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

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.932

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
«___»_____2024 р.

Магістерська дисертація

на здобуття ступеня магістра

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

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Бібліотека для інтелектуального створення наборів даних зображень»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІП-32мп
Українець Дмитро Русланович _____

Керівник:
доцент кафедри ІПІ, к.т.н., доц.
Баклан Ігор Всеволодович _____

Рецензент:
доцент кафедри ІСТ, к.т.н., доц.
Голубєв Леонтій Петрович _____

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

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Едуард ЖАРІКОВ

«___»_____2024р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Українцю Дмитру Руслановичу

1) Тема дисертації «Бібліотека для інтелектуального створення наборів даних зображень», науковий керівник дисертації Баклан Ігор Всеволодович, к.т.н., доц., затверджені наказом по університету від «08» листопада 2024 р. № 5016-с

2) Термін подання студентом дисертації «16» грудня 2024 р.

3) Об’єкт дослідження – програмне забезпечення для інтелектуального створення наборів даних зображень.

4) Предмет дослідження – методи та засоби створення програмного забезпечення для інтелектуального створення наборів даних зображень.

5) Перелік завдань, які потрібно розробити – аналіз існуючих методів та проблем автоматизованого створення наборів даних зображень; розробка методів автоматизованого збору, генерації та аугментації зображень; удосконалення методів автоматизованого анотування та оцінки якості зображень; інтеграція

розроблених методів у єдиний конвеєр для створення наборів даних; експериментальна перевірка ефективності запропонованих методів.

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

7) Орієнтовний перелік публікацій – одна публікація

8) Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9) Дата видачі завдання «1» вересня 2024 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Огляд наявної літератури	01.09.2024	
2	Аналіз існуючих методів та реалізацій	10.09.2024	
3	Постановка та формалізація задачі	20.09.2024	
4	Огляд, аналіз та удосконалення методів та алгоритмів автоматизованого збору, генерації та аугментації зображень.	20.10.2024	
5	Проектування архітектури	25.10.2024	
6	Написання програмної реалізації	10.11.2024	
7	Виконання експериментальних досліджень	14.10.2024	
8	Оформлення пояснювальної записки	18.11.2024	
9	Подання дисертації на попередній захист	22.11.2024	
10	Подання дисертації на захист	16.12.2024	

Студент Дмитро УКРАЇНЕЦЬ

Науковий керівник Ігор БАКЛАН

РЕФЕРАТ

Розмір пояснювальної записки – 116 аркушів, містить 3 ілюстрації, 30 таблиць, 5 додатків, 36 посилань на джерела.

Актуальність теми. У роботі розглянуто проблему створення наборів даних зображень, показано основні особливості існуючих рішень описаних проблем, їх переваги та недоліки. Виявлено потребу в удосконаленні методів автоматизації створення наборів даних зображень, генерації синтетичних зображень та покращення якості зображень, а також в розробці програмного забезпечення для реалізації цих методів для широкого кола задач машинного навчання.

Мета дослідження. Основною метою є підвищення ефективності створення високоякісних наборів даних зображень, придатних для широкого кола задач машинного навчання.

Об’єкт дослідження: програмне забезпечення для інтелектуального створення наборів даних зображень.

Предмет дослідження: методи та засоби створення програмного забезпечення для інтелектуального створення наборів даних зображень.

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

- аналіз існуючих методів та проблем автоматизованого створення наборів даних зображень;
- розробка методів автоматизованого збору, генерації та аугментації зображень;
- удосконалення методів автоматизованого анотування та оцінки якості зображень;
- інтеграція розроблених методів у єдиний конвеєр для створення наборів даних;
- експериментальна перевірка ефективності запропонованих методів.

Наукова новизна результатів магістерської дисертації полягає в тому, що розроблено програмне рішення, яке на відміну від інших надає користувачеві можливості для створення наборів даних зображень для різних задач машинного навчання з можливістю повної автоматизації процесу. Результат досягнутий шляхом розробки методів для аугментації та створення синтетичних зображень з використанням генеративної моделі Stable Diffusion, методу покращення зображення за допомогою Stable Diffusion та методу фільтрації нерелевантних зображень за допомогою моделей автоматичного тегування та LLM ChatGPT-4o.

Практичне значення отриманих результатів полягає в тому, що завдяки автоматизації процесів збору, обробки та анотування даних значно скорочуються часові та людські ресурси, необхідні для створення наборів даних зображень. Універсальність системи забезпечується підтримкою різних задач машинного навчання, що робить продукт доступним для широкого кола користувачів. Інтеграція сучасних технологій, таких як генеративні моделі та великі мовні моделі, забезпечують створення більш різноманітних та якісних наборів даних. Економічна ефективність досягається через зменшення витрат на ручну обробку.

Зв'язок з науковими програмами, планами, темами. Робота виконувалась на кафедрі інформатики та програмної інженерії Національного технічного університету України "Київський політехнічний інститут імені Ігоря Сікорського".

Апробація. Наукові положення дисертації пройшли апробацію на VII Міжнародній науково-практичній конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології SoftTech-2024»

Публікації. Наукові положення дисертації опубліковані в:

1) Українець Д.Р. Бібліотека для інтелектуального створення наборів даних зображень // Інженерія програмного забезпечення і передові інформаційні технології (SoftTech-2024) : матеріали тез доповідей VII Всеукраїнської

наук.-практ. конф. молодих вчених та студентів (м. Київ, 19-22 листопада 2024). – К. : КПІ ім. Ігоря Сікорського, 2024.

Ключові слова: НАБІР ДАНИХ ЗОБРАЖЕНЬ, МАШИННЕ НАВЧАННЯ, STABLE DIFFUSION, ГЕНЕРАТИВНІ МОДЕЛІ, АНОТАЦІЯ, ОЦІНКА ЯКОСТІ ЗОБРАЖЕНЬ, СИНТЕТИЧНІ ДАНІ, PYTHON.

ABSTRACT

Explanatory note size – 116 pages, contains 3 illustrations, 30 tables, 5 applications, 36 references.

Topicality. The work examines the problem of creating image datasets, shows the main features of existing solutions to the described problems, their advantages and disadvantages. It has been concluded that there is a need to improve methods for automating the creation of image datasets, generating synthetic images and improving image quality, and to develop software to implement these methods for a wide range of machine learning tasks.

The aim of the study. The main goal is to increase the efficiency of creating high-quality image datasets suitable for a wide range of machine learning tasks.

The object of research: software for intelligent creation of image datasets.

The subject of research: software for intelligent creation of image datasets.

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

- analysis of existing methods and problems of automated creation of image datasets;
- development of methods for automated image collection, generation and augmentation;
- improvement of methods for automated annotation and image quality assessment;
- integration of the developed methods into a single pipeline for creating datasets;
- experimental verification of the effectiveness of the proposed methods.

The scientific novelty of the results of the master's dissertation is that a software solution has been developed that, unlike others, provides the user with the ability to create image datasets for various machine learning tasks with the possibility of full automation of the process. The result was achieved by developing methods for augmentation and creation of synthetic images using the Stable Diffusion generative model, a method for

image enhancement using Stable Diffusion, and a method for filtering irrelevant images using automatic tagging models and LLM ChatGPT-4o.

The practical value of the obtained results is that the automation of data collection, processing and annotation significantly reduces the time and human resources required to create image datasets. The versatility of the system is ensured by the support of various machine learning tasks, which makes the product available to a wide range of users. Integration of modern technologies, such as generative models and LLM, ensures the creation of more diverse and high-quality datasets. Cost-effectiveness is achieved by reducing the cost of manual processing.

Relationship with working with scientific programs, plans, topics. Work was performed at the Department of Informatics and Software Engineering of the National Technical University of Ukraine «Kyiv Polytechnic Institute. Igor Sikorsky».

Approbation. The scientific provisions of the dissertation were tested at the VII International Scientific and Practical Conference of Young Scientists and Students "Software Engineering and Advanced Information Technologies SoftTech-2024".

Publications. The scientific provisions of the dissertation were published in:

1) Ukrainets D.R. Library for Intelligent Image Dataset Creation // Software engineering and advanced information technologies (SoftTech-2024): materials of abstracts of the VII All-Ukrainian scientific and practical conference of young scientists and students (Kyiv, 19-22 November 2024) - K. : Igor Sikorsky Kyiv Polytechnic Institute, 2024.

Keywords: IMAGE DATASET, MACHINE LEARNING, STABLE DIFFUSION, GENERATIVE MODELS, ANNOTATION, IMAGE QUALITY ASSESSMENT, SYNTHETIC DATA, PYTHON.

Пояснювальна записка

до магістерської дисертації

на тему: **Бібліотека для інтелектуального створення наборів даних зображень.**

виконав студент групи ІП-32мп

Українець Дмитро Русланович

ЗМІСТ

ВСТУП 14
1 ОГЛЯД СУЧАСНИХ ДОСЛІДЖЕНЬ В ОБЛАСТІ ІНТЕЛЕКТУАЛЬНОГО СТВОРЕННЯ НАБОРІВ ДАНИХ ЗОБРАЖЕНЬ 17
1.1 Опис предметної області 17
1.2 Огляд літератури 19
1.2.1 Основні етапи розвитку наукової думки 19
1.2.2 Використання алгоритмів машинного навчання в інтелектуальному створенні наборів даних зображень 21
1.2.3 Відомі області застосування згенерованих наборів даних зображень 22
1.2.4 Методи оцінки якості зібраних та згенерованих зображень 24
1.2.4 Існуючі виклики та невирішені проблеми в сфері інтелектуального створення наборів даних зображень 25
1.3 Аналіз існуючих рішень 27
1.3.1 Parcel2d 28
1.3.2 LSUN 31
1.3.3 DatasetDM 34
1.4 Постановка задачі 36
Висновки по розділу 38
2 МЕТОДИ ТА ЗАСОБИ ПОБУДОВИ СИСТЕМИ СТВОРЕННЯ НАБОРУ ДАНИХ ЗОБРАЖЕНЬ 40
2.1 Структура системи 40
2.2 Методи та засоби збору зображень 41
2.3 Методи та засоби генерації зображень 42
2.4 Методи та засоби аугментації зображень 48
2.4.1 Геометричні перетворення зображень 48
2.4.2 Фотометричні перетворення зображень 49
2.4.3 Аугментація зображень за допомогою генеративних моделей 50

2.5 Методи та засоби автоматичного та напівавтоматичного анотування 51
2.5.1 Анотація для задач класифікації, детекції та сегментації..... 51
2.5.2 Анотація для задач генерації зображень..... 54
2.6 Методи та засоби оцінки та покращення якості зображень 55
Висновки по розділу 59
3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 61
3.1 Моделювання та аналіз програмного забезпечення 61
3.2 Архітектура програмного забезпечення 62
3.3 Конструювання програмного забезпечення 66
3.3.1 Опис створених методів та технічних рішень 66
3.3.2 Використані утиліти та бібліотеки 69
3.3.3 Аналіз системних вимог 71
Висновки по розділу 72
4 ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО РІШЕННЯ 74
4.1 Оцінка часової ефективності..... 74
4.2 Оцінка ефективності фільтрації нерелевантних зображень 76
4.3 Оцінка ефективності автоматичної генерації тегів та текстового опису природньою мовою 78
4.4 Оцінка ефективності автоматичного анотування 79
Висновки по розділу 79
5 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ 81
5.1 Опис ідеї проекту 81
5.2 Технологічний аудит ідеї проекту 83
5.3 Аналіз ринкових можливостей запуску стартап-проекту 86
5.4 Розроблення ринкової стратегії проекту 99
5.5 Розроблення маркетингової програми стартап-проекту 104
Висновки по розділу 109
ВИСНОВКИ 110

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 112
ДОДАТОК А 117
ДОДАТОК Б 118
ДОДАТОК В 119
ДОДАТОК Г 120
ДОДАТОК Д 158

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

GPU – відеокарта;

CNN – згорткова нейронна мережа;

VAE – варіаційний автоенкодер;

API – інтерфейс програмування додатків;

LLM – велика мовна модель;

LoRA – нейронна мережа низькорівневої адаптація;

Веб-скрапінг – збір даних з мережі Інтернет шляхом автоматизованого перегляду сторінок.

ВСТУП

Створення високоякісних наборів даних зображень є ключовою проблемою у розвитку систем комп'ютерного зору та машинного навчання. Із зростанням вимог до точності та надійності цих систем постає необхідність у великих, різноманітних та репрезентативних наборах даних зображень. Однак процес збору, анотування та забезпечення якості таких наборів даних традиційно є трудомістким і вимагає значних зусиль з боку людини.

Дана дисертація присвячена вирішенню проблеми автоматизованого та інтелектуального створення високоякісних наборів даних зображень за допомогою об'єднання передових методів машинного навчання, комп'ютерного зору та генеративних моделей. Робота спрямована на розробку комплексного програмного рішення, здатного самостійно збирати, генерувати, анотувати та оцінювати набори даних зображень для різноманітних застосувань.

Актуальність теми дослідження визначається зростаючим попитом на якісні набори даних для задач машинного навчання в різних галузях, включаючи комп'ютерний зір, медицину, безпеку та промисловість. Одночасно вартість ручного створення та забезпечення якості таких наборів є високою, що робить автоматизовані методи більш привабливими.

Новизна запропонованого підходу полягає у створенні методу інтелектуального створення наборів даних зображень, який поєднує різноманітні алгоритми та техніки для повної автоматизації збору, обробки та оцінки якості зображень. Зокрема, у роботі пропонується модифікація існуючих алгоритмів оцінки якості зображень та покращення їх якості, а також розробка інтегрованого конвеєра для безперервного перетворення даних.

Метою дисертаційного дослідження є підвищення ефективності створення високоякісних наборів даних зображень, придатних для широкого кола задач машинного навчання, шляхом розробки методів та програмного забезпечення, які дозволять скоротити часові витрати та покращити якість набору даних. Для досягнення поставленої мети передбачається вирішення таких завдань:

- аналіз існуючих методів та проблем автоматизованого створення наборів даних зображень;
- розробка методів автоматизованого збору, генерації та аугментації зображень;
- удосконалення методів автоматизованого анотування та оцінки якості зображень;
- інтеграція розроблених методів у єдиний конвеєр для створення наборів даних;
- експериментальна перевірка ефективності запропонованих методів.

Об'єктом дослідження є програмне забезпечення для інтелектуального створення наборів даних зображень, а предметом - методи та засоби створення програмного забезпечення для інтелектуального створення наборів даних зображень.

Дослідження ґрунтується на аналізі новітніх досягнень у галузі машинного навчання, комп'ютерного зору та обробки зображень. Зокрема, у роботі розглядаються методи глибокого та активного навчання для автоматизованого анотування зображень, генеративні моделі для синтетичного доповнення даних, а також підходи до оцінювання якості зображень. Результати дисертаційної роботи можуть знайти застосування в розробці інтелектуальних систем машинного навчання та комп'ютерного зору для різноманітних галузей, таких як медицина, безпека, промисловість та сільське господарство. Запропоноване програмне

рішення дозволить істотно зменшити витрати на створення якісних наборів даних зображень, які є ключовими для ефективного навчання моделей.

1 ОГЛЯД СУЧАСНИХ ДОСЛІДЖЕНЬ В ОБЛАСТІ ІНТЕЛЕКТУАЛЬНОГО СТВОРЕННЯ НАБОРІВ ДАНИХ ЗОБРАЖЕНЬ

1.1 Опис предметної області

Інтелектуальне створення наборів даних зображень передбачає використання передових методів для збору, оцінки якості, анотування, аугментації зображень таким чином, щоб підвищити їхню корисність для навчання моделей машинного навчання.

Основними аспектами інтелектуального створення наборів даних є:

- автоматизований збір даних:
  - вилучення зображень з Інтернету та збір зображень – інструменти та алгоритми для вилучення зображень з Інтернету на основі заданих критеріїв мають вирішальне значення. Це включає використання імовірнісних методів для збору веб-зображень [3] та специфічних методів для скрапінгу, сфокусованого на домені, вказаному користувачем, наприклад, на моді [4];
  - синтетична генерація даних – такі методи, як генеративні змагальні мережі (GAN) та інші методи синтетичної генерації даних, використовуються для створення зображень, які штучно розширюють набір даних [21]. Такі інструменти, як Stable Diffusion, використовуються для створення високоякісних синтетичних зображень для спеціалізованих завдань [17];
- анотування даних:
  - ручне та автоматизоване анотування – використання активного навчання дозволяє підвищити якість автоматичної анотації даних завдяки невеликій кількості створених вручну анотацій для конкретної задачі, це дозволяє досягти балансу між швидкістю та

точністю анотації зображень [10]. Автоматизована анотація зображень з використанням глибокого навчання та згорткових нейронних мереж (CNN) дозволить виконувати анотацію зібраних зображень взагалі без участі людини, проте ціною точності [24] [25];

  - контроль якості анотацій – забезпечення якості анотацій за допомогою таких методів, як порівняння результатів різноманітних моделей для анотації зображень [12] та, наприклад, застосування багатоекспертних фреймворків [23];

- аугментація даних – використання різноманітних методів аугментації даних для збільшення різноманітності набору даних, включаючи геометричні перетворення, доповнення колірного простору та генеративні нейронні мережі такі як GAN [5] чи Stable Diffusion [17]. Також можливе використання різних ігрових рушіїв, таких як unreal engine, для створення синтетичних зображень шляхом побудови відповідної сцени та можливості швидко змінювати такі параметри як освітлення, точка зору чи текстури зображуваних об’єктів [19]. Доповнення даних покращує надійність моделі та продуктивність у задачах класифікації зображень [6];
- управління та інтеграція наборів даних:
  - людино-машинна синергія – використання безперервного та ітеративного процесу, в якому людський вхід уточнює машинні дані, забезпечуючи розвиток та покращення набору даних з часом [8].
  - інтеграція конвеєрів – розробка конвеєра генеративних даних, який безперешкодно інтегрує різні етапи збору, доповнення та анотування даних [15];
- оцінка якості:
  - оцінка якості зображень – впровадження методів оцінки та забезпечення якості зображень у наборі даних. Це включає вирішення

таких проблем, як розмиття при русі та розфокусування [20];

– оцінка юзабіліті набору даних – використання інструментів для оцінки ефективності набору даних для навчання моделей машинного навчання, гарантуючи, що він відповідає необхідним стандартам для передбачуваного застосування [26];

1.2 Огляд літератури

1.2.1 Основні етапи розвитку наукової думки

Створення високоякісних наборів даних зображень є критично важливим для успішного навчання моделей машинного навчання. Однак це завдання пов'язане з численними викликами, від забезпечення репрезентативності та різноманітності даних до подолання проблем масштабованості та забезпечення високої якості зображень. Протягом останніх десятиліть науковці докладали значних зусиль для розробки автоматизованих та інтелектуальних підходів, здатних спростити та вдосконалити процес створення наборів даних зображень.

Еволюція наукової думки у цій галузі пройшла кілька ключових етапів, відображаючи постійний технологічний прогрес та вдосконалення методів. На ранніх етапах основна увага приділялася простим імовірнісним підходам до збору та інтеграції зображень з інтернету. Поступово з'явилися перші методи машинного навчання, такі як згорткові нейронні мережі, здатні автоматизувати процес анотування зображень.

Значним проривом став розвиток генеративних змагальних мереж (GAN) та інших потужних генеративних моделей, які дозволили створювати реалістичні синтетичні зображення та збагачувати існуючі набори даних.

Паралельно відбувався прогрес у сфері оцінки якості згенерованих зображень та наборів даних, з розробкою спеціалізованих метрик та удосконаленням

суб'єктивних методів оцінювання. Однак, незважаючи на досягнуті успіхи, ця галузь і досі стикається з низкою невирішених проблем, пов'язаних із забезпеченням високої якості, репрезентативності та масштабованості створюваних наборів даних.

Ключові етапи розвитку наукової думки в сфері інтелектуального створення наборів даних:

- до 2015 року:
  - використання простих імовірнісних моделей для автоматизованого збору зображень з Інтернету [3];
  - початкові спроби створення великих наборів даних зображень шляхом поєднання людської та автоматизованої праці [27];
  - застосування перших згорткових нейронних мереж (CNN) для автоматичного анотування зображень [13];
  - створення великомасштабних наборів даних, таких як LSUN, за допомогою методів машинного та активного навчання [7];
- 2015-2020:
  - поширення використання генеративних змагальних мереж (GAN) для генерації реалістичних синтетичних зображень [21];
  - розробка GAN для створення нових варіацій існуючих зображень з метою збагачення наборів даних [21];
  - впровадження мультиекспертних фреймворків для підвищення точності автоматичного анотування [23];
- 2020-2023:
  - розвиток методів активного навчання для підвищення ефективності анотування [10];
  - поява Stable Diffusion як потужної альтернативи GAN для генерації

високоякісних зображень та аугментації існуючих зображень [17];
  - дослідження застосування GAN та Stable Diffusion в спеціалізованих галузях, таких як медицина та сільське господарство [21];
  - вивчення проблем масштабування, інтеграції різнорідних джерел даних та забезпечення конфіденційності [8];
- 2023 - сьогодення:
  - поєднання дифузійних моделей з автоматичним анотуванням для створення анотованих наборів даних [18];
  - розробка синтетичних наборів даних для передтренування класифікаційних моделей [30];
  - дослідження адаптивних згорткових нейронних мереж як більш точного методу для анотування зображень [24].

1.2.2 Використання алгоритмів машинного навчання в інтелектуальному створенні наборів даних зображень

Алгоритми та моделі машинного навчання відіграють вирішальну роль у процесах генерації, доповнення та анотування зображень для створення високоякісних наборів даних. Для ефективного виконання цих завдань застосовуються різні методи та підходи.

Для генерації зображень широко використовуються генеративні змагальні мережі (GAN). GAN, як зазначають Lu та ін., можуть створювати реалістичні синтетичні зображення, протиставляючи одна одній дві нейронні мережі: генератор, який створює зображення, і дискримінатор, який оцінює їхню достовірність [21]. Такі методи, як Stable Diffusion, є більш сучасною альтернативою GAN і сприяють створенню більш високоякісних синтетичних зображень, необхідних для розширення наборів даних у спеціалізованих задачах [17].

Для доповнення зображень використовують різні методи, щоб збільшити різноманітність і надійність наборів даних, але з них методами машинного навчання є лише генеративні нейронні мережі, такі як GAN та Stable Diffusion. Lu та ін. в своїй роботі описують також GAN для створення нових варіацій існуючих зображень агрокультур [21]. Таким ж шляхом як GAN можна застосувати Stable Diffusion, проте застосування обох цих методів машинного навчання потребує дуже багато ресурсів, зокрема електроенергії та достатньо потужних відеокарт[16].

В анотуванні зображень важливу роль відіграють моделі глибокого навчання, зокрема, згорткові нейронні мережі (CNN). Автоматичну анотацію зображень можна значно покращити, використовуючи ШНМ у поєднанні з моделями глибокого навчання, як підкреслюють Murthy та ін. [13]. Методи активного навчання, які передбачають ітеративне навчання моделі та використання її прогнозів для вибору найбільш інформативних зразків для анотування людиною, також відіграють ключову роль в ефективному маркуванні наборів даних [9] [10]. Мультиекспертні фреймворки, які поєднують знання різних моделей для підвищення точності анотування, обговорюють Bahrololoum, і Nezamabadi-pour [23].

Крім того, моделі, які використовують імовірнісні підходи для збору веб-зображень, можуть автоматизувати збір і початкове маркування великомасштабних наборів даних. Yanai та Kobus описують методи збору зображень з Інтернету на основі імовірнісних моделей, які допомагають створювати великі та різноманітні набори даних [3].

1.2.3 Відомі області застосування згенерованих наборів даних зображень

Інтелектуально створені набори даних зображень знаходять широке застосування в різних галузях, кожна з яких отримує вигоду від унікальних переваг, що надаються синтетичними і аугментованими даними зображень, а також оцінкою

їх якості та створеними анотаціями. Основними сферами застосування є комп'ютерний зір, генеративні нейронні мережі, медицина, безпека та інші спеціалізовані галузі.

Комп'ютерний зір є одним з найважливіших споживачів наборів даних зображень. У цій галузі синтетичні зображення використовуються для навчання та перевірки моделей для виявлення об'єктів, класифікації та сегментації зображень. Наприклад, системи збору та пошуку зображень, пов'язаних з модою, використовують великі набори даних для вдосконалення візуальних пошукових систем і систем рекомендацій [4]. Побудова набору даних LSUN також підкреслює використання великомасштабних наборів зображень у розробці надійних моделей глибокого навчання для розуміння сцен та розпізнавання об'єктів [7].

Медицина широко використовує набори зображень для підвищення точності діагностики та медичних досліджень. Високоякісні набори зображень медичних ушкоджень створюються за допомогою активного навчання та інших методів, які допомагають у навчанні моделей для виявлення та класифікації різних медичних станів [9]. Крім того, синтетичні методи генерації даних, такі як GAN, використовуються для доповнення наборів медичних даних, тим самим підвищуючи продуктивність діагностичних алгоритмів при виявленні рідкісних захворювань і відхилень [21].

Програми безпеки отримують вигоду від синтетичних наборів даних в таких сферах, як спостереження і перевірка особи. Автоматизовані методи анотування і доповнення даних покращують навчання систем розпізнавання облич, відстеження об'єктів і алгоритмів виявлення аномалій. Це підвищує точність і надійність систем безпеки, розгорнутих у державному і приватному секторах [25].

Сільське господарство - ще одна сфера, де синтетичні набори даних зображень є цінними. Такі методи, як GAN для доповнення зображень,

допомагають у навчанні моделей для моніторингу посівів, виявлення хвороб і оцінки врожайності, що має вирішальне значення для точного землеробства [21].

Промислові додатки також використовують згенеровані набори даних зображень для контролю якості та виявлення дефектів у виробничих процесах. Синтетичні набори даних використовуються для навчання моделей, які перевіряють продукцію на виробничих лініях, забезпечуючи постійне дотримання стандартів якості [22].

1.2.4 Методи оцінки якості зібраних та згенерованих зображень

Оцінка якості та повноти зібраних та згенерованих наборів даних зображень має вирішальне значення для забезпечення їхньої ефективності для навчання моделей машинного навчання. Для оцінки цих аспектів були розроблені і застосовуються різні методи, що включають як кількісні, так і якісні показники.

Метрики якості зображень:

- оцінка якості зображення (IQA) – традиційні метрики, такі як пікове відношення сигнал/шум (PSNR), яке показує рівень спотвореності через стискання та індекс структурної подібності (SSIM), який показує подібність зображень, зазвичай використовуються для оцінки зібраних та згенерованих зображень [20];
- ідентифікація та кількісна оцінка розмитості – методи ідентифікації та кількісної оцінки розмитості від руху та розфокусування мають вирішальне значення для забезпечення чіткості згенерованих зображень. Наприклад, Dindin та ін. розробили методи виявлення та вимірювання розмитості на зображеннях, що є важливим для збереження високої якості зображень [14];
- початкова відстань Фреше (FID) – FID вимірює відстань між розподілами

ознак реального та згенерованого зображень за допомогою попередньо навченої початкової мережі. Це надійна метрика для оцінки реалістичності згенерованих зображень, де нижчі показники вказують на вищу якість [21];

- початкова оцінка (Inception Score, IS) – IS оцінює якість і різноманітність згенерованих зображень, використовуючи попередньо навчену початкову мережу для їх класифікації, а потім вимірює ентропію прогнозованого розподілу класів. Вищі показники свідчать про кращу якість та різноманітність [21];
- оцінювач якості зображення природності (NIQE) – NIQE оцінює природність і якість зображень без використання еталонних зображень. Він обчислює оцінку якості на основі статистичних характеристик, отриманих зі статистики природних сцен [20];
- середній бал оцінки (MOS) – MOS - це суб'єктивний показник, отриманий шляхом усереднення оцінок людей, які оцінюють якість зображень за заздалегідь визначеною шкалою. Він особливо корисний для визначення суб’єктивних аспектів якості зображення [20].

1.2.4 Існуючі виклики та невирішені проблеми в сфері інтелектуального створення наборів даних зображень

Інтелектуальне автоматизоване створення високоякісних наборів даних зображень пов'язане з низкою невирішених проблем і завдань. Ці проблеми охоплюють технічні, методологічні та практичні аспекти збору, генерації, аугментації та анотування наборів даних:

- підтримка високої якості зображень – забезпечення стабільно високої якості зображень у великих наборах даних залишається складним завданням. Хоча такі метрики, як SSIM, PSNR, FID та IS допомагають,

вони можуть не повністю охоплювати всі аспекти якості зображення, такі як наприклад артефакти, отримані під час аугментації або генерації [21];

- розмитість і артефакти – методи виявлення та пом'якшення розмиття від руху, розфокусованих ділянок та інших артефактів потребують подальшого вдосконалення, щоб забезпечити чіткість і зручність використання зображень [14];
- помилки автоматизованої анотації - незважаючи на досягнення в галузі глибокого та активного навчання, автоматизовані методи анотування все ще можуть спричиняти помилки, особливо в складних або неоднозначних випадках [11]. Це вимагає значного втручання людини для перевірки та виправлення анотацій, що може бути ресурсомістким [13] ;
- суб'єктивність та упередженість – анотації можуть бути суб'єктивними, що призводить до невідповідностей та упередженості в наборі даних. Багатоекспертні структури допомагають пом'якшити це, але не усувають основну проблему [23];
- охоплення різноманітних сценаріїв – забезпечення репрезентативності наборів даних для широкого розмаїття реальних сценаріїв є складним завданням. Це включає в себе охоплення різних умов освітлення, перспектив, варіацій об'єктів і рідкісних подій [3];
- обмеження синтетичних даних – хоча синтетичне генерування даних (наприклад, за допомогою GAN) може допомогти розширити набори даних, ці методи іноді можуть створювати нереалістичні або надмірно гомогенізовані зображення, які не повністю відображають мінливість реального світу [21], і при неправильному використанні можуть навіть погіршувати роботу навченої моделі;
- обробка великих обсягів – ефективна обробка великих обсягів даних зображень залишається логістичною проблемою. Сюди входять питання,

пов'язані зі зберіганням, обчисленнями і передачею даних [2]. Крім того, деякі з методів генерації, анотації та аугментації даних потребують значно більшу кількість ресурсів, ніж більшість користувачів можуть виділити;

– інтеграція різнорідних джерел даних – об'єднання даних з різних джерел, кожне з яких має власний формат і якість, у цілісний і високоякісний набір даних є складним завданням і часто вимагає тривалої попередньої обробки [8];
– конфіденційність даних – використання зображень, які містять особисту або конфіденційну інформацію, викликає значні занепокоєння щодо конфіденційності. Забезпечення дотримання правил захисту даних (наприклад, GDPR) є важливим, але складним завданням [20];
– суб'єктивний характер оцінки якості – багато методів оцінювання якості ґрунтуються на суб'єктивних судженнях, або на оцінках людей (MOS), або на неявних припущеннях при розробці метрик. Створення більш об'єктивних і надійних метрик є постійним напрямком досліджень [20].

Незважаючи на значні досягнення, сфера автоматизованого створення високоякісних наборів даних зображень стикається з численними невирішеними проблемами. Серед них – підвищення точності анотацій, досягнення різноманітності та репрезентативності, вирішення проблем масштабування, а також розробка більш досконалих метрик оцінювання якості окремих зображень та створеного датасету в цілому. Вирішення цих проблем має вирішальне значення для подальшого розвитку інтелектуальних систем створення наборів даних зображень, чому і буде присвячена дана робота.

1.3 Аналіз існуючих рішень

Існує багато підходів для автоматизованого створення наборів даних зображень, які зазвичай полягають в комбінації різних методів для збору, генерації, розмітці, аугментації та оцінці якості зображень, інколи пропускаючи деякі з

наведених етапів, інколи не автоматизуючи їх повністю, наприклад надавши роботу по розмітці людині, а інколи добавляючи інші проміжні етапи. Декілька підходів та їх особливостей можна буде розглянути на прикладі існуючих рішень, наведених далі.

1.3.1 Parcel2d

Особливості рішення, детально описано в [1]. Parcel2d представляє собою повністю автоматизований конвеєр для генерації синтетичних наборів даних зображень, призначених для задач виявлення та сегментації екземплярів об'єктів. Процес створення набору даних складається з чотирьох основних етапів.

Перший етап - це збір зображень, який здійснюється шляхом веб-скрапінгу зображень з популярних пошукових систем зображень, таких як Google, Bing, Yahoo та Baidu. Для об'єктів інтересу (наприклад посилки) використовується 9 різних пошукових запитів на англійській, німецькій та китайській мовах. У прикладі наведеному в статті було зібрано понад 21 000 зображень посилок. Для сторонніх об'єктів випадково обирається певна кількість категорій (у прикладі 100 категорій) з набору даних ShapeNetSem і використано їх як пошукові запити, так було зібрано понад 12 000 зображень сторонніх об'єктів.

На другому етапі здійснюється відбір зображень за певними критеріями. Спочатку застосовується попередня обробка: видаляються дуже маленькі зображення та зображення з неоднорідним фоном (на основі аналізу колірної різноманітності пікселів на краях зображення). Потім використовується модель Rembg на основі U2-Net для сегментації об'єктів на зображеннях та видалення зображень з низькою якістю сегментації. Авторами пропонуються три різні типи наборів даних, що відрізняються методами додаткового відбору зображень:

- Parcel2D Plain - відбір лише за формою масок (близьких до опуклих);
- Parcel2D CNN - відбір за допомогою натренованої згорткової нейронної

мережі Mask R-CNN на виявлення і відбір зображень з рівно одним об’єктом;

– Parcel2D Manual - ручний відбір зображень експертами;

На третьому етапі генеруються самі синтетичні зображення. Об'єкти інтересу та сторонніх об'єкти випадковим чином розміщуються та масштабуються на фонових зображеннях з набору SUN397. Для кожної конфігурації генерується чотири різні версії зображення, використовуючи різні методи змішування об'єктів з фоном: без змішування, гаусове розмиття, розмиття руху та змішування Пуассона. Це робиться для зменшення впливу локальних артефактів від вставлення об'єктів.

Для оцінки якості згенерованих наборів даних авторами було зібрано реальний набір даних Parcel2D Real із 96 валідаційними та 297 тестовими зображеннями посилок. Для збору даних використовувалась спеціальна камера з можливістю захоплення кольорових зображень та даних глибини. Анотації посилок на зображеннях були згенеровані автоматично за допомогою алгоритму сегментації площин та виявлення об'єктів.

У ході експериментів автори аналізують вплив різних методів відбору зображень на якість кінцевих наборів даних та здатність натренованих на них моделей узагальнюватися на реальні дані. Також проводиться аблаційне дослідження впливу використання методів змішування на якість синтетичних зображень.

Загалом, представлене рішення пропонує гнучкий та повністю автоматизований конвеєр для створення великих спеціалізованих наборів даних зображень, призначених для задач машинного навчання в комп'ютерному зорі. Однак воно має певні обмеження, пов'язані з реалістичністю згенерованих синтетичних даних та необхідністю потужних обчислювальних ресурсів. Крім того це рішення може бути використане лише для створення наборів даних зображень

для детекції об’єктів, для інших задач воно не пристосоване. На рисунку 1.1 наведено схему роботи цього рішення [1].

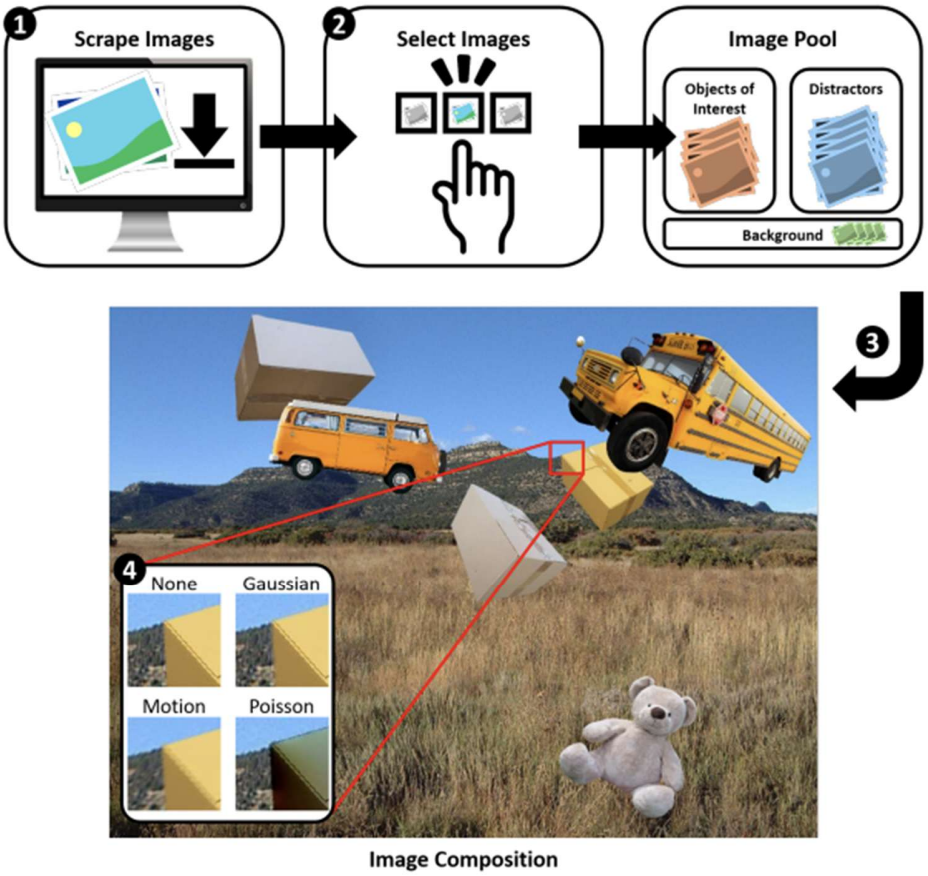

Рисунок 1.1 — Пайплайн створення набору даних Parcel2d

Переваги:

– повністю автоматизований конвеєр для створення наборів даних для детекції об’єктів без необхідності ручного збору та розмітки зображень;
– можливість генерації великих різноманітних наборів даних з високою ефективністю;
– використання методів змішування для зменшення артефактів від вставки зображень, що підвищує якість синтетичних даних;
– можливість порівняння різних методів відбору зображень та їх впливу на

якість навченої моделі;

- наявність реального набору даних для коректної оцінки якості синтетичних наборів.

Недоліки:

- якість синтетичних зображень обмежена, оскільки вони створюються шляхом простого вставлення об'єктів на фонове зображення, без урахування реалістичних ефектів освітлення, тіней тощо;
- можливість застосування цього рішення лише для задач детекції об’єктів;
- обмежена застосовність до специфічних об'єктів, для яких може бути складно знайти достатню кількість зображень за допомогою веб-скрапінгу.

1.3.2 LSUN

Рішення, описане в статті [7] спрямоване на створення великомасштабної бази даних зображень для покращення алгоритмів візуального розпізнавання. Основна ідея полягає в комбінуванні зусиль машин та людей для збору та розмітки мільйонів зображень. Це рішення складається з кількох ключових компонентів і працює в ітеративний спосіб:

- збір та первинна обробка зображень
  - збір мільйонів URL зображень за допомогою пошукової системи (Google Images) з використанням сценарних запитів та синонімів;
  - завантаження та обрізка зображень до мінімального розміру 256x256 пікселів;
- ініціалізація анотованого набору даних:
  - випадковий відбір невеликої частини зображень для кожної категорії сцен;

  - ручна подвійна анотація відібраних зображень через краудсорсинг (Amazon Mechanical Turk) з ретельним контролем якості;
  - створення тестового набору з повністю вручну анотованими зображеннями для оцінки якості;
- ітеративний цикл анотації з використанням глибокого навчання:
  - кластеризація зображень на основі витягнутих ознак з попередньо натренованої моделі;
  - навчання бінарного класифікатора (напр. SVM) на вручну анотованих зображеннях для розділення кластерів;
  - автоматична анотація впевнено класифікованих зображень за допомогою класифікатора;
  - передача складних невпевнено класифікованих зображень на наступну ітерацію;
  - на пізніших ітераціях - тонке налаштування (fine-tuning) моделі глибокого навчання на новоанотованих даних;
- контроль якості анотацій:
  - використання тестового набору для моніторингу якості автоматичних анотацій;
  - ретельний дизайн інтерфейсу для працівників AMT з прикладами, інструкціями та вбудованими контрольними запитаннями;
  - подвійне анотування кожного зображення та приймання лише узгоджених анотацій;
  - онлайн та прихована перевірка якості відповідей працівників AMT.

Цикл ітерацій продовжується допоки не буде досягнуто бажаної кількості якісно анотованих зображень для кожної категорії. На рисунку 1.2 наведено пайплайн створення набору даних LSUN [7].

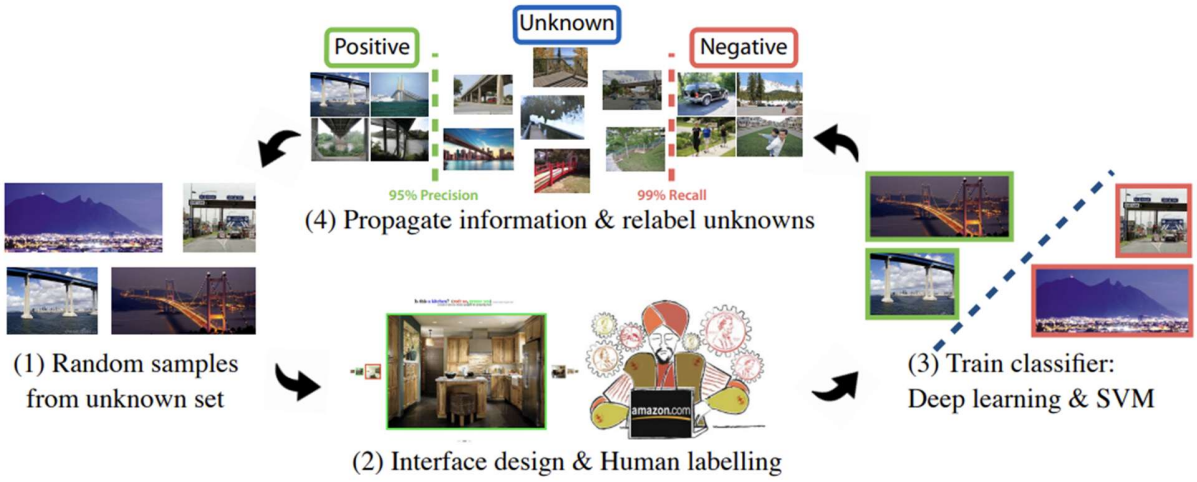

Рисунок 1.2 – Пайплайн створення набору даних LSUN

Переваги:

- забезпечує здатність швидко створити набагато більші анотовані набори даних, ніж за допомогою лише ручної анотації, завдяки автоматизації процесу;
- забезпечує високий рівень якості анотацій завдяки ретельному контролю якості анотацій від людей та автоматичному усуненню низькоякісних анотацій за допомогою моделей машинного навчання;
- реєструє ефективне використання людських зусиль, оскільки люди залучаються лише для анотації найскладніших прикладів, у той час як прості випадки обробляються автоматично;
- забезпечує адаптивність завдяки ітеративному тонкому налаштуванню моделей на нових даних, що дозволяє поступово підлаштовуватись під складні приклади;
- реєструє можливість використання потужних обчислювальних ресурсів для тренування моделей глибокого навчання.

Недоліки:

- реєструє складність у реалізації повного циклу, яка вимагає ретельного проектування, налаштування та інтеграції різних компонентів;
- реєструє потребу в потужних обчислювальних ресурсах, оскільки тренування моделей глибокого навчання є обчислювально витратним процесом;
- реєструє потенційні упередження у зібраних даних через використання пошукових систем як джерела зображень;
- реєструє необхідність ручної анотації для складних випадків та ініціалізації процесу, незважаючи на автоматизацію;
- реєструє витрати на оплату праці анотаторів, хоча і менші, ніж при повністю ручній анотації;
- обмежене використання, підходить лише для задач класифікації.

1.3.3 DatasetDM

У статті [18] представлено рішення під назвою DatasetDM, яке полягає в автоматизованому створенні наборів даних із семантичними анотаціями для різноманітних задач комп'ютерного зору. Ключовою ідеєю є використання попередньо натренованої дифузійної моделі для генерації синтетичних зображень, а також навчання універсального декодера сприйняття (P-Decoder) для виведення семантичних анотацій з латентних представлень дифузійної моделі. Для навчання декодера потрібно лише близько 1% наявних розмічених даних, що дозволяє генерувати необмежену кількість синтетичних анотованих зображень. Згенеровані дані можна потім використовувати для навчання різних моделей для задач сегментації, оцінки глибини, виявлення людських поз тощо. Особливістю методу є можливість керованої різноманітності генерованих зображень завдяки використанню великих мовних моделей, таких як GPT-4, для генерації різноманітних текстових підказок. Автори також демонструють, що синтетичні дані

з DatasetDM забезпечують значне поліпшення продуктивності для різноманітних завдань і збільшують стійкість до зсуву розподілів даних.

Переваги:

- забезпечує генерацію необмеженої кількості синтетичних зображень із високоякісними семантичними анотаціями;
- забезпечує значне поліпшення продуктивності для різних завдань комп'ютерного зору, таких як семантична сегментація, виявлення екземплярів, оцінка глибини та виявлення людських поз;
- забезпечує стійкість до зсуву розподілів даних та ефективне вирішення проблеми довгого хвоста;
- забезпечує керовану різноманітність генерованих зображень завдяки використанню великих мовних моделей;
- забезпечує гнучкість та можливість композиції нових завдань, таких як редагування зображень.

Недоліки:

- реєструє потребу невеликої кількості розмічених даних (близько 1% наявного набору даних) для навчання декодера сприйняття;
- реєструє можливість отруєння синтетичними даними, яке може відбутися при наявності в початковому наборі даних спеціальним чином модифікованих даних, які модифікують для перешкоджання навчання на них генеративних моделей;
- реєструє можливість виродження генеративних моделей, яке може відбуватися при ітеративному навчанні моделі генерації зображень на згенерованих нею ж зображеннях;
- реєструє обмеження якості генерованих зображень поточними

можливостями дифузійних моделей та великих мовних моделей.

1.4 Постановка задачі

Метою дисертаційного дослідження є підвищення ефективності створення високоякісних наборів даних зображень, придатних для широкого кола задач машинного навчання, шляхом розробки методів та програмного забезпечення, які дозволять скоротити часові витрати та покращити якість набору даних. Для досягнення поставленої мети передбачається вирішення таких завдань:

- аналіз існуючих методів та проблем автоматизованого створення наборів даних зображень;
- розробка методів автоматизованого збору, генерації та аугментації зображень;
- удосконалення методів автоматизованого анотування та оцінки якості зображень;
- інтеграція розроблених методів у єдиний конвеєр для створення наборів даних;
- експериментальна перевірка ефективності запропонованих методів.

Створений продукт повинен відповідати наступним вимогам:

- надавати можливість створювати набори даних зображень під різні задачі, такі як класифікація, виявлення об’єктів, генерація;
- збір зображень:
  - продукт повинен мати можливість здійснювати веб-скрапінг зображень з різних пошукових систем за заданими пошуковими запитами;
  - продукт повинен підтримувати використання різних мовних варіантів та синонімів для пошукових запитів;
  - продукт повинен здійснювати первинну обробку зібраних зображень,

включаючи видалення занадто малих, зображень, що не відповідають запиту, тощо;

- генерація синтетичних зображень:
  - продукт повинен мати можливість використовувати генеративні моделі, такі як Stable Diffusion, для створення синтетичних зображень;
  - продукт повинен забезпечувати керовану різноманітність синтетичних зображень шляхом використання великих мовних моделей;
- аугментація зображень:
  - продукт повинен надавати можливість здійснювати геометричні перетворення зображень;
  - продукт повинен підтримувати доповнення колірного простору зображень;
  - продукт повинен мати можливість використовувати генеративні моделі для аугментації зображень;
- анотування зображень:
  - продукт повинен забезпечувати автоматизоване анотування зображень за допомогою глибоких нейронних мереж;
  - продукт повинен надавати можливість ручного анотування для складних випадків;
- оцінка якості зображень:
  - продукт повинен підтримувати використання традиційних метрик якості зображень;
  - продукт повинен забезпечувати оцінку наявності та ступеня розмитості та артефактів на зображеннях;
- управління та інтеграція наборів даних:

  - продукт повинен інтегрувати різні етапи створення набору даних в єдиний конвеєр;
  - продукт повинен надавати можливість контролю якості та фільтрації/видалення низькоякісних зразків;
- технічні вимоги:
  - продукт повинен мати здатність обробляти великі обсяги даних зображень;
  - продукт повинен підтримувати використання потужних обчислювальних ресурсів (GPU) для задач глибокого навчання;
  - інтуїтивно зрозумілий інтерфейс;
  - простота у використанні;
  - простота у налагодженні.

Висновки по розділу

У першому розділі проведено ґрунтовний аналіз предметної області інтелектуального створення наборів даних зображень. Розглянуто основні аспекти цього процесу, такі як збір зображень, генерація синтетичних даних, аугментація, анотування та оцінка якості зображень. Проаналізовано сучасні дослідження та досягнення у застосуванні методів машинного навчання, зокрема генеративних моделей, глибоких нейронних мереж і активного навчання для вирішення завдань цієї сфери.

Наведено огляд відомих областей застосування згенерованих наборів даних зображень, включаючи комп'ютерний зір, медицину, безпеку та промисловість. Також висвітлено існуючі методи оцінки якості зібраних та згенерованих зображень, такі як традиційні метрики якості зображень, виявлення розмитості та артефактів, а також метрики реалістичності на кшталт FID та Inception Score.

Проаналізовано сучасні виклики та невирішені проблеми в галузі інтелектуального створення наборів даних зображень, зокрема проблеми забезпечення високої якості зображень, помилок автоматизованої анотації, охоплення різноманітних сценаріїв та обробки великих обсягів даних.

Розглянуто кілька існуючих рішень для автоматизованого створення наборів даних зображень, включаючи Parcel2d, LSUN та DatasetDM. Для кожного рішення наведено опис його особливостей, переваг та недоліків.

В результаті проведеного аналізу сформульовано постановку задачі дослідження, визначено мету, призначення та основні завдання розробки системи для ефективного та автоматизованого створення високоякісних наборів даних зображень для широкого кола задач машинного навчання.

2 МЕТОДИ ТА ЗАСОБИ ПОБУДОВИ СИСТЕМИ СТВОРЕННЯ НАБОРУ ДАНИХ ЗОБРАЖЕНЬ

2.1 Структура системи

Для реалізації функціональних вимог, описаних в розділі 1 потрібно реалізувати декілька компонентів системи, для частини з яких потрібно буде дослідити і обрати найбільш ефективні методи і/або за потреби покращити їх або розробити нові. Структура системи для інтелектуального створення набору даних зображень наступна:

- веб-додаток:
  - інтерфейс для конфігурування та запуску задач зі створення наборів даних;
  - візуалізація прогресу та результатів;
- конвеєр обробки даних:
  - модуль збору зображень (веб-скрапінг);
  - модуль генерації синтетичних зображень (інтеграція з Stable Diffusion);
  - модуль аугментації зображень;
  - модуль автоматичного анотування;
  - модуль оцінки якості зображень.

В цій роботі увагу приділено саме конвеєру обробки даних, який буде створюватися за допомогою бібліотеки, що була розроблена в процесі виконання цієї роботи, інші компоненти є важливими для користувача, але не несуть в собі ніякої наукової чи практичної новизни, тому описані не будуть. Модулі конвеєру даних можуть в залежності від потреб користувача використовуватися в довільному порядку, при умові що будуть отримувати на вхід зображення (не стосується модулю збору зображень, який буде основним постачальником цих зображень).

2.2 Методи та засоби збору зображень

Ефективний збір зображень є критично важливим етапом для забезпечення достатнього обсягу та якості вхідних даних для подальшого створення набору даних. В модулі збору зображень будуть використовуватися як веб-скрапінг, так і отримання зображень через API.

Одним з найпоширеніших способів збору зображень є веб-скрапінг – автоматизований процес вилучення даних зі веб-сторінок. Для ефективного веб-скрапінгу зображень рекомендується використовувати спеціалізовані фреймворки, такі як Scrapy або Selenium. Ці інструменти мають вбудовані можливості з паралельного завантаження контенту, обходу захисних механізмів (CAPTCHA, затримки, блокування IP) та оптимізації запитів. Основною проблемою вебскрапінгу є його повільна швидкість в порівнянні зі збором зображень через API, та правові проблеми, оскільки значна частина ресурсів в інтернеті не дозволяють збирати з них зображення автоматизованими засобами.

Для оптимізації швидкості та ефективності веб-скрапінгу слід застосовувати такі методи [2]:

- паралельний скрапінг декількох сайтів/сторінок одночасно через багатопоточність або багатопроцесорність;
- інкрементальний скрапінг для збору лише нових або оновлених зображень;
- використання проксі-серверів та ротація user-agent для обходу блокувань;
- використання хмарних сервісів або потужних віртуальних машин для розподіленого скрапінгу;
- виконання запитів до різних пошукових систем на декількох мовах для забезпечення максимальної різноманітності зображень.

Також при веб-скрапінгу зображень для деяких задач, наприклад для навчання моделі генерації зображень, може бути корисно збирати не тільки зображення, але й їх описи/теги [1], які присутні на деяких сайтах, наприклад в соціальних мережах, але через різноманітний можливий формат та розміщення цього опису на різних ресурсах це можна зробити лише для обмеженої кількості найбільших веб ресурсів, вказавши формат та розміщення для кожного з них окремо при розробці.

Багато веб-ресурсів надають доступ до своїх баз даних зображень через спеціалізовані API, що може бути більш ефективним та швидким способом ніж веб-скрапінг [2]. Прикладами є Flickr API, Shutterstock API, Unsplash API та інші, наприклад API різних соціальних мереж. Проте при використанні API слід враховувати ліцензійні обмеження, вартість та обмеження за квотами.

2.3 Методи та засоби генерації зображень

До винайдення дифузійних моделей для генерації зображень в 2021 існував лише один ефективний метод генерації зображень – GAN, генеративно-змагальна мережа, що складається з двох нейронних мереж: генератора та дискримінатора, які навчаються одночасно. Генератор намагається створити реалістичні зображення, тоді як дискримінатор намагається відрізнити синтетичні зображення від реальних. Процес навчання продовжується до тих пір, поки дискримінатор не зможе більше відрізняти згенеровані зображення від зображень з набору даних [21]. Але при роботі з GAN виникало багато проблем, зокрема нестабільність процесу навчання. GAN складаються з двох частин: генератора і дискримінатора, які навчаються одночасно у змагальному процесі. Це може призводити до кількох проблем [15]:

– проблема зникаючого або вибухаючого градієнта, що виникає через нестабільність під час навчання дискримінатора та генератора одночасно. Невідповідність у швидкості навчання може призвести до ситуації, коли

генератор не встигає за дискримінатором або навпаки, що ускладнює знаходження рівноваги і погіршує кінцевий результат;

– режим колапсу, коли генератор починає генерувати одноманітні або дуже схожі між собою зображення, не здатний створювати різноманітний вміст. Причиною може бути нестійкість під час тренування або недостатній обсяг та різноманітність навчальних даних.

Через важкість в навчанні GAN та неможливість створення універсальної моделі для генерації зображень різноманітних предметів через те, що при збільшенні кількості можливих концепцій модель ставала все більш нестабільною, на сьогодні моделі на основі прихованої дифузії витіснили GAN.

Принцип роботи моделей прихованої дифузії наведено на рисунку 2.1 [31]

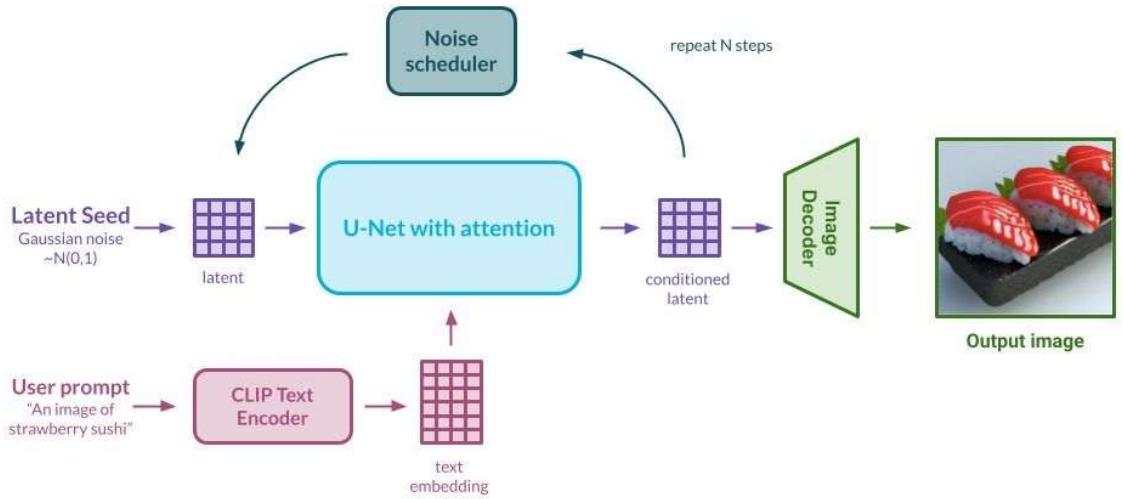

Рисунок 2.1 – Схема роботи моделей прихованої дифузії

На цьому рисунку бачимо, що спочатку текстова підказка за допомогою CLIP кодувальника перетворюється у текстове вбудування, після чого надходить на вхід моделі прихованої дифузії, що реалізована на основі U-Net, також крім текстового

вбудування на вхід моделі надходить випадково згенерований шум, який модель прихованої дифузії ітеративно усуває шум, намагаючись створити зображення, найбільш близьке до вхідного тексту (точніше не саме зображення, а його прихований простір, що містить основні ознаки майбутнього зображення). Після цього спеціальний декодер, що називається VAE (варіаційний автокодувальник) декодує прихований простір (наприклад розмірністю 4x64x64) у кінцеве зображення.

CLIP (Contrastive Language–Image Pre-training) – нейронна мережа, що вивчає зв'язок між текстовою семантикою та її візуальними представленнями [32]. CLIP навчається на сотнях мільйонів зображень і пов'язаних із ними підписах, вивчаючи, наскільки даний фрагмент тексту пов'язаний із зображенням. Тобто, замість того, щоб намагатися передбачити заголовок для зображення, CLIP замість цього просто дізнається, наскільки даний заголовок пов'язаний із зображенням. Дифузія складається з двох етапів, пряма дифузія та зворотна дифузія.

Пряма дифузія поступово перетворює навчальні зображення на шум. Зворотна дифузія ж навпаки, перетворює шум на характерне зображення, подібне до зображень з навчального набору даних [32].

Щоб виконати зворотню дифузію, нам потрібно знати, скільки шуму додається до зображення. Відповідь полягає в тому, щоб навчити модель нейронної мережі передбачати доданий шум . Він називається предиктором шуму в дифузії. Це модель U-Net. Таким чином працює звичайна дифузія, але через велику розмірність зображень, яка потребує значних обчислень, використовується прихована дифузія. Замість того, щоб працювати в просторі зображень великої розмірності, ця модель спочатку стискає зображення в прихований простір, наприклад у випадку Stable Diffusion зображення розмірністю 3x512x512 стискається до 4x64x64 за допомогою VAE, який виділяє основні ознаки.

Нейронна мережа Variation Autoencoder (VAE) складається з двох частин: кодера та декодера. Кодер стискає зображення до нижчого розмірного представлення в прихованому просторі. Декодер відновлює зображення з прихованого простору[32].

Існуючі моделі зазвичай відрізняються параметрами, їх кількість та навчальними даними, але можливі і деякі архітектурні відмінності, які проте важко дослідити через те, що більшість таких моделей мають закритий код. Порівняння найбільш популярних моделей наведено в таблиці 2.1.

Таблиця 2.1 – Порівняння моделей генерації зображень

Характеристика	Midjourney	Stable Diffusion 1.5	Stable Diffusion XL	DALL-E
Розробник	Midjourney	Stability AI	Stability AI	OpenAI
Рік випуску першої версії	2022	2022	2023	2021
Спосіб генерації	Text2image, image2image, inpainting, outpainting	Text2image, image2image, inpainting, outpainting, карти глибини, controlnet (виявлення країв, пози і багато інших можливих вхідних додаткових умов)	Text2image, image2image, inpainting, outpainting, карти глибини, controlnet (виявлення країв, пози і багато інших можливих вхідних додаткових умов)	Text2image, image2image, inpainting, outpainting

Продовження таблиці 2.1

Характеристика	Midjourney	Stable Diffusion 1.5	Stable Diffusion XL	DALL-E
Розмір зображень	до 1024x1024	Верхня межа обмежена лише об’ємом відеопамяті, оптимальні зображення отримуються при розмірах 512x512-512x968	Верхня межа обмежена лише об’ємом відеопамяті, оптимальні зображення при 1024x1024	до 1024x1024
Можливість донавчання	Ні	Так	Так	Ні
Вартість	$10-$120/місяць	Безкоштовно для особистого використання	Безкоштовно для особистого використання	$0.01-$0.08 за зображення
Системні вимоги	Не можна встановити на власний пристрій, доступ до моделі через Discord сервер	6 гб відеопамяті, 8 гб RAM. (можна використовувати і з меншою кількістю відеопамяті, але впливає на максимальний розмір зображення та час генерації)	12 гб відеопамяті, 16 гб RAM. (можна використовувати і з меншою кількістю відеопамяті, але впливає на максимальний розмір зображення та час генерації)	Не можна встановити на власний пристрій, доступ через веб інтерфейс або ChatGPT4

Midjourney і DALL-E мають дещо кращу якість вихідних зображень, зокрема вони краще розуміють запити і повертають ближчі до запиту зображення. Проте їхньою проблемою є обмеженість в тому, що вони можуть згенерувати, і хоча в Stable Diffusion є таке ж обмеження, її можна донавчити для конкретної задачі за допомогою LoRA [33], тому що в цієї моделі відкритий код. Наприклад якщо попросити DALL-E згенерувати якусь конкретну маловідому людину, то він поверне зображення випадкової людини, як і інші моделі, але якщо для Stable Diffusion навчити додаткову LoRA модель, передавши їй приблизно 30 фотографій потрібної людини (чи будь якої потрібної речі, концепту чи стилю), то після цього можна буде генерувати будь які зображення, які може генерувати Stable Diffusion, але вже з цією людиною. На тренування такої додаткової мережі потрібно лише декілька годин, в залежності від параметрів, кількості зображень та потужності відеокарти. Оскільки застосунок, що розробляється у цій роботі орієнтований на широке коло задач, тому в модулі генерації зображень було вирішено використовувати Stable Diffusion, версію 1.5 якщо користувач не потребує зображень великого розширення з складною композицією і XL у всіх інших випадках. Спочатку за допомогою модулю збору зображень будуть збиратися зображення з інтернету, після цього, якщо користувач обрав відповідну опцію, на частині з цих зображень буде навчено LoRA, щоб модель краще могла генерувати подібні зображення, після чого буде згенерована задана кількість зображень.

Також є підтип моделей Stable Diffusion, що називаються LCM (Latent Consistency Models), що дозволяють генерувати якісне зображення за значно меншу кількість ітерацій (3-4 в порівнянні з 15-20 у звичайних моделей), але зменшуючи при цьому варіативність можливих зображень. Ця версія також буде запропонована як одна з опцій на вибір користувачу.

2.4 Методи та засоби аугментації зображень

Аугментація даних - це техніка збільшення обсягу та різноманітності наборів даних шляхом застосування різних перетворень до наявних зразків даних. У контексті наборів даних зображень, аугментація зображень допомагає підвищити стійкість навчених моделей до різноманітних варіацій вхідних даних, таких як зміна масштабу, повороти, зсуви та інші геометричні й фотометричні перетворення. Це є особливо важливим для запобігання перенавчанню моделей та покращення їх здатності узагальнювати на нові, не представлені в навчальному наборі, дані. В модулі аугментації зображень будуть використовуватися такі типи аугментацій як геометричні, фотометричні та генеративні.

2.4.1 Геометричні перетворення зображень

Геометричні перетворення є одними з найпоширеніших методів аугментації зображень. Вони включають операції, які змінюють розмір, форму або орієнтацію зображення, не змінюючи безпосередньо його пікселі. До основних геометричних перетворень належать [5]:

- масштабування – зміна розмірів зображення за допомогою операцій підвибірки або інтерполяції. Це дозволяє імітувати різні відстані до об'єкта на зображенні;
- повороти – обертання зображення на певний кут, зазвичай у діапазоні від -180 до 180 градусів. Це допомагає моделі навчитися розпізнавати об'єкти в різних орієнтаціях;
- зсуви та обрізання – зсув зображення по горизонталі, вертикалі або по діагоналі, а також вирізання певних областей зображення. Такі перетворення імітують зміни точки зору або композиції сцени;
- віддзеркалення – горизонтальне або вертикальне відбиття зображення. Це є корисним для навчання моделей на симетричних об'єктах або сценах;

- перспективні перетворення – застосування перспективних спотворень до зображення, імітуючи ефекти перспективи та проекції;
- еластичні деформації – локальні деформації зображення за допомогою нелінійних геометричних перетворень, що імітують ефекти згинання або розтягування об'єктів.

Ці перетворення можна застосовувати як окремо, так і в комбінації одне з одним, забезпечуючи різноманітність аугментованих даних.

2.4.2 Фотометричні перетворення зображень

На додачу до геометричних перетворень, фотометричні перетворення дозволяють змінювати яскравість, контрастність, відтінки кольору та інші характеристики зображень, імітуючи різні умови освітлення, шуми та спотворення. Деякі з найбільш використовуваних фотометричних перетворень [6]:

- зміна яскравості – додавання або віднімання константи від значень пікселів зображення для імітації різних рівнів освітленості;
- зміна контрастності – збільшення або зменшення контрастності зображення, що імітує ефекти освітлення та тіней;
- додавання шуму – введення різних типів шуму, таких як гаусівський шум, шум "сіль і перець" або шум Пуассона, для імітації реалістичних спотворень та артефактів зображень;
- зміна гамма-корекції – застосування нелінійних перетворень яскравості зображення за допомогою гамма-корекції;
- зміна балансу білого – регулювання балансу білого для імітації різних температур кольорів і джерел освітлення;
- зміна насиченості кольору – збільшення або зменшення насиченості кольорів на зображенні;
- застосування фільтрів – використання різноманітних фільтрів, таких як

розмиття, згладжування, підвищення різкості тощо, для імітації ефектів оптичних систем.

Фотометричні перетворення можна застосовувати як окремо, так і в поєднанні з геометричними перетвореннями для отримання більш різноманітних аугментованих зображень. Бібліотеки, такі як OpenCV та albumentation для Python, надають зручні інструменти для реалізації геометричних та фотометричних перетворень і будуть використані при розробці модуля аугментації зображень.

2.4.3 Аугментація зображень за допомогою генеративних моделей

Поряд із традиційними методами аугментації зображень, такими як геометричні та фотометричні перетворення, останнім часом набув популярності підхід, що використовує генеративні моделі, зокрема дифузійні моделі такі як Stable Diffusion.

Окрім генерації абсолютно нових синтетичних зображень, що буде виконуватися в модулі генерації зображень, за допомогою Stable Diffusion можна виконувати img2img операцію, тобто передавати моделі крім текстового опису зображення ще й саме зображення, яке потрібно змінити та рівень зміни від 0 до 1, де при 0 в результаті отримується те ж саме зображення, при 1 – абсолютно інше. Цей метод дозволить створювати подібні на існуючі в наборі даних зображення, змінюючи стиль, кольори, текстури зображень, а при великому значенню рівня змін і змінювати композицію зображення [32]. Більш керованими інструментами є inpainting та outpainting, перший дозволяє застосовувати img2img до конкретної ділянки зображення, визначеної маскою, наприклад видаляти об’єкти з зображення, а другий – розширювати існуюче зображення, догенеруючи зображення навколо існуючого, враховуючи його контекст, що дозволить створювати з простого зображення якогось об’єкту зображення з складнішою композицією, що наприклад буде корисно в задачі детекції об’єктів.

В залежності від обраного типу набору даних, що створюється, для аугментації буде застосовуватися один або декілька з цих методів, а також створюватися маски для inpainting, наприклад якщо буде вказано, що набір даних буде складатися з зображень людей, то використовуючи моделі для детекції людей на існуючих зображеннях будуть знаходитись обмежувальні рамки людей, які і будуть масками для inpainting, дозволяючи змінювати фон зображення, залишаючи людину незмінною, або навпаки.

2.5 Методи та засоби автоматичного та напівавтоматичного анотування

Анотації відіграють таку ж важливу роль як і самі зображення у різноманітних завданнях машинного навчання, таких як класифікація, детекція об'єктів, генерація зображень та інших. Для створення анотації до кожного зображення вручну потрібно достатньо багато часу, особливо якщо набір даних буде використовуватися для навчання генеративних мереж, де потрібно максимально детально і при цьому стисло описати все, що є на зображенні, або для навчання моделей сегментації, де потрібно точно відмітити контури об’єктів. Тому в тих випадках де це можливо зручно використовувати різноманітні моделі для автоматичної анотації зображень, які будуть інтегровані в модулі анотації зображень моєї системи. Але різні задачі машинного навчання ставлять різні виклики перед автоматизацією процесу анотації.

2.5.1 Анотація для задач класифікації, детекції та сегментації

Для задач автоматичної класифікації, детекції та сегментації основною проблемою є те, що зазвичай набори даних для таких задач збираються з ціллю навчити модель розпізнавати те, що не можуть розпізнавати існуючі моделі, або щоб робити це краще за існуючі моделі. Тому участь людини в таких випадках необхідна. Проте застосування деяких існуючих підходів дозволить значно

пришвидшити процес анотації, зробивши її майже автоматичною, що і буде реалізовано в модулі анотації зображень.

Перш за все для анотації зображень потрібно буде навчити модель, і щоб це зробити з мінімальною участю людини потрібно використовувати комбінацію перенесення навчання (transfer learning), коли щоб навчити модель розпізнавати якийсь об’єкт донавчається модель, що вміє розпізнавати схожі об’єкти, наприклад YOLOv8 вміє детектити та класифікувати 1000 різних об’єктів, серед них наприклад котів, але не вміє розрізняти конкретні породи котів. Донавчивши цю модель на невеличкій кількості розмічених зображень котів різних порід, якої б не вистачило для навчання нової моделі, можна отримати модель достатньої високої точності для подальшої розмітки набору даних. Є декілька способів, які можна використовувати та комбінувати щоб отримати початковий набір даних для донавчання моделі анотації:

- використання слабких анотацій (таких як пошукові запити, теги, метадані та підписи зображень на сайтах), отриманих шляхом веб-скрапінгу разом з частиною зображень. Це підходить для задачі класифікації, проте не підходить для задачі сегментації та детекції, крім того зазвичай такі анотації не надто точні;
- використання вже існуючого набору даних меншого розміру, найкращий та найпростіший спосіб, проте можливий не у всіх ситуаціях;
- використання синтетичного набору даних, який створюється шляхом розміщення об’єкту, який потрібно детектити чи сегментувати на випадкових зображеннях традиційними методами або за допомогою генеративних мереж. Не підходить для класифікації;
- ручна розмітка частини зібраних зображень. Найнадійніший, але потребуючий участі людини метод.

Після донавчання моделі на створеному одним з методів набору даних використовується підхід активного навчання, який полягає в тому, щоб задіяти петлю між моделлю машинного навчання та людиною-анотатором. Спочатку донавчається модель на невеликому наборі анотованих даних. Потім ця модель використовується для оцінки інформативності неанотованих прикладів у пулі незанотованих даних. Найбільш інформативні приклади відбираються та передаються людині-анотатору для анотування. Ці нові анотовані приклади додаються до навчального набору, модель перенавчається, і цикл повторюється до досягнення бажаного рівня продуктивності. Критерії інформативності можуть бути різними, наприклад впевненість моделі в отриманому прогнозі або значна відмінність зображення від інших зображень в наборі даних.

Іншим підходом є застосування моделей сегментації по текстовому опису, наприклад CLIPSeg [34].

Модель CLIPSeg використовує попередньо навчену модель CLIP з додатковим трансформерним декодером. Ця модель має унікальну здатність генерувати бінарні карти сегментації на основі довільних текстових або візуальних запитів без необхідності перенавчання або донавчання моделі.

Архітектура CLIPSeg складається з замороженої CLIP моделі як магістралі, до якої додається мінімальний декодер, що дозволяє виконувати щільне передбачення. Модель може працювати з трьома основними типами завдань сегментації: сегментацією за описом, zero-shot сегментацією та one-shot сегментацією.

Її також можна використовувати і для автоматичної анотації, надаючи на вхід зображення та пошуковий запит, який описує те, що потрібно знайти на зображенні, а також для надійності опис фону, щоб чіткіше відділити шукане зображення. Далі

з отриманої бінарної карти сегментації обраховуються обмежувальні рамки як координати найбільш віддалених точок в кожному з чотирьох напрямів зображення.

Такий підхід дозволить повністю автоматизувати анотацію зображень, тому буде використаний в цій роботі.

2.5.2 Анотація для задач генерації зображень

Ціллю анотації при створенні набору даних для навчання моделей генерації зображень є створення максимально точного та одночасно стислого текстового опису зображення. Для цієї задачі застосовується два типи анотацій:

- тегування, де зображення описується як сукупність коротких тегів, таких як наприклад «1чоловік», «вигляд збоку», «ніч», «сигарета»;
- опис природньою мовою, наприклад те ж зображення можна описати як «чоловік вночі тримає сигарету та дивиться вбік».

Підхід тегування має проблеми з описом складних композицій, оскільки їх можна неоднозначно трактувати, проте він дозволяє простіше виконувати анотацію у випадку великої кількості об’єктів на зображенні і одночасно з тим в більшості випадків більш зрозумілий для моделей генерації зображення. Опис природньою мовою дозволяє більш детально описати взаємозв’язки між об’єктами, що будуть генеруватися на зображенні, проте складніший для інтерпретації моделями, особливо у випадку коли потрібно детально описати зображення. Найкращим рішенням буде комбінація цих двох підходів.

Хоч і ручна анотація зображень для генерації зображень складніша ніж для більшості інших задач, її автоматизація є простішою, достатньо застосувати існуючу модель для анотації, наприклад BLIP для створення анотацій природньою мовою та WD VIT Tagger для анотації у вигляді тегів. Ці моделі можуть описувати і ті об’єкти та концепції, які не можуть генерувати генеративні моделі, але у випадку

якщо створюється набір даних з рідкісним об’єктом, наприклад конкретною людиною, потрібно до створеної анотації автоматично додати тег, який буде позначати цей об’єкт, наприклад прізвище цієї людини. Цей тег користувач зможе задати до початку створення набору даних і змінити або видалити після перегляду результатів анотації, якщо в ньому не буде потреби.

2.6 Методи та засоби оцінки та покращення якості зображень

Одним із важливих етапів створення набору даних є оцінка якості зібраних зображень, видалення дублікатів та зображень, якість яких не вийде покращити, а також покращення зображень, що потребують цього. При цьому оцінка якості є суб’єктивною, оскільки для різних задач одне і те ж значення певної метрики зображення може як свідчити про те, що зображення буде корисним в наборі даних, так і навпаки. Наприклад для задач класифікації корисно коли зображення містять шуми та ускладнення різного походження, такі як погане освітлення чи розмиття, оскільки це дозволить моделі бути більш гнучкою. Але для задачі генерації зображень немає потреби в цих ускладненнях, навіть навпаки, якщо достатньо велика кількість зображень певного об’єкту будуть розмиті, то після навчання потрібно буде явно вказувати моделі що цей об’єкт повинен зображуватися нерозмитим, а такі вказівки вона буде розуміти не кожного разу, що ускладнить користування нею користувачеві.

Отже в модулі оцінки будуть встановлені різні порогові значення метрик якості зображення для різних методів машинного навчання.

Можуть бути використані такі метрики та методи оцінки якості зображень:

- метрики якості зображень[20]:
  - NIQE, який є сліпою метрикою оцінки якості зображень, тобто вона не потребує наявності еталонного зображення для порівняння. NIQE

оцінює якість шляхом вимірювання відхилення від статистичних характеристик природних зображень. Ця метрика базується на аналізі локальних ознак і вважається досить ефективною для визначення якості спотворених зображень. Нижчі значення NIQE вказують на кращу якість зображення, оскільки вони свідчать про більшу відповідність зображення природним;

– BRISQUE – ще одна сліпа метрикою оцінки якості, яка, як і NIQE, не потребує еталонного зображення. Вона аналізує характеристики зображення, зокрема його яскравість, контрастність та структуру текстур. BRISQUE використовує природні статистики для визначення спотворень, таких як розмитість, шум або блокінг артефактів. Нижчі значення BRISQUE вказують на вищу якість зображення, оскільки свідчать про меншу кількість спотворень;
– PIQE — це метрика, яка також не потребує еталонного зображення. Вона аналізує регіони зображення та визначає кількість спотворених ділянок, базуючись на кількості помітних артефактів, таких як шум, розмитість або спотворення кольорів. PIQE оцінює якість локально і вважається ефективною для виявлення проблем в окремих ділянках зображення. Вищі значення PIQE вказують на нижчу якість зображення, оскільки вони відображають більшу кількість спотворень;
– LIQE є відносно новою метрикою, яка також не потребує еталонного зображення. Ця метрика використовує логарифмічне перетворення для більш точного моделювання сприйняття людиною якості зображень. LIQE аналізує локальні особливості зображення і враховує різні типи спотворень, такі як шум або артефакти компресії. Як і інші метрики, нижчі значення LIQE вказують на краще

сприйняття якості зображення;

  - співвідношення площі заднього та переднього фону зображення, менше значення показує що зображення корисне для задач детекції, тоді як більше – що для задач генерації та класифікації;
- методи оцінки якості зображень:
  - градієнтний метод виявлення розмиття, що заснований на аналізі змін інтенсивності пікселів зображення шляхом обчислення та аналізу градієнтів [14];
  - виявлення шуму – для виявлення різних типів шуму (гауссівського, імпульсного, періодичного тощо) на зображеннях використовуються методи, засновані на аналізі статистичних властивостей зображення, таких як дисперсія, кореляційні функції або просторово-частотні характеристики;
  - метод виявлення блокових артефактів стиснення на основі аналізу статистики на границях блоків;
  - метод оцінки експозиції на основі аналізу гістограми яскравості.

Після оцінки якості зображень за описаними метриками та методами для зображень, які згідно цієї оцінки можна покращити, буде застосовано відповідний метод покращення:

- підвищення роздільної здатності:
  - генеративно-змагальна мережа для підвищення роздільної здатності (наприклад Real-ESRGAN);
- видалення шуму:
  - адаптивний медіанний фільтр для видалення імпульсного шуму;
  - фільтр Віннера для видалення гауссівського шуму;
- корекція експозиції та балансу білого:

  - алгоритм корекції експозиції за допомогою гістограмного вирівнювання;
  - метод корекції балансу білого за сірим світом;
- видалення артефактів стиснення:
  - фільтр для видалення блочних артефактів стиснення (Deblocking Filter);
  - алгоритм для усунення зубчастих контурів (Deringing Algorithm);
- відновлення зображень:
  - використання інпейнтингу моделей генерації зображень для заповнення відсутніх або пошкоджених/розмитих областей;
- покращення контрасту та деталізації:
  - адаптивне підсилення контрасту (Adaptive Contrast Enhancement).
  - нелінійне підсилення різкості (Unsharp Masking);
  - використання img2img методу моделей генерації зображень з низьким рівнем шуму для додавання деталей зображення без значної зміни його структури;
- зміна співвідношення площі заднього та переднього фону:
  - зменшення площі заднього фону шляхом сегментації зображення, та обрізання частин, що містять задній фон;
  - збільшення площі заднього фону шляхом використання методу outpainting моделей генерації зображень, що буде розширювати зображення шляхом догенерації його з усіх боків враховуючи контекст.

Також буде використовуватися алгоритм dHash для знаходження та видалення дублікатів зображень, який полягає в наступному [35]:

- зображення масштабується до фіксованого розміру (наприклад, 512x512

пікселів);

- обчислюється різниця сірого між суміжними пікселями по горизонталі та вертикалі;
- для кожної різниці формується бінарне значення: 1, якщо різниця більше порогового значення, і 0 – інакше;
- отриманий бінарний рядок інтерпретується як 64-бітне ціле число - це і є хеш-код зображення (dHash);
- знаходження відстані Хеммінга між хеш кодами зображень набору даних та видалення зображень, що мають відстань нижче межі (видаляється те зображення, в якого будуть гірші значення метрик якості).

Отже в модулі оцінки якості зображень спочатку будуть визначатися та дублікати та дуже схожі зображення, дублікати видалятися відразу. Далі для всіх зображень буде оцінюватися їх якість, для наборів подібних зображень буде залишатися зображення з найкращими показниками якості, також до зображень з показниками нижче певного порогу будуть застосовані відповідні методи покращення, після чого ще раз виконана оцінка якості, а зображення з значеннями нижче порогу видалення будуть видалятися відразу. Таким чином буде сформовано фінальний набір зображень.

Висновки по розділу

У даному розділі було розглянуто структуру системи інтелектуального створення наборів даних зображень, що складається з веб-додатку для взаємодії з користувачем, бекенд-сервісу для керування задачами, конвеєра обробки даних, системи зберігання даних та обчислювальної інфраструктури. Основну увагу було приділено опису методів і засобів для реалізації модулів конвеєра обробки даних.

Для модуля збору зображень розглянуто методи паралельного вебскрапінгу з використанням фреймворків Scrapy та Selenium, а також збір зображень через

різноманітні API. Описано способи підвищення ефективності та обходу захисних механізмів при вебскрапінгу.

Для модуля генерації зображень детально проаналізовано принципи роботи новітніх дифузійних моделей, зокрема Stable Diffusion, їх переваги над традиційними GAN. Наведено порівняння популярних моделей генерації за різними характеристиками. Розглянуто можливості адаптації Stable Diffusion до специфічних задач шляхом тренування додаткових LoRA моделей.

У модулі аугментації зображень описано методи геометричних та фотометричних перетворень, а також новітній підхід з використанням генеративних моделей, зокрема методи img2img, inpainting та outpainting Stable Diffusion.

Для автоматичного анотування зображень запропоновано використовувати комбінацію передачі навчання на невеликому наборі даних та активного навчання для задач детекції, класифікації та сегментації. Для генеративних задач можна застосовувати існуючі моделі BLIP та WD VIT Tagger з доповненням специфічних тегів.

Розглянуто різноманітні метрики та алгоритми оцінки якості зображень, методи виявлення дефектів та покращення якості з використанням традиційних підходів та генеративних моделей. Описано метод dHash для видалення дублікатів зображень.

Отже, у розділі комплексно проаналізовано сучасні підходи до створення наборів даних зображень з інтелектуальним застосуванням передових методів машинного навчання та обробки зображень.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Моделювання та аналіз програмного забезпечення

Для опису послідовності бізнес процесу взаємодії користувача з бібліотекою та взаємодії компонент бібліотеки між собою було побудовано діаграму послідовності наведену на рисунку А.1 в Додатку А.

На цій діаграмі зображено лише базовий сценарій роботи з бібліотекою, оскільки більшість компонент можна використовувати в довільному порядку або не використовувати взагалі.

Програмна реалізація бібліотеки буде складатися з реалізації описаних на діаграмі компонент, які будуть об’єднуватися користувачем у один конвеєр. Опис кожної з цих компонент:

– компонента збору зображень реалізує функціонал веб-скрапінгу, здійснюючи пошук та завантаження зображень за заданими критеріями. Ця компонента також виконує первинну фільтрацію зображень за розміром та релевантністю до заданої теми;
– компонента аугментації зображень відповідає за розширення набору даних шляхом створення модифікованих версій існуючих зображень. Вона дозволяє застосовувати різноманітні геометричні перетворення, такі як повороти, масштабування та відображення, а також модифікації колірного простору. У співпраці з генеративною моделлю ця компонента надає можливості створювати більш складні варіації зображень, що збагачує різноманітність набору даних;
– компонента генерації зображень, базована на генеративній моделі Stable Diffusion, виконує кілька важливих функцій. По-перше, вона створює повністю синтетичні зображення за заданими параметрами. По-друге, бере участь у процесі аугментації, створюючи нові варіації існуючих зображень. По-третє, допомагає покращувати якість проблемних

зображень, використовуючи режим генерації «зображення в зображення»;

- компонента анотації представляє собою систему, що підтримує два режими роботи. У режимі активного навчання вона постійно оновлює модель анотації на основі ручних анотацій користувача, що дозволяє покращувати якість автоматичної анотації з часом. У режимі автоматизованого анотування модуль використовує вже навчену модель для створення анотацій без втручання користувача. Цей підхід забезпечує баланс між якістю анотацій та ефективністю процесу. Ця компонента дозволяє генерувати анотації, мітки та описи зображень у вигляді природньої мови або тегів;
- компонента оцінки та покращення якості зображень виконує комплексний аналіз кожного зображення за різними параметрами. Вона дозволяє оцінити метрики якості, такі як рівень розмитість та наявність артефактів, а також перевіряє відповідність зображень їх цільовому призначенню. У разі виявлення проблем, модуль може застосовувати як традиційні методи обробки зображень, так і використовувати генеративні моделі для покращення якості. Після кожного етапу покращення проводиться повторна оцінка, щоб забезпечити відповідність встановленим критеріям якості.

3.2 Архітектура програмного забезпечення

За основу архітектури бібліотеки було взято компонентну архітектуру. Деталізація цієї архітектури зображена на рисунку Б.1 у Додатку Б.

Кожен компонент складається з таких класів:

- компонента збору зображень:
  - клас ImageScrapper:
    - searchImages(query: list, savePath: str, maxImage: int, searchEngines: list, worker_count: int) – пошук і завантаження

зображень;

- filterImages(imagesPath: str, minResolution: int, tags: list, filterSimilar: bool) – фільтрація зображень за розміром, тегами та схожістю;
- set_chrome_options() – налаштування опцій для браузера Chrome.
- create_target_folder(folder_name: str, target_path: str, search_term: str) – створення папки для збереження результатів;
- persist_image(folder_path: str, url: str) – збереження зображень з URL;
- save_img_urls(img_urls: list, target_folder: str, max_workers: int) – багатопотокове завантаження зображень;
- fetch_image_urls(query: str, max_links_to_fetch: int, wd: webdriver.Chrome, engine: str) – отримання URL зображень з пошукових систем;

- компонента аугментації зображень:
  - клас ImageProcessor:
    - set_min_resolution(resolution: str) – встановлення мінімальної роздільної здатності;
    - preprocess_images(images_path: str, preprocess_params: dict) – попередня обробка зображень;
    - augment_images(images_path: str, augmentations: list, hyperparameters: dict, num_augmentations: int) – аугментація зображень;
    - apply_geometric_transformations(images_path: str, transformations: list, degrees: int, num_transformations: int) – геометричні трансформації;

        - apply_color_space_changes(images_path: str, color_space_options: list, num_variations: int) – зміна колірного простору;
        - resize(images_path: str, scale: float, fixed_width: int, fixed_height: int) – зміна розміру зображень;
        - _read_yolo_bbox(bbox_path: str) – читання обмежувальних рамок у форматі YOLO;
        - _save_yolo_bbox(bbox_path: str, bboxes: list) – збереження рамок у форматі YOLO;
- компонента генерації зображень:
    - клас SyntheticImageGenerator:
        - delete_background(image_path: str, expand_padding: int) – видалення фону із зображення;
        - img2img(image_path: str, desc: str, prompts: list, output_dir: str, use_LLM: bool, lora: dict, lora_strength: float, denoising_strength: float, num_samples: int) – перетворення зображення з використанням моделі;
        - img2img_quality_fix(image_path: str, desc: str, prompts: list, output_dir: str, use_LLM: bool, lora: dict, lora_strength: float, denoising_strength: float, num_samples: int) – покращення якості зображень;
        - add_transparent_background_with_mask(image_path: str, padding: int) – додавання прозорого фону;
        - expand_white_mask(mask: Image, padding: int) – розширення білої маски;
        - resize_with_padding(image: Image, target_width: int, target_height: int) – зміна розміру зображення з додаванням полів;

        - inpaint_images_with_random_prompts(image_paths: list, output_dir: str, prompts: list) – генерація зображень із заливкою областей;
- компонента анотації:
    - клас ImageAnnotator:
        - autoCaption(imagesPath: str) – автоматичне створення підписів для зображень;
        - auto_bbox(imagesPath: list, class_description: list, save_bbox_image: bool) – генерація обмежувальних рамок для зображень;
        - pil_bbox_to_yolo(bbox: list, image_width: int, image_height: int, class_index: int, txt_path: str) – конвертація рамок PIL у формат YOLO;
        - auto_segmentation(imagesPath: str, segmentsDesc: list, modelName: str, save_mask: bool) – автоматична сегментація зображень;
        - merge_text_files(folder_path: str, split_char_count: int) – об'єднання текстових файлів із тегами;
- компонента оцінки та покращення якості:
    - клас QualityMetrics:
        - calculate_metrics(image_path: str) – обчислення метрик якості зображення;
        - auto_enhance_image(image_path: str, output_path: str) – автоматичне покращення якості зображення;
        - cut_background(image_path: str) – обрізання фону;
        - outpaint(image_path: str) – генерація зображення поза межами існуючого;

– sd_enchance_image(image_path: str, out_path: str) – покращення зображення за допомогою генеративної моделі.

Діаграма класів, яка описує класи, що відповідають описаним вище компонентам та зв’язки між ними наведена на рисунку В.1 у Додатку В.

3.3 Конструювання програмного забезпечення

3.3.1 Опис створених методів та технічних рішень

Окрім застосування вже існуючих методів та алгоритмів для різних етапів створення набору даних, які описані у розділі 2, було також розроблено декілька власних методів, що будуть описані в наступних підпунктах.

3.3.1.1 Аугментація даних та покращення якості зображень з використанням Stable Diffusion

В задачі аугментації даних Stable Diffusion використовується двома методами.

Перший метод полягає в застосуванні режиму генерації, де на вхід подається зображення з набору даних, для більшої гнучкості передається текстовий опис зображення у вигляді набору тегів або опису природньою мовою, після цього задаються такі параметри як denoising strength, який набуває значень від 0 до 1 і вказує наскільки повинно відрізнятися згенероване зображення від вхідного, де 0 – зображення ідентичне, 1 – зображення повністю інше, зазвичай краще всього залишати значення на рівні 0.6, але в залежності від складності зображення може бути змінений. Особливістю методу є також динамічна зміна опису зображення, який передається разом з зображенням, це реалізується двома методами на вибір, якщо передано теги, то використовується пошук подібних тегів по списку та їх заміна, наприклад заміна стилю зображення з modern на retro, що змінить освітлення і деякі інші деталі на зображенні у відповідності до нового тегу, зберігаючи композицію. Більш просунутим варіантом є використання LLM у вигляді ChatGPT, який буде змінювати опис зображення таким чином, щоб основні

об’єкти на зображенні збереглись, але з внесенням значних змін, які недоступні при використанні початкового опису зображення. Підказкою для отримання потрібного опису буде текст «Нижче наведено опис зображення, основною темою якого є *запит за яким користувач ініціалізував збір зображень* , зміни опис таким чином, щоб основна тема зображення та пов’язані з нею об’єкти не зазнали значних змін, при цьому щоб опис зображення в цілому значно відрізнявся від вхідного, це потрібно для аугментації набору даних, повертай тільки змінений опис, без додаткових пояснень. Ось вхідний опис зображення: *вхідний опис зображення у вигляді тегів або опису природньою мовою*». Символом * в підказці виділено значення відповідних змінних. Підказка може бути змінена користувачем на довільну, але важливо щоб вона містила текст «повертай тільки змінений опис, без додаткових пояснень» або текст подібного змісту.

Другий метод полягає в застосуванні режиму домальовування. На вхід подаються відібрані користувачем зображення, більшу частину яких займає зображення об’єкту, який є темою набору даних, наприклад конкретна людина. Наступним кроком є застосування моделі RMBG [36] для видалення фону. Після видалення фону створюється порожнє зображення заданого або випадкового розміру. Далі зображення з видаленим фоном масштабується до випадкового розміру та розміщується на випадковій ділянці порожнього (білого) зображення або на декількох ділянках, координати цього зображення зберігаються, щоб використовувати їх як обмежувальні рамки в датасеті для задач детекції. В наступному кроці отримане зображення передається в модель Stable Diffusion разом з маскою чорного кольору, яка буде визначена шляхом сегментації, та довільним запитом, який може бути переданий користувачем або створений на основі теми за допомогою LLM. Цей метод ефективний для аугментації даних для детекції об’єктів, оскільки містить чіткі обмежувальні рамки, що відкидає потребу в розмітці значної частини набору даних.

Для покращення якості зображення використовується режим генерації, де на вхід подається зображення з дефектом, та опис цього зображення у вигляді набору тегів, далі при потребі змінюється рівень denoising strength, після чого з тегів, що описують зображення вилучається тег або теги, що описують дефект зображення, якщо такі теги наявні, і ці теги додаються у негативну підказку, яка описує чого не має бути на згенерованому зображенні, а інші теги передаються у позитивну підказку, після чого виконується генерація зображення, на якому дефект буде менш виражений або відсутній, далі відбувається автоматичне тегування зображення обраною моделлю, і якщо серед тегів відсутні теги, описуючі дефекти, або якщо встановлена кількість повторень перевищена, зображення повертається як покращене, в іншому випадку всі кроки повторюються ще раз.

3.3.1.2 Метод фільтрування нерелевантних зображень

Для відсіювання нерелевантних зображень, що можуть бути зібрані з пошукових систем, і містять не зовсім те, що описано користувачем використовується метод, першим кроком якого є генерація тегів обраною користувачем моделлю, наприклад WD14 або DeepDanbooru. Після цього LLM ChatGPT проаналізує створені набори тегів та запит, який користувач передав для створення набору даних і поверне вказану кількість тегів, наявність яких в описі зображення свідчить про те, що зображення не підходить під запит користувача. Такі зображення будуть переміщені в окрему директорію, де користувач при бажанні зможе їх переглянути. Ось підказка, яка буде використана для аналізу тегів: «Тобі надато опис набору зображень у вигляді тегів:

теги до кожного зображення, розділені знаком нового рядка

Визначте *кількість тегів* ключових тегів (які з'являються в тексті вище і не перетинаються за значенням), які чітко вказують на те, що зображення не відповідає опису, і які точно не з'являться в зображеннях, що відповідають опису:

* запит за яким користувач ініціалізував збір зображень *. Зверни увагу, що не

всі зображення можуть бути позначені коректно, оскільки тегування виконувалося нейронною мережею. Виведи теги через кому в один рядок без пояснень».

3.3.2 Використані утиліти та бібліотеки

Опис утиліт та бібліотек які були використані для розробки наведено в таблиці 3.1.

Таблиця 3.1 – Опис утиліт

№ п/п	Назва утиліти/бібліотеки	Опис застосування
1	Visual Studio Code	Головне середовище розробки програмного забезпечення дипломної роботи.
2	PyTorch	Python фреймворк машинного навчання, що використовується для конструювання та навчання нейронних мереж
3	x_transformers	Python бібліотека для використання моделей на основі Transformer
4	pandas	Бібліотека для обробки та аналізу структурованих даних, використовується для роботи з файлами та структурування даних
5	selenium	Фреймворк для автоматизації веб-браузера, зокрема збору зображень та взаємодії з LLM
6	undetected_chromedriver	Спеціальна версія ChromeDriver для обходу виявлення автоматизації, використовується для збору зображень та взаємодії з LLM

Продовження таблиці 3.1

№ п/п	Назва утиліти/бібліотеки	Опис застосування
7	fake_useragent	Pyton бібліотека для генерації випадкових User-Agent заголовків, використовується для маскування автоматизованих запитів
8	cv2 (OpenCV)	Бібліотека комп'ютерного зору, використовується для обробки зображень та розрахунку метрик якості
9	pyiqa	Python бібліотека для оцінки якості зображень
10	torch	Фреймворк глибокого навчання PyTorch, використовується для роботи з моделями дифузії та нейронних мереж
11	PIL	Бібліотека для роботи з зображеннями, використовується для маніпуляцій з зображеннями в процесі аугментації
12	diffusers	Бібліотека для роботи з моделями дифузії, зокрема Stable Diffusion
13	accelerate	Бібліотека для оптимізації та прискорення навчання моделей на різних обчислювальних пристроях
14	peft	Бібліотека для ефективного тонкого налаштування великих моделей, включає підтримку LoRA та інших методів
15	sd-scripts	Набір скриптів для ефективного навчання LoRA
16	gradio	Python бібліотека для швидкого створення інтерфейсів

3.3.3 Аналіз системних вимог

Однією з найбільш ресурсомістких частин програми є навчання LoRA моделей, яке потребує відеокарту з об’ємом відеопамяті 8 гігабайт або більше, генерація зображень, для якої достатньо 4 гігабайт відеопамяті, а також використання нейронних мереж для автоматизованого анотування зображень і оцінки їх якості. Хоча ці операції можуть бути виконані на сучасних процесорах, продуктивність без спеціалізованого графічного прискорювача значно знижується, що впливає на час обробки великих наборів даних. Хоча система здатна працювати на процесорі, рекомендоване обладнання включає графічні прискорювачі з підтримкою CUDA-ядер, що дозволяє прискорити обчислення за допомогою відеокарт Nvidia.

Відсутність таких відеокарт призведе до значного збільшення часу виконання задач анотації, генерації зображень та обробки даних. Тому, мінімальні системні вимоги програми повинні відповідати стандартам сучасних операційних систем, але для досягнення максимальної продуктивності рекомендується використовувати обладнання з графічними прискорювачами, які підтримують CUDA.

В таблиці 3.2 наведено мінімальні системні вимоги, а в таблиці 3.3 – рекомендовані.

Таблиця 3.2 – Мінімальні системні вимоги

№ п/п	Назва вимоги	Мінімальна конфігурація
1	Тип процесору	Intel Core i3
2	Об‘єм ОЗП	8 гб

Таблиця 3.3 – Рекомендовані системні вимоги

№ п/п	Назва вимоги	Рекомендована конфігурація
1	Тип процесору	Intel Core i7
2	Об‘єм ОЗП	16 гб
3	Графічний прискорювач	RTX 2060 Super

Висновки по розділу

На початку третього розділу було проведено моделювання та аналіз програмного забезпечення, де за допомогою діаграми послідовності було описано базовий сценарій взаємодії користувача з бібліотекою та взаємодію компонент бібліотеки між собою. Було детально описано призначення та функціонал кожної з компонент системи: компоненти збору зображень, компоненти аугментації, компоненти генерації зображень на базі Stable Diffusion, компоненти анотації та компоненти оцінки й покращення якості зображень.

Далі було розроблено та представлено архітектуру програмного забезпечення, в основу якої покладено компонентну архітектуру. Було створено детальну діаграму класів, яка відображає структуру кожного компонента та взаємозв'язки між ними. Для кожного компонента було описано основні класи та їх методи, що забезпечують необхідну функціональність системи.

В підрозділі конструювання програмного забезпечення було представлено три оригінальні методи:

- метод аугментації даних та покращення якості зображень з використанням Stable Diffusion, який включає два підходи: генерацію на основі вхідного зображення з динамічною зміною опису та метод домальовування для створення даних для задач детекції;

- метод оцінки придатності зображення для задач класифікації та детекції об’єктів, що базується на комплексному аналізі характеристик зображення через сегментацію та оцінку локальних і глобальних параметрів;
- метод фільтрування нерелевантних зображень, який використовує комбінацію автоматичного тегування та аналізу через LLM для відсіювання невідповідних зображень.

Було проведено аналіз та документування використаних утиліт та бібліотек, де описано 16 основних інструментів, що забезпечують функціонування системи. Також було визначено та описано мінімальні та рекомендовані системні вимоги, з особливим акцентом на необхідність графічного прискорювача з підтримкою CUDA для оптимальної роботи системи.

4 ОЦІНКА ЕФЕКТИВНОСТІ РОЗРОБЛЕНОГО РІШЕННЯ

4.1 Оцінка часової ефективності

Для оцінки часової ефективності було виконано заміри тривалості кожного з етапів створення набору даних запропонованим рішенням та аналогами, якщо такий етап був наявний в аналогів. Кожен замір виконувався 10 разів, після чого обчислювалося середнє значення. Параметри що впливають на час виконання етапу вказані в таблиці після назви етапу в дужках. Результати замірів заокруглено до цілих значень секунд та наведено в таблиці 4.1. У випадку якщо етап відсутній в аналога, замість значення введено символ дефісу.

Таблиця 4.1 – Середня тривалість виконання етапів запропонованого рішення в секундах

Етап	Тривалість етапу у запропонованому рішенні	Тривалість етапу Parcel2d	Тривалість етапу DatasetDM
Збір зображень (пошукова система Google, 100 зображень)	126 с	164 с	-
Збір зображень (пошукова система Bing, 100 зображень)	18 с	59 с	-
Збір зображень (пошукова система Baidu, 100 зображень)	40 с	81 с	-

Продовження таблиці 4.1

Етап	Тривалість етапу у запропонованому рішенні	Тривалість етапу Parcel2d	Тривалість етапу DatasetDM
Фільтрація нерелевантних зображень (100 зображень, з використанням LLM)	93 с	-	-
Фільтрація нерелевантних зображень (100 зображень, без використанням LLM)	5 с	9 с	-
Генерація тегів (100 зображень)	77 с	-	-
Генерація опису природньою мовою (100 зображень)	982 с	-	-
Генерація обмежувальних рамок (100 зображень)	98 с	24 с	-
Аугментація стандартними методами (обертання, обтинання, зсув, 100 зображень)	11 с	11 с	-
Генерація синтетичних зображень (1 зображення, розмір 512x512 пікселів)	6 с	0.5 с	7 с
Оцінка та покращення якості зображення (1 ітерація покращення, 1 зображення)	10 с	-	-

Згідно результатів експериментів, наведених в таблиці 4.1, на етапах збору зображень запропоноване рішення є ефективнішим ніж аналог Parcel2d в 1.5-3 рази в залежності від пошукової системи, це досягається шляхом паралельного завантаження зображень. Етап фільтрації зображень без застосування LLM в запропонованого рішення швидший в 1.8 разів ніж в Parcel2d через відсутність застосування сегментації на цьому етапі. Етап генерації обмежувальних рамок в запропонованого рішення займає більше часу, оскільки використовується більш універсальна модель CLIP, тоді як в Parcel2d використовується спеціально навчена для одного типу об’єктів CNN, яка має значно менше параметрів ніж CLIP. Етап аугментації стандартними методами приблизно однаковий по часовій ефективності через використання вже готових реалізацій цих методів. Генерація синтетичних зображень в Parcel2d набагато швидша ніж в запропонованого рішення та DatasetDM, оскільки не використовує дифузійні моделі як два інших рішення, а просто накладає зображення на інші зображення, тому якість таких синтетичних зображень нижча.

Отже більшість етапів в розробленому рішенні мають кращу часову ефективність ніж в аналогах, а ті етапи, що мають меншу часову ефективність мають цьому обґрунтування.

4.2 Оцінка ефективності фільтрації нерелевантних зображень

Для оцінки ефективності фільтрації нерелевантних зображень було проведено наступний експеримент:

- зібрано 100 зображень за пошуковим запитом «Червона панда фото» модулем збору зображень;
- вручну визначено кількість зображень які не відповідають цьому запиту, наприклад намальованих картинок, а не фотографій, таких виявилось 23;
- виконано автоматичне тегування зібраних зображень;

– застосовано функцію для визначення 10 тегів нерелевантних зображень з LLM моделлю ChatGPT-4o mini та ChatGPT-4o;
– зображення що містять нерелевантні теги було відфільтровано та переміщено в окремі папки для кожної моделі LLM, обчислено матриці невідповідностей, наведені в таблицях 4.2-4.3.

Таблиця 4.2 – Матриця невідповідності фільтрування нерелевантних зображень ChatGPT-4o mini

Результат	Релевантні зображення	Нерелевантні зображення
Відфільтровані правильно	63	10
Відфільтровані неправильно	9	14

Таблиця 4.3 – Матриця невідповідності фільтрування нерелевантних зображень ChatGPT-4o

Результат	Релевантні зображення	Нерелевантні зображення
Відфільтровані правильно	73	0
Відфільтровані неправильно	4	19

По цих матрицях розрахуємо точність кожного з методів за цією формулою:

$$ACC = \frac{(TP + TN)}{(P + N)}, \quad (4.1)$$

де TP – релевантні зображення відфільтровані правильно, TN – нерелевантні зображення відфільтровані правильно, P та N – кількість релевантних та нерелевантних зображень відповідно.

Отже з використанням ChatGPT-4o mini отримана точність склала 80,2%, тоді як для ChatGPT-4o – 95,8%, що свідчить про достатню ефективність запропонованого методу.

4.3 Оцінка ефективності автоматичної генерації тегів та текстового опису природньою мовою

Для оцінки ефективності моделей автоматичного тегування та створення текстового опису природньою мовою було зібрано 100 випадкових зображень, і до кожного з них згенеровано теги та текстовий опис кожною з моделей з порогом впевненості 0.35, далі було переглянуто всі теги та текстові описи, за кожну помилку в тегах або описі віднімався 1 бал, максимум до 3 балів з одного зображення, після чого було обраховано відсоток набраних балів від максимальних 300 балів, це значення було записано в таблицю 4.4 як показник точності моделі.

Таблиця 4.4 Точність генерації тегів або текстового опису природньою мовою різних моделей

Модель	Точність
WD14 ViT v3	81.33%
WD14 ConvNeXTV2 v1	78.33%
mld-caformer.dec-5-97527	74.67%
WD14 moat tagger v2	78%
BLIP	92.66%

Отже найточнішою моделлю для тегування виявилась модель WD14 ViT v3, тоді як точність моделі для текстового опису BLIP сягає 92.66%, що достатньо для задач навчання моделей генерації зображень.

4.4 Оцінка ефективності автоматичного анотування

Для оцінки автоматичного анотування (визначення обмежувальних рамок об’єктів на зображенні), було обрано 1000 вручну анотованих зображень з однією обмежувальною рамкою на зображення. Далі для цих зображень було згенеровано обмежувальні рамки за допомогою розробленого рішення, після чого для кожного зображення було обраховано співвідношення площі перетину до площі об’єднання створених вручну обмежувальних рамок та згенерованих бібліотекою. Було отримано середнє значення цього співвідношення для всіх зображень, яке склало 91.65% при переведенні у відсотки, що є достатньо високим значенням для практичного застосування цього рішення.

Висновки по розділу

У даному розділі було проведено оцінку часової ефективності, точності фільтрації нерелевантних зображень, автоматичної генерації тегів і текстових описів, а також автоматичного анотування зображень. Результати дослідження підтвердили високу ефективність запропонованого рішення в порівнянні з аналогами. Завдяки оптимізації алгоритмів і використанню паралельного завантаження зображень вдалося суттєво зменшити час виконання більшості етапів, особливо таких, як збір і фільтрація зображень. Етапи, що мали меншу часову ефективність, продемонстрували обґрунтовану доцільність використання універсальних і точніших моделей.

Фільтрація нерелевантних зображень із застосуванням великих мовних моделей, зокрема ChatGPT-4o, забезпечила високу точність, що досягла 95,8%. Це свідчить про ефективність обраного підходу для покращення якості зібраних даних.

Автоматичне тегування і створення текстових описів природною мовою показали високу точність роботи. Найкращі результати в генерації тегів продемонструвала модель WD14 ViT v3, а для текстових описів — модель BLIP, що забезпечує достатній рівень точності для задач навчання моделей генерації зображень. У свою чергу, автоматичне анотування зображень продемонструвало високу якість із середнім рівнем перекриття між вручну створеними та автоматично згенерованими обмежувальними рамками на рівні 91,65%, що є достатнім для практичного використання.

Таким чином, отримані результати підтверджують ефективність розробленого рішення та його перевагу над аналогами.

5 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проекту

Для опису ідеї проекту проаналізуємо зміст ідеї що пропонується, можливі напрямки застосування, та основні вигоди що може отримати користувач цього програмного продукту. Результати аналізу наведено у таблиці 5.1.

Таблиця 5.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка бібліотеки для автоматизованого створення наборів даних зображень, включаючи їх фільтрацію, розмітку, аугментацію, створення синтетичних зображень, оцінку та покращення зібраних зображень	Комп'ютерний зір (класифікація, розпізнавання об’єктів, генерація зображень	Зниження витрат на створення наборів даних зображень, в тому числі і часових, підвищення якості та масштабованості наборів даних зображень для задач класифікації, розпізнавання об'єктів та генерації зображень.

Аналіз техніко-економічних переваг ідеї наведено в таблиці 5.2, де властивості програмного продукту що розглядається порівнюється з конкурентами.

Таблиця 5.2 — Опис ідеї стартап-проекту

№ п/п	Техніко-економічні характеристики ідеї	Товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Parcel 2d	LSUN	DatasetDM			
1.	Веб-скрапінг зображень	+	+	+	-		+	
2.	Фільтрація нерелевантних зображень	+	+	-	-			+
3.	Аугментація набору даних	+	+	-	+			+
4.	Створення синтетичних зображень за допомогою генеративних моделей	+	-	-	+			+

Продовження таблиці 5.2

№ п/п	Техніко-економічні характеристики ідеї	Товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проект	Parcel 2d	LSUN	DatasetDM			
5.	Створення наборів даних для багатьох задачі машинного навчання	+	-	-	-			+
6.	Повна автоматизація процесу	+-	+-	-	-			+

5.2 Технологічний аудит ідеї проекту

Для проведення технологічного аудиту ідеї проекту було проаналізовано які технології є необхідними для реалізації ідеї цього проекту. У таблиці 5.3 проведено аналіз технологічної здійсненності ідеї проекту.

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Автоматизація збору та вилучення даних з Інтернету	Веб-скрапінг, Selenium	Наявна	Доступна
2.	Автоматичне анотування зображень	Згорткові нейронні мережі (CNN), візуальні трансформери (VIT), CLIPSeg, BLIP, WD	Навчені моделі наявні, але потребують доопрацювання	Доступна
3.	Аугментація даних стандартними методами	Геометричні та фотометричні трансформації, бібліотека albumentations	Наявна	Доступна
4.	Аугментація даних шляхом створення синтетичних зображень	Дифузійні моделі генерації зображень, Stable Diffusion	Моделі для генерації наявні, але потрібно пристосувати їх для задачі аугментації	Доступна

Продовження таблиці 5.3

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
5.	Фільтрація нерелевантних зображень	Аналіз вмісту зображень по його текстовому опису, Google Image, LLM ChatGPT-4o в поєднанні з моделлю тегування WD	Частково наявна, але лише для зображень що мають детальний текстовий опис	Не доступна
6.	Оцінка якості зображень	Нейронні мережі, метрики NIQE, BRISQUE, PIQE, LIQE, бібліотека pyiqa	Наявні	Доступні
7.	Покращення якості зображення	Класичні алгоритми зменшення шуму, артефактів стиснення, генеративна модель Stable Diffusion	Наявні, потребують доопрацювання	Доступні
Обрані технології реалізації ідеї проекту: Selenium, CLIPSeg, BLIP, WD, albumentation, Stable Diffusion, pyiqa, ChatGPT-4o				

За результатами аналізу таблиці можна зробити висновок, що технологічна реалізація проекту є можливою. Обраний технологічний шлях базується на застосуванні доступних технологій, які вже наявні на ринку і описані в останньому рядку таблиці. Ці технології забезпечують ефективність реалізації та адаптацію проекту до поточних потреб.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

Першим кроком визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, є аналіз попиту. У таблиці 5.4 проведено попередню характеристику потенційного ринку стартап-проекту.

Таблиця 5.4 – Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку	Характеристика
1.	Кількість головних гравців	Близько 10
2.	Загальний обсяг продаж	Ринок оцінюється в мільярди доларів через високий попит на набори даних для AI і комп'ютерного зору.
3.	Динаміка ринку	Зростає
4.	Наявність обмежень для входу	Потреба у високій обчислювальній потужності та доступі до великих масивів даних.
5.	Специфічні вимоги до стандартизації	Відповідність вимогам конфіденційності, зокрема GDPR
6.	Середня норма рентабельності в галузі	40-50%

За результатами аналізу таблиці можна зробити висновок, що ринок є привабливим для входження. Стрімке зростання динаміки ринку, великий обсяг продаж та високий попит на інноваційні рішення у сфері автоматизації роботи з даними створюють сприятливі умови для стартапу. Незважаючи на певні обмеження, такі як необхідність дотримання стандартів конфіденційності та вимоги до обчислювальних ресурсів, потенційна рентабельність проекту свідчить про його перспективність. Основна характеристика майбутніх клієнтів наведена у таблиці 5.5.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Створення високоякісних наборів даних для AI-моделей	Стартапи та компанії в сфері штучного інтелекту (AI), компанії з розробки програмного забезпечення	Зацікавлені в швидкому отриманні наборів даних для тренування моделей, високі вимоги до точності	Сумісність з існуючими AI-фреймворками, висока якість даних
Підвищення ефективності алгоритмів комп'ютерного зору	Компанії, що займаються комп'ютерним зором та робототехнікою	Зацікавлені в обробці великих обсягів даних, потребують швидкої аугментації наборів зображень	Простота інтеграції з існуючими системами, швидкість роботи

Продовження таблиці 5.5

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Покращення якості даних для медичних досліджень	Медицини, біотехнологічні компанії, дослідницькі лабораторії	Високі вимоги до точності та достовірності даних, обмеження на використання даних для конфіденційності	Висока точність, відповідність стандартам медичної практики
Виявлення дефектів у виробництві	Промислові підприємства, які використовують автоматизовані системи контролю якості	Шукають ефективні способи для автоматичного виявлення дефектів та економії часу на перевірку	Надійність, масштабованість, адаптація до конкретного виробництва
Створення наборів даних для безпеки та моніторингу	Компанії в сфері безпеки, урядові та приватні охоронні компанії	Зацікавлені в швидкому створенні наборів для систем розпізнавання осіб, відео-аналітики	Конфіденційність, висока якість зображень та даних

Важливо розглянути фактори що можуть перешкоджати ринковому впровадженню проекту. Такі фактори розглянуто у таблиці 5.6

Таблиця 5.6 – Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1.	Конкуренція на ринку	Присутність великих компаній з аналогічними рішеннями	Визначення унікальних переваг, диференціація продукту
2.	Нестабільність ринку	Зміни в ринкових умовах, економічна нестабільність	Адаптація стратегії до змін на ринку, диверсифікація
3.	Проблеми з фінансуванням	Високі витрати на запуск та підтримку проекту	Пошук інвесторів, зменшення витрат, залучення грантів
4.	Технічні обмеження	Невідповідність обраних технологій до реальних вимог	Оцінка та адаптація технологій під специфіку проекту
5.	Зміни в законодавстві	Нові вимоги до конфіденційності, захисту даних	Відповідність новим стандартам, оновлення політик
6.	Технічні проблеми із масштабуванням	Неможливість швидкого масштабування проекту	Пошук оптимальних технічних рішень для масштабування

Крім цього, існують фактори що сприяють ринковому впровадженню проекту. Ці фактори розглянуто у таблиці 5.7.

Таблиця 5.7 — Фактори можливостей

№	Фактор	Зміст загрози	Можлива реакція компанії
1.	Високий попит на набори даних для AI	Зростаючий ринок AI та комп'ютерного зору	Розширення продукту для різних галузей
2.	Розвиток технологій глибокого навчання	Удосконалення технологій для автоматизованого анотування	Адаптація новітніх моделей, зниження вартості обробки даних
3.	Підвищення інтересу до синтетичних даних	Зростання популярності синтетичних наборів даних	Розробка рішень для генерації високоякісних даних
4.	Прогрес у розробці генеративних моделей	Покращення генеративних моделей, таких як GAN та Stable Diffusion	Використання нових моделей для покращення якості даних
5.	Технології в області безпеки	Високий попит на рішення для розпізнавання облич, відео-аналітики	Розробка спеціалізованих продуктів для безпеки
6.	Вихід на нові ринки	Розширення застосування технології на нові ринки (наприклад, медицину, сільське господарство)	Адаптація продукту до специфічних вимог ринку

Ступеневий аналіз конкуренції на ринку розглянуто у таблиці 5.8.

Таблиця 5.8 — Ступеневий аналіз конкуренції на ринку

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1.	Тип конкуренції: монополістична	Є кілька великих компаній, які домінують на ринку наборів даних для AI	Необхідно розробляти унікальні пропозиції на основі якості та специфічності даних
2.	За рівнем конкурентної боротьби: національний	Основна конкуренція відбувається на національному рівні, хоча є й міжнародні гравці	Адаптація стратегії під специфіку місцевих ринків, орієнтація на національні потреби
3.	За галузевою ознакою: внутрішньогалузева	Конкуренція між компаніями, що займаються створенням наборів даних для машинного навчання	Активне впровадження інновацій, підтримка високої якості даних та їх анотацій
4.	Конкуренція за видами товарів: товарно-родова	Конкуренція в межах родової категорії товарів, що стосуються зображень та наборів даних для AI	Потрібно створювати набори даних, які відповідають специфічним потребам різних галузей (медицина, безпека, сільське господарство)

Продовження таблиці 5.8

№	Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
5.	За характером конкурентних переваг: цінова	Конкуренція, зокрема в ціні наборів даних або послуг, що їх супроводжують	Визначення оптимальних цінових стратегій, комбінування ціни та якості даних
6.	За інтенсивністю: марочна	Конкуренція між компаніями, орієнтованими на брендинг та репутацію в галузі машинного навчання	Інвестування в розвиток бренду через створення якісних та надійних наборів даних, що відповідають вимогам ринку

Детальний аналіз умов конкуренції в галузі за М. Портером наведено у таблиці 5.9.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	DatasetDM, Parsel2d	LSUN	Google, Bing, Baidu, фотостоки	ML-інженери	Платформи для замовлення ручного збору наборів даних
Висновки	Низька інтенсивність через різницю в функціоналі	Так, є можливості для входу на ринок, потенційні конкуренти є, але їх небагато	Більша частина зображень в Інтернеті доступна через пошукові системи, тому вони обмежують збір цих зображень напряму, продаючи доступ до їхніх продуктів	Клієнти диктують умови щодо якості датасетів	Обмеження невеликі, оскільки ці платформи сильно програють в часі збору набору даних

За результатами аналізу можна зробити висновок, що ринок для стартапу в сфері створення наборів даних має можливості для входу, з низькою інтенсивністю конкуренції завдяки різниці в функціоналі конкурентів. Потенційні конкуренти існують, але їх небагато. Постачальники даних та клієнти мають значний вплив на ринок, диктуючи умови щодо якості та специфікацій. Водночас обмеження для роботи на ринку незначні, оскільки платформи для ручного збору наборів даних програють в часі збору.

Для того, щоб бути конкурентоспроможним на ринку, проект має забезпечити високу якість і різноманітність даних, швидкість збору, а також відповідати вимогам клієнтів щодо специфікації. Важливим аспектом є конкурентоспроможні ціни та стабільні партнерські відносини з постачальниками даних. Фактори конкурентоспроможності докладніше розглянуто в таблиці 5.10.

Таблиця 5.10 — Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1.	Якість даних	Висока якість наборів даних дозволяє забезпечити точність і ефективність моделей машинного навчання, що є ключовим для задоволення вимог клієнтів.
2.	Швидкість збору даних	Збір даних у реальному часі або за мінімальний проміжок часу дозволяє компанії швидко реагувати на зміни в попиті та бути більш ефективною у порівнянні з конкурентами.

Продовження таблиці 5.10

№	Фактор конкурентоспроможності	Обґрунтування
3.	Адаптивність до потреб клієнтів	Гнучкість у налаштуванні наборів даних під специфічні вимоги клієнтів дозволяє виграти конкурентну боротьбу за рахунок персоналізованих пропозицій.
4.	Інноваційність технологій	Використання новітніх технологій для автоматизації збору та анотування даних дозволяє скоротити витрати та покращити якість результату, даючи конкурентні переваги.
5.	Конкурентоспроможні ціни	Встановлення привабливих цін на основі економії від автоматизації процесів та зниження витрат дозволяє приваблювати більшу кількість клієнтів.
6.	Партнерські відносини з постачальниками	Наявність надійних постачальників даних, що гарантують якість та безперервність поставок, зміцнює позиції на ринку та знижує ризики.

Базуючись на визначених факторах конкурентоспроможності було проведено аналіз слабких сторін стартап-проекту. Порівняльний аналіз наведено у таблиці 5.11.

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін бібліотеки для інтелектуального створення наборів даних зображень

№	Фактор конкурентоспроможності	Бали 1- 20	Рейтинг товарів-конкурентів у порівнянні з запропонованим						
			-3	-2	-1	0	1	2	3
1.	Якість даних	16			+				
2.	Швидкість збору даних	14			+				
3.	Адаптивність до потреб клієнтів	18	+						
4.	Інноваційність технологій	18		+					
5.	Конкурентоспроможні ціни	19			+				
6.	Партнерські відносини з постачальниками	16					+		

Сильні та слабкі сторони проекту, а також можливості та загрози наведені у таблиці 5.12.

Таблиця 5.12 — SWOT аналіз стартап-проекту

Сильні сторони (S): - висока якість та різноманітність наборів даних; - швидкість збору та обробки даних завдяки автоматизації; - адаптивність до специфічних вимог клієнтів; - інноваційний підхід до автоматизації створення наборів даних.	Слабкі сторони (W): - залежність від постачальників зовнішніх даних для створення наборів; - обмеженість ресурсів для масштабування на великі ринки або нові сегменти; - потреба в значних інвестиціях на початкових етапах для розвитку технології та інфраструктури.
Можливості (O): - розширення ринку завдяки зростаючому попиту на набори даних для AI і комп'ютерного зору; - використання нових технологій для генерації синтетичних даних; - зростання попиту на індивідуальні набори даних для різних галузей (медицина, сільське господарство); - партнерства з великими компаніями, що шукають готові рішення для машинного навчання.	Загрози (T): - поява нових, більш інноваційних конкурентів з кращими технологіями; - нестабільність ринку або зміни в політиці щодо захисту даних та конфіденційності; - зниження вартості збору даних за допомогою нових дешевших технологій або альтернативних платформ.

У таблиці 5.13 наведено альтернативи ринкового впровадження стартап-проекту.

Таблиця 5.13 — Альтернативи ринкового впровадження стартап-проєкту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Розробка унікальної пропозиції на основі високої якості та швидкості збору наборів даних	Висока	6-12 місяців
2.	Партнерство з великими компаніями для спільного доступу до даних та технологій	Низька	12-18 місяців (переговори, укладення угод)
3.	Інвестування в маркетингові стратегії для залучення клієнтів через конкурентоспроможні ціни	Середня	3-6 місяців
4.	Вихід на нові ринки (медицина, сільське господарство) через адаптацію продукту під конкретні потреби	Середня	8-12 місяців
5.	Використання технології генерації синтетичних даних для зменшення витрат на збір даних	Висока	3-5 місяців

Після аналізу альтернатив з таблиці 5.13 було обрано альтернативу номер 5 через найнижчі строки реалізації та високу ймовірність отримання ресурсів.

5.4 Розроблення ринкової стратегії проекту

Першим кроком розроблення ринкової стратегії проекту є визначення стратегії охоплення ринку. Опис цільових груп потенційних користувачів наведено у таблиці 5.14.

Таблиця 5.14 — Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	ML-інженери в компаніях, що працюють з AI	Висока: потреба у великій кількості даних для тренування моделей	Високий попит на високоякісні, анотовані набори даних для навчання моделей	Середня: існують великі конкуренти, що надають аналогічні послуги	Середня: потребує технологічних інвестицій та високої якості даних
2.	Медичні компанії та дослідницькі лабораторії	Висока: потреба у медичних наборах даних для розпізнавання захворювань	Помірний попит на медичні набори даних, в основному специфічні запити	Середня: конкуренція з великими компаніями в цій галузі	Висока: необхідні спеціалізовані дані та сертифікація

Продовження таблиці 5.14

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
3.	Платформи для автоматизації бізнес-процесів	Середня: використовують набори даних для аналізу і прогнозування бізнес-процесів	Середній попит на стандартизовані набори даних для аналізу	Висока: конкуренція з великими платформами для обробки даних	Середня: доступ до великої кількості бізнес-кейсів
4.	Ентузіасти, що самостійно навчають власні моделі машинного навчання, зокрема моделі генерації зображень	Висока: потреба в великій кількості якісних даних, недоступних звичайним користувачам	Високий попит на високоякісні набори даних зображень з текстовими описами для навчання моделей	Низька: більшість конкурентів націлені на корпоративний ринок	Середня: потребує більш детальної оцінки потреби користувачів та їх платоспроможності
Обрано групи – 1 та 4					

На основі проведеного аналізу цільових груп потенційних користувачів визначається стратегія охоплення ринку. Базову стратегію розвитку сформовано у таблиці 5.15.

Таблиця 5.15 — Визначення базової стратегії розвитку

Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Надання функціональності що відсутня у товарів-замінників, підвищення якості функціоналу що є у конкурентів	Стратегія концентрованого маркетингу	Висока якість даних, швидкість збору, гнучкість під потреби клієнтів	Стратегія диференціації

Наступним важливим кроком є вибір стратегії конкурентної поведінки. Визначення базової стратегії конкурентної поведінки наведено у таблиці 5.16.

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні, є конкуренти на ринку	Компанія буде забирати споживачів у конкурентів через кращу якість та швидкість поставок	Компанія буде диференціювати продукт шляхом покращення функціональності та зручності використання	Стратегія виклику лідера: флангова атака на слабкі сторони лідерів ринку

Наступним кроком є розробка стратегії позиціонування на основі вимог до продукту та обраної базової стратегії розвитку. Визначення стратегії позиціонування наведено у таблиці 5.17.

Таблиця 5.17 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1.	Висока якість та точність даних для тренування моделей AI	Стратегія лідерства за витратами	Висока швидкість збору даних, інноваційні методи автоматизації збору даних	«Інноваційний», «Точність», «Швидкість»
2.	Гнучкість у налаштуванні наборів даних під специфічні потреби	Стратегія диференціації	Спеціалізовані набори даних для різних галузей, адаптовані під вимоги клієнтів	«Індивідуальний підхід», «Адаптивність», «Гнучкість»
3.	Конкурентоспроможні ціни з високою якістю	Стратегія диференціації	Конкурентні ціни при збереженні високої якості даних	«Доступність», «Ціна-якість», «Надійність»

Відповідно до проведеного аналізу можна зробити висновок, що найкраще обрати як базову стратегію розвитку – стратегію диференціації, як базову стратегію конкурентної поведінки – стратегію виклику лідеру.

5.5 Розроблення маркетингової програми стартап-проекту

У таблиці 5.18 визначено ключові переваги концепції потенційного товару

Таблиця 5.18 — Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1.	Висока якість даних для тренування моделей AI	Точні, високоякісні та різноманітні набори даних для покращення ефективності AI	Швидкість збору даних, висока точність, більший рівень автоматизації процесу збору
2.	Індивідуальні набори даних під специфічні потреби	Можливість адаптації наборів даних для конкретних вимог клієнта	Гнучкість в налаштуванні даних під конкретні вимоги, спеціалізовані рішення для різних галузей
3.	Конкурентоспроможні ціни з високою якістю	Доступні ціни для якісних наборів даних	Конкурентні ціни при збереженні високої якості, доступність продукту для широкого кола клієнтів
4.	Швидкість отримання наборів даних	Створення наборів даних за короткий час, що дозволяє прискорити розробку моделей	Швидкість і ефективність у зборі та анотуванні даних, що дозволяє заощаджувати час клієнтів

Продовження таблиці 5.18

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
5.	Різноманітність даних	Можливість створення синтетичних даних при недостатній кількості зображень	Інноваційні методи створення синтетичних зображень, базовані на Stable Diffusion

Наступним кроком є розробка трирівневої маркетингової моделі товару. Опис трьох рівнів моделі товару наведено у таблиці 5.19.

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
1. Товар за задумом	Товар покликаний задовольнити потреби в високоякісних, різноманітних та доступних наборах даних для машинного навчання і комп'ютерного зору, що дозволяє покращити ефективність моделей AI

Продовження таблиці 5.19

Рівні товару	Сутність та складові		
2. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	Вартість обслуговування	М	Вр
	Інтеграція у програмне забезпечення	М	Вр/Тх/Тл/Е
	Забезпечення відмовостійкості	Нм	Вр/Тх
	Забезпечення високого рівня швидкості	М	Вр/Тх/Тл
	Інтерфейс	М	Вр/Тх/ Е
	Адаптація до запитів клієнта	М	Вр/Тх/Ор
	Якість: проведення експериментів для відбору найкращих моделей машинного навчання		
	Пакування: програмна бібліотека		
	Марка: IDSynth		
3. Товар із підкріпленням	До продажу: наявна повна документація, акції на придбання декількох ліцензій, знижки для певних сегментів на покупку товару		
	Після продажу: додаткова підтримка спеціалістів налаштування, підтримка з боку розробника		
За рахунок чого потенційний товар буде захищено від копіювання: захист інтелектуальної власності, ліцензія			

М/Нм – монотонні/немонотонні; Вр/Тх/Тл/Е/Ор – вартісні/ технічні/ технологічні/ ергономічні/ органолептичні. Далі слід визначити цінові межі, якими необхідно керуватись при встановленні ціни на потенційний товар. Визначення меж встановлення ціни наведено у таблиці 5.20.

Таблиця 5.20 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
Від 4000 гривень за 1000 зображень	Від 10000 гривень на місяць	Від 80000 гривень на місяць	Від 800 гривень до 8000 гривень за місяць в залежності від плану

Наступним кроком є визначення оптимальної системи збуту. У таблиці 5.21 наведено формування системи збуту.

Таблиця 5.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Пробний період, купівля ліцензії на використання	Контакти зі споживачами, формування попиту, досліди в області маркетингу	Канал нульового рівня (виробник безпосередньо продає товару клієнту)	Ліцензія на використання

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій. Концепцію маркетингових комунікацій наведено у таблиці 5.22.

Таблиця 5.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Установи та підприємства можуть мати різні ситуації, де може знадобитися розроблене рішення	Соцмережі, пошта, конференції	Висока точність даних, швидкість збору та адаптивність під потреби клієнтів	Підкреслити швидкість, точність та інноваційність продукту, що допоможе покращити ефективність моделей AI	"Максимальна ефективність ваших моделей за допомогою точних і швидких даних. Ваші моделі — наші дані!"

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

Висновки по розділу

Проведений аналіз показав, що є значний потенціал для ринкової комерціалізації стартапу, спрямованого на створення наборів даних для машинного навчання та комп'ютерного зору. На ринку є попит на високоякісні, різноманітні та спеціалізовані набори даних. Динаміка ринку є позитивною, оскільки зростає потреба в ефективних рішеннях для збору та обробки даних. Рентабельність проекту на ринку залежить від здатності стартапу швидко реагувати на зміни попиту, підтримувати високу якість продуктів і встановлювати конкурентоспроможні ціни.

Перспективи впровадження проекту позитивні, оскільки цільові групи споживачів мають високу готовність сприйняти продукт, зокрема ML-інженери та ентузіасти, що самостійно навчають моделі машинного навчання. Бар’єри входження на ринок є, але вони не є надмірно високими, особливо для компаній, які зможуть забезпечити високу якість та швидкість збору даних. Стан конкуренції в галузі дозволяє знайти свою нішу, хоча існують великі гравці. Конкурентоспроможність проекту може бути забезпечена через інноваційні підходи, високу якість даних і гнучкість в налаштуванні під специфічні потреби клієнтів.

Для ринкової реалізації проекту доцільно обрати стратегію диференціації, спрямовану на надання індивідуальних рішень для різних галузей, а також стратегію концентрованого маркетингу для залучення основних цільових груп споживачів. Це дозволить зберегти конкурентні переваги та захистити продукт від копіювання.

Подальша імплементація проекту є доцільною, оскільки ринок для продукту має значний потенціал, а стратегія розвитку та позиціонування дозволяє ефективно охопити цільові групи та забезпечити стійкий попит на товар.

ВИСНОВКИ

У процесі виконання магістерської дисертації розглянуто питання пов’язані з автоматизованим створенням високоякісних наборів даних зображень.

На основі аналізу сучасних методів і рішень, сформульовано задачу розробки бібліотеки, що об’єднує інструменти збору, розмітки, анотування, аугментації, оцінки та покращення якості зображень а також створення синтетичних зображень для розширення наборів даних. Для реалізації даного підходу було розроблено методи, що використовують сучасні технології машинного навчання, зокрема моделі на основі прихованої дифузії, трансформери та LLM. Для програмної реалізації було використано мову Python та її бібліотеки selenium, pytorch, transformers, diffusers, а також генеративну модель Stable Diffusion. Все перераховане є безкоштовним та має відкритий код, що забезпечує прозорість та доступність створеного інструменту для широкого кола дослідників та розробників.

Реалізоване програмне забезпечення забезпечує автоматизацію основних етапів: від збору даних до оцінки якості створених зображень. Проведено маркетинговий аналіз, що включає опис ідеї, аудит технологічної бази, аналіз ринкових можливостей і розробку ринкової стратегії для запропонованого інструменту.

Результати роботи магістерської дисертації пройшли апробацію та опубліковані у збірнику тез конференції SoftTech-2024.

Наукова новизна одержаних результатів полягає в наступному:

вперше:

– запропоновано метод фільтрації нерелевантних зображень з використанням моделей автоматичного тегування та LLM ChatGPT-4o;

– запропоновано два методи для синтетичної генерації та аугментації наборів даних зображень за допомогою моделей прихованої дифузії Stable Diffusion;

удосконалено:

– зменшено час збору зображень з пошукових систем у 1.5 рази в порівнянні з аналогами;

здобуло подальший розвиток:

– інтеграція інструментів для збору, генерації, аугментації, анотування, оцінки та покращення якості зображень в єдину бібліотеку;

– підтримка різних сценаріїв створення набору даних зображень для широкого кола задач машинного навчання.

Практична значимість отриманих результатів полягає у розробці бібліотеки, здатної значно скоротити витрати часу і ресурсів на створення високоякісних наборів даних, що є критично важливим для задач машинного навчання у таких галузях, як медицина, безпека та промисловість. Універсальність системи забезпечується підтримкою різних задач машинного навчання, що робить продукт доступним для широкого кола користувачів. Інтеграція сучасних технологій, таких як генеративні моделі та великі мовні моделі, забезпечують створення більш різноманітних та якісних наборів даних. Економічна ефективність досягається через зменшення витрат на ручну обробку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Naumann A., Hertlein F., Zhou B., Dorr L., Furmans K. Scrape, Cut, Paste and Learn: Automated Dataset Generation Applied to Parcel Logistics. 21st IEEE International Conference on Machine Learning and Applications (ICMLA). Nassau, Bahamas, 2022. pp. 1026–1031.
2) Hofstetter R. A Step-by-Step Guide for Data Scraping. The Machine Age of Customer Insight. Emerald Publishing Limited, Leeds, 2021. pp. 129–143.
3) Yanai K., Kobus B. Probabilistic web image gathering. Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval. Association for Computing Machinery, New York, 2010. pp. 57–64.
4) Konishi K., Toyama T., Watanabe A. Fashion-related Image Gathering and Retrieval Using Meta Data Generated at Image Sharing Site. Transactions of the Japanese Society for Artificial Intelligence. 2010. vol. 25, no. 1. pp. 25–36.
5) Xu M., Yoon S., Fuentes A., Park D. A Comprehensive Survey of Image Augmentation Techniques for Deep Learning. Pattern Recognition. 2023. vol. 137.
6) Mikołajczyk A., Grochowski M. Data Augmentation for Improving Deep Learning in Image Classification Problem. International Interdisciplinary PhD Workshop (IIPhDW). Świnoujście, Poland, 2018. pp. 117–122.
7) Song F. Y. Y. Z. S., Xiao A. S. J. LSUN: Construction of a Large-scale Image Dataset Using Deep Learning with Humans in the Loop. 2015.
8) Zhang Y., Sun Q., Zhou Y., He Z., Yin Z., Wang K., Liu Z. Bamboo: Building Mega-scale Vision Dataset Continually with Human-Machine Synergy. 2022.
9) Zhou J., Cao R., Kang J., Guo K., Xu Y. An Efficient High-quality Medical Lesion Image Data Labeling Method Based on Active Learning. 2020.
10) Shu J. P., Li J., Jin Z. F. Construction of Crack Image Dataset Using Active Learning. In Bridge Safety, Maintenance, Management, Life-Cycle, Resilience

and Sustainability. CRC Press, 2022. pp. 1238–1248.

11) Chen Y., Zeng X., Chen X., Guo W. A Survey on Automatic Image Annotation. Applied Intelligence. 2020. vol. 50. pp. 3412–3428.
12) Wu B., Jia F., Liu W., Ghanem B. Diverse Image Annotation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017. pp. 2559–2567.
13) Murthy V. N., Maji S., Manmatha R. Automatic Image Annotation Using Deep Learning Representations. Proceedings of the 5th ACM on International Conference on Multimedia Retrieval. 2015. pp. 603–606.
14) Dindin M., Radermecker O., Missler P. L. Motion and Out-of-focus Blurriness: Identification and Quantification. NIST PSCR Challenge. 2020.
15) Yeh C. H., Cheng T. Y., Hsieh H. Y., Lin C. E., Ma Y., Markham A., Chen Y. Gen4Gen: Generative Data Pipeline for Generative Multi-Concept Composition. 2024.
16) Siemens S., Kästner M., Reithmeier E. Synthetically Generated Microscope Images of Microtopographies Using Stable Diffusion. In Automated Visual Inspection and Machine Vision V. SPIE, 2023. vol. 12623. pp. 52–59.
17) Hong S., Choi B., Ham Y., Jeon J., Kim H. Hyper-Scale Construction Dataset Synthesis Through Stable Diffusion for Machine Learning Training. 2023.
18) Wu W., Zhao Y., Chen H., Gu Y., Zhao R., He Y., Shen C. DatasetDM: Synthesizing Data with Perception Annotations Using Diffusion Models. Advances in Neural Information Processing Systems. 2023. vol. 36. pp. 54683–54695.
19) Wei Y., Akinci B. Synthetic Image Data Generation for Semantic Understanding in Everchanging Scenes Using BIM and Unreal Engine. In Computing in Civil Engineering 2021. 2021. pp. 934–941.
20) Chandler D. M. Seven Challenges in Image Quality Assessment: Past, Present,

and Future Research. International Scholarly Research Notices. 2013.

21) Lu Y., Chen D., Olaniyi E., Huang Y. Generative Adversarial Networks (GANs) for Image Augmentation in Agriculture: A Systematic Review. Computers and Electronics in Agriculture. 2022.
22) Buslaev A., Iglovikov V. I., Khvedchenya E., Parinov A., Druzhinin M., Kalinin A. A. Albumentations: Fast and Flexible Image Augmentations. Information. 2020. vol. 11, no. 2.
23) Bahrololoum A., Nezamabadi-pour H. A Multi-expert Based Framework for Automatic Image Annotation. Pattern Recognition. 2017. vol. 61. pp. 169–184.
24) Jayaraj R., Lokesh S. Automatic Image Annotation Using Adaptive Convolutional Deep Learning Model. Intelligent Automation & Soft Computing. 2023. vol. 36, no. 1.
25) Adnan M. M., Rahim M. S. M., Khan A. R., Saba T., Fati S. M., Bahaj S. A. An Improved Automatic Image Annotation Approach Using Convolutional Neural Network-Slantlet Transform. IEEE Access. 2022. vol. 10. pp. 7520–7532.
26) DeCost B. L., Holm E. A. A Computer Vision Approach for Automated Analysis and Classification of Microstructural Image Data. Computational Materials Science. 2015. vol. 110. pp. 126–133.
27) Li L. J., Fei-Fei L. Optimol: Automatic Online Picture Collection via Incremental Model Learning. International Journal of Computer Vision. 2010. vol. 88. pp. 147–168.
28) Sebastianelli A., Del Rosso M. P., Ullo S. L. Automatic Dataset Builder for Machine Learning Applications to Satellite Imagery. SoftwareX. 2021.
29) Verma A., Yadav A. K., Kumar M., Yadav D. Automatic Image Caption Generation Using Deep Learning. Multimedia Tools and Applications. 2024. vol. 83, no. 2. pp. 5309–5325.

30) Hayamizu R., Nakamura S., Takashima S., Kataoka H., Sato I., Inoue N., Yokota R. SIFTer: Self-improving Synthetic Datasets for Pre-training Classification Models. Synthetic Data for Computer Vision Workshop@CVPR. 2024.
31) Kusmierczyk T., Czakon J. Diffusion Models in Practice Part 1: The Tools of the Trade [Електронний ресурс]. 2023. Режим доступу до ресурсу: https://deepsense.ai/diffusion-models-in-practice-part-1-the-tools-of-the-trade/.
32) Rombach R., Blattmann A., Lorenz D., Esser P., Ommer B. High-Resolution Image Synthesis with Latent Diffusion Models. Conference on Computer Vision and Pattern Recognition. 2022.
33) Hu E. J., Shen Y., Wallis P., Allen-Zhu Z., Li Y., Wang S., Chen W. LoRA: Low-Rank Adaptation of Large Language Models. International Conference on Learning Representations. 2021.
34) Lüddecke T., Ecker A. S. Image Segmentation Using Text and Image Prompts. Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR). 2022.
35) Suryawanshi D. Image Recognition: Detection of Nearly Duplicate Images. Diss. California State University Channel Islands. 2018.
36) BRIA Background Removal v1.4 Model Card [Електронний ресурс]. 2024. Режим доступу до ресурсу: https://huggingface.co/briaai/RMBG-1.4.

ДОДАТКИ

ДОДАТОК А

Діаграма послідовності

Рисунок А.1 – Діаграма послідовності роботи з бібліотекою

ДОДАТОК Б

Діаграма архітектури

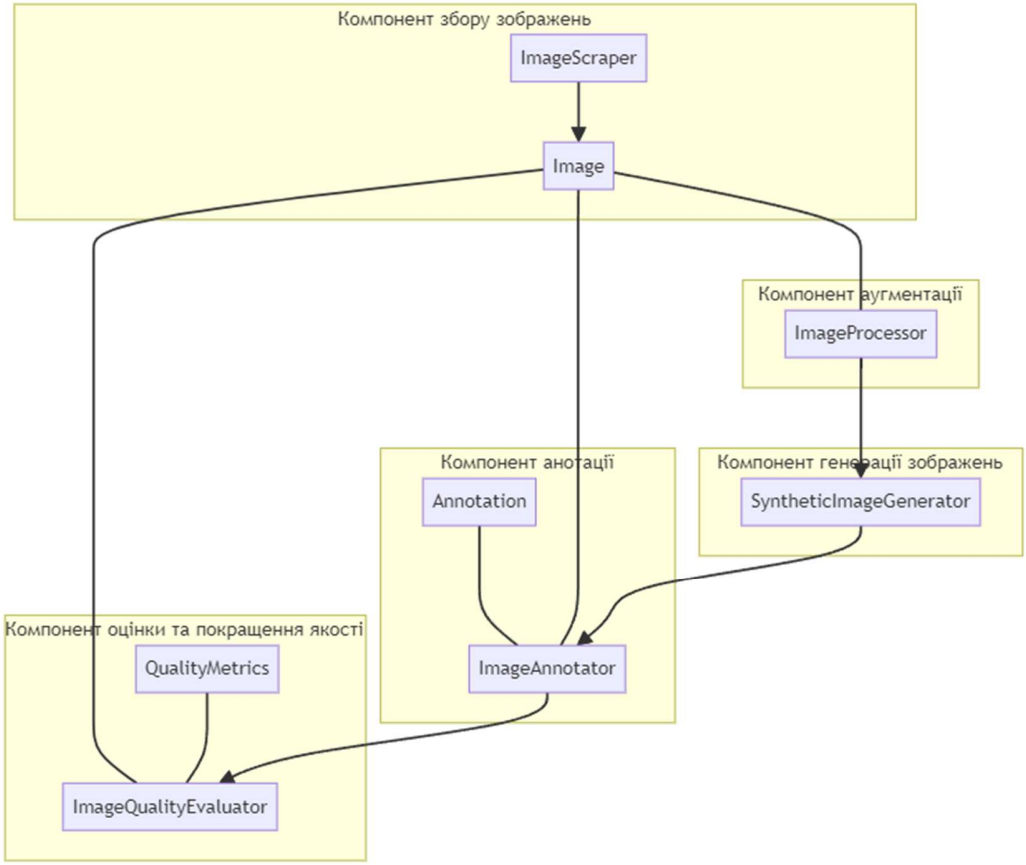

Рисунок Б.1 – Діаграма архітектури бібліотеки

ДОДАТОК В

Діаграма класів

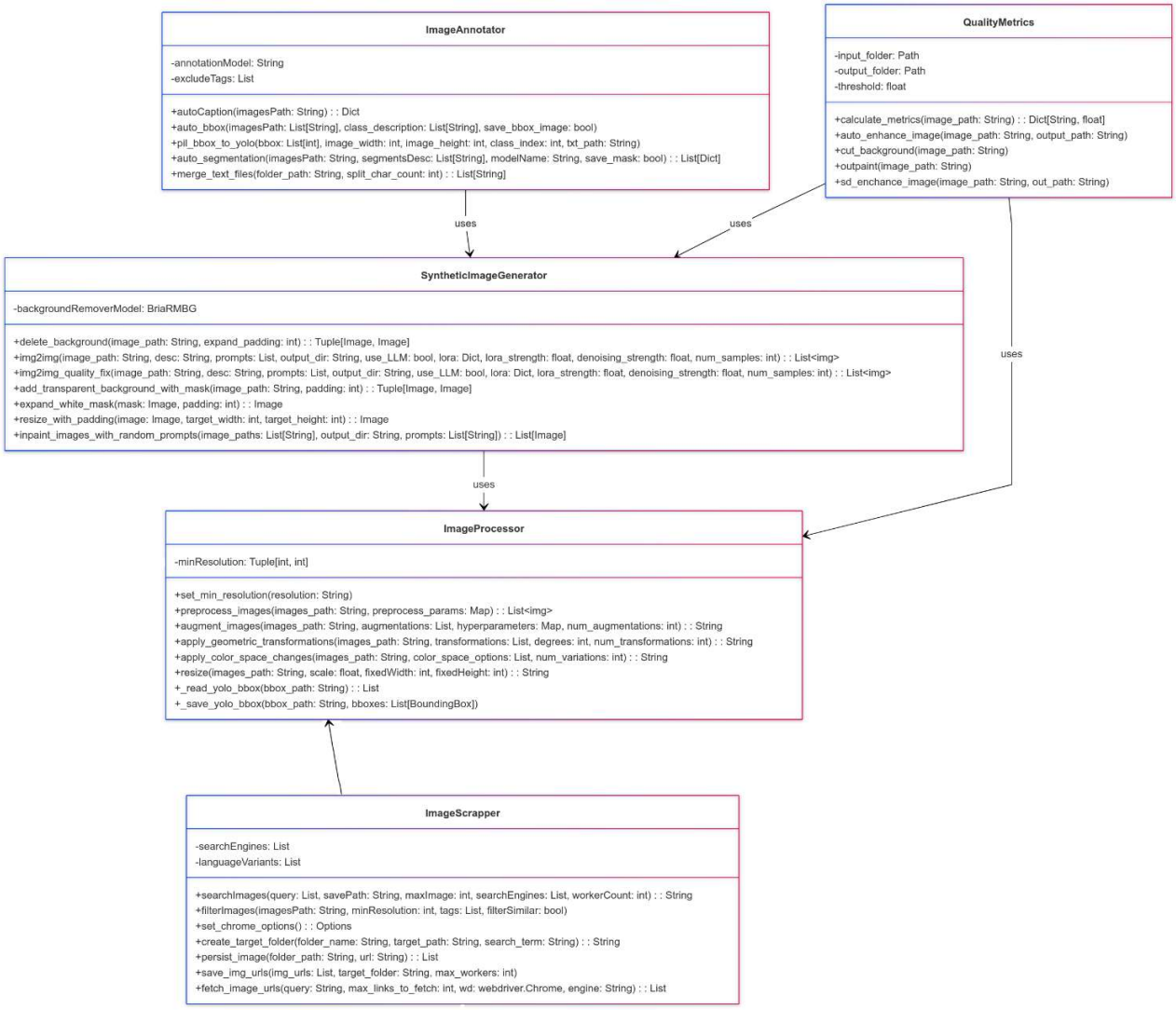

Рисунок В.1 – Діаграма класів бібліотеки

ДОДАТОК Г

ЛІСТИНГ ПРОГРАМНОГО КОДУ

Файл: Augmentator.py

```
import os
from PIL import Image, ImageEnhance
import cv2
import numpy as np
import albumentations as A
from albumentations.pytorch import ToTensorV2
import torchvision.transforms as T
class ImageProcessor:
    def __init__(self):
        self.min_resolution = None
        self.color_space_augmentations = []
        self.geometric_transformations = []
    def set_min_resolution(self, resolution):
        self.min_resolution = tuple(map(int, resolution.split('x')))
    def _read_yolo_bbox(self, bbox_path):
        """Read YOLO format bounding boxes from file"""
        if not os.path.exists(bbox_path):
            return []
        bboxes = []
        with open(bbox_path, 'r') as f:
            for line in f:
                class_id, x_center, y_center, width, height = map(float, line.strip().split())
                bboxes.append([x_center, y_center, width, height, class_id])
        return bboxes
    def _save_yolo_bbox(self, bbox_path, bboxes):
        """Save bounding boxes in YOLO format"""
        with open(bbox_path, 'w') as f:
            for bbox in bboxes:
                # Convert back to class_id, x_center, y_center, width, height
                class_id = int(bbox[4])
```

```
            x_center, y_center, width, height = bbox[:4]
            f.write(f"{class_id} {x_center:.6f} {y_center:.6f} {width:.6f} {height:.6f}\n")
def preprocess_images(self, images_path, preprocess_params):
    processed_images = []
    for image_name in os.listdir(images_path):
        if image_name.lower().endswith(('.png', '.jpg', '.jpeg')):
            image_path = os.path.join(images_path, image_name)
            filename = os.path.splitext(image_name)[0]
        else:
            continue
        image = Image.open(image_path)
        if self.min_resolution:
            image = self._resize_to_min_resolution(image)
        image = self._apply_preprocess_params(image, preprocess_params)
        processed_images.append(image)
    return processed_images
def augment_images(self, images_path, augmentations, hyperparameters, num_augmentations=1):
    augmented_images_path = os.path.join(images_path, 'augmented')
    augmented_bbox_path = os.path.join(images_path, 'augmented')
    os.makedirs(augmented_images_path, exist_ok=True)
    os.makedirs(augmented_bbox_path, exist_ok=True)
    # Build transform with bbox support
    transform = self._build_augmentations(augmentations, hyperparameters)
    for image_name in os.listdir(images_path):
        if not image_name.lower().endswith(('.png', '.jpg', '.jpeg')):
            continue
        # Get corresponding bbox file
        bbox_file = os.path.splitext(image_name)[0] + '.txt'
        bbox_file_path = os.path.join(images_path+'\\bbox', bbox_file)
        # Read image and bboxes
        image = cv2.imread(os.path.join(images_path, image_name))
        bboxes = self._read_yolo_bbox(bbox_file_path)
        if not bboxes:  # Skip if no bounding boxes found
            continue
        # Generate multiple augmentations for each image and its bboxes
```

```
        for i in range(num_augmentations):
            # Prepare bboxes for albumentations format (YOLO to normalized coordinates)
            labels = [bbox[4] for bbox in bboxes]
            boxes = np.array([[bbox[0], bbox[1], bbox[2], bbox[3]] for bbox in bboxes])
            # Apply augmentation to both image and bboxes
            transformed = transform(
                image=image,
                bboxes=boxes,
                class_labels=labels
            )
            augmented_image = transformed['image']
            augmented_bboxes = transformed['bboxes']
            # Save augmented image
            image_save_path = os.path.join(augmented_images_path, f"aug_{i+1}_{image_name}")
            cv2.imwrite(image_save_path, augmented_image)
            # Save augmented bboxes
            bbox_save_path = os.path.join(augmented_bbox_path, f"aug_{i+1}_{bbox_file}")
            transformed_bboxes = []
            for bbox, label in zip(augmented_bboxes, transformed['class_labels']):
                # Convert bbox back to YOLO format
                transformed_bboxes.append([bbox[0], bbox[1], bbox[2], bbox[3], label])
            self._save_yolo_bbox(bbox_save_path, transformed_bboxes)
    return augmented_images_path
def apply_geometric_transformations(self, images_path, transformations, degrees, num_transformations=1):
    geo_transformed_path = os.path.join(images_path, 'geometric')
    os.makedirs(geo_transformed_path, exist_ok=True)
    for image_name in os.listdir(images_path):
        if image_name.lower().endswith(('.png', '.jpg', '.jpeg')):
            image_path = os.path.join(images_path, image_name)
            filename = os.path.splitext(image_name)[0]
        else:
            continue
        image = cv2.imread(image_path)
        # Generate multiple geometric transformations for each image
        for i in range(num_transformations):
```

```
            transformed_image = self._apply_geometric_transformations(image, transformations, degrees)
            save_path = os.path.join(geo_transformed_path, f"geo_{i+1}_{image_name}")
            cv2.imwrite(save_path, transformed_image)
    return geo_transformed_path
def apply_color_space_changes(self, images_path, color_space_options, num_variations=1):
    color_changed_path = os.path.join(images_path, 'color_space')
    os.makedirs(color_changed_path, exist_ok=True)
    for image_name in os.listdir(images_path):
        if image_name.lower().endswith(('.png', '.jpg', '.jpeg')):
            image_path = os.path.join(images_path, image_name)
            filename = os.path.splitext(image_name)[0]
        else:
            continue
        image = cv2.imread(image_path)
        # Generate multiple color space variations for each image
        for i in range(num_variations):
            color_changed_image = self._apply_color_space_changes(image, color_space_options)
            save_path = os.path.join(color_changed_path, f"color_{i+1}_{image_name}")
            cv2.imwrite(save_path, color_changed_image)
    return color_changed_path
def resize(self, images_path, scale=1.0, fixed_width=None, fixed_height=None):
    resized_path = os.path.join(images_path, 'resized')
    os.makedirs(resized_path, exist_ok=True)
    for image_name in os.listdir(images_path):
        if image_name.lower().endswith(('.png', '.jpg', '.jpeg')):
            image_path = os.path.join(images_path, image_name)
            filename = os.path.splitext(image_name)[0]
        else:
            continue
        image = Image.open(image_path)
        resized_image = self._resize(image, scale, fixed_width, fixed_height)
        save_path = os.path.join(resized_path, f"resized_{image_name}")
        resized_image.save(save_path)
    return resized_path
# Internal helper methods
```

```
def _resize_to_min_resolution(self, image):
    if image.size[0] < self.min_resolution[0] or image.size[1] < self.min_resolution[1]:
        return image.resize(self.min_resolution, Image.LANCZOS)
    return image
def _apply_preprocess_params(self, image, preprocess_params):
    if 'contrast' in preprocess_params:
        enhancer = ImageEnhance.Contrast(image)
        image = enhancer.enhance(preprocess_params['contrast'])
    if 'brightness' in preprocess_params:
        enhancer = ImageEnhance.Brightness(image)
        image = enhancer.enhance(preprocess_params['brightness'])
    if 'sharpness' in preprocess_params:
        enhancer = ImageEnhance.Sharpness(image)
        image = enhancer.enhance(preprocess_params['sharpness'])
    return image
def _build_augmentations(self, augmentations, hyperparameters):
    transforms = []
    bbox_params = A.BboxParams(
        format='yolo',  # Using YOLO format
        label_fields=['class_labels']
    )
    if 'scale' in augmentations:
        scale_limit = hyperparameters.get('scale_limit', 0.2)
        transforms.append(A.RandomScale(scale_limit=scale_limit, p=0.5))
    if 'rotate' in augmentations:
        max_angle = hyperparameters.get('rotate_max_angle', 30)
        transforms.append(A.Rotate(limit=max_angle, p=0.5))
    if 'shift' in augmentations:
        shift_limit = hyperparameters.get('shift_limit', 0.1)
        transforms.append(A.ShiftScaleRotate(shift_limit=shift_limit, scale_limit=0, rotate_limit=0, p=0.5))
    if 'crop' in augmentations:
        crop_height = hyperparameters.get('crop_height', 100)
        crop_width = hyperparameters.get('crop_width', 100)
        transforms.append(A.RandomCrop(height=crop_height, width=crop_width, p=0.5))
    if 'flip' in augmentations:
```

```
        transforms.append(A.HorizontalFlip(p=0.5))
    if 'perspective' in augmentations:
        transforms.append(A.Perspective(p=0.5))
    if 'elastic_transform' in augmentations:
        alpha = hyperparameters.get('alpha', 1.0)
        sigma = hyperparameters.get('sigma', 50.0)
        transforms.append(A.ElasticTransform(alpha=alpha, sigma=sigma, alpha_affine=alpha * 0.1, p=0.5))
    if 'brightness' in augmentations:
        brightness_limit = hyperparameters.get('brightness_limit', 0.2)
        transforms.append(A.RandomBrightnessContrast(brightness_limit=brightness_limit, p=0.5))
    if 'contrast' in augmentations:
        contrast_limit = hyperparameters.get('contrast_limit', 0.2)
        transforms.append(A.RandomBrightnessContrast(contrast_limit=contrast_limit, p=0.5))
    if 'add_noise' in augmentations:
        transforms.append(A.GaussNoise(var_limit=(10.0, 50.0), p=0.5))
    if 'gamma' in augmentations:
        gamma_limit = hyperparameters.get('gamma_limit', (80, 120))
        transforms.append(A.RandomGamma(gamma_limit=gamma_limit, p=0.5))
    if 'color_saturation' in augmentations:
        transforms.append(A.HueSaturationValue(hue_shift_limit=10, sat_shift_limit=15, val_shift_limit=10, p=0.5))
    return A.Compose(
        transforms,
        bbox_params=bbox_params
    )
```

Файл: SynteticImageGenerator.py

```
import torch
from PIL import Image, ImageDraw
import random
from diffusers import AutoPipelineForInpainting
from skimage import io
import torch, os
from PIL import Image
from diffusion_utils.RMBG.briarmbg import BriaRMBG
from diffusion_utils.RMBG.utilities import preprocess_image, postprocess_image
```

```
import torch
from diffusers import StableDiffusionImg2ImgPipeline, StableDiffusionDepth2ImgPipeline
from transformers import AutoTokenizer, AutoModel
import numpy as np
from PIL import Image, ImageDraw, ImageOps
import pandas as pd
import os
import shutil
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from time import sleep
import undetected_chromedriver as uc
from selenium.webdriver.support.ui import WebDriverWait
from fake_useragent import UserAgent
from selenium.webdriver.support import expected_conditions as EC
from typing import Union, List, Dict, Optional
import json
from PIL import Image, ImageDraw, ImageOps, ImageFilter, ImageChops
import random
class SyntheticImageGenerator:
    def __init__(self, use_background_remove = True):
        net = BriaRMBG()
        self.device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        # net = BriaRMBG.from_pretrained("briaai/RMBG-1.4")
        # net.to(self.device)
        if use_background_remove:
            self.background_remover_model = BriaRMBG()
            self.background_remover_model = BriaRMBG.from_pretrained("briaai/RMBG-1.4")
            self.background_remover_model.to(self.device)
            self.background_remover_model.eval()
        else:
            self.background_remover_model = None
        self.pipeline = StableDiffusionDepth2ImgPipeline.from_pretrained(
            "stabilityai/stable-diffusion-2-depth",
```

```
        torch_dtype=torch.float16,
    ).to("cuda")
    #self.pipeline.enable_model_cpu_offload()
    #self.background_remover_model = net
def delete_background(self, image_path, expand_padding=0):
    model_input_size = [1024,1024]
    orig_im = io.imread(image_path)
    orig_im_size = orig_im.shape[0:2]
    image = preprocess_image(orig_im, model_input_size).to(self.device)
    result=self.background_remover_model(image)
    result_image = postprocess_image(result[0][0], orig_im_size)
    pil_im = Image.fromarray(result_image)
    #pil_im.save('non_exp_mask.jpg', "JPEG", quality=100)
    pil_im_less = pil_im
    if expand_padding > 0:
        pil_im_less = self.expand_white_mask(pil_im, int(expand_padding/2))
        pil_im = self.expand_white_mask(pil_im, expand_padding)
        pil_im.save('exp_mask.jpg', "JPEG", quality=100)
    no_bg_image = Image.new("RGBA", pil_im.size, (0,0,0,0))
    orig_image = Image.open(image_path)
    no_bg_image.paste(orig_image, mask=pil_im)
    #no_bg_image.save('no_bg_image2.png', "PNG", quality=100)
    #pil_im = self.expand_white_mask(pil_im, expand_padding)
    return no_bg_image, pil_im_less
def gpt_prompter(self, prompt):
    op = webdriver.ChromeOptions()
    op.add_argument(f"user-agent={UserAgent.random}")
    op.add_argument("user-data-dir=./")
    op.add_experimental_option("detach", True)
    op.add_experimental_option("excludeSwitches", ["enable-logging"])
    driver = uc.Chrome(chrome_options=op)
    driver.get('https://chat.openai.com/')
    sleep(5)
    #inputElements = driver.find_elements(By.CSS_SELECTOR, "p.placeholder")
    #inputElement2 = driver.find_elements(By.CSS_SELECTOR, "div.ProseMirror")[0]
```

```
    i = 0
    #inputElement2.click()
    inputElements = driver.find_elements(By.CSS_SELECTOR, "p.placeholder")
    inputElement2 = driver.find_element(By.ID, "prompt-textarea")
    inputElement2.click()
    driver.execute_script("arguments[0].innerText = arguments[1];", inputElements[0], prompt)
    sleep(2)
    inputElement2.send_keys(Keys.ENTER)
    while True:
        try:
            stop_btn = driver.find_element(By.CSS_SELECTOR, "[data-testid='stop-button']")
            print("Елемент з data-testid='stop-button' знайдено!")
            sleep(5)
        except:
            print("Елемент з data-testid='stop-button' не знайдено.")
            break
    inputElements = driver.find_elements(By.TAG_NAME, "p")
    sleep(5)
    results = []
    for element in inputElements:
        results.append(element.text)
    print(results)
    # results = results[0-count:]
    # print(results)
    driver.quit()
    return ' '.join(results)
def read_yolo_annotations(self, txt_file_path, img_width, img_height):
    """
    Зчитує рамки з YOLO-формату і перетворює їх у формат xmin, ymin, xmax, ymax.
    :param txt_file_path: Шлях до текстового файлу з YOLO-рамками.
    :param img_width: Ширина зображення.
    :param img_height: Висота зображення.
    :return: Список рамок у форматі xmin, ymin, xmax, ymax.
    """
    bboxes = []
```

```
    with open(txt_file_path, "r") as file:
        for line in file:
            parts = line.strip().split()
            if len(parts) != 5:
                raise ValueError(f"Неправильний формат рядка: {line}")
            cls, x_center, y_center, width, height = map(float, parts)
            xmin = int((x_center - width / 2) * img_width)
            ymin = int((y_center - height / 2) * img_height)
            xmax = int((x_center + width / 2) * img_width)
            ymax = int((y_center + height / 2) * img_height)
            bboxes.append({"class": int(cls), "xmin": xmin, "ymin": ymin, "xmax": xmax, "ymax": ymax})
    return bboxes
def img2img(self, image_path, desc, prompts, output_dir, use_LLM = False, lora = None, lora_strength = 1, denoising_strength = 0.6,num_samples = 1):
    torch.cuda.empty_cache()
    output_images = []
    input_image = Image.open(image_path)
    width, height = input_image.size
    if 512 / width > 1.25 or 512 / height > 1.25 or width / 512 > 1.5 or height / 512 > 1.5:
        if width < height:
            height = int(512 * (height/width))
            width = 512
        else:
            width = int(512 * (width/height))
            height = 512
    # input_image = input_image.convert('RGB')
    # width, height = input_image.size
    width = (width // 8) * 8
    height = (height // 8) * 8
    input_image = input_image.resize((width, height))
    if not output_dir:
        output_dir = '\\'.join(image_path.split('\\')[:-1])
    n_prompt = "ugly, deformed, not detailed, bad architectures, blurred, too much blurred, motion blur, text, watermark"
    res_path = []
    for prompt in prompts:
        print(prompt)
```

```
if use_LLM:
    gpt_prompt = f"Below is a description of an image whose main topic is {desc}, using simple words, change the description so that the main topic of the image and related objects do not change significantly, while the image description as a whole is significantly different from the input, this is required for augmentation of the dataset, return only the changed description, without further explanation. Here is the input image description: {prompts}'"
    prompt = self.gpt_prompter(gpt_prompt)
for _ in range(num_samples):
    result = self.pipeline(prompt=prompt, image=input_image, negative_prompt=n_prompt, strength=denoising_strength)
    output_images.extend(result.images)
    image_id = str(random.randint(0,999999))
    with open(f"{output_dir}\\generated_{prompt}_{image_id}.txt", 'w') as fp:
        fp.write(prompt)
    result.images[0].save(f"{output_dir}\\generated_{prompt}_{image_id}.png")
    if os.path.exists(output_dir+'\\bbox'):
        if 'Local\\Temp' in image_path:
            txt_file_path = f"{output_dir}\\bbox\\" + image_path.split('\\')[-1].split('.')[0] + '.txt'
        else:
            txt_file_path = image_path.split('.')[0] + '.txt'
            if '\\bbox\\' not in image_path:
                txt_file_path = txt_file_path.replace('\\bbox\\', '\\')
            else:
                image_path = image_path.replace('\\bbox\\', '\\')
        # with Image.open(image_path) as img:
        img_width, img_height = result.images[0].size
        # Зчитування рамок
        #try:
        bboxes = self.read_yolo_annotations(txt_file_path, img_width, img_height)
        bbox = [bboxes[0]['xmin'], bboxes[0]['ymin'], bboxes[0]['xmax'], bboxes[0]['ymax']]
        self.pil_bbox_to_yolo(bbox, img_width, img_height, txt_path=f"{output_dir}\\bbox\\generated_{prompt}_{image_id}.txt")
        img = result.images[0]
        draw = ImageDraw.Draw(img)
        draw.rectangle(bbox, outline="red", width=2)
        path = f"{output_dir}\\bbox\\generated_{prompt}_{image_id}.png"
        img.save(path)
    res_path.append(f"{output_dir}\\generated_{prompt}_{image_id}.png")
print(res_path)
#del self.pipeline
```

```
        return res_path

    def img2img_quality_fix(self, image_path, desc, prompts, output_dir, use_LLM = False, lora = None, lora_strength = 1, denoising_strength = 0.6,num_samples = 1):

        #del self.pipeline

        torch.cuda.empty_cache()

        self.pipeline = StableDiffusionImg2ImgPipeline.from_pretrained(

            "SG161222/Realistic_Vision_V2.0",

            torch_dtype=torch.float16,

        ).to("cuda")

        torch.cuda.empty_cache()

        if not output_dir:

            output_dir = '\\'.join(image_path.split('\\')[:-1])

        output_images = []

        input_image = Image.open(image_path)

        target_min_side = 512

        width, height = input_image.size

        # Determine the scaling factor to make the smaller side equal to target_min_side

        if width < height:

            scale_factor = target_min_side / width

        else:

            scale_factor = target_min_side / height

        # Calculate new dimensions

        new_width = int(width * scale_factor)

        new_height = int(height * scale_factor)

        # Resize the image

        input_image = input_image.resize((new_width, new_height), Image.Resampling.LANCZOS)

        input_image = input_image.convert('RGB')

        width, height = input_image.size

        print(width)

        width = (width // 8) * 8

        height = (height // 8) * 8

        input_image = input_image.resize((width, height), Image.Resampling.LANCZOS)

        n_prompt = "ugly, deformed, not detailed, blurred, too much blurred, motion blur, watermark, jpeg artifacts"

        for prompt in prompts:

            if use_LLM:
```

```
            gpt_prompt = f'"Below is a description of an image whose main topic is {desc}, change the description so that the main topic of the image and related objects do not change significantly, while the image description as a whole is significantly different from the input, this is required for augmentation of the dataset, return only the changed description, without further explanation. Here is the input image description: {prompts}'"

            prompt = self.gpt_prompter(gpt_prompt)

        for _ in range(num_samples):

            result = self.pipeline(prompt=prompt, image=input_image, negative_prompt=n_prompt, strength=denoising_strength)

            output_images.extend(result.images)

            result.images[0].save(f"{output_dir}\\" + image_path.split('\\')[-1])

    #del pipeline

    torch.cuda.empty_cache()

    return output_images

  def add_transparent_background_with_mask(self, imagePath, padding):

    """

    Add a transparent background around an image and create a mask for the new background.

    Parameters:

        image (PIL.Image): The input image.

        padding (int): The number of pixels to add as padding on each side.

    Returns:

        tuple: A tuple containing:

            - Image with a transparent background (PIL.Image).

            - Mask indicating the transparent background (PIL.Image).

    """

    # Ensure the image has an alpha channel

    image = Image.open(imagePath)

    if image.mode != 'RGBA':

        image = image.convert('RGBA')

    # Get the dimensions of the original image

    width, height = image.size

    # Calculate new dimensions

    new_width = width + 2 * padding

    new_height = height + 2 * padding

    # Create a new image with a transparent background

    transparent_image = Image.new('RGBA', (new_width, new_height), (0, 0, 0, 0))

    # Paste the original image in the center of the new transparent image

    transparent_image.paste(image, (padding, padding))

    # Create a mask for the transparent area (1 for non-transparent, 0 for transparent)
```

```
    mask = Image.new('L', (new_width, new_height), 0)
    mask.paste(255, (padding, padding, padding + width, padding + height))
    return transparent_image, mask
def expand_white_mask(self, mask, padding):
    """
    Expand a white mask on a black background by converting nearby black pixels to white.
    Parameters:
        mask (PIL.Image): The input mask (white on black background).
        padding (int): The number of pixels to expand the white area by.
    Returns:
        PIL.Image: The expanded mask.
    """
    # Ensure the mask is in grayscale mode (L mode)
    if mask.mode != 'L':
        mask = mask.convert('L')
    # Apply blur to create a gradient at the edges
    blurred_mask = mask.filter(ImageFilter.GaussianBlur(radius=padding))
    # Threshold to make the blurred area fully white
    expanded_mask = blurred_mask.point(lambda p: 255 if p > 0 else 0, mode='1').convert('L')
    return expanded_mask
def resize_with_padding(self, image, target_width, target_height):
    """Resize image preserving aspect ratio and adding padding if necessary"""
    # Get current dimensions
    old_width, old_height = image.size
    # Calculate scaling factors
    width_ratio = target_width / old_width
    height_ratio = target_height / old_height
    # Use the smaller ratio to ensure image fits within target dimensions
    ratio = min(width_ratio, height_ratio)
    # Calculate new dimensions
    new_width = int(old_width * ratio)
    new_height = int(old_height * ratio)
    # Resize image
    resized_image = image.resize((new_width, new_height), Image.Resampling.LANCZOS)
    # Create new image with padding
```

```
    new_image = Image.new('RGB', (target_width, target_height), (0, 0, 0))
    # Calculate padding
    left_padding = (target_width - new_width) // 2
    top_padding = (target_height - new_height) // 2
    # Paste resized image onto padded background
    new_image.paste(resized_image, (left_padding, top_padding))
    return new_image
def create_random_positioned_image(self, image, mask, target_width, target_height,
                                   min_scale=0.3, max_scale=1.5, fit_to_canvas=True):
    """
    Resize and position image randomly within target dimensions, ensuring it stays inside canvas if fit_to_canvas=True.
    Parameters:
        image (PIL.Image): The image to resize and position.
        mask (PIL.Image): The corresponding mask for the image.
        target_width (int): Width of the target canvas.
        target_height (int): Height of the target canvas.
        min_scale (float): Minimum scale factor.
        max_scale (float): Maximum scale factor.
        fit_to_canvas (bool): If True, ensures object remains within canvas.
    Returns:
        (PIL.Image, PIL.Image): The new image and mask positioned on a blank canvas.
    """
    bbox = mask.getbbox()
    if bbox:
        image = image.crop(bbox)
        mask = mask.crop(bbox)
    else:
        print("No white pixel found, skipping crop")
    # Get current dimensions
    old_width, old_height = image.size
    if fit_to_canvas:
        # Random scale that ensures the object fits entirely within the canvas
        max_scale_factor = min(target_width / old_width, target_height / old_height)
        scale_factor = random.uniform(min(min_scale, max_scale_factor), min(max_scale, max_scale_factor))
    else:
```

```
        # Random scale with no constraint to fit the canvas
        scale_factor = random.uniform(min_scale, max_scale)
    # Calculate scaled dimensions (maintaining aspect ratio)
    new_width = int(old_width * scale_factor)
    new_height = int(old_height * scale_factor)
    # Resize image and mask
    resized_image = image.resize((new_width, new_height), Image.Resampling.LANCZOS)
    resized_mask = mask.resize((new_width, new_height), Image.Resampling.LANCZOS)
    # Create new blank canvas
    new_image = Image.new('RGB', (target_width, target_height), (0, 0, 0))
    new_mask = Image.new('RGB', (target_width, target_height), (0, 0, 0))
    # Calculate random position
    if fit_to_canvas:
        # Ensure object remains within canvas bounds
        left_padding = random.randint(0, max(0, target_width - new_width))
        top_padding = random.randint(0, max(0, target_height - new_height))
    else:
        # Allow for random placement (may go outside bounds)
        max_left = max(0, target_width - new_width)
        max_top = max(0, target_height - new_height)
        left_padding = random.randint(0, max_left)
        top_padding = random.randint(0, max_top)
    # Paste resized images onto new backgrounds
    new_image.paste(resized_image, (left_padding, top_padding))
    new_mask.paste(resized_mask, (left_padding, top_padding))
    bbox = new_mask.getbbox()
    #del self.pipeline
    return new_image, new_mask, bbox
def inpaint_images_with_random_prompts(
    self,
    image_paths,
    output_dir,
    prompts,
    num_generations_per_image=1,
    width=512,
```

```
    height=512,
    background_remove=True,
    model_id="redstonehero/ReV_Animated_Inpainting",
    seed_value=99999,
    min_scale=1,  # Minimum scale factor for object size
    max_scale=1.8,  # Maximum scale factor for object size
    fit_to_canvas=False,
    expand_padding=0,
    save_mask=False,
    save_bbox=True
):
    torch.cuda.empty_cache()
    pipe = AutoPipelineForInpainting.from_pretrained(model_id, torch_dtype=torch.float16)
    pipe.enable_model_cpu_offload()
    if not output_dir:
        output_dir = '\\'.join(image_path.split('\\')[:-1])
    results = []
    # Make sure dimensions are multiples of 8
    width = (width // 8) * 8
    height = (height // 8) * 8
    for image_index, image_path in enumerate(image_paths):
        try:
            # Get image without background and its mask using delete_background
            if background_remove:
                no_bg_image, mask = self.delete_background(image_path, expand_padding)
            else:
                no_bg_image, mask = self.add_transparent_background_with_mask(image_path, 200)
            # Ensure both images are RGB
            no_bg_image = no_bg_image.convert('RGB')
            mask = mask.convert('RGB')
            for generation_index in range(num_generations_per_image):
                # Create randomly positioned and scaled version for each generation
                positioned_image, positioned_mask, bbox = self.create_random_positioned_image(
                    no_bg_image,
                    mask,
```

```
    width,
    height,
    min_scale,
    max_scale,
    fit_to_canvas
)
# Convert mask to correct format
mask_array = np.array(positioned_mask)
mask_array = np.mean(mask_array, axis=2)  # Convert to grayscale
mask_array = mask_array > 128  # Convert to binary
mask_image = Image.fromarray((~mask_array * 255).astype(np.uint8))  # Invert and convert to PIL Image
# Randomly select a prompt from the provided list
prompt = random.choice(prompts)
negative_prompt = "painting, digital art, 3d art, low quality"
# Set random seed if not specified
current_seed = seed_value if seed_value != -1 else random.randint(0, 999999)
# Generate the result using the inpainting pipeline
result = pipe(
    prompt=prompt,
    negative_prompt=negative_prompt,
    image=positioned_image,
    mask_image=mask_image,
    num_inference_steps=30,
    guidance_scale=6,
    strength=1,
    padding_mask_crop=32,
    generator=torch.manual_seed(current_seed)
).images[0]
image_id = str(random.randint(0, 999999))
output_file_name = f"{output_dir}\\image_{image_index}_gen_{image_id}.png"
result.save(output_file_name, "JPEG", quality=100)
with open(f"{output_dir}\\image_{image_index}_gen_{image_id}.txt", 'w') as fp:
    fp.write(prompt)
if save_bbox:
    if not os.path.exists(output_dir+'\\bbox'):
```

```
                os.makedirs(output_dir+'\\bbox')
            self.pil_bbox_to_yolo(bbox, width, height, txt_path=f"{output_dir}\\bbox\\image_{image_index}_gen_{image_id}.txt")
            img = result
            draw = ImageDraw.Draw(img)
            draw.rectangle(bbox, outline="red", width=2)
            path = f"{output_dir}\\bbox\\image_{image_index}_gen_{image_id}.png"
            img.save(path)
        if save_mask:
            positioned_mask.save(f"{output_dir}\\image_{image_index}_gen_{image_id}_mask.jpg", "JPEG", quality=100)
        results.append({
            "image_index": image_index,
            "generation_index": image_id,
            "output_file": output_file_name,
            "bbox": bbox,
            "prompt": prompt,
            "seed": current_seed
        })
    except Exception as e:
        print(f"Error processing image {image_path}: {str(e)}")
        continue
return results
def pil_bbox_to_yolo(self, bbox, image_width, image_height, class_index=0, txt_path=None):
    """
    Convert PIL bbox (left, upper, right, lower) to YOLO format.
    Parameters:
        bbox (list or tuple): Bounding box in PIL format [left, upper, right, lower].
        image_width (int): Width of the image.
        image_height (int): Height of the image.
        class_index (int): Class index for the object (default is 0).
    Returns:
        str: A string in YOLO format: "class_index center_x center_y width height".
    """
    left, upper, right, lower = bbox
    # Calculate YOLO format values
    bbox_width = right - left
```

```
        bbox_height = lower - upper
        center_x = left + bbox_width / 2
        center_y = upper + bbox_height / 2
        # Normalize values
        center_x /= image_width
        center_y /= image_height
        bbox_width /= image_width
        bbox_height /= image_height
        # Format as string
        yolo_line = f"{class_index} {center_x:.6f} {center_y:.6f} {bbox_width:.6f} {bbox_height:.6f}"
        if txt_path:
            with open(txt_path, 'a') as file:
                file.write(yolo_line + '\n')
        return yolo_line
```

Файл: QualityMetrics.py

```
import cv2
import pyiqa
import os
import shutil
from pathlib import Path
import cv2
import numpy as np
from PIL import Image, ImageEnhance
from src.diffusion_utils.SynteticImageGenerator import SyntheticImageGenerator
class QualityMetrics:
    def __init__(self, input_folder, output_folder, threshold=0.4, sd_gen=None):
        self.input_folder = Path(input_folder)
        self.output_folder = Path(output_folder)
        self.threshold = threshold
        self.output_folder.mkdir(parents=True, exist_ok=True)
        if sd_gen:
            self.sd_gen = sd_gen
        else:
            self.sd_gen = SyntheticImageGenerator(False)
```

```
    self.metrics = {
        'niqe': pyiqa.create_metric('niqe'),
        'brisque': pyiqa.create_metric('brisque'),
        'piqe': pyiqa.create_metric('piqe'),
        'liqe': pyiqa.create_metric('liqe', as_loss=False)
    }
def calculate_metrics(self, image_path):
    try:
        img = Image.open(image_path)
        width, height = img.size
        if width < 224 or height <224:
            if width < height:
                img = img.resize((224, int(224*(height/width))))
            else:
                img = img.resize((int(224*(width/height)), 224))
        metrics_values = {
            'niqe': float(self.metrics['niqe'](img).item()),
            'brisque': float(self.metrics['brisque'](img).item()),
            'piqe': float(self.metrics['piqe'](img).item()),
            'liqe': float(self.metrics['liqe'](img).item())
        }
        print(metrics_values)
        ranges = {
            'niqe': (0, 10),
            'brisque': (0, 100),
            'piqe': (0, 100),
            'liqe': (0, 10)
        }
        normalized_metrics = {}
        for metric, value in metrics_values.items():
            min_val, max_val = ranges[metric]
            if metric == 'liqe':
                normalized_metrics[metric] = 1*((value - min_val) / (max_val - min_val))
            else:
                normalized_metrics[metric] = 1 - (value - min_val) / (max_val - min_val)
```

```
        total_score = sum(normalized_metrics.values()) / len(normalized_metrics)
        metrics_values['total_score'] = total_score
        return metrics_values
    except Exception as e:
        print(f"Помилка при обробці {image_path}: {str(e)}")
        return None
def check_ML_task_quality(self, image_path, threshold1 = 0.5, threshold2 = 0.6):
    print(image_path)
    bbox_path = image_path.split('\\')
    bbox_path.insert(-1, 'bbox')
    txt_file_path = '\\'.join(bbox_path).split('.')[0] + '.txt'
    if os.path.exists(txt_file_path):
        with open(txt_file_path, "r") as file:
            for line in file:
                parts = line.strip().split()
                if len(parts) != 5:
                    raise ValueError(f"Неправильний формат рядка: {line}")
                cls, x_center, y_center, width, height = map(float, parts)
                Ml_class = []
                if width*height > threshold1:
                    Ml_class.append('Classification/Generation')
                if width*height < threshold2:
                    Ml_class.append('Detection')
                return Ml_class
    else:
        print('Need create bbox first')
        return None
def auto_enhance_image(self, image_path, output_path):
    """Автоматично покращує якість зображення"""
    def is_blurry(image, threshold=500):
        """Перевіряє, чи розмите зображення."""
        gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        laplacian_var = cv2.Laplacian(gray, cv2.CV_64F).var()
        print('laplac', laplacian_var)
        return laplacian_var > threshold
```

```
def is_overexposed(image, threshold=200):
    """Перевіряє, чи пересвічене зображення."""
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    brightness = hsv[:, :, 2].mean()
    print('brig', brightness)
    return brightness > threshold
def needs_jpeg_artifact_reduction(image, threshold=60):
    def zero_pad_img(Y):
        mx,my = Y.shape
        x = int(np.ceil((mx%8)/8)*8+mx//8*8)
        y = int(np.ceil((my%8)/8)*8+my//8*8)
        img = np.zeros((x,y), dtype = np.uint8)
        img[0:mx,0:my]=Y
        return img
    def smooth(y, box_pts):
        box = np.ones(box_pts)/box_pts
        y_smooth = np.convolve(y, box, mode='same')
        return y_smooth
    def get_DCT_coefs(img):
        w, h = img.shape
        dct_output = np.zeros((8,8,int(w*h/64)), dtype=int)
        ci=0
        for i in range(0, w, 8):
            for j in range(0, h, 8):
                image_part = img[i:i+8, j:j+8]
                dct_coefs = cv2.dct(image_part)
                dct_output[:, :, ci] = dct_coefs
                ci+=1
        return dct_output
    def get_DC_histogram(DC, N=2000):
        bins = range(0,N,1)
        hcount = np.histogram(DC, bins=bins, density=True)[0]
        return hcount
    def FFT_output(hcount):
        x = np.fft.fft(hcount.astype(float))
```

```
    x = np.fft.fftshift(x)
    x = 20*np.log10(np.abs(x))
    x = smooth(x, 3)
    return x
def find_quality_level(x, N):
    # Find distance between peaks in FFT
    # Use a reference magnitude
    ref = x[N//2-1]
    # Find locations that are 12 db lower than max magn.
    r=np.where((x-ref)>-12)
    local_maxs=[]
    for i in r[0]:
        if i<3 or i>x.shape[0]-3: # ignore the edges of FFT
            continue
        # Find local peaks that have snr greater than 1 dB with 3*2 pixel width
        if x[i]>x[i-1] and x[i]>x[i+1] and x[i]>x[i-3]+1 and x[i]>x[i+3]+1:
            local_maxs.append(i)
    # Use lookup table and get the compression level
    K_table = np.array([1, 3, 4, 6, 7, 9, 12, 19, 39, 100]) # This is the lookup table that I extracted using non-compressed image set
    if len(local_maxs)==1: # if no local max
        Q=100
        K=100
    else:
        Q=0
        # Select the most frequent distance between peaks
        d = np.bincount(np.diff(local_maxs)).argmax()
        K = d/N*100
        # Use lookup table, interpolate and find compression
        for i in range(9):
            if K>=K_table[i] and K<K_table[i+1]:
                Q=(i+1)*10+(K-K_table[i])/(K_table[i+1]-K_table[i])*10
    print("Q:", Q)
    return round(Q), local_maxs
img = Image.open(image)
# 2. Use luminance channel
```

```
    img = img.convert('YCbCr')
    Y = np.array(img)[:,:,0]
    # 3. Zero pad to multiple of 8 at each side
    img = zero_pad_img(Y)
    img = img.astype(np.float64)
    # 4. Get DCT coefs.
    dct_output = get_DCT_coefs(img)
    # 5. DC term
    DC = dct_output[0,0,:]
    # 6. DC histogram
    N=2000 # number of bins in histogram
    hcount=get_DC_histogram(DC, N)
    # 7. FFT of DC term
    x = FFT_output(hcount)
    # 8. Quality level
    Q, local_maxs = find_quality_level(x, N)
    return Q < threshold
def reduce_jpeg_artifacts(image):
    """Зменшує JPEG-артефакти."""
    denoised_image = cv2.fastNlMeansDenoisingColored(image, None, 10, 10, 7, 21)
    smoothed_image = cv2.bilateralFilter(denoised_image, d=9, sigmaColor=75, sigmaSpace=75)
    return smoothed_image
def sharpen_image(image):
    """Підвищує різкість зображення."""
    kernel = np.array([[0, -1, 0],
                       [-1, 5, -1],
                       [0, -1, 0]])
    return cv2.filter2D(image, -1, kernel)
# Завантаження зображення
image = cv2.imread(image_path)
print(type(image))
if image is None:
    print("Помилка: Не вдалося завантажити зображення.")
    return
# Покращення якості зображення
```

```
        print("Аналіз стану зображення...")
        pil_image = Image.fromarray(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
        if needs_jpeg_artifact_reduction(image_path):
        # JPEG-артефакти
            print("Зменшення JPEG-артефактів...")
            image = reduce_jpeg_artifacts(image)
        else:
            # Розмитість
            if is_blurry(image):
                print("Виявлено розмитість. Застосовується підвищення різкості...")
                image = sharpen_image(image)
        # Пересвічення
        if is_overexposed(image):
            print("Виявлено пересвічення. Регулюється контраст і яскравість...")
            pil_image = Image.fromarray(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
            enhancer = ImageEnhance.Contrast(pil_image)
            pil_image = enhancer.enhance(0.8)  # Зменшення контрасту
            enhancer = ImageEnhance.Brightness(pil_image)
            pil_image = enhancer.enhance(0.8)  # Зменшення яскравості
            image = cv2.cvtColor(np.array(pil_image), cv2.COLOR_RGB2BGR)
        # Збереження результату
        cv2.imwrite(output_path, image)
        print(f"Покращене зображення збережено у {output_path}")
    def cut_background(self, image_path):
        image_path = str(image_path)
        no_bg_image, mask = self.sd_gen.delete_background(image_path)
        img = Image.open(image_path)
        bbox = mask.getbbox()
        img.crop(tuple(bbox))
        img.save(image_path)
    def outpaint(self, image_path):
        with open(str(image_path).split('.')[0]+'.txt', 'r') as file:
            prompt = file.readline()
        self.sd_gen.inpaint_images_with_random_prompts([str(image_path)], '\\'.join(str(image_path).split('\\')[:-1]), [prompt], background_remove=False,
min_scale=0.6, max_scale=0.61)
```

```
def sd_enchance_image(self, image_path, out_path):
    with open(str(image_path).split('.')[0]+'.txt', 'r') as file:
        prompt = file.readline()
    self.sd_gen.img2img_quality_fix(str(image_path), '', [prompt], '\\'.join(str(image_path).split('\\')[:-1]))
def check_quality(self, ml_task = 'Classification/Generation', classic_qi=True, sd_qi = True, repeat_count = 2):
    results = {}
    image_extensions = ('.jpg', '.jpeg', '.png', '.bmp', '.tiff')
    for i in range(repeat_count):
        for image_path in self.input_folder.glob('*'):
            if image_path.suffix.lower() in image_extensions:
                print(f"Обробка {image_path.name}...")
                if ml_task in ['Classification/Generation', 'Detection']:
                    task = self.check_ML_task_quality(str(image_path))
                    print('\n')
                    print(task)
                    if task is not None and len(task) == 1 and ml_task != task[0]:
                        if ml_task == 'Classification/Generation':
                            self.cut_background(image_path)
                        elif ml_task == 'Detection':
                            self.outpaint(image_path)
                metrics = self.calculate_metrics(str(image_path))
                if metrics:
                    results[image_path.name] = metrics
                    print(self.threshold)
                    if metrics['total_score'] < self.threshold and i == repeat_count-1:
                        shutil.move(
                            str(image_path),
                            str(self.output_folder / image_path.name)
                        )
                        print(f"Переміщено {image_path.name} (оцінка: {metrics['total_score']:.2f})")
                    elif metrics['total_score'] < self.threshold:
                        print('improve')
                        if classic_qi:
                            self.auto_enhance_image(image_path, image_path)
                        if sd_qi:
```

```
                self.sd_enchance_image(image_path, image_path)
            else:
                print(f"{image_path.name} (оцінка: {metrics['total_score']:.2f})")
    return results
```

Файл: ImageScrapper.py

```
import sys
from pathlib import Path
base_path = Path(__file__).parent.parent.parent
sys.path.append(base_path.as_posix())
import time
import csv
import datetime
import hashlib
import io
import os
import shutil
import requests
from PIL import Image
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from selenium.webdriver.chrome.service import Service
import concurrent.futures
from src.scraping.search_engines import search_engine_factory
from src.config import ROOT_PATH
from src.similar_image.similar_image_remover import SimilarImagesRemover
from src.tags_analyze.gpt_analyze import TagsAnalyzer
#from googletrans import Translator
from time import sleep
import undetected_chromedriver as uc
from fake_useragent import UserAgent
class ImageScrapper():
    def filterImages(self, imagesPath, minResolution=32, tags=[], filterSimilar = True):
        for filename in os.listdir(imagesPath):
            if filename.lower().endswith(('.png', '.jpg', '.jpeg', '.bmp', '.gif', '.tiff')):
```

```
            image_path = os.path.join(imagesPath, filename)
            try:
                with Image.open(image_path) as img:
                    width, height = img.size  # Отримуємо розміри зображення з заголовка
                    if width < minResolution or height < minResolution:
                        os.remove(image_path)
                        os.remove(image_path.replace('.jpg', '.txt'))
                        print(f"Deleted {filename} (size: {width}x{height})")
            except Exception as e:
                print(f"Could not process {filename}: {e}")
    if filterSimilar:
        remover = SimilarImagesRemover(imagesPath)
        similar_images, num_images_removed = remover.remove_similar_images()
        print(f"Removed {num_images_removed} redundant images.")
    if tags:
        os.makedirs(imagesPath+'\\removed', exist_ok=True)
        for filename in os.listdir(imagesPath):
            if filename.endswith('.txt'):
                file_path = os.path.join(imagesPath, filename)
                with open(file_path, 'r', encoding='utf-8') as file:
                    content = file.read()
                    split_content = content.split(', ')
                if any(item in tags for item in split_content):
                    shutil.move(file_path, os.path.join(imagesPath+'\\removed', filename))
                    base_name = os.path.splitext(filename)[0]
                    for ext in ['.jpg', '.png', '.pdf']:
                        other_file_path = os.path.join(imagesPath, base_name + ext)
                        if os.path.exists(other_file_path):
                            shutil.move(other_file_path, os.path.join(imagesPath+'\\removed', base_name + ext))
```

Файл: ImageAnnotator.py

```
from typing import Generator, Iterable
from tagger.tagger.interrogator import Interrogator
from PIL import Image, ImageDraw
from pathlib import Path
import argparse
```

```
import os
import torch
import numpy as np
import matplotlib.pyplot as plt
from transformers import AutoProcessor, CLIPSegForImageSegmentation
from PIL import Image
from tagger.tagger.interrogators import interrogators
from tagger.tagger.captioners import captioners
class ImageAnnotator():
    def __init__(self, annotation_model, exclude_tags = [], mode = 'interrogate', threshold = 0.35, use_cpu = False):
        self.annotation_model = annotation_model
        self.interrogator = None
        if mode == 'interrogate':
            self.interrogator = interrogators[annotation_model]
            if use_cpu:
                self.interrogator.use_cpu()
        elif mode == 'caption':
            self.captioner = captioners[annotation_model]
        self.exclude_tags = exclude_tags
        self.threshold = threshold
    def autoCaption(self, imagesPath):
        model = self.captioner['model']
        processor = self.captioner['processor']
        model.eval()
        device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
        model.to(device)
        # Function to generate captions for an image
        def generate_caption(image_path):
            image = Image.open(image_path).convert("RGB")
            inputs = processor(images=image, return_tensors="pt").to(device)
            outputs = model.generate(**inputs)
            caption = processor.decode(outputs[0], skip_special_tokens=True)
            return caption
        # Dictionary to store image data and captions
        data = {'image_id': [], 'image_data': [], 'caption': []}
```

```
        # Process each image in the directory
    for image_filename in os.listdir(imagesPath):
        if image_filename.lower().endswith(('.png', '.jpg', '.jpeg')):
            image_path = os.path.join(imagesPath, image_filename)
            caption = generate_caption(image_path)
            data['image_id'].append(image_filename)
            data['caption'].append(caption)
            print('\n', caption)
            caption_path = '.'.join(image_path.split('.')[:-1]) + '.txt'
            with open(caption_path, 'w') as fp:
                fp.write(caption)
    print(data)
    return data
def auto_bbox(self, imagesPath: list, class_description: list, save_bbox_image=True):
    class_description.append('background')
    result = self.auto_segmentation(imagesPath, class_description)
    print(class_description)
    #print(labels, bboxs)
    if not os.path.exists(imagesPath+'\\bbox'):
        os.makedirs(imagesPath+'\\bbox')
    for i in result:
        print(i)
        if save_bbox_image:
            img = Image.open(i[0])
            draw = ImageDraw.Draw(img)
            for bbox in i[3]:
                draw.rectangle(bbox, outline="red", width=2)
            path = i[0].split('\\')
            path.insert(-1, 'bbox')
            path = '\\'.join(path)
            img.save(path)
        for j in range(len(i[2])):
            print('\\'.join(i[0].split('\\')[:-1]) + '\\bbox\\' + i[0].split('\\')[-1].split('.')[0] + '.txt')
            self.pil_bbox_to_yolo(i[3][j], i[1][0], i[1][1], class_description.index(i[2][j]), '\\'.join(i[0].split('\\')[:-1]) + '\\bbox\\' + i[0].split('\\')[-1].split('.')[0]
+ '.txt')
```

```
def pil_bbox_to_yolo(self, bbox, image_width, image_height, class_index=0, txt_path=None):
    """
    Convert PIL bbox (left, upper, right, lower) to YOLO format.
    Parameters:
        bbox (list or tuple): Bounding box in PIL format [left, upper, right, lower].
        image_width (int): Width of the image.
        image_height (int): Height of the image.
        class_index (int): Class index for the object (default is 0).
    Returns:
        str: A string in YOLO format: "class_index center_x center_y width height".
    """
    left, upper, right, lower = bbox
    # Calculate YOLO format values
    bbox_width = right - left
    bbox_height = lower - upper
    center_x = left + bbox_width / 2
    center_y = upper + bbox_height / 2
    # Normalize values
    center_x /= image_width
    center_y /= image_height
    bbox_width /= image_width
    bbox_height /= image_height
    # Format as string
    yolo_line = f"{class_index} {center_x:.6f} {center_y:.6f} {bbox_width:.6f} {bbox_height:.6f}"
    if txt_path:
        with open(txt_path, 'w') as file:
            file.write(yolo_line + '\n')
    return yolo_line
def auto_segmentation(self, imagesPath, segmentsDesc, modelName = "CIDAS/clipseg-rd64-refined", save_mask=False):
    processor = AutoProcessor.from_pretrained(modelName)
    model = CLIPSegForImageSegmentation.from_pretrained(modelName)
    result = []
    for image_filename in os.listdir(imagesPath):
        if image_filename.lower().endswith(('.png', '.jpg', '.jpeg')):
            image_path = os.path.join(imagesPath, image_filename)
```

```
            filename = os.path.splitext(image_filename)[0]
        else:
            continue
        image = Image.open(image_path)
        inputs = processor(text=segmentsDesc, images=[image] * len(segmentsDesc), padding=True, return_tensors="pt")
        outputs = model(**inputs)
        masks = outputs.logits
        mask_no = 0
        labels = []
        bboxs = []
        for mask in masks:
            segment = mask.detach().numpy()
            segment = (segment> 0).astype(np.uint8)
            segment_resized = Image.fromarray(segment).resize(image.size, resample=Image.NEAREST)
            segment_resized = np.array(segment_resized)
            segment_rescaled = (segment_resized * 255).astype(np.uint8)
            segment_image = Image.fromarray(segment_rescaled)
            if save_mask:
                segment_image.save(os.path.join(imagesPath, filename + '_' + segmentsDesc[mask_no] + '_mask' + '.png'))
            if segmentsDesc[mask_no] != 'background':
                labels.append(segmentsDesc[mask_no])
                bboxs.append(segment_image.getbbox())
            mask_no += 1
        result.append([image_path, image.size, labels, bboxs])
    return result
def autoAnnotate(self, imagesPath, model = None, confidenceThreshold = 0.35, batchSize = 5, overwrite = True, ext = '.txt'):
    root_path = Path(imagesPath)
    with open('taglist.txt', 'w') as tl:
        for image_path in self.explore_image_files(root_path):
            caption_path = image_path.parent / f'{image_path.stem}{ext}'
            if caption_path.is_file() and not overwrite:
                # skip if caption exists
                print('skip:', image_path)
                continue
            print('processing:', image_path)
```

```
            tags = self.image_interrogate(image_path, not True, self.parse_exclude_tags())
            tags_str = ', '.join(tags.keys())
            with open(caption_path, 'w') as fp:
                fp.write(tags_str)
                tl.write(str(caption_path) + ';' + tags_str + '\n')
    return self.merge_text_files(imagesPath)
def merge_text_files(self, folder_path, split_char_count = 10000):
    merged_text_lst = []
    merged_text = ''
    for filename in os.listdir(folder_path):
        if filename.endswith('.txt'):
            file_path = os.path.join(folder_path, filename)
            with open(file_path, 'r', encoding='utf-8') as file:
                merged_text += file.read() + '\n'  # Додаємо вміст файлу та перенос рядка
                if len(merged_text) > split_char_count:
                    merged_text_lst.append(merged_text)
                    #print(merged_text)
                    merged_text = ''
    #print(merged_text)
    if len(merged_text_lst) == 0:
        merged_text_lst.append(merged_text)
    return merged_text_lst
def parse_exclude_tags(self) -> set[str]:
    if self.exclude_tags is None:
        return set()
    tags = []
    for str in self.exclude_tags:
        for tag in str.split(','):
            tags.append(tag.strip())
    # reverse escape (nai tag to danbooru tag)
    reverse_escaped_tags = []
    for tag in tags:
        tag = tag.replace(' ', '_').replace('\(', '(').replace('\)', ')')
        reverse_escaped_tags.append(tag)
    return set([*tags, *reverse_escaped_tags])  # reduce duplicates
```

```
    def image_interrogate(self, image_path, tag_escape: bool, exclude_tags: Iterable[str]) -> dict[str, float]:
        """
        Predictions from a image path
        """
        im = Image.open(image_path)
        result = self.interrogator.interrogate(im)
        return Interrogator.postprocess_tags(
            result[1],
            threshold=self.threshold,
            escape_tag=tag_escape,
            replace_underscore=tag_escape,
            exclude_tags=exclude_tags)
    def explore_image_files(self, folder_path: Path) -> Generator[Path, None, None]:
        """
        Explore files by folder path
        """
        for path in folder_path.iterdir():
            if path.is_file() and path.suffix in ['.png', '.jpg', '.jpeg', '.webp']:
                yield path
            elif path.is_dir():
                yield from self.explore_image_files(path)
```

Файл: captioners.py

```
from transformers import CLIPProcessor, CLIPModel
from transformers import Blip2Processor, Blip2ForConditionalGeneration
captioners = {
    'clip': {'model': CLIPModel.from_pretrained("openai/clip-vit-base-patch32"), 'processor': CLIPProcessor.from_pretrained("openai/clip-vit-base-patch32")},
    'clip laion': {'model': CLIPModel.from_pretrained("laion/CLIP-ViT-H-14-laion2B-s32B-b79K"), 'processor': CLIPProcessor.from_pretrained("laion/CLIP-ViT-H-14-laion2B-s32B-b79K")},
    'blip': {'model': Blip2ForConditionalGeneration.from_pretrained("Salesforce/blip2-opt-2.7b"), 'processor': Blip2Processor.from_pretrained("Salesforce/blip2-opt-2.7b")}
}
```

Файл: gpt_analyze.py

```
import pandas as pd
```

```
import os

import shutil

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from time import sleep

import undetected_chromedriver as uc

from selenium.webdriver.support.ui import WebDriverWait

from fake_useragent import UserAgent

from selenium.webdriver.support import expected_conditions as EC

import getpass

class TagsAnalyzer:

    def __init__(self):

        pass

    def merge_text_files(self, folder_path):

        merged_text = ""

        for filename in os.listdir(folder_path):

            if filename.endswith('.txt'):

                file_path = os.path.join(folder_path, filename)

                with open(file_path, 'r', encoding='utf-8') as file:

                    merged_text += file.read() + '\n'  # Додаємо вміст файлу та перенос рядка

        return merged_text

    def get_list_for_delete(self, desc, tags, count = 10, use_account = True, login = None, password = None, max_tags_batch_count = 3):

        if len(tags) == 0:

            print('No tags')

            return None

        elif len(tags) == 1:

            prompts =  [f'''Here is the contents of the image tags file, the first column contains the path to the image, the second column contains the image tags:

            {tags}

            Identify {count} key tags (that appear in the text above and do not overlap in meaning) that clearly indicate that the image does not clearly match the description and that would definitely not appear in images that do match the description:

            "{desc}". Please note that not all images can be tagged correctly, as the tagging was performed by a neural network. Write the tags separated by commas in one line without explanation

            ''']

        else:

            prompts = []
```

```
        prompts.append(f"Here is the contents of the image tags file, the first column contains the path to the image, the second column contains the image tags:

        {tags[0]}

        Identify {count} key tags (that appear in the text above and do not overlap in meaning) that clearly indicate that the image does not clearly match the description and that would definitely not appear in images that do match the description:

        "{desc}". Please note that not all images can be tagged correctly, as the tagging was performed by a neural network. Write the tags separated by commas in one line without explanation

        ")

        for tags_batch in tags[1:]:

            prompts.append(f"Update the list you sent using this image tags file:

                {tags_batch}

                Still only return a comma-separated list of tags and write {count} key tags (that appear in the text above and do not overlap in meaning) that clearly indicate that the image does not clearly match the description and that would definitely not appear in images that do match the description:

                "{desc}". Please note that not all images can be tagged correctly, as the tagging was performed by a neural network. Write the tags separated by commas in one line without explanation as well:")

    op = webdriver.ChromeOptions()

    op.add_argument(f"user-agent={UserAgent.random}")

    op.add_argument("user-data-dir=./")

    op.add_experimental_option("detach", True)

    op.add_experimental_option("excludeSwitches", ["enable-logging"])

    driver = uc.Chrome(chrome_options=op)

    driver.get('https://chat.openai.com/')

    if use_account:

        if login is None or password is None:

            MAIL = input('Введіть логін (електронну пошту) до сервісу open.ai: ')#"dimon2000.1982@gmail.com"

            PASSWORD = getpass.getpass('Пароль: ')

        else:

            MAIL = login

            PASSWORD = password

        sleep(3)

        inputElements = driver.find_elements(By.CSS_SELECTOR, "[data-testid='login-button']")

        inputElements[0].click()

        sleep(5)

        mail = driver.find_elements(By.TAG_NAME,"input")[0]

        mail.send_keys(MAIL)

        mail.send_keys(Keys.ENTER)

        sleep(5)
```

```
    password= driver.find_element(By.ID,"password")
    password.send_keys(PASSWORD)
    password.send_keys(Keys.ENTER)
    sleep(3)
wait = WebDriverWait(driver, 10)
sleep(5)
#inputElements = driver.find_elements(By.CSS_SELECTOR, "p.placeholder")
#inputElement2 = driver.find_elements(By.CSS_SELECTOR, "div.ProseMirror")[0]
i = 0
#inputElement2.click()
for prompt in prompts:
    inputElements = driver.find_elements(By.CSS_SELECTOR, "p.placeholder")
    inputElement2 = driver.find_element(By.ID, "prompt-textarea")
    inputElement2.click()
    driver.execute_script("arguments[0].innerText = arguments[1];", inputElements[0], prompt)
    sleep(2)
    inputElement2.send_keys(Keys.ENTER)
    while True:
        try:
            stop_btn = driver.find_element(By.CSS_SELECTOR, "[data-testid='stop-button']")
            print("Елемент з data-testid='stop-button' знайдено!")
            sleep(5)
        except:
            print("Елемент з data-testid='stop-button' не знайдено.")
            break
    inputElements = driver.find_elements(By.TAG_NAME, "p")
    sleep(5)
results = []
for element in inputElements:
    results.append(element.text)
print(results)
# results = results[0-count:]
# print(results)
driver.quit()
return [tag.strip() for tag in results[-2].split(',')]
```

ДОДАТОК Д

Результати перевірки роботи на співпадіння

StrikePlagiarism.com

National Technical University of Ukraine Igor Sikorskyi Kyiv Politech Institute

Дата звіту 12/4/2024
Дата редагування 12/4/2024

Документ прийнятий

Звіт подібності

метадані

Заголовок
ІП-32мп_Українець_ПЗ

Науковий керівник / Експерт

Автор **Баклан І.В.**

підрозділ
ФІОТ, К-а інформатики та програмної інженерії

Тривога

У цьому розділі ви знайдете інформацію щодо текстових спотворень. Ці спотворення в тексті можуть говорити про МОЖЛИВІ маніпуляції в тексті. Спотворення в тексті можуть мати навмисний характер, але частіше характер технічних помилок при конвертації документа та його збереженні, тому ми рекомендуємо вам підходити до аналізу цього модуля відповідально. У разі виникнення запитань, просимо звертатися до нашої служби підтримки.

Заміна букв	ß	0
Інтервали	A→	0
Мікропробіли		3
Білі знаки	ß	0
Парафрази (SmartMarks)	a	14

Обсяг знайдених подібностей

Коефіцієнт подібності визначає, який відсоток тексту по відношенню до загального обсягу тексту було знайдено в різних джерелах. Зверніть увагу, що високі значення коефіцієнта не автоматично означають плагіат. Звіт має аналізувати компетентна / уповноважена особа.

0.75%
0.81%
КП 1

10	12478	96705
Довжина фрази для коефіцієнта подібності 2	Кількість слів	Кількість символів

Подібності за списком джерел

Нижче наведений список джерел. В цьому списку є джерела із різних баз даних. Колір тексту означає в якому джерелі він був знайдений. Ці джерела і значення Коефіцієнту Подібності не відображають прямого плагіату. Необхідно відкрити кожне джерело і проаналізувати зміст і правильність оформлення джерела.

10 найдовших фраз — Колір тексту

ПОРЯДКОВИЙ НОМЕР	НАЗВА ТА АДРЕСА ДЖЕРЕЛА URL (НАЗВА БАЗИ)	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	https://ela.kpi.ua/bitstreams/36b08709-1403-4fe0-b623-3d35bebb19a9/download	21	0.17 %
2	https://ela.kpi.ua/bitstream/123456789/25798/1/Linevych_magistr.pdf	19	0.15 %
3	https://ela.kpi.ua/bitstreams/4dec312a-f34d-401f-a6b5-63e385ca6a13/download	16	0.13 %
4	https://ela.kpi.ua/bitstreams/36b08709-1403-4fe0-b623-3d35bebb19a9/download	11	0.09 %

5	113-Левченко_Денис_Андрійович-2023 [illegible] 6/5/2024 Donetsk National University named after V. Stus (Donetsk National University named after V. Stus)	11	0.09 %
6	https://ela.kpi.ua/bitstreams/36b08709-1403-4fe0-b623-3d35bebb19a9/download	10	0.08 %
7	https://ela.kpi.ua/bitstreams/36b08709-1403-4fe0-b623-3d35bebb19a9/download	8	0.06 %
8	http://old2.donnuet.edu.ua/index.php/en/vstupnyku/reitynhovyi-spysok-abituriientiv-dlia-zdobuttia-stupenia-bakalavr-na-osnovi-okr-molodshyi-spetsialist/rabochye-prohrammi/kit/3441-18-1-osnovy-naukovyh-doslidgen-dfn/file	5	0.04 %

з домашньої бази даних (0.00 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)

з програми обміну базами даних (0.09 %)

ПОРЯДКОВИЙ НОМЕР	ЗАГОЛОВОК	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	113-Левченко_Денис_Андрійович-2023 [illegible] 6/5/2024 **Donetsk National University named after V. Stus (Donetsk National University named after V. Stus)**	**11 (1)**	**0.09 %**

з Інтернету (0.72 %)

ПОРЯДКОВИЙ НОМЕР	ДЖЕРЕЛО URL	КІЛЬКІСТЬ ІДЕНТИЧНИХ СЛІВ (ФРАГМЕНТІВ)	
1	**https://ela.kpi.ua/bitstreams/36b08709-1403-4fe0-b623-3d35bebb19a9/download**	**50 (4)**	**0.40 %**
2	**https://ela.kpi.ua/bitstream/123456789/25798/1/Linevych_magistr.pdf**	**19 (1)**	**0.15 %**
3	**https://ela.kpi.ua/bitstreams/4dec312a-f34d-401f-a6b5-63e385ca6a13/download**	**16 (1)**	**0.13 %**
4	http://old2.donnuet.edu.ua/index.php/en/vstupnyku/reitynhovyi-spysok-abituriientiv-dlia-zdobuttia-stupenia-bakalavr-na-osnovi-okr-molodshyi-spetsialist/rabochye-prohrammi/kit/3441-18-1-osnovy-naukovyh-doslidgen-dfn/file	5 (1)	0.04 %

Список прийнятих фрагментів

ПОРЯДКОВИЙ НОМЕР	ЗМІСТ	КІЛЬКІСТЬ ОДНАКОВИХ СЛІВ (ФРАГМЕНТІВ)
https://ela.kpi.ua/bitstreams/36b08709-1403-4fe0...		**8 (0.06%)**
1	розділу у підрозділі 2.1 було проведено моделювання та аналіз програмного забе...	8 (0.06%)