Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
“____ ” _____ 20__ р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
«Програмне забезпечення автоматизованого розподілення за вмістом графічної інформації»

Виконав здобувач вищої освіти
IV курсу, групи КІ-21-2
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Крюков К.Є.

Керівник проекту
к.т.н., доцент_____ Дрєєв О. М.
Рецензент _____

м. Кропивницький

Міністерство освіти і науки України
Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*
Кафедра *Кібербезпеки та програмного забезпечення*
Освітній ступінь *бакалавр*
Галузь знань ._12___"Інформаційні технології"
Спеціальність *123 Комп'ютерна інженерія*
Освітньо-професійна (освітньо-наукова) програма "Комп'ютерна інженерія"

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____О.А.Смірнов
« __ » _____ 20__ року

ЗАВДАННЯ НА ВИПУСКНУ КЛАВІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ

Крюкову Костянтину Євгенійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення автоматизованого розподілення за вмістом графічної інформації*
керівник роботи *Дрєєв Олександр Миколайович к.т.н. доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу *№__47-02_ від _17 .01 .2025 року*
2. Строк подання студентом роботи до захисту *23.05.2025 р.*
3. Мета та завдання кваліфікаційної бакалаврської роботи*: Метою розробки є програмне забезпечення автоматизованого розподілення за вмістом графічної інформації*
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Висновки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

6. Дата видачі завдання « 17 » 01 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів кваліфікаційної бакалаврської роботи	Примітка
1.	Аналіз існуючих систем	10.03.2025 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2025 р.	
3.	Розробка моделі компонента	20.03.2025 р.	
4.	Розробка структур даних	25.03.2025 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2025 р.	
6.	Програмування алгоритмів	10.04.2025 р.	
7.	Оформлення ПЗ	17.04.2025 р.	
8.	Попередній захист роботи	22.05.2025 р.	

Дата видачі завдання
17 січня 2025 р.

Студент _____ Крюков К. Є.
(підпис)

Завдання прийнято до виконання
17 січня 2025 р.

Керівник роботи _____ Дрєєв О. М.
(підпис)

АНОТАЦІЯ

Крюков К. Є. Програмне забезпечення автоматизованого розподілення за вмістом графічної інформації. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

У даній кваліфікаційній бакалаврській роботі розроблено програмне забезпечення, яке призначено для автоматизованого розподілення за вмістом графічної інформації

Метою роботи є створення системи автоматизованого розподілення за вмістом графічної інформації

Результат роботи – програмна реалізація для пошуку та організації зображень на основі їх вмісту.

В процесі роботи над реалізацією системи виконано дослідження існуючих методів, алгоритмів та програмних засобів. Розроблено та реалізовано власне програмне забезпечення, здійснено опис всіх його компонентів.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ архітектури IBM PC з ОС Windows 10/11.

Програму розроблено на мовах програмування Python.

Ключові слова: база даних, СУБД, масштабованість

ABSTRACT

Kriukov K. E. Software for automated distribution by content of graphic information. 123 Cybersecurity. Central Ukrainian National Technical University. Kropyvnytskyi 2025.

In this bachelor's qualification work, software has been developed, which is intended for automated distribution by content of graphic information

The purpose of the work is to create a system of automated distribution by content of graphic information

The result of the work is a software implementation for searching and organizing images based on their content.

In the process of working on the implementation of the system, a study of existing methods, algorithms and software tools was carried out. The own software was developed and implemented, and all its components were described.

A convenient user interface was developed. Instructions for working with the software tools are provided.

The program can be used on IBM PC architecture computers with Windows 10/11.

The program was developed in the Python programming language.

Keywords: database, DBMS, scalability

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

YOLO (You Only Look Once) – алгоритм глибинного навчання, призначений для розпізнавання об’єктів на зображеннях та у відео, що працює в режимі реального часу.

SSD(Single Shot Detector) – алгоритм глибинного навчання, призначений для виявлення об’єктів на зображеннях. Його особливістю є одноступенева (single-shot) детекція – він визначає координати об’єктів та їхні класи одним проходом через нейронну мережу.

Штучна нейрона мережа (ШНМ) – математична конструкція, що черпає натхнення з організації та функціонування людського мозку. Вона утворена значною кількістю взаємозалежних базових компонентів – нейронів, які синхронізовано виконують операції з інофрмацією.

СУБД (Системи управління базами даних) - набір взаємопов’язаних даних та програм для доступу до цих даних. Надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних.

MySQL - поширена безкоштовна реляційна СУБД. Вона підтримує SQL (Structured Query Language) що робить її надійним вибором для розробки веб-проєктів та веб-сайтів.

PostgreSQL - потужна реляційна СУБД з підтримкою розширених функцій, таких як JSON, транзакції, геоінформаційні дані (PostGIS) та ін.

MongoDB - документоорієнтована база даних, яка зберігає дані у форматі JSON.

Cassandra - розподілена база даних, орієнтована на швидкість і масштабність, що підходить для великих обсягів даних.

OpenCV - безкоштовна бібліотека комп'ютерного зору, яка дозволяє

маніпулювати зображеннями та відео для виконання різноманітних завдань - від відображення зображення з веб-камери до потенційного навчання робота розпізнаванню реальних об'єктів.

PIL - покращена версія оригінальної бібліотеки PIL, яка забезпечує прості засоби для роботи з зображеннями в Python.

DL (діафрагментне моделювання) - підмножини штучного інтелекту, натхненної структурою та процесами людського мозку для аналізу та інтерпретації складних шаблонів даних.

Convolutional Neural Network (CNN) – різновид штучної нейронної мережі, спеціалізованої для ефективної роботи та аналізу даних з мережевою структурою, особливо зображень.

Recurrent Neural Network (RNN) – тип штучної нейронної мережі, який вирізняється наявністю петель в архітектурі, що дають змогу утримувати дані про попередні компоненти послідовності.

Long Shor-Term Memory (LSTM) – різновид рекурентної нейронної мережі, що був спеціально розроблений для ефективного утримування і застосування тривалих взаємозалежностей у даних, що подаються послідовно.

ВСТУП

В сучасному інформаційному середовищі графічні дані відіграють центральну роль у накопиченні, розповсюдженні та розумінні інформації. Стрімке збільшення обсягів візуального контенту, особливо у галузях медіа, освіти, наукових досліджень, промисловості та соціальних мереж, висуває вимоги до ефективних інструментів для його обробки та упорядкування. Насамперед виникає потреба у засобах, здатних автоматично аналізувати, класифікувати та впорядкувати зображення за їхнім змістом.

Традиційні методи визначення категорії графічних даних переважно включають людський фактор, що відчутно гальмує темп та ускладнює розширення таких процедур. Зважаючи на це, програмні рішення, які втілюють автоматичне сортування зображень відповідно до їхнього наповнення, набувають особливої значущості. Це рішення дозволяє прискорити обробку візуальної інформації, забезпечити її організованість, а також запропонувати користувачеві інтуїтивно зрозумілий інтерфейс для взаємодії із системою.

Програмне забезпечення автоматизованого розподілення за вмістом графічної інформації слугує для відшукання та впорядкування графічних файлів, спираючись на їх внутрішній зміст, а не метадані на кшталт назв чи позначок. Має кілька ключових аспектів щодо програмного забезпечення:

Основні компоненти:

1) Аналіз зображення – використовується для аналізу вмісту зображень, вмикаючи кольори, текстуру та форми. Застосовуються алгоритми такі, як гістограми кольорів, фільтри Габора та локальні ознаки.

2) Інтексація – після аналізу зображень, дані зберігають в базі даних у формі, яка дозволяє швидко їх шукати. Зазвичай використовується структура даних, такі як КД – дерева або локальні індекси.

3) Пошук – користувач може завантажити зображення або обрати його з

каталогу і система знаходить схожі зображення на основі вмісту. Реалізуються різні методики схожості.

4) Відображення результатів – система надає результати у формі ескізів або повноформатних зображень, з можливістю фільтрації за різними критеріями.

Мета й завдання дослідження. Метою роботи є програмне забезпечення автоматизованого розподілення за вмістом графічної інформації.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно використовувати для пошуку та організації зображень.

Отже, розробка та впровадження програмного забезпечення автоматизованого розподілення за вмістом графічної інформації є актуальною задачею, яка вимагає постійного вдосконалення і розробки нових рішень та вирішувалася у цій кваліфікаційній роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система програмного забезпечення автоматизованого розподілення може бути адаптована під різноманітні потреби. Її застосування визначається функціональними можливостями та конкретним середовищем. Для прикладу можемо розглянути систему комп'ютерного зору.

Система комп'ютерного зору має такі призначення:

- Обробка зображень.
- Автоматизація.
- Аналіз даних.

Обробка зображень проводить аналіз та зображення для виявлення об'єктів, розпізнавання облич, або сегментації зображень. Включає алгоритми машинного навчання, зокрема нейронні мережі, такі як YOLO (You Only Look Once) чи SSD(Single Shot Detector).

Автоматизація використовує технології для автоматизації прогресів, таких як контроль якості на виробництві чи моніторингу системи.

Аналіз даних – збирає та проводить аналіз даних для отримання корисної інформації, яка може використовуватись для прийняття рішень.

1.2 Область застосування

Програмне забезпечення автоматизованого розподілення за вмістом графічної інформації застосовується:

1) Промислове виробництво – У сучасних умовах розвитку цифрових технології автоматизовані системи обробки графічної інформації відіграють важливу роль у забезпечені ефективності виробничих процесів. Програмне

забезпечення, спроможне автоматично розпізнавати, аналізувати та класифікувати зображення, широко використовується в різних галузях промисловості для вирішення задач контролю якості, сортування продукції, моніторингу обладнання та безпеки на виробництві. Контроль якості продукції є одним із найпоширеніших прикладів застосування автоматизованої візуальної контролю якості. Камери з надзвичайно чітким зображенням інтегровані у виробничі процеси, знімають готову продукцію, а спеціалізоване програмне забезпечення проводить аналіз цих знімків за заданими параметрами:

- Перевірка геометричних розмірів;
- Виявлення дефектів фарбування, швів, тріщин;
- Перевірка правильності маркування чи штрих-кодів.

2) Державне управління та безпека – автоматизовані системи аналізу графічної інформації контролюють за громадською системою, ефективне управління інфраструктури та захист стратегічних об’єктів. Завдяки здатності розпізнавати на картинках об’єкти, контексти й операції, ці системи гарантують високу швидкість реагування, мінімалізацію впливу людського чинника та ухвалення рішень на основі актуальних даних. Системи автоматичного аналізу відео з камер спостережень можуть виявляти скупчення людей, підозрілу поведінку, розпізнавати обличчя осіб, що перебувають в розшуку, ідентифікувати потенційні загрози (зброя, вибухівка, залишені речі), виявляти випадки агресії, бійок та нещасних випадків.

3) Реклама та маркетинг – дозволяє аналізувати поведінку споживачів та їх інтереси, що допомагає точно налаштувати рекламу на конкретні цільові групи. Програмне забезпечення може автоматично класифікувати зображення для побудови профілів користувачів. На основі зібраних даних можна автоматично сегментувати аудиторію на групи, що дозволяє точніше націлювати рекламу. Може включати адаптацію візуальних елементів, кольорів та повідомлень. Рекламні компанії можуть бути адаптовані в реальному часі в залежності від реакції користувачів. Автоматичне виявлення небажаного чи неприйнятного

контенту також корисне в рекламних системах.

4) Освіта та наука – програмні рішення, що автоматизовано розпізнають, сортують та розподіляють зображення, набувають дедалі більшого значення. Вони суттєво покращують організацію навчального процесу, дозволяють ефективніше працювати з масштабними наборами інформації та стимулюють розробку новаторських методик у науковій діяльності. Дозволяє зробити навчальний процес інтерактивним і персоналізованим, покращує керування навчальним матеріалами, прискорює проведення наукових досліджень, зберігає та систематизує візуальні дані для майбутніх поколінь.

5) Соціальні мережі – дозволяє планувати публікації графічного контенту заздалегіть. Наприклад можна створити пост із зображенням або відео та встановити дату та час, коли він з'явиться в стрічці новин. Це зручно для підтримки активності профілю без постійного ручного втручання. Може адаптувати графіку в залежності від індивідуальних користувачів.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми кваліфікаційної бакалаврської роботи

Найкращі фотоорганізатори зі штучним інтелектом пропонують складні опції, які задовольняють потреби аматорів та професіоналів. Ці інструменти зазвичай інтегрують розширені здатності штучного інтелекту для покращення функцій пошуку, дозволяючи користувачам легко знаходити зображення за допомогою смарт-тегів та персональних фільтрів. Крім того, можливості редагування на основі штучного інтелекту дають змогу автоматично вдосконалювати світлини, гарантуючи найкращу якість зображення з мінімальним ручним втручанням.

Крім того, технологія штучного інтелекту допомагає розвантажувати цифрові бібліотеки шляхом визначення дублікатів і непотрібних зображень, що спрощує керування медіафайлами та звільняє місце для зберігання. Функції конфіденційності також є важливим аспектом цих органайзерів із надійними налаштуваннями, які дозволяють користувачам контролювати, хто переглядає їхні медіафайли. Освітні можливості, які надають ці платформи, наприклад навчальні посібники та сесії експертів, використовують ШІ для адаптації навчального досвіду, роблячи його більш інтерактивним і корисним.

Організатори світлин, що базуються на штучному інтелекті, застосовують алгоритми машинного навчання для автоматичного позначення, сортування та поділу світлин за категоріями на основі їхнього змісту, дати, місця, де вони зроблені, та інших важливих чинників. Ці розумні інструменти набувають великого значення в добу цифрових технологій, надаючи нам можливість швидко знаходити потрібні світлини та з легкістю ними ділитися.

Найкращі фотоорганізатори на основі штучного інтелекту:

Mylio Photos - вдосконалена програма для керування медіафайлами, яка використовує штучний інтелект для значного покращення організації, пошуку та редагування цифрових спогадів. Додаток пропонує розумні теги штучного інтелекту та складну функцію пошуку, що дозволяє користувачам ефективно знаходити зображення на різних пристроях без ручного пошуку. AI розпізнає вміст і контекст, спрощуючи процес виявлення за допомогою інтуїтивно зрозумілого інтерфейсу, який включає перегляди календаря, спеціальні теги та географічні дані. Інструменти редагування в Mylio Photos розширені штучним інтелектом, автоматично регулюють колір, покращують якість зображення та тонко налаштовують такі елементи, як баланс білого та експозиція. Користувачі можуть створювати спеціальні налаштування за допомогою цих інтелектуальних функцій, забезпечуючи оптимальне представлення фотографій із мінімальним ручним втручанням.

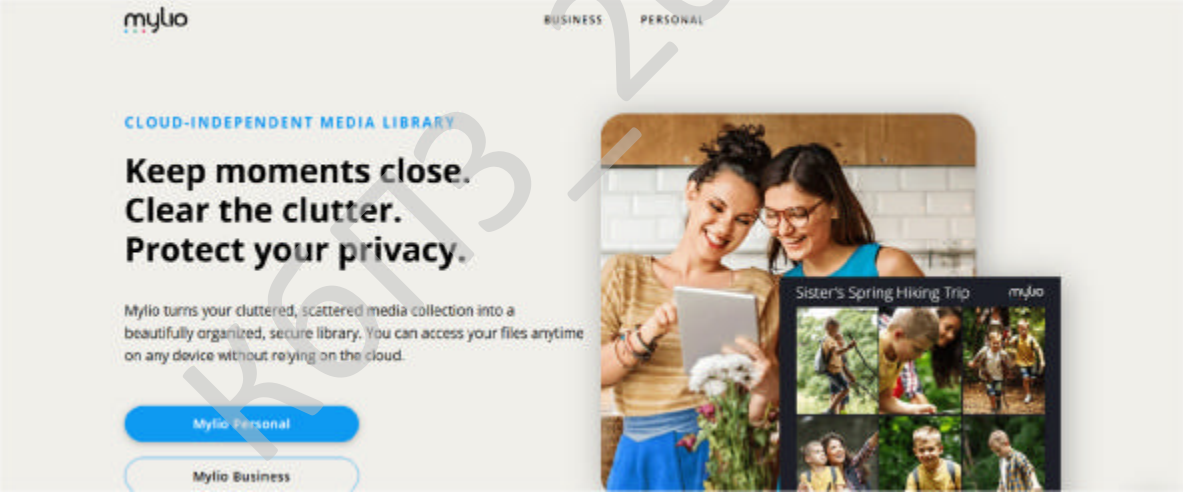

Рисунок 2.1 – Mylio Photos

Крім того, керовані штучним інтелектом інструменти очищення DeClutter і DeDuplication допомагають користувачам керувати своїми колекціями фотографій, визначаючи та видаляючи дублікати та непотрібний безлад. Це не тільки звільняє місце для зберігання, але й гарантує, що зберігаються лише найрелевантніші фотографії. Mylio Photos також інтегрує різноманітні пристрої

зберігання та облікові записи в єдине рішення, забезпечуючи уніфіковане керування медіафайлами на кількох платформах без спеціалізації на сховищі. Ця інтелектуальна інтеграція дозволяє користувачам підтримувати комплексне уявлення про свої медіа-колекції, підвищуючи доступність і керування.

Програма приділяє особливу увагу конфіденційності та контролю, надаючи користувачам можливість створювати окремі простори для обміну медіафайлами з різними групами, такими як родина, друзі чи колеги. Ці простори захищені паролями та дозволами, що гарантує, що користувачі мають повний контроль над своєю конфіденційністю.

-**Управління медіафайлами, розширене штучним інтелектом**: Mylio Photos використовує штучний інтелект для спрощення пошуку та організації фотографій на різних пристроях, використовуючи розумні теги та складну панель пошуку для зручності доступу.

- Інтелектуальні можливості редагування: програма містить інструменти на основі штучного інтелекту, які автоматично регулюють параметри фотографій, як-от колір і баланс білого, що дозволяє покращувати зображення високої якості.

- Ефективне очищення за допомогою ШІ: інструменти DeClutter і DeDuplication використовують штучний інтелект для виявлення та видалення непотрібних файлів і дублікатів, допомагаючи ефективно керувати сховищем і звільняти його.

- Розумна інтеграція рішень для зберігання: На відміну від традиційних систем зберігання, Mylio Photos об’єднує різні пристрої зберігання та облікові записи в уніфіковане рішення для керування, підвищуючи доступність і організацію.

- Конфіденційність і можливості навчання: Платформа забезпечує конфіденційність користувачів за допомогою настроюваних налаштувань спільного доступу та пропонує навчання за допомогою штучного інтелекту через вебінари та персоналізовані рекомендації експертів.

Одним із найкращих на ринку фотоорганізаторів на основі штучного

інтелекту є **PhotoPrism**, програма, яка допомагає користувачам ефективніше й ефективніше керувати та впорядковувати колекцію цифрових фотографій. Це дозволяє сортувати, позначати та класифікувати ваші фотографії на основі певних критеріїв, таких як дата, місце та вміст. PhotoPrism також пропонує розширені можливості пошуку, які полегшують пошук конкретних фотографій у вашій колекції.

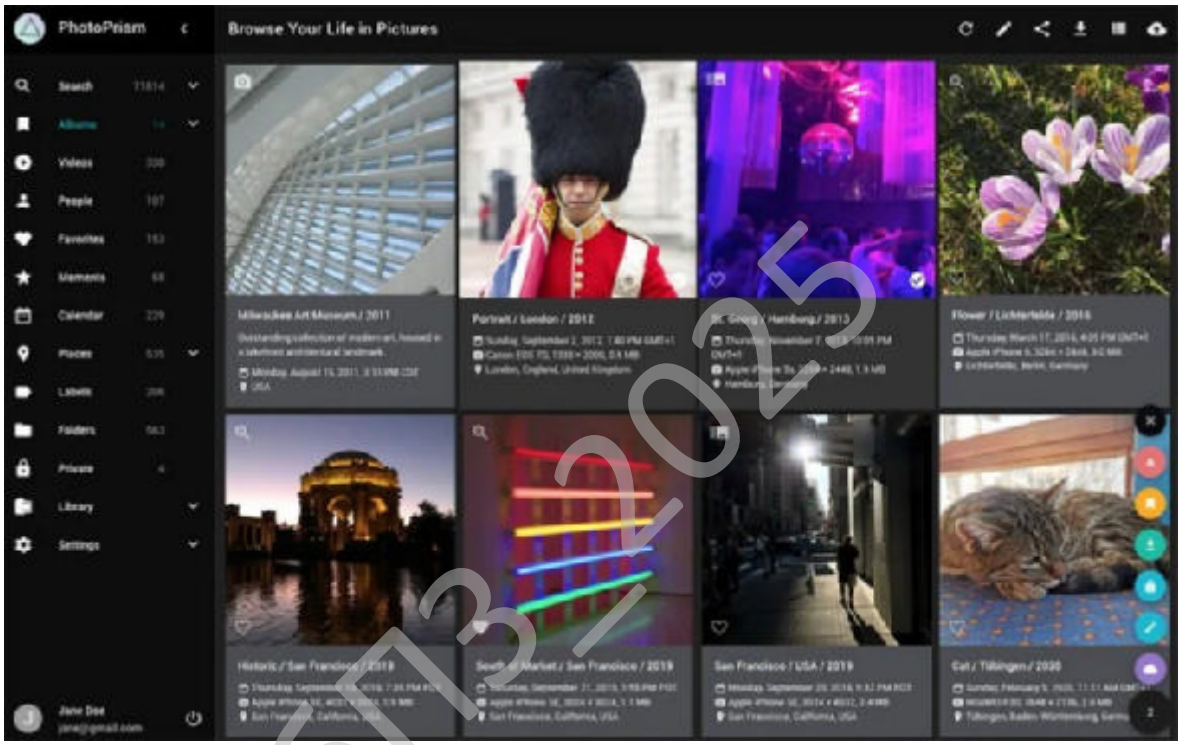

Рисунок 2.2 – PhotoPrism

Однією з головних функцій програми є використання штучного інтелекту для автоматичного позначення та класифікації фотографій на основі їх вмісту. Він покладається на машинне навчання для аналізу кожної фотографії та ідентифікації об’єктів, людей та різноманітних інших деталей.

Крім цих функцій, він також може синхронізуватися з різними хмарними службами зберігання.

Основні функції PhotoPrism:

-Підтримує різні типи файлів (RAW, JPEG, PNG).

-Розширені інструменти редагування, як-от обрізання та зміна розміру.

-Автоматичне виявлення та видалення дублікатів..

-Настроювані параметри спільного доступу.

Excire — це ще один потужний організатор фотографій зі штучним інтелектом, який допомагає сортувати вашу бібліотеку цифрових фотографій. Ви можете використовувати цей інструмент, щоб знаходити та впорядковувати свої фотографії на основі таких критеріїв, як предмет, розташування та колір. Excire використовує вдосконалені алгоритми машинного навчання для аналізу фотографій і автоматичного позначення їх тегами на основі їх вмісту, що полегшує їх пошук пізніше. Він також пропонує розширені інструменти пошуку з певними критеріями, або ви можете створити власні критерії для задоволення конкретних вимог.

Рисунок 2.3 - Excire

Однією з унікальних особливостей Excire є те, що він легко інтегрується з Adobe Lightroom, що полегшує керування та впорядкування бібліотеки фотографій із програми.

Основні функції Excire:

-Розпізнавання зображень на основі AI.

-Настроюваний пошук.

-Інтеграція з Adobe Lightroom..

-Дружній до користувача інтерфейс.

QuMagie — це інтелектуальний органайзер фотографій, який використовує AI, щоб допомогти вам сортувати, класифікувати та отримувати цифрові фотографії. Це дає змогу автоматично впорядковувати та позначати тегами вашу бібліотеку фотографій. Алгоритми штучного інтелекту, застосовані інструментом, аналізують ваші фотографії та автоматично розпізнають і позначають людей, об’єкти, сцени та місця, що полегшує пошук фотографій за різними критеріями.

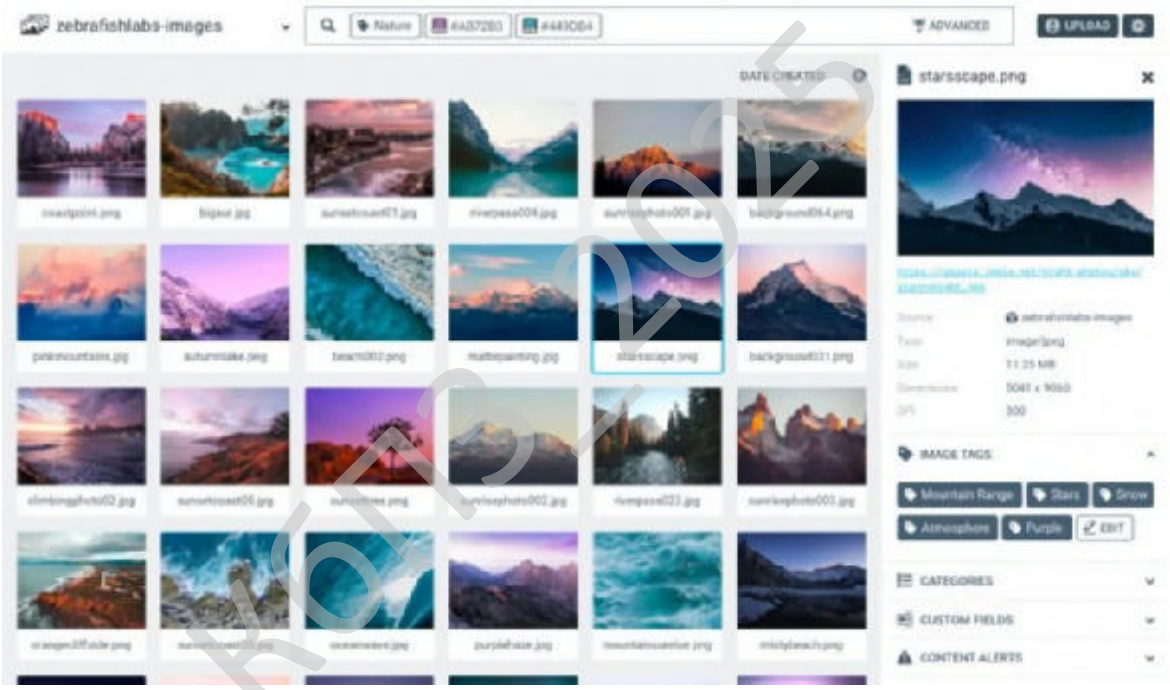

Рисунок 2.4 – QuMagie

QuMagie також пропонує інтелектуальне створення альбомів, де він автоматично групує ваші фотографії в альбоми на основі людей, місць, дат, подій та інших критеріїв. Ви також можете створювати спеціальні альбоми за власними критеріями пошуку, перш ніж ділитися ними з іншими. Окрім цих функцій, ви також можете видаляти дублікати та працювати в автономному режимі.

QuMagie можна порівняти з різними платформами, такими як Windows,

macOS і Linux.

Основні функції QuMagie:

- Розширене розпізнавання зображень.
- Розумне створення альбому.
- Видалення дублікатів.
- Автономний режим.

Imgix — це хмарна платформа для обробки та доставки зображень, яка також пропонує розширені функції організації фотографій на основі штучного інтелекту. За допомогою Imgix ви можете швидко й легко впорядкувати свою бібліотеку фотографій і зробити її більш зручною для пошуку та доступністю. Інструментом користуються понад 2,000 компаній по всьому світу.

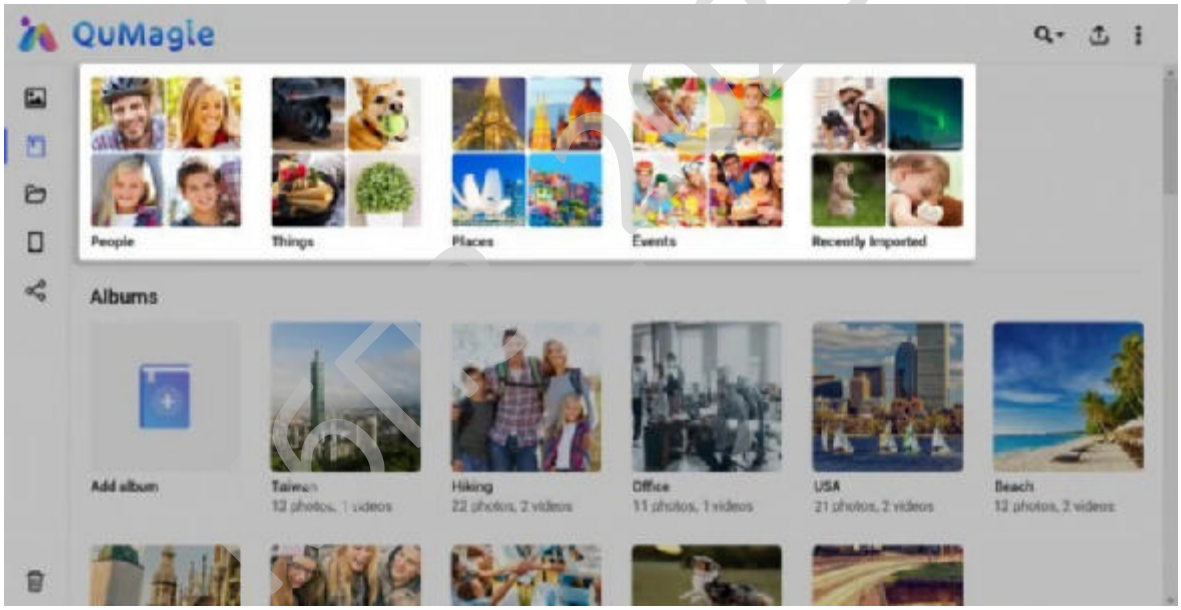

Рисунок 2.5 – Imgix

Потужна технологія обробки зображень Imgix дає змогу змінювати розмір, обрізати та маніпулювати зображеннями в режимі реального часу, що полегшує оптимізацію фотографій для будь-якого екрана чи пристрою. Інструмент також має систему тегів на основі штучного інтелекту, яка може автоматично позначати ваші зображення тегами на основі ряду критеріїв, таких як вміст і колір. Ви також можете створювати спеціальні теги.

Деякі з інших пропозицій включають інтелектуальну доставку вмісту, коли інструмент автоматично надає найкраще можливе зображення для кожного користувача на основі його пристрою та мережі, а також створення динамічних зображень, що дає змогу створювати динамічні зображення, які змінюються в режимі реального часу залежно від користувача. поведінка та інші змінні.

Основні функції Imgix:

-Настроювані теги зображень.

-Розумна доставка контенту.

-Створення динамічного зображення.

-Інтеграція API з широким спектром API.

Monument — це розумний пристрій для зберігання та організації фотографій, який пропонує різноманітні корисні функції для щоденного використання. Після налаштування він автоматично створює резервні копії файлів із комп'ютера, смартфонів, SD-карт і жорстких дисків.

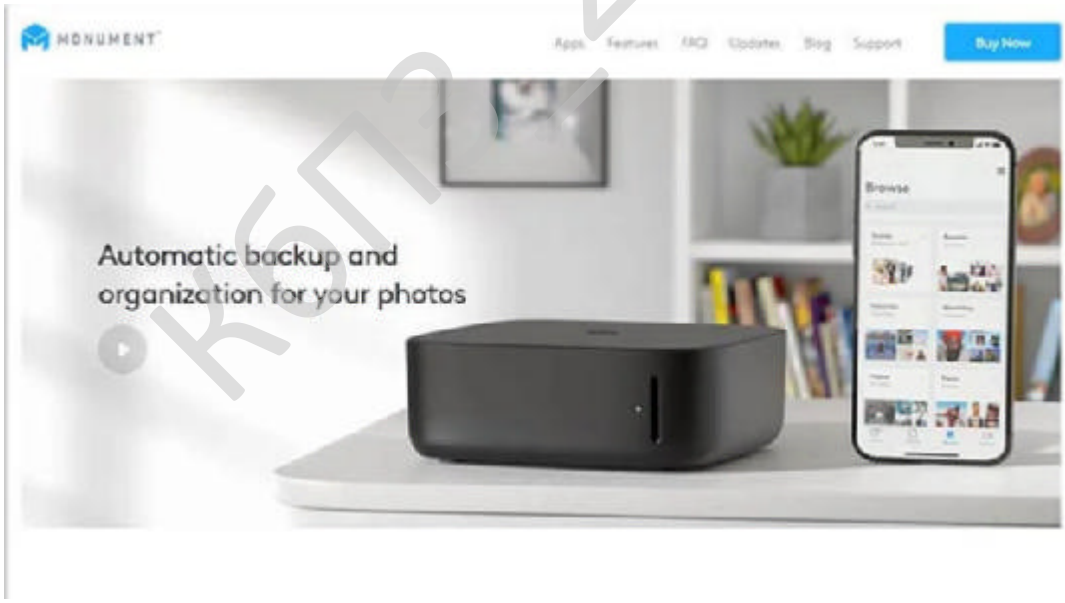

Рисунок 2.6 – Monument

Використовуючи штучний інтелект, Monument сортує ваші файли за датою, місцезнаходженням, камерою, особою та пейзажем, роблячи їх легко доступними та доступними для пошуку. Monument також підтримує до п'яти

окремих облікових записів, забезпечуючи безпеку та доступність даних кожного з однієї платформи. Крім того, він дозволяє створювати спільні альбоми з іншими користувачами.

Основні функції Monument:

-Інтелектуальна організація фотографій.

-Автоматичне резервне копіювання.

-Спільні альбоми.

-Підтримка кількох пристроїв.

Edenphotos — це рішення для зберігання та організації зображень на базі штучного інтелекту, яке надає користувачам інтуїтивно зрозумілий та ефективний спосіб керування цифровими фотографіями. Платформа автоматично позначає фотографії за допомогою передової технології розпізнавання зображень і класифікує їх за відповідними темами та ситуаціями. Це гарантує, що користувачі можуть легко знаходити та отримувати доступ до своїх фотографій без необхідності ручного сортування.

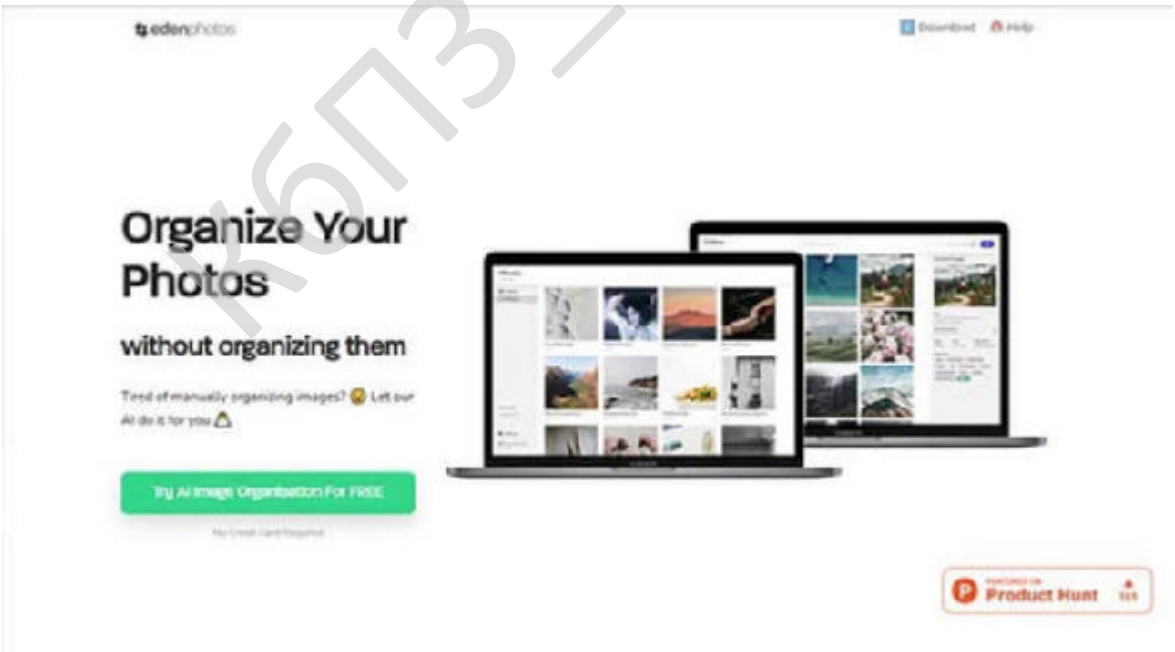

Рисунок 2.7 – Edenphotos

Однією з ключових переваг edenphotos є його хмарна система зберігання,

яка дозволяє користувачам безпечно зберігати свої фотографії та отримувати до них доступ з будь-якого місця та в будь-який час. Платформа також дуже гнучка, підтримує майже всі формати зображень, у тому числі ті, які використовуються користувачами Canon. Це робить його ідеальним рішенням для фотографів і любителів, яким потрібно керувати великими колекціями фотографій.

Основні функції edenphotos:

-Хмарне сховище.

-Підтримка майже всіх форматів зображень.

-Зручний інтерфейс з можливостями пошуку та фільтрації.

-Інструменти редагування та спільного використання.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Відзначені системи, технології, архітектури та програмні рішення забезпечують потужну екосистему для роботи з даними, обробки зображень, тексту та розробки складних програмних продуктів.

1) **Системи**.

Система управління базами даних (СУБД) – набір взаємопов'язаних даних та програм для доступу до цих даних. Надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних. СУБД ділиться на 2 типи – реляційні та нереляційні.

Реляційні СУБД – це MySQL та PostgreSQL.

MySQL – поширена безкоштовна реляційна СУБД. Вона підтримує SQL (Structured Query Language) що робить її надійним вибором для розробки веб-проєктів та веб-сайтів. Її популярність серед комерційних продуктів обумовлена простотою інтеграції та наявністю великої спільноти користувачів.

PostgreSQL – потужна реляційна СУБД з підтримкою розширених функцій, таких як JSON, транзакції, геоінформаційні дані (PostGIS) та ін. Є

ідеальним вибором для створення додатків, які вимагають гнучкості та масштабності.

Нереляційні СУБД – це MongoDB та Cassandra.

MongoDB – Документоорієнтована база даних, яка зберігає дані у форматі JSON. Вона є оптимальним вибором для проектів, де структура даних може бути нестабільною та потребує динамічної адаптації.

Cassandra – розподілена база даних, орієнтована на швидкість і масштабність, що підходить для великих обсягів даних. Гарантує високу доступність і відмовостійкість, що робить її оптимальним вибором для реальних часових застосувань.

Системи обробки зображень надають ефективні інструменти для вирішення широкого кола задач у галузі комп'ютерного зору. З розвитком технологій, зокрема, глибокого навчання, потенціал аналізу зображень невпинно зростає. Має відомі бібліотеки – OpenCV та PIL (Pillow).

OpenCV (Open Source Computer Vision) - безкоштовна бібліотека комп'ютерного зору, яка дозволяє маніпулювати зображеннями та відео для виконання різноманітних завдань - від відображення зображення з веб-камери до потенційного навчання робота розпізнаванню реальних об'єктів. Застосовується в промисловості для забезпечення якості продукції, у сфері транспорту для розвитку автономних транспортних засобів та ін. Основні можливості – обробка зображень (зміна розміру, обертання, фільтрація, корекція кольору, виявлення країв), виявлення об’єктів (використання алгоритмів, такі як HIG і Haar Cascades для виявлення облич, автомобілів тощо), сегментація (розділення зображення на частини для подальшого аналізу), трасування об’єктів (відстеження рухомих об’єктів у відео за допомогою алгоритмів).

PIL (Pillow) – покращена версія оригінальної бібліотеки PIL, яка забезпечує прості засоби для роботи з зображеннями в Python. Більше підходить для простих завдань, по типу обробка зображень для веб-додатків чи базові маніпуляції з графікою. Основні можливості – базові операції (відкриття,

збереження, зміна формату зображень), обробка зображень (зміна розміру, обертання, обрізка, застосування фільтрів), графіка (додавання тексту, малювання простих фігур на зображення).

Системи для машинного навчання – має дієві інструменти, котрі знайшли застосування у багатьох галузях. Прогрес у сфері технологій, особливо глибинне навчання та обробка природної мови, веде до безперервного розширення можливостей машинного навчання, відкриваючи перед нами нові горизонти для його використання. Найбільш відомі платформи: TensorFlow і PyTorch.

TensorFlow – широко вживана бібліотека Python для глибинного навчання. Знайомство з TensorFlow та його потужними засобами, такими як TensorBoard і TensorFlow Serving сприяє створенню, навчанню та розгортанню нейронних мереж, що робить її чудовою для досліджень та комерційного застосування. Застосовується в комп'ютерному зорі, обробці природної мови, фінансовому аналізі та робототехніці.

PyTorch – Бібліотека для машинного навчання, була створена Facebook, що забезпечує динамічну архітектуру, дозволяючи, легко створювати та навчати моделі. Застосовується в наукових дослідженнях, обробці зображень, НЛП, та віртуальних помічниках.

2) **Технології**.

Нейронні мережі застосовуються для вирішення широкого спектра завдань, включаючи класифікацію зображень, обробку текстової інформації та прогнозування. Ці моделі можуть мати різну складність, від простих (з одним шаром) до складних (багатошарових структур), такими як Convolutional Neural Networks (CNN) для обробки зображення.

Основні типи нейронних мереж:

- Convolutional Neural Network (CNN).
- Recurrent Neural Network (RNN).
- Long Shor-Term Memory (LSTM).

CNN (Convolutional Neural Network) використовується в основному для обробки зображень. Має спеціалізовані шари для виявлення рис, саме таких як контури, текстури і форми. Складається з конволюційних шарів, що виконують операцію з фільтрами для витягнення ознак, а також з шару підвибірки, що зменшує розміри зображення, зберігаючи важливу інформацію.

Recurrent Neural Network (RNN) – Використовується для обробки послідовностей, таких як текст чи часові ряди. Запам'ятовують інформацію про попередні елементи послідовності, роблячи їх ідеальними для таких завдань, як обробка **природної мови.** Страждають від проблеми затухаючого градієнта, що ускладнює навчання на довгих послідовностях.

Long Short-Term Memory (LSTM) – Спеціалізований вид RNN (Recurrent Neural Network), який має здатність зберігати інформацію протягом тривалих періодів. Застосовується для прогнозування послідовностей, наприклад, текстових рядків чи музичних композицій.

Обробку природної мови (NLP) – сфера штучного інтелекту (ШІ), що розвивається на взаємодії між комп'ютерами та людською мовою. Його метою є розробка алгоритмів і моделей, що дозволяють комп'ютерам розуміти, аналізувати, генерувати та проводити маніпуляції мовою в зрозумілій для людини формі.

Його ключові функції включають токенізацію, стеммінг та лематизацію, частковий розбір речень, розпізнавання іменованих сутностей, аналіз настроїв, генерація тексту та машинний переклад.

Основні технології та бібліотеки:

Natural Language Toolkit (NLTK) – це бібліотека для Python, яка пропонує інструменти для токенізації, стемінгу, часткового аналізу, аналізу залежностей та ін.

spaCy – це високопродуктивна бібліотека для NLP, яка підтримує векторинізацію, частковий аналіз, виявлення іменованих сутностей та інші складні завдання. Оптимізована для швидкості та використовує сучасні моделі.

Transformers – це бібліотека від Hugging Face, що містить попередньо навчені моделі, саме такі як BERT, GPT, T5 для виконання різноманітних завдань.

3) **Архітектура.**

Галузь обробки зображень зазнала революції з появою DL (діафрагментного моделювання) – підмножини штучного інтелекту, натхненної структурою та процесами людського мозку для аналізу та інтерпретації складних шаблонів даних. Традиційно обробка зображень значною мірою спиралася на ручне вилучення ознак та класичні методи машинного навчання (ML), які вимагали значних знань у предметній області та часто мали проблеми з мінливістю та складністю, властивими візуальним даним. Ці методи, хоча й ефективні в конкретних, чітко визначених завданнях, не мали гнучкості та масштабованості, необхідних для обробки різноманітної та багатовимірної природи зображень реального світу.

Діагностичне моделювання, що характеризується здатністю вивчати ієрархічні представлення безпосередньо з необроблених даних, усунуло багато обмежень традиційних підходів. Впровадження багатошарових нейронних мереж (NN) дозволило моделям автоматично виявляти складні візерунки та ознаки, які раніше були недосяжні за допомогою ручних методів. Цей перехід від ручної розробки ознак до автоматизованого навчання ознак ознаменував собою поворотний момент в обробці зображень, дозволивши досягти значного прогресу як у точності, так і в узагальнюваності в широкому спектрі застосувань.

Одним із найважливіших проривів у DL стала можливість обробляти великомасштабні набори даних зображень, що забезпечило основу для розробки надійних та узагальнюваних моделей. Ці моделі не лише досягли успіху в традиційних завданнях обробки зображень, таких як класифікація та сегментація, але й відкрили нові шляхи для інновацій у сферах, які раніше вважалися занадто складними або обчислювально обтяжливими. Наявність великих наборів даних та збільшення обчислювальної потужності, зокрема завдяки використанню

графічних процесорів (GPU), ще більше прискорили цей прогрес, зробивши DL домінуючою парадигмою в обробці зображень.

Архітектурні досягнення в моделях DL також відіграли вирішальну роль у цій еволюції. Розробка складніших і глибших мереж, здатних фіксувати широкий спектр візуальних ознак у різних масштабах, дозволила обробляти зображення з безпрецедентною точністю. Ці моделі розвивалися для обробки різних аспектів обробки зображень, від низькорівневих завдань, таких як шумозаглушення та надвисокої роздільної здатності, до високорівневих завдань, таких як виявлення об'єктів та семантична сегментація. Кожне нове покоління моделей спиралося на успіхи своїх попередників, включаючи нові механізми для підвищення ефективності навчання, зменшення обчислювальних витрат та покращення інтерпретованості моделі.

Більше того, універсальність DL сприяла його застосуванню в численних галузях, демонструючи його здатність вирішувати складні та специфічні для кожної предметної області проблеми. Адаптивність моделей DL до різних типів візуальних даних — від природних зображень до медичних сканувань — призвела до проривів у різних галузях, суттєво вплинувши як на дослідження, так і на галузеву практику. Це утвердило DL не лише як інструмент для вирішення проблем обробки зображень, але й як фундаментальну технологію, що стимулює інновації в широкому спектрі наукових і технологічних зусиль.

Незважаючи на ці досягнення, застосування глибокого навчання (DL) в обробці зображень не позбавлене труднощів. Залежність від великих, маркованих наборів даних викликає занепокоєння щодо масштабованості цих моделей для завдань, де анотовані дані є рідкісними або їх важко отримати. Крім того, високі обчислювальні вимоги до навчання глибоких мереж, особливо зі зростанням складності моделей, створюють значні перешкоди для багатьох дослідників та практиків. Інтерпретація моделей глибокого навчання також залишається критичним питанням, особливо у високовартісних застосуваннях, де розуміння процесу прийняття рішень моделі є таким же важливим, як і її точність.

Сервер відповідає на запити клієнта, реалізовує бізнес-логіку, здійснює доступ до бази даних та надає результати клієнту. Розгортання сервера може відбуватися на фізичному або віртуальному обладнанні.

Переваги та недоліки.

Переваги: має ясне розділення між інтерфейсом користувача та логікою бізнесу, що спрощує розробку та обслуговування. Легко масштабувати сервери для обробки зростаючого навантаження. Сервер забезпечує безпеку інформації шляхом здійснення контролю над доступом до даних та ресурсів.

Недоліки: надає навантаження на сервер – якщо кількість клієнтів зростає, то сервер може стати «вузьким місцем» у системі. Залежний від з'єднання – саме клієнт не може отримати доступ до даних без постійного зв'язку з сервером. Обмеження продуктивності – втрата часу на оброблення запиту, це може впливати на швидкість роботи програми.

Мікросервісна архітектура відрізняється від класичної монолітної архітектури тим, що застосунок формується з незалежних, дрібних сервісів, кожен з яких реалізує конкретну інфомацію. Основними компонентами мікросервісної архітектури є мікросервіси та API Gateway.

Кожен мікросервіс є окремим модулем, який виконує певну бізнес-функцію. Спілкуються між собою через API, зазвичай за допомогою HTTP/REST або messaging систем.

API Gateway – Центральна точка для управління запитами, що надходять до мікросервісів. Він може виконувати функції маршрутизації, аутентифікації, обробки запитів та зворотних відповідей.

Переваги та недоліки.

Переваги: масштабність – окремі мікросервіси можна легко масштабувати під час підвищення навантаження. Гнучкість у технологіях – команди можуть використовувати різні мови програмування та технології для кожного мікросервісу. Швидке впровадження – оновлення одного мікросервісу не вимагає перезавантаження всього додатку, що прискорює цикл розробки. Знижена

складність – кожен мікросервіс є невеликим, що покращує підтримку та розуміння.

Недоліки: Складність управління – чим більше сервісів, тим більше зусиль з управлінням, моніторингом та маршрутизації запитів. Проблема з інтеграцією – спілкування між ним, може призвести до затримок та помилок, особливо при мережевих збоїв. Тестування – може бути складним через їх залежність від інших сервісів.

4) **Програмне рішення**

Фреймворк для веб-розробки – це програмний каркас, призначений для створення вебзастосунків, служб або ресурсів. Спрощує розробку, частково за рахунок автоматизації, позбавляючи, від необхідності написання рутинного коду.

Основними бібліотеками є Django та Flask.

Django – це високорівневий фреймворк для мови Python, який дозволяє швидко створювати веб-додатки за рахунок вбудованих інструментів та бібліотек. Застосовується для створення складних веб-додатків, таких як соціальні мережі, платформи електронної комерції, системи управління контентом.

Його основні можливості:

- **Object-Relational Mapping (ORM)** – дозволяє працювати з базами даних за допомогою Python-коду, без необхідності написання SQL-запитів.

- Аутентифікація – в програмі передбачені інтегровані модулі для керування користувачами, їх реєстрацією, аутентифікацією та авторизацією.

- Адміністративна панель – автоматично генерує адміністративний інтерфейс для управління моделями даних.

- Безпека – захищає від загроз, таких типу як CSRF, XSS та SQL-ін’єкції.

Flask – легкий веб-фреймворк для мови Python, що забезпечує простоту та гнучкість у розробці веб-додатків. Застосовується для розробки простих веб-додатків, мікросервісів і API, а також для навчальних проєктів.

Його основні можливості:

- Модульність – легко налаштовується та розширюється за допомогою сторонніх бібліотек.

- Простота – мінімалістичний підхід, що дозволяє швидко розпочати проєкт без зайвих налаштувань.

2.3 Розгорнута постановка завдання

В сучасному світі, де цифровий прогрес кардинально змінює лад життя, спостерігаємо шалене збільшення обсягів графічного матеріалу. Це стосується багатьох сфер, від безпеки до маркетингу, навчання та повсякденних додатків. У зв'язку з цим постає нагальна потреба в програмному забезпеченні, яке б мало здатність автоматично аналізувати та категоризувати графічний контент на основі його змісту. Такий підхід дає змогу впорядкувати зберігання інформації, прискорити пошук потрібних даних, удосконалити класифікацію та збільшити продуктивність обробки зображень.

Суть цієї задачі – розробити програмне забезпечення, який реалізує автоматизований процес розподілення (сортування або класифікації) зображень, враховуючи їхній вміст, застосовуючи передові технології комп'ютерного зору та машинного навчання. Вона повинна мати інтуїтивно зрозумілий інтерфейс користувача, дозволяти імпортувати зображення, виконувати їхню попередню обробку, здійснювати аналіз вмісту, класифікувати зображення відповідно до заданих або автоматично згенерованих категорій а також зберігати або візуалізувати отримані результати.

Функціональними вимогами є:

- Завантаження та зберігання графічних файлів.
- Аналіз вмісту.
- Класифікація та тегування.
- Розподіл.
- Аналітика.

Завантаження та зберігання. Користувачі повинні мати можливість завантажувати графічні файли в систему та зберігати їх у структурованих каталогах. **Аналіз вмісту**. Використання алгоритмів обробки зображень для автоматичного визначення вмісту графіки (приклад, об’єкти, кольори, текстури). **Класифікація та тегування**. Автоматизоване тегування графічних матеріалів на основі їх вмісту для полегшення пошуку. **Розподіл**. Можливість автоматичного розподілу графіки до різних каналів (тобто, веб-сайти, соц. мережі, внутрішні системи). **Аналітика**. Збір даних про використання графіки та її ефективність у різних контекстах.

Нефункціональними вимогами є:

-Продуктивність.

-Безпека.

-Масштабованість.

Продуктивність. Система повинна обробляти великі обсяги графічних даних з прийнятною швидкістю (скажімо, обробка до 1000 зображень на годину) **Безпека** – забезпечує захистом даних користувачів, а також контролює доступ до вмісту. **Масштабованість**. Система має забезпечувати можливість горизонтального масштабування для ефективного оброблення зростаючих навантажень даних.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЄКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Програмне забезпечення виконує такі функції, як:

- Завантаження зображення.
- Попередня обробка зображення.
- Аналіз вмісту зображення.
- Класифікація та маркування.
- Автоматизоване розподілення.
- Інтерфейс користувача.
- Журнал та аналітика.

Модуль **завантаження** виконує функцію отримання зображення від користувача для його подальшої обробки. Він пропонує зручний та надійний інтерфейс для імпортування графічних файлів, перевіряє їх на відповідність формату та готує до передачі в модуль попередньої обробки. Основними функціями є вибір файлів, перевірка типу файлів, перевірка метаданих та завантаження у систему. Вибір файлу – можливість завантаження одного чи декількох зображень одночасно, підтримує такі дії як кнопка «Завантажити файли», перетягувати файли у зону завантаження. Підтримує лише формати .jpg, .png, .jpeg. Файли, що не підтримує формат – відхиляє з повідомленням про помилку. Зчитує базових метаданих. Зберігає зображення у буфеті чи окремій директорії, має призначення унікального ідентифікатора для кожного зображення. Передає посилання чи файлових об’єктів у модуль попередньої обробки.

Модуль **попередньої обробки** відповідає за приведення всіх вхідних зображень до стандартизованого вигляду, що забезпечує коректну та стабільну роботу подальших етапів комп’ютерного зору (аналіз, класифікація,

розпізнавання). Головними завданнями є мінімізація негативного впливу «зашумлених» або недостатньо освітлених зображень, приведення вхідних даних до єдиного формату (розміри, яскравість, кольоровий простір), що покращує точність розпізнавання об'єктів та підготовка даних перед передачею в модель глибокого навчання (CNN, YOLO).

Модуль **аналізу** контенту — ключовий елемент системи. Головне його призначення — виявлення, розпізнавання та визначення місцезнаходження об'єктів на картинках, разом з формуванням семантичного опису сцени. Основними функціями аналізу є виявлення об’єктів - визначає розташування об’єктів на зображенні у вигляді координат прямокутника, класифікація об’єктів – присвоює кожному виявленому об’єкту свій клас (наприклад, людина, автомобіль, дерево, тварина, будівля). Сегментація – поділяє зображення на області, які відповідає певним класам, відокремлює об’єкти одного типу між собою.

Модуль **класифікації** виконує задачу визначення приналежності зображення до однієї або кількох категорій, а також наділяє його семантичними мітками, що розкривають його зміст, стиль та особливості. Система створює логічне представлення зображення, що згодом застосовується для сортування, пошуку, збереження та аналізу. Ключовими задачами модуля є класифікація за стилем – визначає типи зображення за його джерелом або способом отримання (фотографії, скани, ілюстрації, скріншоти), визначення присутності тексту – якщо зображення містить текстові елементи, система додає відповідну ознаку.

Модуль **автоматизованого розподілу** відповідає за сортування зображень у чітко структуровану систему зберігання. Він використовує попередньо визначені категорії, теги та метадані. Головна мета модуля — суттєво полегшити навігацію, пошук, архівування, а також подальшу обробку або експорт файлів, при цьому мінімізуючи участь користувача. Основними функціями автоматизованого розподілення є розподілення за категоріями – на основі основної категорії зображення визначається шлях чи структура зберігання,

розподілення за стилем – створює додаткові ієрархії чи підкаталоги стилю, уникнення конфліктів – якщо файл існує, то система перейменовує новий файл (наприклад, img_0569_(1).png). Його переваги – економія часу (включає ручне сортування), стандартизація зберігання даних, підготовка до подальшої обробки або архівування.

Інтерфейс користувача гарантує інтуїтивну, продуктивну та комфортну взаємодію із системою. Головна функція – надати користувачеві можливість завантажувати зображення, спостерігати результати класифікації, власноруч корегувати категорії, здійснювати пошук, фільтрувати, експортувати або видаляти дані, отримувати статистичні дані обробки. Основними компонентами інтерфейсу є головна панель – має коротку інформацію про поточний стан системи (статистика, кнопки швидкого доступу - завантажити, пошук, зберігання в текстовому файлі, вихід з програми), модуль завантаження зображення – може завантажити одне або декілька зображень, підтримує такі формати як JPEG, PNG, експорт – передає дані в текстовому файлі. Перевагами є простота у використанні навіть для нефахівців, повна прозорість результатів обробки, швидке ручне втручання у разі потреби, гнучкість експорту та інтеграції з іншими сервісами, можливість масштабування та адаптація під проєкт.

Журналювання та аналітика забезпечує прозорість, контроль і зворотний зв’язок про процеси, що відбуваються в системі. Дає змогу відстежувати всі дії, виконані системою або користувачем, накопичування статистику щодо обробки зображення, виявлення помилок, збої, а також оцінювати ефективність роботи.

3.2 Розробка структурної схеми

На рис. 3.1 зображена структурна схема. В цій схемі є вхідні дані, обробка даних, модуль завантаження, збереження, аналізу та ін.

В вхідні дані ми надаємо фотозображення, що може його обробити.

Обробка даних – попередня підготовка вхідної інформації, тобто, очищення, форматування. В модуль завантаження ми завантажуємо зображення, яке ми саме хочемо аналізувати. Модуль аналізу виконує глибокий аналіз отриманих даних. Розпізнає об’єкти на зображеннях чи в даних. Аналізує вміст, тобто, надає семантичний чи контекстуальний аналіз інформації. Модуль збереження – зберігає результати аналізу в текстовому повідомлені.

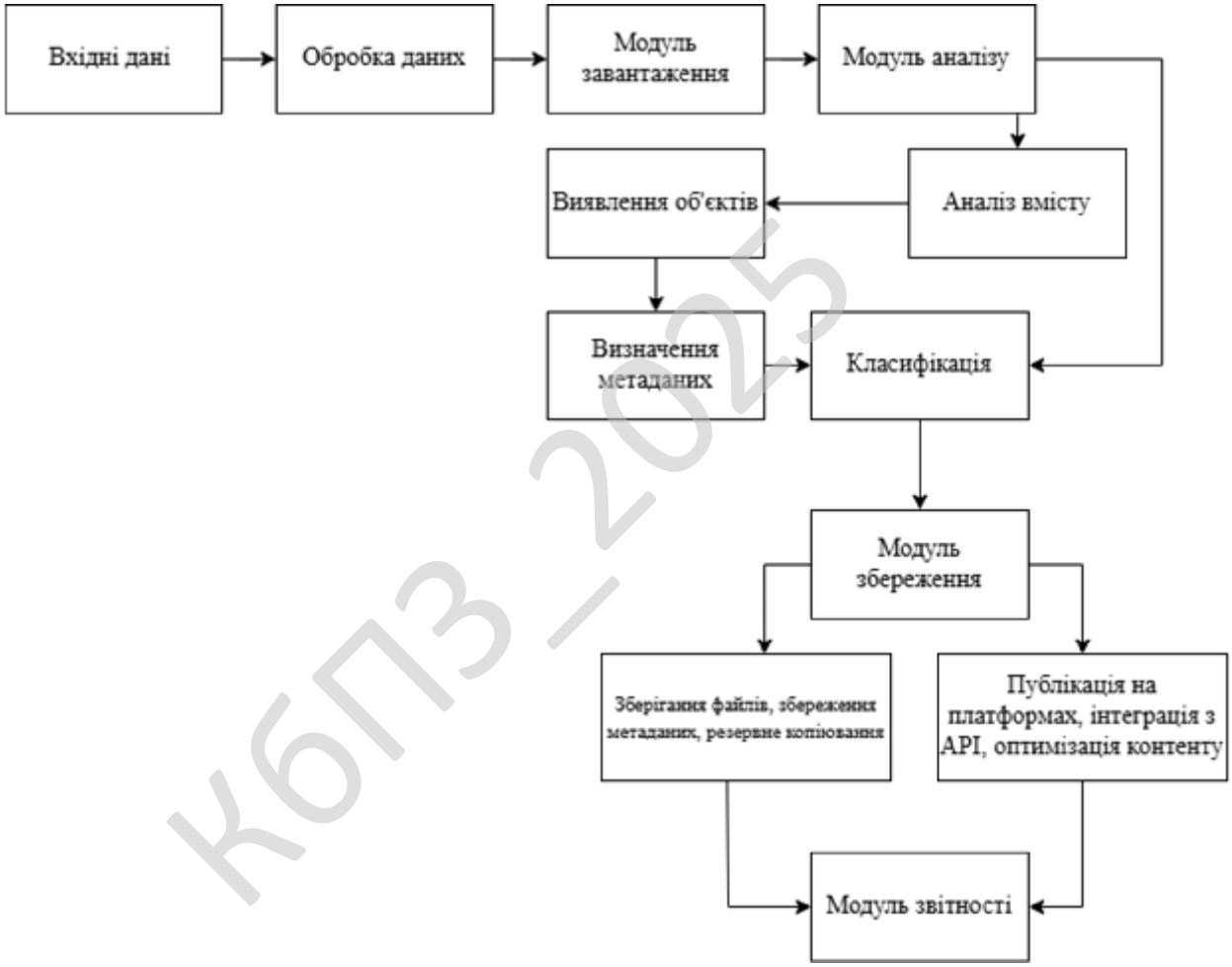

Рисунок. 3.1 – Структурна схема

3.3 Розробка функціональної схеми

Для дієвого вирішення завдання автоматизованого розподілу графічної інформації за змістом, було спроєктовано функціональну схему програмного забезпечення. Вона демонструє ключові стадії опрацювання, аналізу та

класифікації зображень.

Основними компонентами системи є модуль завантаження – забезпечує імпорт зображення з локального сховища. Підтримує масове завантаження та перевірку формату файлу, обробка зображення - виконує уніфікацію вхідних зображень: змінює розмір, нормалізує кольорову гаму, зменшує рівень шумів, а також перетворює їх у формат, що відповідає вимогам моделі аналізу, аналіз вмісту – ідентифікує об’єкти на зображення, модуль класифікації забезпечує логічне сортування зображення згідно з результатами класифікації, модуль зберігання результатів та виводу інформації – зберігає в файловій системі.

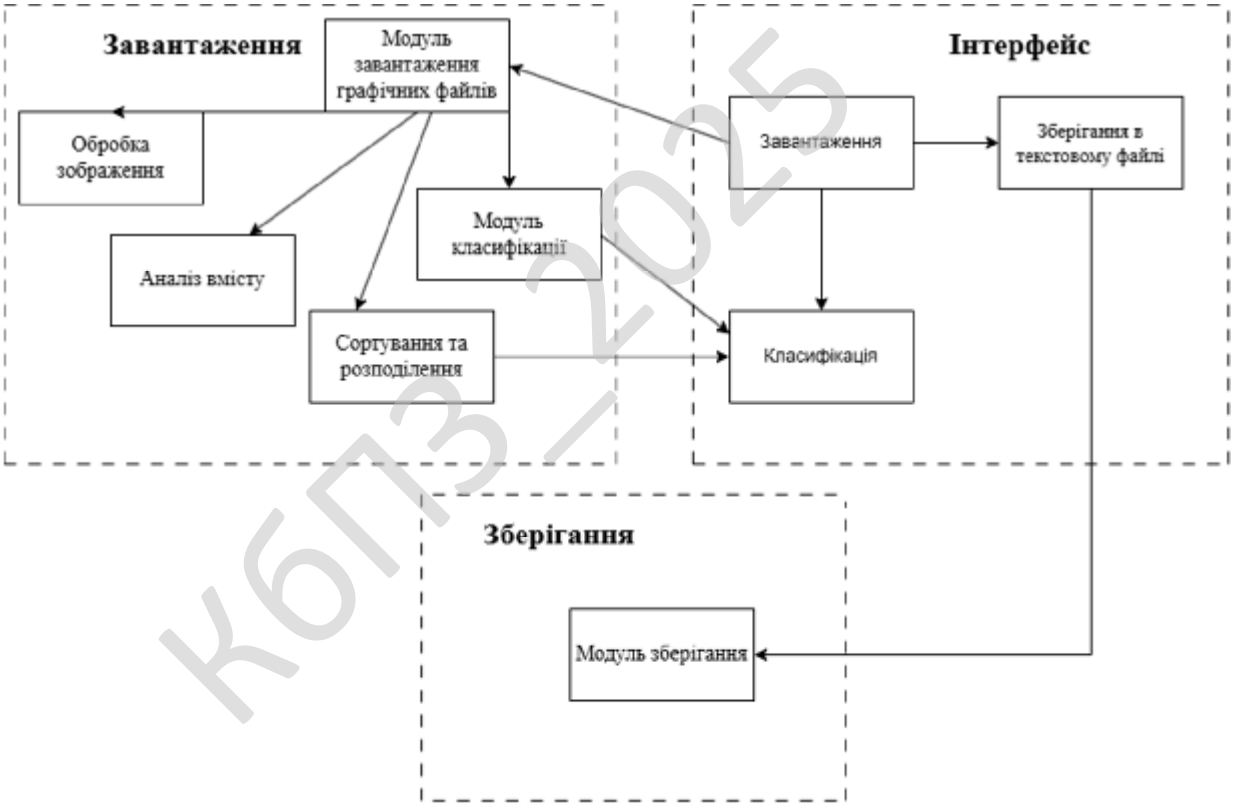

Рисунок. 3.2 – Функціональна схема

3.4 Розробка діаграми процесів

Діаграма процесів — це графічне представлення послідовності дій, які виконуються в рамках певного процесу. Вона показує, як різні етапи

взаємопов’язані, які ролі беруть участь у виконанні завдань, а також де можуть виникати затримки або проблеми**.**

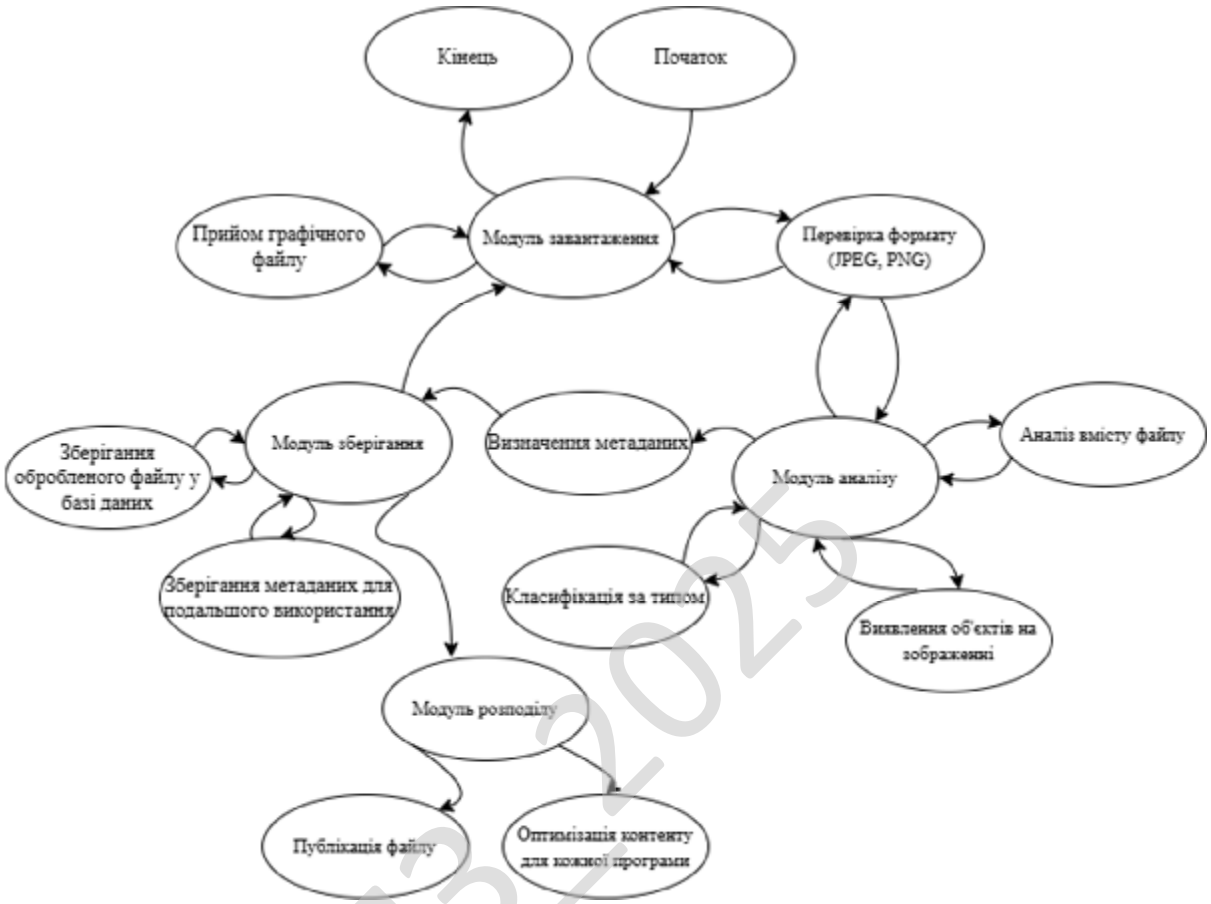

Рисунок 3.3 – Діаграма процесів

На рис 3.3 зображена діаграма процесів програмного забезпечення:

- Початок – обробка графічного файлу.

- Модуль завантаження файлу – відповідає за завантаження графічного файлу в цю систему.

- Прийом графічного файлу – відповідає для подальшої обробки зображення.

- Перевірка формату – аналізує формат файлу, що визначає чи підходить саме цей формат чи ні.

- Визначення метаданих – витягує метадані з файлу, що може бути використаний для подальшої обробки зображення.

- Модуль зберігання – зберігає формат в базі даних.
- Модуль рендерингу – відповідає за підготовку файлу до відображення.
- Публікація файлу – може бути доступним для користувачів.
- Аналіз вмісту – аналізує вміст файлу для подальшої класифікації.
- Класифікація за темою – визначає тематику зображення для його організації в системі.
- Кінець – завершення процесу.

4. РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

В межах функціональності впроваджено механізм класифікації та сортування зображень, керуючись їхнім змістом. Етапи розподілення зображення:

- Завантаження зображень – можливість завантажувати один або кілька зображень через графічний інтерфейс. Зберігається у тимчасовому каталозі для подальшої обробки.
- Попередня обробка – проходить етап нормалізації.
- Аналіз вмісту зображень – виконує розпізнавання об’єктів, що знаходяться на зображенні.
- Класифікація – на основі розпізнаних об’єктів чи ознак зображення відноситься до певної категорії (Наприклад, природа, тварини, техніка, люди, транспорт).
- Розподілення та сортування – переміщує зображення у відповідні підкаталоги за визначеними категоріями. Створює вивід у вигляді візуальної галереї, де зображення відображаються згруповано.
- Збереження та представлення результатів – зберігає результати класифікації.

Система задіє алгоритми машинного навчання та комп’ютерного бачення для автоматичного розпізнавання та класифікації зображень, що забезпечує ефективне прийняття рішень щодо подальшої обробки та маршрутизації кожного зображення.

На рис 4.1 буде блок-схема основної програми та те, що в ньому знаходиться.

Початок:

- Завантажує графічний файл.
- Початок роботи ObjectDetector: завантажує модель YOLOv11 (що використовується для виявлення об’єктів). Передає зображення в модель, що повертає координати об’єктів та їх назви.
- Працює класифікатор: надсилає зображення через API стороннього сервісу SightEngine. Отримує інформацію про тип зображення. Обирає найімовірніший тип.
- Працює сегментатор: використовує модель DeepLabV3+ з ResNet101 для семантичної сегментації.
- Класифікує зображення.
- Виявляє об’єкти, що знаходяться в зображенні.
- Класифікує.
- Створює звіт.

Кінець.

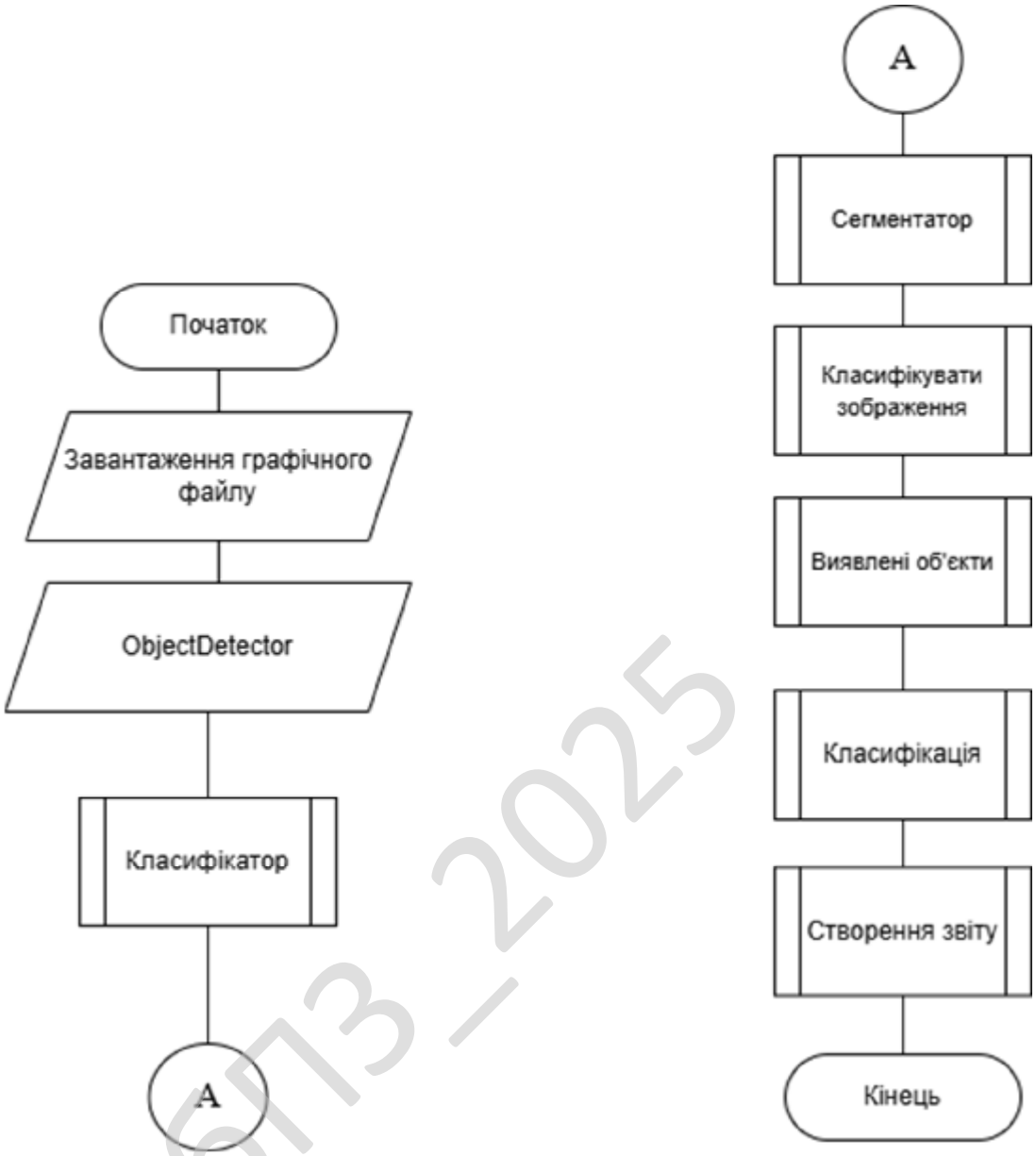

Рисунок 4.1 – Блок-схема роботи основної програми

На рис 4.2 буде блок схема, що відповідає самої програмі, та опис її.

Початок:

- Завантаження графічного файлу: користувач обирає зображення, яке буде аналізуватись.

- Класифікатор: програма визначає загальний тип зображення (фото, малюнок, скан тощо).

- Сегментатор: програма поділяє зображення на окремі області – приклад, виділяє фон та об'єкти.

- Класифікувати зображення: виконує глибокий аналіз, що визначає, що саме зображено (наприклад: «автомобіль», «дерево», «людина»).

- Виявлені об’єкти: об’єкти, що знайдені на зображені, зберігаються та можуть бути показані у вигляді прямокутників із підписами.

- Створення звіту: формує результат аналізу – список об’єктів, тип зображення, можливо – підсумкову картинку.

- Кінець – Завершення роботи програми.

Рисунок 4.2 – Блок-схема роботи підпрограми

Отже, ця програма являє собою потужну систему комп’ютерного зору, що аналізує зображення з різних точок зору: тобто, визначає що на ньому зображено (детекція), який його тип (класифікація), як воно структуровано (сегментація) та як його можна описати словами (CLIP-контекст).

Використані бібліотеки та функції:

```
        import os
import threading
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk, ImageDraw, ImageFont
from detector import ObjectDetector, Classifier, Segmenter, ImageAnalyzer
from download_assets import download_all_assets
        import requests
import torch
        import clip
        import urllib.request
```

Створення списку фотозображень користувача та збереження їх в базу даних. В створеному програмному забезпечені використано бібліотеку os та glob для пошуку файлів з зображеннями. Функція модуля os дозволяють створювати відносні та абсолютні шляхи з абстрагуванням від використаної операційної системи. Для цього функція path.join приймає фрагменти шляхів до файлу та форматує остаточний шлях за правилами поточної операційної системи.

Комп'ютерний зір дозволяє використовувати властивості людського зору на комп'ютері. Комп'ютер може бути у вигляді смартфона, безпілотника, камери відеоспостереження, МРТ-сканера тощо, з різними сенсорами для сприйняття. Сенсор створює зображення в цифровому вигляді, яке має бути інтерпретоване комп'ютером. Основний будівельний блок такої інтерпретації або інтелекту пояснюється в наступному розділі. Різні проблеми, які виникають у комп'ютерному зорі, можуть бути ефективно вирішені за допомогою методів глибокого навчання.

Використані програмні функції та модулі:

Функція os відповідає за створення відносні і абсолютні шляхи з абстрагуванням від використаної операційної системи. Функція path.join приймає фрагменти шляхів до файлу та форматує остаточний шлях за правилами поточної операційної системи. Функція glob відповідає за пошуки файлів зображення. Модуль threading відповідає за роботу з потоками, дозволяє запускати одночасне виконання декілька частин коду в межах однієї програми.

Модуль tkinter є стандартною бібліотекою, що створює графічний інтерфейс користувача, створює вікна, кнопки, поля вводу, мітки, меню, діалоги та ін. from PIL import Image, ImageTk, ImageDraw, ImageFont - відповідає за імпорт модулів з бібліотеки PIL, що використовується для роботи з зображенням. from detector import ObjectDetector, Classifier, Segmenter, ImageAnalyzer – відповідає за імпорт класи з модуля detector, що призначена для оброки зображень за допомогою комп'ютерного зору. Модуль ObjectDetector відповідає за виявлення об 'єктів на зображені. Classifier – відповідає за класифікацію зображення чи частини зображення. Segmenter – виділяє пікселі, що належить конкректним об'єктам. ImageAnalyzer – відповідає за загальний аналіз зображення, поєднує всі можливості чи виконує мета-аналіз. from download_assets import download_all_assets – відповідає за імпорт функції з модуля. Функція download_all_assets відповідає за завантаження всіх необхідних ресурсів для роботи з програмою.

Функція requests відповідає за завантаження даних. Функція torch відповідає за створення та запуск нейронної мережі. Функція clip вміє зіставляти текст та зображення. Може описати зображення словами та модель покаже, яке зображення найкраще відповідає опису. Urllib.request відповідає за інтернет-запити. Використовується для завантаження файлів з URL.

З точки зору класифікації зображень - це завдання достовірного позначення всього зображення об'єктом або поняттям. Серед прикладів - класифікація статі за зображенням обличчя людини, визначення типу домашньої тварини, позначення фотографій тегами тощо.

4.2 Захист розробленого програмного забезпечення

З метою забезпечення надійності, безпеки та авторських прав розробленого програмного забезпечення були реалізовані такі заходи захисту:

- Авторське право та ліцензування:

Цей код написаний власноруч з дотриманням умов відкритих ліцензій (прикладом є MIT, Apache 2.0).

Треті бібліотеки, такі як PyTorch, Ultralystic YOLO, torchvision та CLIP, використовуються з дотриманням їхніх ліцензій.

- Захист від несанкціонованого доступу:

Ключі до зовнішніх API (скажімо для SightEngine) зберігають окремо від основного коду.

У фінальній версії програми передбачено шифрування або приховування конфіденційних даних.

- Контроль доступу та безпечний запуск:

Програмне забезпечення може бути адаптовано до обмеженого кола користувачів (локально чи через автентифікацію).

Вбудовані перевірки типів даних та форматів вхідних файлів зменшують ризик помилок або атак через несподівані формати.

- Інтеграція з системами контролю версій:

Весь процес розробки вівся через систему Git, що дозволяє відстежувати зміни та захищати від втрати чи пошкодження коду.

Також дозволяє уникнути внесення шкідливих змін стороннім особам.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Імплементація розробленої системи аналізу зображень в реальних виробничих умовах потребує послідовного підходу, який гарантуватиме стабільність функціонування, масштабованість та простоту обслуговування.

Основні етапи впровадження:

1) Підготовка середовища.

- Вибір серверного чи хмарного середовища (AWS, Azure чи локальний сервер).

- Встановлення необхідного програмного забезпечення: Python, бібліотеки машинного навчання (Torch, torchvision, Ultralystics YOLO, CLIP тощо).

2) Розгортання моделей та налаштування ресурсів (GPU, CPU, пам’ять).

- Інтеграція в існуючу IT-інфраструктуру.

- Забезпечення сумісності із вже наявними інформаційними системами.

- Створення API-інтерфейсу чи інтеграційного модуля, через який інші системи можуть надсилати зображення на аналіз.

3) Тестування та валідація.

- Проведення функціонального тестування всіх модулів (детектор, класифікатор, сегментатор, аналізатор контексту).

- Перевірка точності та швидкості на реальних прикладах (вмикаючи граничні випадки).

- Збір відгуків від кінцевих користувачів чи операторів.

4) Масштабування та повномасштабне розгортання.

- Після успішного пілотного запуску – розгортання системи на повну потужність.

- Створення резервних копій, впровадження механізмів оновлення,

моніторингу та техпідтримки.

Нижче будуть наведені скріншоти роботи самої програми:

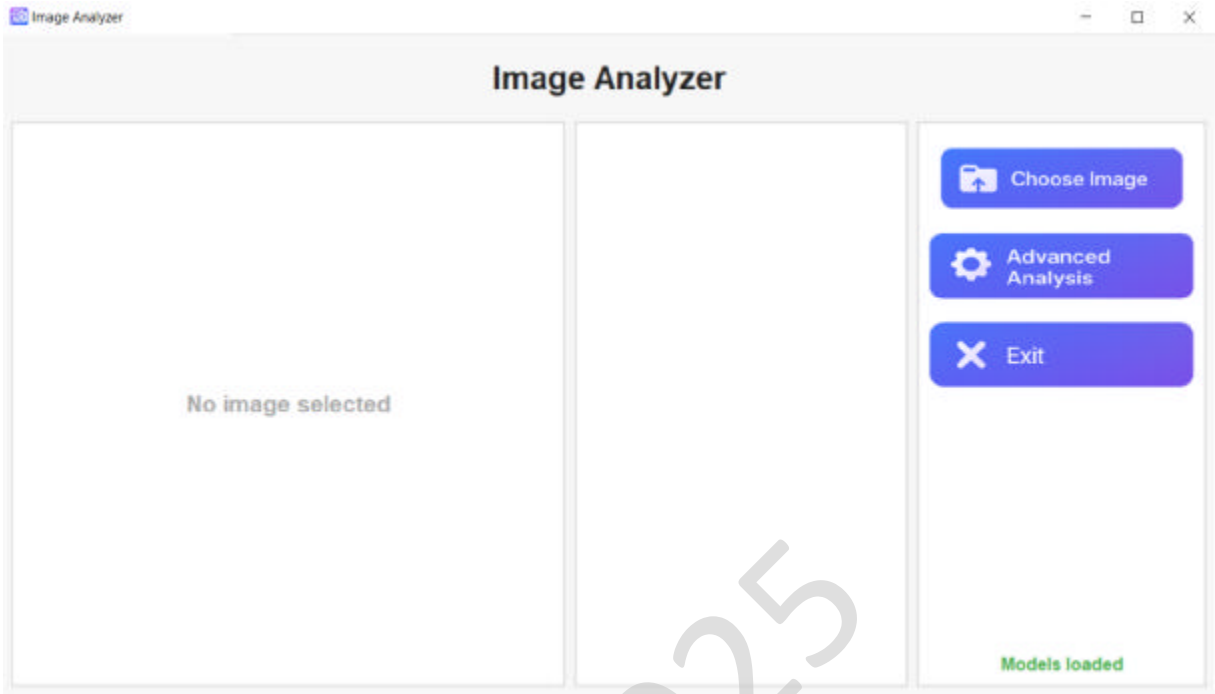

Рисунок 5.1 - Програма

Рисунок 5.2 – Розпізнавання фотографії кота

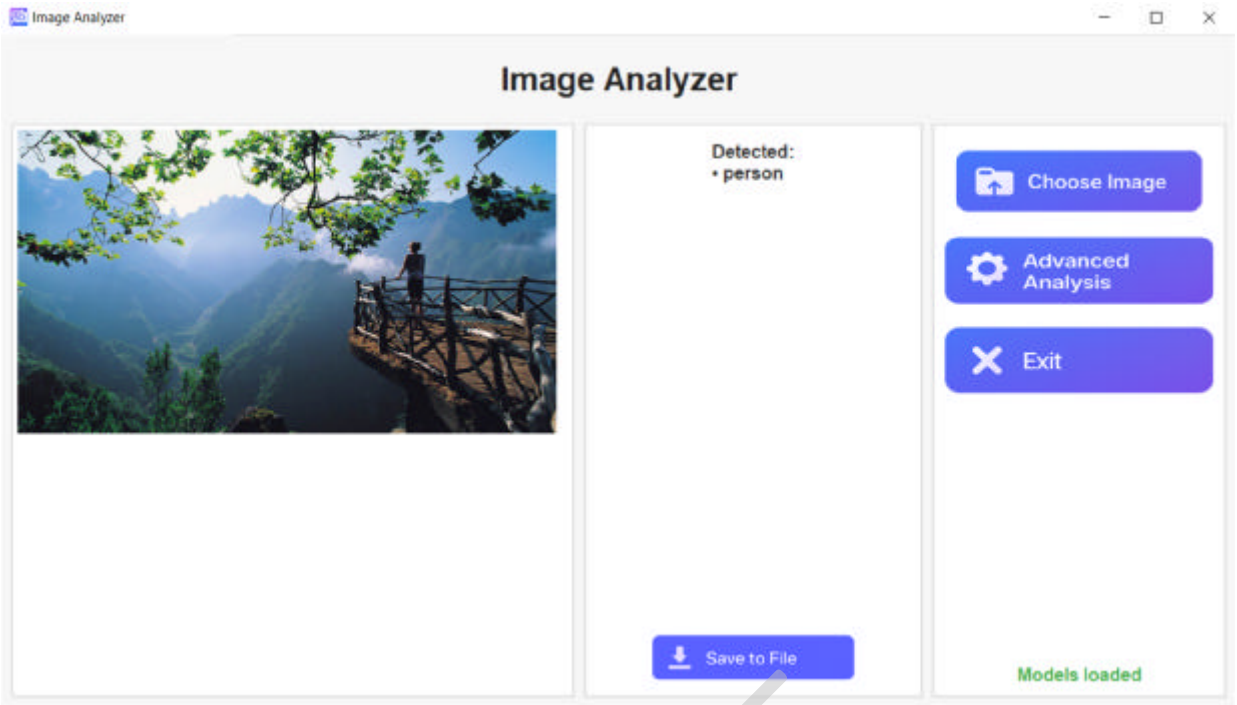

Рисунок 5.3 – Розпізнавання фотографії природи

Рисунок 5.4 - Процес розпізнавання кота

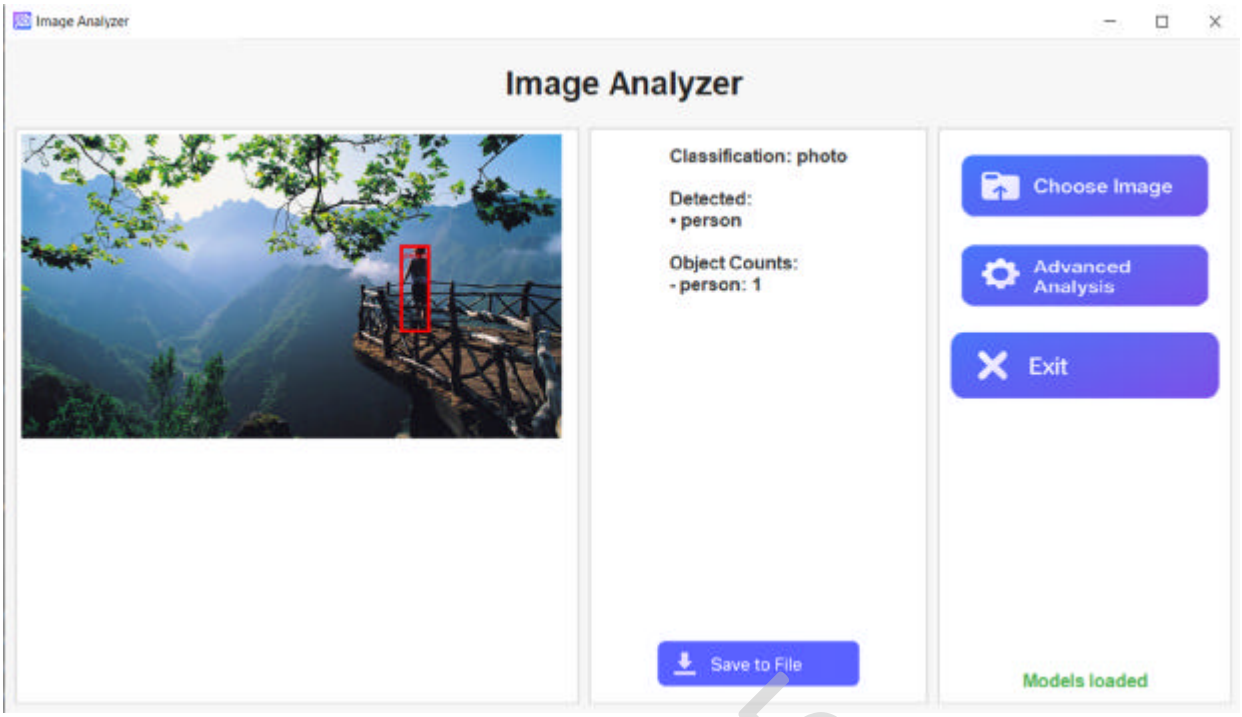

Рисунок 5.5 – Процес розпізнавання природи

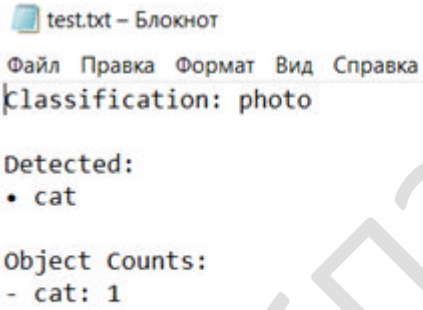

Рисунок 5.6 – Збереження результатів

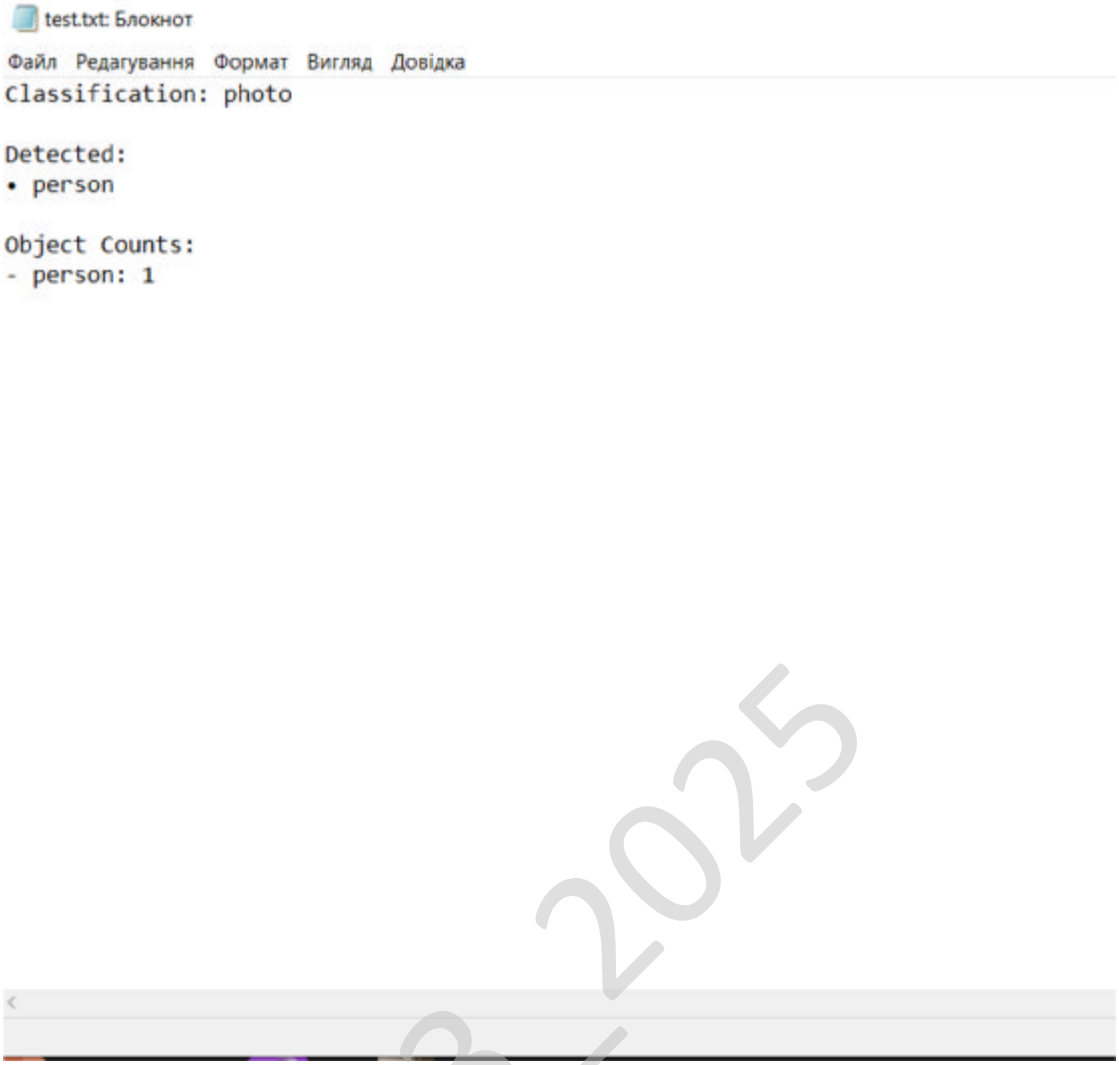

Рисунок 5.7 – Збереження результату в текстовому вигляді

6 ОСНОВНІ ВИСНОВКИ

У процесі розробки програмного забезпечення для автоматизованого розподілення графічної інформації було створено ефективну систему, що здатна здійснювати інтелектуальний аналіз зображень з подальшим класифікуванням та маршрутизацією файлів відповідно до змісту. Основою системи стали сучасні підходи комп'ютерного зору, штучного інтелекту та глибокого машинного навчання, що забезпечило високу точність розпізнавання об'єктів, сцен, текстів та інших семантичних елементів на зображеннях.

Результати впровадження показали, що система ефективно виконує поставлені задачі, демонструючи стабільну роботу та високу точність класифікації навіть при обробці значних обсягів графічних даних. Це дозволило суттєво скоротити час, необхідний для ручної обробки інформації, підвищить ступінь автоматизації процесів та зменшити ймовірність помилок, пов'язаних з людським фактором.

Значущість розробленого ПЗ підтверджується широкими можливостями його практичного застосування у різних галузях:

- У сфері безпеки – для аналізу відео та фото з метою виявлення об'єктів чи подій, що потребують уваги.
- В електронному документообігу – для автоматичної обробки сканованих матеріалів та їх класифікації.
- У торгівлі – для автоматичної категоризації товарів за фотографіями.

Таким чином, також були враховані аспекти масштабованості, адаптивності та інтеграції системи у більш складні програмні комплекси, що робить її придатною для подальшого вдосконалення та розширення функціоналу.

Отже, розроблене програмне забезпечення є актуальним, технічно обґрунтованим рішенням, здатним забезпечити ефективну автоматизацію процесів розподілу графічної інформації, що відповідає сучасним вимогам інформаційних

систем.

На перспективу розвитку спрямовані більш адаптивних моделей машинного навчання, інтеграція з хмарними сервісами та розширення навчальної бази даних для тренування системи на нових категоріях зображень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Goodfellow I., Bengio Y., Courville A. *Deep Learning*. — MIT Press, 2016. — 775 p. - https://files.batistalab.com/teaching/attachments/chem584/Ian_front_matter.pdf

2. Szeliski R. *Computer Vision: Algorithms and Applications*. — Springer, 2010. — 812 p. - https://books.google.com.ua/books?hl=uk&lr=&id=QptXEAAAQBAJ&oi=fnd&pg=PR9&dq=2.%09Szeliski+R.+Computer+Vision:+Algorithms+and+Applications.+%E2%80%94+Springer,+2010.+%E2%80%94+812+p.&ots=BNyhw0Uzvo&sig=8PZuVE4b8BQ9a1S0FTCsoDj9TV4&redir_esc=y#v=onepage&q&f=false

3. Chollet F. *Deep Learning with Python*. — Manning Publications, 2017. — 384 p.

4. Bishop C. M. *Pattern Recognition and Machine Learning*. — Springer, 2006. — 738 p. - https://link.springer.com/book/9780387310732

5. Russell S., Norvig P. *Artificial Intelligence: A Modern Approach*. — Pearson, 2021. — 1136 p.

6. Howse J., Minichino J. *Learning OpenCV 4 Computer Vision with Python 3*. — Packt Publishing, 2019. — 400 p. - https://books.google.com.ua/books?hl=uk&lr=&id=iNlOCwAAQBAJ&oi=fnd&pg=PP1&dq=6.%09Howse+J.,+Minichino+J.+Learning+OpenCV+4+Computer+Vision+with+Python+3.+%E2%80%94+Packt+Publishing,+2019.+%E2%80%94+400+p.&ots=iV_Jd2Vpm7&sig=kGSi1BGWThX23Rzcga3VxQ7ucTc&redir_esc=y#v=onepage&q&f=false

7. Shanmugamani R. *Deep Learning for Computer Vision*. — Packt Publishing, 2018. — 320 p. - https://books.google.com.ua/books?hl=uk&lr=&id=6tRJDwAAQBAJ&oi=fnd&pg=PP1&dq=7.%09Shanmugamani+R.+Deep+Learning+for+Computer+Vision.+%E2%80%

94+Packt+Publishing,+2018.+%E2%80%94+320+p.&ots=FfgcA_JJx4&sig=oOD4AW0DmHSlTe2ZH1zqD5LMFEo&redir_esc=y#v=onepage&q&f=false

8. Prince S. J. D. *Computer Vision: Models, Learning, and Inference*. — Cambridge University Press, 2012. — 580 p. - https://books.google.com.ua/books?hl=uk&lr=&id=PmrICLzHutgC&oi=fnd&pg=PA1&dq=8.%09Prince+S.+J.+D.+Computer+Vision:+Models,+Learning,+and+Inference.+%E2%80%94+Cambridge+University+Press,+2012.+%E2%80%94+580+p.&ots=cUMVSoO03v&sig=LPtl5bhqlZEbN_pmmHUCiLHBHSo&redir_esc=y#v=onepage&q&f=false

9. Forsyth D. A., Ponce J. *Computer Vision: A Modern Approach*. — Pearson, 2011. — 816 p. - https://dl.acm.org/doi/abs/10.5555/580035

10. Zhu S.-C., Mumford D. *A Stochastic Grammar of Images*. — Now Publishers Inc., 2007. — 136 p. - https://www.nowpublishers.com/article/Details/CGV-018

11. Minaee S. et al. *Image Segmentation Using Deep Learning: A Survey*. — arXiv:2001.05566, 2020. - https://ieeexplore.ieee.org/abstract/document/9356353

12. Zhang Z. et al. *Image classification with deep learning: A review*. — *Neurocomputing*, Vol. 439, 2021, pp. 221–234. - https://www.mdpi.com/1424-8220/25/2/531

13. Tetzlaff K. et al. *Performance of automated image classification*. — ResearchGate, 2022. - https://www.researchgate.net/profile/Jochen-Hartmann/publication/363738835_Performance_of_automated_image_classification/links/632c0fcb70cc936cd328df15/Performance-of-automated-image-classification.pdf

14. He K. et al. *Deep Residual Learning for Image Recognition*. — *CVPR*, 2016. - https://openaccess.thecvf.com/content_cvpr_2016/html/He_Deep_Residual_Learning_CVPR_2016_paper.html

15. Redmon J., Farhadi A. *YOLOv3: An Incremental Improvement*. — arXiv:1804.02767, 2018. - https://ask.qcloudimg.com/draft/2661027/i16j8zgndj.pdf

16. Mairal J. *Sparse coding for machine learning, image processing and*

computer vision. — PhD thesis, ENS Cachan, 2010. - https://hal.science/tel-00595312/

17. Chen Y., Wang J. Z. *Image Categorization by Learning and Reasoning with Regions*. — *Journal of Machine Learning Research*, 2004. - https://www.jmlr.org/papers/volume5/chen04a/chen04a.pdf

18. Li J., Wang J. Z. *Automatic Linguistic Indexing of Pictures by a Statistical Modeling Approach*. — *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2003. - https://ieeexplore.ieee.org/abstract/document/1227984

19. Datta R. et al. *Image Retrieval: Ideas, Influences, and Trends of the New Age*. — *ACM Computing Surveys*, 2008. - https://dl.acm.org/doi/abs/10.1145/1348246.1348248

20. Joshi D. et al. *Aesthetics and Emotions in Images: A Computational Perspective*. — *IEEE Signal Processing Magazine*, 2011. - https://ieeexplore.ieee.org/abstract/document/5999579

21. Krizhevsky A., Sutskever I., Hinton G. E. *ImageNet Classification with Deep Convolutional Neural Networks*. — *Communications of the ACM*, 2017. - https://dl.acm.org/doi/pdf/10.1145/3065386

22. Simonyan K., Zisserman A. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. — arXiv:1409.1556, 2014. - https://arxiv.org/abs/1409.1556

23. Szegedy C. et al. *Going Deeper with Convolutions*. — *CVPR*, 2015. - https://arxiv.org/abs/1412.1441

24. Huang G. et al. *Densely Connected Convolutional Networks*. — *CVPR*, 2017. https://openaccess.thecvf.com/content_cvpr_2017/html/Huang_Densely_Connected_Convolutional_CVPR_2017_paper.html

25. Tan M., Le Q. V. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. — *ICML*, 2019. - https://proceedings.mlr.press/v97/tan19a.html?ref=jina-ai-gmbh.ghost.io

26. Dosovitskiy A. et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. — *ICLR*, 2021. - https://arxiv.org/pdf/2010.11929/1000

27. Chen T. et al. *A Simple Framework for Contrastive Learning of Visual*

Representations. — *ICML*, 2020. - https://proceedings.mlr.press/v119/chen20j.html

28. He K. et al. *Mask R-CNN*. — *ICCV*, 2017. - https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf?ref=https://githubhelp.com

29. Zoph B. et al. *Learning Transferable Architectures for Scalable Image Recognition*. — *CVPR*, 2018. - https://openaccess.thecvf.com/content_cvpr_2018/html/Zoph_Learning_Transferable_Architectures_CVPR_2018_paper.html

30. Carion N. et al. *End-to-End Object Detection with Transformers*. — *ECCV*, 2020. - https://link.springer.com/chapter/10.1007/978-3-030-58452-8_13

31. Liu Z. et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. — *ICCV*, 2021. - https://openaccess.thecvf.com/content/ICCV2021/papers/Liu_Swin_Transformer_Hierarchical_Vision_Transformer_Using_Shifted_Windows_ICCV_2021_paper.pdf

32. Xie S. et al. *Aggregated Residual Transformations for Deep Neural Networks*. — *CVPR*, 2017. - https://openaccess.thecvf.com/content_cvpr_2017/papers/Xie_Aggregated_Residual_Transformations_CVPR_2017_paper.pdf

33. Howard A. G. et al. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. — arXiv:1704.04861, 2017.

34. Iandola F. N. et al. *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size*. — arXiv:1602.07360, 2016. - https://arxiv.org/abs/1602.07360

35. Zhang H. et al. *Mixup: Beyond Empirical Risk Minimization*. — *ICLR*, 2018. - https://arxiv.org/abs/1710.09412

36. Yun S. et al. *CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features*. — *ICCV*, 2019. - https://openaccess.thecvf.com/content_ICCV_2019/papers/Yun_CutMix_Regularization_Strategy_to_Train_Strong_Classifiers_With_Localizable_Features_ICCV_2019_pap

er.pdf

37. Cubuk E. D. et al. *AutoAugment: Learning Augmentation Policies from Data*. — *CVPR*, 2019. - https://openaccess.thecvf.com/content_CVPR_2019/papers/Cubuk_AutoAugment_Learning_Augmentation_Strategies_From_Data_CVPR_2019_paper.pdf

38. Zoph B. et al. *Neural Architecture Search with Reinforcement Learning*. — *ICLR*, 2017. - https://arxiv.org/abs/1611.01578

39. Liu H. et al. *DARTS: Differentiable Architecture Search*. — *ICLR*, 2019. - https://arxiv.org/abs/1806.09055

40. Real E. et al. *Regularized Evolution for Image Classifier Architecture Search*. — *AAAI*, 2019. - https://ojs.aaai.org/index.php/AAAI/article/view/4405

41. Han S. et al. *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. — *ICLR*, 2016. - https://arxiv.org/abs/1510.00149

42. Jacob B. et al. *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*. — *CVPR*, 2018. - https://openaccess.thecvf.com/content_cvpr_2018/papers/Jacob_Quantization_and_Training_CVPR_2018_paper.pdf

43. Howard A. et al. *Searching for MobileNetV3*. — *ICCV*, 2019. - https://openaccess.thecvf.com/content_ICCV_2019/papers/Howard_Searching_for_MobileNetV3_ICCV_2019_paper.pdf

44. Sandler M. et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*.— *CVPR*, 2018. - https://openaccess.thecvf.com/content_cvpr_2018/papers/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.pdf

45. Chen L. C. et al. *Rethinking Atrous Convolution for Semantic Image Segmentation*. — arXiv:1706.05587, 2017. - https://arxiv.org/abs/1706.05587

46. «Ощадливий стартап: Як сучасні підприємці використовують безперервні інновації для створення радикально успішного бізнесу» – Ерік Ріс

47. " Гнучке управління проектами: Створення інноваційних продуктів" – Джим Шаубер - https://books.google.com.ua/books?hl=uk&lr=&id=VuFpkztwPaUC&oi=fnd&pg=PT9&dq=%22Agile+project+management:+Creating+innovative+products+%22++Jim+Schauber&ots=CwDJZfZl9G&sig=zcOrmoeI7xFAyqXDPDeEZMot3k&redir_esc=y#v=onepage&q&f=false

48. «Розробка програмного забезпечення в Google» - Тітус Вінтерс, Том Маншрек, Хайрум Райт https://books.google.com.ua/books?hl=uk&lr=&id=WXTTDwAAQBAJ&oi=fnd&pgPT14&dq=%22Software+Development+at+Google++Titus+Winters,+Tom+Manshreck,+Hyrum+Wright&ots=pVpnc2Lha3&sig=lASoOBdrQhBPEpy3zUZ6vlhCVM&redir_esc=y#v=onepage&q=%22Software%20Development%20at%20Google%20%20Titus%20Winters%2C%20Tom%20Manshreck%2C%20Hyrum%20Wright&f=false

49. «Управління вимогами до програмного забезпечення: Підхід, заснований на використанні кейсів» - Дін Леффінгвелл, Дон Відріг

50. Вовк С.М., Гнатушенко В.В., Бондаренко М.В. *Методи обробки зображень та комп'ютерний зір.* - https://it.nmu.org.ua/ua/scientific_method_materials/books/MOZKZ.pdf

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розпізнавання об’єктів, що зображені на фото.

2 Підстава для розробки

Підставою для розробки служить завдання на кваліфікаційну бакалаврську роботу №47-02 від 17.01.2025 року, видане на кафедрі кібербезпеки та програмного забезпечення.

3 Мета та призначення розробки

Метою кваліфікаційної бакалаврської роботи є розробка програмного забезпечення автоматизованого розподілення за вмістом графічної інформації

4 Джерела розробки

Джерелом цієї кваліфікаційної бакалаврської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– систему розпізнавання об’єктів, що зображенні на фотографії;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер AMD Ryzen 3 5300U with Radeon Graphics 2.60 GHz або сумісні з ним.

5.8.2 Мова програмування

Програму розроблено на мовах програмування Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД.

7 Перелік документів, що розробляються

- Структурна схема системи. – 1 аркуш
- Функціональна схема системи. – 1 аркуш
- Діаграма процесів. – 1 аркуш
- Блок-схема алгоритму роботи програми. – 1 аркуш
- Пояснювальна записка. – 54 аркуша

8 Етапи розробки

8.1 Збір і обробка інформації по темі кваліфікаційної бакалаврської роботи. Постановка задачі на виконання кваліфікаційної бакалаврської роботи (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень кваліфікаційної бакалаврської роботи.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання кваліфікаційної бакалаврської роботи на попередній захист 24.05.2025 р.

11.2 Подання кваліфікаційної бакалаврської роботи на захист 05.06.2025 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи

за першим (бакалаврським) рівнем вищої освіти

_____ О. М. Дрєєв

Програмне забезпечення автоматизованого розподілення за вмістом графічної інформації

Лістинг програми

Код документу 12

Носій: CD/DVD-диск

Загальна кількість аркушів: 9

Літера: РП

Кропивницький – 2025 року

Gui_app

```
import os
import threading
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk, ImageDraw, ImageFont
from detector import ObjectDetector, Classifier, Segmenter, ImageAnalyzer
from download_assets import download_all_assets

BG = "#F7F7F7"
PANEL = "#FFFFFF"
BORDER = "#E0E0E0"
TEXT_COLOR = "#333"
FONT_PLACEHOLDER = ("Roboto", 16, "bold")
FONT_TEXT = ("Roboto", 12, "bold")

class App(tk.Tk):
    def __init__(self):
        super().__init__()
        self.iconbitmap(os.path.join("UI", "icon.ico"))
        self.title("Image Analyzer")
        self.geometry("1100x600")
        self.configure(bg=BG)

        self.canvas = tk.Canvas(self, bg=BG, highlightthickness=0)
        self.canvas.pack(fill="both", expand=True)

        self.canvas.create_text(550, 40, text="Image Analyzer", fill="#222",
font=("Roboto", 22, "bold"))

        self.canvas.create_rectangle(10, 80, 510, 590, outline=BORDER,
fill=PANEL, width=2)
        self.canvas.create_rectangle(520, 80, 820, 590, outline=BORDER,
fill=PANEL, width=2)
        self.canvas.create_rectangle(830, 80, 1090, 590, outline=BORDER,
fill=PANEL, width=2)

        self.img_canvas = tk.Canvas(self, bg=PANEL, width=490, height=500,
highlightthickness=0)
        self.img_canvas.place(x=15, y=85)
        self.img_canvas.create_text(245, 250, text="No image selected",
fill="#AAA", font=FONT_PLACEHOLDER)

        self.result_panel = tk.Canvas(self, bg=PANEL, width=290, height=500,
highlightthickness=0)
        self.result_panel.place(x=525, y=85)
        self.result_text_id = self.result_panel.create_text(
            145, 10, text="", fill=TEXT_COLOR, font=FONT_TEXT, anchor="n",
width=270)

        self.status = self.canvas.create_text(960, 570, text="Loading
models...", fill="#888", font=FONT_TEXT)

        self.button_files = [
            "choose_img_button.png",
            "advanced_analysis_button.png",
            "exit_button.png"
        ]
        self.actions = [self.choose, self.advanced, self.quit]
        self.buttons = []
        self.buttons_images = []
        self.original_images = []

        for i, (file, action) in enumerate(zip(self.button_files,
```

```
self.actions)):
            path = os.path.join("UI", file)
            img = Image.open(path).resize((240, 60), Image.Resampling.LANCZOS)
            img_tk = ImageTk.PhotoImage(img)
            self.original_images.append(img)
            self.buttons_images.append(img_tk)

            x, y = 960, 130 + i * 80
            btn = self.canvas.create_image(x, y, image=img_tk)
            self.canvas.tag_bind(btn, "<Button-1>", lambda e, idx=i:
self.animate_button(idx))
            self.buttons.append(btn)
        self.disabled_images = {}
        self.disable_buttons([0, 1])
        self.save_button_img = None
        self.save_button = None

        threading.Thread(target=self.setup, daemon=True).start()
    def disable_buttons(self, indices):
        for idx in indices:
            img_path = os.path.join("UI", self.button_files[idx])
            im = Image.open(img_path).convert("RGBA").resize((240, 60),
Image.Resampling.LANCZOS)
            alpha = im.split()[3].point(lambda p: int(p * 0.4))
            im.putalpha(alpha)
            ph = ImageTk.PhotoImage(im)
            self.disabled_images[idx] = ph
            self.canvas.itemconfig(self.buttons[idx], image=ph)
            self.canvas.itemconfig(self.buttons[idx], state="disabled")
    def enable_buttons(self, indices):
        for idx in indices:
            img_path = os.path.join("UI", self.button_files[idx])
            im = Image.open(img_path).resize((240, 60),
Image.Resampling.LANCZOS)
            ph = ImageTk.PhotoImage(im)
            self.buttons_images[idx] = ph
            self.canvas.itemconfig(self.buttons[idx], image=ph)
            self.canvas.itemconfig(self.buttons[idx], state="normal")
    def setup(self):
        download_all_assets()
        self.det = ObjectDetector()
        self.clf = Classifier()
        self.seg = Segmenter()
        self.ia = ImageAnalyzer()
        self.canvas.after(0, lambda: self.canvas.itemconfig(self.status,
text="Models loaded", fill="#4CAF50"))
        self.canvas.after(0, lambda: self.enable_buttons([0, 1]))
    def animate_button(self, index):
        btn = self.buttons[index]
        steps = 5
        scale_down = 0.92
        scale_up = 1.0

        def scale(step):
            factor = scale_down + (scale_up - scale_down) * step / steps
            img = self.original_images[index].resize(
                (int(240 * factor), int(60 * factor)), Image.Resampling.LANCZOS)
            img_tk = ImageTk.PhotoImage(img)
            self.buttons_images[index] = img_tk
            self.canvas.itemconfig(btn, image=img_tk)
            x, y = 960, 130 + index * 80
            self.canvas.coords(btn, x, y)
        def animate_down(i=0):
            if i <= steps:
                scale(i)
                self.after(10, lambda: animate_down(i + 1))
            else:
                animate_up(steps)
        def animate_up(i):
```

```
            if i >= 0:
                scale(i)
                self.after(10, lambda: animate_up(i - 1))
            else:
                self.actions[index]()
        animate_down()
    def choose(self):
        path = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg
*.png *.jpeg")])
        if not path:
            return
        im = Image.open(path)
        im.thumbnail((480, 500))
        ph = ImageTk.PhotoImage(im)
        self.img_canvas.image = ph
        self.img_canvas.delete("all")
        self.img_canvas.create_image(0, 0, anchor="nw", image=ph)
        self.path = path
        det = self.det.detect(path)
        if isinstance(det, list) and det and isinstance(det[0], dict):
            unique_labels = sorted(set(d["label"] for d in det))
            txt = "Detected:\n" + ("\n".join(f"• {label}" for label in
unique_labels) if unique_labels else "None")
        else:
            txt = "Detected:\n" + ("None" if not det else "• " + str(det))
        self.result_panel.itemconfig(self.result_text_id, text=txt)
        self.show_save_button()
    def advanced(self):
        if not hasattr(self, "path"):
            return
        cls = self.clf.classify(self.path)
        results = self.ia.perform_extended_analysis(self.path)
        det = results.get("detected_objects", [])
        object_counts = results.get("object_counts", {})
        object_boxes = [obj['coords'] for obj in det]
        unique_labels = set(d["label"] for d in det)
        txt = f"Classification: {cls}\n\n"
        txt += "Detected:\n" + "\n".join(f"• {label}" for label in
unique_labels) + "\n\n"
        txt += "Object Counts:\n" + (
            "\n".join(f"- {k}: {v}" for k, v in object_counts.items()) if
object_counts else "None")
        self.result_panel.itemconfig(self.result_text_id, text=txt)
        try:
            im = Image.open(self.path).convert("RGB")
            draw = ImageDraw.Draw(im)
            font_size = 10
            font = ImageFont.truetype("arial.ttf", font_size)
            for i, box in enumerate(object_boxes):
                if len(box) == 4:
                    x1, y1, x2, y2 = box
                    draw.rectangle([x1, y1, x2, y2], outline="red", width=4)
                    if i < len(det):
                        label = det[i]["label"]
                        draw.text((x1 + 5, y1 + 5), label, fill="red",
font=font)
            im.thumbnail((480, 500), Image.LANCZOS)
            ph = ImageTk.PhotoImage(im)
            self.img_canvas.delete("all")
            self.img_canvas.create_image(0, 0, anchor="nw", image=ph)
            self.img_canvas.image = ph
        except Exception as e:
            print("Помилка при запускі та рисуванні:", e)
        self.show_save_button()
    def show_save_button(self):
        if self.save_button:
            return
        img_path = os.path.join("UI", "save_to_file_button.png")
        img = Image.open(img_path).resize((180, 40), Image.Resampling.LANCZOS)
```

```
        self.save_button_img = ImageTk.PhotoImage(img)
        x, y = 145, 470
        self.save_button = self.result_panel.create_image(x, y,
image=self.save_button_img)
        self.result_panel.tag_bind(self.save_button, "<Button-1>", lambda e:
self.save_result())
    def save_result(self):
        txt = self.result_panel.itemcget(self.result_text_id, "text")
        if not txt.strip():
            return
        path = filedialog.asksaveasfilename(defaultextension=".txt",
filetypes=[("Text files", "*.txt")])
        if path:
            with open(path, "w", encoding="utf-8") as f:
                f.write(txt)

if __name__ == "__main__":
    App().mainloop()
```

detector

```
import requests
import torch
import os
import clip
from PIL import Image, ImageDraw
from ultralytics import YOLO
from torchvision import models, transforms
from download_assets import MODELS_DIR
class ObjectDetector:
    def __init__(self):
        model_path = os.path.join(MODELS_DIR, "yolov11l.pt")
        self.model = YOLO(model_path)
    def detect(self, image_path):
        results = self.model(image_path, verbose=False)[0]
        names = results.names
        detections = []
        for box in results.boxes:
            cls_id = int(box.cls.item())
            label = names[cls_id]
            coords = box.xyxy[0].tolist()  # [x1, y1, x2, y2]
            detections.append({"label": label, "coords": coords})
        return detections
    def draw_boxes(self, image_path, detections):
        image = Image.open(image_path).convert("RGB")
        draw = ImageDraw.Draw(image)
        for det in detections:
            x1, y1, x2, y2 = det["coords"]
            draw.rectangle([x1, y1, x2, y2], outline="red", width=3)
            draw.text((x1 + 5, y1 + 5), det["label"], fill="red")
        return image
class Classifier:
    def __init__(self):
        self.api_user = "622806118"
        self.api_secret = "wmFnKWhBqe4gLrzgPRYiuAKtNc8VvHNk"
        self.url = "https://api.sightengine.com/1.0/check.json"
    def classify_type(self, type_dict):
        if not type_dict:
            print("Could not determine image type (empty dictionary).")
            return None
        most_likely_type = max(type_dict, key=type_dict.get)
        print("Image type:", most_likely_type)
        return most_likely_type
    def classify(self, image_path):
        with open(image_path, "rb") as image_file:
            files = {'media': image_file}
            params = {
                'models': 'type',
                'api_user': self.api_user,
                'api_secret': self.api_secret
            }

            response = requests.post(self.url, files=files, data=params)

        if response.status_code == 200:
            result = response.json()
            type_info = result.get("type")
            return self.classify_type(type_info)
        else:
            print(f"Error: {response.status_code}, {response.text}")
            return None
class Segmenter:
    def __init__(self):
        weight_path = os.path.join(MODELS_DIR, "deeplabv3_resnet101_coco-
586e9e4e.pth")
        self.model = models.segmentation.deeplabv3_resnet101(aux_loss=True)
        state_dict = torch.load(weight_path, map_location="cpu")
```

```
        if "model" in state_dict:
            state_dict = state_dict["model"]

        self.model.load_state_dict(state_dict)
        self.model.eval()

        self.transform = transforms.Compose([
            transforms.Resize(256),
            transforms.CenterCrop(224),
            transforms.ToTensor(),
        ])
    def segment(self, image_path):
        image = Image.open(image_path).convert("RGB")
        tensor = self.transform(image).unsqueeze(0)
        with torch.no_grad():
            output = self.model(tensor)["out"][0]
        mask = output.argmax(0).byte().cpu().numpy()
        return Image.fromarray(mask)
class CLIPContextAnalyzer:
    def __init__(self):
        self.device = "cuda" if torch.cuda.is_available() else "cpu"
        self.model, self.preprocess = clip.load("ViT-B/32", device=self.device)

    def analyze(self, image_path):
        image = Image.open(image_path)
        image_input = self.preprocess(image).unsqueeze(0).to(self.device)
        categories = ["person", "tree", "building", "nature", "car", "animal",
"flower", "camera", "laptop", "phone"]
        import clip
        text_inputs = torch.cat([clip.tokenize(f"a photo of a {category}") for
category in categories]).to(self.device)
        with torch.no_grad():
            image_features = self.model.encode_image(image_input)
            text_features = self.model.encode_text(text_inputs)
        image_features /= image_features.norm(dim=-1, keepdim=True)
        text_features /= text_features.norm(dim=-1, keepdim=True)
        similarity = (image_features @ text_features.T).squeeze(0)
        values, indices = similarity.cpu().topk(3)
        return [categories[idx] for idx in indices]


class ImageAnalyzer:
    def __init__(self):
        self.detector = ObjectDetector()
        self.classifier = Classifier()
        self.segmenter = Segmenter()
        self.clip_analyzer = CLIPContextAnalyzer()
    def perform_extended_analysis(self, image_path):
        detections = self.detector.detect(image_path)
        classification = self.classifier.classify(image_path)
        segmentation = self.segmenter.segment(image_path)
        clip_analysis = self.clip_analyzer.analyze(image_path)
        count_by_label = {}
        for det in detections:
            label = det["label"]
            count_by_label[label] = count_by_label.get(label, 0) + 1
        results = {
            "detected_objects": detections,
            "classification": classification,
            "segmentation": segmentation,
            "clip_analysis": clip_analysis,
            "object_counts": count_by_label
        }
        return results
    def draw_annotated_image(self, image_path):
        detections = self.detector.detect(image_path)
        return self.detector.draw_boxes(image_path, detections)
```

download_assets

```
import os
import urllib.request

BASE_DIR = os.getcwd()
MODELS_DIR = os.path.join(BASE_DIR, "models")

def ensure_dirs():
    os.makedirs(MODELS_DIR, exist_ok=True)

def download_file(url, dest):
    if not os.path.exists(dest) or not is_file_valid(dest, url):
        print(f"▼ Downloading {os.path.basename(dest)}...")
        urllib.request.urlretrieve(url, dest)
        if is_file_valid(dest, url):
            print(f"+ Downloaded: {os.path.basename(dest)}")
        else:
            print(f"- Download failed: {os.path.basename(dest)}. File is
corrupted.")
            os.remove(dest)
            download_file(url, dest)
    else:
        print(f"* Already exists: {os.path.basename(dest)}")
def is_file_valid(file_path, url):
    try:
        remote_size = int(urllib.request.urlopen(url).headers['Content-Length'])
        local_size = os.path.getsize(file_path)
        return remote_size == local_size
    except Exception as e:
        print(f"Помилка при перевірці розміру файлу: {e}")
        return False
def download_all_assets():
    ensure_dirs()
    download_file(
        "https://huggingface.co/Ultralytics/YOLO11/resolve/main/yolo11l.pt",
        os.path.join(MODELS_DIR, "yolov11l.pt")
    )
    deeplabv3_url =
"https://download.pytorch.org/models/deeplabv3_resnet101_coco-586e9e4e.pth"
    deeplabv3_dest = os.path.join(MODELS_DIR, "deeplabv3_resnet101_coco-
586e9e4e.pth")
    download_file(deeplabv3_url, deeplabv3_dest)
```