Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ ПОШУКУ ЗОБРАЖЕНЬ В ЛОКАЛЬНОМУ СХОВИЩІ ПЕРСОНАЛЬНОГО КОМП'ЮТЕРА ЗА ТЕКСТОВИМ ЗАПИТОМ
(тема)

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

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Яковлева О. В.
(посада, прізвище, ініціали)

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Кошелю Владиславу Олеговичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка застосунку для пошуку зображень в локальному сховищі персонального комп'ютера за текстовим запитом

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 24 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотеки комп'ютерного зору сімейства CLIP, мови програмування Python і TypeScript, база даних Qdrant, фреймворк Electron, бібліотеки FastAPI, NumPy, PyTorch, середовище створення діаграм PlantUML.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз існуючих підходів та застосунків для пошуку зображень за текстовим запитом.
2. Дослідження можливостей сучасний мультимодальних моделей для побудови спільного простору зображень і текстів.
3. Формування набору даних для проведення експериментів, який міститиме зображення та декілька варіантів їхніх описів.
4. Проведення порівняльного аналізу моделей для отримання векторів ознак, використовуючи методи перетворення розмірності векторів та різні підходи візуалізації ознак з метою наочного аналізу схожості між векторними представленнями зображень і відповідними текстовими описами.
5. Вибір інструмента векторного пошуку та моделі отримання векторів ознак зображень та текстових описів для виконання роботи.
6. Проєктування і розробка алгоритму пошуку зображень і застосунку.
7. Аналіз якості пошуку та швидкодії роботи програмного продукту.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми пошуку зображень за текстовими запитами в локальному середовищі, наявні методи та програмні рішення до вирішення задачі пошуку, постановка задачі, аналіз кількості зображень на комп'ютерах користувачів в рамках проведеного опитування, сформований набір даних для експериментів, структура моделей отримання векторів ознак зображень та текстових описів, дослідження з векторами ознак, перевірка подібності векторів ознак зображень та їхніх текстових описів, висновок щодо обрання моделі, алгоритм пошуку на основі векторів, визначення порогу для прийняття рішення відповідності пошуковому запиту, діаграми варіантів використання та взаємодії компонентів застосунку, ілюстрація роботи застосунку, аналіз точності роботи застосунку.

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

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів та моделей	15.04.25-20.04.25	
5	Проєктування застосунку	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Яковлева О. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 87 с., 15 табл., 43 рис., 3 дод., 59 джерел.

ПОШУК ЗОБРАЖЕНЬ, ТЕКСТОВИЙ ЗАПИТ, КОСИНУСНА ПОДІБНІСТЬ, CLIP-МОДЕЛІ, ОФЛАЙН ЗАСТОСУНОК, ЛОКАЛЬНЕ ЗБЕРІГАННЯ, ІНДЕКСАЦІЯ ЗОБРАЖЕНЬ, ОБРОБКА ЗОБРАЖЕНЬ, OPENAI CLIP, OPENCLIP, APPLE MOBILECLIP, T-SNE, PCA, UMAP, FAISS, ELASTICSEARCH, QDRANT.

Об’єктом роботи є процес пошуку зображень за їх змістом на основі текстового запиту.

Метою роботи є розробка застосунку для пошуку зображень у локальному сховищі персонального комп’ютера за текстовим запитом.

В роботі було проведено аналіз застосунків з організації зображень на пристрої, методів отримання змістовної інформації з зображень, моделей штучного інтелекту для поєднання зображень з текстовими описами, було проведено експерименти щодо роздільності класів та пошуку порогу міри подібності тексту з зображенням для коректної роботи пошукової системи.

У результаті роботи здійснена програмна реалізація системи для пошуку зображень, збережених в локальному сховищі персонального комп’ютера, за текстовим описом.

IMAGE SEARCH, TEXT QUERY, COSINE SIMILARITY, CLIP MODELS, OFFLINE APPLICATION, LOCAL STORAGE, IMAGE INDEXING, IMAGE PROCESSING, OPENAI CLIP, OPENCLIP, APPLE MOBILECLIP, T-SNE, PCA, UMAP, FAISS, ELASTICSEARCH, QDRANT.

The subject of this work is the process of image retrieval based on their content using textual queries.

The objective of the work is to develop an application for searching images stored in the local storage of a personal computer using a text query.

The work includes an analysis of applications for organizing images on a device, methods for extracting meaningful information from images, artificial intelligence models that connect images with textual descriptions, and experiments on class separability and determining the similarity threshold between text and image to ensure the proper functioning of the search system.

As a result of the work, a software implementation of a system was developed that enables image search within the local storage of a personal computer based on textual descriptions.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Гб – гігабайти

датасет – сукупність даних, яку використовують для тренування, оцінювання та тестування алгоритмів та моделей

ембединг – векторні представлення тексту, зображень чи інших об’єктів у багатовимірному просторі

млн – мільйони

млрд – мільярди

ОС – операційна система

ВСТУП

У сучасних умовах інформаційного суспільства обсяг цифрового медіаконтенту зростає експоненційно. Зображення відіграють ключову роль у наукових дослідженнях, комерційних проєктах, освітній діяльності та повсякденному спілкуванні. Ефективний пошук візуальних даних дозволяє прискорити обробку інформації, вдосконалити аналіз зображень та підвищити якість ухвалення рішень у різноманітних галузях – від медицини й промислового виробництва до соціальних мереж і архівних систем.

Разом із значним розповсюдженням цифрових камер та доступністю зберігання інформації на персональних комп’ютерах і мобільних пристроях виникає проблема організації величезних масивів файлів. Користувачі стикаються із затримками на пошук необхідних зображень, дублюванням даних і неможливістю швидко формувати тематичні колекції. Традиційні файлові менеджери обмежені за функціональністю текстового індексування, що ускладнює роботу з мультимедіа, особливо коли ключовим є змістовий контекст зображення.

Водночас активне поширення хмарних сервісів, у тому числі і для організації власних медіафайлів, створює додаткові ризики для безпеки персональних даних. Публікація чутливої інформації або завантаження великої кількості особистих фотографій на віддалені сервери піддає користувачів ймовірності несанкціонованого доступу та втрати контролю над власним контентом. Через це зростає потреба у розробці локальних інструментів, які забезпечують повноцінний пошук за текстовим запитом без використання зовнішніх ресурсів, водночас зберігаючи конфіденційність і автономність зберігання.

Актуальність роботи полягає у тому, що сучасні користувачі щоденно накопичують тисячі зображень – від особистих фото до професійних ілюстрацій, і традиційні методи пошуку (за назвою файлу чи тегуванням вручну) не відповідають темпу та обсягам таких даних. В умовах, коли обсяг

світового візуального контенту швидко зростає, відсутність інструментів швидкої і зручної семантичної індексації локальних колекцій призводить до значних часових витрат на пошук і повторного завантаження раніше знайдених файлів.

Крім того, розвиток технологій обробки природної мови та комп’ютерного зору сприяє появі високоефективних алгоритмів семантичного пошуку, які традиційні файлові менеджери наразі не використовують. Реалізація локального застосунку, здатного інтерпретувати текстові запити і знаходити відповідні зображення без звернення до зовнішніх серверів, дозволяє поєднати інноваційні методи машинного навчання з гарантією збереження конфіденційності.

Окрему вагу набуває відповідність регуляторним вимогам щодо захисту персональних даних (наприклад, GDPR або аналогічні національні стандарти), де локальні рішення знижують ризики витоку інформації та утримують дані в межах користувацької машини. Таким чином, розробка швидкого й надійного застосунку для пошуку зображень у локальному сховищі є критичною задачею для забезпечення ефективного управління цифровими активами, збереження приватності та оптимізації робочих процесів у різних сферах людської діяльності.

Дана робота присвячена створенню програмного продукту для пошуку зображень на персональному комп’ютері за текстовими запитами. Особлива увага приділяється дослідженню і вибору моделі отримання векторів ознак зображень і текстів, що якнайкраще підходить для вирішення задачі, та пошуку оптимального порогу фільтрації нерелевантних зображень.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ПИТАННЯ ПОШУКУ ЗОБРАЖЕНЬ ЗА ТЕКСТОВИМ ЗАПИТОМ

1.1 Підходи до пошуку зображень

Пошук зображень є одним із важливих питань в галузі комп'ютерного зору та інформаційних технологій. З розвитком технологій існує низка підходів, які забезпечують ефективний пошук і класифікацію візуального контенту.

Один із перших методів полягає у використанні метаданих. Це підхід, який базується на текстовій інформації, пов'язаній із зображенням, наприклад, назви, теги або інша схована в метадані файлу інформація. Хоча цей метод є простим у реалізації, він має обмеження, пов'язані з суб'єктивністю, потенційною неповнотою описів та неможливістю врахування контенту в контексті пошуку.

З розвитком технологій з'явилися методи, що базуються на аналізі вмісту зображень. Пошук за зразком-зображенням використовує такі ознаки, як гістограми кольорів, дескриптори та інші візуальні характеристики [1, 2]. Це дозволяє здійснювати пошук на основі подібності до заданого зразка.

Важливим проривом стало впровадження сіамських нейронних мереж, які можуть навчатися розпізнавати подібність між різними зображеннями. Ці мережі допомагають ідентифікувати зображення, що мають схожі елементи, навіть якщо вони відрізняються за кольором або освітленням.

Нарешті, мультимодальні моделі глибокого навчання здатні зв'язувати текстову та візуальну інформацію, що значно покращує можливості пошуку. Ці моделі забезпечують контекстуальне розуміння і можуть знаходити зображення на основі текстових запитів, які не обмежуються явними метаданими.

1.2 Аналіз програмного забезпечення для вирішення задачі пошуку зображень

Пошук зображень пройшов шлях від класичних методів, заснованих на метаданих, виявлених ознаках та ключових словах, до сучасних рішень зі штучного інтелекту. Сьогодні контекстуальний пошук за допомогою штучного інтелекту (ШІ) дозволяє знаходити зображення за текстовим описом без попередньої розмітки або знання специфічних метаданих чи тегів, що робить процес простішим і зручнішим.

В таблиці 1.1 наведено порівняння сучасних сервісів для систематизації, каталогізації й пошуку зображень як в локальному, так і в хмарному сховищі [3].

Таблиця 1.1 – Класифікація інструментів на основі підходу до пошуку зображень

Метадані	**Ручне тегування**	**Тегування з ШІ**	**Контекстний ШІ-пошук**
Spotlight [4] 🖥️ FREE	Adobe Bridge [7] 🖥️ FREE	Amazon [12] ☁️ 💿	Google Photos [16] ☁️ 💿
Dropbox [5] ☁️ 💿	FastStoneImage [8] 🖥️ FREE	Apple Photos [13] 🖥️ ☁️ FREE	Excire Foto [15] 🖥️ ⏳
Пошук Windows 🖥️ FREE	XnView MP [9] 🖥️ FREE	Adobe Lightroom [14] ☁️ ⏳	Lenso [19] ☁️ 💰
Microsoft Photos [6] 🖥️ FREE	PicaJet [10] 🖥️ 💿	Excire Foto [15] 🖥️ ⏳	
	digiKam [11] 🖥️ FREE	Google Photos [16] ☁️ 💿	
		Dropbox 💰 [5]	
		Microsoft OneDrive [17] ☁️ 💿	
		ACDSee Photo Studio [18] 🖥️ ☁️ ⏳	

У таблиці прийняті наступні позначення:

– десктопний застосунок – 🖥️;

– хмарний застосунок – ☁️;

– безкоштовний застосунок – 🆓;

– безкоштовний застосунок із обмеженнями – 💿;

– платний застосунок з безкоштовним пробним періодом – ⏳;

– платний застосунок – 💰.

PicaJet – це локальний менеджер зображень, який забезпечує пошук метаданих і фільтрацію за параметрами. Підтримує функції групування, легкий імпорт з різних джерел, управління категоріями та тегами. Основним мінусом є відсутність оновлень з 2014 року, що робить його застарілим.

Google Photos – це хмарний сервіс із розширеними функціями пошуку на базі штучного інтелекту. Система індексує завантажені зображення й дозволяє користувачам здійснювати пошук за допомогою природної мови. Перевагою є автоматичне резервне копіювання, але необхідність підключення до Інтернету та проблеми з конфіденційністю можуть бути мінусами.

Також варто згадати ZeroSearch – алгоритм, створений науковцями для пошуку зображень на основі текстових описів. Цей алгоритм не є програмним рішенням, але документ описує, як наразі просуваються наукові дослідження з цього питання.

Найфункціональнішим застосунком в ході аналізу було визначено Excire Foto – локальний фотоорганайзер, що використовує технології машинного навчання для аналізу зображень, включаючи пошук за текстом. Він забезпечує швидку обробку, конфіденційність і роботу в автономному режимі, а також підтримує пошук за вільним текстовим описом, геолокацією, схожими зображеннями та розпізнаванням облич. Недоліком є комерційна, платна модель розповсюдження без відкритого коду.

При першому завантаженні цього застосунку, відображається інструкція користувача. Наприклад, там вказано таку важливу інформацію:

– деталі про додавання фото (рис. 1.1);

– деталі роботи пошуку за вільним текстовим запитом (рис. 1.2);

– деталі про редагування додаткових метаданих для поліпшеної фільтрації чи групування фото;

– деталі про створення колекцій, слайд-шоу та організації фотоколекцій.

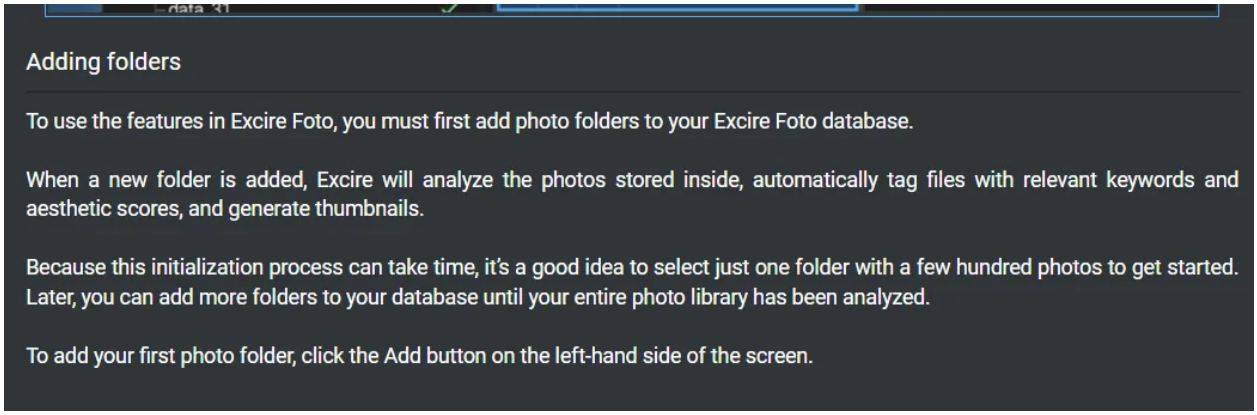


Рисунок 1.1 – Опис можливості додавання папок

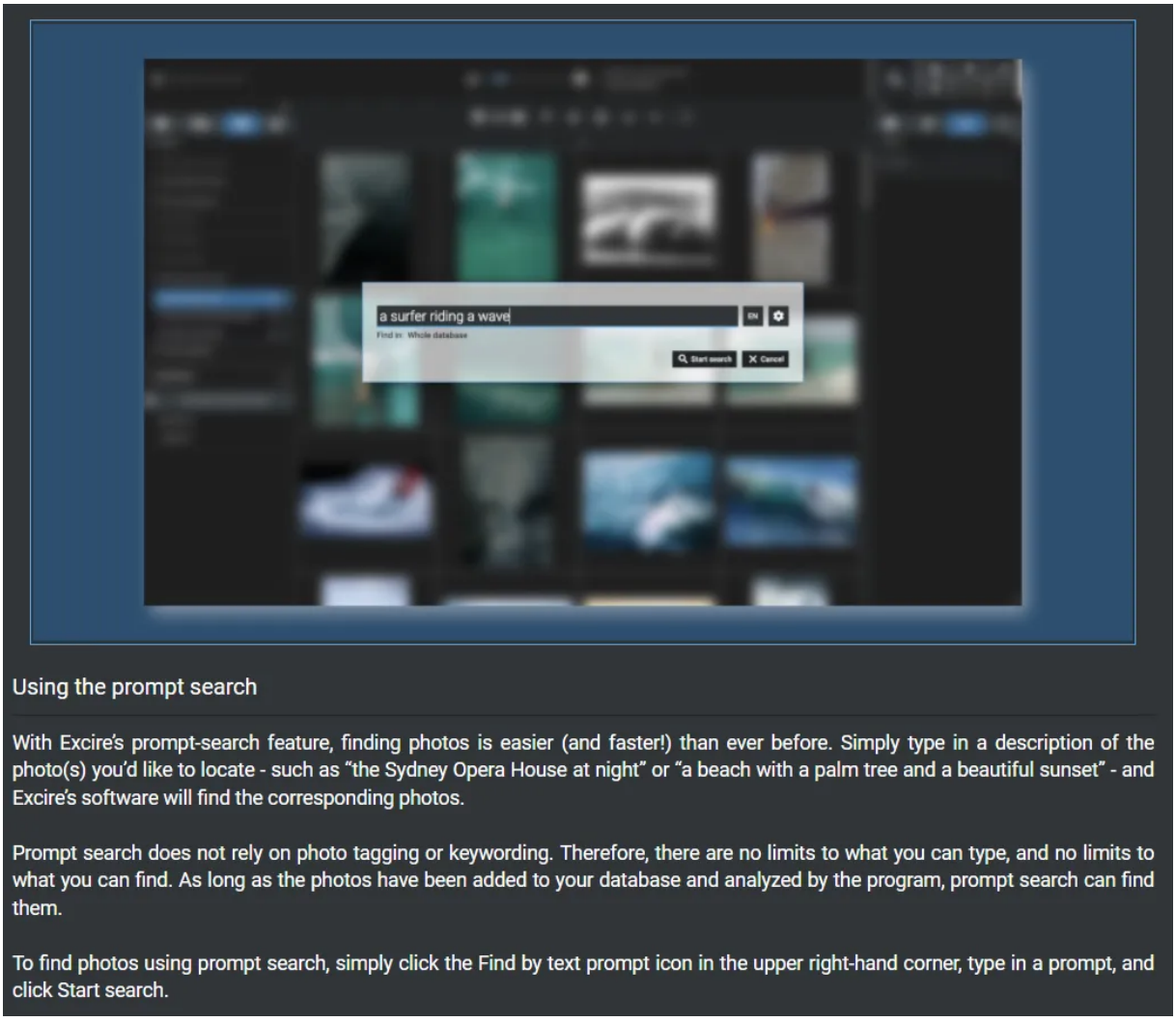


Рисунок 1.2 – Опис можливості пошуку за вільним текстовим запитом

Було проаналізовано швидкодію індексації та витрати дискового простору після процесу індексації. Нижче (рис. 1.3) наведено згенерований звіт про завершення індексації каталогу з 3638 фотографіями розміром 16 Гб.

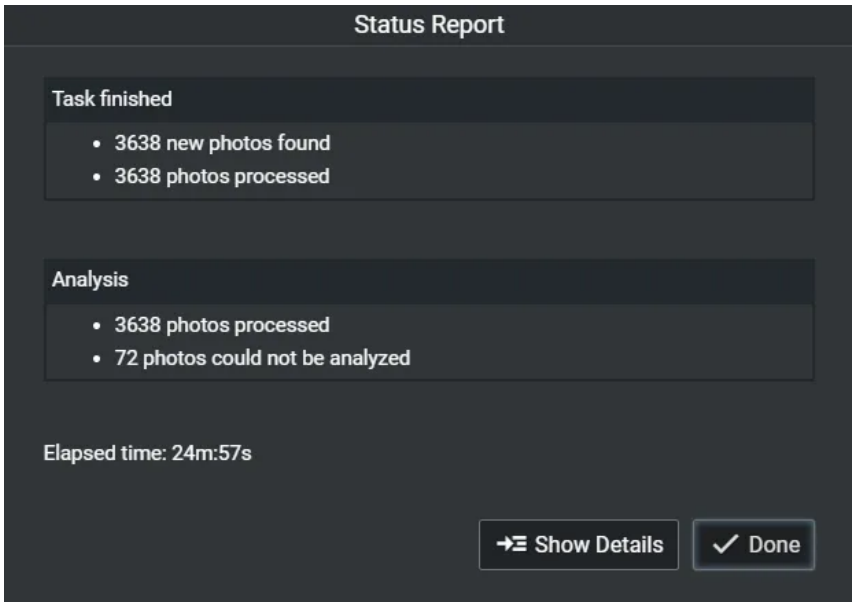


Рисунок 1.3 – Звіт про індексацію

Процес обробки каталогу складається з п’яти етапів, деталі яких є прихованими для користувача. Перші 4 етапи – це аналіз метаданих, етап №5 – детальний аналіз кожного зображення з генерацією інформації для подальшого функціонування пошуку. Етапи №1-4 відбулися за 4 хвилини загалом, етап №5 – за 20 хвилин. На етапі швидкого аналізу щосекундно оброблялось приблизно 20 фотографій. На останньому ж етапі – оброблялось лише 3 зображення. Розмір вільного дискового простору зменшився з 82,4 Гб до індексації до 81,3 Гб після неї, розмір даних індексації складає 1,1 Гб.

Отже, в ході аналізу було виявлено слабкі місця, що можуть бути вдосконалені в поточній роботі:

– витрата дискового простору на збереження індексованих даних;
– швидкодія процесу індексації.

Проведений аналіз показав, що лише небагато сучасних інструментів підтримують контекстуальний пошук зображень за текстовими запитами, і

майже всі з них є платними. Серед таких рішень можна виділити Google Photos, Lenso.ai та Excire Foto, причому лише Excire Foto працює у локальному режимі. Таким чином, на сьогодні спостерігається дефіцит локальних безкоштовних застосунків для пошуку зображень в локальному сховищі комп'ютера на основі текстового запиту.

1.3 Методи пошуку зображень за контентом

1.3.1 Методи пошуку на основі зображення-зразка

Пошук зображень на основі зразка є важливою задачею в комп'ютерному баченні. Існує безліч методів, які дозволяють ефективно вирішувати цю задачу, починаючи від класичних алгоритмів до сучасних підходів, заснованих на глибокому навчанні.

До появи глибоких нейронних мереж пошук на основі зображення-зразка базувався на ознаках зображень, що отримувались за допомогою розрахунків за евристичними алгоритмами.

Одним із важливих методів були визначення дескрипторів зображень. Такими алгоритмами є, наприклад, SIFT (Scale-Invariant Feature Transform), SURF (Speeded Up Robust Features) або ORB (Oriented FAST and Rotated BRIEF). Вони виділяють ключові точки на зображенні, після чого створюючи дескриптори для опису цих точок. Потім ці дескриптори використовуються для порівняння з іншими зображеннями. Наприклад, SIFT є методом, який забезпечує стабільність до змін масштабу та обертання, що робить його надзвичайно корисним для різноманітних застосувань. SURF і ORB є швидшими альтернативами SIFT з високою точністю [20, 21].

Інший підхід будувався на глобальних ознаках зображення, які описують зображення цілком, на відміну від локальних дескрипторів, що характеризують зображення у ключових точках. До таких глобальних характеристик належать гістограми яскравості та кольору, моменти

зображення, наприклад, моменти Церніке, інваріанти Фур'є, ознаки Уолша, вейвлет-коефіцієнти, гістограми напрямків градієнтів, матриці збігів [22], характеристики Лавса [23, 24] та багато інших [25]. Деякі з них описують форму (інваріанти Фур'є, ознаки Уолша), деякі розподіл кольору або яскравості, інші текстурні особливості. Частина з них, а саме Церніке, перетворення Фур'є, мають інваріантність до масштабування, обертання та зсуву, що робить їх ефективними для розпізнавання об'єктів у різних позиціях, інші потребують ще додаткової обробки для забезпечення інваріантності.

З появою глибокого навчання підходи до пошуку зображень значно змінилися. Згорткова нейронна мережа CNN дозволяє витягувати високорівневі ознаки з зображень, які є інваріанті до геометрічних перетворень, шумів та стійкі до інших змін. Наприклад, ResNet [26] чи EfficientNet [27] можна використовувати для отримання векторів ознак, обрізаючи останній шар мережі та порівнюючи вектори за допомогою метрик для визначення найбільш подібних зображень.

На основі таблиці 1.2 можна зробити висновок, що отримання ознак за CNN є найбільш перспективним, але такий підхід вимагає наявності великого набору даних для навчання та потужних обчислювальних ресурсів. Тому в багатьох прикладних задачах досі актуальним залишається використання класичних методів або їх комбінації з нейромережевим підходом [28, 29].

Таблиця 1.2 – Головні властивості деяких методів пошуку на основі зображення зразка

Метод	**Що описує**	**Інваріант-ність до геометричних трансформа-цій**	**Інваріант-ність до шумів**	**Як забезпечити інваріантність**	**Рекомендо-вано застосо-вувати**
1	**2**	**3**	**4**	**5**	**6**
SURF	Локальні ознаки (ключові точки та їх околиці)	Масштаб, обертання, зсув	Стійкий до шуму	Уже вбудована в алгоритм (вбудована шкала і орієнтація у фільтрах)	Баланс між точністю та швидкістю

Продовження таблиці 1.2

1	2	3	4	5	6
SIFT	Локальні ознаки (ключові точки та їх околиці)	Масштаб, обертання, зсув	Стійкий до помірного шуму	Уже вбудована в алгоритм (масштабно-орієнтована нормалізація)	Точне порівняння об’єктів у змінних умовах (стабільність важливіша за швидкість)
ORB	Локальні ознаки (ключові точки та їх околиці)	Масштаб, обертання (частково), зсув	Менш стійкий, ніж SIFT/SURF	Часткова інваріантність вбудована у алгоритм (FAST для точок + BRIEF з ротаційною нормалізацією на основі моментів яскравості)	Потреба в обмежених обчислювальних ресурсах, реальний час
Гістограма яскравості або кольору	Глобальний розподіл яскравості або кольору	Масштаб, обертання, зсув	Частково	Попереднє усереднення, нормалізація гістограми	Груба попередня фільтрація зображень
Моменти Церніке	Форма об’єкта	Масштаб, обертання, зсув	Часткова	Попередня обробка: центрування мас, нормалізація масштабу	Пошук за формою (тільки у задачах, де об’єкт розташовано на однорідному фоні)
Інваріанти Фур’є	Форма (контур об’єкта)	Масштаб, обертання, зсув	Помірна	Попередня обробка: виділення замкнутого контуру, центроване представлення Постобробка: нормалізація амплітуд	Порівняння контурів та силуетів (тільки у задачах, де об’єкт розташовано на однорідному фоні)
Ознаки Уолша	Глобальні геометричні/ структурні особливості	Частково масштаб та зсув	Стійкі до помірного шуму	Попереднє центрування та масштабування до фіксованого розміру	Аналіз бінарних зображень, технічних схем
Вейвлет-коефіцієнти	Частотна структура та текстура	Частково	Стійкі до шуму	Використання вейвлетів, інваріантних до обертання (наприклад, steerable wavelets)	Аналіз текстури, компресія

Продовження таблиці 1.2

1	2	3	4	5	6
Ознаки Лавса (Laws)	Текстура через фільтрацію шаблонами	-	Обмежена	Попередня обробка: нормалізація яскравості. Обчислення матриць в різних напрямках.	Аналіз однорідності текстури, класифікація матеріалів
Гістограми напрямків градієнтів (HOG)	Структурні характеристики (краї та напрямки)	Зсув, частково масштаб	Стійкий	Попередня обробка: виділення країв, розбиття зображення на блоки, вирівнювання освітлення. Постобробка: нормалізація блоків та адаптивних вікон під час обчислення ознак	Розпізнавання об'єктів (особливо людей), аналіз сцени
Матриці збігів	Текстура, просторові співвідношення яскравостей	-	-	Попередня обробка: квантування зображення за рівнями яскравості. Обчислення матриць в різних напрямках.	Аналіз текстури на малих зображеннях
CNN ознаки (глибокі ознаки)	Високорівневі абстрактні ознаки (форма, текстура, контекст)	Так (залежить від навчання)	Так (навчання на зашумлених даних)	Аугментація даних при навчанні, використання згорткової архітектури	Універсальне представлення зображень для класифікації, пошуку та кластеризації

1.3.2 Методи пошуку на основі текстового опису контенту

Традиційних методів пошуку часто виявляється недостатньо. Стрімкий розвиток мультимодальних моделей штучного інтелекту значно покращив можливості пошуку, дозволяючи поєднувати різного роду інформацію, насамперед текстову з візуальною, для отримання більш точних результатів.

Мультимодальні моделі, такі як CLIP, стали проривом у цій галузі. CLIP

поєднує текстові та візуальні ембединги в спільний простір [30]. Це дозволяє знаходити зображення, відповідні заданому текстовому опису, без необхідності ручного тегування. Це досягається шляхом навчання на великій кількості пар «зображення-текст», після якого модель стає розуміти контекст обох типів даних. Важливо зазначити, що CLIP є комерційною розробкою, але існують і відкриті для аналізу та модифікацій аналоги, наприклад, OpenCLIP, які надають можливість дослідникам, розробникам та іншим охочим використовувати подібні технології безкоштовно [31].

Іншою важливою моделлю є ALIGN від Google Research, яка також об'єднує текстові та візуальні ембединги в спільному просторі [32]. ALIGN, як і CLIP, використовує контрастивне навчання в парах «зображення-текст» на величезних наборах даних, що дозволяє їй досягати високої точності в задачах пошуку. Обидві моделі схожі за принципом роботи, але головна різниця полягає в більшому за розміром та менш вичищеному наборі даних для навчання моделі ALIGN. Цей пункт є критичним для порівняння моделей, адже CLIP навчався на 400 мільйонах пар, а ALIGN – на 1,8 мільярдах пар.

Ще одним сучасним і перспективним підходом є модель Florence-2 від Microsoft [33]. Florence-2 розроблена як універсальна мультимодальна система, що створена насамперед для вирішення різноманітних візуально-лінгвістичних задач: опис зображень, відповідь на питання стосовно зображень, побудова детальних візуальних ембедингів тощо. Florence-2 масштабовано навчалася на великому й різноманітному наборі пар «зображення-текст», що забезпечує глибше розуміння контексту та високі результати як у стандартних задачах пошуку, так і в інших мультимодальних застосуваннях. За своїми властивостями вона наближається до моделей CLIP і ALIGN, проте орієнтується на ще ширший спектр застосувань.

Крім того, модель ViLT (Vision-and-Language Transformer) пропонує інший архітектурний підхід [34]. На відміну від вищевказаних моделей, які окремо будують текстові та візуальні ембединги, а потім співставляють їх у спільному просторі, ViLT об'єднує обробку тексту та зображення вже на рівні

трансформера. В цій моделі зображення передається у вигляді патчів (малих фрагментів), які, разом із токенізованим текстом, подаються на вхід одному й тому ж трансформеру. Це дозволяє моделі безпосередньо вивчати взаємодію між текстовою та візуальною інформацією. Такий підхід суттєво прискорює процес обробки – ViLT не має окремої глибокої нейромережі для зображень, а натомість використовує лінійне перетворення з пікселів у ембединги патчів. Незважаючи на меншу обчислювальну складність, ViLT демонструє конкурентні результати на задачах пошуку зображень за текстом та візуально-лінгвістичних завданнях, що свідчить про достатню ефективність інтеграції тексту й зображень на рівні єдиної трансформерної архітектури.

Для ефективного пошуку зображень за текстовим запитом необхідно мати швидкі та масштабовані інструменти для обробки великих обсягів даних. Одним з них є Faiss, бібліотека бази даних від Meta Research, яка спеціалізується на швидкому пошуку схожих векторів [35]. Завдяки різним варіантам індексації Faiss дозволяє ефективно зберігати, оновлювати та шукати в різних за параметрами наборах векторних даних (рис. 1.4).

Іншим популярним рішенням є Elasticsearch, що забезпечує потужний пошук та аналіз даних у реальному часі [36]. Хоча Elasticsearch традиційно використовується для текстового пошуку, його можливості можна розширити для роботи з векторними даними за допомогою плагінів, таких як Elasticsearch KNN. Ще однією базою даних, що набирає свою популярність, є Qdrant зі спеціалізацією саме на векторному й комбінованому пошуку з метаданими [37]. Завдяки своїй архітектурі, вона може обробляти великі обсяги даних, що робить її привабливим вибором для задач, пов’язаних з мультимодальними ембедингами.

Використання мультимодальних моделей і векторних баз даних відкриває нові горизонти для пошуку зображень на основі текстового опису. Ці технології забезпечують не лише точність, але й гнучкість, дозволяючи адаптуватися до різноманітних сценаріїв використання. Вони є важливим кроком у розвитку систем штучного інтелекту, які можуть ефективно

взаємодіяти з різними типами даних, забезпечуючи користувачам багатий та інтуїтивно зрозумілий досвід.

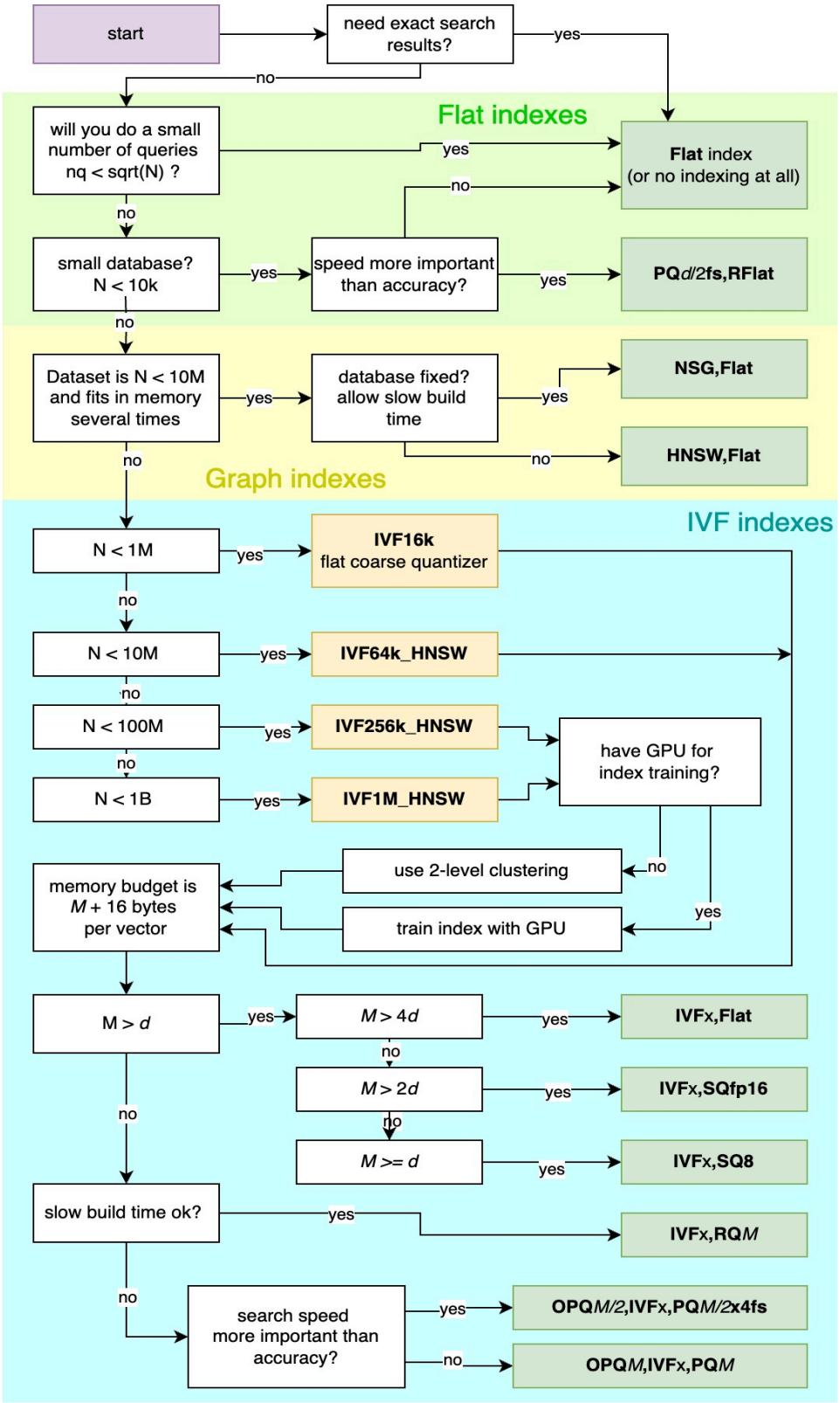


Рисунок 1.4 – Рекомендований алгоритм для вибору режиму індексації в бібліотеці Faiss

1.4 Постановка задачі

Отже, згідно вказаному вище, на сьогодні існує нестача локальних безкоштовних застосунків для пошуку зображень на комп’ютері з можливістю контекстуального пошуку на основі текстового запиту. Більшість наявних рішень або є комерційними, або працюють лише у хмарному середовищі, що може викликати питання конфіденційності та потребує постійного доступу до Інтернету. Це створює актуальну потребу в розробці застосунку, здатного здійснювати пошук зображень за їх змістом у локальному сховищі персонального комп’ютера за текстовим запитом на основі мультимодальних моделей штучного інтелекту.

Об’єктом роботи є процес пошуку зображень за їх змістом на основі текстового запиту.

Метою роботи є розробка застосунку для пошуку зображень у локальному сховищі персонального комп’ютера за текстовим запитом.

Для досягнення цієї мети необхідно вирішити такі завдання:

– провести аналіз існуючих підходів та програмного забезпечення для пошуку зображень за текстовим запитом;

– дослідити можливості сучасних мультимодальних моделей для побудови спільного простору зображень і текстів;

– сформувати набір даних для проведення експериментів, що буде містити зображення та декілька варіантів їх опису;

– виконати порівняльний аналіз моделей для отримання ембедингів та обрати модель, яка якнайкраще підходить для вирішення завдання пошуку зображень за текстовим описом (використовуючи методи перетворення розмірності векторних просторів, таких як *t*-SNE, PCA, UMAP, та різні підходи візуалізації ознак, наприклад, графіків розсіювання, теплових карт і кластерних діаграм, наочно проаналізувати схожість між векторними представленнями зображень та відповідними текстовими описами; вивчити питання якості кластеризації, використовуючи метрику косинусної подібності

та метод силуетів);

– визначити значення порогу для прийняття рішення про відповідність зображення текстовому опису;

– дослідити інструменти векторного пошуку, зокрема Faiss, Elasticsearch, Qdrant тощо;

– розробити алгоритм пошуку зображень за текстовим описом;

– спроєктувати та реалізувати десктопний застосунок для пошуку зображень за текстовим запитом у локальному сховищі персонального комп’ютера.

2 РОЗРОБЛЕННЯ ТА АНАЛІЗ МОДЕЛЕЙ І АЛГОРИТМІВ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ ПОШУКУ ЗОБРАЖЕНЬ ЗА ТЕКСТОВИМ ЗАПИТОМ

2.1 Аналіз кількості зображень на комп'ютерах користувачів

З грудня 2024 року по травень 2025 року було проведено міні-опитування щодо визначення кількості зображень на персональних комп'ютерах потенційних користувачів програмним продуктом (рис. 2.1).

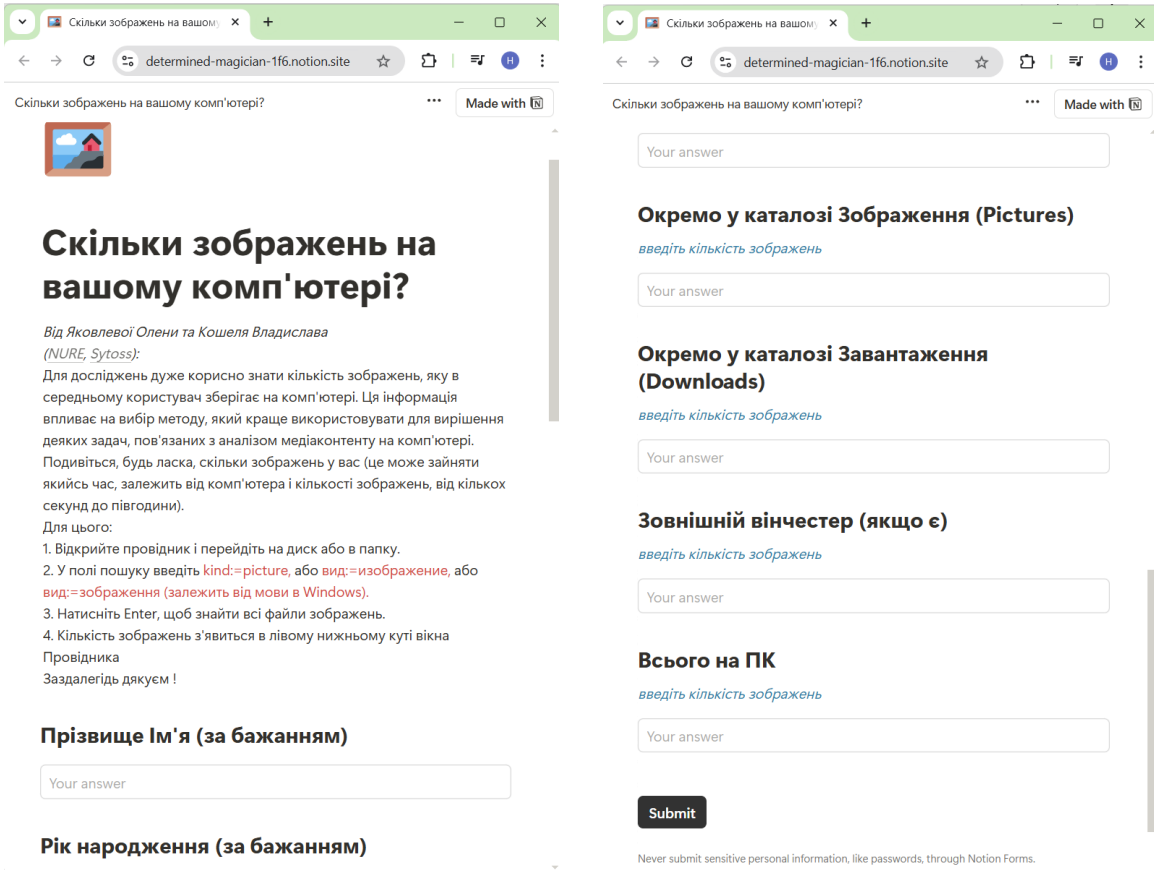


Рисунок 2.1 – Ілюстрація сторінки опитування

Опитуваним було запропоновано дізнатись точну кількість зображень на пристрої під керуванням ОС Windows, порівняти цю кількість із кількістю картинок в каталогах «Зображення», «Завантаження», на дисках C, D і зовнішніх накопичувачах. Для відображення точної кількості зображень в

системі існує команда фільтрації в пошуку Windows, що відбирає лише зображення будь-якого розміру, формату та з будь-яких шляхів – «вид:=зображення».

Для тих, хто не мав пристроїв під керуванням ОС Windows, було запропоновано приблизно оцінити свою медіатеку на різних пристроях і надати інформацію щодо приблизної сумарної кількості зображень.

Згідно результатів, наведених в таблиці А.1, можна скласти висновок про нерівномірно кількість зображень на пристроях опитуваних. Середня загальна кількість зображень на пристрої дорівнює 296 559 штук, а медіана – 45 005 зображень. Якщо користувач має зображення на зовнішніх накопичувачах, то середня їх кількість дорівнює 710 362, а медіана – 29 176 зображень. Приклади відповідей наведено на рисунку 2.2.

Результати даного аналізу дозволять надалі обрати ефективний алгоритм пошуку у векторних базах даних, який спирається на кількість проіндексованих в системі зображень.

Кількість зображень на комп'ютері

Form builder | Responses +

Aa Questio...	Respon...	Submis...	Прізви...	# Рік нар...	Чи п...	# Всього ...	# Диск D	# Диск C	# Зовніш...	# Окрем...	# Окрем...
New submission	Anonymou	April 24, 2025 11:07 PM	Ляхова Дарина	1999	☐	70245				537	2092
New submission	Anonymou	April 24, 2025 10:19 PM			☑	40297	21080	15906	3311	38	35
New submission	Anonymou	April 24, 2025 10:14 PM	Ковтун Христина	1983	☑	16821	0	16821	0	0	111
New submission	Anonymou	April 24, 2025 9:33 PM			☐	146	12884	40721		9	20
New submission	Anonymou	April 24, 2025 7:41 PM	Лілія Шумиляк	1981	☑	119698	12985	60192	0	39	61
New submission	Anonymou	February 1, 2025 6:00 PM			☑	2739022	351239	46114	2332684	1647	59
New submission	Anonymou	January 29, 2025 2:41 PM	Vladyslav Kharchenko	2003	☐	119746		119746		0	0
New submission	Anonymou	January 27, 2025 5:26 PM	Походенко Аліна	2006	☑	1972	288	1684		120	136
New submission	Anonymou	January 27, 2025 1:53 PM		2005	☐	62306	0	62306	0	27	1594
New submission	Anonymou	January 24, 2025 1:26 PM	Дацок Женя	2005	☑	27981	0	27981	0	797	961
New submission	Anonymou	January 22, 2025 8:59 PM	Яковлева Олена	1974	☑	2007029	759860	80658	1166509	2731	1778

Рисунок 2.2 – Приклади відповідей на опитування

2.2 Формування набору даних для досліджень

Задача пошуку зображень вимагає ретельного аналізу наявних рішень та тестування їх якості, точності та повноти. Перед тим, як виконувати ці завдання, необхідно скласти певний чіткий набір даних – датасет, для обробки обраними моделями в рамках подальших експериментів.

Так як задача пошуку складається з двох компонент – введеного текстового запиту та очікуваних зображень у відповідь, тестувальний набір даних складатиметься саме з таких пар «текст-зображення». В рамках цієї роботи було створено датасет зі 100 пар даного виду, згрупованих в 10 класів, по 10 пар на кожний клас [38]. Обрані класи представляють як суміжні, так і віддалені тематичні категорії: «акваріуми», «коти», «Різдво», «одяг», «комп'ютери», «собаки», «квіти», «їжа», «хмарочоси», «заходи сонця» (рис. 2.3). Суміжною є пара класів «хмарочоси» – «заходи сонця» (рис. 2.4).

Включення суміжних класів дозволяє оцінити здатність моделей розрізняти тонкі семантичні відмінності. Віддалені ж за змістом класи допомагають перевірити загальну класифікаційну потужність моделей.

Фотографії було обрано як з особистого архіву автора роботи, так і з безкоштовних онлайн-стоків, що не потребують вказання авторства та дозволяють використання фотографій у власних некомерційних цілях [39]. Формат фотографій – JPEG, роздільна здатність – різна, аби бути наближеним до реальних умов користування пошуковою системою.

Рисунок 2.3 – Приклад зображень з тестувального набору даних

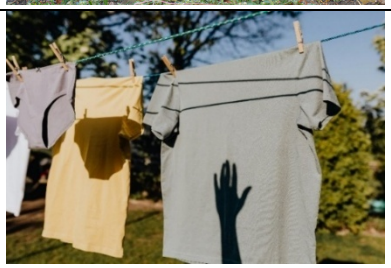
Рисунок 2.4 – Приклад зображень із суміжних класів «хмарочоси» та «заходи сонця»

Через стрімкий розвиток потужностей та якості сучасних нейромереж та омнімоделей ШІ, для автоматизації генерації описів до набору даних було обрано використовувати можливості даних сервісів. На момент створення датасету, однією з найпопулярніших доступних мультимодальних моделей ШІ була GPT-4o від OpenAI. Через легкість під’єднання до середовища виконання коду та наявного налаштованого проєкту, в рамках задачі створення текстових описів було обрано саме цю модель.

Основа роботи з великими мовними моделями – це передача на вхід інформації у вигляді промпту. Для поточної задачі було складено промпт-прохання англійською мовою, адже вихідний опис планується отримати також англійською – «Imagine that you are a user of an application for searching images by short text queries. Write this short query for given image. It should be JSON format with one field 'query'.». Модель GPT-4o має режим відповіді у форматі JSON, що задається в параметрах перед запитом [40]. До промпту було додано зображення у форматі Base64. На виході застосунок, що автоматизує створення датасету, отримує валідний JSON об’єкт, який далі зберігається в директорію поруч із зображенням з аналогічною назвою до розширення. Наприклад, від кореневої директорії «datasets/» зображення має відносний шлях «./class_name/1.jpg», а її опис – «./class_name/1.json». Дана ієрархія дозволяє в подальшому гнучко аналізувати як пари «зображення-текст», так і окремі компоненти цих пар.

В таблиці 2.1 надано приклади пар з набору даних.

Таблиця 2.1 – Приклади пар «зображення-текст» зі створеного навчального набору даних

Зображення	Текстовий опис англійською мовою
	gray and white kitten sitting on grass
	clothesline with t-shirts and shadow of hand
	office building behind trees at sunset
	landscape sunset over hills with grassy foreground

2.3 Аналіз сімейства моделей CLIP для опису зображень і текстових запитів

2.3.1 Аналіз оригінальної моделі CLIP від OpenAI

Сімейство моделей з контрастивним мультимодальним навчанням розпочало своє існування з початку 2021 року, коли ініціативна група OpenAI

оприлюднила власне дослідження [30]. В ньому організація описала архітектуру нейромережі, метод навчання та показники роботи після тестування. Використовуючи дуже великий набір пар «зображення-опис» (400 мільйонів проти приблизно 1,28 мільйонів у ImageNet), зібраних з інтернету, науковці контрастивно навчили мережу, яка зближує ембедінги правильних пар, та віддаляє неправильні. Цей підхід, за словами групи дослідників, дозволив досягнути на моделі без донавчання на ImageNet показників точності під час zero-shot класифікації, тобто визначенні класу без попередніх інструкцій, які б допомагали моделі, розміром в 76,2 %. Схожий результат мала модель ResNet-50, яку спеціально навчали на наборі даних ImageNet, тобто, на відміну від CLIP, вона мала розуміння про отримані дані.

Через головну особливість даної мережі, а саме спільний простір векторів зображень та текстів, які на етапі навчання були відкориговані на розділення правильних пар, вона є чудовим вибором для задачі пошуку зображень за текстовим описом. Архітектура CLIP та її підхід роботи складається з елементів, вказаних на рисунку 2.5.

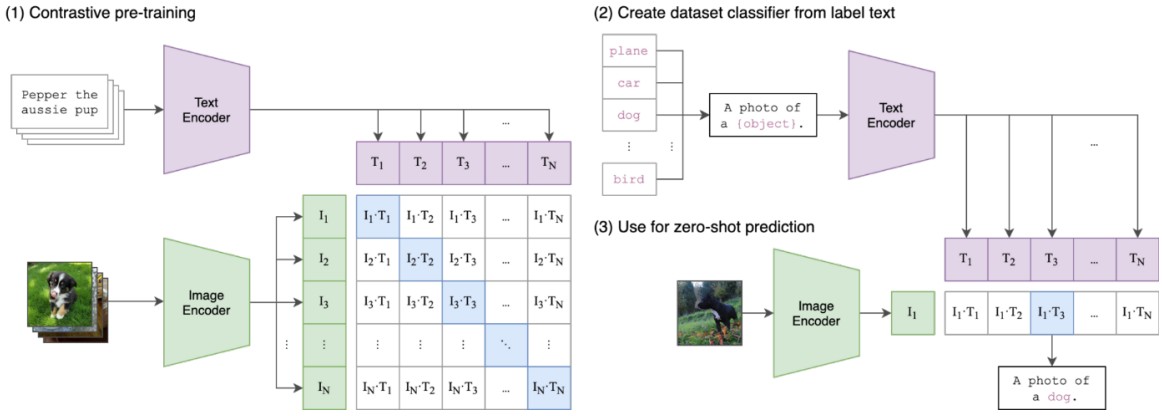


Рисунок 2.5 – Архітектурний підхід в моделі CLIP [30]

Модель складається із двох енкодерів: для зображень та тексту. В якості текстового енкодеру використовуються трансформери (GPT, BERT тощо), а для енкодерів зображень використовуються моделі Visual Transformer або ResNet з різними параметрами (роздільна здатність, кількість параметрів, розмір вихідних векторів, розмір ядра тощо). Від обраних комбінацій залежить

якість отриманих ембедингів та швидкодія, при чому ця залежність – обернено пропорційна, чим вища якість – тим повільніша модель.

На етапі навчання, коли обидва енкодери створили власні ембединги, CLIP їх нормалізує та мінімізує втрати за допомогою математичних операцій. Після навчання було отримано модель, що для текстового або візуального входу повертає вектор із 512 значень, який максимально близько характеризує цю пару.

2.3.2 Аналіз відкритої імплементації моделі OpenCLIP

Через той факт, що оригінальна модель від OpenAI має закриту імплементацію, це унеможливлювало виконання дій над нею, наприклад, донавчання чи перенавчання на іншому датасеті, експерименти з архітектурою тощо. Цю проблему було вирішено завдяки організаціям MLFoundations [41] та LAION [42], які на базі оригінальної статті від OpenAI створили модель OpenCLIP. Її головна особливість – повністю відкритий код, що дозволило проводити різноманітні експерименти. Архітектура цієї мережі повністю повторює оригінальний CLIP, а саме два енкодери для тексту та зображення.

Організація LAION має власні набори даних [43], що складаються із 400 мільйонів та 2 мільярдів пар зображень з їх текстовими описами. Після навчання моделей з різними візуальними енкодерами, але однаковою базою на цих датасетах, було отримано результати, що точність zero-shot передбачення класів була однакова, а то і краща за оригінальний CLIP [44]. Це показало, що збільшений набір навчальних даних допоміг моделі більш коректно розуміти контекст.

У таблиці 2.2 вказано порівняння варіацій моделі OpenCLIP, модифікованих за візуальним енкодером або за набором навчальних даних. В таблиці можна побачити зміну точності zero-shot класифікації зображень з датасету ImageNet в порівнянні з аналогічними варіаціями моделі CLIP.

Таблиця 2.2 – Порівняльна таблиця варіацій моделей OpenCLIP [45]

Варіація	**Навчальний набір даних**	**Кількість пар**	**Середня точність zero-shot на 38 наборах даних**	**Аналогічна модель від OpenAI**	**Середня точність zero-shot на 38 наборах даних з моделлю від OpenAI**	**Δ**
ViT-B/32 224x224	LAION-400M	0,4 млрд	50,74%	ViT-B/32 224x224	52,45%	-1,71%
ViT-B/16 224x224	LAION-400M	0,4 млрд	56,21%	ViT-B/16 224x224	56,26%	-0,05%
ViT-L/14 224x224	LAION-400M	0,4 млрд	59,71%	ViT-L/14 224x224	61,73%	-2,02%
ViT-B/32 224x224	LAION-2B	2 млрд	56,94%	ViT-B/32 224x224	52,45%	**+4,49%**
ViT-B/16 224x224	LAION-2B	2 млрд	58,66%	ViT-B/16 224x224	56,26%	**+2,4%**
ViT-L/14 224x224	LAION-2B	2 млрд	62,05%	ViT-L/14 224x224	61,73%	**+0,32%**
ViT-H/14 224x224	LAION-2B	2 млрд	64,19%	Відсутня, максимальна модель – ViT-L/14	61,73% для моделі ViT-L/14	**+2,46%**
ViT-g/14	LAION-2B	2 млрд	64,27%	Відсутня, максимальна модель – ViT-L/14	61,73% для моделі ViT-L/14	**+2,54%**
ViT-bigG/14	LAION-2B	2 млрд	66,67%	Відсутня, максимальна модель – ViT-L/14	61,73% для моделі ViT-L/14	**+4,94%**

2.3.3 Аналіз моделі MobileCLIP

Хоча сімейство моделей і показує гарні результати в різних бенчмарках, але кількість параметрів для старших варіацій сягає великих значень, від того і швидкодія стає меншою. Це є проблемою для задачі локального запуску інференсу моделі на середньостатистичних домашніх комп’ютерах або навіть смартфонах. Для вирішення цієї проблеми відділ Machine Learning Research (досліджень машинного навчання) компанії Apple у квітні 2024 року оприлюднив свою роботу, що стосується оптимізації архітектури моделі CLIP

та способу її навчання для полегшення розміру моделі, при цьому не втрачаючи показників точності на різних бенчмарках [46].

В рамках цієї роботи науковці зробили наступні покращення:

– поліпшили навчальний набір даних, попередньо згенерувавши більш якісні підписи до зображень за допомогою моделі CoCa;

– використали новий метод мультимодального підсиленого навчання (multi-modal reinforced training);

– створили нові оптимізовані архітектури енкодерів зображень та тексту. Для енкодера зображень створено гібридний візуальний трансформер MCi, що базується на архітектурі FastViT [46]. Для енкодера тексту ж створено трансформер Text-RepMixer, який, за словами дослідників, є меншим, швидшим та однаковим за продуктивністю в порівнянні з текстовими енкодерами оригінальних моделей CLIP (рис. 2.6).

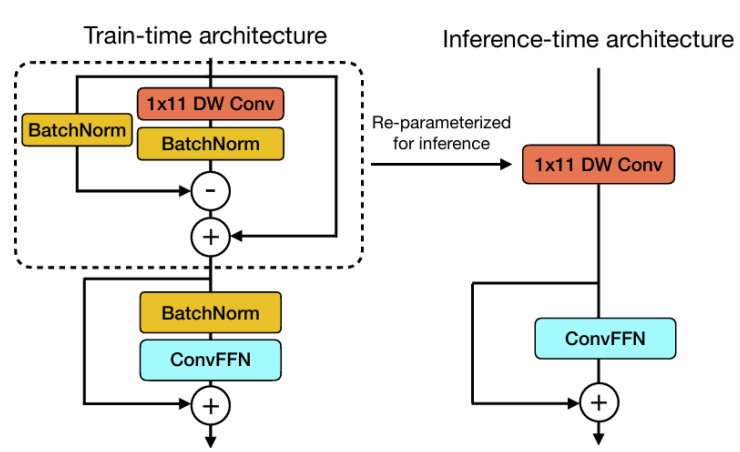


Рисунок 2.6 – Архітектура текстового енкодеру моделі MobileCLIP [46]

Таким чином, була створена оптимізована версія моделі CLIP – MobileCLIP. Згідно дослідженням, найменша її версія MobileCLIP-S0 за точністю zero-shot класифікації є на рівні з оригінальною моделлю OpenAI CLIP ViT-B/16, при цьому в 4,8 разів швидше та 2,8 разів менша. Найбільша ж версія MobileCLIP-B (LT) має точність 77,2%, що є порівняним із OpenAI CLIP ViT-L/14@336. Більш детальне порівняння даних щодо моделей відображено в таблиці 2.3.

Таблиця 2.3 – Порівняльна таблиця моделей OpenCLIP (на наборі LAION-2B) та MobileCLIP [45, 46]

Варіація моделі MobileCLIP	**Кількість параметрів моделі MobileCLIP (млн)**	**Середня точність zero-shot на 38 наборах даних**	**Схожа за точністю модель OpenCLIP**	**Кількість параметрів моделі OpenCLIP (млн)**	**Середня точність zero-shot на 38 наборах даних з моделлю OpenCLIP**	**Δ точність**	**% параметрів від OpenCLIP**
S0	53,8	58,1%	ViT-B/16	149,62	58,66%	-0,56%	35,96%
S1	84,9	61,3%	ViT-L/14	427,62	62,05%	-0,75%	19,85%
S2	99,1	63,7%	ViT-g/14 s12b_b45k	1366,68	62,99%	**+0,71%**	7,25%
B	149,7	65,2%	ViT-g/14 s34b_b88k	1366,68	64,27%	**+0,93%**	10,95%
B (LT)	149,7	65,8%	ViT-g/14 s34b_b88k	1366,68	64,27%	**+1,53%**	10,95%

Отже, згідно з порівнянням кількості параметрів та середньої точності, модель MobileCLIP дійсно показує результати на рівні з більшими та важчими моделями OpenCLIP. Цей факт є вирішальним для задачі вибору моделі, що використовуватиметься в локальних середовищах на не надто потужних пристроях користувачів застосунку з пошуку зображень за текстовим описом.

Для виконання подальших експериментів було вирішено обрати три моделі з різних підсімейств зі схожими характеристиками. За порівняльну характеристику було обрано середню точність zero-shot класифікації. В ході аналізу було обрано такі моделі:

– OpenAI CLIP ViT-L/14@224 – середня точність дорівнює 61,73% (при кількості параметрів моделі в розмірі 427,62 млн) [45];

– OpenCLIP ViT-g/14 s34b_b88k – середня точність дорівнює 64,27% (при кількості параметрів моделі в розмірі 1366,68 млн);

– MobileCLIP B (LT) – середня точність дорівнює 65,8% (при кількості параметрів моделі в розмірі 149,7 млн).

2.4 Методи зменшення розмірності простору ознак для візуалізації

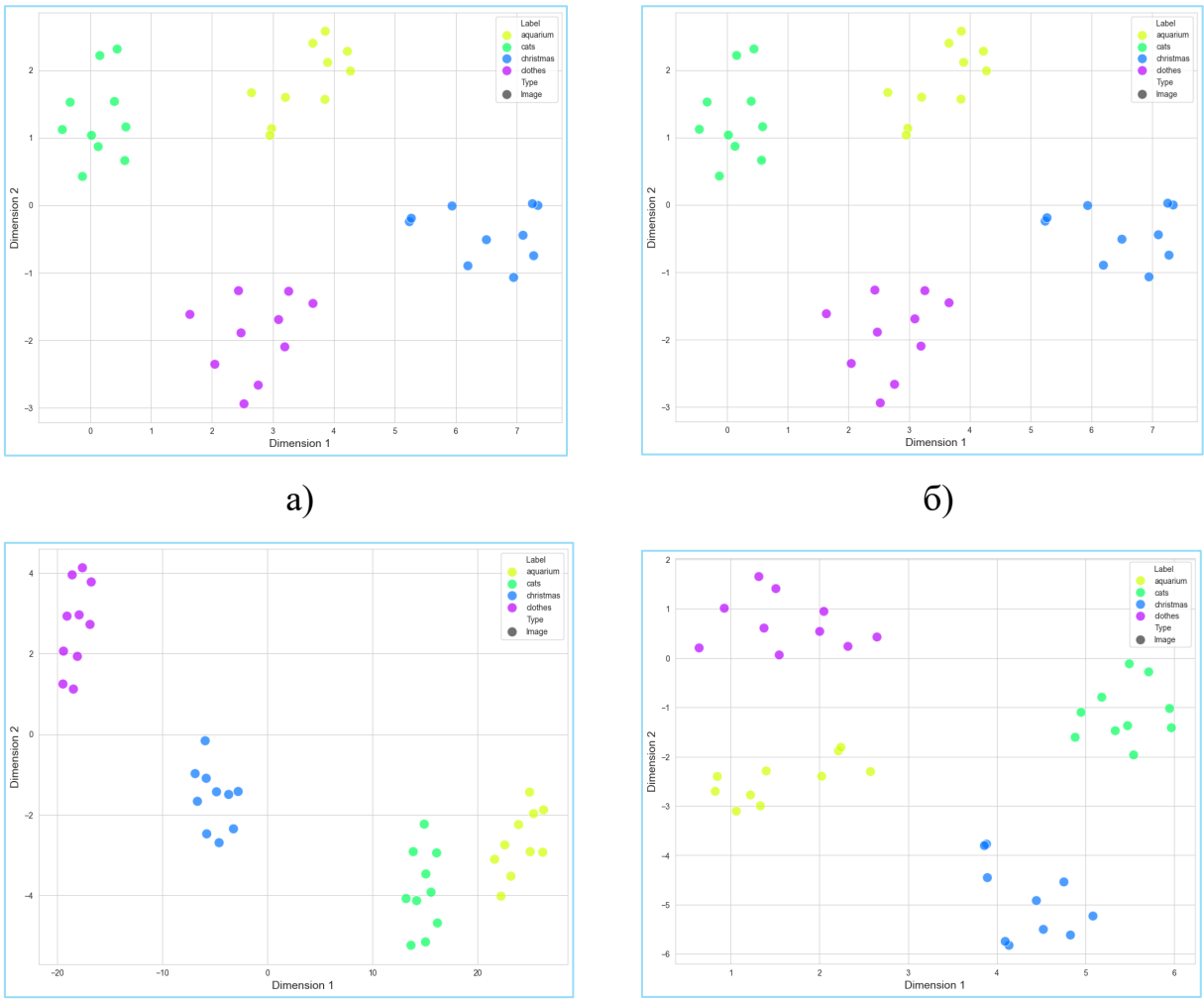
Так як вектори ознак сімейства моделей CLIP є багаторозмірними, а саме вони мають 512 компонент, то для їхньої візуалізації в подальшій дослідницькій роботі необхідно приводити їх до векторів дво- або тривимірного простору, щоб надалі відобразити на площині. Для цієї задачі існують алгоритми зниження розмірності. Найпопулярнішими на момент написання даної роботи є PCA, *t*-SNE та UMAP.

Метод PCA (Principal Component Analysis, англ. Метод головних компонент) є лінійним методом, який шукає компоненти вектора з найбільшою дисперсією серед усіх даних та проєктує цілі ембедінги на вказану кількість цих головних компонент.

Метод *t*-SNE (*t*-Distributed Stochastic Neighbor Embedding, англ. Стохастичне вкладення сусідів із *t*-розподілом) теж є нелінійним методом. Він зберігає локальні відстані, тобто близькі об’єкти в проєктованому просторі є такими ж близькими, як і у багатовимірному просторі.

UMAP (Uniform Manifold Approximation and Projection, англ. Уніформне апроксимування та проєктування многовиду) – також нелінійний. Зберігає як локальні, так і частково глобальні структури. Є швидшим та краще масштабованим за *t*-SNE.

На рисунку 2.7 відображено візуалізацію проєкцій ембедингів зображень з чотирьох класів, зроблених кожним із вищевказаних методів, а також комбінацією методів PCA (зниження розмірності до 40 компонент) та *t*-SNE (подальше зниження до двох компонент). Останній спосіб є більш оптимальним з точки зору витрат комп’ютерних потужностей, адже алгоритму *t*-SNE легше обробляти 40 компонент замість 512 з оригінального ембедингу.



а) б)

в) г)

Рисунок 2.7 – Візуалізація проєкцій ембедингів зображень
з використанням методів зниження розмірності:
а) PCA; б) *t*-SNE; в) UMAP; г) комбінований (PCA та *t*-SNE)

2.5 Міра подібності ембедингів сімейства моделей CLIP

У процесі навчання модель приводить свої вектори ознак до одиничної довжини за допомогою l^2 нормалізації. Це призводить до того, що довжина вектора завжди дорівнює одиниці, і відрізнятися ембединги починають лише напрямком. Також в ході тренування моделі CLIP мінімізують контрастивний лосс (втрати), в якому позитивні пари мають максимально можливий скалярний добуток своїх векторів, які є одиничними. І через те, що скалярний

добуток одиничних векторів – це косинус кута між ними, то в ході використання моделі використовується косинусна міра схожості.

Якщо порівнювати із іншими варіантами мір подібності, то, наприклад, скалярний добуток є чутливим до довжини векторів, та не враховує семантику цих ембедингів. А евклідова відстань між двома одиничними векторами є тісно пов'язаною з косинусом, але згідно формули необхідно підраховувати квадрат відстані, що є більш важкою операцією для пристроїв.

Формула косинусної схожості виглядає наступним чином:

$$sim_{cos}(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|_2 \, \|\vec{v}\|_2}, \tag{2.1}$$

де $\vec{u} \cdot \vec{v}$ – скалярний добуток двох векторів;

$\|\vec{u}\|_2$ – l^2-норма вектору $\vec{u}$;

$\|\vec{v}\|_2$ – l^2-норма вектору $\vec{v}$.

Значення косинусної схожості коливаються від -1,0 до +1,0, де -1,0 – це повністю протилежні вектори, 0 – це перпендикулярні вектори, а +1,0 – це абсолютно однакові за напрямком вектори.

У таблиці 2.4 наведено приклади значень мір косинусних схожостей для різних пар «зображення-текст» зі створеного тестувального набору даних.

Таблиця 2.4 – Приклади підрахованих мір косинусної схожості для різних пар «зображення-текст»

Зображення	**Текст**	**Косинусна схожість**
1	**2**	**3**
	aquarium with plants and fish	+0,25

Продовження таблиці 2.4

1	2	3
	black cat on colorful rug indoors	+0,27
	calico cat sitting on concrete path in garden	-0,11
	cute puppies playing outside	-0,07
	pink coneflowers in garden	-0,13

Перші дві пари – відповідні одна одній, тому косинусна схожість там коливається між значеннями 0,25–0,27. Інші ж пари – невідповідні, отже і косинусна схожість для них показує відсутність семантичного співпадіння.

2.6 Візуалізація ембедингів сімейства моделей CLIP

Для подальшого коректного дослідження необхідно обрати спосіб зменшення розмірності простору векторів ознак серед описаних в підрозділі 2.4. Для виконання цієї задачі було обрано метод візуального порівняння проєкцій чотирьох ембедингів на просторову близькість.

Було обрано певне зображення із класу «хмарочоси». Цей клас було

обрано не випадково, адже він є суміжним за складеними описами для класу «заходи сонця». Для обраного зображення підібрано три описи:

– опис, що складає пару;
– опис зображення із суміжного класу «заходи сонця»;
– опис зображення із зовсім іншого за семантикою класу «собаки».

Очікувано, що для першого опису візуальна близькість повинна бути мінімальною, для другого – більш далекою, а для третього – ще дальшою. На рисунку 2.8 наведено візуалізації чотирьох методів, де круг – це ембединг зображення, а хрест – ембединг тексту. Дані позначки для цього виду діаграм і надалі використовуватимуться впродовж всієї роботи.

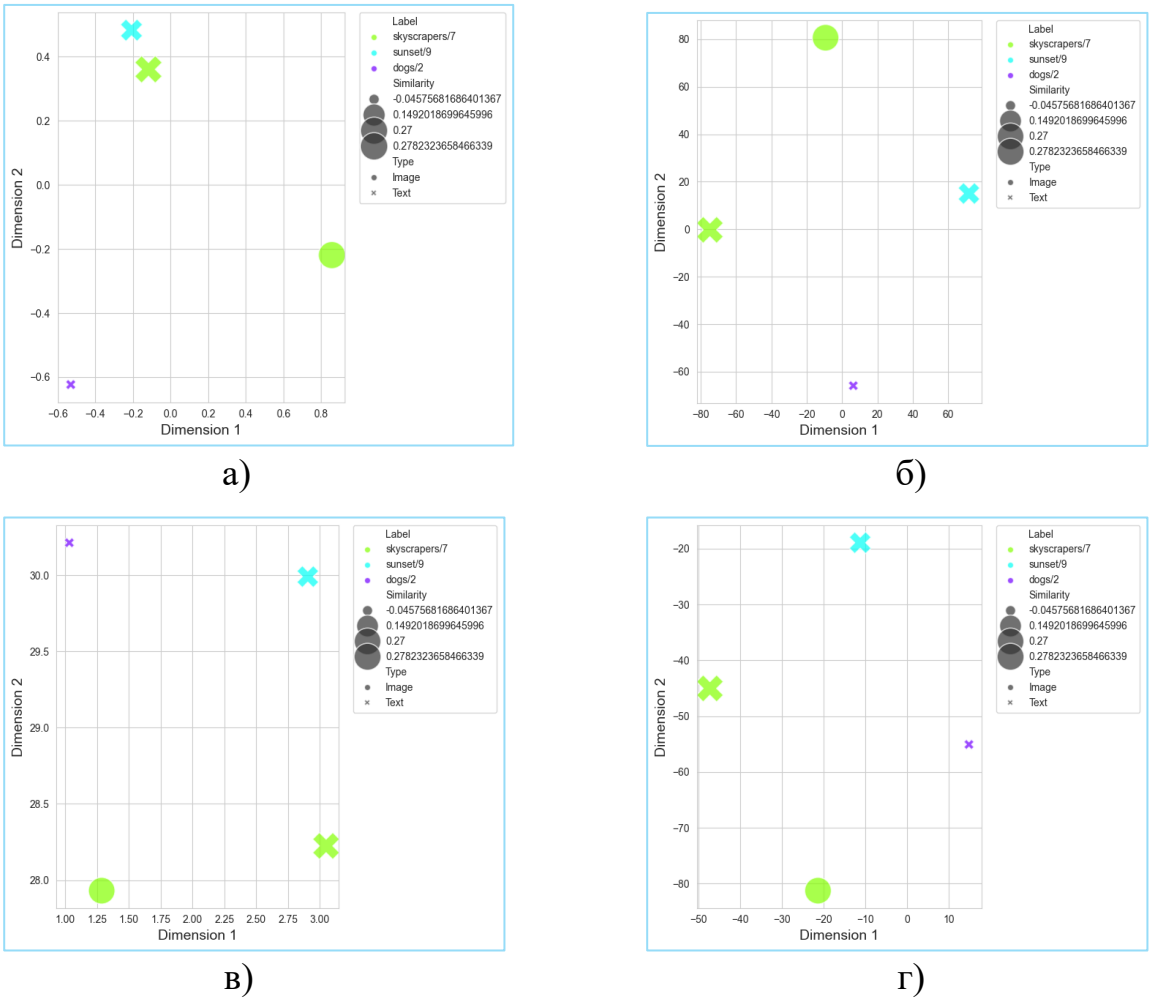


Рисунок 2.8 – Візуалізація схожості ембедингу зображення та трьох описів з використанням різних методів зменшення розмірності:
а) PCA; б) *t*-SNE; в) UMAP; г) комбінований (PCA та *t*-SNE)

В ході аналізу експерименту було виявлено, що для методу PCA та UMAP візуально відстані між ембедингами є в очікуваному порядку. Метод *t*-SNE обрав перші два тексти як приблизно однакові за відстанню, а комбінований метод врахував останній опис із невідповідного класу як найбільш близький. Якщо порівнювати методи PCA та UMAP, слід залишити останній, адже він є продуктивним, працює краще на великих даних, а PCA урізає деякі несуттєві дані, які можуть вплинути на візуалізацію більших обсягів, як наприклад 100 ембедингів тестувального набору даних.

2.7 Аналіз якості кластеризації для сімейства моделей CLIP

2.7.1 Візуалізація класів на основі ембедингів варіацій моделей CLIP для зображень та їх текстових описів

Одним із критеріїв відбору моделі, хоча і не таким точним, як числові порівняння, є аналіз візуалізації розсіювання отриманих векторів ознак. Для виконання даного експерименту ембединги зображень і тексту, згенеровані кожною моделлю, було нормалізовано до одиничної довжини (l^2-нормалізація), об’єднано в єдиний масив, зменшено розмірність до двох компонент за допомогою обраного методу UMAP та візуалізовано на двовимірній площині. Результати візуалізації показано в таблиці 2.5.

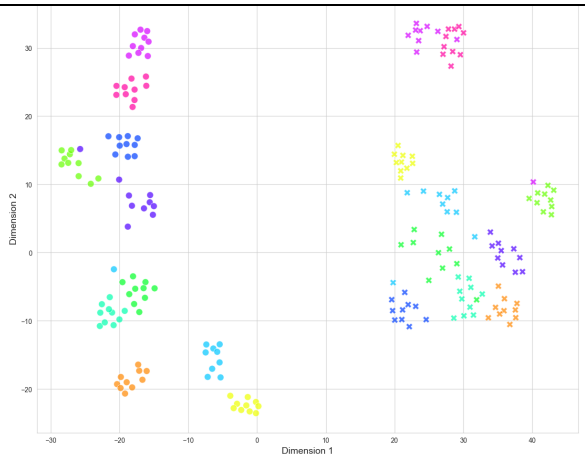
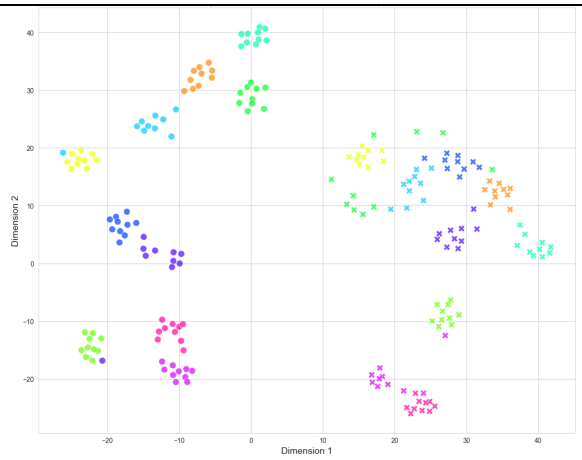
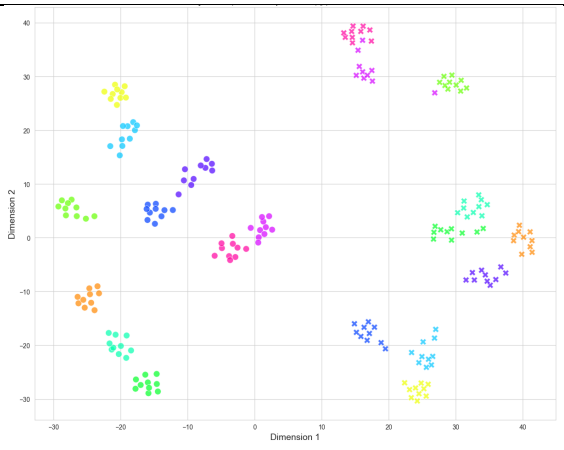
Найкращий результат, де кожний клас конкретно згрупувався в свою хмару, є у моделі MobileCLIP B (LT). Ембединги інших моделей є більш розпливчатими в рамках класу, і для деяких елементів без явної візуальної допомоги у вигляді різних кольорів неможливо визначити належність до окремого кластеру.

Також можна помітити, що ембединги зображень та тексту не згрупувались у більш об’ємну семантичну групу, де всі об’єкти одного класу (як зображення, так і описи) належать одному кластеру. Це можна пояснити високою розмірністю компонент векторів ознак, та більш абстрактною

структурою самих ембедингів, адже моделі навчалися на дуже великому обсягу даних, який безпосередньо впливає на структуру вихідного вектору.

Цікаве спостереження, що кластери суміжних класів «хмарочоси» (пурпурний колір) та «заходи сонця» (рожевий) є не такими роздільними, як інші класи, що є природнім логічним висновком.

Таблиця 2.5 – Діаграма розсіювання класів для ембедингів різних моделей

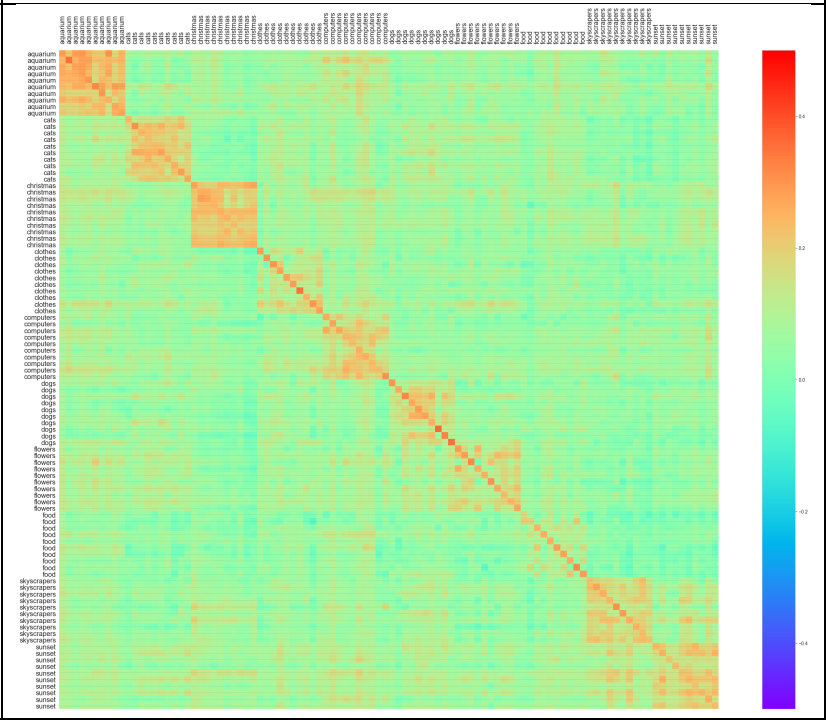
Модель	**Візуалізація**
OpenAI CLIP ViT-L/14@224	
OpenCLIP ViT-g/14 s34b_b88k	
MobileCLIP B (LT)	

2.7.2 Підрахунок схожості між ембедингами зображень та текстових описів

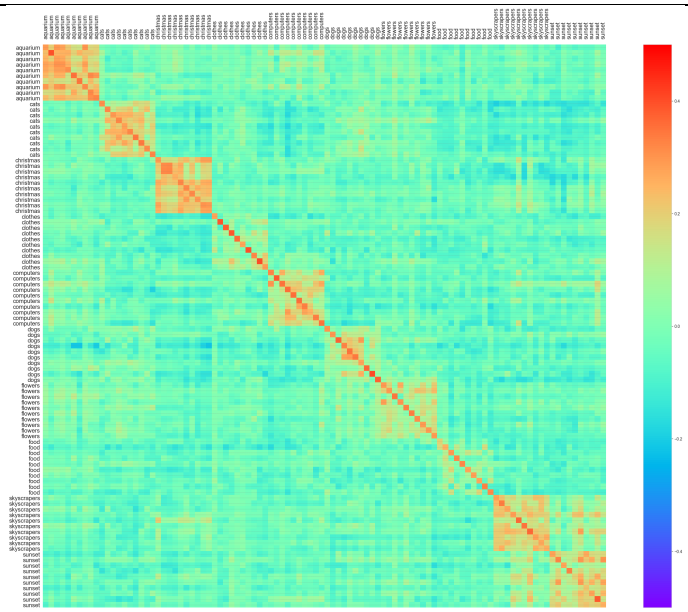
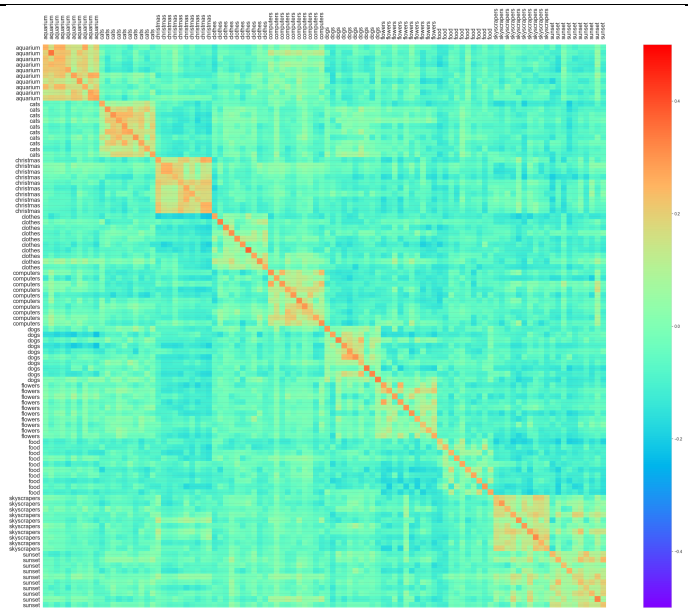
Більш якісним аналізом моделей є представлення статистики через цифри. Отже, за міру якості кластеризації було обрано середні значення мір косинусних подібностей для ембедингів у рамках класів, визначених датасетом.

Попередньо, було візуалізовано ці міри тепловою картою від -0,5 до 0,5 градацією кольорами райдуги. Ця карта показує розбіжність в мірах подібності у рамках класу та поза його межами. Результати візуалізації теплових карт для ембедингів кожної моделі наведено в таблиці 2.6.

Таблиця 2.6 – Теплові карти значень косинусних подібностей для кожної з пар ембедингів, створених різними моделями

Модель	**Візуалізація**
1	**2**
OpenAI CLIP ViT-L/14@224	

Продовження таблиці 2.6

1	2
OpenCLIP ViT-g/14 s34b_b88k	
MobileCLIP B (LT)	

Також для кожної моделі було підраховано середні міри косинусної подібності для всього класу, для правильних пар кожного класу (діагональ кожного з квадратів на візуалізації) та середню міру серед правильних пар для всього набору даних (діагональ всієї матриці на візуалізації). Результати для кожної моделі наведено в таблиці 2.7.

Такі класи, як «одяг», «їжа», «квіти» мають мінімальні значення в рамках класів, адже це пов’язано із різноманітним наповненням датасету. Для більшого набору даних ці значення б вирівнялись. Дана незбалансованість в

даних пояснюється бажанням охопити різні варіанти пар (близькі та не дуже) для подальшого визначення оптимального порогу косинусної подібності для функціонування застосунку.

Таблиця 2.7 – Середні значення мір косинусної схожості для кожної моделі в рамках класів

Модель	**Клас**	**Середнє значення для всього класу (всі пари)**	**Середнє значення для відповідних пар класу**
OpenAI CLIP ViT-L/14@224	Акваріум	0,23	0,29
	Коти	0,20	0,26
	Різдво	0,22	0,25
	Одяг	0,14	0,29
	Комп’ютери	0,18	0,26
	Собаки	0,16	0,30
	Квіти	0,15	0,27
	Їжа	0,11	0,26
	Хмарочоси	0,18	0,26
	Заходи сонця	0,18	0,24
	За всіма класами	**0,18**	**0,27**
OpenCLIP ViT-g/14 s34b_b88k	Акваріум	0,23	0,32
	Коти	0,18	0,32
	Різдво	0,22	0,30
	Одяг	0,08	0,35
	Комп’ютери	0,17	0,34
	Собаки	0,12	0,34
	Квіти	0,13	0,31
	Їжа	0,05	0,29
	Хмарочоси	0,19	0,31
	Заходи сонця	0,17	0,28
	За всіма класами	**0,15**	**0,32**
MobileCLIP B (LT)	Акваріум	0,19	0,27
	Коти	0,16	0,28
	Різдво	0,17	0,25
	Одяг	0,07	0,30
	Комп’ютери	0,14	0,28
	Собаки	0,09	0,30
	Квіти	0,08	0,27
	Їжа	0,03	0,25
	Хмарочоси	0,15	0,27
	Заходи сонця	0,12	0,23
	За всіма класами	**0,12**	**0,27**

Середні значення косинусних схожостей для тих пар, що не знаходяться в рамках одного класу, тобто не є коректними, складає (чим менше, тим краще):

– 0,07 для моделі OpenAI CLIP ViT-L/14@224;
– -0,03 для моделі OpenCLIP ViT-g/14 s34b_b88k;
– -0,06 для моделі MobileCLIP B (LT).

Отже, модель MobileCLIP B (LT) найкраще визначає неналежність елементу до класу, що є критичним для задачі відсіювання некоректних зображень для заданого текстового опису.

2.7.3 Вибір моделі отримання ембедингів для подальшої роботи

В таблиці Б.1 наведено мінімальні та максимальні значення міри косинусної подібності для кожної моделі. Також було проаналізовано схожість ембедингів зображень в рамках класу та поза ними за допомогою методу силуетів, результати наведено на рисунках 2.9 – 2.11.

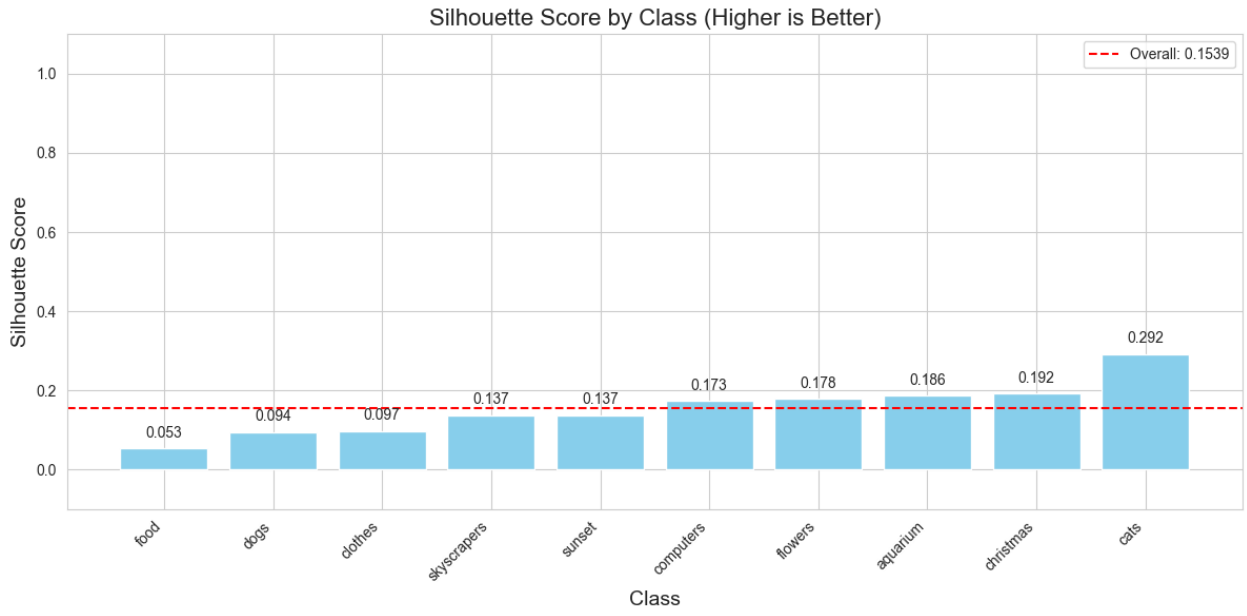


Рисунок 2.9 – Метод силуетів для ембедингів зображень моделі OpenAI

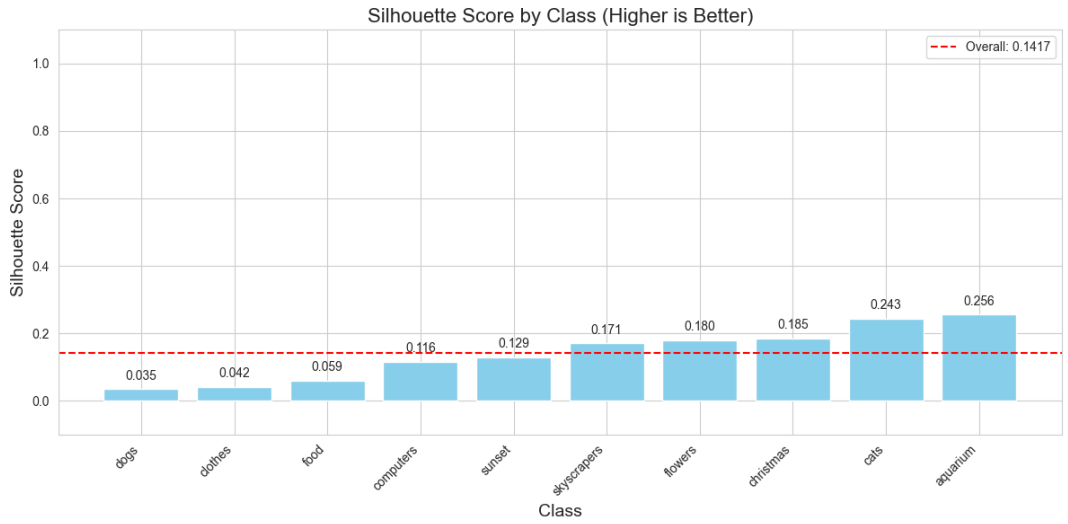


Рисунок 2.10 – Метод силуетів для ембедингів зображень моделі OpenCLIP

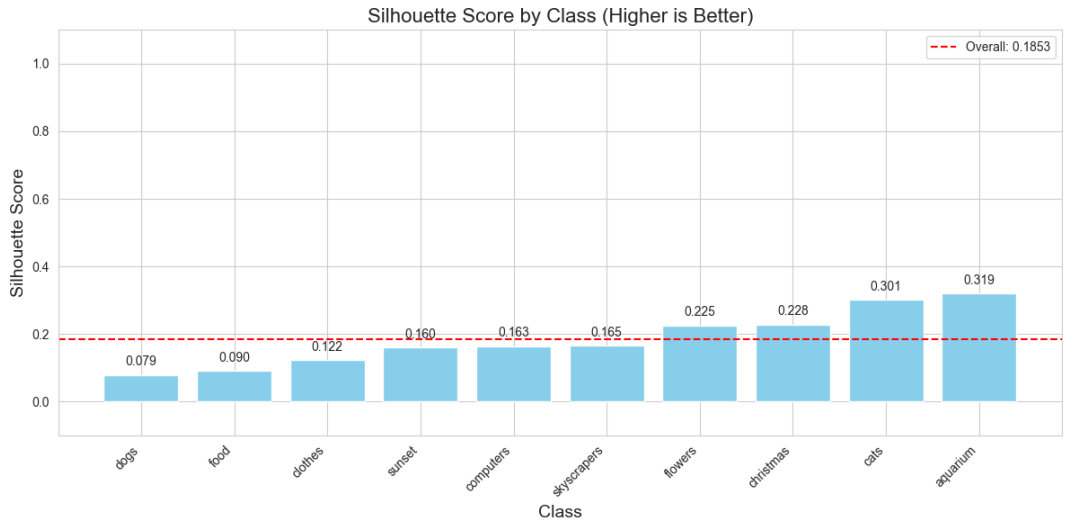


Рисунок 2.11 – Метод силуетів для ембедингів зображень моделі MobileCLIP

За методом силуетів модель MobileCLIP показала невелику перевагу над іншими моделями в якості кластеризації, але цієї переваги недостатньо для точного визначення, чи дійсно ця модель краща. Згідно візуального методу аналізу, а саме за діаграмою розсіювання ембедингів, найкраще себе показала також модель від Apple, адже там кластери є найбільш явно згрупованими.

Щодо числових даних, оригінальна модель від OpenAI у зв'язці її великого розміру, від того повільної роботи, та отриманих результатів середніх значень мір схожості, не є оптимальною, адже гірше всіх визначає належність до класу та навпаки, неналежність до непотрібних класів.

Інші дві моделі є порівняно схожими за якістю кластеризації, тому перевагу слід віддати тій, яка більш оптимізована та створює менше навантаження на систему під час роботи. Через велику різницю в кількості параметрів між моделями OpenCLIP та Apple MobileCLIP, а саме 1366,68 млн проти 149,7 млн відповідно, теоретично можна очікувати набагато вищу швидкодію моделі від Apple. Саме її було обрано в якості подальшого аналізу, проєктування застосунку та перевірки його роботи.

2.8 Аналіз різних варіантів текстових описів для одного зображення та одного класу

Наступний експеримент (рис. 2.12) полягає в тому, що для п’яти зображень, з яких три – з різних класів, а два – з одного класу, було створено по п’ять описів (чотири англійською, п’ятий – українською). Описи згенеровані моделлю GPT-4o на різних налаштуваннях температури відповіді.

Через те, що модель MobileCLIP вміє правильно декодувати лише англійську мову, для українського тексту було використано систему Google перекладач із автоматичним визначенням мови запиту [47]. Таким чином буде оцінено якість перекладу запитів, та той факт, наскільки перекладений іншомовний текст буде впливати на схожість. Цей дослід показує, наскільки різні описи мають вплив на схожість в рамках одного зображення.

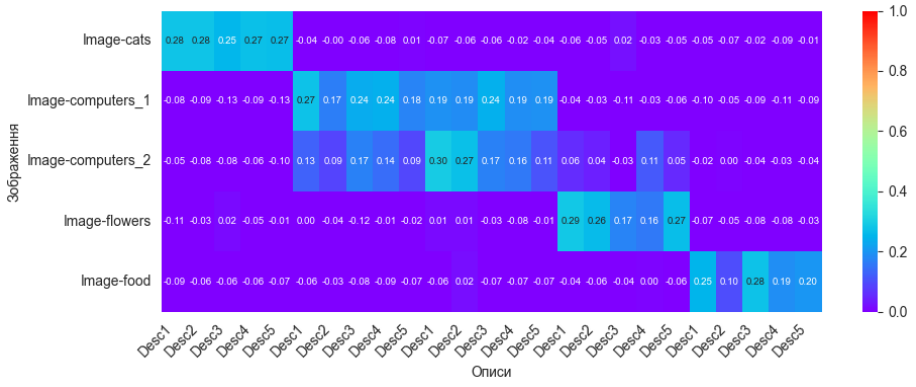
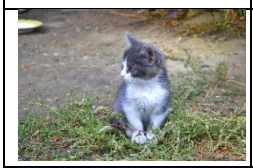
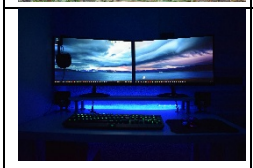
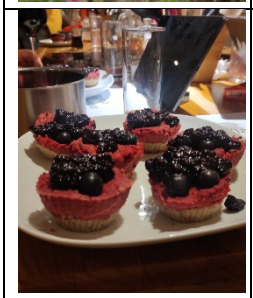


Рисунок 2.12 – Теплова карта мір подібностей для зображень та описів

У таблиці 2.8 вказані отримані косинусні подібності разом із середніми значеннями як для зображення, так і для описів.

Таблиця 2.8 – Подібність різних описів одного зображення згідно косинусної міри

Зображення	**Опис 1 (температура 0,0)**	**Опис 2 (температура 0,2)**	**Опис 3 (температура 0,6)**	**Опис 4 (температура 0,8)**	**Опис 5 (температура 1,0)**	**Середнє значення міри подібності**
	gray and white kitten 0,28	kitten sitting outside 0,28	small cat in garden 0,25	cute kitten on ground 0,27	кошеня на траві 0,27	**0,27**
	dual monitor setup 0,27	gaming keyboard and mouse 0,17	blue LED computer desk 0,24	dual screen workstation 0,24	ігрова комп'ютерна установка 0,18	**0,22**
	desktop computer with pink lighting 0,3	purple themed workspace 0,27	home office setup with monitor 0,17	keyboard and mouse in dark room 0,16	робочий стіл з підсвіткою 0,11	**0,2**
	pink coneflowers 0,29	echinacea blossoms 0,26	garden flowers in summer 0,17	vibrant pink petals 0,16	рожеві квіти ехінацеї 0,27	**0,23**
	berry cheesecake cupcakes 0,25	dessert table setting 0,1	mini berry tarts 0,28	fruit-topped cupcakes 0,19	десерт з ягодами 0,2	**0,2**
	0,28	**0,22**	**0,22**	**0,2**	**0,2**	**0,23**

Експеримент показав, що модель із температурою 0,0 видає найбільш подібні описи, отже в наступних дослідах буде використано саме такий параметр.

Опис українською, перекладений на англійську, показує подібну до англійського міру схожості.

Також було проведено експеримент для перевірки схожості різних «середніх» описів для кожного класу. Під «середнім» мається на увазі, що моделі GPT-4o було надано всіх представників кожного класу (по 10 зображень) із проханням створити п’ять описів (чотири описи англійською, один – українською), що загальними словами характеризують надану групу картинок. Для перевірки схожості в рамках класу був обрахований середній вектор ознак зображень, та наданий на порівняння із ембедингами кожного опису. Результати експерименту відображені на рисунку 2.13 і таблиці В.1.

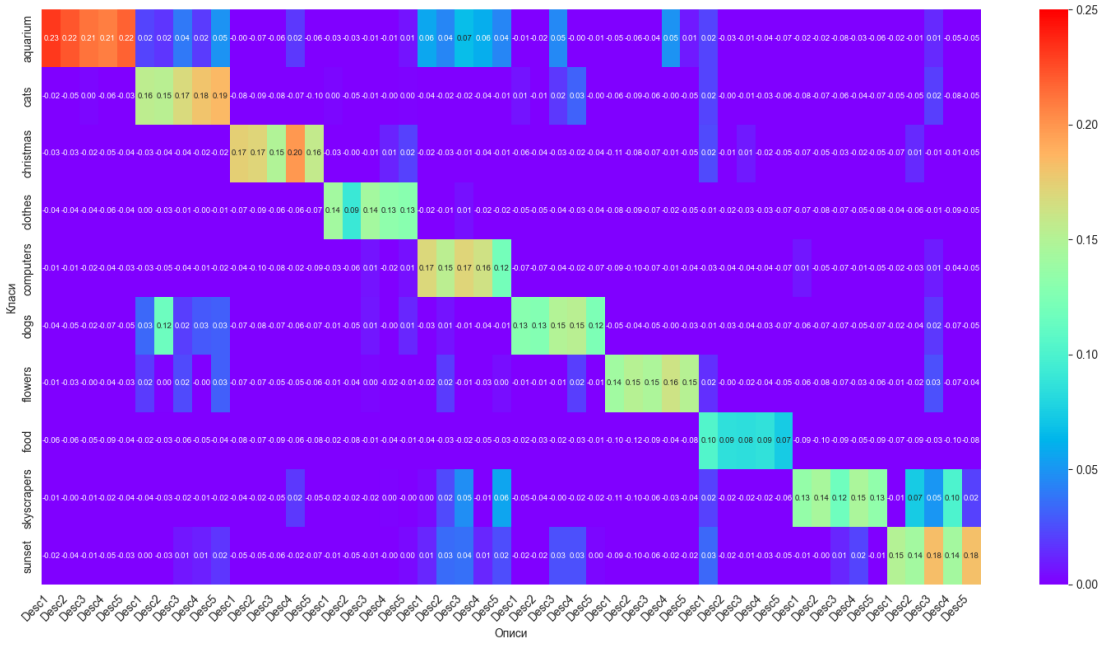


Рисунок 2.13 – Теплова карта мір подібностей для різних середніх описів в рамках кожного класу

Можна скласти висновок, що різні за структурою, але подібні за змістом та сенсом описи як одного зображення, так і взагалом класу, також розрізняються моделлю MobileCLIP як одна семантична структура. А описи, що не відповідають своїм парам, також розрізняються за мірою подібності як не схожі (візуально це підтверджується більш темними ділянками за межами прямокутників відповідних пар).

Також в ході експерименту продемонстровано, що описи українською, перекладені на англійську мову є не гіршими за подібністю в порівнянні з оригіналами описів англійською. Отже, під час розробки застосунку є можливим використання перекладача з автоматичним розпізнаванням мови для перекладу іноземних пошукових запитів. Це зробить застосунок багатомовним та автоматично зручнішим для користування людям, що не володіють англійською мовою.

2.9 Пошук універсального текстового опису для класів зображень

В ході попереднього дослідження було згенеровано п’ять описів для кожного з десяти класів набору даних, а також підраховано міри подібності до усередненого значення ембедингів зображень. Для подальшого тестування готового застосунку необхідно обрати описи, що найповніше характеризують кожний клас. В якості міри повноти було обрано косинусну подібність між середнім ембедингом зображень класу та цього опису. Найповнішим описом вважатиметься той, що має максимальний показник подібності серед своїх конкурентів. Результати вибору наведено в таблиці 2.9.

Таблиця 2.9 – Результати вибору загального опису кожного класу

Клас	**Опис**	**Косинусна подібність**
Акваріум	home aquariums with fish	**0,23**
Коти	домашні коти на вулиці та в приміщенні	**0,19**
Різдво	Christmas trees with lights and decorations	**0,17**
Одяг	fashion items for everyday wear	**0,14**
Комп’ютери	technology workspace	**0,17**
Собаки	collection of dog photos	**0,15**
Квіти	outdoor and indoor flowering plants	**0,16**
Їжа	variety of different restaurant and homemade dishes	**0,10**
Хмарочоси	contemporary business district	**0,15**
Заходи сонця	sunset and dusk scenes outdoors	**0,18**

2.10 Пошук на основі ембедингів моделі CLIP

2.10.1 Підхід щодо пошуку

Узагальнений підхід до пошуку зображень за текстовим запитом на основі сімейства моделей CLIP, у тому числі й MobileCLIP, можна представити у декількох етапах.

Перший етап – індексація зображень. Користувач надає список зображень або директорію, яку необхідно проіндексувати. Під індексацією мається на увазі, що кожне зображення кодується в ембединг та зберігається у локальній базі даних разом із метаданими. Серед метаданих може бути назва файлу, шлях у файловій системі, розмір, додаткові теги, мітки, дата створення, геолокація тощо.

Другий етап – побудова ембедингу із текстового запиту користувача. Після того, як користувач ввів запит, він кодується у вектор ознак тією ж моделлю, що використовувалась для кодування зображень.

Далі – порівняння ембедингів та фільтрація за метаданими. Ця процедура виконується базою даних, яка приймає на вхід вектор і додаткові дані для пошуку. Порівняння відбувається за допомогою вже визначеної раніше міри схожості – косинусної подібності між векторами. Вона дозволяє оцінити, наскільки вектори напрямлені в одну область, отже наскільки вони є семантично близькими одне до одного.

Для того, щоб уникнути нерелевантних результатів, наступним етапом є відсікання результатів за порогом міри подібності. Цей поріг необхідно обирати експериментально на тестовому наборі даних, адже якщо обрати його замалим – користувач отримає забагато зайвих зображень, а якщо завеликим – користувач може втратити деякі результати, що мають менше деталей.

2.10.2 Вибір бази даних для ефективного пошуку

Вибір бази даних, яка зберігатиме дані та робитиме пошук за запитом, є важливим етапом під час проєктування застосунку. Під час вибору слід опиратися на результати аналізу кількості зображень на пристроях потенційних користувачів у підрозділі 2.1. Серед популярних наразі рішень для операцій над векторними даними, як було сказано у підрозділі 1.3.2, є Meta AI Faiss, ElasticSearch, його відкритий аналог OpenSearch та Qdrant.

Faiss хоч і забезпечує найвищу швидкість пошуку і масштабованість до векторів, що не вміщаються в оперативну пам’ять, але він не має вбудованої підтримки фільтрації за метаданими і потребує додаткових рішень для зберігання в файловій системі. ElasticSearch, хоч і пропонує повноцінне зберігання в файлах та має можливості індексації метаданих, але відчуває суттєві затримки при пошуку в великій базі векторів – швидкість може сягати до 200 мілісекунд [48]. OpenSearch успадковує від Elastic схожі можливості, додаючи плагіни для *k*-NN пошуку за векторами, проте його індексація зазвичай повільніша за вузькоспеціалізовані векторні бази даних. Qdrant же спеціалізується на векторному пошуку з фільтрацією за метаданими, підтримує файлове зберігання колекцій та індексування символьних полів, що значно прискорює пошук із фільтрами навіть при десятках тисяч векторів.

Для оптимальної комбінації швидкодії та використання простору, із додатковими можливостями фільтрації за метаданими, в ході проєктування програмного продукту було вирішено зупинитись на спеціалізованому рішенні – векторній базі даних Qdrant, що також має можливість індексації за допомогою алгоритму ієрархічного навігабельного малосвітового графу (HNSW) для дуже великих обсягів даних, що перевищують медіанну кількість зображень з опитування потенційних користувачів.

2.11 Підбір порогу для визначення коректності пошуку

Після вибору бази даних та методики пошуку, необхідно визначитись із порогом міри косинусної схожості, що визначатиме – чи є знайдене зображення релевантним до пошукового запиту. Поріг буде вибрано в діапазоні від 0,0 до 0,15 із кроком 0,001. На кожній ітерації буде проведено калькуляцію таких метрик для кожного класу: Precision, Recall та F1 за цими двома метриками.

Precision – це метрика, що відображає точність пошуку 10 зображень даного класу для поточного порогу. Тобто ця метрика показує відношення релевантних результатів до всіх знайдених. Якщо всього було знайдено 100 зображень, із яких лише 10 – з потрібного класу, то вона дорівнюватиме 0,1. Формула метрики precision має наступний вигляд:

$$precision = \frac{TP}{TP+FP}, \tag{2.2}$$

де TP – True positives, релевантні зображення, які були правильно знайдені;

FP – False positives, нерелевантні зображення, які були помилково визнані релевантними.

Recall же відображає, чи знайдено все, що очікувано. Якщо в результаті пошуку було знайдено всі 10 необхідних зображень, то recall дорівнюватиме 1,0. Якщо тільки 5 – то 0,5. Формула метрики recall має наступний вигляд:

$$recall = \frac{TP}{TP+FN}, \tag{2.3}$$

де TP – True positives, релевантні зображення, які були правильно знайдені;

FN – False negatives, релевантні зображення, які не були знайдені.

Метрика F1 є гармонійним середнім значенням кожної з вищеописаних метрик. Формула метрики F1 має наступний вигляд:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}. \quad (2.4)$$

Після визначення метрик для кожного класу на даній ітерації, було підраховано середнє значення кожної метрики.

Графік підрахунку метрик наведено на рисунку 2.14. На перетині кривих precision, recall та F1 знаходиться оптимальний поріг для пошуку.

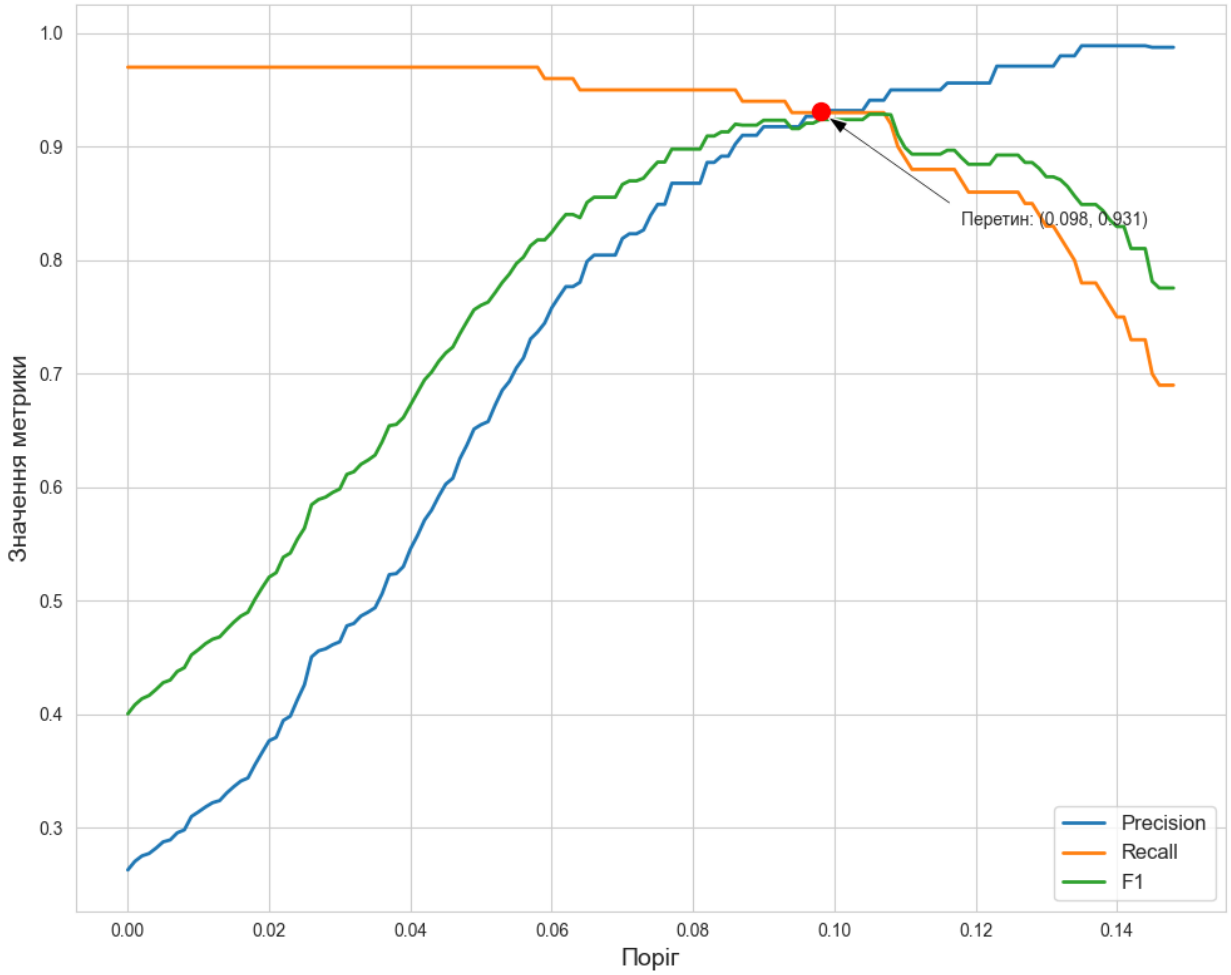


Рисунок 2.14 – Криві метрик precision, recall та F1

Згідно аналізу, оптимальним порогом міри подібності є 0,098. Саме це значення і буде використовуватись в застосунку для відсіювання нерелевантних результатів.

3 РОЗРОБКА ДЕСКТОПНОГО ЗАСТОСУНКУ ДЛЯ ПОШУКУ ЗОБРАЖЕНЬ ЗА ТЕКСТОВИМ ОПИСОМ

3.1 Вибір інструментарію для розробки програмного забезпечення

Для розробки десктопних застосунків існує безліч різних варіантів програмних продуктів, які допоможуть створити застосунок різного рівня. В епоху масової комп'ютеризації під час проєктування бажано охопити якомога більше користувачів, тому надійним рішенням для створення застосунку є використання кросплатформних технологій, тобто тих, що дозволяють надати готовий застосунок кінцевому користувачеві під різні операційні системи: Windows, macOS чи дистрибутиви Linux.

Вибір мови програмування, що використовуватиметься для розробки застосунку, також є вирішальним фактором як для якісної роботи програми, такі і для розробницького досвіду (DX – developer experience), показника, наскільки зручною є обрана платформа для програміста. Рейтинг популярних мов програмування TIOBE має топ-10 найпопулярніших мов, які наразі використовуються всесвітньо. Цей топ, що зображено на рисунку 3.1, показує, наскільки поширеною та використовуваною є наразі та чи інша мова.

Apr 2025	Apr 2024	Change	Programming Language	Ratings	Change
1	1		Python	23.08%	+6.67%
2	3	↑	C++	10.33%	+0.56%
3	2	↓	C	9.94%	-0.27%
4	4		Java	9.63%	+0.69%
5	5		C#	4.39%	-2.37%
6	6		JavaScript	3.71%	+0.82%
7	7		Go	3.02%	+1.17%
8	8		Visual Basic	2.94%	+1.24%
9	11	↑	Delphi/Object Pascal	2.53%	+1.06%
10	9	↓	SQL	2.19%	+0.57%

Рисунок 3.1 – Рейтинг популярних мов програмування TIOBE станом на квітень 2025 року [49]

Також на вибір мови програмування і рішення щодо архітектури застосунку впливають і самі технології, що використовуватимуться в програмі. Наприклад, обрана модель MobileCLIP B (LT) підтримує десктопний запуск лише у середовищах виконання мови Python [50].

3.1.1 Вибір платформи для розробки frontend частини

Мова Python також підтримує і створення програм з візуальним графічним інтерфейсом, наприклад, за допомогою використання бібліотек Qt [51] чи Tkinter [52], проте для створення більш функціонального інтерфейсу, до якого звик потенційний користувач в епоху інтернет-застосунків, було вирішено обрати вебтехнології у десктопному форматі. Для цього симбіозу існує фреймворк Electron [53], який вбудовує браузер Chromium [54] із середовищем виконання JavaScript та TypeScript коду Node.js [55] в єдину систему, що відображається користувачеві як суцільний віконний застосунок.

Так як цей програмний продукт уможливлює використання всіх сучасних технологій для створення вебзастосунків, а саме використання останніх стандартів HTML5, CSS та JavaScript, то було вирішено використати популярну бібліотеку React [56] для створення компонентних застосунків. В якості мови програмування для цієї frontend частини було обрано мову TypeScript [57], яка є надбудовою над стандартним JavaScript із явним вказанням типів об’єктів, чого не дозволяє робити її «родич». Ця мова є більш зручною для розробницького досвіду, адже завдяки підказкам сучасних середовищ розробки, які транспілюють мову TypeScript у JavaScript в реальному часі, про помилки у використанні мови стає відомо перед запуском програми, і це дозволяє зекономити час розробки та передбачити потенційні помилки під час виконання.

3.1.2 Вибір рішення для backend частини

Frontend частина застосунку буде поєднана із backend частиною, в якій відбуватиметься бізнес-логіка з маніпуляцією даними, за допомогою протоколу HTTP та підходу REST API (Representational State Transfer Application Programming Interface, інтерфейс програмування застосунків з передачею репрезентативного стану). Так як обрана модель ШІ може запускатись лише в середовищі Python, то вся інфраструктура backend так само використовуватиме цю мову програмування. В якості бібліотеки, що оброблятиме HTTP запити, було обрано FastAPI [58] через його легке налаштування та використання. Для обробки зображень та текстових запитів, а саме перетворення їх у вектори ознак, в розділі 2 було вирішено використовувати модель MobileCLIP B (LT), яка може бути викликана через спільний програмний інтерфейс моделей OpenCLIP.

Проіндексовані ембедінги зображень зберігатимуться в векторній базі даних Qdrant, що також надає програмний інтерфейс для використання в середовищі мови Python. Також інструментарієм цієї бази даних буде виконуватись і пошук згідно описаного алгоритму в підрозділі 2.10.1.

3.2 Проєктування застосунку

3.2.1 Опис функціональності та бізнес-кейсів програмного продукту

За логікою використання, застосунок має лише одного актора взаємодії – Користувача, який виконуватиме всі дії з графічного інтерфейсу.

Дії, які може виконувати Користувач, наведено на рисунку 3.2 – діаграмі варіантів використань (прецедентів). Ці варіанти є основною для подальшого моделювання різних частин програмного продукту в наступних підрозділах.

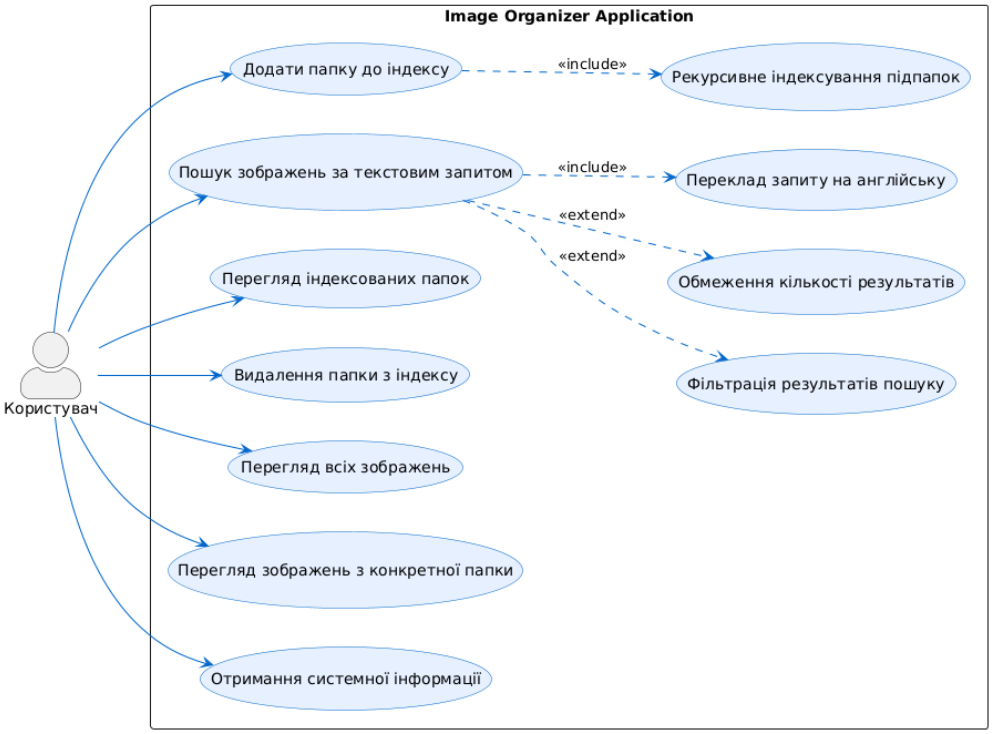


Рисунок 3.2 – Діаграма варіантів використань застосунку

3.2.2 Моделювання застосунку

На вищому рівні backend-частини існує сервер FastAPI, який приймає всі HTTP-запити від клієнта і викликає бізнес-логіку відповідно до отриманого запиту. На сервері наявні такі відкриті вхідні точки:

– GET / – перевірка доступності серверу;

– POST /folders/add – додає каталог із зображеннями до індексу для пошуку. Приймає тіло запиту, що складається з полів path (шлях до папки) та флаг recursive (визначає, чи треба індексувати вкладені директорії);

– GET /folders – повертає список всіх папок, які були додані до індексу;

– DELETE /folders/{folder_path} – видаляє папку та всі її зображення з індексу. Приймає параметр folder_path (шлях до папки), що мусить бути URL-кодованим [59];

– POST /search – здійснює пошук зображень за текстовим описом. Приймає тіло запиту, що складається з полів query (текстовий опис), limit

(опціонально, максимальна кількість повернених зображень);

– GET /images – повертає всі проіндексовані зображення;

– GET /folders/{folder_path}/images – повертає всі зображення з вказаної папки. Приймає параметр folder_path (шлях до папки), що мусить бути URL-кодованим [59];

– GET /system/info – повертає інформацію про розмір бази даних та використовувану модель.

Так як на frontend-частині, яка надсилає ці запити до серверу, використовуються вебтехнології, то там також існує система шляхів (роутингу), а саме:

– початкова сторінка, головне меню – «/»;

– екран з деревом проіндексованих директорій – «/folders/*»;

– сторінка пошуку за текстовим описом – «/search»;

– сторінка перегляду всіх проіндексованих зображень – «/all».

На рисунку 3.3 наведено діаграму взаємодії компонентів системи та Користувача із системою.

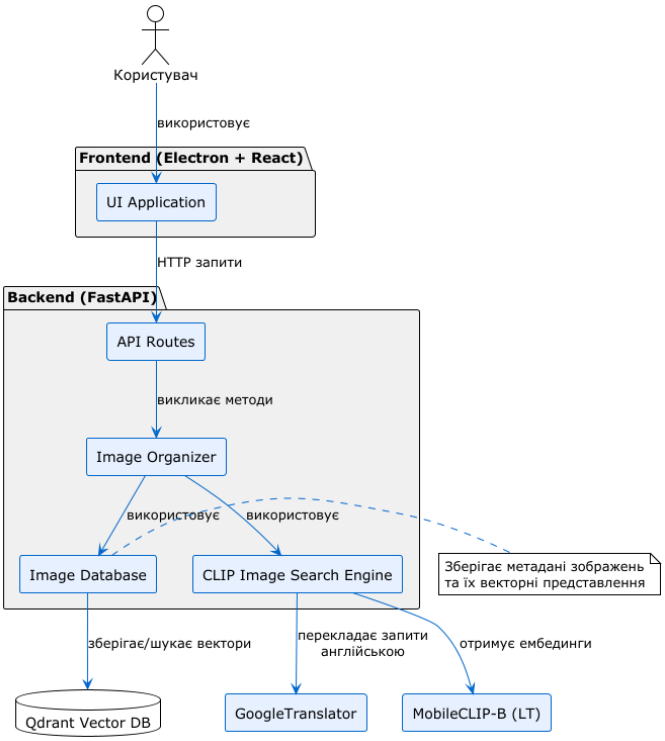


Рисунок 3.3 – Діаграма взаємодії компонентів системи

На рисунку 3.4 наведено діаграму класів backend-частини застосунку з детальним описом властивостей та методів кожного класу.

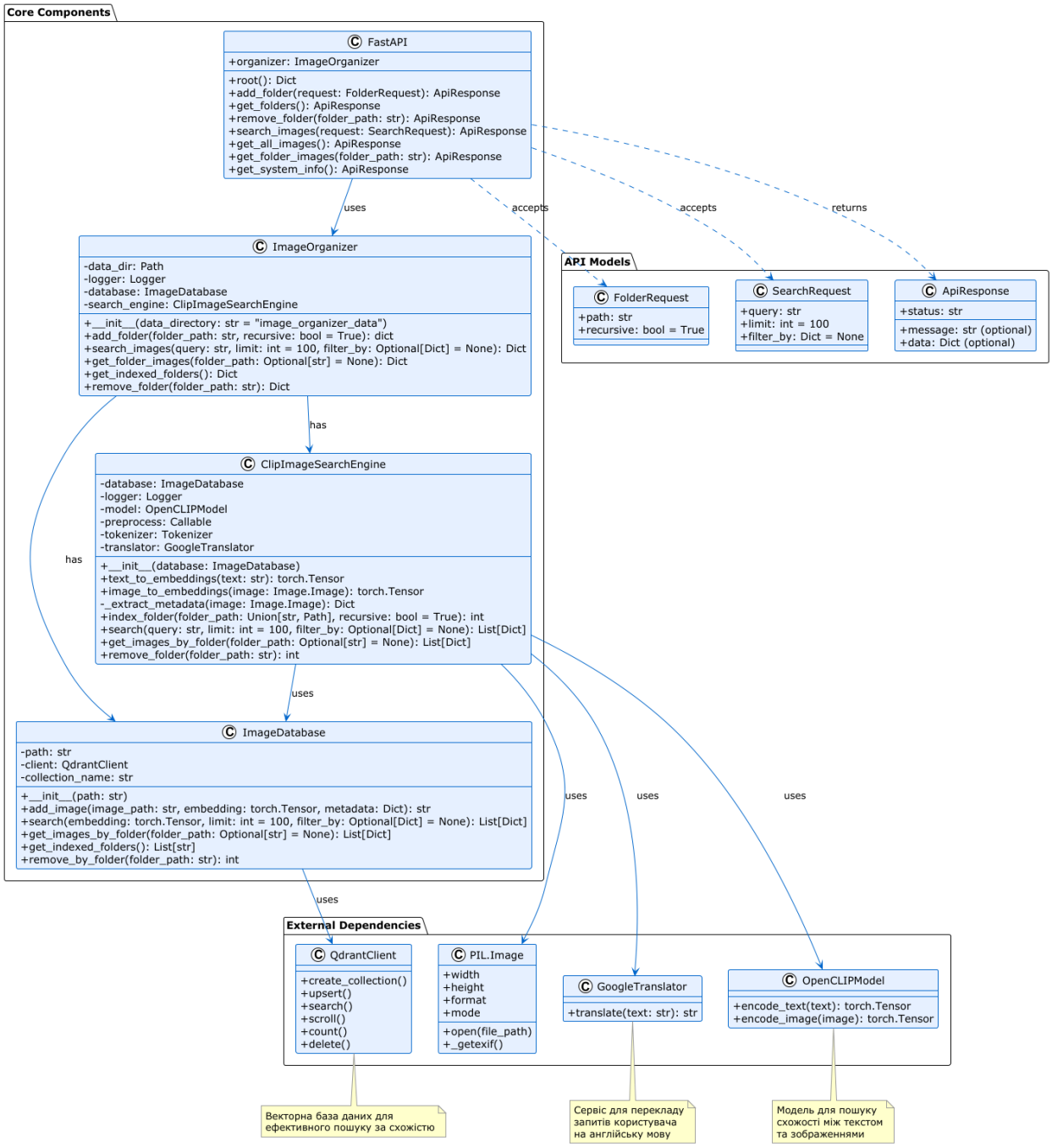


Рисунок 3.4 – Діаграма класів backend-частини застосунку

На рисунку 3.5 наведено діаграму послідовності для індексації зображень. На рисунку 3.6 наведено діаграму послідовності для пошуку зображень.

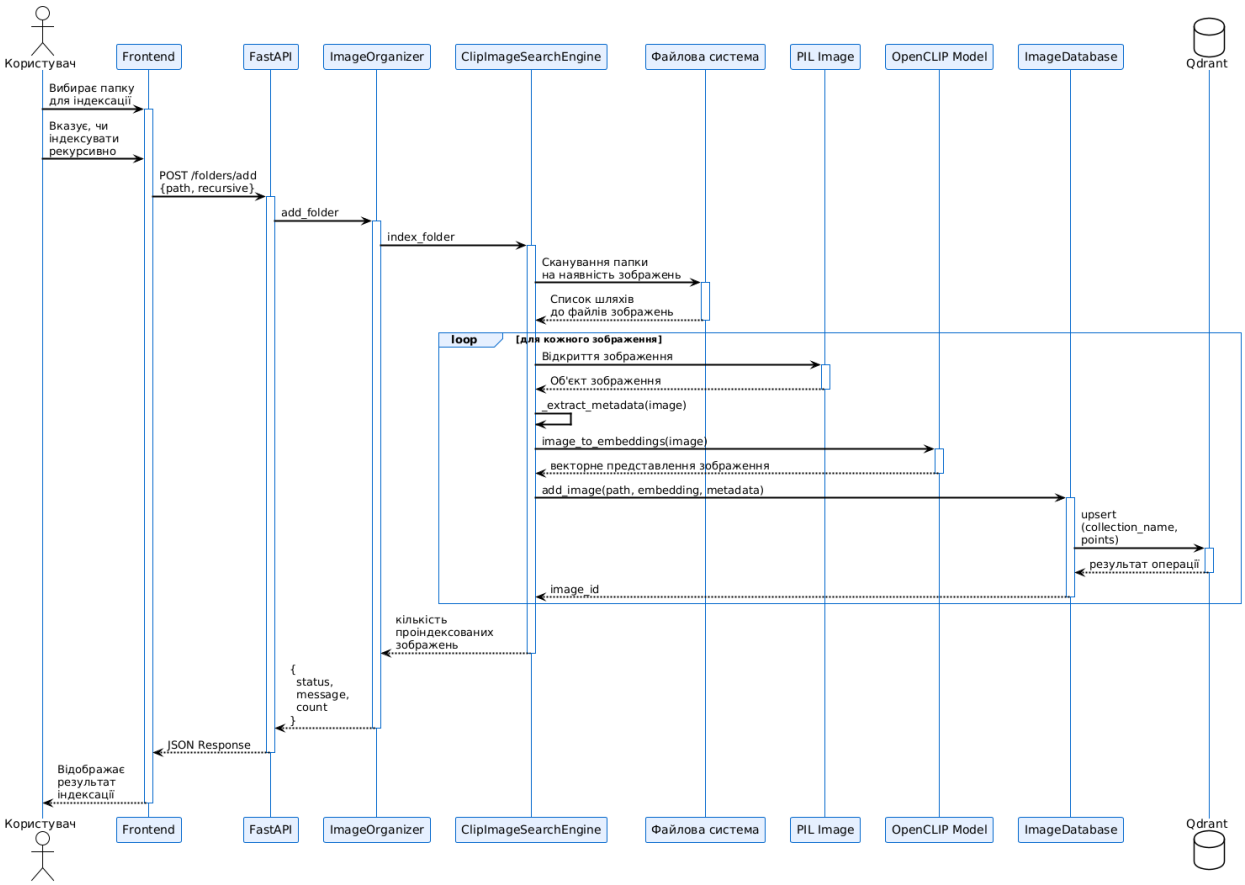


Рисунок 3.5 – Діаграма послідовності для функціоналу індексації зображень

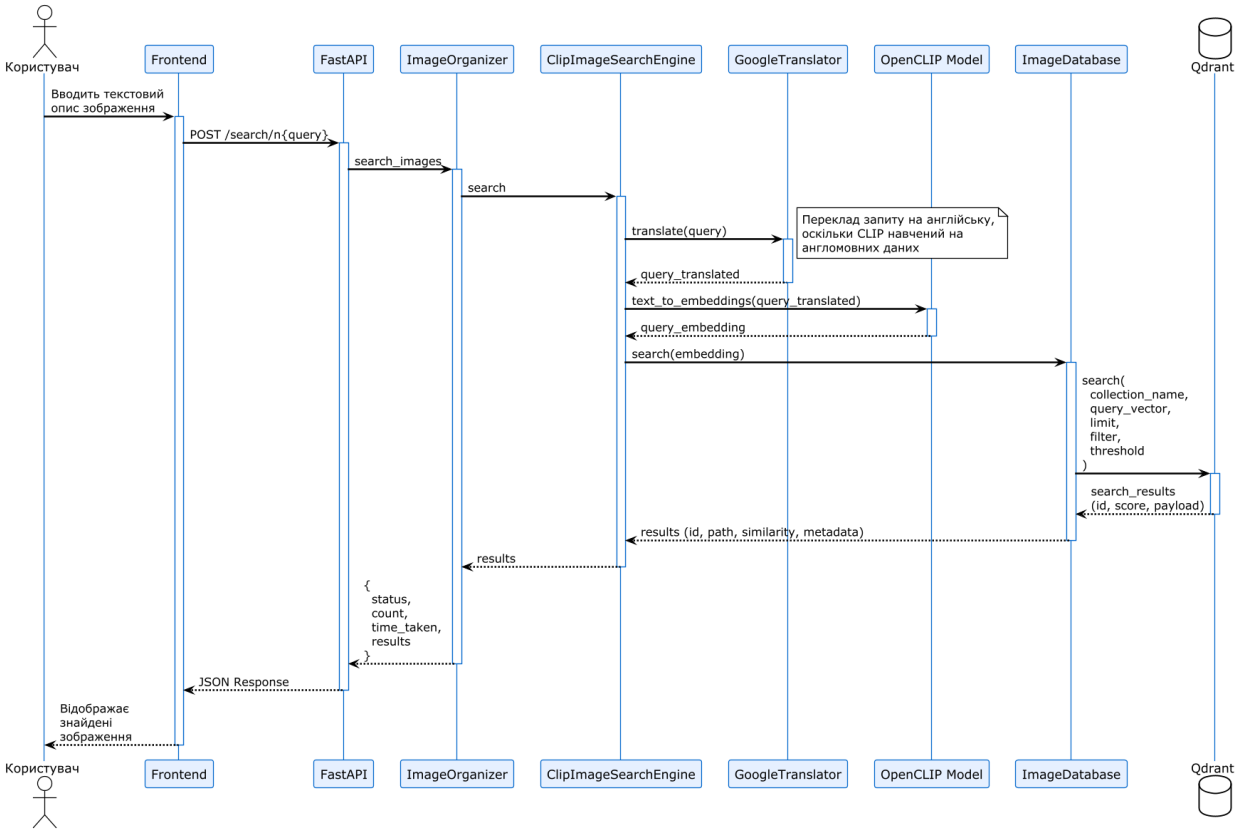


Рисунок 3.6 – Діаграма послідовності для функціоналу пошуку зображень

На рисунку 3.7 наведено діаграму послідовності для перегляду всіх зображень з обраної директорії.

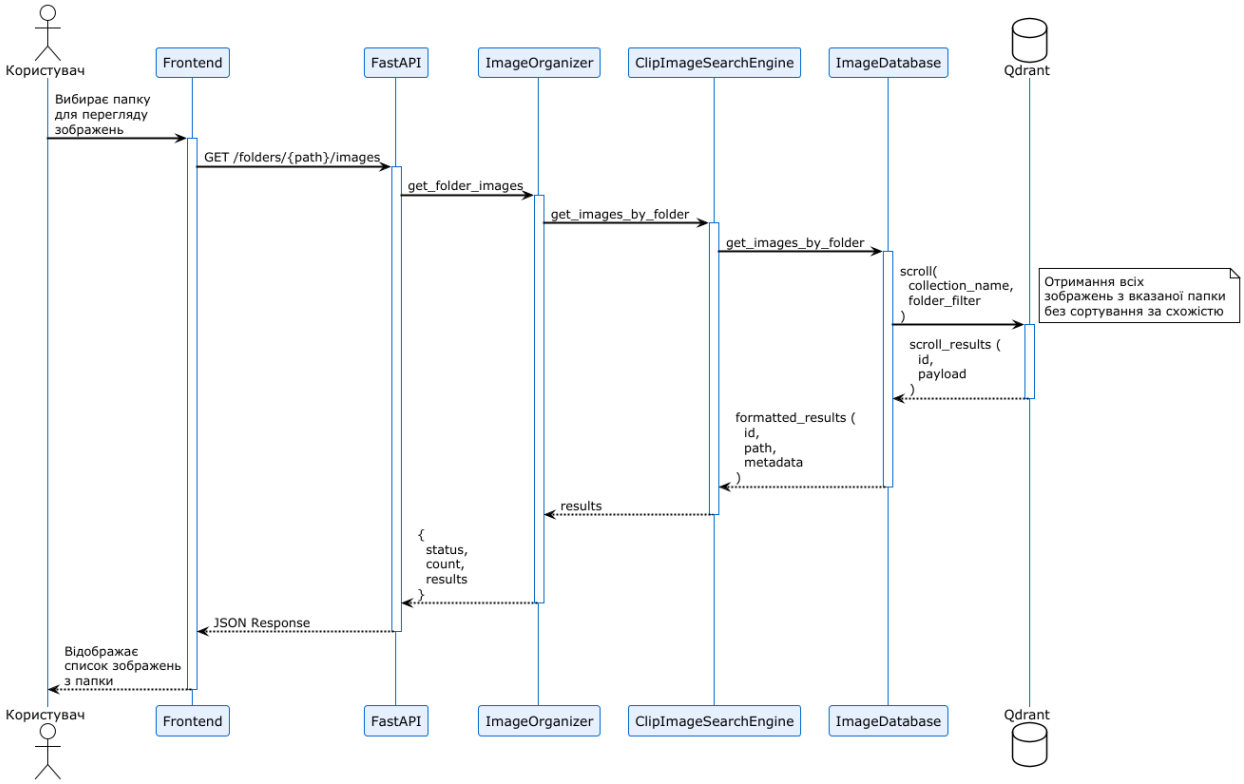


Рисунок 3.7 – Діаграма послідовності для функціоналу перегляду зображень з папки

Всі зображення повертаються у вигляді рядку шляху для оптимізації навантаження на систему, яка не повинна повертати повний файл зображення. І вже frontend-частина визначає, коли необхідно завантажувати картинку під час процесу рендерингу. Також, під час відображення багатьох картинок одночасно (наприклад, у результатах пошуку), підвантажуються не повнорозмірні варіанти, а ескізи розміром до 300 пікселів.

Важливим етапом для початку роботи застосунку є ініціалізація під час запуску. Так як клієнтська частина не може бути функціональною без працюючого серверу, під час відкриття вікна необхідно створити процес Python-серверу та чекати, поки він повністю не запуститься. Повна діаграма стартової ініціалізації наведена на рисунку 3.8.

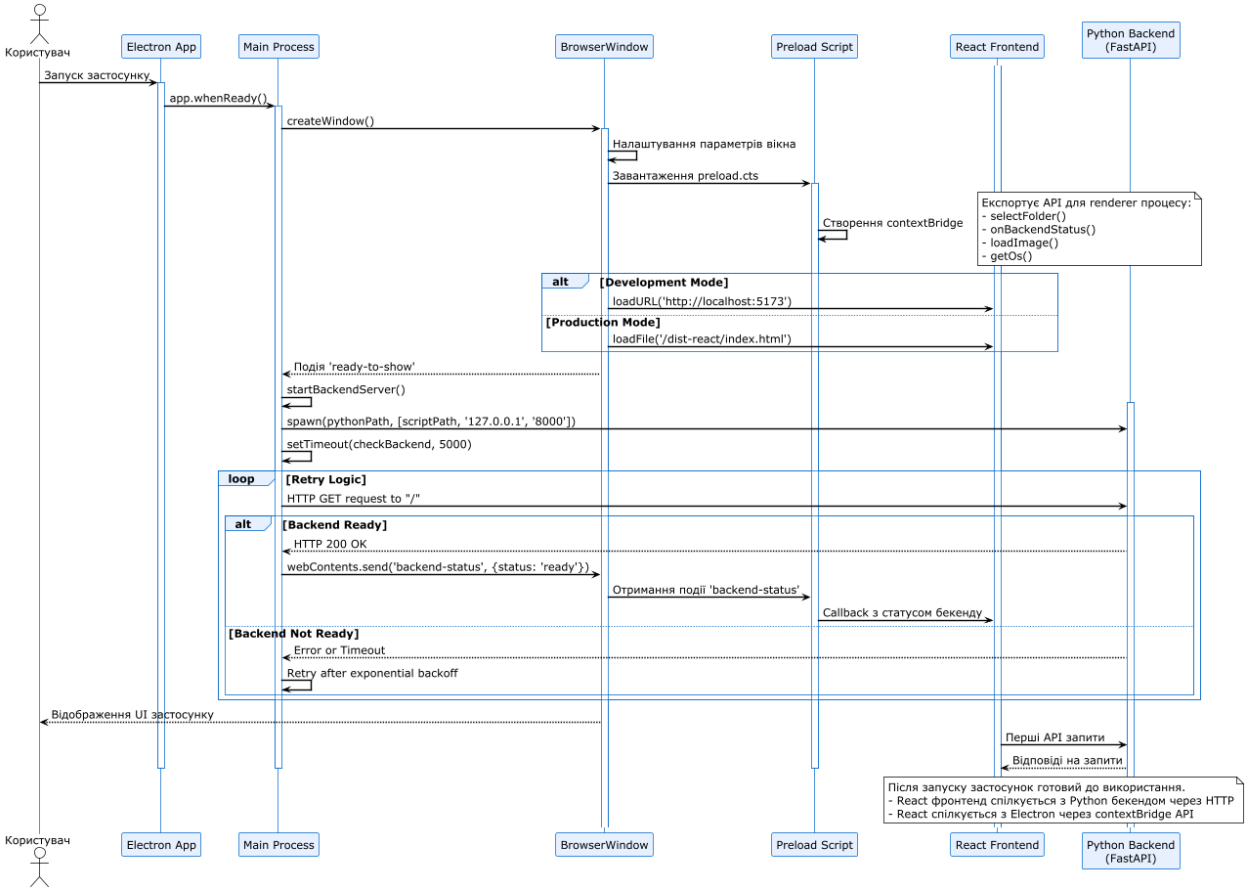


Рисунок 3.8 – Діаграма запуску застосунку

Під час роботи застосунку, клієнт відправляє HTTP-запити на сервер за локальною адресою localhost:8000 та відображає отримані результати.

React застосунок не може явно отримати контент за шляхом, що йде до локального сховища на пристрої виконання. Тому для завантаження зображень використовується основна технологія Electron – контекстний міст, що захищає вебрушій від прямого використання системних інтерфейсів. А саме так, що React частина звертається до розширеного під час ініціалізації JavaScript-об’єкту Window, в якому створено метод loadImage. Він приймає параметр шляху фотографії, обробляє цю картинку (завантажує із системи, зменшує розмір та стискає форматом JPEG із якістю 70%) і повертає готовий ескізний варіант в форматі base64, який легко вбудовується в компонент зображення під час рендерингу.

3.3 Ілюстрація роботи

Застосунок стартує запуском команди npm run dev з директорії розташування коду. Після цього, відкривається вікно (рис. 3.9) та паралельно стартує backend частина. Frontend очікує, доки сервер запуститься, запитуючи статус кожні 3 секунди. Якщо протягом трьох спроб сервер не зможе запуститись, програма відобразить помилку старту.

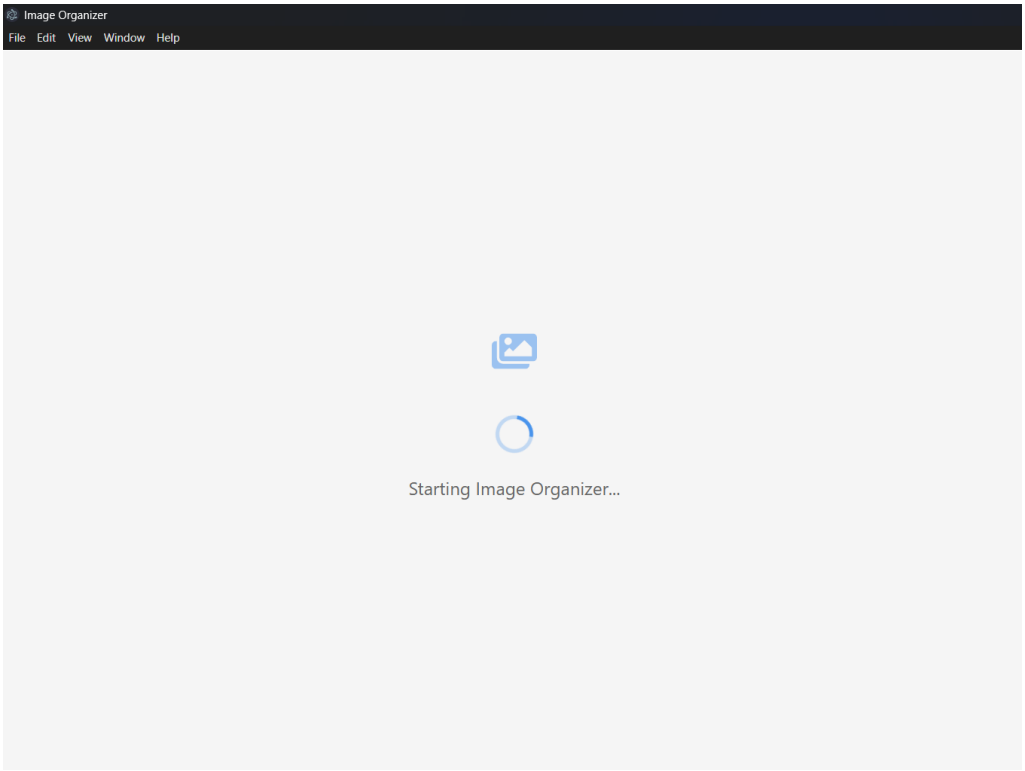


Рисунок 3.9 – Вікно запуску застосунку

Після старту, програма відобразить головне меню (рис. 3.10) з основними діями: «управління папками», «пошук зображень», «відображення всіх зображень». Також внизу меню наведено основні статистичні дані, скільки всього директорій проіндексовано, який розмір бази даних та яка модель використовується для створення ембедингів.

При виборі першої дії з управління папками, відкривається дерево файлової системи, в якому відображаються шляхи лише проіндексованих директорій (рис. 3.11).

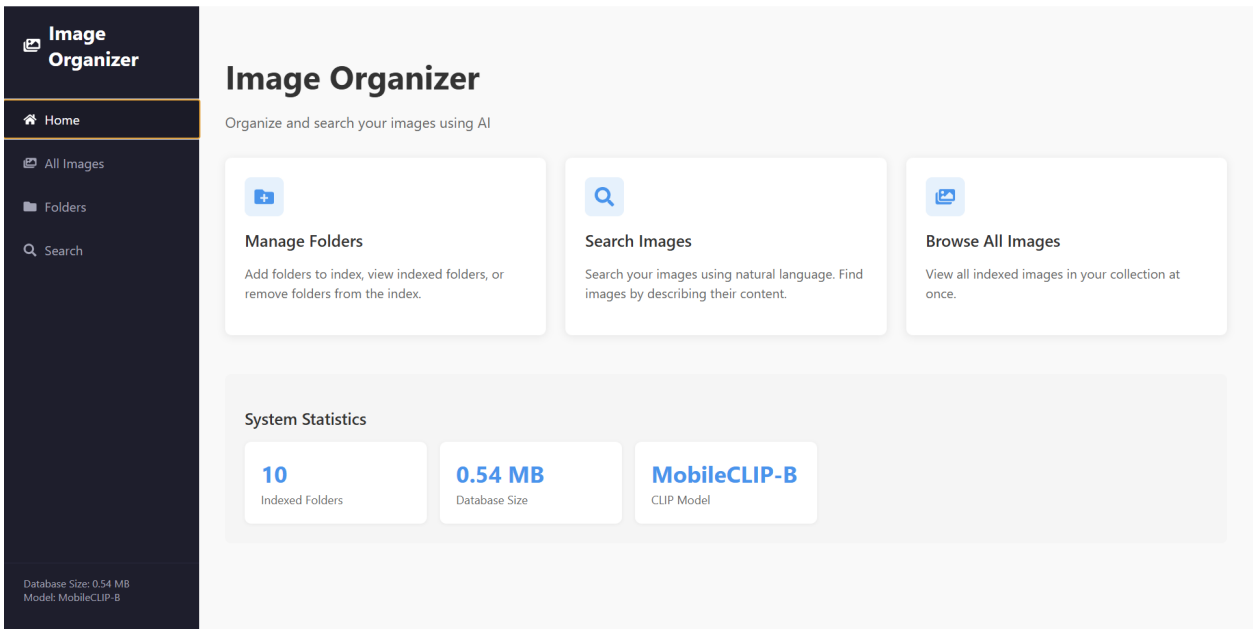


Рисунок 3.10 – Головне меню програми

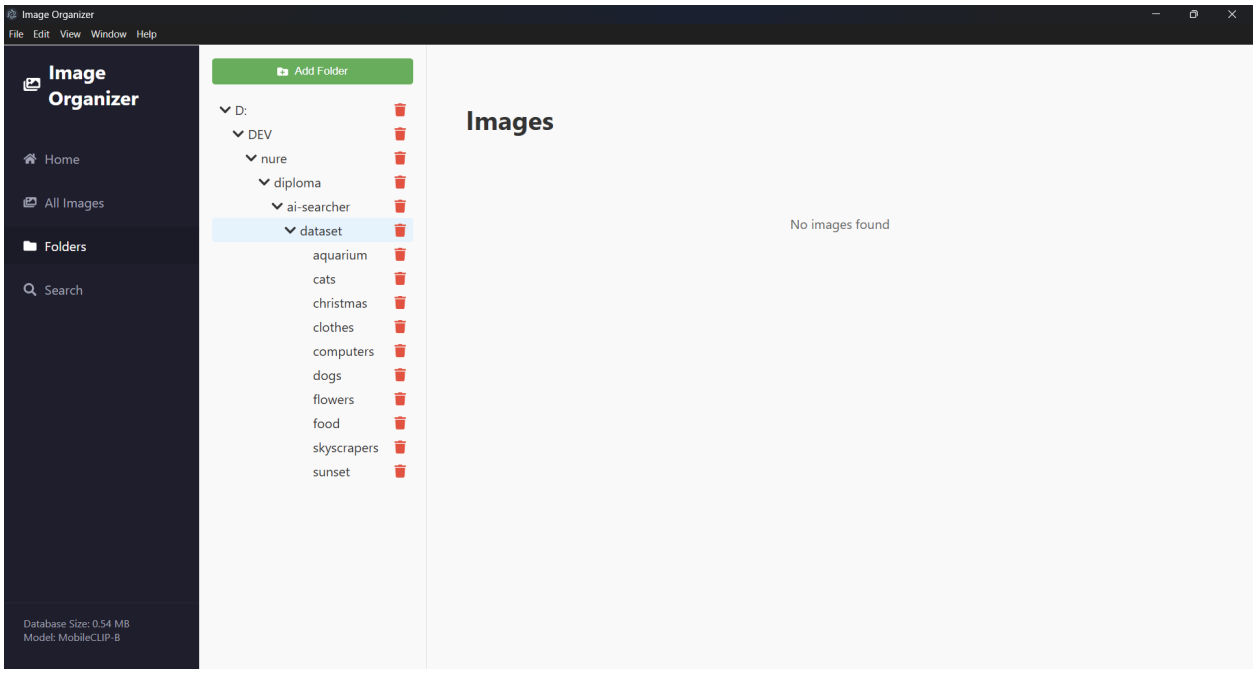


Рисунок 3.11 – Меню каталогів

Після натиснення на назву каталогу, якщо в ньому є проіндексовані зображення, вони відображатимуться у сітці (рис. 3.12).

При натисненні на фотографію, відкривається повнорозмірний переглядач із додатковою інформацією, що було збережено під час індексації. Ця сторінка відображена на рисунку 3.13.

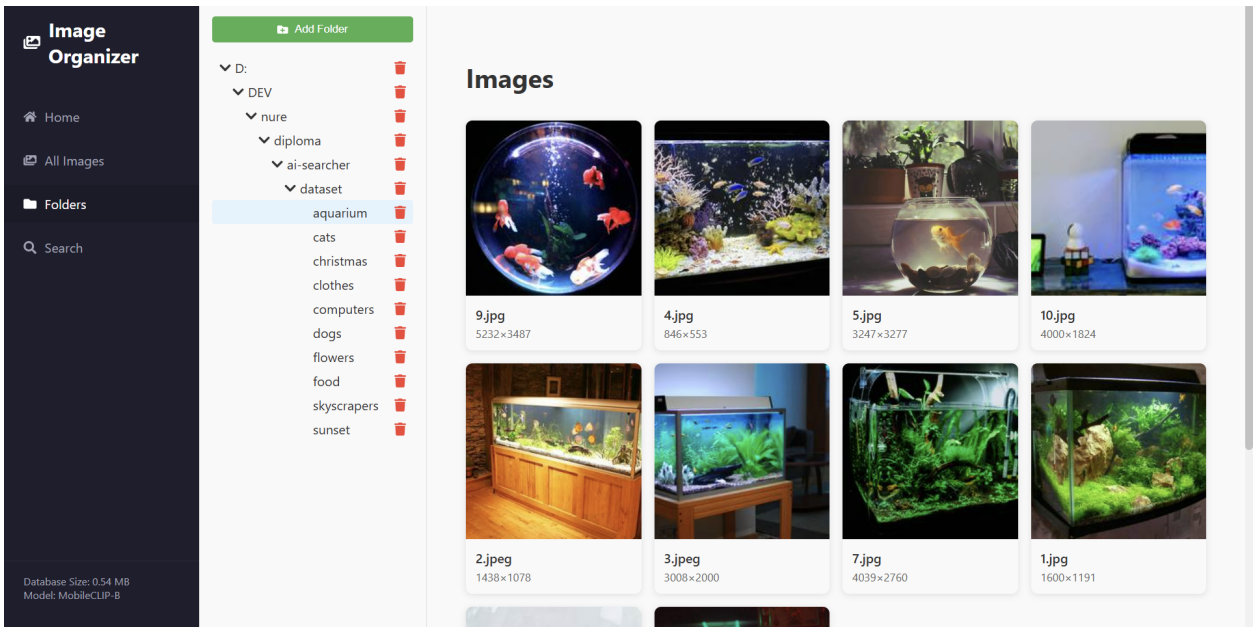


Рисунок 3.12 – Меню каталогів із обраною директорією з проіндексованими зображеннями

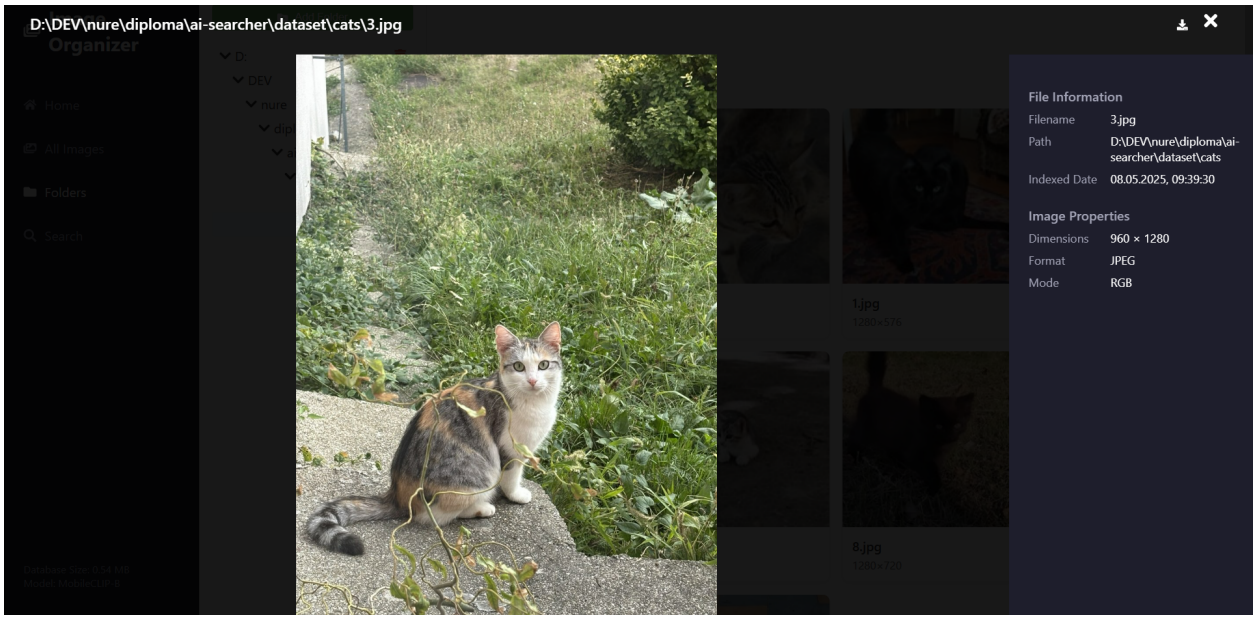


Рисунок 3.13 – Повнорозмірний перегляд з метаданими

Повертаючись в головне меню, при виборі «Пошук зображень», відкривається сторінка з полем для вводу текстового запиту (рис. 3.14).

Після введення текстового запиту та натискання на кнопку «Пошук», відбувається запит на сервер, перекладання запиту на англійську мову, створення текстового вектору ознак, пошук за ним у векторній базі даних та

повернення результатів, обмежених за вказаною мірою схожості. Отриманий результат у вигляді масиву оброблюється клієнтською частиною та відображається користувачеві.

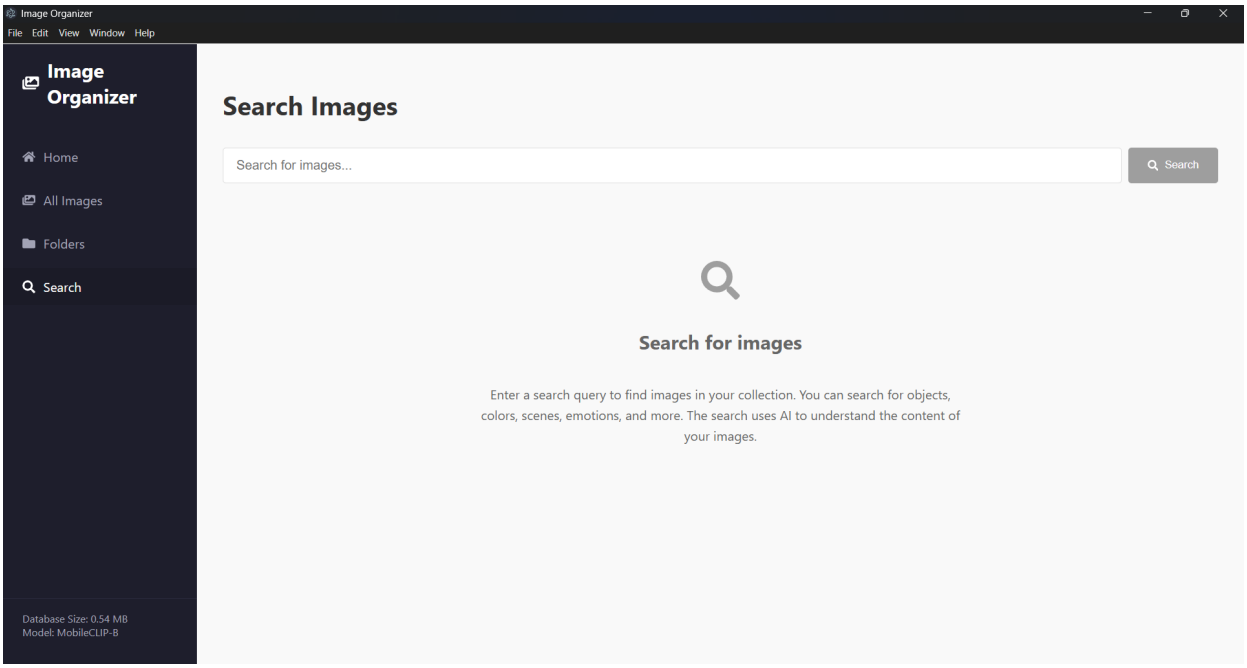


Рисунок 3.14 – Сторінка пошуку

Сторінка з результатами пошуку проілюстрована на рисунках 3.15, 3.16.

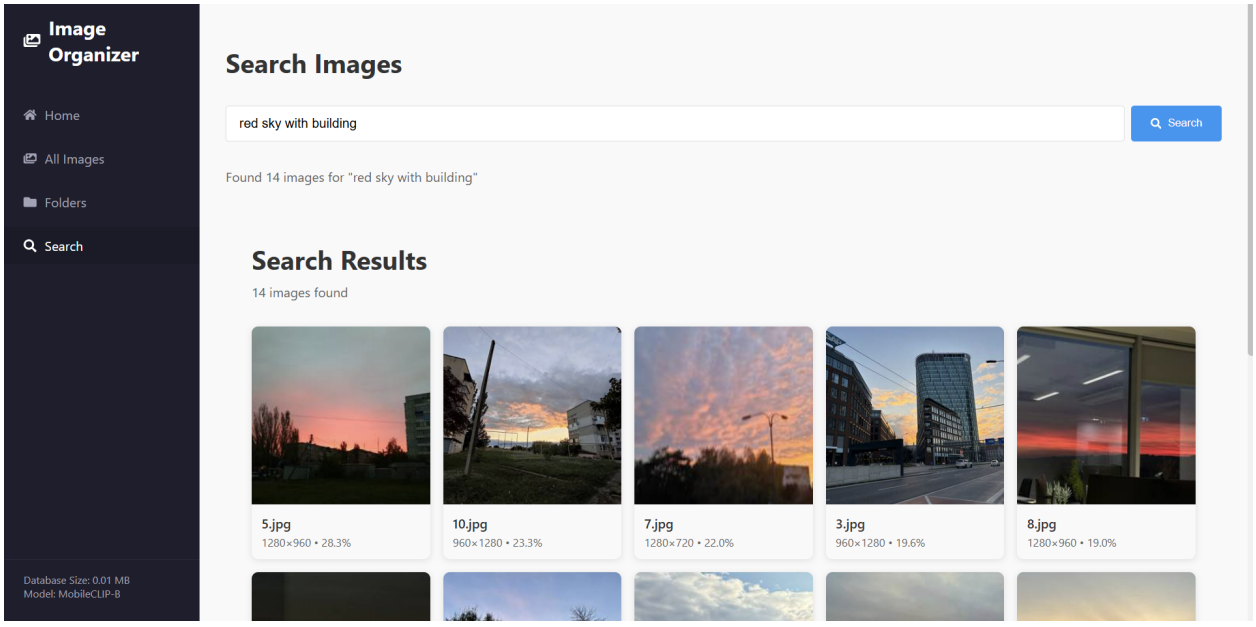


Рисунок 3.15 – Результат пошуку англійською мовою

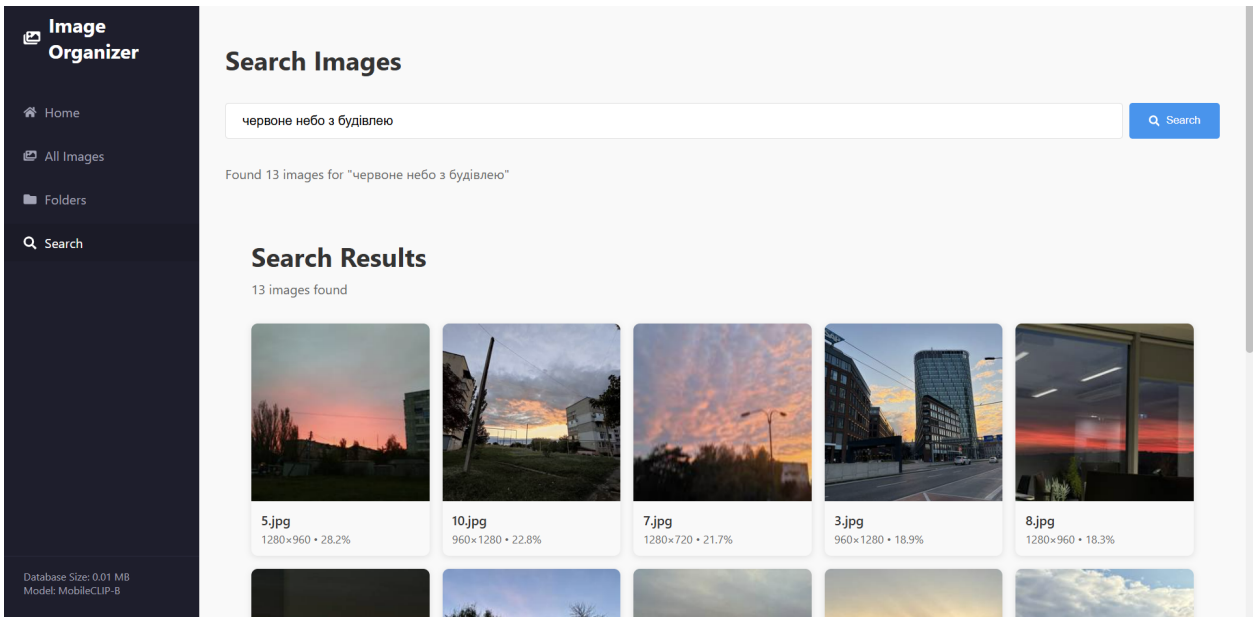


Рисунок 3.16 – Результат пошуку українською мовою

Для додавання зображень в базу даних системи, необхідно перейти на вкладку «Управління папками» та натиснути кнопку «Додати папку». Відкриється модальне вікно (рис. 3.17) з проханням обрати каталог та за бажанням проіндексувати не тільки файли на першому рівні, а піти вглиб та проаналізувати всі файли у вкладених директоріях.

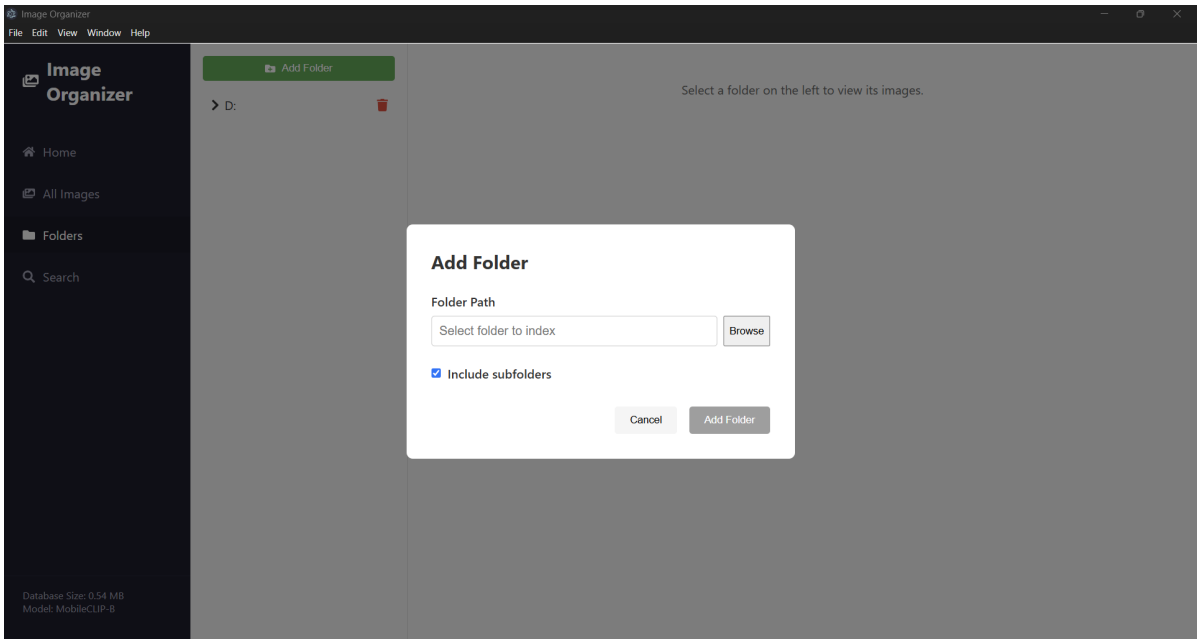


Рисунок 3.17 – Модальне вікно вибору директорії для індексації

Для виклику системного діалогового вікна з вибором каталогу (рис. 3.18) також було використано можливість контекстного мосту, згаданого вище, для виклику функції Electron.

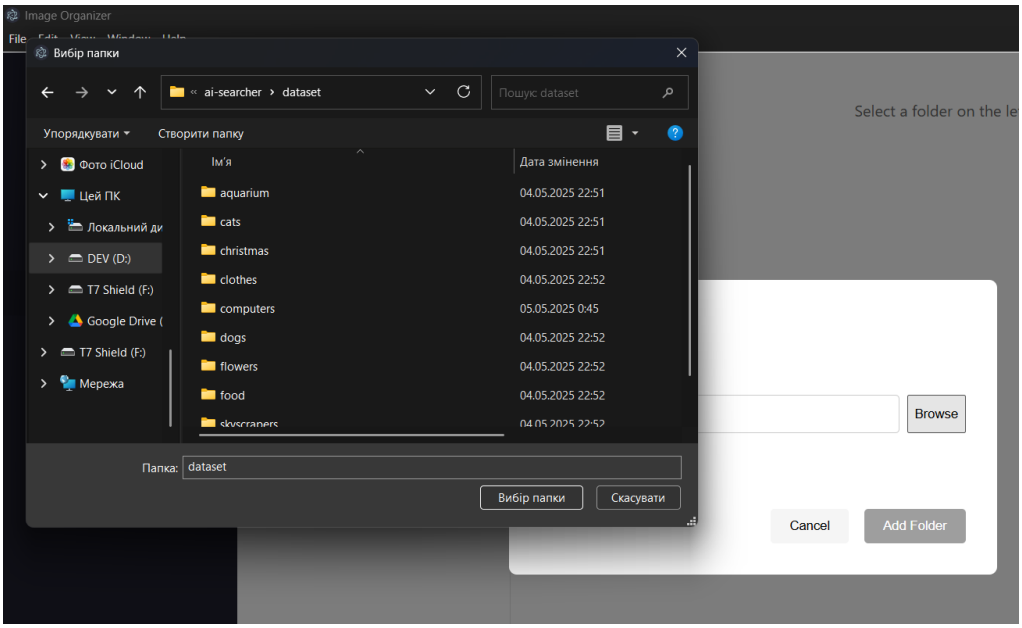


Рисунок 3.18 – Виклик системного діалогового вікна для вибору директорії

Після підтвердження вибору, frontend частина надсилає запит про додавання файлів на сервер, сервер надсилає статус про готовність та починає роботу з індексації файлів. Тим часом для користувача застосунок відображає статус, що індексація наразі відбувається (рис. 3.19). Після успішної індексації в застосунку відображається повідомлення про результат (рис. 3.20).

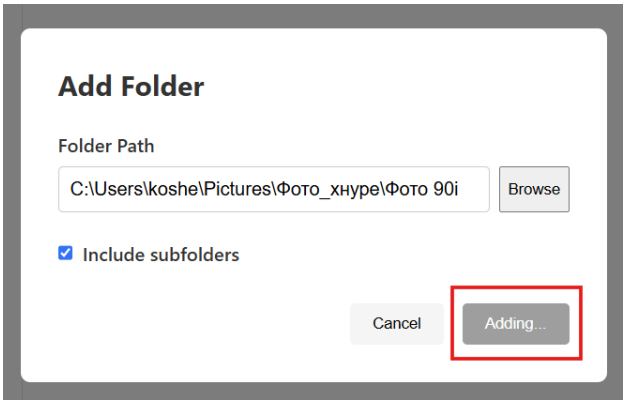


Рисунок 3.19 – Статус індексації

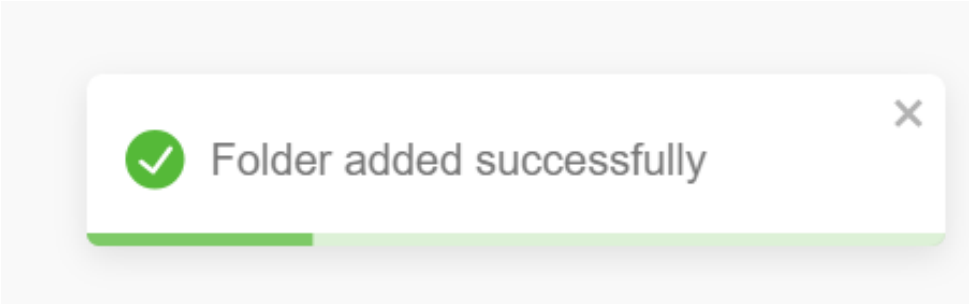


Рисунок 3.20 – Повідомлення про результат

3.4 Аналіз результатів пошуку

3.4.1 Оцінка точності

Подальшим етапом після розробки є тестування застосунку. Оцінка точності пошуку є критичною для правильної роботи програмного продукту. Для перевірки точності було створено окремий набір даних методом, описаним в підрозділі 2.2. Цей набір складається з десяти аналогічних класів, по п’ять зображень на кожний клас. Всі вони були проіндексовані системою для тестування, що повторює бізнес-логіку застосунку з використанням порогу, що був отриманий в підпункті 2.11.

Першою метрикою точності пошуку є Recall@1 – визначення, чи на першому місці знаходиться знайдене за його ж описом зображення. Це бінарна метрика, що відповідає на запитання «так» або «ні». В середньому для всіх 50 фотографій Recall@1 становить 94%, що є результатом, прийнятним для даного роду застосунків.

Друга метрика точності пошуку – Precision@5, Recall@5 та їхнє збалансоване середнє F1. Вони показують точність та повноту зображень, що повинні бути знайденими за середніми описами кожного класу, обраними в підрозділі 2.9. Результати цих метрик наведено в таблиці 3.1.

Так як краще всього спиратись на метрику F1, середній результат по всіх класах складає 95,48%, що є дуже добрим показником точності знаходження зображень для класу.

Таблиця 3.1 – Результати метрик тестування пошуку зображень в рамках класів

Клас	**Precision@5**	**Recall@5**	**F1**
Акваріум	100%	100%	100%
Коти	100%	100%	100%
Різдво	100%	80%	88,89%
Одяг	100%	100%	100%
Комп’ютери	100%	100%	100%
Собаки	100%	100%	100%
Квіти	83,33%	100%	90,90%
Їжа	100%	60%	75%
Хмарочоси	100%	100%	100%
Заходи сонця	100%	100%	100%
В середньому	**98,33%**	**94%**	**95,48%**

3.4.2 Оцінка швидкості

В ході тестування було заміряно показники швидкодії індексації, пошуку і окремо створення ембедингів тексту та зображення. Експерименти проводились на різній кількості даних, після ітерацій проводилось обчислення середнього значення метрики. Комп’ютер для тестування має центральний процесор Intel Core i5-12500H, 16 Гб оперативної пам’яті та встановлену ОС Windows, що є пристроєм середнього класу із гідною продуктивністю.

Оцінка часу створення ембедингів зображень складається з прогону 10, 50 та 100 зображень через модель MobileCLIP-B (LT) з обрахунком середнього часу кодування інформації та середньої кількості оброблених зображень в секунду. Результати цього експерименту наведено на рисунках 3.21 і 3.22. Згідно результатів складається висновок, що середній час є відносно сталим та коливається в межах 0,2 секунди на зображення, що є швидким показником.

Схожі результати можна спостерігати і для створення текстових ембедингів. В рамках цього експерименту було обраховано середній час обробки всіх текстових описів без перекладача, з перекладачем та загального часу.

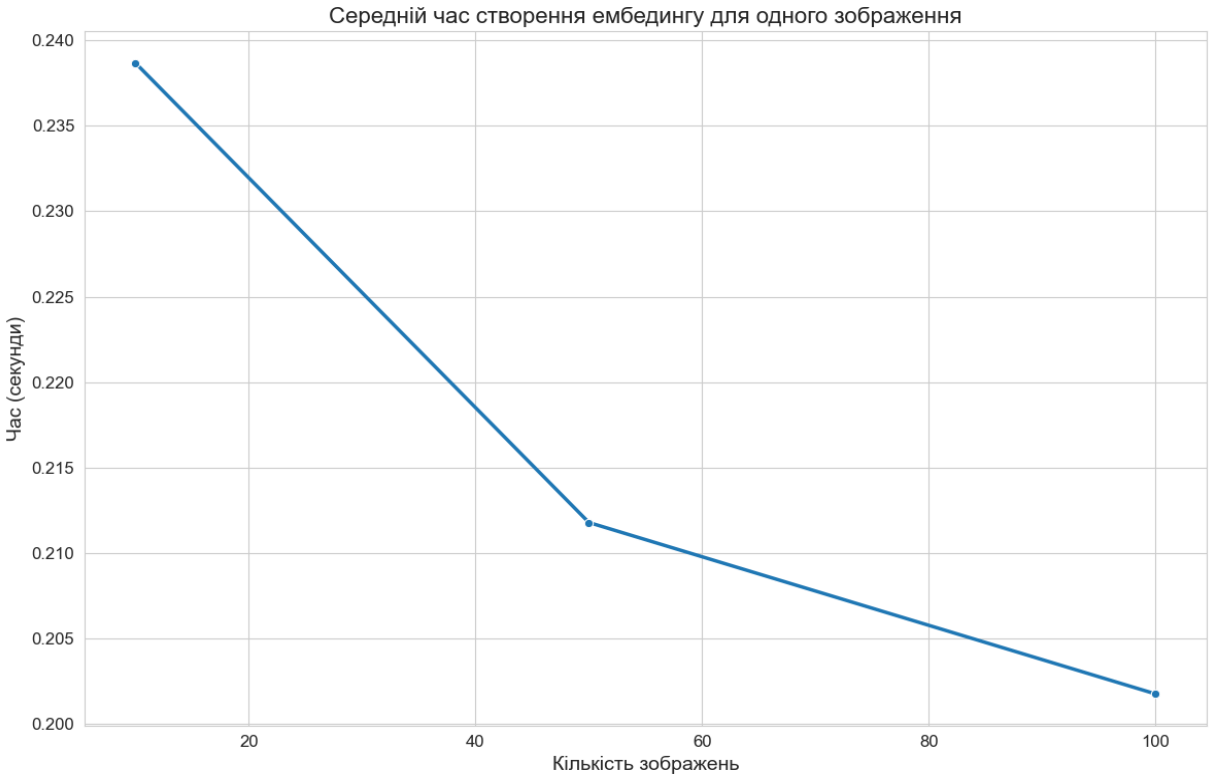


Рисунок 3.21 – Діаграма залежності середнього часу створення ембедингу від кількості зображень

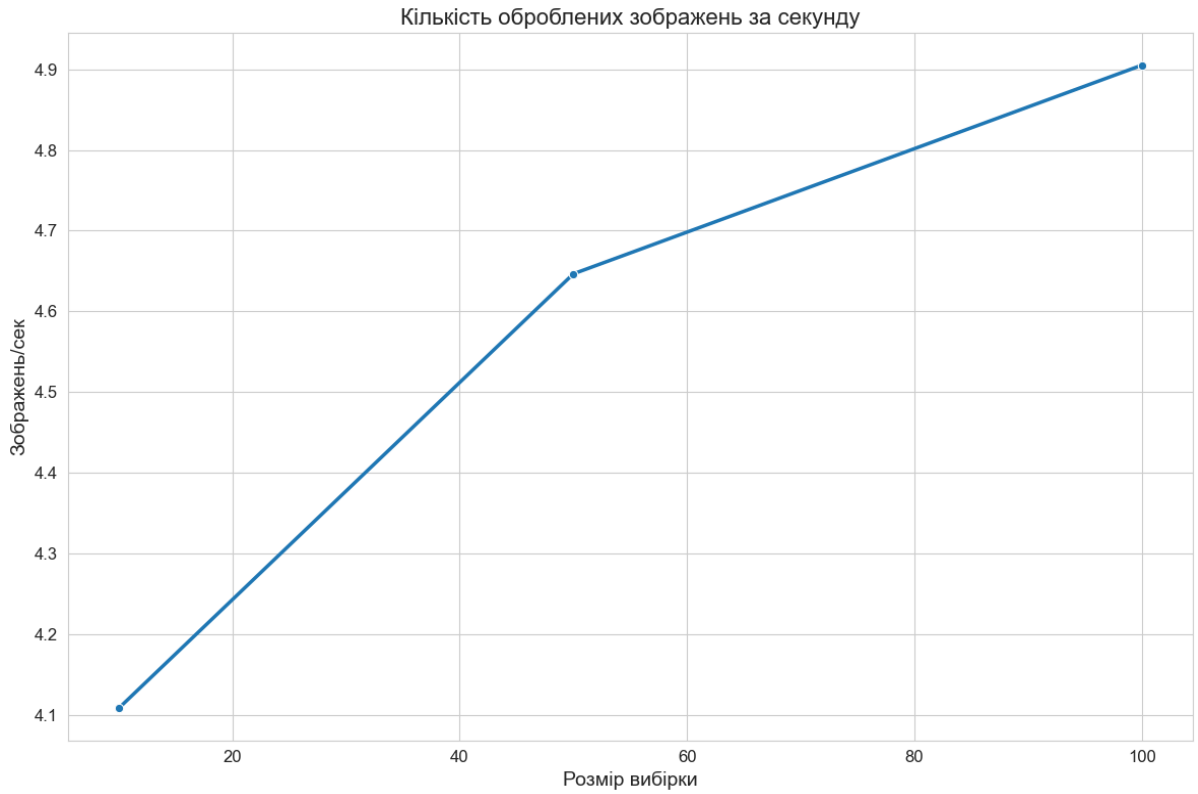


Рисунок 3.22 – Діаграма залежності кількості створених ембедингів від розміру вибірки

Результати, згруповані за кількістю слів у описі, наведено на рисунку 3.23. Згідно них, середній час кодування та перекладу не залежить від кількості лексем у реченні. Загальний час кодування тексту знаходиться в межах 0,25 – 0,3 секунди.

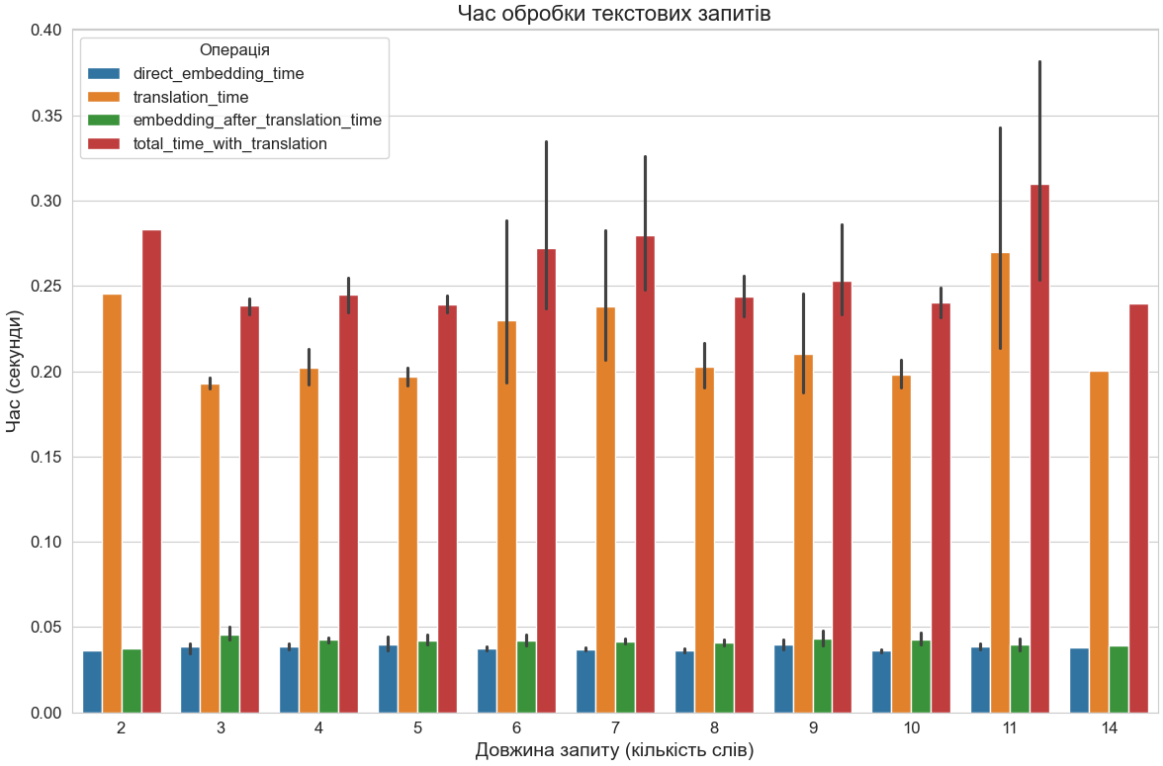


Рисунок 3.23 – Діаграма залежності середнього часу кодування текстових описів від кількості лексем

Швидкодія індексації зображень – це метрика середнього часу запису в векторну базу даних. Випадковим чином було згенеровано 1 000 000 векторів з характеристикою, аналогічною векторам моделі MobileCLIP – 512 вимірів зі значеннями типу float32 від 0 до 1. Було проаналізовано такі розміри даних: 100, 1 000, 10 000, 50 000, 100 000, 150 000, 200 000, 500 000, 750 000 та 1 000 000 додавань в базу. Результати аналізу швидкодії індексації наведено на рисунку 3.24. Після 100 000 є просадка в швидкості індексації вдвічі. Але кожний ембединг додавався за час в проміжку між 0,001 та 0,003 секунди, що в реаліях є моментальним та не відчутним. Ця кількість залежить від кількості зображень та алгоритму обробки бази даних.

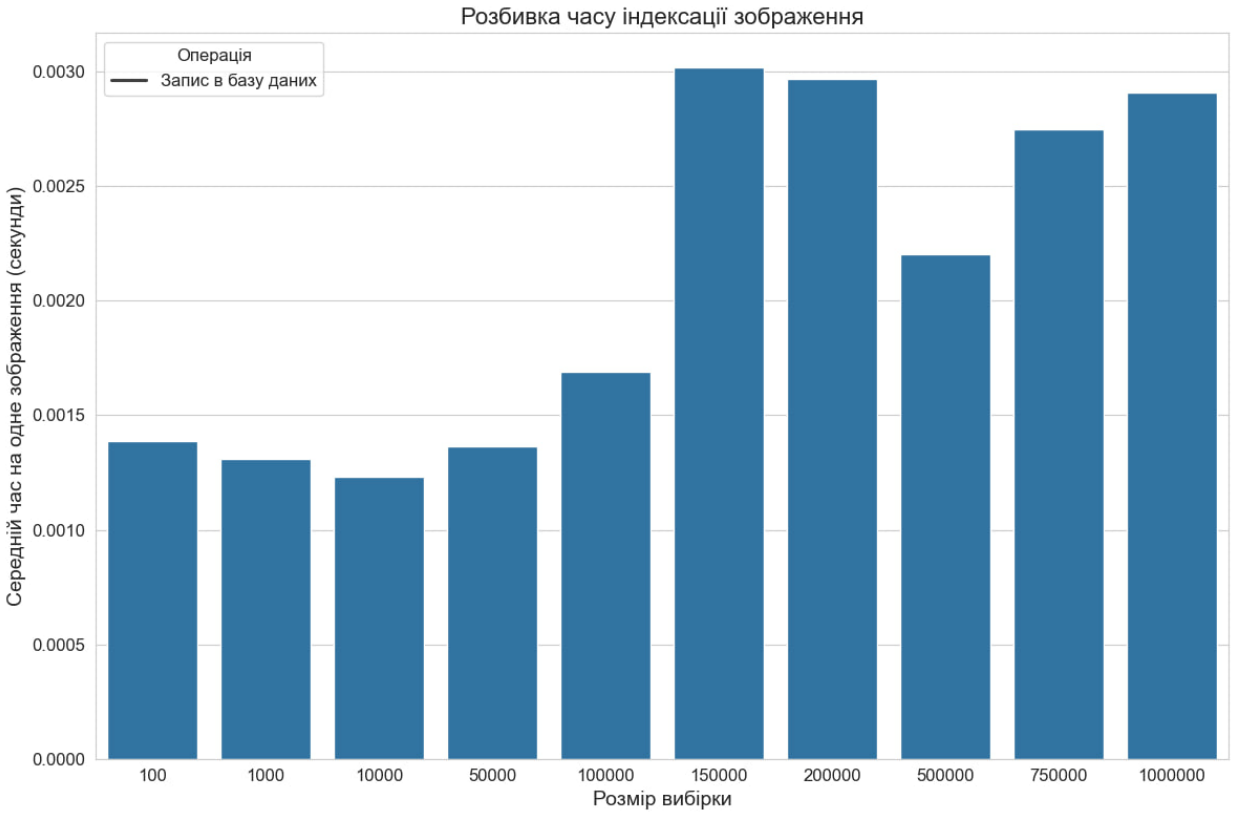


Рисунок 3.24 – Діаграма залежності середнього часу індексації від розміру індексованого набору даних

Наступний експеримент показує швидкість пошуку випадково згенерованого вектору в базі даних з різною наповненістю: 100, 1 000, 10 000 та 100 000 об’єктів. Для кожного розміру проводилось 10 пошукових запитів з метою обрати середнє значення з отриманих метрик швидкості. Результати відображені на рисунку 3.25.

Для різних кількостей векторів в базі даних середня швидкість є приблизно однакова, а саме коливається від 1 до 6 мілісекунди. Також швидкість залежить від параметрів пошуку, а саме від ліміту елементів. Наприклад, для 100 000 об’єктів в базі пошук, обмежений десятьма результуючими векторами, є на 1 мілісекунду швидше за пошук сотні векторів. Це є гарним результатом швидкодії, що забезпечує прийнятний рівень взаємодії з програмним продуктом.

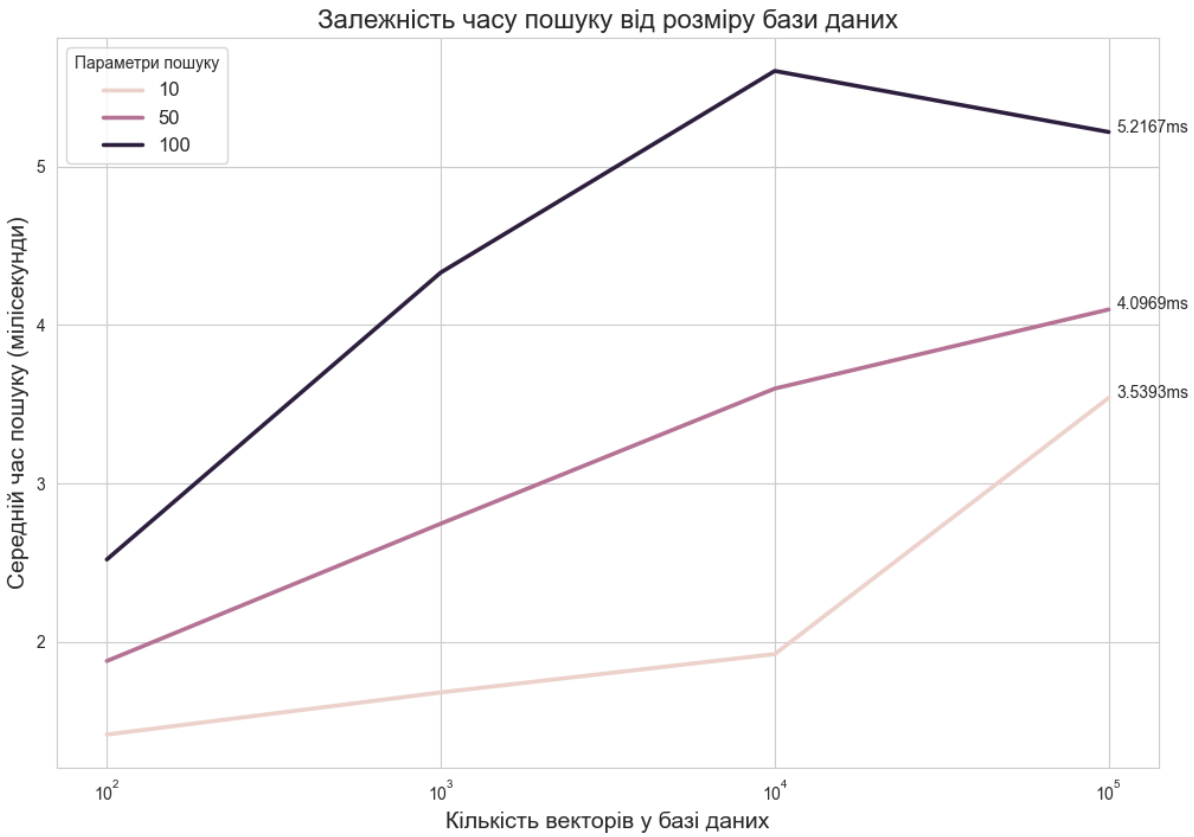


Рисунок 3.25 – Діаграма залежності середнього часу пошуку від наповненості бази даних

Отже, після проведених тестувань точності та швидкодії можна скласти висновок, що застосунок надає релевантні результати для інших наборів даних з тих же класів, а також головні бізнес-процеси, такі як створення ембедингів, індексація та пошук, відбувається швидко та не знижують зручність чи ефективність застосування програмного рішення.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений застосунок для пошуку зображень з локального сховища персонального комп’ютера за текстовим описом. У ході виконання кваліфікаційної роботи, було проаналізовано наявні офлайн та онлайн застосунки для організації зображень, знайдено їхні переваги та недоліки. Було розглянуто методи отримання інформації з зображень, яку можна використати для задачі пошуку за текстовим описом, та для подальших експериментів обрано три моделі сімейства CLIP, які навчались на парах «зображення – текст» та вміють розрізняти близькість пар цього роду. В ході досліджень на власному наборі даних було обрано модель Apple MobileCLIP-B (LT), що показала найкращий результат для задачі пошуку. Також було підібрано поріг міри косинусної подібності, що складає 0,094, за яким найкращим образом відбувається фільтрація нерелевантних пошукових результатів (метрика F1 складає 0,931).

Під час розробки застосунку було визначено варіанти використання користувачем, обрано технології з прицілом на локальний запуск продукту без під’єднання до мережі Інтернет. Після розробки було проведено тестування швидкодії та точності пошуку на іншій виборці даних, яке показало точність пошуку 0,96 за метрикою F1@5 та час пошуку менше 6 мілісекунд для обсягу зображень до 100 000, що відповідає вимогам до застосунків даного типу.

В якості подальшого розвитку застосунку доцільно вивчити питання можливості фільтрації за метаданими, в тому числі датою зйомки, геопозицією або розміром зображень, а також реалізувати фонову індексацію, що може покращити досвід користування програмою.

Результати роботи апробовано у вигляді 2 тез доповідей під час IV-ої Міжнародній науково-практичній конференції «Наука постіндустріального суспільства: глобалізація та трансформаційні процеси» [3] та XXIX Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті» [38].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, 33(1), pp. 113-125. DOI:10.11591/ijeecs.v33.i1.pp113-125.

2. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, IEEE Access, vol. 12, pp. 73376-73385. DOI: 10.1109/ACCESS.2024.3404371.

3. Yakovleva O., Matúšová S., Koshel V. Implementation of AI approaches in current tools for managing image collections to improve the search capabilities. *Proceedings of the IV Correspondence International Scientific and Practical Conference «Science in motion: classic and modern tools and methods in scientific investigations» in Periodical International scientific journal «Grail of science»*. (February 21, 2025). Vinnytsia, Ukraine – Vienna, Austria. 2025. Vol. 49. P. 752–755. DOI: 10.36074/grail-of-science.21.02.2025.096.

4. Пошук зі Spotlight на Mac. Apple Support. URL: https://support.apple.com/uk-ua/guide/mac-help/mchlp1008/mac (дата звернення 07.04.2025).

5. How to search for images in Dropbox. URL: https://help.dropbox.com/view-edit/image-search?fallback=true (дата звернення 07.04.2025).

6. Пошук фотографій і відео за розташуванням у програмі «Фотографії». URL: https://support.microsoft.com/uk-ua/windows/пошук-фотографій-і-відео-за-розташуванням-у-програмі-фотографії-27b3e790-05d6-4d86-8b69-7a0ffbf72d84 (дата звернення 07.04.2025).

7. Перегляд файлів і керування ними в Adobe Bridge. URL: https://helpx.adobe.com/ua/bridge/using/view-files-bridge.html (дата звернення 07.04.2025).

8. FastStone Image Viewer - Powerful and Intuitive Photo Viewer, Editor and Batch Converter. URL: https://www.faststone.org/FSViewerDetail.htm (дата звернення 07.04.2025).

9. XnView MP · Advanced Image Viewer & Photo Management Software | Free | XnView.com. URL: https://www.xnview.com/en/xnviewmp/ (дата звернення 07.04.2025).

10. Award-winning Digital Photo and Image Management Cataloging Software. Try Free! URL: https://www.picajet.com/en/index.php (дата звернення 07.04.2025).

11. digiKam – About. URL: https://www.digikam.org/about/ (дата звернення 07.04.2025).

12. Image recognition software, ML image analysis, and video analysis – Amazon Rekognition – AWS. URL: https://aws.amazon.com/rekognition/?nc1=h_ls (дата звернення 07.04.2025).

13. Використання функції «Візуальний пошук» для ідентифікації об’єктів на фотографіях і відео на iPhone — служба підтримки Apple (UA). URL: https://support.apple.com/uk-ua/guide/iphone/iph21c29a1cf/ios (дата звернення 07.04.2025).

14. How to find photos in a catalog in Lightroom Classic. URL: https://helpx.adobe.com/ua/lightroom-classic/help/finding-photos-catalog.html (дата звернення 07.04.2025).

15. Excire Foto. URL: https://excire.com/en/excire-foto (дата звернення 07.04.2025).

16. Як шукати людей, об’єкти й місця на фотографіях - Комп’ютер - Google Фото Довідка. URL: https://support.google.com/photos/answer/15235862 (дата звернення 07.04.2025).

17. Швидкий пошук фотографій за допомогою розумних пошукових запитів у OneDrive - Підтримка від Microsoft. URL: https://support.microsoft.com/uk-ua/office/швидкий-пошук-фотографій-за-допомогою-розумних-пошукових-запитів-у-onedrive-4b5e9300-38ea-4afa-8b42-a56e8b3c72d7 (дата звернення 07.04.2025).

18. Features | ACDSee Photo Studio Home. URL: https://www.acdsee.com/en/products/photo-studio-home/features/ (дата звернення 07.04.2025).

19. Lenso.ai - ШІ зворотній пошук зображень. URL: https://lenso.ai/uk (дата звернення 07.04.2025).

20. Gorokhovatskyi, O., & Yakovleva , O. (2024). Medoids as a packing of ORB image descriptors. Advanced Information Systems, 8(2), 5–11. DOI :10.20998/2522-9052.2024.2.01.

21. Yakovleva, O., & Nikolaieva, K. (2020). Research Of Descriptor Based Image Normalization And Comparative Analysis Of SURF, SIFT, BRISK, ORB, KAZE, AKAZE Descriptors. *Advanced Information Systems, 4*(4), 89-101. doi:10.20998/2522-9052.2020.4.13.

22. Яковлева, О. В., & Кускова, І. В. (2006). Дослідження результатів сегментації зображень методом матриць збігів. Вісник Національного технічного університету "ХПІ". №39 - С.164 -171.

23. Яковлева, О. В., & Панченко, І. А. (2007). Застосування енергетичних характеристик Лавса для сегментації зображень. Біоніка інтелекту : науково-технічний журнал. №2(67). - С.94-98.

24. Яковлева О.В., Нестерова О.П. (2009) (2009) Порівняльний аналіз методів характеристик Лавса і матриць збігів у задачах сегментації текстурних зображень. Прикладна радіо-електроніка: науч.-техн. журнал, Том 8, №2. - С.181 - 187.

25. Yakovleva O., Nebeský L, Liakhov P. (2023) Research methods of texture image analysis to solve the texture search problem. Proceedings of the IV International Scientific and Practical Conference. Vienna, Austria. pp. 252-261 URL: https://isg-konf.com/the-world-of-modern-technologies-and-inventions

26. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., & Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System. *CEUR Workshop Proceedings* Vol. 3403. pp. 69-86. ISSN 1613-0073 https://ceur-ws.org/Vol-3403/paper6.pdf

27. Yakovleva, O., Kovač, M., Ardasov, V. & Yeremenko, I. (2023). Study on adding functionality to the Zoom online conference system for monitoring the participant activities. Public Administration and Regional Development, 19(1), pp. 158–183.

28. Ковтуненко, А. Р., Яковлева, О. В., Любченко, В. А., & Янголенко, О. В. (2020). Дослідження сумісного використання математичної морфології та згорткових нейронних мереж для вирішення задачі розпізнавання цінників. Вісник Національного технічного університету ХПІ (3). 24-31.

29. Yakovleva O., Matúšová S., Tvoroshenko I., Isaiev Y. (2024). Visitor counting based on video stream analysis from surveillance cameras. Scientific Journal of Bratislava University of Economics and Management «Public Administration and Regional Development, Economics, Management and Marketing», vol. 20, no. 1, pp. 67–87. ISSN 1337-2955 URL: https://isg-konf.com/computerintegrated-technologies-of-automation-of-technological-processes

30. Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... & Sutskever, I. (2021, July). Learning transferable visual models from natural language supervision. In *International conference on machine learning* (pp. 8748-8763). PmLR.

31. Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., ... & Jitsev, J. (2023). Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2818-2829).

32. Jia, C., Yang, Y., Xia, Y., Chen, Y. T., Parekh, Z., Pham, H., ... & Duerig, T. (2021, July). Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning* (pp. 4904-4916). PMLR.

33. Xiao, B., Wu, H., Xu, W., Dai, X., Hu, H., Lu, Y., ... & Yuan, L. (2024). Florence-2: Advancing a unified representation for a variety of vision tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4818-4829).

34. Kim, W., Son, B., & Kim, I. (2021, July). Vilt: Vision-and-language transformer without convolution or region supervision. In *International conference on machine learning* (pp. 5583-5594). PMLR.

35. Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P. E., ... & Jégou, H. (2024). The faiss library. *arXiv preprint arXiv:2401.08281*.

36. World's most downloaded vector database: Elasticsearch | Elastic. URL: https://www.elastic.co/elasticsearch/vector-database (дата звернення 09.04.2025).

37. Qdrant Vector Database, High-Performance Vector Search Engine – Qdrant. URL: https://qdrant.tech/qdrant-vector-database/ (дата звернення 09.04.2025).

38. Кошель, В. О. (2025). Аналіз моделі OpenAI CLIP для задачі пошуку зображень на персональному комп'ютері за текстовим запитом. *Радіоелектроніка і молодь у XXI столітті: Тези доповідей 29-го Міжнародного молодіжного форуму* (Харків, 16–19 квітня 2025 р.) (Т. 7, с. 65–67). Харків: ХНУРЕ.

39. Ліцензування безкоштовних стокових фото і відео – Pexels. URL: https://www.pexels.com/uk-ua/license/ (дата звернення 27.04.2025).

40. Structured Outputs - OpenAI API. URL: https://platform.openai.com/docs/guides/structured-outputs (дата звернення 27.04.2025).

41. Mlfoundations. URL: https://github.com/mlfoundations (дата звернення 30.04.2025).

42. Large scale openCLIP: L/14, H/14 and g/14 trained on LAION-2B | LAION. URL: https://laion.ai/blog/large-openclip/ (дата звернення 30.04.2025).

43. Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., ... & Jitsev, J. (2022). Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, *35*, 25278-25294.

44. Gadre, S. Y., Ilharco, G., Fang, A., Hayase, J., Smyrnis, G., Nguyen, T., ... & Schmidt, L. (2023). Datacomp: In search of the next generation of multimodal datasets. *Advances in Neural Information Processing Systems*, *36*, 27092-27112.

45. open_clip/docs/openclip_results.csv at main mlfoundations/open_clip. URL:https://github.com/mlfoundations/open_clip/blob/main/docs/openclip_results.csv (дата звернення 30.04.2025).

46. Vasu, P. K. A., Pouransari, H., Faghri, F., Vemulapalli, R., & Tuzel, O. (2024). Mobileclip: Fast image-text models through multi-modal reinforced training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 15963-15974).

47. Google Перекладач. URL: https://translate.google.com (дата звернення 04.05.2025).

48. Elasticsearch Was Great, But Vector Databases Are the Future | by Zilliz | Medium. URL: https://medium.com/%40zilliz_learn/elasticsearch-was-great-but-vector-databases-are-the-future-0d7ec24ab7f9 (дата звернення 06.05.2025).

49. TIOBE Index – TIOBE. URL: https://www.tiobe.com/tiobe-index/ (дата звернення 07.05.2025).

50. ml-mobileclip/README.md at main · apple/ml-mobileclip. URL: https://github.com/apple/ml-mobileclip/blob/main/README.md#getting-started (дата звернення 07.05.2025).

51. Qt for Python. URL: https://doc.qt.io/qtforpython-6/ (дата звернення 07.05.2025).

52. tkinter — Python interface to Tcl/Tk — Python 3.13.3 documentation. URL: https://docs.python.org/3/library/tkinter.html (дата звернення 07.05.2025).

53. Build cross-platform desktop apps with JavaScript, HTML, and CSS | Electron. URL: https://www.electronjs.org/ (дата звернення 07.05.2025).

54. Home. URL: https://www.chromium.org/chromium-projects/ (дата звернення 07.05.2025).

55. Node.js — Запускайте JavaScript будь-де. URL: https://nodejs.org/uk (дата звернення 07.05.2025).

56. React. URL: https://uk.react.dev/ (дата звернення 07.05.2025).

57. TypeScript: JavaScript With Syntax For Types. URL: https://www.typescriptlang.org/ (дата звернення 07.05.2025).

58. FastAPI. URL: https://fastapi.tiangolo.com (дата звернення 07.05.2025).

59. encodeURI() - JavaScript | MDN. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/encodeURI (дата звернення 09.05.2025).