": image.format
        }
        result = mongo_collection.insert_one({
            "image_data": image_data,
            "metadata": metadata,
            "features": features.tolist()
        })
        print(f"Uploaded {image_path} with ID {result.inserted_id}")
        return result.inserted_id
```

Для отримання зображень та їх метаданих з MongoDB використовуються функції get_image_by_id_mongo та get_document_by_id_mongo. Ці функції дозволяють отримати зображення за його унікальним ідентифікатором та відповідні метадані.

Код функцій отримання зображень в MongoDB:

```
def get_image_by_id_mongo(image_id):
    document = mongo_collection.find_one({"_id": ObjectId(image_id)})
    if document:
        image_data = document["image_data"]
        image = Image.open(io.BytesIO(image_data))
        return image
    else:
        print(f"No image found with ID {image_id}")
        return None

def get_document_by_id_mongo(image_id):
    document = mongo_collection.find_one({"_id": ObjectId(image_id)})
    if document:
        return document
    else:
        print(f"No image found with ID {image_id}")
        return None
```

3.1.2 Інтеграція з ElasticSearch

Для реалізації гібридного сховища зображень було обрано ElasticSearch, який надасть змогу робити швидкий пошук схожих зображень за розрахованим нейронною мережею вектором ознак.

Пілся розгортання ElasticSearch [15] в Docker, необхідно створити індекс для швидкого пошуку за векторною подібністю. Для створення індексу необхідно надіслати PUT HTTP-запит.

Код HTTP-запиту на створення індексу в ElasticSearch:

```
curl -X PUT "localhost:9200/image_features" -H 'Content-Type: application/json' -d'
{
  "mappings": {
    "properties": {
      "features": {
        "type": "dense_vector",
        "dims": 2048
      }
    }
  }
}'
```

Для інтеграції з ElasticSearch було обрано в якості клієнту однойменну бібліотеку для Python – elasticsearch.

Код підключення до ElasticSearch:

```
es = Elasticsearch([{'host': 'localhost', 'port': 9200, 'scheme': 'http'}],
    connections_per_node=25,
    request_timeout=60,
    retry_on_timeout=True
)
```

Взаємодія з ElasticSearch включає в себе завантаження векторів ознак зображень, а також отримання схожих зображень за запитом. Розроблена функція завантаження векторів ознак зображень index_image_features.

Код функції завантаження вектору ознак зображення в ElasticSearch:

```
def index_image_features(image_id, features):
    body = {"features": features.tolist()}
    es.index(index="image_features", id=image_id, body=body)
    print(f"Uploaded image {image_id} with features {features}")
```

Реалізацію запиту на отримання найбільш схожих зображень буде описано пізніше в рамках інтеграції з моделлю машинного навчання.

3.2 Реалізація і інтеграція алгоритмів машинного навчання

Машинне навчання в програмі використовується для визначення вектору ознак зображень. Для цього обрано модель ResNet50 [9], яка завантажується та ініціалізується за допомогою бібліотеки Tensorflow [17]. Модель використовується для отримання векторів ознак зображень, які потім зберігаються у MongoDB та Elasticsearch для подальшого порівняння.

Код налаштування моделі ResNet50:

```
model = ResNet50(weights='imagenet', include_top=False, pooling='avg')
```

Використання цієї моделі відбувається для визначення вектору ознак зображення у функції extract_features. Функція завантажує зображення, змінює його розмір до 224x224 пікселів (як це потрібно для ResNet50) і перетворює його у формат, прийнятний для моделі. Потім зображення передається через модель, яка повертає вектор ознак.

Код визначення вектору ознак зображення за моделлю ResNet50:

```
def extract_features(image):
    img = image.resize((224, 224))
    img_data = np.array(img)

    img_data = np.expand_dims(img_data, axis=0)
    img_data = preprocess_input(img_data)

    features = model.predict(img_data, verbose=0)
    return features.flatten()
```

Для порівняння зображень використовуємо функцію compare_images, яка обчислює коефіцієнт схожості між двома векторами ознак. Використовується косинусна подібність [10], яка є метрикою схожості між двома векторами, визначаючи кут між ними. Чим менший кут, тим більша схожість між векторами, для цього використовується функція cosine_similarity бібліотеки Sklearn [20].

Код порівняння схожості зображень за векторами ознак:

```
def compare_images(features1, features2):
    similarity = cosine_similarity([features1], [features2])[0][0]
    return similarity
```

В якості експерименту будемо розглядати три підходи пошуку зображень, використовуючи гібридні сховища зображень. Першим підходом буде пошук зображень на стороні ElasticSearch, який реалізовано у функції search_similar_images_elastic. Функція виконує запит до Elasticsearch, де зберігаються вектори ознак зображень. Запит використовує косинусну подібність для визначення схожості між запитуваним зображенням і зображеннями у базі даних.

Код функції пошуку схожих зображень в ElasticSearch:

```
def search_similar_images_elastic(query_features, num_results):
    query_vector = [float(x) for x in query_features.tolist()]
    body = {
        "query": {
            "script_score": {
                "query": {"match_all": {}},
                "script": {
                    "source": """
                        double                  score                  =
cosineSimilarity(params.query_vector, 'features') + 1.0;
                        if (score > 1.9) { return 0; }
                        return score;
                    """,
                    "params": {"query_vector": query_vector}
                }
            }
        },
        "size": num_results,
        "min_score": 0.1
    }
    try:
        response = es.search(index="image_features", body=body)
```

```
            return response["hits"]["hits"]
        except Exception as e:
            print(f"Error during search: {str(e)}")
            return []
```

Альтернативні методи пошуку схожих зображень будуть відбуватись виключно в MongoDB, тобто не використовуючи гібридні сховища зображень. Функція search_similar_images_mongo відповідає за пошук схожих зображень, обробляючи кожне зображення з бази даних, витягуючи його ознаки і обчислюючи коефіцієнт схожості з запитуваним зображенням.

Код функції пошуку схожих зображень в MongoDB:

```
    def search_similar_images_mongo(features, num_results):
        ids = get_all_ids_mongo()
        similarities = []

        for id in ids:
            image = get_image_by_id_mongo(id)
            if image:
                features_mongo = extract_features(image)
                similarity = compare_images(features, features_mongo)
                if similarity <= 0.9:
                    similarities.append((similarity, id))

        similarities.sort(reverse=True, key=lambda x: x[0])
        top_similarities = similarities[:num_results]

        results = [get_document_by_id_mongo(sim_id) for _, sim_id in
top_similarities]
        return results
```

Більш оптимізованим варіантом пошуку у MongoDB буде попередній розрахунок вектору ознак зображень, збережені його одразу разом з зображенням. Це значно прискорює процес пошуку, оскільки немає необхідності повторно обчислювати ознаки для кожного зображення. Даний підхід реалізовано у функції search_similar_images_mongo_preprocess.

Код функції пошуку схожих зображень в MongoDB з попереднім прорахунком вектору ознак:

```
    def search_similar_images_mongo_preprocess(features, num_results=3):
        ids = get_all_ids_mongo()
        similarities = []
```

```
        for id in ids:
            document = get_document_by_id_mongo(id)
            if document:
                features_mongo = np.array(document["features"])
                similarity = compare_images(features, features_mongo)
                if similarity <= 0.9:
                    similarities.append((similarity, id))

        similarities.sort(reverse=True, key=lambda x: x[0])
        top_similarities = similarities[:num_results]

        results = [get_document_by_id_mongo(sim_id) for _, sim_id in
top_similarities]
        return results
```

Дані функції будуть використанні під час експерименту, буде зроблено заміри і порівняння цих підходів для визначення найбільш ефективного з них.

3.3 Тестування програмного забезпечення

Тестування програмного забезпечення є важливим етапом, який забезпечує впевненість у правильності та ефективності реалізації розробленої системи. Система включає кілька компонентів, таких як завантаження зображень у сховища даних, витягнення ознак зображень, пошук схожих зображень за допомогою MongoDB та ElasticSearch. Тестування проводилось для кожного з цих компонентів, щоб переконатися в їх коректній роботі. Це ж тестування буде відбуватись в рамках проведення експерименту.

Тестування вимагає написання допоміжних функцій, перші з них – це функції по завантаженню дата-сету зображень в гібридне сховище даних, для підготовки даних для роботи з ним.

Код функцій завантаження дата-сету зображень в гібридне сховище:

```
def upload(image_path):
    image = Image.open(image_path)
    features = extract_features(image)
    image_id = upload_image_mongo(image_path, features)
    index_image_features(image_id, features)
    return image_id

def upload_dataset(num):
    image_paths = get_random_image_paths(directory, num)
    counter = 0
```

```
        for image_path in image_paths:
            image_id = upload(image_path)
            counter += 1
            print(f"Изображение {counter} загружено с ID: {image_id}")
```

Після виконання пошуку схожих зображень, результат необхідно вивести на екран для наглядності виконаного пошуку. Відображення результату реалізовано в функції display_images [19].

Код функції відображення результату пошуку схожих зображень:

```
    def display_images(
       original_image, similar_images, from_elastic=False):
        fig, axs = plt.subplots(1, len(similar_images) + 1, figsize=(15,
5))

        if len(similar_images) == 1:
            axs = [axs]

        axs[0].imshow(original_image)
        axs[0].axis('off')
        axs[0].set_title("Original image")

        for i, hit in enumerate(similar_images):
            if from_elastic:
                image_id = hit["_id"]
                image = get_image_by_id_mongo(image_id)
                score = hit["_score"] - 1
            else:
                image_id = hit["_id"]
                document = get_document_by_id_mongo(image_id)
                image = get_image_by_id_mongo(image_id)
                original_image_features = extract_features(original_image)
                score      =      compare_images(original_image_features,
document["features"])

            if image:
                axs[i + 1].imshow(image)
                axs[i + 1].axis('off')
                axs[i + 1].set_title(f"Score: {score:.2f}")

        plt.show()
```

Дані функції допоможуть у проведені експерименту, щоб зробити заміри ефективності кожного з підходів необхідних для аналізу.

4 ОПИС ЕКСПЕРЕМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

4.1 Планування експериментів

Результатами проведених експериментів мають бути дані для аналізу і формування висновків щодо доцільності використання обраного методу машинного навчання ResNet50 [9] з гібридним підходом до зберігання зображень у вирішенні проблеми – швидкого пошуку схожих зображень в сховищі даних.

Для цього визначено провести порівняння трьох різних підходів для вирішення цієї проблеми.

Перший підхід. Збереження зображення в єдиному сховищі MongoDB. При пошуку схожих зображень перебрати усі зображення зі сховища, розрахувати вектор ознак і вирахувати коефіцієнт їхньої схожості.

Другий підхід. Збереження зображення в єдиному сховищі MongoDB з попереднім прорахунком вектору ознак зображення і збережені їх в цьому ж сховищі. При пошуку схожих зображень перебрати усі зображення зі сховища, взяти вектор ознак зі сховища і вирахувати коефіцієнт їхньої схожості.

Третій підхід. Збереження зображення в сховищі MongoDB. Попередній прорахунок вектора ознак зберігається в ElasticSearch. Для пошуку схожих зображень робимо запит в ElasticSearch, пошук відбувається на стороні сховища.

Проведення експерименту має відбуватись з використанням цих трьох підходів для визначення їх ефективності.

Для якісного проведення експерименту необхідно виконати ці три тести на різних обсягах даних.

Необхідно зробити заміри часу виконання кожного експерименту. Це і буде головним критерієм для оцінки

Експерименти будуть проводитись на локальному комп’ютері з 8 ядрами процесора M1 Apple Silicon, 8 ГБ оперативної пам'яті, використовуючи Python 3.8, MongoDB версії 4.4, Elasticsearch версії 7.10 та TensorFlow версії 2.4 [17].

4.2 Опис тестових даних

В якості тестових зображень, необхідно було обрати великий дата-сет зображень, гарної якості, одного формату, для якісного і зручного проведення експериментів. Зображення мають бути, як різноманітні, так і мати певну схожість (деякі з них), для наочного бачення коректності роботи програми.

Було обрано дата-сет з зображеннями різних тварин з Kaggle [21]. Даний дата-сет налічує 5400 зображень 90 видів тварин розподілених за окремими директоріями. Зображення якісні в форматі .jpg.

Під час виконання експериментів даний дата-сет буде завантажено в сховище і на основі нього буде відбуватися пошук схожих зображень.

Правильним результатом виконання тестових програм буде, те що результуючі схожі зображення відносяться до одного типу тварин.

Експеримент буде проведено на різних обсягах даних. Визначимо наступні обсяги даних – 10, 20, 50, 100, 500, 1000, 5400. Для цього буде взято необхідну кількість випадкових зображень з цього дата-сету.

4.3 Проведення експериментів

Першим кроком виконання експерименту наповнюємо сховища зображень необхідним обсягом даних [21], ввикористовуючи функцію upload_dataset.

Після цього почергово виконуємо три експерименти:

– використання виключно MongoDB;

– використання виключно MongoDB з попереднім прорахунком векторів ознак;

– використання MongoDB в зв’язці з ElasticSearch.

На кожному етапі робимо замір часу, використовуючи функцію time.time() і виводимо його разом з результатом пошуку схожих зображень.

Код проведення експериментів:

```
upload_dataset(1000)
image_path = get_random_image_path(directory)
```

```
    image = Image.open(image_path)

    start_time = time.time()
    res1 = test1(image)
    end_time = time.time()
    print(f"Test 1 (Mongo without preprocessing) time: {end_time -
start_time:.4f} seconds")
    display_images(image, res1, False)

    start_time = time.time()
    res2 = test2(image)
    end_time = time.time()
    print(f"Test 2 (Mongo with preprocessing) time: {end_time -
start_time:.4f} seconds")
    display_images(image, res2, False)

    start_time = time.time()
    res3 = test3(image)
    end_time = time.time()
    print(f"Test 3 (Mongo + Elastic) time: {end_time - start_time:.4f}
seconds")
    display_images(image, res3, True)
```

Дані експерименти було проведено на різних обсягах даних: 10, 20, 50, 100, 500, 1000, 5400 зображень. Результати зафіксовано. Отримані результати на об’ємах 1000 та 5400 зображень представлено нижче (див. рис. 2 та рис. 3).

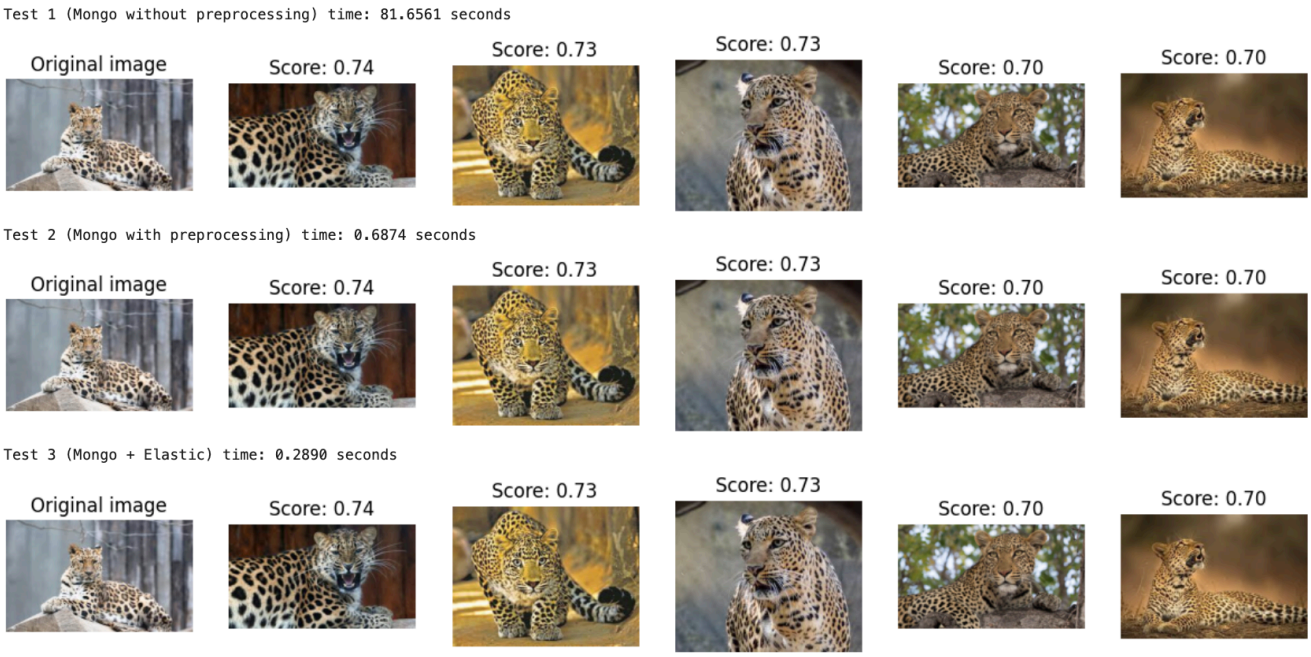

Рисунок 2 – Результат експерименту на 1000 зображеннях (рисунок створено самостійно)

Рисунок 3 – Результат експерименту на 5400 зображеннях (рисунок створено самостійно)

Порівнявши час виконання кожного експерименту, варіант з використанням гібридного сховища зображень виявився найбільш ефективним. Усі зафіксовані результати трьох експериментів представлено в таблиці (див. таб. 2). Результати представлені у секундах.

Таблиця 2 – Результати експериментів (таблицю створено самостійно)

Експеримент	10	20	50	100	500	1000	5400
MongoDB	0.881	1.591	4.008	7.54	38.126	81.656	541.729
MongoDB з попереднім прорахунком	0.08	0.095	0.113	0.143	0.442	0.687	3.254
MongoDB з ElasticSearch	0.156	0.120	0.173	0.138	0.391	0.289	0.208

На кожному етапі проведення експерименту, виконання повторювалось по декілька раз, для фіксування похибки заміру часу. Похибка незначна і не впливає на результати експерименту і зроблені висновки.

4.4 Аналіз результатів

Результати проведених експериментів показують значні відмінності у швидкості пошуку схожих зображень між трьома підходами: використання тільки MongoDB, використання MongoDB з попереднім прорахунком векторів ознак, та використання гібридного сховища у якості комбінації MongoDB з Elasticsearch.

Для розуміння залежності часу виконання пошуку від обсягів даних побудуємо графік (див. рис. 4).

Рисунок 4 – Графік залежності часу виконання від обсягів даних трьох підходів (рисунок створено самостійно)

На графіку наглядно можна побачити лінійну залежність часу виконання від кількості зображень в методі використання тільки MongoDB без попереднього прорахунку. Причому даний підхід досить затратний. Це пояснюється тим, що кожного разу під час пошуку схожих зображень система повинна перебирати всі зображення та обчислювати вектори ознак «на льоту». Це призводить до суттєвих витрат часу при великих обсягах даних. Оцінка складності алгоритму – O(n).

Для більш наглядної оцінки наступних двох підходів, було прийнято рішення виключити цей підхід з графіку (див. рис. 5).

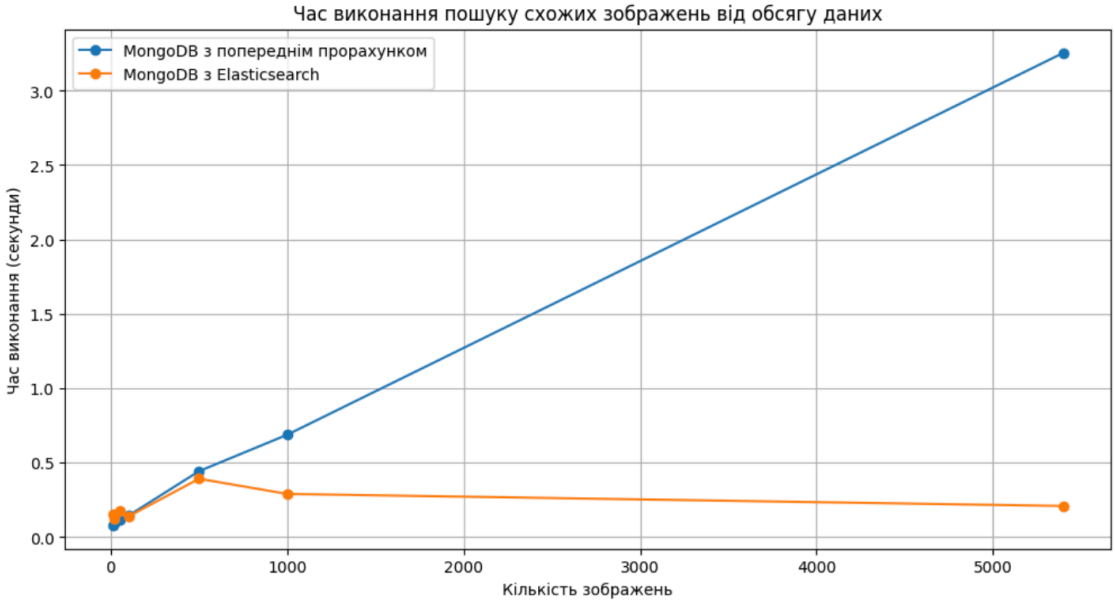

Рисунок 5 – Графік залежності часу виконання від обсягів даних двох підходів (рисунок створено самостійно)

Використання MongoDB з попереднім прорахунком векторів ознак значно покращує час виконання пошуку. Це пояснюється тим, що вектори ознак вже збережені у базі даних і система не витрачає час на їх обчислення під час пошуку. Однак все одно ми бачимо лінійне збільшення часу виконання від обсягів даних. Це пояснюється тим що виконується повний перебір в сховищі для пошуку найбільш схожих зображень. Оцінка складності алгоритму – O(n).

Зовсім інша залежність з використанням гібридного сховища зображень у якості MongoDB з ElasticSearch. На результатах не видно залежності часу від обсягів даних. Це обумовлюється тим що, використання ElasticSearch для пошуку максимально ефективне і за рахунок індексації має логарифмічну складність – O(log(n)) [15], для якої обрані дані об’єми даних значно малі.

Використання гібридного сховища у якості комбінації MongoDB та Elasticsearch дозволяє максимально використати сильні сторони кожної технології. MongoDB забезпечує гнучке та надійне зберігання даних, тоді як Elasticsearch надає потужні можливості ефективного пошуку.

ВИСНОВКИ

В ході проведення дослідження для кваліфікаційної роботи було виконано аналіз предметної області та розглянуто можливості інтеграції машинного навчання для доступу до даних в гібридних сховищах зображення. Описано їх принципи роботи, виділено особливості, основні відмінності, переваги та недоліки.

У процесі роботи було поставлено задачу реалізувати алгоритм пошуку схожих зображень в сховищі, було розроблено та протестовано програмну частину, яка реалізує кілька підходів до зберігання та пошуку зображень, що дозволило зробити висновки щодо їх ефективності та доцільності використання в реальних умовах.

В результаті роботи сформовано звіт, у якому представлено аналіз предметної галузі, основні підходи машинного навчання та гібридних сховищ зображень, описано принципи, на яких вони працюють, вказано на схожість та відмінності між ними. Була сформульована задача, спроектована програмна частина, описана реалізація, і проведено дослідження. Було визначено три підходи проведення експериментів, для отримання і аналізу результатів.

Перш за все, ми визначили, що традиційний підхід, який полягає у зберіганні зображень у єдиному сховищі MongoDB без попереднього розрахунку векторів ознак, є найменш ефективним з точки зору швидкості пошуку. Результати експериментів показали, що цей метод має значні затримки при обробці великих обсягів даних, що робить його непридатним для задач, де потрібна висока швидкість доступу до зображень. А при збільшені обсягів даних час виконання пошуку збільшується, що унеможливлює пошук.

Водночас, використання підходу з попереднім розрахунком векторів ознак та їх зберіганням у MongoDB дозволяє значно скоротити час пошуку схожих зображень. Такий підхід забезпечує швидший доступ до даних за рахунок попередньої обробки зображень і збереження результатів обчислень, що робить його більш ефективним рішенням на невеликих обсягах даних. Однак даний підхід все одно має лінійну залежність часу виконання від обсягів даних, тому не може використовуватись на великих обсягах даних.

Найбільш ефективним виявився гібридний підхід, який поєднує зберігання зображень у MongoDB та векторів ознак у Elasticsearch. Завядки індексації і потужним можливостям пошуку ми досягли логарифмічної залежності часу виконання від обсягів даних, що робить його Цей метод дозволяє не тільки швидко знайти схожі зображення, але й забезпечує гнучкість у пошукових запитах та високу точність результатів завдяки можливостям Elasticsearch. Експерименти підтвердили, що цей підхід має найнижчий час виконання запитів, особливо при обробці великих обсягів даних.

Таким чином, результати дослідження підтверджують доцільність використання гібридних сховищ зображень з інтеграцією методів машинного навчання для оптимізації процесу пошуку. Використання MongoDB та Elasticsearch у поєднанні з попереднім розрахунком векторів ознак дозволяє досягти високої ефективності та точності при роботі з великими обсягами візуальних даних.

Важливим висновком є також те, що правильний вибір методів машинного навчання та технологій зберігання даних значно впливає на результати роботи системи. Використання моделі ResNet50 для екстракції ознак зображень та алгоритму косинусної подібності для порівняння векторів ознак виявилося ефективним рішенням для задачі пошуку схожих зображень.

Загалом, проведене дослідження показало, що інтеграція машинного навчання в процеси управління та обробки даних у гібридних сховищах зображень відкриває нові можливості для покращення ефективності та точності роботи з великими обсягами візуальної інформації. Це дозволяє забезпечити швидкий та надійний доступ до необхідних даних у різних сферах застосування, включаючи медицину, безпеку та електронну комерцію.

Мета завдання досягнута за рахунок визначення у роботі ефективності використання методів машинного навчання для доступу до даних в гібридних сховищах зображень. У результаті роботи було підібрано найефективнішу модель машинного навчання і гібридне сховище зображень для вирішення поставленої задачі.

Результати цього дослідження можуть бути використані для прийняття рішення про те, який підхід використовувати для збереження і організації зображень у сховищі і який алгоритм та модель машинного навчання використовувати для пошуку схожих зображень.

Результати даної кваліфікаційної роботи було опубліковано в журналі «Біоніка інтелекту», який включено до Переліку наукових фахових видань України, категорія «Б», технічні науки, затверджено наказом МОНУ від 02.07.2020 № 886 (дивись Додаток Г).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Використання машинного навчання для оптимізації доступу до даних в гібридному сховищі зображень / І. Кириченко, Г. Терещенко, Д. Панасенко// Біоніка інтелекту. – Харків: ХНУРЕ. – 2023. – № 1 (99). – С. 59-67.

2. James, Blessing & Asagba, P. (2017). Hybrid Database System for Big Data Storage and Management. International Journal of Computer Science, Engineering and Applications. 7. 15-27. 10.5121/ijcsea.2017.7402.

3. Kyrychenko, I., Tereshchenko, G. Proniuk, G., Geseleva, N. “Predicate Clustering Method and its Application in the System of Artificial Intelligence”, 2023 7th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2023), 2023. – CEUR-WS, 2023, ISSN 16130073. - Volume 3396, PP. 395 - 406.

4. Kyrychenko, I., Nazarov, O., Huliiev, N., Avdieiev, O. “Selection of Artificial Neural Networks for Disease Prediction”, 2023 7th International Conference on Computational Linguistics and Intelligent Systems (COLINS-2023), 2023. – CEUR-WS, 2023, ISSN 16130073. - Volume 3387, PP. 236-248.

5. Berahmand, Kamal & Daneshfar, Fatemeh & Salehi, Elaheh & Li, Yuefeng & Xu, Yue. (2024). Autoencoders and their applications in machine learning: a survey. Artificial Intelligence Review. 57. 10.1007/s10462-023-10662-6.

6. Tabian, I., Fu, H., & Khodaei, Z.S. (2019). A Convolutional Neural Network for Impact Detection and Characterization of Complex Composite Structures. Sensors, 19(22), 4933. DOI: 10.3390/s19224933.

7. Asokan, Anju & Jude, Anitha & Patrut, Bogdan & Danciulescu, Dana & D, Jude. (2020). Deep Feature Extraction and Feature Fusion for Bi-temporal Satellite Image Classification. Computers, Materials & Continua. 66. 373-388. 10.32604/cmc.2020.012364.

8. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778. doi:10.1109/CVPR.2016.90.

9. Meghana, Avuthu Sai. "Age and Gender prediction using Convolution,

ResNet50 and Inception ResNetV2." International Journal of Advanced Trends in Computer Science and Engineering 9, no. 2 (April 25, 2020): 1328–34. http://dx.doi.org/10.30534/ijatcse/2020/65922020.

10. Understanding Cosine Similarity and Cosine Distance in Depth. URL: https://pub.aimind.so/understanding-cosine-similarity-and-cosine-distance-in-depth-cc91eac3ef2 (дата звернення: 29.04.2024).

11. Matallah, Houcine & Belalem, Ghalem & Bouamrane, K.. (2020). Evaluation of NoSQL Databases: MongoDB, Cassandra, HBase, Redis, Couchbase, OrientDB. International Journal of Software Science and Computational Intelligence. 12. 71-91. 10.4018/IJSSCI.2020100105.

12. Znakhur, Serhii & Znakhur, Liudmyla. (2022). Similar goods search based on FAISS. Bulletin of Kharkov National Automobile and Highway University. 40. 10.30977/BUL.2219-5548.2022.96.0.40.

13. Docker Compose Documentation. URL: https://docs.docker.com/compose/ (дата звернення: 05.05.2024).

14. MongoDB Documentation. URL: https://www.mongodb.com/docs/manual/ (дата звернення: 05.05.2024).

15. Wong, Wai-Tak. (2019). Advanced Elasticsearch 7.0: A practical guide to designing, indexing, and querying advanced distributed search engines.

16. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3), 211-252. doi:10.1007/s11263-015-0816-y.

17. Keras Applications: ResNet. URL: https://keras.io/api/applications/resnet/ (дата звернення: 20.04.2024).

18. Pang, Bo, Erik Nijkamp, and Ying Nian Wu. "Deep Learning With TensorFlow: A Review." Journal of Educational and Behavioral Statistics 45, no. 2 (September 10, 2019): 227–48. http://dx.doi.org/10.3102/1076998619872761.

19. Nelli, Fabio. (2018). The NumPy Library: With Pandas, NumPy, and Matplotlib. 10.1007/978-1-4842-3913-1_3.

20. Scikit-learn: Machine Learning in Python. URL: https://scikit-learn.org/stable/user_guide.html (дата звернення: 05.05.2024).

21. Animal Image Dataset (90 Different Animals) on Kaggle. URL: https://www.kaggle.com/datasets/iamsouravbanerjee/animal-image-dataset-90-different-animals (дата звернення: 04.05.2024).