

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСОБІВ
РОЗПІЗНАВАННЯ ГРАФІЧНИХ ОБРАЗІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF
GRAPHIC IMAGE RECOGNITION MEANS

Виконав: студент 2 курсу, групи КНТ-142м

Спеціальності 121 Інженерія програмного

(код і найменування спеціальності)

забезпечення

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

РЕДЬКО М.В.

(ПРІЗВИЩЕ та ініціали)

Керівник ОЛІЙНИК А.О.

(ПРІЗВИЩЕ та ініціали)

Рецензент КОЦУР М.І.

(ПРІЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет КНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

РЕДЬКА Максима Вадимовича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація засобів розпізнавання графічних образів. Research and Software Implementation of Graphic Image Recognition Means

керівник проєкту (роботи) д.т.н., професор, ОЛІЙНИК Андрій Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “ 14 ” листопада 2023 року № 437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Матеріали і методи. 3. Опис програми. 4. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	ОЛІЙНИК А.О., професор		
Нормоконтроль	КАМІНСЬКА Ж.К., асистент		

7. Дата видачі завдання “ 16 ” жовтня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій	2 тиждень	Розділ 2
	розробки		
4	Розробка структури програми.	3 тиждень	Розділ 3
5	Розробка програми.	4-7 тижні	Розділи 3,4
6	Тестування та експериментальне дослідження	8 тиждень	Розділ 4
	програмного забезпечення.		
7	Оформлення пояснювальної записки та документів	9-10 тижні	Додатки
	до неї.		
8	Нормоконтроль та рецензування.	11 тиждень	
9	Захист роботи.	12 тиждень	

Студент(ка)

(підпис) Максим РЕДЬКО
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

(підпис) Андрій ОЛІЙНИК
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
83 с., 27 рис., 4 табл., 3 дод., 16 джерел.

PYTHON, PYCHARM, НЕЙРОННІ МЕРЕЖІ, ПРОГРАМНЕ
ЗАБЕЗПЕЧЕННЯ, РОЗПІЗНАВАННЯ ОБРАЗІВ.

Об'єкт дослідження – процес розробки програмного забезпечення для розпізнавання образів.

Предмет дослідження – методи та програмні засоби для розв'язання завдань розпізнавання графічних образів.

Метою роботи є дослідження та розробка програмного забезпечення для розпізнавання графічних образів.

Матеріали, методи та технічні засоби: мова програмування Python, середовище розробки PyCharm, програмна бібліотека OpenCV.

Результати. Розроблено програмне забезпечення для розпізнавання графічних образів за допомогою мови програмування Python та середовища розробки PyCharm.

Практична цінність роботи полягає у розробці програмного забезпечення, що реалізує методи розпізнавання образів на основі штучного інтелекту та дозволяє розв'язувати задачі розпізнавання графічних образів.

Висновки. Виконано проєктування програмного забезпечення для розпізнавання образів. Розроблено програмне забезпечення для розпізнавання образів. Здійснено тестування розробленого програмного забезпечення для розпізнавання образів.

Галузь використання – програмні засоби для розпізнавання образів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 83 pages, 27 figures, 4 tables, 3 appendixes, 16 sources.

PYTHON, PYCHARM, NEURAL NETWORKS, SOFTWARE, PATTERN RECOGNITION.

The object of research is the process of developing software for pattern recognition.

The subject of the research is methods and software tools for solving graphic image recognition tasks.

The purpose of this work is to develop the software for the recognition of graphic images.

Materials, methods and technical tools: Python programming language, PyCharm development environment, OpenCV software library.

Results. The software tools for recognizing graphic patterns using Python programming language and PyCharm development environment have been developed.

The practical value of the work lies in the development of software that implements pattern recognition methods based on artificial intelligence and allows you to solve problems of graphic pattern recognition.

Conclusions. The software for solving pattern recognition problems has been designed. The software for solving pattern recognition problems has been developed. The developed software for solving pattern recognition problems has been tested.

Scope of use – software tools for pattern recognition.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ.....	9
1 Аналіз предметної області.....	11
1.1 Методи та засоби розпізнавання образів.....	11
1.2 Аналіз програмного забезпечення для розпізнавання образів.....	21
1.3 Висновки за розділом 1	30
2 Матеріали і методи	32
2.1 Вибір мови програмування	32
2.2 Вибір середовища та інших засобів розробки для створення програмного забезпечення для розпізнавання графічних образів.....	34
2.3 Розробка моделі взаємодії користувача з програмним забезпеченням для розпізнавання графічних образів	36
2.4 Висновки за розділом 2	38
3 Опис програми	39
3.1 Структура програмного забезпечення для розпізнавання графічних образів	39
3.2 Функціонування програмного забезпечення для розпізнавання графічних образів.....	41
3.3 Особливості синтезу розпізнавальних моделей для реалізації програмного забезпечення для розпізнавання графічних образів.....	42
3.4 Висновки за розділом 3	46
4 Експлуатація, тестування та експериментальне дослідження програми.....	47
4.1 Призначення й умови застосування програми	47
4.2 Характеристики програми для розпізнавання графічних образів	48
4.3 Інструкція по експлуатації програми.....	48
4.3.1 Звернення до програми.....	48
4.3.2 Вхідні й вихідні дані	49
4.3.3 Повідомлення.....	49

4.4 Виконання програмного забезпечення для розпізнавання графічних образів	49
4.5 Тестування та дослідження програмного забезпечення для розпізнавання графічних образів.....	51
4.6 Висновки за розділом 4	52
Висновки	53
Перелік джерел посилання	56
Додаток А Технічне завдання	58
Додаток Б Фрагмент тексту програми	62
Додаток В Слайди презентації	76

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface, прикладний програмний інтерфейс;

CNN – Convolutional Neural Network, згортова нейронна мережа;

ГА – генетичний алгоритм;

НМ – нейронна мережа;

ПЗ – програмне забезпечення;

ШІ – штучний інтелект;

ШНМ – штучні нейронні мережі.

ВСТУП

Використання програмних засобів для розпізнавання образів має велику актуальність в сучасному світі і знаходить застосування в різних сферах. Розпізнавання образів є важливою складовою для розвитку алгоритмів машинного навчання та штучного інтелекту, допомагаючи програмним системам бачити та розпізнавати об'єкти навколо штучних програмно-апаратних інтелектуальних систем [1]-[7].

Програмні засоби розпізнавання образів допомагають лікарям у виявленні та діагностиці різних захворювань на основі зображень, таких як рентгенограми, МРТ, КТ та інші, що в свою чергу дозволяє розробляти більш ефективні методи лікування та виявлення хвороб на ранніх стадіях. Програмні засоби управління автономними автомобілями використовують системи розпізнавання образів для виявлення інших транспортних засобів, пішоходів та об'єктів на дорозі для безпечного руху та уникнення аварій. Промислові підприємства використовують програмні системи розпізнавання образів для контролю якості виробів, виявлення дефектів та автоматизації виробничих процесів [1]-[7].

Проте деякі програмні засоби при розв'язанні завдань розпізнавання образів можуть робити суттєві помилки, що обумовлено використанням неефективних моделей та методів розпізнавання образів. Інші програми вимагають великої обчислювальної потужності, що може бути проблемою для вбудованих систем або мобільних пристроїв. Ефективність роботи деяких програмних систем розпізнавання образів може залежати від умов навколишнього середовища, таких як освітлення, погодні умови, кути огляду тощо, що може призвести до низької стійкості подібних систем в різних ситуаціях [1]-[7].

Тому актуальною є розробка програмного забезпечення для розпізнавання графічних образів.

У дипломній кваліфікаційній роботі магістра розв'язується актуальне завдання розробки програмного забезпечення для розпізнавання графічних образів.

Для досягнення поставленої мети у кваліфікаційній роботі магістра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для розпізнавання образів;
- розробити модель взаємодії користувача з програмним забезпеченням для розпізнавання образів;
- здійснити проєктування програмного забезпечення для розв'язання завдань розпізнавання графічних образів;
- створити програмне забезпечення для розв'язання завдань розпізнавання графічних образів;
- виконати тестування розробленого програмного забезпечення для розпізнавання графічних образів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Методи та засоби розпізнавання образів

Розпізнавання образів представляє собою процес, що забезпечує здатність комп'ютерів визначати структурні відповідності в наборах даних та використовувати цю інформацію для прийняття рішень або прогнозування шляхом використання різних алгоритмів обробки інформації. Ця технологія використовується у сучасних системах штучного інтелекту та глибокого навчання [1]-[5].

Розпізнавання образів застосовується для автоматизації процесів виявлення та аналізу шаблонів у великих наборах даних [1]-[5]. Оскільки існує велика кількість підходів до ідентифікації цих структурних відповідей, вибір конкретного методу залежить від різних факторів [1]-[5], зокрема, постанови завдання (постанова конкретного завдання розпізнавання образів), об'єму та різноманітності даних, які доступні для аналізу (цифрові дані, часові ряди або точкові дані), відомостей про обчислювальну потужність та обсяг сховища, доступних для обробки та зберігання даних [1]-[5].

Отже, розпізнавання образів є універсальним підходом, який адаптується до конкретних завдань і доступних ресурсів для ефективного вирішення завдань обробки даних [1]-[5].

Розпізнавання образів є процесом аналізу вхідних даних з метою визначення та ідентифікації різних шаблонів або закономірностей. Ця процедура використовується для прогнозування майбутніх подій, категоризації інформації та поліпшення процесів прийняття рішень. Є два основних підходи до розпізнавання образів: дослідницький та описовий [1]-[5].

Дослідницьке розпізнавання образів спрямоване на ідентифікацію шаблонів даних у загальному контексті, тоді як описове розпізнавання шаблонів розпочинається з класифікації виявлених шаблонів. Таким чином,

розпізнавання образів може працювати в обох сценаріях, використовуючи різні методи залежно від завдання та форми даних [1]-[5].

Розпізнавання образів є необхідною складовою для інтелектуальних систем. В основі цих систем є комп'ютерні алгоритми, спроектовані для аналізу та інтерпретації різних видів даних, таких як текст, зображення та аудіофайли [1]-[5].

Відзначимо, що розпізнавання образів є більш широким поняттям, ніж комп'ютерний зір, який фокусується лише на розпізнаванні зображень. Воно включає в себе автоматизовану і машинну обробку, опис, класифікацію та групування шаблонів, і має велике застосування в різних галузях, від біології та психології до медицини, маркетингу та штучного інтелекту [1]-[5].

Поняття образу може охоплювати будь-яку важливу сутність, яку необхідно визначити і виконати її ідентифікацію. У цьому контексті образи виявляють подібні тенденції або структури в різних формах даних. Наприклад, образом може бути відбиток пальця, написане слово в рукописі, обличчя особи або аудіозапис мовного сигналу. Образ можна спостерігати фізично, якщо мова йде про зображення або відео, або аналітично, використовуючи математичні методи та статистичні алгоритми [1]-[5].

При розв'язанні завдання неконтрольованої класифікації, модель розпізнавання образів призначається до невизначеного класу (дослідженого). Зазвичай завдання розпізнавання образів ставляться як завдання класифікації або категоризації. Класи можуть бути визначені або встановлені системою (контрольована класифікація) або вивчені з подібних моделей (неконтрольована класифікація). Розпізнавання образів постійно розвивається завдяки виникненню програм, які стають не лише більш складними, але й вимагають більше обчислювальних потужностей [1]-[5].

Мета розпізнавання образів ґрунтується на ідеї, що прийняття рішень людиною якимось чином пов'язане із розпізнаванням образів. Тому основна мета розпізнавання образів полягає в поліпшенні розуміння цих складних

механізмів у процесах прийняття рішень та автоматизації цих електронних функцій за допомогою комп'ютерів [1]-[5].

Розпізнавання образів описується як аналіз спроможності комп'ютерних програм та апаратного забезпечення до спостереження оточуючого оточення, здатність навчатися відрізняти цікаві об'єкти від їхнього фону та приймати точні рішення щодо класифікації шаблонів [1]-[5]. Під час процесу розпізнавання предмети класифікуються по певних категоріях. У цій сфері існують різні визначення розпізнавання образів через її значущий обсяг та постійний розвиток [1]-[5]. Інше визначення розпізнавання образів розглядає його як наукову галузь, метою якої є класифікація об'єктів на основі декількох категорій або класів. Розпізнавання образів також становить складову більшості систем штучного інтелекту, спрямованих на прийняття рішень [1]-[5].

Розпізнавання образів може бути описано як процес зведення інформації, відображення інформації або позначення інформації в іншій формі. У галузі обчислювальної техніки, розпізнавання образів включає в себе процес порівняння інформації, що зберігається в базі даних, з характеристиками представлених даних [1]-[5].

Розпізнавання образів тісно пов'язане із штучним інтелектом (ШІ). Штучний інтелект спрямований на створення систем, які можуть моделювати розумову діяльність людини і імітувати її поведінку. В частині ШІ, спрямованій на розпізнавання образів, системи призначені для виконання складних завдань, таких як розпізнавання облич і об'єктів. Таким чином, розпізнавання образів становить важливу частину сфери штучного інтелекту [1]-[5].

У наш час з розвитком методів та засобів штучного інтелекту, розпізнавання образів та машинне навчання використовуються для створення моделей машинного навчання, які можуть швидко та точно розпізнавати та виділяти унікальні образи та шаблони в наборах даних [1]-[5].

Двома домінуючими підходами до розпізнавання образів є статистичне розпізнавання образів (або теорія прийняття рішень) та синтаксичне (або структурне) розпізнавання образів. Третій важливий підхід базується на теорії штучних нейронних мереж (ШНМ). Важливо розуміти, що немає універсального методу, який би був ідеальним для розв'язання всіх завдань розпізнавання образів. Тому в залежності від конкретної проблеми можуть використовуватися різні методи або їх комбінації [1]-[5].

Статистичне розпізнавання образів було розроблено та досліджено задовго до появи нейронних мереж. У статистичному розпізнаванні образів основний принцип полягає в тому, що образи групуються за їх характеристиками, і кількість цих характеристик визначає простір ознак. Важливо, щоб характеристики були обрані так, щоб вони не перекривалися та допомагали утворювати кластери образів. Метод розпізнавання образів та основі статистичного моделювання навчається на основі наданої інформації та використовує шаблони для подальшого розпізнавання та навчання. Основною метою статистичного розпізнавання образів є вибір таких характеристик, за якими можна класифікувати вектори образів у вимірному просторі [1]-[5].

Синтаксичне розпізнавання образів застосовується для вирішення когнітивних завдань, пов'язаних із складними шаблонами, які можна розглядати з ієрархічної точки зору. У цьому підході синтаксична модель базується на елементарних моделях низького рівня, наприклад, символах алфавіту. Модель визначається у термінах цих елементарних компонентів та їх взаємодії. Один із прикладів цієї взаємодії – спосіб об'єднання їх у слова та речення. Крім того, структурний аналіз синтаксичних шаблонів надає опис того, як цей конкретний шаблон будується з елементарних шаблонів. Тому цей підхід використовується для випадків, коли образу мають унікальну структуру, наприклад, сигнали кардіограми або текстуровані зображення, що формуються за різними правилами. Важливо враховувати, що синтаксичний підхід може призвести до значних обчислювальних складнощів, оскільки

імовірність комбінаторного розмаху вимагає великих наборів даних і обчислювальних ресурсів для подальших досліджень [1]-[5].

Метод зіставлення зі зразком є одним із найпростіших методів розпізнавання образів. Цей метод ґрунтується на загальній операції, що дозволяє виміряти подібність між двома сутностями. Зіставлення образів намагається виявити подібності у зразку, базуючись на еталонному образі. Зазвичай метод зіставлення зі зразком використовується для ідентифікації невеликих ділянок зображень, що відповідають зразкам. Метод зіставлення зі зразком застосовується в таких областях, як медична обробка зображень, контроль якості на виробництві, навігація за допомогою програмного забезпечення і розпізнавання обличчя [1]-[5].

Розпізнавання образів за допомогою нейронних мереж є широко використовуваним методом у наш час. Нейронні мережі складаються з паралельних обчислювальних одиниць, нейронів, які імітують прийняття рішень людиною. Їх можна розглядати як масивні паралельні комп'ютерні системи з багатьма простими процесорами і взаємозв'язками (нейронами). Глибоке навчання, яке використовує глибоку згортову нейронну мережу (Convolutional Neural Network, CNN), є найбільш популярним методом машинного навчання для завдань класифікації та розпізнавання образів [1]-[5]. Розпізнавання образів нейронною мережею виявляє перевагу, оскільки може неодноразово змінювати ваги в ітераційному процесі. Глибоке навчання нині визнано найефективнішим методом розпізнавання образів [1]-[5].

Використання гібридного підходу до розпізнавання образів засновано на поєднанні машинного навчання та різних методів розпізнавання образів, що дозволяє досягти оптимальних результатів або використовувати найкращі та ефективніші шаблонні детектори [1]-[5]. Саме тому багато програмних засобів розпізнавання образів використовують гібридні підходи для покращення продуктивності систем розпізнавання образів, адаптуючи їх під конкретні програми, залежно від типу та наявності даних. Наприклад, методи глибокого навчання можуть давати різні результати, але вони вимагають великої

обчислювальної потужності, тоді як більш прості математичні методи, як правило, є більш ефективними. Крім того, перед використанням моделей розпізнавання образів часто використовуються методи попередньої обробки даних. Використання гібридних підходів допомагає покращити загальну продуктивність програмного забезпечення розпізнавання образів [1]-[5].

Створення програмних систем розпізнавання образів передбачає виконання декількох кроків, зокрема, збір та початкову обробку даних, створення моделі подання даних та прийняття рішень. Процес розпізнавання образів можна розглядати наступним чином [1]-[5]:

- збір цифрової інформації;
- очищення даних від надмірного шуму та перешкод;
- аналіз і визначення важливих або інформативних атрибутів;
- групування об'єктів у логічні сегменти або класи (побудова розпізнавальної моделі);
- використання отриманої інформації для подальшого розпізнавання образів та вирішення інших проблем аналізу даних [1]-[5].

При створення програмної системи розпізнавання образів на етапі збору інформації виконується активний процес отримання сирової вхідної інформації або сигналів для подальшого аналізу. На етапі виділення ознак виконується визначення найбільш суттєвих ознак або шаблонів у сирих даних. Етап класифікації пов'язаний з призначенням міток даним на основі виявлених шаблонів або характеристик. Етап подальшої обробки є процесом уточнення результатів класифікації з метою підвищення точності та зменшення помилок [1]-[5].

Розроблення програмних систем розпізнавання образів передбачає дотримання низки важливих принципів [1]-[5]. Одним із ключових принципів при створенні програмних систем розпізнавання образів є використання модульного дизайну, що сприяє легкій інтеграції різних компонентів та алгоритмів. Останнім часом у сфері виявлення графічних об'єктів точність та ефективність алгоритмів росте за ступеневою залежністю. Модульність

дозволяє замінювати компоненти та впроваджувати нові алгоритми, поступово підвищуючи продуктивність системи розпізнавання образів [1]-[5].

Правильний вибір методів навчання розпізнавальних моделей відіграє важливу роль і залежить від конкретної задачі та наявних даних. Серед найбільш поширених методів навчання варто виділити контрольоване, пів контрольоване та неконтрольоване навчання. Наприклад, у сфері комп'ютерного зору часто використовується контрольоване навчання, яке передбачає навчання системи розпізнавання образів на основі позначених даних. Це дозволяє системі розпізнавання образів вивчати шаблони та робити точні прогнози на основі доступних даних [1]-[5].

Розпізнавання графічних образів є широко розповсюдженим варіантом застосування систем розпізнавання образів. У цьому випадку система комп'ютерного зору навчається розпізнавати конкретні шаблони на зображеннях і у майбутньому ідентифікувати зображення, в яких містяться ці шаблони. Розпізнавання графічних образів може бути використано для таких завдань, як розпізнавання облич на фотографіях, розрізнення об'єктів, визначення орієнтації та положення об'єктів або ключових точок [1]-[5].

Розпізнавання відео представляє собою підкатегорію розпізнавання графічних зображень, яка дає можливість ідентифікувати об'єкти, події або дії на відеозаписах. У відео аналітиці цей підхід може бути використаний для таких завдань, як ідентифікація осіб, виявлення вторгнень, виявлення руху, спостереження за об'єктами в реальному часі та відстеження об'єктів [1]-[5].

Використання методів розпізнавання образів для прогнозування стану фондового ринку є традиційним, але складним методом оцінки майбутньої цінності акцій компаній та інших торгових активів. Протягом багатьох десятиліть досліджувалися лінійні та машинні методи навчання [1]-[5].

Оптичне розпізнавання символів представляє собою процес визначення оптичних шаблонів на цифрових зображеннях. Для досягнення розпізнавання символів застосовують сегментацію зображення, виділення особливостей та подальшу класифікацію [1]-[5].

Використання методів розпізнавання текстових образів на основі машинного навчання є важливим для створення, аналізу та перекладу тексту. Такі шаблони використовуються для розуміння людської мови і генерації текстових повідомлень. В процесі розпізнавання тексту слова використовуються для класифікації документів і автоматичного виявлення важливих фрагментів тексту. Таким чином, розпізнавання текстових шаблонів грає важливу роль у виявленні шахрайства у фінансовій та страховій сферах [1]-[5].

У випадку розпізнавання рукописного тексту, розпізнавання образів використовується для ідентифікації шаблонів шляхом порівняння рукописних слів і підписів. Для комп'ютерного розпізнавання рукописних слів, які вводяться спеціальними засобами, використовуються різноманітні методи. Однак виявлення та розпізнавання рукописного тексту є складним завданням через його неправильну та різку форму [1]-[5].

Алгоритми аналізу зображень спрямовані на виявлення та ідентифікацію певних образів на візуальних зображеннях, щоб розпізнати конкретні об'єкти (процес виявлення об'єктів). Однією з типових задач розпізнавання образів є класифікація, де нейронні мережі використовуються для визначення класів для зображень або їх фрагментів на основі їхнього вмісту. Це є основою для візуального пошуку, де користувачі можуть швидко знаходити та порівнювати зображення за тегами [1]-[5].

Програмні системи розпізнавання образів на основі даних у вигляді мовленнєвих сигналів дозволяють комп'ютерним машинам отримувати та інтерпретувати голосові команди та інші аспекти усного спілкування. Розпізнавання мовлення базується на методах машинного навчання для розпізнавання образів, що дозволяє системам розпізнавати усну мову та надавати переклад [1]-[5].

Системи розпізнавання образів також можуть використовувати машинне навчання для аналізу та ідентифікації людських емоцій на зображеннях або відео. Їх метою є виявлення настрою, думок та намірів

аудиторії або клієнта. Глибоке навчання може бути використано для розпізнавання конкретних виразів обличчя та моделей рухів. Отримана інформація може бути використана для поліпшення маркетингових кампаній та обслуговування клієнтів [1]-[5].

Переваги застосування методів розпізнавання образів різняться залежно від конкретних програмних застосунків. Як правило, виявлення закономірностей в даних допомагає проводити аналіз та прогнозувати майбутні тенденції або створювати системи раннього попередження на основі конкретних показників тенденцій [1]-[5].

Серед інших переваг можна виділити те, що методи розпізнавання образів дозволяють виявляти закономірності, як допомагають ідентифікувати об'єкти під різними кутами та з різних відстаней, зокрема розпізнавати обличчя, за допомогою глибокого навчання відео або аналіз руху. Крім того, інтелектуальні системи розпізнавання графічних образів можуть використовуватися для розпізнавання людей за ходою або рухом [1]-[5].

Алгоритми розпізнавання образів дозволяють виявляти випадки, які можуть залишитися непоміченими для людини. Моделі алгоритмів можуть виявляти тонкі зміни в даних або кореляції між факторами в великих обсягах даних. Це особливо важливо в медичній галузі, наприклад, для дослідження аномалій за допомогою зображень магнітно-резонансної томографії. В інформаційній та комп'ютерній безпеці, розпізнавання образів використовується, наприклад, у програмних системах виявлення вторгнень, а також для моніторингу комп'ютерних мереж або систем [1]-[5].

Програмні системи розпізнавання образів використовуються також для прогнозування даних і майбутніх тенденцій, зокрема, у сферах, де потрібно передбачати та аналізувати рухи на ринках, визначати ціни на акції та інші можливості інвестицій або встановлювати тенденції поведінки об'єкта на автомобільних дорогах [1]-[5].

Методи розпізнавання образів можуть бути застосовані до різних видів цифрових даних, включаючи зображення, текст та відео. Проведення аналізу

для виявлення закономірностей сприяє систематизації результатів та дозволяє здійснювати обґрунтовані висновки. Розпізнавання образів може бути використане для автоматизації та повного розв'язання складних аналітичних завдань [1]-[5].

Застосування методів розпізнавання образів є досить важливим для планування діяльності підприємств, оскільки воно дозволяє виявляти тенденції та отримувати інформацію про дані, що може сприяти процесу прийняття рішень та розробці стратегій. Наприклад, компанія у сфері роздрібної торгівлі може використовувати розпізнавання образів для аналізу даних щодо продажів та ідентифікації популярних та менш популярних продуктів. Це дозволить приймати обґрунтовані рішення стосовно товарів, які тримати в асортименті та які рекламувати. Крім того, розпізнавання образів може знайти застосування в маркетингу та обслуговуванні клієнтів, допомагаючи виявляти моделі поведінки та вподобань клієнтів, що може служити основою для розробки цільових кампаній та підвищення задоволеності клієнтів [1]-[5].

У галузі інформатики та машинного навчання розпізнавання образів являє собою технологію, яка порівнює інформацію, що зберігається у базі даних, з вхідними даними [1]-[5].

Результатом процесу розпізнавання образів може бути призначення класу, групи або передбачення змінних. Таким чином, питання про те, в чому різниця між розпізнаванням образів та класифікацією, виявляється несуттєвим, оскільки алгоритм класифікації є складовою частиною задач машинного навчання з контрольованим навчанням, де головною метою є призначення об'єктів до певних класів [1]-[5].

У системах автоматизованої діагностики в галузі охорони здоров'я технології розпізнавання образів дозволяють рятувати життя. Розпізнавання графічних образів є основою комп'ютерних систем діагностики, які сприяють лікарям у визначенні шляхів подальшого лікування [1]-[5]. Програмні засоби медичного діагностування на основі технологій розпізнавання образів

дозволяють виявляти вроджені вади розвитку на ранніх стадіях, розпізнавати та класифікувати лейкоцити та тканини як здорові або хворі та розв'язувати інші завдання медичного діагностування [1]-[5].

Розпізнавання обличчя та візуальний пошук є двома ключовими способами використання технології розпізнавання графічних образів. Методології розпізнавання образів користуються надзвичайною популярністю в галузі комп'ютерного зору. Вони застосовуються у програмній розробці додатків, які можуть вчитися на основі зображень з метою подальшого ефективного виявлення та розпізнавання об'єктів на цих зображеннях, подібно до способу сприйняття інформації людиною [1]-[5].

Визначено, що розпізнавання образів представляє собою процес, який забезпечує здатність комп'ютерів визначати структурні відповідності в наборах даних та використовувати цю інформацію для прийняття рішень або прогнозування шляхом використання різних алгоритмів обробки інформації. Відзначено, що розпізнавання образів є більш широким поняттям, ніж комп'ютерний зір, який фокусується лише на розпізнаванні зображень, зазначено, що поняття образу може охоплювати будь-яку важливу сутність, яку необхідно визначити і виконати її ідентифікацію. Проаналізовано підходи та методи розв'язання задачі розпізнавання образів. Проаналізовано етапи процесу розпізнавання образів. Відзначено, що у наш час широко використовуваним методом розпізнавання образів є застосування штучних нейронних мереж, зокрема, згортових нейронних мереж з використанням алгоритмів глибокого навчання. Проаналізовано галузі практичного застосування програмних систем розпізнавання образів. Визначено переваги використання методів розпізнавання образів.

1.2 Аналіз програмного забезпечення для розпізнавання образів

Проаналізуємо програмне забезпечення підтримки процесу розпізнавання образів [6]-[10].

Програмне забезпечення Imagga [6]-[8] є автоматизованим рішенням для прикріплення тегів до фотографій та управління категоріями, спрямованим на обробку великого обсягу зображень. Програмне забезпечення Imagga (рис. 1.1) представлено як прикладний програмний інтерфейс (application programming interface, API) для розпізнавання цифрових зображень і включає бібліотеку, що дозволяє класифікувати зображення та управляти їхніми метаданими. За допомогою інструментів пошуку і фільтрації можна легко знаходити та керувати мультимедійними файлами [6]-[8].



Рисунок 1.1 – Програмне забезпечення Imagga [6], [8]

Також програмне забезпечення Imagga надає можливість створення звітів та проведення аналізу. Це програмне забезпечення схоже на інші API для управління цифровими активами, такі як Box, Airtable або Canto Digital Asset Management. Проте, порівнявши Imagga з іншими API для управління цифровими активами, які надають можливість розпізнавання зображень в

режимі онлайн безкоштовно, можна відзначити його значно більшу доступність [6], [8].

Серед особливостей програмного забезпечення Imagga для розв'язання завдань розпізнавання графічних образів можна виділити такі [6]-[8]:

- засоби прикріплення тегів до зображень: програмне забезпечення Imagga дозволяє автоматично розпізнавати об'єкти та сцени на зображеннях і прикріплювати до них відповідні теги, що полегшує організацію та пошук зображень [6], [8];

- управління категоріями: програмне забезпечення дозволяє створювати та керувати категоріями для легкого групування зображень за темами або проектами [6], [8];

- робота з метаданими та атрибутами: програмне забезпечення Imagga дозволяє додавати метадані та атрибути до зображень, що полегшує опис та ідентифікацію активів [6], [8];

- засоби пошуку та фільтрації: програмне забезпечення Imagga надає інструменти для ефективного пошуку та фільтрації зображень на основі їхніх характеристик та тегів [6], [8];

- засоби створення звітів та аналізу: програмне забезпечення Imagga надає можливості користувачу генерувати звіти про використання активів та проводити аналіз для кращого розуміння їхнього використання та ефективності [6], [8];

- крос-платформеність: програмне забезпечення Imagga підтримується на різних платформах, що робить його доступним для розробників з різних операційних систем [6], [8];

- легкість використання: Imagga надає зручний інтерфейс та документацію, яка полегшує роботу з програмним забезпеченням [6], [8];

- засоби API: існує можливість використовувати програмне забезпечення Imagga як API для інтеграції з іншими програмними застосунками та сервісами [6], [8];

– доступність безкоштовного розпізнавання зображень: програмне забезпечення Imagga надає можливість безкоштовного розпізнавання зображень в режимі онлайн, що робить його доступним для широкого кола користувачів [6], [8].

Серед недоліків програмного забезпечення Imagga варто відзначити те, що деякі функції та обсяги використання Imagga можуть бути платними, що може бути обмеженням для користувачів з невеликим бюджетом. Крім того, для ефективної роботи цієї програми користувачам може знадобитися підготовка та категоризація великих обсягів даних, що може бути затратним процесом як з боку часу, так і ресурсів [6], [8].

Програмне забезпечення LogoGrab (рис. 1.2) [6], [9] надає можливість користувачам розпізнавати логотипи, символи та бренди. Це програмне забезпечення дозволяє організації відслідковувати та керувати вмістом графічного медіа в різних екосистемах, таких як соціальні мережі, телерадіомовлення та онлайн вебплатформи [6], [9].

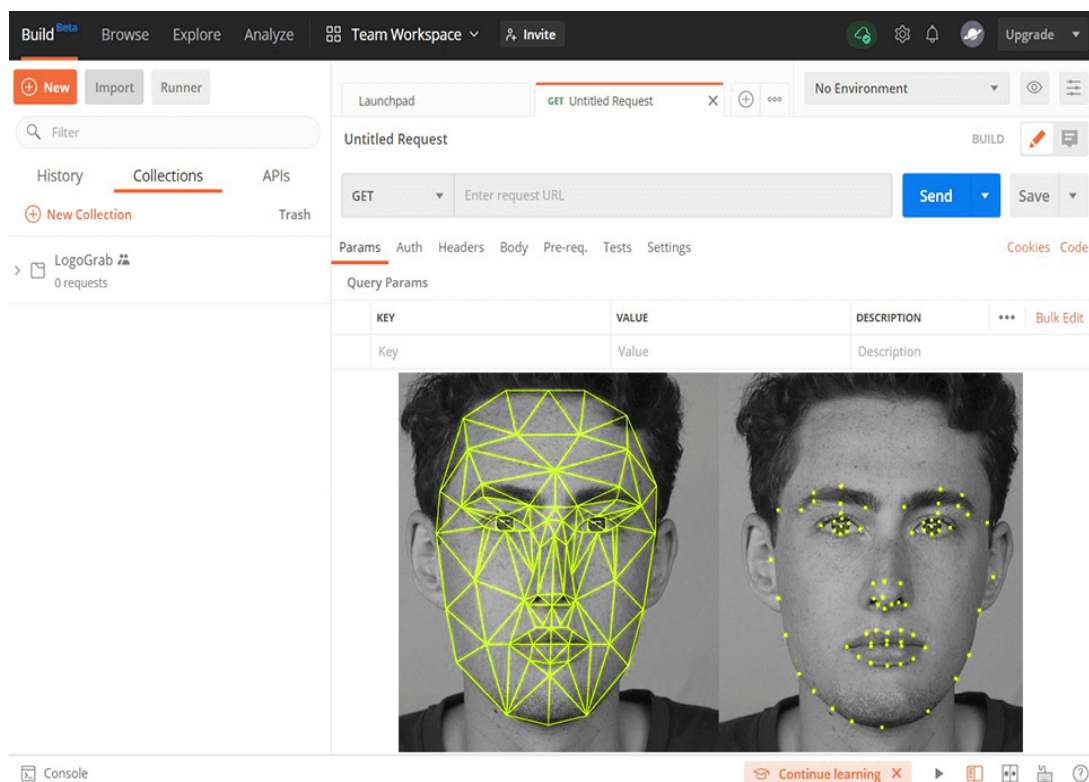


Рисунок 1.2 – Програмне забезпечення LogoGrab [6], [9]

Програмне забезпечення LogoGrab може бути легко інтегрованим у різноманітні програмні платформи. Перехід до використання LogoGrab та отримання від нього високоякісної інформації може бути надзвичайно легким процесом [6], [9].

Серед особливостей програмного забезпечення LogoGrab для розв'язання завдань розпізнавання графічних образів можна виділити такі [6], [9]:

- засоби розпізнавання логотипів і брендів: програмне забезпечення LogoGrab здатне ідентифікувати логотипи, символи та бренди на зображеннях і відео. Ця функція дозволяє точно визначати наявність бренду в різних медіа-ресурсах [6], [9];

- відстеження медіа-контенту: програмне забезпечення LogoGrab дозволяє відстежувати та аналізувати графічний медіа-контент, що містить логотип або символ. Це корисно для вимірювання рекламного впливу та розуміння споживчої поведінки [6], [9];

- широкий спектр застосувань: програмне забезпечення LogoGrab може використовуватися в різних сферах, включаючи маркетинг, моніторинг бренду, аналіз реклами та вимірювання ефективності кампаній [6], [9];

- інтеграція з різними платформами: програмне забезпечення LogoGrab легко інтегрується з різними соціальними медіа, вебсервісами та іншими платформами для моніторингу та аналізу медіа-контенту [6], [9];

- зручний інтерфейс: інтерфейс користувача LogoGrab є інтуїтивно зрозумілим і допомагає легко керувати медіа-вмістом та отримувати необхідну інформацію [6], [9];

- підтримка різних мов та регіонів: програмне забезпечення LogoGrab може розпізнавати логотипи на зображеннях з різних мов та географічних регіонів [6], [9];

- висока точність розпізнавання: програмне забезпечення LogoGrab відзначається високою точністю розпізнавання, що робить його надійним інструментом для потреб у відстеженні медіа-контенту [6], [9];

- аналітика та звіти: програмне забезпечення LogoGrab надає можливість створювати аналітичні звіти та аналізувати дані про використання бренду в різних контекстах [6], [9];

- безпека даних: програмне забезпечення LogoGrab забезпечує високий рівень безпеки для збереження і обробки ваших даних [6], [9];

- підтримка клієнтів: команда підтримки LogoGrab надає необхідну допомогу у використанні програми [6], [9].

Серед недоліків програмного забезпечення LogoGrab варто відзначити те, що деякі функції та обсяги використання LogoGrab можуть бути платними, що може бути обмеженням для користувачів з невеликим бюджетом. Крім того, для ефективної роботи цієї програми користувачам може знадобитися певний досвід та навички у налаштуванні та використанні програми [6], [9]. Іншим недоліком є те, що працюючи з медіа-контентом в Інтернеті, програмне забезпечення LogoGrab може бути залежним від доступу до різних джерел медіа, і якщо цей доступ обмежений або змінюється, це може вплинути на результати роботи програми [6], [9]. Крім того, порівняно з іншими програмними рішеннями для розпізнавання та моніторингу медіа-контенту, програмне забезпечення LogoGrab може мати обмежену функціональність або не підтримувати всі можливості, які можуть бути необхідні для конкретних завдань [6], [9].

Програмне забезпечення Clarifai (рис. 1.3) [6], [10] являє собою засіб API для розпізнавання зображень, що може визначати, класифікувати і тлумачити вміст зображень і відео за допомогою методів машинного навчання і штучного інтелекту [6], [10]. Застосування технологій комп'ютерного зору дозволяє програмному забезпеченню Clarifai для обробки візуальних даних допомагати різним організаціям розпізнавати вміст у своїх даних, що сприяє більш ефективному вирішенню їх завдань [6], [10].

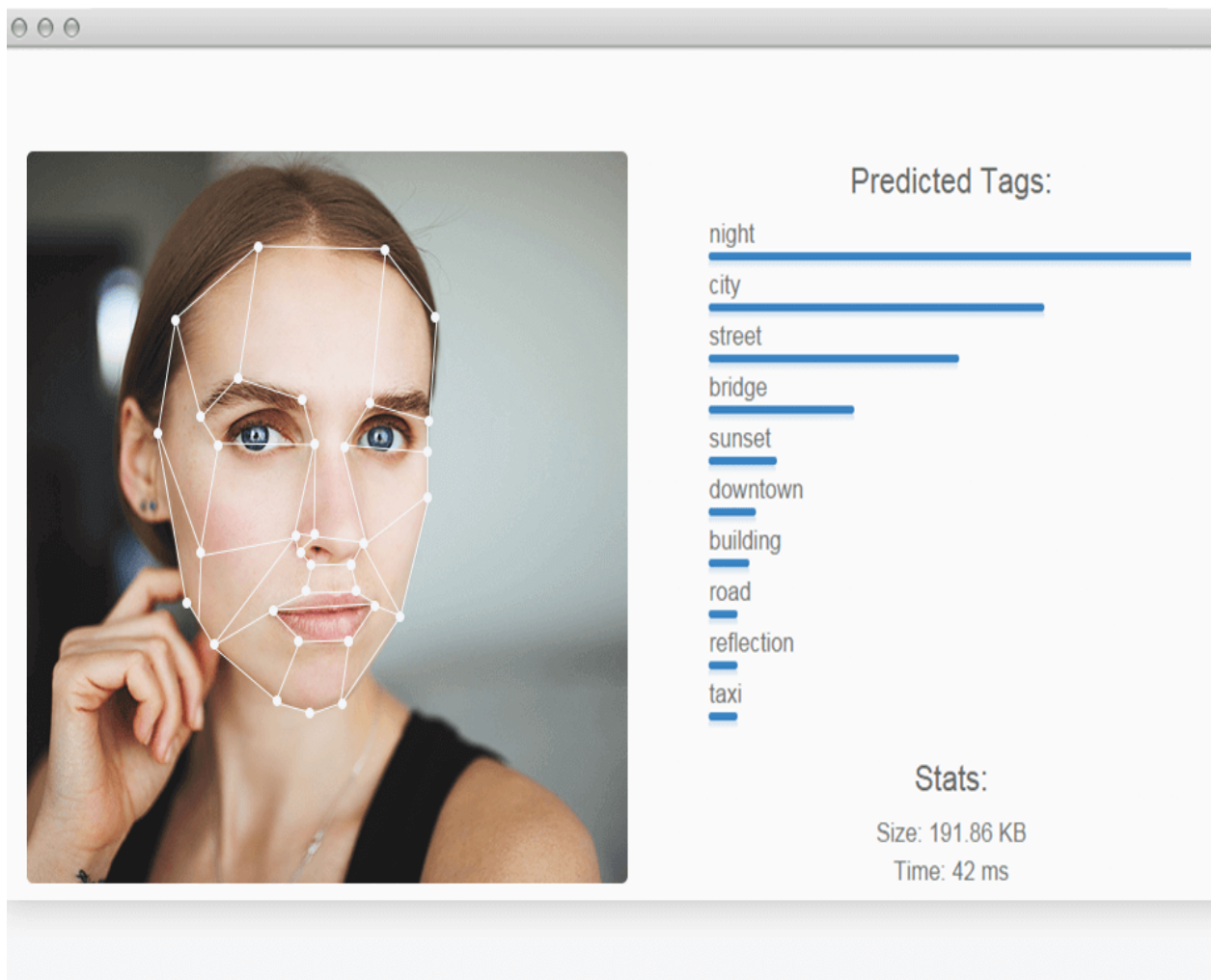


Рисунок 1.3 – Програмне забезпечення Clarifai [6], [10]

Серед особливостей програмного забезпечення Clarifai для розв'язання завдань розпізнавання графічних образів можна виділити такі [6], [10]:

- засоби розпізнавання зображень: програмне забезпечення Clarifai використовує машинне навчання для точної ідентифікації об'єктів, людей, тварин та інших елементів на зображеннях [6], [10];

- класифікація контенту: програмне забезпечення Clarifai дозволяє класифікувати вміст зображень, визначаючи, наприклад, який об'єкт зображено на фото або яка атмосфера на відео [6], [10];

- інтерпретація вмісту: програмне забезпечення Clarifai може тлумачити інформацію на зображеннях, надаючи контекст та зміст візуальних даних [6], [10];

– відмінна точність: завдяки використанню передових алгоритмів машинного навчання, програмне забезпечення Clarifai досягає високої точності в розпізнаванні та класифікації зображень [6], [10];

– API для інтеграції: програмне забезпечення Clarifai надає засоби інтеграції з іншими програмними продуктами у вигляді API, яке дозволяє інтегрувати його функціональність у різні програмні застосунки та програмні сервіси [6], [10];

– застосування в багатьох сферах: програмне забезпечення Clarifai знайде застосування в різних галузях, включаючи медицину, маркетинг, рекламу, електронну торгівлю, безпеку та інші [6], [10];

– підтримка організацій: програмне забезпечення Clarifai використовується різними організаціями для аналізу та розпізнавання великих обсягів візуальних даних [6], [10];

– застосування засобів та технологій штучного інтелекту: програмне забезпечення Clarifai використовує методи штучного інтелекту для здійснення аналізу та інтерпретації зображень [6], [10];

– система підтримки: програмне забезпечення Clarifai надає підтримку та навчання для користувачів, що допомагає їм максимально використовувати всі його можливості [6], [10];

– ефективне вирішення завдань: завдяки розпізнаванню та класифікації вмісту, програмне забезпечення Clarifai допомагає користувачам більш ефективно виконувати аналіз зображень та використовувати візуальні дані для своїх потреб [6], [10].

Серед недоліків програмного забезпечення Clarifai варто відзначити те, що деякі тарифні плани Clarifai можуть бути високими за вартістю для деяких користувачів. Крім того, програмне забезпечення Clarifai вимагає постійного доступу до Інтернету для роботи, що може бути не зручним у випадках з обмеженим Інтернет з'єднанням або в умовах, де конфіденційність даних вимагає їх обробки на локальному рівні [6], [10]. Іншим недоліком є те, що для непрофесійних користувачів або тих, хто не має досвіду з машинним

навчанням, програмне забезпечення Clarifai може бути складним у використанні і вимагати додаткового навчання та підтримки [6], [10].

Порівняльну характеристику програмного забезпечення для розпізнавання образів наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння програмного забезпечення для розпізнавання образів

Критерій порівняння	Imagga [8]	LogoGrab [9]	Clarifai [10]
Робота з метаданими та атрибутами	+	+	—
Управління категоріями зображень	+	—	—
Засоби прикріплення тегів до зображень	+	—	+
Безкоштовність	+—	+	—
Засоби API	+	+—	+

За результатами проведеного аналізу можна зробити висновок, що у наш час існує досить багато програмних засобів для розв’язання завдань розпізнавання образів. Проте деякі програмні засоби при розв’язанні завдань розпізнавання образів можуть робити суттєві помилки, що обумовлено використанням неефективних моделей та методів розпізнавання образів. Інші програми вимагають великої обчислювальної потужності, що може бути проблемою для вбудованих систем або мобільних пристроїв. Ефективність роботи деяких програмних систем розпізнавання образів може залежати від умов навколишнього середовища, таких як освітлення, погодні умови, кути огляду тощо, що може призвести до низької стійкості подібних систем в різних ситуаціях. Тому актуальною є розробка програмного забезпечення для розпізнавання графічних образів.

При розробці програмного забезпечення для розпізнавання графічних образів необхідно забезпечити такі функціональні вимоги:

- можливість розпізнавання транспортних засобів на вхідних зображеннях;
- можливість введення шляху до файлу з зображенням;
- можливість запиту програмою шляху до файлу з зображенням;
- реалізація оброблення помилок, що виникають при роботі з програмою;
- відображення часу виконання процесу розпізнавання;
- виведення результатів розпізнавання.

1.3 Висновки за розділом 1

Визначено, що розпізнавання образів представляє собою процес, який забезпечує здатність комп'ютерів визначати структурні відповідності в наборах даних та використовувати цю інформацію для прийняття рішень або прогнозування шляхом використання різних алгоритмів обробки інформації. Відзначено, що розпізнавання образів є більш широким поняттям, ніж комп'ютерний зір, який фокусується лише на розпізнаванні зображень, зазначено, що поняття образу може охоплювати будь-яку важливу сутність, яку необхідно визначити і виконати її ідентифікацію. Проаналізовано підходи та методи розв'язання задачі розпізнавання образів. Проаналізовано етапи процесу розпізнавання образів. Відзначено, що у наш час широко використовуваним методом розпізнавання образів є застосування штучних нейронних мереж, зокрема, згортових нейронних мереж з використанням алгоритмів глибокого навчання. Проаналізовано галузі практичного застосування програмних систем розпізнавання образів. Визначено переваги використання методів розпізнавання образів.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для розв'язання завдань розпізнавання

образів. Проте деякі програмні засоби при розв'язанні завдань розпізнавання образів можуть робити суттєві помилки, що обумовлено використанням неефективних моделей та методів розпізнавання образів. Інші програми вимагають великої обчислювальної потужності, що може бути проблемою для вбудованих систем або мобільних пристроїв. Ефективність роботи деяких програмних систем розпізнавання образів може залежати від умов навколишнього середовища, таких як освітлення, погодні умови, кути огляду тощо, що може призвести до низької стійкості подібних систем в різних ситуаціях. Тому актуальною є розробка програмного забезпечення для розпізнавання графічних образів.

Сформульовано функціональні вимоги до програмного забезпечення для розпізнавання графічних образів.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Вибір мови програмування

При розробці програмного забезпечення для розпізнавання графічних образів було використано мову програмування Python [11], [12].

Python – це високорівнева мова програмування, спрямована на полегшення написання ефективного і читабельного коду. У стандартній бібліотеці мови міститься велика кількість корисних функцій, і на ринку існує велика кількість розширень та бібліотек, які додають додатковий функціонал та алгоритми. Python підтримує різні парадигми програмування, включаючи об'єктно-орієнтоване, структурне, функціональне і імперативне програмування. Основними характеристиками цієї мови є динамічна типізація, автоматичне управління пам'яттю, можливість інтроспекції, обробка виняткових ситуацій та підтримка багатопотокових обчислень. Функції та класи можна організовувати у модулі, який в свою чергу може бути включений у пакет [11], [12].

Мова Python відкриває перед розробниками різноманітні шляхи підходу до програмування. Це може бути об'єктно-орієнтований стиль, процедурне програмування або загальне програмування. Завдяки різноманіттю модулів конфігурації, як-от обробка винятків, абстракція та робота з об'єктами даних, зокрема, з класами, розробники можуть використовувати різні підходи та методології [11], [12].

Серед переваг Python слід відзначити високу читабельність коду, швидкість розробки, можливість запускати програми на різних платформах та легку інтеграцію з іншими мовами завдяки різним інтерфейсам. Крім того, для Python існує велика кількість додаткових бібліотек і фреймворків, що розширюють можливості мови [11], [12].

На мові програмування Python створено велику кількість сторонніх бібліотек і фреймворків, які полегшують розробку специфічних програмних застосунків [11], [12].

Мова програмування Python дозволяє створювати косплатформні програмні застосунки, які можуть працювати під керівництвом різних операційних систем [11], [12].

Недоліком Python є те, що в порівнянні з деякими іншими мовами, він може бути менш продуктивним у виконанні ресурсоемних обчислень і операцій через свою вищу рівень абстракції [11], [12].

Обґрунтування вибору мови програмування для розробки програмного забезпечення для розпізнавання графічних образів наведено у таблиці 2.1.

Таблиця 2.1 – Обґрунтування вибору мови програмування для розробки програмного забезпечення для розпізнавання графічних образів

Критерій порівняння мов програмування	Мова програмування		
	Python	C++	Ruby
Наявність програмних модулів для розпізнавання образів	+	+	+—
Велика спільнота розробників	+	+—	—
Зручність для створення ПЗ для розпізнавання графічних образів	+	—	—
Читабельність коду	+	+	+—
Висока швидкодія	+	+	+—

Отже, для реалізації програмного забезпечення для розпізнавання графічних образів обрано мову програмування Python, яка є високорівневою, інтерпретованою мовою програмування, що є досить простою для використання при розробці різноманітних програм, включаючи вебзастосунки,

високопродуктивні обчислення, та програмні засоби штучного інтелекту й аналізу даних.

2.2 Вибір середовища та інших засобів розробки для створення програмного забезпечення для розпізнавання графічних образів

В якості середовища розробки для створення програмного забезпечення для розпізнавання графічних образів було обрано PyCharm [13], [14].

PyCharm IDE – це один з найуспішніших редакторів, який користуються популярністю серед професійних розробників і програмістів, що працюють з мовою Python. Великий набір функцій PyCharm робить його потужним інструментом, який не перевантажує користувача, навпаки, багато з цих функцій спрощують роботу, особливо для новачків в галузі програмування на Python [13], [14].

Для розробки програмного забезпечення для розпізнавання графічних образів також було використано бібліотеку з відкритим вихідним кодом OpenCV [15], яка впроваджує алгоритми комп'ютерного зору, обробки зображень і багато іншого. Вона реалізована на мовах програмування C/C++ і може бути використана в різних середовищах, включаючи C/C++, Python, Java, Ruby, Matlab і C#. OpenCV підтримує різні операційні системи, такі як Microsoft Windows, Linux, Mac OS X, Android і iOS [15]. Ця бібліотека продовжує розвиватися, постійно доповнюючись новими алгоритмами і функціями. Наразі кількість реалізованих алгоритмів перевищує 2500, що робить OpenCV дуже потужним інструментом для швидкого розроблення програмного забезпечення для обробки даних і розпізнавання [15].

Для розробки програмного забезпечення для розпізнавання графічних образів також було використано бібліотеку Keras [16], яка призначена для роботи з нейронними мережами і написана мовою програмування Python. Вона використовується як розширення до інших фреймворків, таких як

Deeplearning4j, TensorFlow і Theano. Головною метою Keras є полегшення розробки нейронних мереж і вона містить велику кількість реалізованих компонентів, таких як шари, функції активації, оптимізатори і т. д [16].

Бібліотека Keras відзначається швидким темпом розвитку і вважається другим за швидкістю росту фреймворком для глибокого навчання після TensorFlow. Вона також третя за розміром, після TensorFlow і Caffe [16].

Головною перевагою бібліотеки Keras є її простота, яка дозволяє легко створювати моделі і високо швидко розробляти прототипи. Ця бібліотека ідеально підходить для швидкого створення і тестування нейронних мереж для вирішення конкретних завдань [16].

Обґрунтування вибору середовища розробки для створення програмного забезпечення для розпізнавання графічних образів наведено у таблиці 2.2.

Таблиця 2.2 – Обґрунтування вибору середовища розробки для створення програмного забезпечення для розпізнавання графічних образів

Критерій порівняння середовищ розробки	Середовище розробки		
	PyCharm	Microsoft Visual Studio	Qt Creator
Підтримка різних операційних систем	+	—	+
Велика кількість розширень	+	+	+—
Зручність розробки на мові Python	+	—	—
Зручність для створення ПЗ для розпізнавання графічних образів	+	—	—
Зручність редактору коду	+	—	—

Таким чином, для створення програмного забезпечення для розпізнавання графічних образів обрано середовище розробки PyCharm, що являє собою універсальний інструмент для створення програм, написаних на мові програмування Python, дозволяє створювати програми для різних операційних систем та є досить зручним для розробки програм різного призначення.

2.3 Розробка моделі взаємодії користувача з програмним забезпеченням для розпізнавання графічних образів

З метою можливості розробки програмного забезпечення для розпізнавання графічних образів створимо відповідну модель у вигляді діаграми прецедентів.

При розробленні моделі взаємодії користувача з програмним забезпеченням для розпізнавання графічних образів враховуємо те, що користувачі програми мають можливості розпізнавання транспортних засобів на вхідних зображеннях, введення шляху до файлу з зображенням, запит програмою шляху до файлу з зображенням, оброблення помилок, що виникають при роботі з програмою, відображення часу виконання процесу розпізнавання, виведення результатів розпізнавання.

Основний успішний сценарій користувача при роботі з програмним забезпеченням для розпізнавання графічних образів можна описати таким чином:

- користувач програмного забезпечення для розпізнавання графічних образів запускає програму стандартними засобами командного рядка операційної системи;
- після запуску програми для розпізнавання графічних образів користувачу відображається повідомлення з проханням ввести шлях до файлу з графічним зображенням;

- користувач програми для розпізнавання графічних образів бачить повідомлення та вводить шлях до файлу з графічним зображенням;
- користувач здійснює розв’язання задачі розпізнавання образів;
- користувач бачить результат розпізнавання образів, зокрема інформацію про час розпізнавання;
- користувач завершує роботу з програмним забезпеченням для розпізнавання графічних образів.

Враховуючи сценарії роботи користувачів при роботі з програмним забезпеченням для розпізнавання графічних образів, модель взаємодії користувача з програмним забезпеченням для розпізнавання графічних образів подамо у вигляді діаграми прецедентів, зображеної на рис. 2.1.

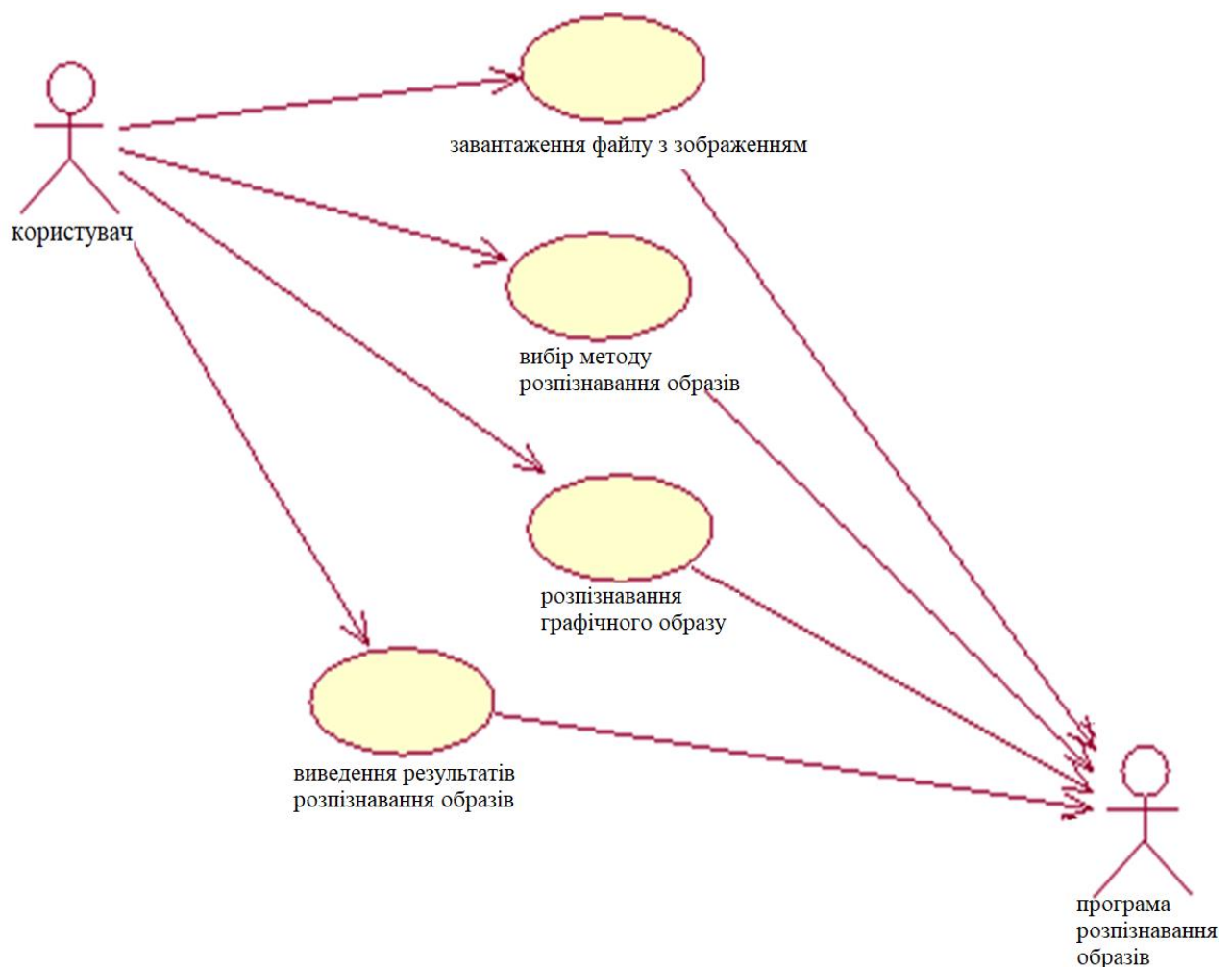


Рисунок 2.1 – Модель взаємодії користувача з програмним забезпеченням для розпізнавання графічних образів

Таким чином, створено модель взаємодії користувача з програмним забезпеченням для розпізнавання графічних образів, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для розпізнавання графічних образів.

2.4 Висновки за розділом 2

Для реалізації програмного забезпечення для розпізнавання графічних образів обрано мову програмування Python, яка є високорівневою, інтерпретованою мовою програмування, що є досить простою для використання при розробці різноманітних програм, включаючи вебзастосунки, високопродуктивні обчислення, та програмні засоби штучного інтелекту й аналізу даних.

Для створення програмного забезпечення для розпізнавання графічних образів обрано середовище розробки PyCharm, що являє собою універсальний інструмент для створення програм, написаних на мові програмування Python, дозволяє створювати програми для різних операційних систем та є досить зручним для розробки програм різного призначення.

Створено модель взаємодії користувача з програмним забезпеченням для розпізнавання графічних образів, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для розпізнавання графічних образів.

3 ОПИС ПРОГРАМИ

3.1 Структура програмного забезпечення для розпізнавання графічних образів

Структуру програмного забезпечення для розпізнавання графічних образів подамо у вигляді діаграми пакетів (рис. 3.1).

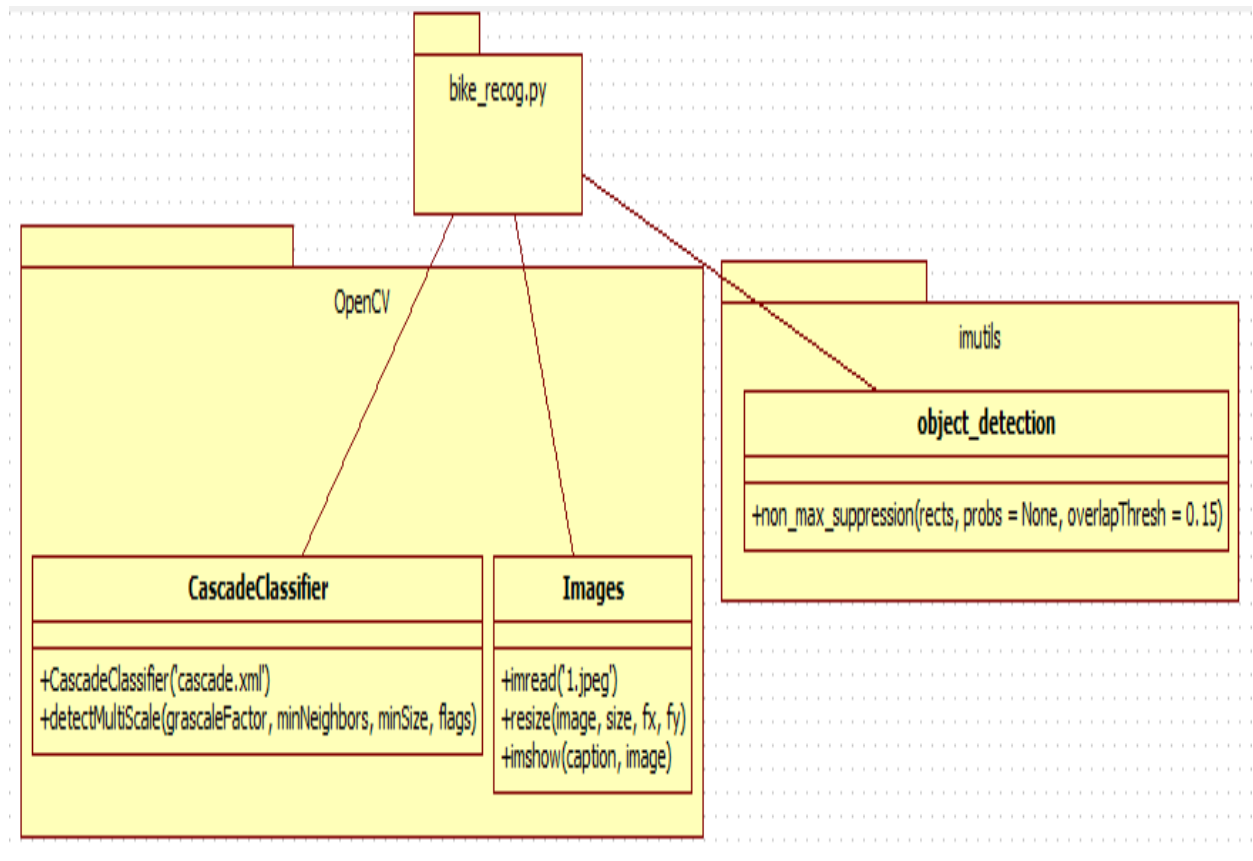


Рисунок 3.1 – Структура програмного забезпечення для розпізнавання графічних образів

Як видно, основними пакетами програмного забезпечення для розпізнавання графічних образів є такі:

- пакет `object_detection`, який визначає (ідентифікує) певну область зображення, на якій розташовано деякий об'єкт;

- пакет CascadeClassifier, який за допомогою каскадного класифікатора визначає належність зображення певному класу (наприклад, належність зображення до класу велосипедів чи автомобілів);
- пакет images, який містить засоби для роботи з зображеннями.

На рис. 3.2 наведено структуру програмного компоненту, що відповідає за синтез нейромережевої розпізнавальної моделі на основі використання генетичних алгоритмів штучного інтелекту.

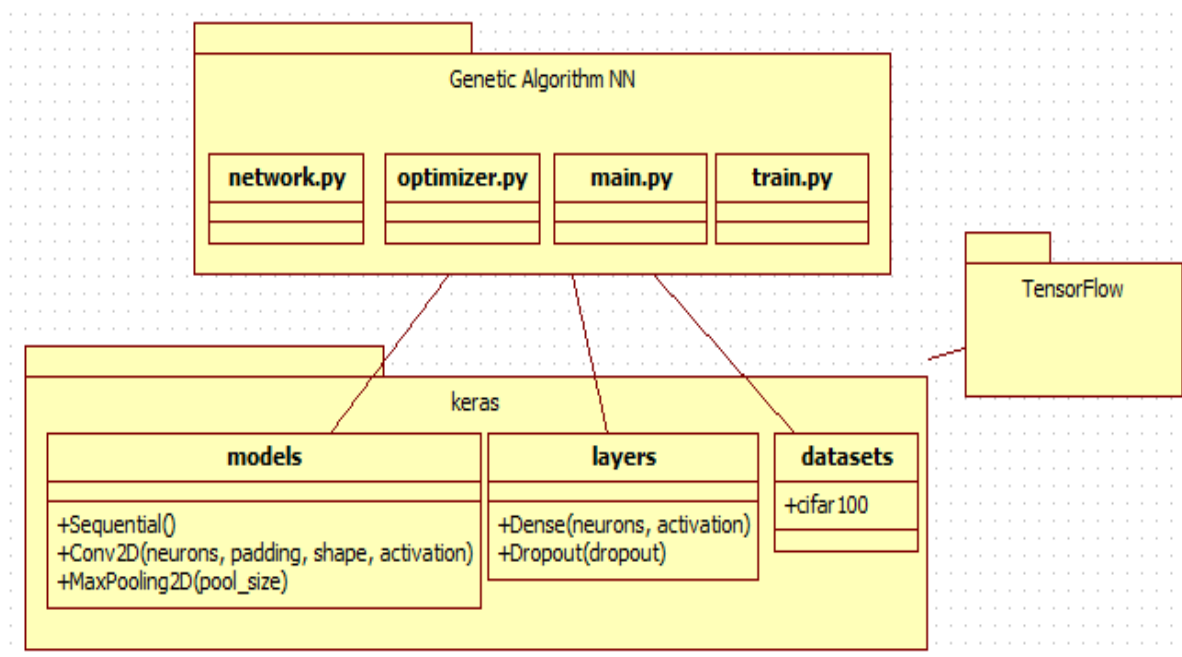


Рисунок 3.2 – Структура програмного компоненту, що відповідає за синтез нейромережевої розпізнавальної моделі на основі використання генетичних алгоритмів

Як видно з рис. 3.2, для синтезу нейромережевої розпізнавальної моделі на основі використання генетичних алгоритмів використовуються засоби бібліотеки keras, зокрема структури models, що дозволяють створювати нейромережеві моделі заданої архітектури, структури шари layers, що дозволяють визначити особливості кожного шару нейромоделі, а також засоби для роботи з наборами даних datasets.

3.2 Функціонування програмного забезпечення для розпізнавання графічних образів

Схема функціонування програмного забезпечення для розпізнавання графічних образів наведена на рис. 3.3.

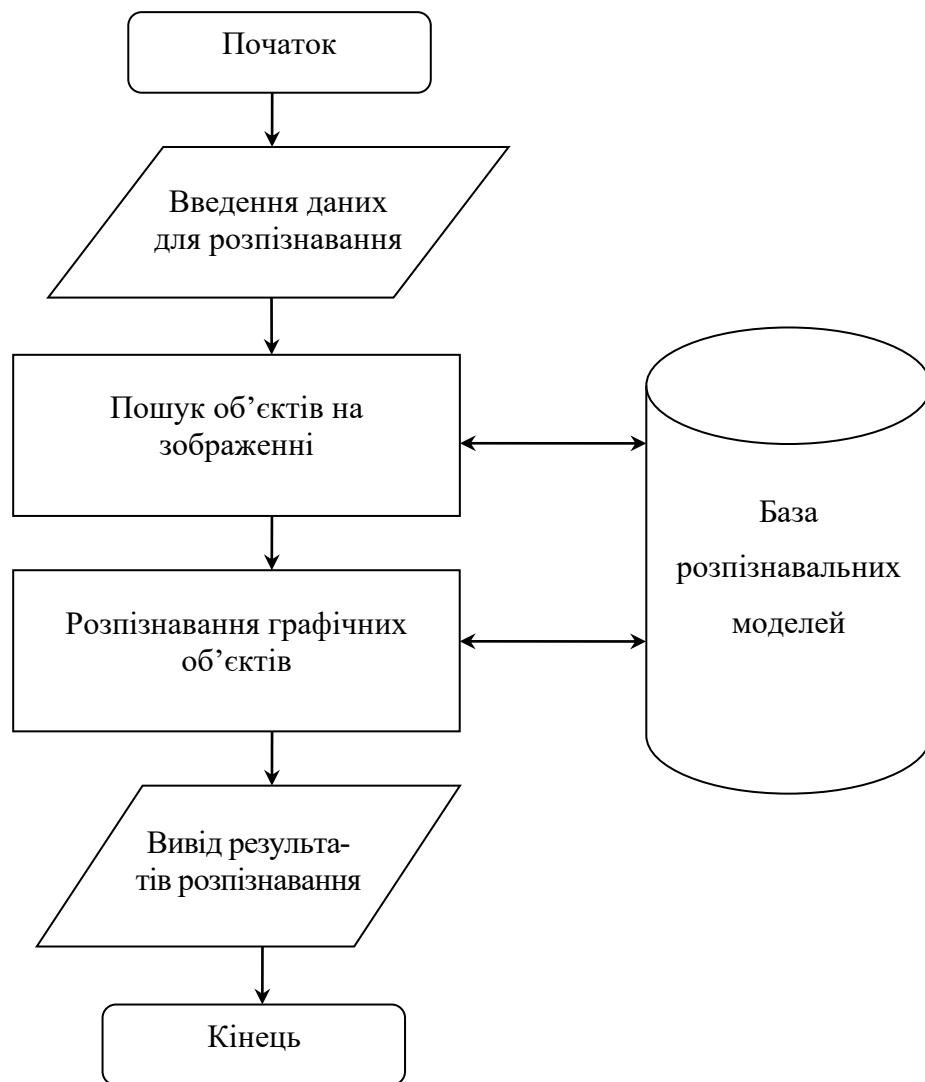


Рисунок 3.3 – Схема функціонування програмного забезпечення для розпізнавання графічних образів

На початку взаємодії користувача з програмою для розпізнавання графічних образів необхідно ввести інформацію щодо розташування файлу з зображенням. Після цього програмне забезпечення для розпізнавання

графічних образів, використовуючи попередньо синтезовані розпізнавальні моделі, виконує ідентифікацію областей, на яких зображено об'єкти. Після цього знайдені об'єкти програма ідентифікує, тобто відносить їх до певного класу (наприклад, велосипеди, автомобілі, вантажні автомобілі та інше), для цього також використовуються попередньо навчені розпізнавальні моделі на основі нейромережових технологій. Після завершення процесу розпізнавання графічних образів виконується виведення результатів розпізнавання на екран. На останньому етапі відбувається завершення роботи програмного забезпечення для розпізнавання графічних образів.

3.3 Особливості синтезу розпізнавальних моделей для реалізації програмного забезпечення для розпізнавання графічних образів

Як видно з рис. 3.3 важливу роль у процесі розпізнавання графічних образів відіграє база розпізнавальних моделей.

Відповідні моделі синтезуються на основі генетичних методів штучного інтелекту. Як базис для синтезу моделей обрано згорткові нейронні мережі, оскільки вони дозволяють ефективно розв'язувати завдання розпізнавання графічних образів.

Тому для функціонування програмного забезпечення для розпізнавання графічних образів необхідно виконати синтез розпізнавальних моделей.

На етапі підготовки до навчання розпізнавальної моделі створюються вибірки зображень, де позитивні об'єкти включають в себе зображення велосипедів (або інших транспортних засобів), а негативні – зображення без відповідних транспортних засобів. Для навчання розпізнавальної моделі було використано 4200 позитивних та 7400 негативних зображень. На рис. 3.4 наведено приклади позитивних зображень для створення моделі для розпізнавання велосипедів, на рис.3.5 наведено приклади негативних зображень для створення моделі для розпізнавання велосипедів.



Рисунок 3.4 – Приклади позитивних зображень для створення моделі для розпізнавання велосипедів



Рисунок 3.5 – Приклади негативних зображень для створення моделі для розпізнавання

Також було створено описові файли для обох видів зображень. Приклад описового файлу для позитивних зображень наведено на рис. 3.6.

```

bike\bike_1.jpg 1 0 0 728 429
bike\bike_2.jpg 1 0 0 220 186
bike\bike_3.jpg 1 0 0 210 126
bike\bike_4.jpg 1 0 0 590 383
bike\bike_5.jpg 1 0 0 1200 800
bike\bike_6.bmp 1 0 0 274 431
bike\bike_7.jpg 1 0 0 155 92
bike\bike_8.jpg 1 0 0 424 253

```

Рисунок 3.6 – Приклад описового файлу для позитивних зображень

Як видно, на рис. 3.6, структура описового файлу з позитивними зображеннями, містить інформацію про шлях до файлу, назву файлу, кількість позитивних об'єктів у файлі, а також координати початку та кінця регіону, де знаходиться позитивний об'єкт.

Описові файли для негативних зображень містять лише інформацію про шлях до файлу та назву файлу.

Після того, як було сформовано навчальні вибірки було виконано синтез каскадного класифікатора за допомогою програмного модулю `opencv_traincascade` бібліотеки OpenCV.

Як підсумок генерується файл `cascade.xml`, що являє собою модель на основі каскадного класифікатора, який може бути включений у програмне забезпечення для подальшого використання при розпізнаванні велосипедів на графічних зображеннях. Недоліком такого підходу є суттєва чутливість каскадного класифікатора до вхідних параметрів, внаслідок чого якість розпізнавання безпосередньо залежить від таких параметрів:

- кількість рівнів у каскадному класифікаторі;
- відсоток правильно розпізнаних позитивних зображень. Збільшення цього параметра призводить до збільшення кількості неправильних визначень. Цей параметр залежить від конкретної вихідної вибірки;
- максимальний допустимий рівень помилкових спрацювань на кожному етапі класифікатора;
- відсоток позитивних зображень від загальної кількості, які беруть участь у навчанні;

– кількість та різноманітність зображень (присутність об'єкта в різних ракурсах на зображеннях) у навчальній вибірці.

Крім моделей на основі каскадного класифікатора було здійснено синтез розпізнавальних моделей на основі згорткових нейронних мереж. Для цього було використано бібліотеки Keras та TensorFlow. Як вхідні дані використовувався набір CIFAR 100, що складається з 45 тисяч зображень, 100 класів об'єктів та 20 категорій об'єктів. Після завершення синтезу нейромережевої моделі, було здійснено її перевірку на п'яти тисячах зображеннях. Після завершення процесу синтезу, розпізнавальні моделі на основі нейронних мереж були збережені у форматах *.json та *.h5 для подальшого використання у програмному забезпеченні для розпізнавання образів.

На рис. 3.7 наведено графік процесу синтезу нейромережевої розпізнавальної моделі.

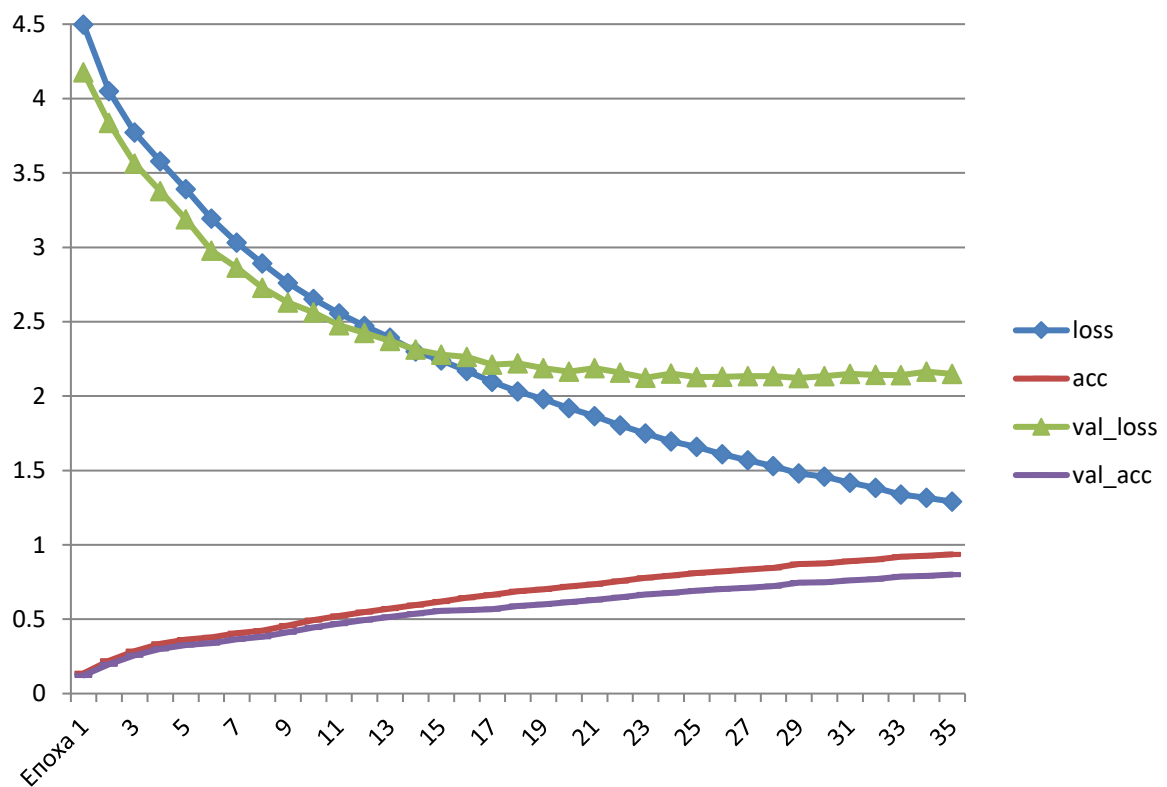


Рисунок 3.7 – Графік процесу синтезу нейромережевої розпізнавальної моделі

На рис. 3.7 використано такі позначення: `loss` – втрати навчання на навчальних даних, `acc` – точність на навчальних даних, `val_loss` – втрати на перевірочних даних, `val_acc` – точність на перевірочних даних.

3.4 Висновки за розділом 3

Розроблено програмне забезпечення для розпізнавання графічних образів. Запропоновано структуру програмного забезпечення для розпізнавання графічних образів. Визначено, що основними пакетами програмного забезпечення для розпізнавання графічних образів є такі: пакет `object_detection`, який визначає (ідентифікує) певну область зображення, на якій розташовано деякий об'єкт; пакет `CascadeClassifier`, який за допомогою каскадного класифікатора визначає належність зображення певному класу (наприклад, належність зображення до класу велосипедів чи автомобілів); пакет `images`, який містить засоби для роботи з зображеннями.

Описано функціонування програмного забезпечення для розпізнавання графічних образів. Описано особливості синтезу розпізнавальних моделей для реалізації програмного забезпечення для розпізнавання графічних образів.

4 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

4.1 Призначення й умови застосування програми

Програма призначена для розпізнавання графічних образів. Програма написана на мові Python, яка є високорівневою, інтерпретованою мовою програмування, що є досить простою для використання при розробці різноманітних програм, включаючи вебзастосунки, високопродуктивні обчислення, та програмні засоби штучного інтелекту й аналізу даних. В якості середовища розробки для створення програми обрано PyCharm, що являє собою універсальний інструмент для створення програм, написаних на мові програмування Python, дозволяє створювати програми для різних операційних систем та є досить зручним для розробки програм різного призначення.

Програмне забезпечення для розпізнавання графічних образів забезпечує виконання таких функцій:

- можливість розпізнавання транспортних засобів на вхідних зображеннях;
- можливість введення шляху до файлу з зображенням;
- можливість запиту програмою шляху до файлу з зображенням;
- реалізація оброблення помилок, що виникають при роботі з програмою;
- відображення часу виконання процесу розпізнавання;
- виведення результатів розпізнавання.

Для експлуатації програмного забезпечення для розпізнавання графічних образів необхідно таке апаратне та програмне забезпечення:

- оперативна пам'ять об'ємом 2 Гб і більше;
- процесор із тактовою частотою 1,2 GHz і більше;
- вільне місце на диску 50 Мб;
- встановлені програмні засоби Python 3, бібліотека OpenCV, бібліотеки imutils та numpy;

– встановлена операційна система.

Функціональні характеристики розробленого програмного забезпечення для розпізнавання графічних образів наведено у технічному завданні (додаток А). Фрагмент тексту програмного забезпечення для розпізнавання графічних образів наведено у додатку Б.

4.2 Характеристики програми для розпізнавання графічних образів

Програмне забезпечення для розпізнавання графічних образів може працювати безперервно, в залежності від потреб користувача. Програмне забезпечення для розпізнавання графічних образів може використовуватись в будь-якій операційній системі. Для коректного функціонування програми для розпізнавання графічних образів вміст директорії з програмним забезпеченням не можна змінювати.

Для успішного виконання програмного забезпечення для розпізнавання графічних образів необхідно, щоб основний файл програми та файли з розпізнавальними моделями розташовувалися в одній теці, при цьому файли зображень можуть бути розташовані у різних теках.

4.3 Інструкція по експлуатації програми

4.3.1 Звернення до програми

Для запуску програмного забезпечення для розпізнавання графічних образів необхідно запустити головний файл, що знаходиться в кореневій директорії програми.

Запуск програми користувач може здійснити стандартними засобами командного рядку операційної системи. При запуску програми користувач може вказати файли зображень як аргументи командного рядка або надати їх після відповідного повідомлення програми (наприклад, D:/005.jpg).

4.3.2 Вхідні й вихідні дані

Вхідними даними до програмного забезпечення для розпізнавання графічних образів є файл з зображенням, яке необхідно розпізнати.

Вихідними даними програмного забезпечення для розпізнавання графічних образів є зображення, на якому виділено розпізнаний транспортний засіб, або повідомлення про неуспішність процесу розпізнавання транспортного засобу.

4.3.3 Повідомлення

Користувач програмного забезпечення для розпізнавання графічних образів може отримати такі повідомлення: повідомлення про необхідність завантаження файлу з графічним зображенням, повідомлення про успішне розпізнавання графічного об'єкту.

4.4 Виконання програмного забезпечення для розпізнавання графічних образів

Після активації програмне забезпечення для розпізнавання графічних образів переглядає вказаний файл, проводить обробку та повертає зображення, на якому виділено розпізнаний транспортний засіб у зеленому обрамленні. У випадку невдалого розпізнавання, повертається початкове зображення.

Користувач програмного забезпечення для розпізнавання графічних образів здійснює його запуск стандартними засобами командного рядку операційної системи. У користувача є можливість передати шлях до файлу, який необхідно розпізнавати, як параметр при запуску основного файлу через командний рядок. Якщо при запуску програми не було передано ніякі параметри, вона виводить запит на введення шляху до файлу, який необхідно розпізнати. Приклади запуску програми наведено на рис. 4.1.

```

C:\>python bike_recog.py c:/001.jpeg
[INFO] detection took: 1.404s
C:\>python bike_recog.py
Enter image file path: c:/001.jpeg
[INFO] detection took: 0.889s

```

Рисунок 4.1 – Приклади запуску програми

Після завершення розпізнавання користувачу виводиться інформація про час розпізнавання.

Крім того, після розпізнавання користувачу виводиться зображення з розпізнаним транспортним засобом.

На рис. 4.2 наведено приклади успішного розпізнавання велосипедів.



Рисунок 4.2 – Приклади успішного розпізнавання велосипедів за допомогою програмного забезпечення для розпізнавання образів

На рис. 4.3 наведено приклад невдалого розпізнавання велосипедів.



Рисунок 4.3 – Приклади невдалого розпізнавання велосипедів за допомогою програмного забезпечення для розпізнавання образів

4.5 Тестування та дослідження програмного забезпечення для розпізнавання графічних образів

Виконано тестування програмного забезпечення для розпізнавання графічних образів.

Виявлено, що програма для розпізнавання графічних образів функціонує правильно та злагоджено. Вона реалізує всі функціональні вимоги та успішно виконує свою основну задачу – розпізнавання графічних образів.

Як відзначено у попередніх розділах, важливу роль у програмному забезпеченні розпізнавання образів відіграють розпізнавальні моделі. Тому доцільно дослідити ефективність їх використання для розпізнавання певних об'єктів.

У табл. 4.1 наведено результати синтезу розпізнавальних моделей на основі повнозв'язних нейронних мереж (НМ) та згорткових нейронних мереж.

Таблиця 4.1 – Результати синтезу розпізнавальних моделей на основі повнозв'язних нейронних мереж та згорткових нейронних мереж

Критерій порівняння	Повнозв'язна НМ	Згорткова НМ 4 шари	Згорткова НМ 6 шарів	Згорткова НМ 6 шарів (ГА)
Час навчання	2 г. 17 хв. 21 с.	3 г. 49 хв. 51 с.	12 г. 48 хв. 12 с.	10 г. 12 хв. 42 с.
Точність розпізнавання	62,41%	78,93%	95,39%	94,2%

Як видно з табл. 4.1, результати експериментального дослідження показали, що для розпізнавання у програмному забезпеченні доцільно використовувати шестишарову згорткову нейронну мережу, оскільки така розпізнавальна модель забезпечує прийнятну точність.

Таким чином, розроблене програмне забезпечення дозволяє виконувати розпізнавання графічних образів.

4.6 Висновки за розділом 4

Описано програмне забезпечення для розпізнавання графічних образів. Виконано тестування розробленого програмного забезпечення для розпізнавання графічних образів. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє розв'язувати задачі розпізнавання графічних образів.

ВИСНОВКИ

В ході виконання дипломної кваліфікаційної роботи магістра було проаналізовано та досліджено процес розробки програмного забезпечення для розпізнавання образів.

Визначено, що розпізнавання образів представляє собою процес, який забезпечує здатність комп'ютерів визначати структурні відповідності в наборах даних та використовувати цю інформацію для прийняття рішень або прогнозування шляхом використання різних алгоритмів обробки інформації. Відзначено, що розпізнавання образів є більш широким поняттям, ніж комп'ютерний зір, який фокусується лише на розпізнаванні зображень, зазначено, що поняття образу може охоплювати будь-яку важливу сутність, яку необхідно визначити і виконати її ідентифікацію. Проаналізовано підходи та методи розв'язання задачі розпізнавання образів. Проаналізовано етапи процесу розпізнавання образів. Відзначено, що у наш час широко використовуваним методом розпізнавання образів є застосування штучних нейронних мереж, зокрема, згортових нейронних мереж з використанням алгоритмів глибокого навчання. Проаналізовано галузі практичного застосування програмних систем розпізнавання образів. Визначено переваги використання методів розпізнавання образів.

За результатами проведеного аналізу зроблено висновок, що у наш час існує досить багато програмних засобів для розв'язання завдань розпізнавання образів. Проте деякі програмні засоби при розв'язанні завдань розпізнавання образів можуть робити суттєві помилки, що обумовлено використанням неефективних моделей та методів розпізнавання образів. Інші програми вимагають великої обчислювальної потужності, що може бути проблемою для вбудованих систем або мобільних пристроїв. Ефективність роботи деяких програмних систем розпізнавання образів може залежати від умов навколишнього середовища, таких як освітлення, погодні умови, кути огляду тощо, що може призвести до низької стійкості подібних систем в різних

ситуаціях. Тому актуальною є розробка програмного забезпечення для розпізнавання графічних образів.

Сформульовано функціональні вимоги до програмного забезпечення для розпізнавання графічних образів.

Для реалізації програмного забезпечення для розпізнавання графічних образів обрано мову програмування Python, яка є високорівневою, інтерпретованою мовою програмування, що є досить простою для використання при розробці різноманітних програм, включаючи вебзастосунки, високопродуктивні обчислення, та програмні засоби штучного інтелекту й аналізу даних.

Для створення програмного забезпечення для розпізнавання графічних образів обрано середовище розробки PyCharm, що являє собою універсальний інструмент для створення програм, написаних на мові програмування Python, дозволяє створювати програми для різних операційних систем та є досить зручним для розробки програм різного призначення.

Новизна роботи полягає в тому, що запропоновано модель взаємодії користувача з програмним забезпеченням для розпізнавання графічних образів, яка подана у вигляді діаграми прецедентів, та дозволяє розроблювати програмне забезпечення для розпізнавання графічних образів.

Практичне значення роботи полягає в тому, що розроблено програмне забезпечення для розпізнавання графічних образів.

Запропоновано структуру програмного забезпечення для розпізнавання графічних образів. Визначено, що основними пакетами програмного забезпечення для розпізнавання графічних образів є такі: пакет `object_detection`, який визначає (ідентифікує) певну область зображення, на якій розташовано деякий об'єкт; пакет `CascadeClassifier`, який за допомогою каскадного класифікатора визначає належність зображення певному класу (наприклад, належність зображення до класу велосипедів чи автомобілів); пакет `images`, який містить засоби для роботи з зображеннями.

Описано функціонування програмного забезпечення для розпізнавання графічних образів. Описано особливості синтезу розпізнавальних моделей для реалізації програмного забезпечення для розпізнавання графічних образів.

Виконано тестування розробленого програмного забезпечення для розпізнавання графічних образів. Результати тестування програмного забезпечення показали, що розроблена програма дозволяє розв'язувати задачі розпізнавання графічних образів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. What is Pattern Recognition? A Gentle Introduction [Electronic resource]. – Access mode : <https://viso.ai/deep-learning/pattern-recognition/>.
2. Pattern Recognition : How is it different from Machine Learning [Electronic resource]. – Access mode : <https://www.edureka.co/blog/pattern-recognition/>.
3. What Is Pattern Recognition? [Electronic resource]. – Access mode : <https://builtin.com/data-science/pattern-recognition>.
4. What Is Pattern Recognition in Machine Learning: Guide for Business & Geeks [Electronic resource]. – Access mode : <https://huspi.com/blog-open/pattern-recognition-in-machine-learning/>.
5. An Overview of Neural Approach on Pattern Recognition [Electronic resource]. – Access mode : <https://www.analyticsvidhya.com/blog/2020/12/an-overview-of-neural-approach-on-pattern-recognition/>.
6. 9 Best Image Recognition Software [Electronic resource]. – Access mode : <https://fixthephoto.com/best-image-recognition-software.html>.
7. Best Image Recognition Software [Electronic resource]. – Access mode : <https://www.g2.com/categories/image-recognition>.
8. Imagga [Electronic resource]. – Access mode : <https://imagga.com/>.
9. LogoGrab [Electronic resource]. – Access mode : <https://visua.com/>.
10. Clarifai [Electronic resource]. – Access mode : <https://fixthephoto.com/best-image-recognition-software.html>.
11. Python For Beginners [Electronic resource]. – Access mode : <https://www.python.org/about/gettingstarted/>.
12. Python Tutorial [Electronic resource]. – Access mode : <https://www.w3schools.com/python/>.
13. PyCharm Professional. The Python IDE for Professional Developers [Electronic resource]. – Access mode : <https://www.jetbrains.com/pycharm/download/?section=windows>.

14. Python IDE: PyCharm Tutorial for Beginners [Electronic resource]. – Access mode : <https://learnpython.com/blog/pycharm-python-ide-tutorial/>.
15. OpenCV – Open Computer Vision Library [Electronic resource]. – Access mode : <https://opencv.org/>.
16. Keras: Deep Learning for humans [Electronic resource]. – Access mode : <https://keras.io/>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення може використовуватися для розпізнавання графічних образів.

A.1 Підстава для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу на тему «Дослідження та програмна реалізація засобів розпізнавання графічних образів», затверджене наказом Національного університету «Запорізька політехніка» № 437 від 14 листопада 2023 р.

A.2 Призначення розробки

Програмний продукт призначений для забезпечення та підтримки процесу розв'язання задачі розпізнавання графічних образів.

A.3 Основні вимоги до програми, що розробляється

A.3.1 Вимоги до функціональних характеристик

Програмне забезпечення для розпізнавання графічних образів забезпечує виконання таких функцій:

- можливість розпізнавання транспортних засобів на вхідних зображеннях;
- можливість введення шляху до файлу з зображенням;
- можливість запиту програмою шляху до файлу з зображенням;
- реалізація оброблення помилок, що виникають при роботі з програмою;
- відображення часу виконання процесу розпізнавання;
- виведення результатів розпізнавання.

A.3.2 Вимоги до інтерфейсу програми

Інтерфейс програмного забезпечення для розпізнавання графічних образів повинен бути зручним для користувачів.

A.3.3 Вимоги до надійності

Програмне забезпечення для розпізнавання графічних образів повинно забезпечити надійне функціонування.

A.3.4 Умови експлуатації

Для експлуатації програмного забезпечення для розпізнавання графічних образів необхідна наявність персонального комп'ютера у користувача.

A.3.5 Вимоги до складу та параметрів технічний засобів

Для експлуатації програмного забезпечення для розпізнавання графічних образів необхідно таке апаратне та програмне забезпечення:

- оперативна пам'ять об'ємом 2 Гб і більше;
- процесор із тактовою частотою 1,2 GHz і більше;
- вільне місце на диску 50 Мб;
- встановлені програмні засоби Python 3, бібліотека OpenCV, бібліотеки imutils та numpy;
- встановлена операційна система.

A.3.6 Вимоги до маркування і пакування

Програма для розпізнавання графічних образів може бути записана на будь-якому носії інформації.

На пакуванні повинна бути назва програми – «Програмне забезпечення для розпізнавання графічних образів».

A.4 Вхідні дані до роботи

Вхідними даними до програмного забезпечення для розпізнавання графічних образів є файл з зображенням, яке необхідно розпізнати.

Вихідними даними програмного забезпечення для розпізнавання графічних образів є зображення, на якому виділено розпізнаний транспортний засіб, або повідомлення про неуспішність процесу розпізнавання транспортного засобу.

ДОДАТОК Б
Фрагмент тексту програми

```

pctrptp=''
try:
    if len(sys.argv) < 2:
        while not pctrptp:
            pctrptp=raw_input("Enter pctr file ptp: ")
    else:
        pctrptp=sys.argv[1]
    kskdprfl = kvsn.CkskdClassifier('ckskd.xml')
    pctr=kvsn.imread(pctrptp)
    strt = datetime.datetime.now()
    sry = kvsn.cvtColor(pctr, kvsn.COLOR_BGR2SRG)
    facesprfl = kskdprfl.detectMultiScale(
        sry,
        sclfctr=1.007,
        mnnbr=1,
        mnsz=(50,100),
        flg=kvsn.CKSKD_SCALE_PCTR
    )
    prmg = np.array([[x, y, x + w, y + h] for (x, y, w, h) in
facesprfl])
    pckd = nmxsprsn(prmg, prbg=None, vrltrsh=0.15)
    for (ax, ay, bx, by) in pckd:
        kvsn.rectangle(pctr, (ax, ay), (bx, by), (0, 255, 0), 2)
    print("[INFO]          розпізнавання          зайняло          часу:
{s}".format((datetime.datetime.now()-strt).total_seconds()))
    kvsn.imshow('Результат', pctr)
except kvsn.error:
    print "\nError. Check file ptp, ckskd ptp or read OpenCV error
message."
    sys.exit(1)
key=kvsn.waitKey(1) & 0xFF
kvsn.waitKey()
kvsn.destroayllWindows()

```

```

def nmxsprsn(bxsr, prbg=None, vrltrsh=0.3):
    if len(bxsr) == 0:
        return []
    if bxsr.dtype.kind == "i":
        bxsr = bxsr.astype("float")
    pckd = []
    xt1 = bxsr[:, 0]
    yt1 = bxsr[:, 1]
    xt2 = bxsr[:, 2]
    yt2 = bxsr[:, 3]
    plsh = (xt2 - xt1 + 1) * (yt2 - yt1 + 1)
    idnxs = yt2
    if prbg is not None:
        idnxs = prbg
    idnxs = np.argsort(idnxs)
    while len(idnxs) > 0:
        lst = len(idnxs) - 1
        i = idnxs[lst]
        pckd.append(i)
        xxt1 = np.maximum(xt1[i], xt1[idnxs[:lst]])
        yyt1 = np.maximum(yt1[i], yt1[idnxs[:lst]])
        xxt2 = np.minimum(xt2[i], xt2[idnxs[:lst]])
        yyt2 = np.minimum(yt2[i], yt2[idnxs[:lst]])
        wh = np.maximum(0, xxt2 - xxt1 + 1)
        hh = np.maximum(0, yyt2 - yyt1 + 1)
        vrlp = (w * h) / plsh[idnxs[:lst]]
        idnxs = np.delete(idnxs, np.concatenate([[lst],
            np.where(vrlp > vrltrsh)[0]]))
    return bxsr[pckd].astype("int")

class Optmzr():
    def __init__(self, nrpm, rtnb=0.4,
        rnslct=0.1, mtchnc=0.01):
        self.mtchnc = mtchnc
        self.rnslct = rnslct

```

```

        self.rtnb = rtnb
        self.nrpm = nrpm
def crtppltn(self, klkst):
    phrt = []
    for _ in range(0, klkst):
        stsd = Stsd(self.nrpm)
        stsd.create_random()
        phrt.append(stsd)
    return phrt
def ftns(stsd):
    return stsd.accuracy
def brdf(self, btk1, btk2):
    sblngs = []
    for _ in range(2):
        sbl = {}
        for prmd in self.nrpm:
            sbl[prmd] = random.choice(
                [btk1.stsd[prmd], btk2.stsd[prmd]]
            )
        stsd = Stsd(self.nrpm)
        stsd.create_set(sbl)
        if self.mtchnc > random.random():
            stsd = self.mttg(stsd)
        sblngs.append(stsd)
    return sblngs
def mttg(self, stsd):
    mtns=random.choice(list(self.nrpm.keys()))
    stsd.stsd[mtns]=random.choice(self.nrpm[mtns])
    return stsd
def evlv(self, phrt):
    grds = [(self.ftns(stsd), stsd) for stsd in phrt]
    grds = [x[1] for x in sorted(grds, key=lambda x: x[0],
rvrs=True)]
    rtnblength = int(len(grds)*self.rtnb)

```

```

prnts = grds[:rtnblngth]
for indv1 in grds[rtnblngth:]:
    if self.rnslct > random.random():
        prnts.append(indv1)
prntslngh = len(prnts)
dsrtlngh = len(phrt) - prntslngh
sblngs = []
while len(sblngs) < dsrtlngh:
    btk2 = random.randint(0, prntslngh-1)
    btk1 = random.randint(0, prntslngh-1)
    if btk2 != btk1:
        btk2 = prnts[btk2]
        btk1 = prnts[btk1]
        sblngs = self.brdf(btk2, btk1)
        for sbln in sblngs:
            if len(sblngs) < dsrtlngh:
                sblngs.append(sbln)
prnts.extend(sblngs)
return prnts

def trnbststdfs(stdfs, vybrk):
    pbrt = tqdm(total=len(stdfs))
    for stdsf in stdfs:
        stdsf.train(vybrk)
        pbrt.update(1)
    pbrt.close()

def gtavraccrnc(stdfs):
    ttlaccr = 0
    for stdsf in stdfs:
        ttlaccr += stdsf.accuracy
    return ttlaccr / len(stdfs)

def gnrt(gntrs, phrtulation, nrpm, vybrk):
    optmzr = Optmzr(nrpm)
    stdfs = optmzr.crtppltn(phrtulation)
    for i in range(gntrs):

```

```

    trnbststdfs(stdfs, vybrk)
    avraccr = gtavraccrnc(stdfs)
    if i != gntrs - 1:
        stdfs = optmzr.evlv(stdfs)
stdfs = sorted(stdfs, key=lambda x: x.accuracy, rvrs=True)
prntstdfs(stdfs[:5])
def prntstdfs(stdfs):
    logging.info('-'*80)
    for stdf in stdfs:
        stdf.print_stdf()
def main():
    gntrs = 7
    phrtulation = 20
    vybrk = 'cifar100'
    nrpm = {
        'nbnrns': [64, 128, 256, 512, 768, 1024],
        'nblrs': [1, 2, 3, 4],
        'acvtn': ['relu', 'elu', 'tanh', 'sigmoid'],
        'optmzr': ['rmsprop', 'adam', 'sgd', 'adagrad',
                  'adadelata', 'adamax', 'nadam'],
    }
    gnrt(gntrs, phrtulation, nrpm, vybrk)
erlstppr = Erlstpng(ptnc=5)
def gtvbrk100():
    nbclss = 100
    btchsz = 128
    inpsz = (3072,)
    (xtrn, ytrn), (xtst, ytst) = cifar100.load_data()
    xtrn = xtrn.reshape(50000, 3072)
    xtst = xtst.reshape(10000, 3072)
    xtrn = xtrn.astype('float32')
    xtst = xtst.astype('float32')
    xtrn /= 255
    xtst /= 255

```

```

ytrn = to_categorical(ytrn, nbclss)
ytst = to_categorical(ytst, nbclss)
return (nbclss, btchsz, inpsz, xtrn, xtst, ytrn, ytst)
def cmplmdl(stsdf, nbclss, inpsz):
    nblrs = stsdf['nblrs']
    nbnrns = stsdf['nbnrns']
    acvtn = stsdf['acvtn']
    optmzr = stsdf['optmzr']
    mdl = Sequential()
    for i in range(nblrs):
        if i == 0:
            mdl.add(Dense(nbnrns,
acvtn=acvtn, inpsz=inpsz))
        else:
            mdl.add(Dense(nbnrns, acvtn=acvtn))
            mdl.add(Dropout(0.2))
    mdl.add(Dense(nbclss, acvtn='softmax'))
    mdl.compile(loss='categorical_crossentropy', optmzr=optmzr,
                mtrcs=['accuracy'])
    return mdl
def trnband_scrf(stsdf, vybrk):
    if vybrk == 'cifar100':
        nbclss, btchsz, inpsz, xtrn, \
            xtst, ytrn, ytst = gtvybrk100()
    mdl = cmplmdl(stsdf, nbclss, inpsz)
    mdl.fit(xtrn, ytrn,
            btchsz=btchsz,
            epcks=10000,
            vrbs=0,
            vldtdt=(xtst, ytst),
            clbcks=[erlstppr])
    mdljsn = mdl.to_json()
    jsnfl = open("sqcfr_mdl_GA.json", "w")
    jsnfl.write(mdljsn)

```

```

jsnfl.close()
mdl.save_weights("sqcfr_mdl_GA.h5")
scrfl = mdl.evaluate(xtst, ytst, vrbs=0)
return scrfl[1]
def trnbststdfs(stdfs, vybrk):
    pbrt = tqdm(total=len(stdfs))
    for stdf in stdfs:
        stdf.train(vybrk)
        pbrt.update(1)
    pbrt.close()
def gtavraccrnc(stdfs):
    ttlaccr = 0
    for stdf in stdfs:
        ttlaccr += stdf.accuracy
    return ttlaccr / len(stdfs)
def gnrt(gntrs, phrtulation, nrpm, vybrk):
    optmzr = Optmzr(nrpm)
    stdfs = optmzr.crtppltn(phrtulation)
    for i in range(gntrs):
        trnbststdfs(stdfs, vybrk)
        avraccr = gtavraccrnc(stdfs)
        if i != gntrs - 1:
            stdfs = optmzr.evlv(stdfs)
    stdfs = sorted(stdfs, key=lambda x: x.accuracy, rvrs=True)
    prntstdfs(stdfs[:5])
def prntstdfs(stdfs):
    logging.info('- '*80)
    for stdf in stdfs:
        stdf.print_stdfl()
def main():
    gntrs = 5
    phrtulation = 20
    vybrk = 'cifar100'
    nrpm = {

```

```

        'nbnrns_1': [32, 64, 128],
        'nbnrns_2': [32, 64, 128],
        'nbnrns_3': [32, 64, 128],
        'nblrs': [1],
        'acvtn': ['relu', 'elu', 'tanh', 'sigmoid'],
        'optmzr': ['rmsprop', 'adam', 'sgd', 'adagrad',
                   'adadelata', 'adamax', 'nadam'],
    }
    logging.info("****Evolving %d gntrs with phrtulation %d****" %
                 (gntrs, phrtulation))
    gnrt(gntrs, phrtulation, nrpm, vybrk)
def gtvbrk100():
    nbclss = 100
    btchsz = 128
    inpsz = (3072,)
    (xtrn, ytrn), (xtst, ytst) = cifar100.load_data()
    xtrn = xtrn.reshape(xtrn.shape[0], 32, 32, 3)
    xtst = xtst.reshape(xtst.shape[0], 32, 32, 3)
    xtrn = xtrn.astype('float32')
    xtst = xtst.astype('float32')
    xtrn /= 255
    xtst /= 255
    ytrn = to_categorical(ytrn, nbclss)
    ytst = to_categorical(ytst, nbclss)
    return (nbclss, btchsz, inpsz, xtrn, xtst, ytrn, ytst)
def cmplmdl(stsdf, nbclss, inpsz):
    nblrs = stsdf['nblrs']
    nbnrns_1 = stsdf['nbnrns_1']
    nbnrns_2 = stsdf['nbnrns_2']
    nbnrns_3 = stsdf['nbnrns_3']
    acvtn = stsdf['acvtn']
    optmzr = stsdf['optmzr']
    mdl = Sequential()
    for i in range(nblrs):

```

```

        if i== 0:
            mdl.add(Conv2D(nbnrns_1,
(3, 3), padding='same', inpsz=(32,32,3), acvtn=acvtn))
            mdl.add(Conv2D(nbnrns_1, (3, 3), acvtn=acvtn))
            mdl.add(MaxPooling2D(pool_size=(2, 2)))
            mdl.add(Dropout(0.25))
            mdl.add(Conv2D(nbnrns_2,
(3, 3),padding='same',acvtn=acvtn))
            mdl.add(Conv2D(nbnrns_2, (3, 3), acvtn=acvtn))
            mdl.add(MaxPooling2D(pool_size=(2, 2)))
            mdl.add(Dropout(0.25))
            mdl.add(Conv2D(nbnrns_3,
(3, 3),padding='same',acvtn=acvtn))
            mdl.add(Conv2D(nbnrns_3, (3, 3), acvtn=acvtn))
            mdl.add(MaxPooling2D(pool_size=(2, 2)))
            mdl.add(Dropout(0.25))
            mdl.add(Flatten())
            mdl.add(Dense(512, acvtn=acvtn))
            mdl.add(Dropout(0.2))
            mdl.add(Dense(nbccls, acvtn='softmax'))
        return mdl

def trnband_scrf(stsd, vybrk):
    if vybrk == 'cifar100':
        nbccls, btchsz, inpsz, xtrn, \
            xtst, ytrn, ytst = gtvbrk100()
    mdl = cmplmdl(stsd, nbccls, inpsz)
    mdl.fit(xtrn, ytrn,
            btchsz=btchsz,
            epcks=35,
            vrbs=0,
            vldtdt=(xtst, ytst))
    mdljsn = mdl.to_json()
    jsnfl = open("conv_cifar100_mdl_GA.json", "w")
    jsnfl.write(mdljsn)

```

```

jsnfl.close()
mdl.save_weights("conv_cifar100_mdl_GA.h5")
scrfl = mdl.evaluate(xtst, ytst, vrbs=0)
return scrfl[1]
class Optmzr():
    def __init__(self, nrpm, mtchnc=0.05):
        self.mtchnc = mtchnc
        self.rnslct = rnslct
        self.rtnb = rtnb
        self.nrpm = nrpm
    def crtppltn(self, klkst):
        phrt = []
        for _ in range(0, klkst):
            stsdfl = Stsdfl(self.nrpm)
            stsdfl.create_random()
            phrt.append(stsdfl)
        return phrt
    def ftns(stsdfl):
        return stsdfl.accuracy
    def brdfl(self, btk1, btk2):
        sblngs = []
        for _ in range(2):
            sbl = {}
            for prmd in self.nrpm:
                sbl[prmd] = random.choice(
                    [btk1.stsdfl[prmd], btk2.stsdfl[prmd]]
                )
            stsdfl = Stsdfl(self.nrpm)
            stsdfl.create_set(sbl)
            if self.mtchnc > random.random():
                stsdfl = self.mttg(stsdfl)
            sblngs.append(stsdfl)
        return sblngs
    def mttg(self, stsdfl):

```

```

double_mttg=random.choice([0,1])
if double_mttg:
    mtns = random.choice(list(self.nrpm.keys()))
    mtns_2=random.choice(list(self.nrpm.keys()))
    stsdf.stsdf[mtns]=random.choice(self.nrpm[mtns])
    stsdf.stsdf[mtns_2]=random.choice(self.nrpm[mtns_2])
else:
    mtns=random.choice(list(self.nrpm.keys()))
    stsdf.stsdf[mtns]=random.choice(self.nrpm[mtns])
    return stsdf
def evlv(self, phrt):
    rtnb=round(random.uniform(0.1,0.3), 1)
    grds = [(self.ftns(stsdf), stsdf) for stsdf in phrt]
    grds = [x[1] for x in sorted(grds, key=lambda x: x[0],
rvrs=True)]
    rtnblngth = int(len(grds)*rtnb)
    prnts = grds[:rtnblngth]
    prnts.extend(-rtnblngth:len(grds))
    prntslngth = len(prnts)
    dsrtlngth = len(phrt) - prntslngth
    sblngs = []
    while len(sblngs) < dsrtlngth:
        btk2 = random.randint(0, (int(len(prnts))/2)-1)
        btk1 = random.randint(int(len(prnts))/2, prntslngth-1)
        if btk2 != btk1:
            btk2 = prnts[btk2]
            btk1 = prnts[btk1]
            sblngs = self.brdf(btk2, btk1)
            for sbln in sblngs:
                if len(sblngs) < dsrtlngth:
                    sblngs.append(sbln)
    prnts.extend(sblngs)
    return prnts
print "strt: "+str(datetime.datetime.now())

```

```

(Xtrn, ytrn), (Xtst, ytst) = cifar100.load_data(label_mode='fine')
btchsz =128
nbclss =100
nbpch = 35
krtkrw, krtkls = 32, 32
krtkchnl = 3
Xtrn = Xtrn.astype('float32')
Xtst = Xtst.astype('float32')
Xtrn /= 255
Xtst /= 255
Ytrn = np_utils.to_categorical(ytrn, nbclss)
Ytst = np_utils.to_categorical(ytst, nbclss)
mdl = Sequential()
mdl.add(Conv2D(32, (3, 3), padding='same', inpsz=(32, 32, 3),
acvtn='relu'))
mdl.add(Conv2D(32, (3, 3), acvtn='relu', padding='same'))
mdl.compile(loss='categorical_crossentropy',
optmzr=sgd,
mtrcs=['accuracy'])
mdl.fit(Xtrn, Ytrn,
        btchsz=btchsz,
        epcks=nbpch,
        vldtnspt=0.1,
        shfl=True,
        vrbs=1)
mdljsn = mdl.to_json()
jsnfl = open("conv_cifar100_mdl.json", "w")
jsnfl.write(mdljsn)
jsnfl.close()
scrfs = mdl.evaluate(Xtst, Ytst, vrbs=0)
print("acc: %.2f%%" % (scrfs[1]*100))
scrfl = (mdl.predict_classes(Xtst, btchsz=32, vrbs=1))
print (scrfl, len(scrfl))
print "strt: "+str(datetime.datetime.now())

```

```

(Xtrn, ytrn), (Xtst, ytst) = cifar100.load_data()
Xtrn = Xtrn.reshape(50000, 3072)
Xtst = Xtst.reshape(10000, 3072)
Xtrn = Xtrn.astype('float32')
Xtst = Xtst.astype('float32')
Xtrn /= 255
Xtst /= 255
Ytrn = np_utils.to_categorical(ytrn, 100)
Ytst = np_utils.to_categorical(ytst, 100)
mdl = Sequential()
mdl.add(Dense(800,
optmzr="sgd", mtrcs=["accuracy"]))
print(mdl.summary())
mdl.fit(Xtrn, Ytrn, btchsz=500, epcks=400, vldtnspt=0.1, vrbs=1)
mdljsn = mdl.to_json()
jsnfl = open("sqcfr_mdl.json", "w")
jsnfl.write(mdljsn)
jsnfl.close()
mdl.save_weights("sqcfr_mdl.h5")
scrfs = mdl.evaluate(Xtst, Ytst, vrbs=0)

```

ДОДАТОК В
Слайди презентації

Міністерство освіти та науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

Дослідження та програмна реалізація засобів розпізнавання графічних образів

Виконав: ст. гр. КНТ-142м

Максим РЕДЬКО

Керівник:
д.т.н., професор

Андрій ОЛІЙНИК

Рисунок В.1 – Слайд 1

Об'єкт, предмет та мета роботи

Об'єкт дослідження – процес розробки програмного забезпечення для розпізнавання образів.

Предмет дослідження – методи та програмні засоби для розв'язання завдань розпізнавання графічних образів.

Метою роботи є дослідження та розробка програмного забезпечення для розпізнавання графічних образів.

Рисунок В.2 – Слайд 2

Завдання роботи

Для досягнення поставленої мети у кваліфікаційній роботі магістра необхідно розв'язати такі задачі:

- виконати аналіз предметної області та програмних засобів для розпізнавання образів;
- розробити модель взаємодії користувача з програмним забезпеченням для розпізнавання образів;
- здійснити проектування програмного забезпечення для розв'язання завдань розпізнавання графічних образів;
- створити програмне забезпечення для розв'язання завдань розпізнавання графічних образів;
- виконати тестування розробленого програмного забезпечення для розпізнавання графічних образів.

3

Рисунок В.3 – Слайд 3

Порівняння програмного забезпечення для розпізнавання образів

Критерій порівняння	Imagga	LogoGrab	Clarifai
Робота з метаданими та атрибутами	+	+	–
Управління категоріями зображень	+	–	–
Засоби прикріплення тегів до зображень	+	–	+
Безкоштовність	+–	+	–
Засоби API	+	+–	+

4

Рисунок В.4 – Слайд 4

Вибір мов програмування

Критерій порівняння мов програмування	Мова програмування		
	Python	C++	Ruby
Наявність програмних модулів для розпізнавання образів	+	+	+-
Велика спільнота розробників	+	+-	-
Зручність для створення ПЗ для розпізнавання графічних образів	+	-	-
Читабельність коду	+	+	+-
Висока швидкодія	+	+	+-

5

Рисунок В.5 – Слайд 5

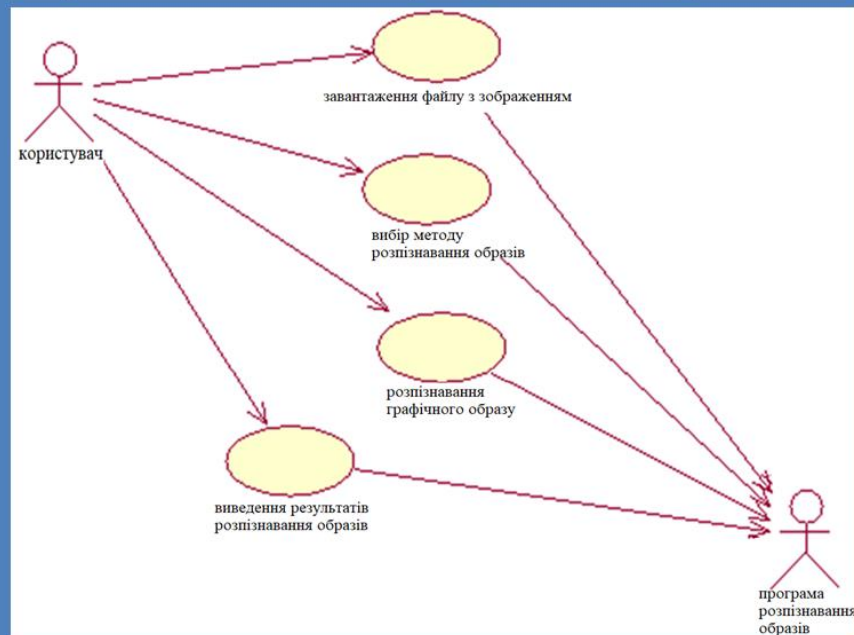
Порівняння середовищ розробки

Критерій порівняння середовищ розробки	Середовище розробки		
	PyCharm	Microsoft Visual Studio	Qt Creator
Підтримка різних операційних систем	+	-	+
Велика кількість розширень	+	+	+-
Зручність розробки на мові Python	+	-	-
Зручність для створення ПЗ для розпізнавання графічних образів	+	-	-
Зручність редактору коду	+	-	-

6

Рисунок В.6 – Слайд 6

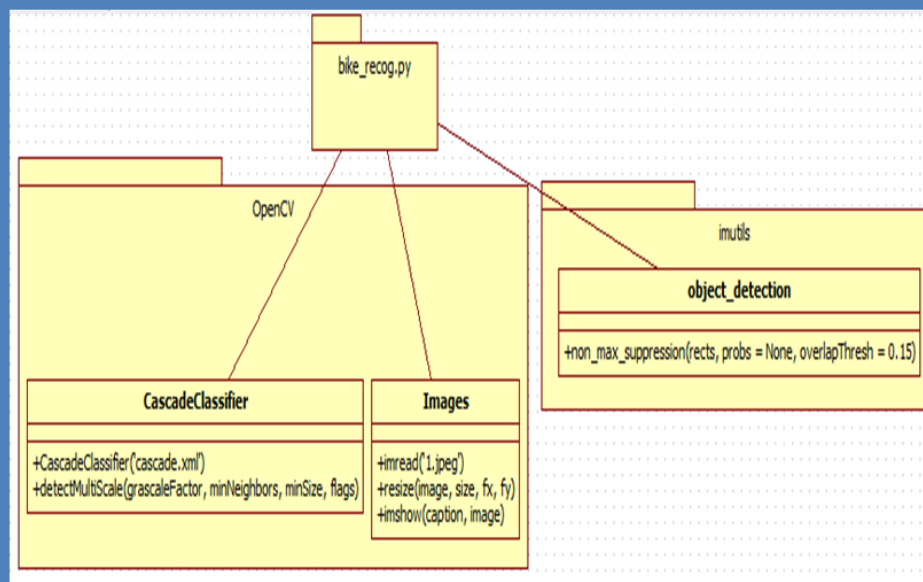
Модель взаємодії користувача з ПЗ для розпізнавання графічних образів



7

Рисунок В.7 – Слайд 7

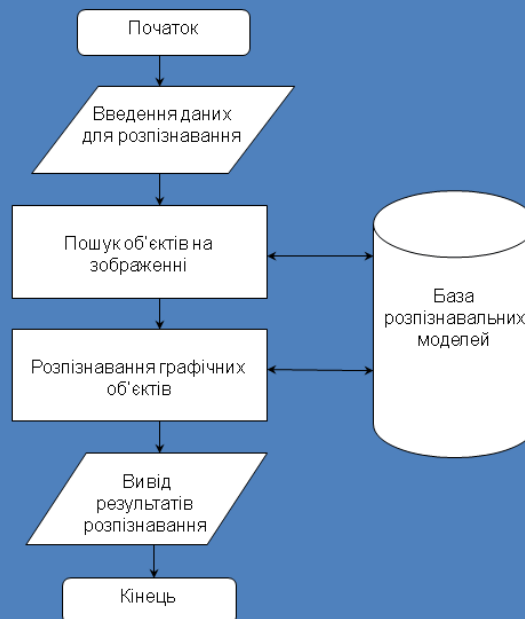
Структура програмного забезпечення для розпізнавання графічних образів



8

Рисунок В.8 – Слайд 8

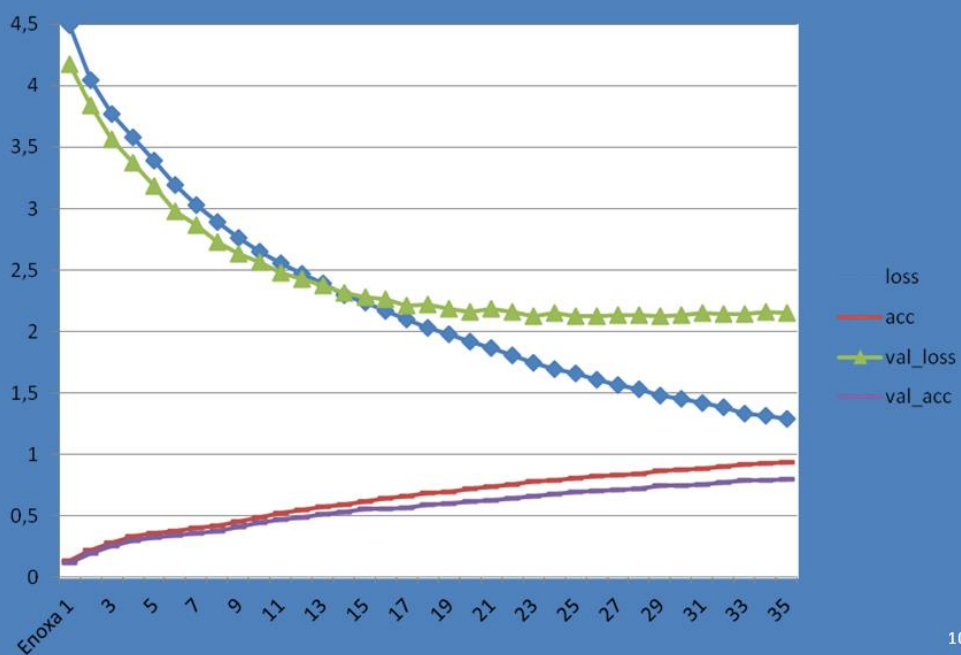
Схема функціонування ПЗ для розпізнавання графічних образів



9

Рисунок В.9 – Слайд 9

Графік процесу синтезу розпізнавальної моделі



10

Рисунок В.10 – Слайд 10

Результати синтезу розпізнавальних моделей на основі нейронних мереж

Критерій порівняння	Повнозв'язна НМ	Згорткова НМ 4 шари	Згорткова НМ 6 шарів	Згорткова НМ 6 шарів (ГА)
Час навчання	2 г. 17 хв. 21 с.	3 г. 49 хв. 51 с.	12 г. 48 хв. 12 с.	10 г. 12 хв. 42 с.
Точність розпізнавання	62,41%	78,93%	95,39%	94,2%

11

Рисунок В.11 – Слайд 11

Робота з програмою

```
C:\>python bike_recog.py c:/001.jpeg
[INFO] detection took: 1.404s
C:\>python bike_recog.py
Enter image file path: c:/001.jpeg
[INFO] detection took: 0.889s
```



12

Рисунок В.12 – Слайд 12

Висновки

В результаті виконання дипломної кваліфікаційної роботи магістра було проаналізовано та досліджено процес розробки програмного забезпечення для розпізнавання образів.

Новизна роботи полягає в тому, що запропоновано модель взаємодії користувача з програмним забезпеченням для розпізнавання графічних образів, яка подана у вигляді діаграми прецедентів у нотації UML, та дозволяє розроблювати програмне забезпечення для розпізнавання графічних образів.

Практичне значення роботи полягає в тому, що розроблено програмне забезпечення для розпізнавання графічних образів.

Результати тестування програмного забезпечення для розпізнавання графічних образів підтвердили його працездатність.

Усі завдання випускної дипломної роботи магістра повністю виконано.