Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

Кваліфікаційна робота бакалавра

на тему: «Розробка програмного забезпечення для пошуку зображень з використанням нейронних мереж»

Виконав: студент групи ІПЗ20-2
Спеціальність 121 «Інженерія програмного забезпечення»
Павлухін Кирило Олександрович
(прізвище та ініціали)
Керівник д.е.н., проф. Корнєєв М.В.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент Університет митної справи та фінансів
(місце роботи)
Доцент кафедри кібербезпеки та інформаційних технологій
(посада)
к.т.н., доцент Клим В.Ю.
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

АНОТАЦІЯ

Павлухін К.О. Розробка програмного забезпечення для пошуку зображень з використанням нейронних мереж.

Кваліфікаційна робота на здобуття освітнього ступеня бакалавра за спеціальністю 121 «Інженерія програмного забезпечення» – Університет митної справи та фінансів, Дніпро, 2024.

Кваліфікаційна робота присвячена розробці системи пошуку зображень з використанням нейронних мереж. Актуальність роботи зумовлена швидким зростанням обсягу візуальної інформації, що потребує ефективних методів її обробки та аналізу. Система пошуку зображень на основі нейронних мереж дозволяє автоматично класифікувати зображення та забезпечувати їх швидкий пошук, що має велике значення для різних галузей, включаючи медицину, безпеку, промисловість та розваги. У процесі дослідження було проведено аналіз сучасних методів розпізнавання зображень, включаючи методи SIFT, SURF, ORB та AKAZE, а також згорткові нейронні мережі. Вибір останніх обґрунтовано їхньою високою точністю та здатністю автоматично виділяти важливі ознаки з візуальних даних.

Для реалізації системи було обрано мову програмування Python та бібліотеку PyTorch, що забезпечує високу гнучкість та продуктивність розробки. Система включає тришарову архітектуру, що розділяє програмне забезпечення на інтерфейсний шар, бізнес-логіку та шар даних, забезпечуючи тим самим спрощення розробки, підтримки та масштабування системи.

Розроблена система демонструє високу ефективність та продуктивність, а також забезпечує гнучкість у використанні та подальшому вдосконаленні. Результати роботи можуть бути використані для подальшого розвитку подібних систем та автоматизації обробки візуальної інформації в інших сферах.

Ключові слова: пошук зображень, нейронні мережі, згорткові нейронні мережі, Python, PyTorch.

ABSTRACT

Pavlukhin K.O. Development of software for image search using neural networks.

Bachelor's thesis for obtaining a degree in Software Engineering, specialty 121. – University of Customs and Finance, Dnipro, 2024.

This qualification work is dedicated to the development of an image search system using neural networks. The relevance of the work is due to the rapid increase in the volume of visual information, which requires effective methods for its processing and analysis. An image search system based on neural networks allows automatic classification of images and ensures their quick search, which is of great importance for various fields, including medicine, security, industry, and entertainment.

The study analyzed modern methods of image recognition, including SIFT, SURF, ORB, and AKAZE methods, as well as convolutional neural networks (CNN). The choice of CNNs is justified by their high accuracy and ability to automatically extract important features from visual data.

For the implementation of the system, the Python programming language and the PyTorch library were chosen, providing high flexibility and productivity in development. The system includes a three-layer architecture that divides the software into a presentation layer, business logic layer, and data layer, thereby simplifying the development, maintenance, and scaling of the system.

The developed system demonstrates high efficiency and productivity, as well as flexibility in use and further improvement. The results of the work can be used for the further development of similar systems and the automation of visual information processing in other fields.

Keywords: image search, neural networks, convolutional neural networks, Python, PyTorch.

ЗМІСТ

ВСТУП

Актуальність проблеми. У сучасному світі обсяг візуальної інформації стрімко зростає, що зумовлює необхідність ефективних методів її обробки та аналізу. Здатність швидко і точно розпізнавати та класифікувати зображення має вирішальне значення для багатьох галузей, включаючи медицину, безпеку, промисловість та розваги. Нейронні мережі, зокрема згорткові нейронні мережі (CNN), стали одним із найбільш потужних інструментів для вирішення цих задач завдяки своїй здатності автоматично виділяти важливі ознаки з візуальних даних та забезпечувати високу точність класифікації.

Розробка системи пошуку зображень з використанням нейронних мереж є актуальною у контексті сучасного розвитку штучного інтелекту та машинного навчання. Такі системи можуть значно покращити ефективність і продуктивність у різних сферах, де необхідно аналізувати великі обсяги зображень. Наприклад, у медичній діагностиці автоматичне розпізнавання зображень може допомогти лікарям швидше і точніше виявляти патології. У системах безпеки – забезпечити надійний контроль доступу та ідентифікацію осіб. У промислових додатках – автоматизувати контроль якості продукції.

Створення нейронної мережі складається з кількох етапів, починаючи з визначення її архітектури до навчання та оцінки результатів. Існує правило: якщо задачу можна вирішити за допомогою традиційного алгоритмічного програмування, використання ШІ не виправдане. Однак, якщо задача не може бути вирішена жодним відомим алгоритмом, використання нейронної мережі може бути доцільним.

Класифікація зображень за допомогою нейронних мереж полягає у присвоєнні категорії або мітки зображенню на основі його вмісту. Цей процес вимагає використання нейронної мережі, яка навчилася розпізнавати об'єкти або патерни на зображеннях.

На сьогодні існує багато методів і технологій для розпізнавання зображень, однак вони потребують подальшого вдосконалення для

підвищення точності та швидкості обробки даних. Вибір оптимальних інструментів та технологій, таких як Python та бібліотека PyTorch, є критичним для успішної реалізації проекту, оскільки це забезпечує гнучкість та продуктивність розробки.

Таким чином тема кваліфікаційної роботи «Розробка програмного забезпечення для пошуку зображень з використанням нейронних мереж» є актуальною.

Метою роботи є автоматизація розпізнавання зображень.

Методи дослідження: обробка та аналіз інформації, методи проектування та розробки нейронних мереж та веб-додатків.

У відповідності до поставленої мети в кваліфікаційній роботі поставлені наступні завдання дослідження:

1. Проаналізувати технічні засоби, що застосовуються для розробки нейронних мереж.

2. Розробити архітектуру веб-додатку з системою пошуку зображень.

3. Розробити програмне забезпечення для пошуку зображень.

4. Провести тестування розробленого веб- додатку.

Об'єктом дослідження є розробка програмного забезпечення для розпізнавання образів.

Предметом дослідження є технології розробки веб-додатку для розпізнавання образів з використанням нейронних мереж.

Робота складається зі вступу, 3-х розділів, висновків, списку використаних джерел з 14 найменувань. Обсяг роботи 60 сторінок кваліфікаційної роботи, 25 рисунків.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналіз існуючих публікацій

У машинному навчанні, розпізнаванні образів та обробці зображень ключовим етапом є виділення ознак з вихідних даних. Цей процес перетворює початковий набір вимірювань у більш інформативний та компактний набір, що полегшує подальше навчання та узагальнення. Виділення ознак також робить дані більш зрозумілими для людей і часто супроводжується зменшенням розмірності даних.

Методи розпізнавання графічних образів можна умовно розділити на три основні групи: попередня фільтрація та підготовка зображення, логічна обробка результатів фільтрації, та алгоритми прийняття рішень на основі логічної обробки. Важливо відзначити, що межі між цими групами досить умовні, оскільки для вирішення конкретних завдань можуть знадобитися різні комбінації методів з цих груп. Іноді для досягнення бажаного результату достатньо використовувати методи лише з однієї групи.

Перша група методів фільтрації спрямована на виділення областей на зображенні без детального аналізу. Ці методи застосовують однакове перетворення до всіх точок зображення, при цьому аналіз самого зображення не проводиться. Проте точки, що пройшли фільтрацію, можна розглядати як області з особливими характеристиками. До прикладу, методи розмивання та контрастування зображення.

Друга група методів зосереджена на пошуку особливих точок, які залишаються стабільними при незначних змінах освітлення та невеликих рухах об'єктів. Такі точки часто використовуються для навчання та класифікації типів об'єктів. Наприклад, класифікатори для розпізнавання пішоходів або людей часто будуються на основі таких точок. Відомі методи вейвлетів можуть бути основою для знаходження таких точок. До цієї групи

належать точки, знайдені за допомогою гістограм спрямованих градієнтів та інших методів, як-от методи каннелярних операторів.

Третя група методів здатна знаходити стабільні точки навіть при повороті зображення. Методи, такі як SURF і SIFT, можуть виявляти особливі точки, що залишаються стабільними при різних трансформаціях, таких як повороти та зміни масштабу. Ці методи використовують складні математичні перетворення для виявлення стійких ознак, що робить їх особливо корисними для задач, де потрібна висока надійність.

Комбінування цих методів дозволяє досягати високої точності розпізнавання графічних образів, адаптуючи рішення до конкретних потреб завдання та характеристик зображення. Кожна з груп методів робить свій внесок у загальний процес, забезпечуючи багатогранний підхід до аналізу та обробки зображень [1].

1.2. Аналіз методів та підходів розпізнавання зображень

Існує кілька відомих методів розпізнавання зображень, які широко використовуються в комп'ютерному зорі, зокрема SIFT, SURF, ORB та AKAZE. Ці алгоритми призначені для виявлення та опису ключових точок на зображеннях і використовуються у різноманітних застосуваннях, таких як розпізнавання об'єктів, зшивання зображень і виявлення характерних особливостей. Розглянемо докладніше деякі з цих методів [2].

Метод SIFT (Scale-Invariant Feature Transform) – це алгоритм комп'ютерного зору, розроблений Девідом Лоу у 1999 році, для виділення та опису ключових особливостей зображень. Він виявився дуже ефективним для розпізнавання об'єктів і вирішення проблеми інваріантності до масштабу та орієнтації. Основні принципи методу SIFT включають:

- Виділення ключових точок

Алгоритм аналізує зображення на різних масштабах і виявляє унікальні точки для подальшого порівняння.

– Обчислення дескрипторів ключових точок

Для кожної ключової точки SIFT обчислює векторний дескриптор, що описує сусідню область точки. Це робить дескриптор стійким до змін масштабу, орієнтації, освітлення та інших факторів.

– Порівняння ключових точок

Обчислені дескриптори можна порівнювати для визначення схожості і вирішення задач розпізнавання або відстеження об'єктів на рухомих зображеннях.

– Інваріантність до масштабу і орієнтації

SIFT забезпечує інваріантність до змін масштабу та орієнтації, що робить його ефективним для розпізнавання об'єктів навіть при різних умовах.

Метод SURF (Speeded-Up Robust Features) – це алгоритм, розроблений Гербертом Байєм та Віктором Лепеттем у 2006 році, який є покращеною версією методу SIFT. Основні характеристики методу SURF включають:

- Швидкість обчислень

SURF має високу швидкість обчислень завдяки ефективним методам обчислення градієнтів та оцінки Гессіану.

- Інваріантність до масштабу та орієнтації

Подібно до SIFT, SURF забезпечує інваріантність до змін масштабу та орієнтації об'єктів.

- Стійкість до змін контрастності

SURF демонструє стійкість до змін контрастності на зображеннях, що дозволяє ефективно виявляти та описувати об'єкти навіть при зміні освітлення.

- Дескриптори ключових точок

SURF генерує числові вектори-дескриптори для кожної ключової точки, які описують оточення точок на зображенні [3].

Хоча алгоритми SIFT, SURF, ORB, та AKAZE не ґрунтуються на традиційних методах штучного інтелекту або машинного навчання, вони використовують математичні та сигнальні методи для виявлення та опису

ключових точок. Проте, деякі з цих алгоритмів, такі як SURF і AKAZE, можуть використовувати техніки машинного навчання у своїх конструкціях.

Ці класичні методи комп'ютерного зору застосовуються в задачах, пов'язаних з аналізом зображень, включаючи розпізнавання облич, відстеження об’єктів, панорамне зшивання зображень та інші. На відміну від сучасних підходів на основі глибокого навчання, такі як згорткові нейронні мережі (CNN), які широко використовують штучний інтелект та машинне навчання, ці класичні методи залишаються важливими інструментами в комп’ютерному зорі.

1.3. Опис процесу класифікації зображень

Класифікація зображень за допомогою нейронних мереж є процесом, де зображенню присвоюється певна категорія або мітка залежно від вмісту, який воно представляє. Цей процес залежить від нейронної мережі, що була тренована для ідентифікації різних об'єктів та візерунків у зображеннях.

Основні кроки у процесі класифікації зображень за допомогою нейронних мереж включають:

1) Підготовка даних

Необхідно адаптувати зображення до формату, який може бути оброблений нейронною мережею. Це означає зміну розміру зображень, нормалізацію значень пікселів для стандартизації даних і видалення будь-якого візуального шуму.

2) Вибір архітектури мережі

Важливо вибрати адекватну архітектуру нейронної мережі, що відповідає завданню класифікації. Згорткові нейронні мережі (CNN) є популярним вибором для аналізу зображень через їхню ефективність у виявленні візуальних патернів.

3) Процес навчання

Навчання мережі включає введення великої кількості зображень з відомими мітками для адаптації вагових коефіцієнтів таким чином, щоб зменшити помилку прогнозування.

4) Тестування та оцінка

Оцінка ефективності мережі відбувається шляхом тестування на незалежних даних. Це дозволяє порівняти передбачені мережею мітки з фактичними, і таким чином оцінити точність класифікації.

5) Інференція

На цьому етапі навчена та протестована модель використовується для класифікації нових зображень, які раніше не були використані під час тренування.[4].

Класифікація зображень за допомогою нейронних мереж знаходить широке застосування в таких сферах, як розпізнавання об'єктів на фотографіях, медична діагностика, автономні транспортні засоби, фільтрація спаму та інші області, де необхідно аналізувати візуальні дані для прийняття рішень.

Розглянемо детально процес класифікації зображень, використовуючи бібліотеку PyTorch. Він є потужним інструментом для навчання нейронних мереж, зокрема для задачі розпізнавання та класифікації зображень. Ось основні етапи такого процесу:

1) Підготовка даних

Спочатку потрібно підготувати набір даних для тренування та тестування. Це включає завантаження зображень, попередню обробку (зміну розміру, нормалізацію тощо) та розподіл на тренувальний і тестовий набори. PyTorch забезпечує зручні інструменти для цієї мети, дозволяючи легко працювати з даними.

2) Вибір архітектури мережі

Вибір відповідного типу нейронної мережі є важливим кроком. Для класифікації зображень найчастіше використовуються згорткові нейронні

мережі (CNN), такі як ResNet, VGG або AlexNet. Ці архітектури спеціально розроблені для роботи з зображеннями і показують високу ефективність.

3) Створення моделі

Після вибору архітектури необхідно визначити модель у PyTorch. Це може бути власна модель або адаптація попередньо натренованої моделі до конкретного завдання. PyTorch надає широкий вибір попередньо натренованих моделей, що можуть бути легко налаштовані для ваших потреб.

4) Навчання моделі

Навчання моделі включає використання тренувального набору даних для оптимізації вагових коефіцієнтів нейронної мережі. Застосовуються функції втрат та методи оптимізації, такі як зворотне поширення помилки (backpropagation), для мінімізації помилки передбачень і покращення продуктивності моделі.

5) Тестування та оцінка

Після етапу навчання модель тестується на незалежному тестовому наборі даних. Для оцінки ефективності моделі обчислюються метрики, такі як точність, відгук, точність та матриця плутанини. Це дозволяє зрозуміти, наскільки добре модель виконує свої завдання.

6) Інференція

Після завершення навчання та тестування модель готова до використання для класифікації нових зображень, які не брали участі у процесі тренування. Це дозволяє застосовувати модель в реальних умовах для автоматизованого розпізнавання образів.

OpenCV відомий як класичний інструмент для обробки зображень, але його пряме призначення не включає тренування нейронних мереж. Проте, він ефективно використовується для підготовки зображень перед їх аналізом за допомогою нейронних мереж.Загальний підхід до класифікації зображень з використанням OpenCV:

1) Завантаження та підготовка зображень

OpenCV використовується для завантаження зображень та їх попередньої обробки, включаючи зміну розміру, нормалізацію інтенсивності пікселів і видалення шуму. Ці підготовчі операції забезпечують якість зображень для подальшої обробки.

2) Вибір нейронної мережі для класифікації

Для класифікації зображень використовуються попередньо навчені нейронні мережі, такі як MobileNet, Inception або ResNet, доступні у фреймворках для машинного навчання, як TensorFlow чи PyTorch. Ці моделі спеціалізуються на розпізнаванні об'єктів і складних завданнях класифікації.

3) Інференція

Оброблені зображення з OpenCV використовуються для інференції за допомогою обраної нейронної мережі. Модель проводить класифікацію та надає передбачені мітки для кожного зображення.

4) Оцінка та візуалізація результатів

Результати класифікації оцінюються шляхом порівняння передбачених міток зі справжніми. Ефективність моделі вимірюється за допомогою таких метрик, як точність, відгук та точність. Для подальшого аналізу результати можна візуалізувати на вихідних зображеннях [5].

Порівнюючи OpenCV та PyTorch, можна зробити висновок, що етапи обробки подібні, але PyTorch має перевагу у можливостях навчання та використання нейронних мереж. Для ефективного розпізнавання об'єктів на зображеннях важливо використовувати згорткові нейронні мережі (CNN), які добре підходять для цієї задачі.

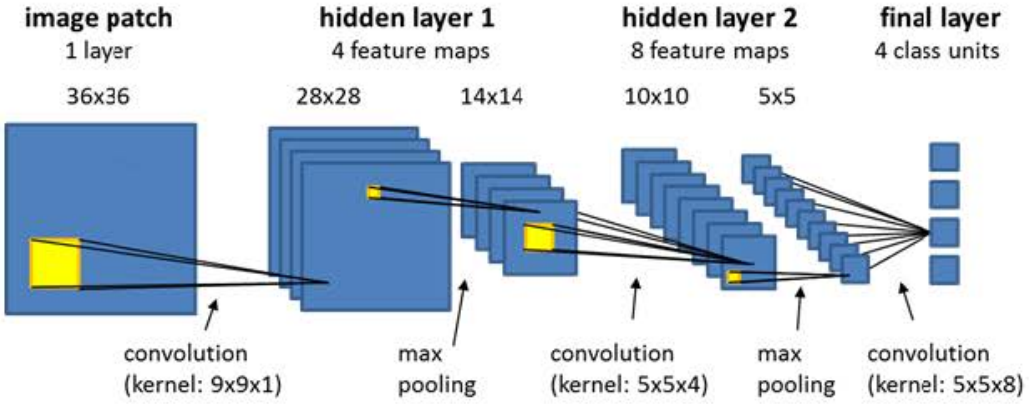

Рисунок 1.1 – Прицнип роботи згорткової нейронної мережі

Такі мережі виявилися значно ефективнішими у обробці зображень порівняно з традиційними багатошаровими перцептронами (MLP). Завдяки своїй архітектурі вони можуть автоматично визначати та використовувати важливі ознаки у вхідних даних, що робить обробку зображень більш ефективною та точною.

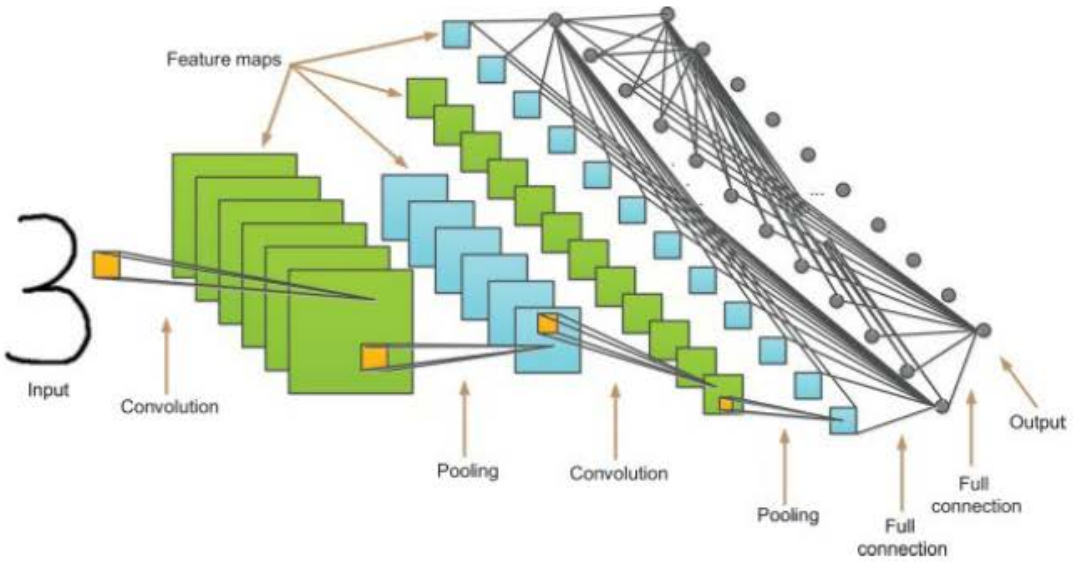

Рисунок 1.2 – Архітектура згорткових нейронних мереж

Згорткова нейронна мережа (CNN) складається з трьох основних компонентів: згортковий шар, шар об’єднання та повністю зв'язаний шар.

Згортковий шар є основним компонентом структури CNN і відіграє ключову роль у процесі обробки даних. Цей шар виконує точковий добуток

між двома матрицями: першою, відомою як ядро або фільтр, яка містить набір параметрів, що визначаються під час навчання мережі, та другою, яка представляє собою обмежену частину рецептивного поля. Важливо зазначити, що ядро має менші просторові розміри порівняно з вихідним зображенням, але його глибина відповідає кількості каналів зображення. Наприклад, для зображення з трьома каналами (RGB) ядро має меншу висоту та ширину, але його глибина охоплює всі три канали зображення.

Таким чином, згортковий шар забезпечує механізм виявлення та виділення ключових ознак у вхідних зображеннях, що робить його основним компонентом обробки зображень у CNN.

Машинне навчання для розпізнавання, наприклад, цифр значно перевершує підходи, що базуються на правилах або евристиках. Використання великого обсягу даних для налаштування параметрів адаптивної моделі забезпечує набагато вищу ефективність. Навчальна множина {x1,...,xN} використовується для оптимізації параметрів моделі. Категорії цифр у цій множині відомі заздалегідь і зазвичай маркуються вручну.

Результат роботи алгоритму машинного навчання виражається функцією y(x), яка приймає нове зображення x як вхід і генерує вихідний вектор y, що кодується аналогічно цільовим вектором. Точна форма функції y(x) визначається під час навчання на основі навчальних даних. Після навчання модель може класифікувати нові зображення з тестового набору. Здатність правильно класифікувати нові приклади, що відрізняються від тих, що використовувалися для навчання, називається узагальненням. У реальних застосуваннях варіативність вхідних векторів настільки велика, що навчальні дані можуть включати лише невелику частину всіх можливих вхідних векторів, тому узагальнення є центральною метою розпізнавання образів.

У більшості практичних застосувань вхідні дані попередньо обробляються з метою перетворення їх у новий простір змінних, де розпізнавання образів стає простішим. Наприклад, у задачі розпізнавання цифр зображення зазвичай перетворюються і масштабуються таким чином,

щоб кожна цифра містилася в рамці фіксованого розміру. Це зменшує варіативність всередині кожного класу цифр і полегшує подальший аналіз.

Цей етап попередньої обробки також називається виділенням ознак. Нові тестові дані повинні проходити ту саму попередню обробку, що й навчальні дані. Попередня обробка може також прискорити обчислення. Наприклад, для розпізнавання облич у реальному часі у відеопотоці високої роздільної здатності потрібно обробляти величезну кількість пікселів за секунду. Використання корисних ознак, які швидко обчислюються, але зберігають дискримінаційну інформацію, є важливим. Ці ознаки потім використовуються як вхідні дані для алгоритму розпізнавання образів.

Під час попередньої обробки важливо зберігати інформацію, яка є критичною для розв'язання задачі. Інакше загальна точність системи може знизитися. Навчання за наявності вхідних векторів з відповідними цільовими векторами називається керованим навчанням.

Розробка ефективної системи розпізнавання образів вимагає ретельного вибору архітектури нейронної мережі, оптимізації її параметрів і обробки даних. Застосування сучасних методів машинного навчання дозволяє досягати високої точності в розпізнаванні та класифікації, забезпечуючи адаптацію моделей до нових умов і варіацій у даних. Важливо також враховувати, що успішна реалізація систем розпізнавання образів залежить не тільки від вибору алгоритмів, але й від якості та обсягу навчальних даних, які використовуються для тренування моделей.

1.4. Аналіз методів оптимізації пошуку зображень

Аналіз сучасних методів прискорення пошуку графічних зображень включає детальне вивчення різноманітних технік і алгоритмів, які спрямовані на підвищення продуктивності та точності обробки великих масивів графічних даних.

Одним із центральних аспектів цього процесу є використання дескрипторів зображень – математичні характеристики або ознаки, які витягуються з зображення для його опису. Дескриптори, такі як SIFT (Scale Invariant Feature Transform), SURF (Speeded Up Robust Features) та CNN (Convolutional Neural Networks), дозволяють ефективно вилучати унікальну та стійку до змін інформацію з зображень. SIFT і SURF є методами, що виділяють ключові точки зображення, стійкі до змін масштабу, поворотів та освітлення, тоді як CNN автоматично вивчають складні залежності між ознаками зображень. Це сприяє точному представленню та порівнянню зображень, що є критично важливим для пошуку схожих об’єктів або сцен у великих наборах даних.

Метрики схожості – математичні методи, що використовуються для оцінки ступеня схожості між двома об'єктами. Вони допомагають оцінити ступінь схожості між дескрипторами, визначаючи наскільки подібними є зображення. Це дозволяє оцінити відповідність зображень пошуковому запиту та забезпечити високу точність результатів.

Алгоритми індексації відіграють ключову роль у прискоренні пошуку. Наприклад, методи хешування використовують функції хешування для призначення унікального коду (хешу) кожному зображенню, що дозволяє швидко виявляти дублікати або схожі зображення за їхніми хеш-кодами. Індексація деревами:

- k–d дерева, розбиває простір даних на рекурсивні підпростори, що дозволяє швидше здійснювати пошук.
- R–дерева групують об’єкти в області простору, що також сприяє ефективному пошуку.

Графові методи, які використовують алгоритми глибинного пошуку (DFS) або ширинного пошуку (BFS), також ефективні для виявлення схожих зображень.

Методи масштабованого пошуку, такі як використання розподілених систем і GPU-прискорення, значно впливають на швидкість і ефективність пошуку. Розподілені системи дозволяють паралельно обробляти та шукати

зображення на різних вузлах системи, що забезпечує ефективне використання ресурсів і швидший пошук у великих обсягах графічної інформації. Використання графічних процесорів (GPU) для прискорення обчислень у задачах пошуку зображень дозволяє обробляти багато операцій паралельно, що значно підвищує ефективність обробки зображень і виконання алгоритмів пошуку.

Аспекти обробки зображень, такі як оптимізація обчислень і використання паралельних обчислень, також є важливими для підвищення швидкості обробки. Методи оптимізації обчислень, як-от пакетна обробка зображень, включають обробку кількох зображень одночасно, що покращує швидкодію. Паралельні обчислення, реалізовані на рівні обчислень, фільтрів або операцій над кількома зображеннями одночасно, сприяють прискоренню аналізу зображень і забезпечують ефективний розподіл завдань між різними обчислювальними одиницями.

Deep Learning – тип машинного навчання, що використовує багатошарові нейронні мережі для вивчення складних моделей і залежностей у даних, значно покращує процес пошуку графічних зображень. Нейронні мережі можуть ефективно вилучати ознаки та автоматично вивчати складні залежності між зображеннями, що підвищує якість розпізнавання та класифікації зображень. Крім того, використання методів машинного навчання для попередньої обробки зображень дозволяє покращити якість і швидкість пошуку. Це включає автоматичне визначення оптимальних параметрів обробки, вилучення шуму, нормалізацію зображень та їх підготовку до подальшого аналізу.

Ці підходи спрямовані на створення більш ефективних і швидких систем пошуку графічних зображень у реальному часі, забезпечуючи високий рівень точності та швидкодії, що є критичним для сучасних застосунків і технологій обробки зображень.

1.5. Висновки до першого розділу. Постановка задач дослідження.

У першому розділі кваліфікаційної роботи було проведено ґрунтовний аналіз предметної області, що включає розгляд існуючих публікацій, методів та підходів до розпізнавання зображень. Встановлено, що розпізнавання образів та обробка зображень є важливими етапами у машинному навчанні, оскільки вони дозволяють перетворювати початкові дані у більш інформативний та компактний формат.

Етап попередньої обробки також називають вилученням ознак. Нові тестові дані повинні проходити ту саму попередню обробку, що й навчальні дані. Попередня обробка може також прискорити обчислення. Наприклад, для розпізнавання облич у реальному часі у відеопотоці високої роздільної здатності потрібно обробляти величезну кількість пікселів за секунду. Натомість використовують корисні ознаки, які швидко обчислюються, але зберігають дискримінаційну інформацію. Ці ознаки потім використовуються як вхідні дані для алгоритму розпізнавання образів.

Було розглянуто основні методи розпізнавання графічних образів, такі як SIFT, SURF, ORB та AKAZE, які широко використовуються у комп'ютерному зорі. Ці методи дозволяють ефективно виявляти та описувати ключові точки на зображеннях, що є важливим для таких задач, як розпізнавання об'єктів, зшивання зображень та виявлення характерних особливостей.

Проаналізовані наступні аспекти для прискорення пошуку: ключові функції пошуку, алгоритми індексації, аспекти обробки зображень, застосування машинного навчання.

Також проаналізовано процес класифікації зображень за допомогою нейронних мереж, зокрема, згорткових нейронних мереж (CNN). Встановлено, що згорткові нейронні мережі мають значні переваги у порівнянні з традиційними багатошаровими перцептронами (MLP), оскільки вони можуть

автоматично визначати та використовувати важливі ознаки у вхідних даних, що робить обробку зображень більш ефективною та точною.

Проведений аналіз дозволяє зробити висновок, що для ефективного розпізнавання та класифікації зображень доцільно використовувати згорткові нейронні мережі, які забезпечують високу точність та продуктивність у задачах комп'ютерного зору. Подальші дослідження та розробка програмного забезпечення будуть базуватися на використанні цих методів та технологій.

Таким чином тема кваліфікаційної роботи «Розробка програмного забезпечення для пошуку зображень з використанням нейронних мереж» є актуальною.

Метою роботи є автоматизація розпізнавання зображень.

У відповідності до поставленої мети в кваліфікаційній роботі поставлені наступні завдання дослідження:

1. Проаналізувати технічні засоби, що застосовуються для розробки нейронних мереж.

2. Розробити архітектуру веб-додатку з системою пошуку зображень.

3. Розробити програмне забезпечення для пошуку зображень.

4. Провести тестування розробленого веб- додатку.

Об'єктом дослідження є розробка програмного забезпечення для розпізнавання образів.

Предметом дослідження є технології розробки веб-додатку для розпізнавання образів з використанням нейронних мереж.

РОЗДІЛ 2. АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ КОМП'ЮТЕРНОГО ЗОРУ

2.1. Вибір засобів для реалізації системи

На сьогодні існує багато реалізацій мереж для розпізнавання даних, інтегрованих з різними мовами програмування. Найпоширенішими мовами у сфері машинного навчання є:

- Python

Найпопулярніша мова завдяки своїй простоті, зрозумілості, великому екосистемі бібліотек (TensorFlow, PyTorch, scikit-learn, Keras, NLTK) та активній спільноті.

- R

Часто використовується статистиками для аналізу даних завдяки своїм інструментам для статистичного аналізу та візуалізації (ggplot2).

- Java

Використовується у підприємницьких застосунках, завдяки своїй продуктивності та переносимості. Популярні бібліотеки: Deeplearning4j, Weka, MOA.

- C++

Відомий швидкістю та ефективністю, використовується для створення високопродуктивних бібліотек (OpenCV, TensorFlow API).

- JavaScript

Набирає популярність завдяки бібліотекам TensorFlow.js та Brain.js, що дозволяють виконувати машинне навчання в браузері.

- Julia

Високопродуктивна мова для технічного обчислення, яка швидко набирає популярність.

- Scala

Використовується завдяки сумісності з Java та функціональним можливостям (бібліотеки Breeze, Spark MLlib).

- MATLAB

Пропонує середовище для числового обчислення з пакетами для штучного інтелекту.

Найпопулярнішими бібліотеками та фреймворками є:

- TensorFlow

Потужний фреймворк від Google для нейронних мереж та глибокого навчання.

- PyTorch

Популярний фреймворк від Facebook для наукових досліджень та експериментів.

- Scikit-learn

Широкий спектр алгоритмів для класифікації, регресії, кластеризації.

- Keras

Високорівневий API для нейронних мереж, що спрощує створення та навчання моделей.

- MXNet

Гнучка та швидка бібліотека для глибокого навчання.

- Caffe

Фреймворк для глибокого навчання, зосереджений на обробці зображень.

Тренування моделей вимагає значних обчислювальних ресурсів, що важко здійснити на звичайному ПК. Для цього використовують GPU або TPU. Найпоширенішими платформами є:

- NVIDIA CUDA – платформа паралельних обчислень, що використовує відеокарти NVIDIA.
- Google Cloud Platform (GCP) – платформа від Google, що пропонує власні сервер для тренування моделей.
- Amazon Web Services (AWS) – платформа від Amazon, що пропонує власні сервер для тренування моделей через сервіс EC2.

- Azure – платформа від Microsoft, що пропонує власні сервер для тренування моделей.

Обираючи платформу для тренування моделей, важливо враховувати вартість, доступність відеокарт, підтримку фреймворків та зручність використання.

Для розробки було обрано мову Python з бібліотекою PyTorch для тренування моделей на платформі Google Colab, яка забезпечує безкоштовний доступ до GPU та TPU. Додаток буде написано мовою Python з використанням Flask для веб-інтерфейсу [15]. Для збереження даних використовуватиметься SQLite, що дозволяє швидко зберігати дані та легко перейти на іншу SQL базу даних для промислової реалізації [7].

Цей набір технологій забезпечує гнучкість, продуктивність та зручність розробки, що дозволить створити ефективний додаток для машинного навчання.

2.2. Вибір методу класифікації

Вибір типу реалізації нейронної мережі є важливим етапом розробки системи. Розглянемо деякі популярні моделі, які були створені раніше та відкриті для використання:

- Faster R-CNN: Це двоетапна модель для об'єктного виявлення, яка спочатку пропонує регіони, а потім передбачає обмежувальні рамки та ймовірності класів у цих регіонах.
- Mask R-CNN: Розширення Faster R-CNN, яке додає гілку для передбачення масок, дозволяючи сегментувати форми об'єктів разом із передбаченням обмежувальних рамок.
- RetinaNet: Одноетапна модель для об'єктного виявлення, відома своєю фокусною втратою та мережею Feature Pyramid Network (FPN), що вирішує дисбаланс класів та ефективно виявляє об'єкти на різних масштабах.

- YOLO (You Only Look Once): Моделі, такі як YOLOv3 та YOLOv4, є одноетапними детекторами, які передбачають обмежувальні рамки та ймовірності класів по всьому зображенню за один прохід, відомі своєю швидкістю.
- SSD (Single Shot MultiBox Detector): Інша одноетапна модель об'єктного виявлення, яка використовує згорткові фільтри для передбачення обмежувальних рамок та ймовірностей класів на кількох масштабах карт ознак.
- EfficientDet: Масштабована та ефективна родина моделей об'єктного виявлення, похідна від EfficientNet, з моделями від EfficientDet-D0 до EfficientDet-D7, що мають різну складність.
- CenterNet: Модель, яка передбачає центри та розміри об'єктів за допомогою теплових карт, спрощуючи задачу об'єктного виявлення до задачі оцінки ключових точок.
- DETR (DEtection TRansformer): Використовує трансформери для об'єктного виявлення, перетворюючи завдання виявлення на проблему передбачення набору без потреби у попередньо визначених ящиках-якорях.
- Cascade R-CNN: Покращує продуктивність, використовуючи каскадну архітектуру для поліпшення результатів об'єктного виявлення на кількох етапах.
- Sparse R-CNN: Вводить розрідженість в обчислення Faster R-CNN, що прискорює інференс без значної втрати точності [10].

Як видно з опису, більшість наведених моделей є згортковими нейронними мережами (CNN). Виняток становить модель DETR (DEtection TRansformer), яка використовує метод трансформерів. Оскільки більшість моделей використовують або повністю залежать від згортки, наша система також буде базуватися на згорткових нейронних мережах [8].

Для обраної системи буде використана одна з згорткових моделей залежно від специфіки завдання. Це забезпечить високу ефективність та точність у виявленні об'єктів.

2.3. Процес підготовки зображень

Кожне зображення має унікальні параметри, такі як розмір і кольорова палітра. Комп'ютер сприймає зображення як набір числових значень, розташованих у координатах матриці. На зображенні можуть бути присутні кілька об'єктів різного розміру, особливо якщо об'єкти знаходяться на різній відстані один від одного. Для обробки таких зображень застосовується механізм згортки (convolution), що дозволяє спочатку визначити об'єкти (визначити прямокутну область, де, ймовірно, знаходиться об'єкт), а потім класифікувати кожен об'єкт окремо як окреме зображення.

Перед тим як подати визначену область зображення до нейронної мережі для класифікації, її необхідно нормалізувати, тобто перетворити на матрицю певного розміру. Значення кожного елемента матриці вказує на яскравість або колір кожного пікселя. Використовуючи цю інформацію, алгоритми машинного навчання можуть аналізувати, класифікувати та взаємодіяти з графічними даними [9].

1) Для чорно–білих зображень, це може бути одна 2D матриця (x, y).

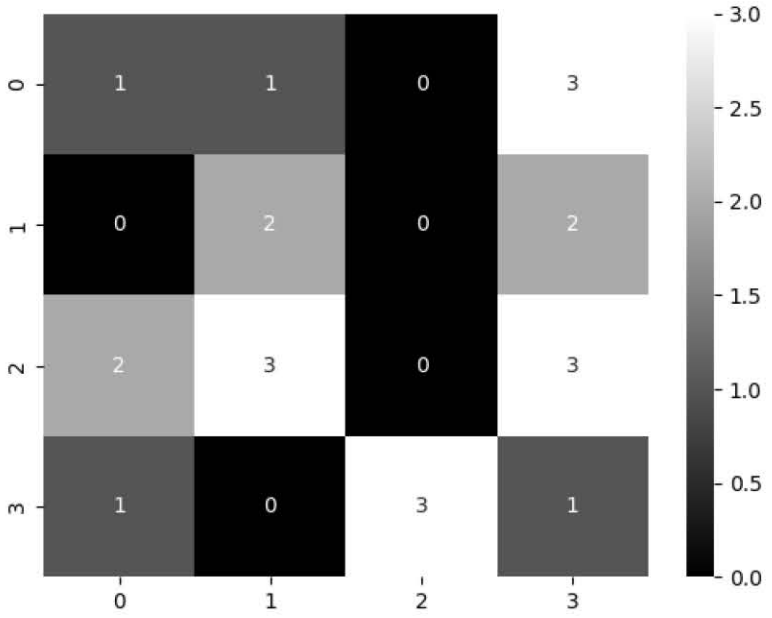

Рисунок 2.1 – Матриця відтінків (чорно–білий)

2) Для кольорових – три матриці для кожного каналу RGB: R(x, y), G(x, y), B(x, y).

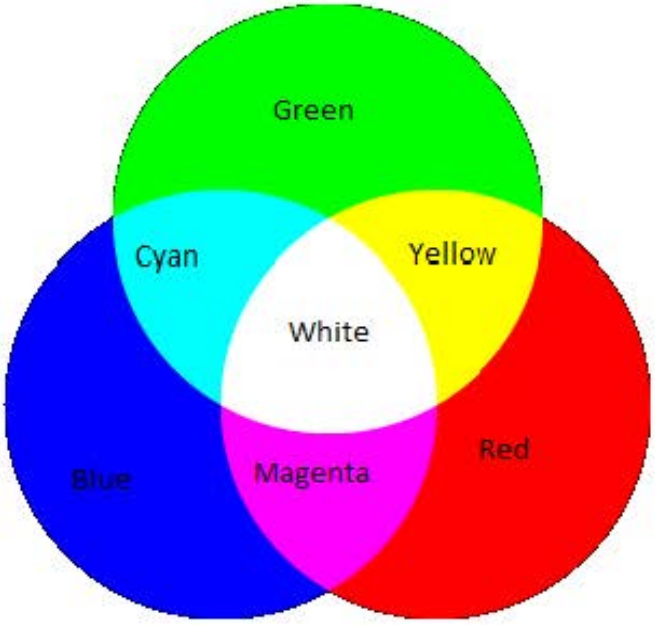

Рисунок 2.2 – Матриця створення нового кольору шляхом перетину 3 кольорів(червоного, зеленого, синього)

Наприклад, у бібліотеках PyTorch та TensorFlow така матриця називається тензором, що представляє собою абстракцію даних, які використовуються для класифікації. Варто також зазначити, що аналогічний підхід застосовується для обробки будь-яких даних. Тобто будь-яка інформація, яку можна подати у вигляді матриці певного розміру, придатна для аналізу за допомогою штучного інтелекту.

2.4 Створення, тренування та покращення моделі

Створення нейронної мережі складається з кількох етапів, починаючи з визначення її архітектури до навчання та оцінки результатів. Існує правило: якщо задачу можна вирішити за допомогою традиційного алгоритмічного програмування, використання ШІ не виправдане. Однак, якщо задача не може

бути вирішена жодним відомим алгоритмом, використання нейронної мережі може бути доцільним.

Якщо без штучного інтелекту не обійтись, першим кроком буде вибір типу моделі. Для обробки зображень зазвичай використовуються згорткові нейронні мережі (Convolutional Neural Networks, CNN). PyTorch надає багато типів мереж, доступних для створення, серед яких:

1) Класифікація зображень (Image Classification Models): AlexNet, VGG (VGG-11, VGG-13, VGG-16, VGG-19), ResNet (ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-152), DenseNet (DenseNet-121, DenseNet-169, DenseNet-201, DenseNet-161), Inception (Inception v3), SqueezeNet (SqueezeNet 1.0, SqueezeNet 1.1), MobileNetV2 EfficientNet (EfficientNet-b0 to EfficientNet-b7), ResNeXt, Wide Residual Networks (WideResNet),

2) Виявлення об’єктів (Object Detection Models): Faster R-CNN, Mask R-CNN, RetinaNet

3) Моделі симантичної сегментації: Semantic Segmentation Models: FCN (Fully Convolutional Networks), DeepLabV3, U-Net, PSPNet (Pyramid Scene Parsing Network)

4) Генеруючі моделі (Generative Models): Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs).

Класифікація зображень – це завдання, що полягає в категоризації всього зображення на заздалегідь визначені класи або категорії. Метою такого аналізу є призначення однієї мітки або класу для всього зображення на основі його вмісту. Результатом роботи моделі буде одна мітка, яка описує вміст усього зображення. Наприклад, визначення того, чи містить зображення кота, собаку, автомобіль або дерево.

Виявлення об'єктів – включає локалізацію та класифікацію кількох об'єктів у межах зображення. Метою аналізу є виявлення та локалізація окремих об'єктів, визначаючи їх присутність, місцезнаходження (обмежуюча рамка) та класову мітку. Отже, мережа відтворює кілька обмежуючих рамок навколо об'єктів разом із відповідними класовими мітками. Це застосовується

для виявлення та локалізації різних об'єктів, таких як автомобілі, люди або велосипеди на зображенні.

Враховуючи характеристики обох типів, можна зробити висновок, що виявлення об'єктів є більш гнучким у майбутньому, оскільки воно включає класифікацію кількох об'єктів на одному зображенні. Виявлення об'єктів включає три основні типи нейронних мереж:

1) RetinaNet – це одноетапний детектор об'єктів, призначений для виявлення об'єктів на зображеннях різних масштабів. Він використовує Focal Loss та Feature Pyramid Network (FPN). RetinaNet також ефективно проводить класифікацію об'єктів.

2) Faster R-CNN – це двоетапний детектор об'єктів, який спочатку пропонує регіони, де можуть знаходитися об'єкти, а потім класифікує ці регіони та локалізує об'єкти. Він не призначений для класифікації зображень як основне завдання.

3) Mask R-CNN – це розширення Faster R-CNN, яке додає можливість сегментації областей об'єктів (створення масок для виявлених об'єктів). Хоча він не спеціалізується на класифікації зображень, він дозволяє отримати маску для кожного виявленого об'єкта.

З наведеного опису можна зробити висновок, що RetinaNet є найбільш підходящою моделлю, оскільки, окрім виявлення об'єктів, вона також ефективно проводить їх класифікацію. Faster R-CNN та Mask R-CNN більше орієнтовані на локалізацію, а не на класифікацію [11].

Для оптимізації системи можна розглянути комбінацію двох нейронних мереж, наприклад, використання Mask R-CNN для локалізації, а потім іншої мережі для класифікації зображень. Проте, це значно ускладнить створення схеми бази даних для збереження зібраних даних для кожного зображення. Тому, для спрощення схеми бази даних, було прийнято рішення створити єдину модель, здатну ефективно виконувати локалізацію та класифікацію об'єктів, а саме RetinaNet.

Наступний крок – визначення кількості шарів та їх конфігурація. Оптимальна кількість шарів для архітектури RetinaNet залежить від складності набору даних, розміру та різноманіття об'єктів, обчислювальних ресурсів та компромісу між точністю та швидкістю виведення.

RetinaNet побудована на основі Feature Pyramid Network (FPN) у поєднанні з підмережами класифікації та регресії обмежувальних рамок. Вона зазвичай використовує піраміду ознак з кількома рівнями для виявлення об'єктів на різних масштабах. Вибір основної мережі (наприклад, ResNet-50, ResNet-101) також впливає на кількість шарів, оскільки ці мережі мають різну глибину. Глибокі мережі можуть захоплювати складніші ознаки, але вони можуть бути «важкими» в обчисленні та схильними до перенавчання, особливо якщо набір даних недостатньо великий. Менш глибокі мережі можуть працювати швидше, але не завжди здатні захопити складніші об'єкти [12].

Підготовка даних для навчання є найбільш трудомістким етапом, особливо коли йдеться про навчання мережі для класифікації зображень. Структура набору даних виглядає наступним чином: у папці «Images» знаходяться самі зображення, а у папці «Annotations» розташовані описи класів зображень з координатами відповідних об'єктів на зображеннях із папки "Images", як показано на рисунку 2.3 та рисунку 2.4.

```
:\Users\Oleksii_Bobko\eclipse-workspace\d
- Sample_Dataset/
  - Images/
    - image1.jpg
    - image2.jpg
    ...
  - Annotations/
    - image1.xml (or .json, .txt, etc.)
    - image2.xml
    ...
```

Рисунок 2.3 – Умовна структура розміщення файлів

```
<annotation>
    <filename>image1.jpg</filename>
    <object>
        <name>object_class</name>
        <bndbox>
            <xmin>10</xmin>
            <ymin>20</ymin>
            <xmax>100</xmax>
            <ymax>150</ymax>
        </bndbox>
    </object>
    <!-- More objects and annotations for image1 -->
</annotation>
```

Рисунок 2.4 – Анотація зображення набору даних

Підготовка даних для навчання нейронних мереж є найскладнішим етапом роботи, особливо у випадку класифікації зображень. Вона включає такі кроки, як масштабування, нормалізація та розподіл даних на тренувальні та тестувальні набори.

Існують три основні підходи до навчання нейронних мереж: навчання з учителем (supervised learning), навчання без учителя (unsupervised learning) та навчання з передачею знань (transfer learning). Кожен з них має свої особливості та застосування.

1) Навчання з учителем (Supervised Learning)

Цей метод передбачає навчання моделі на основі вхідних даних з відповідними мітками або маркуваннями. Модель намагається передбачити результати на основі навчального набору даних і коригується на основі різниці між передбаченими та фактичними результатами. Задачі, такі як класифікація та регресія, часто вирішуються за допомогою навчання з учителем.

2) Навчання без учителя (Unsupervised Learning)

У цьому підході модель намагається виявити приховані структури або залежності у вхідних даних без наявності міток або маркувань. Зазвичай це включає методи кластеризації, зменшення розмірності, асоціативне навчання та інші.

3) Навчання з передачею знань (Transfer Learning)

Це метод, при якому модель, навчена на одній задачі, використовується як стартова точка для іншої, пов'язаної задачі. Знання, отримані під час розв'язання однієї проблеми, передаються та застосовуються до іншої, пов'язаної проблеми. Це зазвичай здійснюється шляхом використання попередньо навчених моделей, що були навчені на великих наборах даних. Основна ідея полягає в тому, що модель, навчена на великому і загальному наборі даних, може слугувати сильною стартовою точкою для вирішення нових задач. Замість навчання нової моделі з нуля, використовується вже існуюча модель, яка доналаштовується на меншому специфічному наборі даних.

Після завершення навчання моделі наступним етапом є її тестування та оцінка:

1) Підготовка тестових даних

Включає завантаження тестового набору даних або створення власного. Ці дані зазвичай не використовуються під час навчання моделі, що дозволяє об'єктивно оцінити її продуктивність.

2) Передача тестових даних у модель – процес подачі тестових даних у модель для отримання результатів. Зазвичай це здійснюється частинами або на всьому наборі тестових даних.

3) Оцінка продуктивності

На цьому етапі обчислюються метрики продуктивності, такі як точність (accuracy), точність передбачень (precision) тощо.

4) Аналіз результатів

Оцінка отриманих результатів для визначення сильних і слабких сторін моделі. Це включає аналіз помилкових передбачень, визначення сценаріїв, на яких модель працює краще або гірше, та ідентифікацію областей для покращення.

Подальше вдосконалення моделі здійснюється на основі результатів тестування. Це може включати покращення точності, швидкодії, оптимізацію

параметрів або зміну архітектури. Якщо модель відповідає заданим критеріям, її впроваджують у систему для подальшого тестування на більш високому рівні, наприклад, на обраній групі користувачів у певній географічній локації.

2.5. Висновки до другого розділу

У другому розділі було проведено аналіз засобів реалізації комп'ютерного зору та вибору оптимальних методів класифікації зображень. Встановлено, що для ефективної реалізації системи класифікації зображень найкраще використовувати мову програмування Python з бібліотеками TensorFlow та PyTorch, що забезпечують широкі можливості для розробки нейронних мереж.

Розглянуто різні методи класифікації, включаючи згорткові нейронні мережі (CNN), які показали високу точність та продуктивність у задачах розпізнавання образів. Особливу увагу приділено методам підготовки та обробки зображень, включаючи нормалізацію, зміну розміру та видалення шуму, що є важливими етапами перед подачею даних до нейронної мережі.

Окрім цього, проаналізовано процес створення та навчання нейронних мереж, а також методи оптимізації, такі як навчання з учителем, навчання без учителя та навчання з передачею знань. Зокрема, підкреслено важливість використання попередньо навчених моделей для підвищення ефективності та точності класифікації зображень.

Таким чином, проведений аналіз засобів реалізації комп'ютерного зору та методів класифікації зображень дозволяє зробити висновок про доцільність використання згорткових нейронних мереж на базі бібліотек TensorFlow та PyTorch для розробки ефективних систем пошуку та класифікації зображень .

РОЗДІЛ 3. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПОШУКУ ЗОБРАЖЕНЬ

3.1. Розробка загальної архітектури веб-додатку

За останнє десятиліття мікросервісна архітектура стала надзвичайно популярною. Проте, на практиці виявляється, що її вартість часто перевищує можливості бізнесу. Як результат, багато компаній обирають спрощений варіант розподілених систем, зокрема тришарову архітектуру.

Тришарова архітектура є одним з найбільш поширених структурних стилів для розробки програмного забезпечення. Вона передбачає розподіл системи на три основні компоненти або шари:

1. Presentation Layer – це шар, що відповідає за взаємодію з користувачем та відображення даних. Він включає всі елементи користувацького інтерфейсу, такі як форми, сторінки, кнопки та інші елементи управління. Основна мета інтерфейсного шару – забезпечити зручний та інтуїтивно зрозумілий спосіб взаємодії користувача із системою.

2. Business Logic Layer – шар, реалізує всі бізнес-правила та обробку даних. Він виступає посередником між інтерфейсним шаром та шаром даних, забезпечуючи виконання логіки програми, перевірку даних, розрахунки та інші бізнес-процеси. Шар бізнес-логіки дозволяє ізолювати бізнес-правила від інших компонентів системи, що полегшує їхнє тестування та підтримку.

3. Data Layer – шар даних, що відповідає за доступ до сховища даних, таких як бази даних або зовнішні сервіси. Він забезпечує зберігання, отримання та оновлення даних необхідних для роботи системи. Шар даних включає в себе механізми для взаємодії з базами даних, такі як ORM (Object-Relational Mapping) інструменти, запити до баз даних та інші засоби управління даними.

Мікросервісна архітектура є сучасним підходом до розробки програмного забезпечення, який передбачає розподіл системи на набір невеликих, незалежних компонентів, відомих як мікросервіси. Кожен мікросервіс відповідає за конкретну функцію або послугу і може розгортатися та масштабуватися окремо.

Основні характеристики мікросервісної архітектури:

1. Незалежність компонентів

Мікросервіси розробляються, тестуються, розгортаються та масштабуються незалежно один від одного. Це дозволяє командам працювати паралельно над різними частинами системи без взаємного втручання.

2. Чітке розмежування функцій

Кожен мікросервіс відповідає за конкретну бізнес-логіку або функціональність, що забезпечує чітке розмежування відповідальностей між компонентами.

3. Взаємодія через API

Мікросервіси взаємодіють між собою через добре визначені API, що дозволяє використовувати різні технології та платформи для кожного мікросервісу. Це забезпечує високу гнучкість у виборі інструментів і технологій.

4. Автономність розгортання та масштабування

Мікросервіси можна розгортати та масштабувати незалежно один від одного, що забезпечує кращу адаптивність до змін у навантаженні та зручність у підтримці та оновленні системи.

Вибір між тришаровою та мікросервісною архітектурою залежить від конкретних потреб та вимог проекту. Тришарова:

- Для монолітних проектів або систем з меншим масштабом.
- Коли бізнес-логіка досить проста і може бути впроваджена в одному шарі.
- Для проектів з обмеженими ресурсами та бюджетом.

Мікросервісна архітектура:

- Потрібна більша масштабованість та надійність.
- Коли бізнес-логіка розділена на багато різних функцій чи модулів, які можна розгортати та масштабувати окремо.
- Потрібно застосовувати різні технології для кожного сервісу

Вибір архітектурного стилю повинен відповідати конкретним цілям і вимогам вашого проекту, а також масштабу і обмеженням, з якими ви стикаєтесь. Наприклад, цей проект є інформаційною системою для аналізу зображень, яка класифікує їх і додає до бази даних. Проект має наступну структуру:

Рисунок 3.1 – Структура програмного продукту

Згідно даної структури, що зображена на рисунку 3.1, програма має 2 компоненти: це сама програма, яка має інструменти для аналізу зображення, та база даних, що зберігає зображення, після аналізу.

3.2. Опис обраних технологій

Для розробки даного програмного забезпечення використовується мова програмування Python. Ця мова пропонує широкий вибір бібліотек, які дозволяють використовувати штучний інтелект, створювати клієнт-серверні додатки та працювати з базами даних для зберігання різноманітної інформації [13].

Для зберігання даних обрана база даних SQLite. SQLite є базою даних з відкритим вихідним кодом, написаною мовою C, та доступною для запитів за

допомогою стандартного SQL. Вона популярна серед розробників завдяки таким перевагам:

- Легкість та простота
- Низькі системні вимоги
- Широке застосування

Для створення інтерфейсу, який дозволяє взаємодіяти з системою, використовується стандартний HTML. Це дозволяє створити відносно простий веб-додаток з мінімалістичним інтерфейсом [14].

3.3. Розробка графічної та функціональної частини веб-додатку.

Далі розглянемо основні етапи створення, структуру та його програмну реалізацію. Як зазначалося раніше, для розробки додатку використовується мова програмування Python.

Для розробки програмного забезпечення використовувався вбудований редактор для Python – IDLE Shell, а також наступні бібліотеки:

:

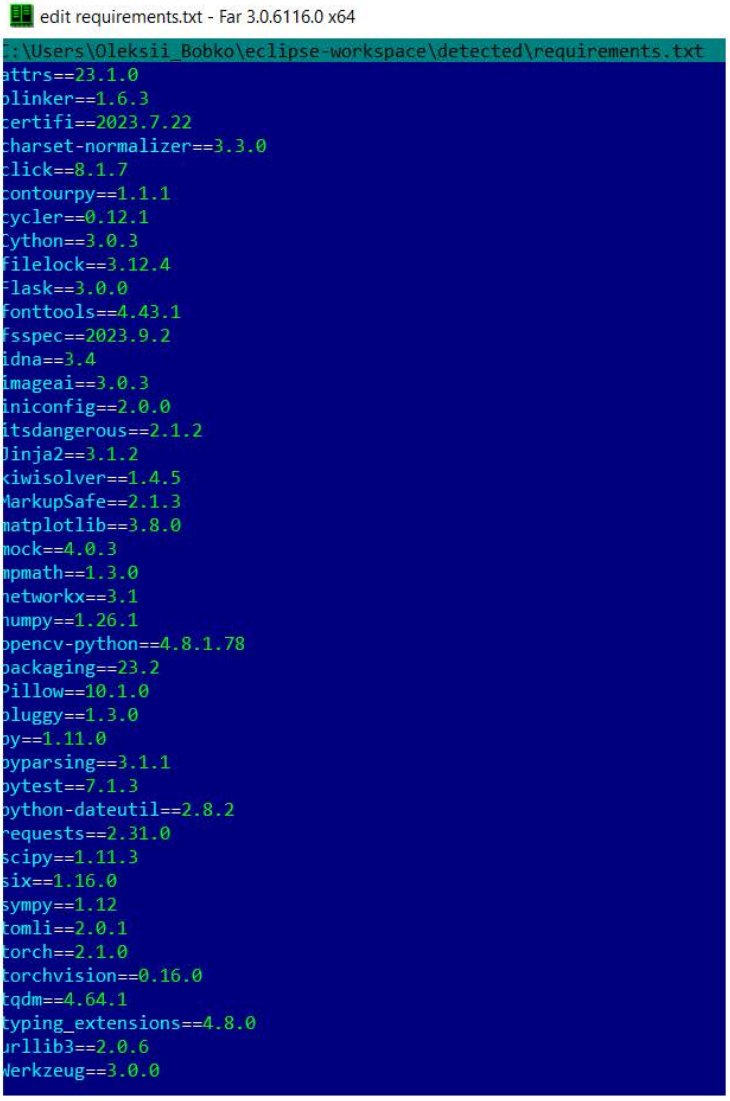

Рисунок 3.2 – Бібліотеки Python, використані в розробці

Python дозволяє встановлювати бібліотеки безпосередньо з одного файлу, як показано на рисунку 3.2, ввівши в термінал Windows команду pip install -r requirements.txt.

Наступними етапами розробки є виконання таких завдань:

1. Розробка функціоналу для взаємодії з інтерфейсом.
2. Забезпечення можливості аналізувати та класифікувати зображення.
3. Створення функцій для взаємодії з базою даних.

Для реалізації зазначеного функціоналу структура програми має наступний вигляд, як зображено на рисунку 3.3:

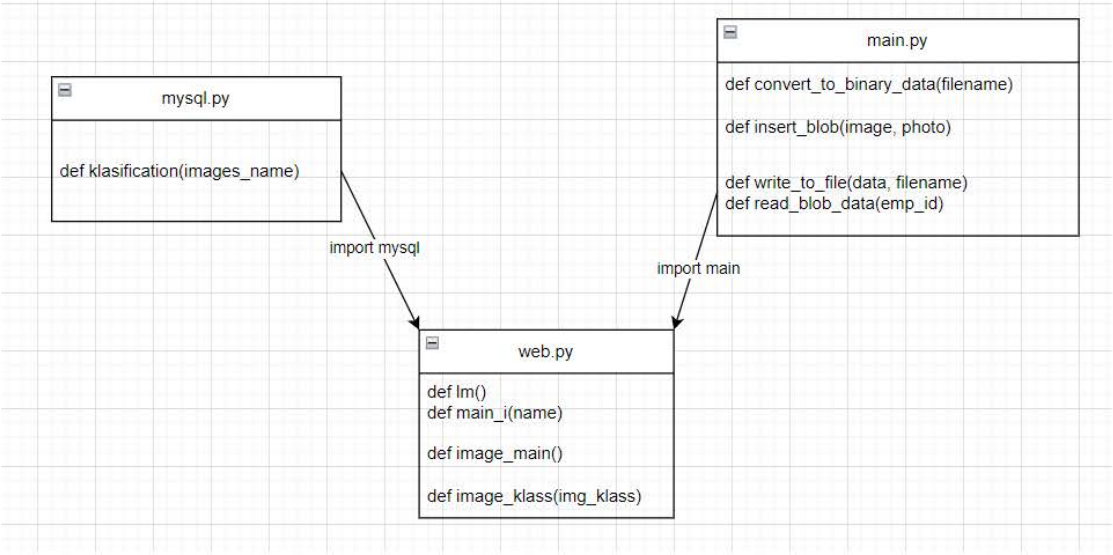

Рисунок 3.3 – Структура програмної реалізації додатку

Розглянемо окремо кожен компонент структури (рис. 3.3). Почнемо з головного компонента web.py. Цей компонент відповідає за створення веб-додатку за допомогою бібліотеки Flask.

```
from flask import Flask, render_template, url_for, redirect, request
import main
import storage


app = Flask(__name__)


@app.route('/local')
def lm():
    return render_template('index.html')

@app.route('/<name>', methods=['POST'])
def main_i(image):
    print(image)
    f = request.files['image']
    main.klasification(image)
    return render_template('index.html')
```

Рисунок 3.4 – Завантаження веб-додатку та його сторінки.

Web.py використовує функції компонентів: storage, main – для взаємодії з базою даних та передачі фотографій для подальшого аналізу, як показано на рисунку 3.5.

```python
@app.route('/main', methods=['POST'])
def image_main():
    if request.method == 'POST':
        f = request.files['image']
        f.save("static/images/" + f.filename)
        klass = main.classification(f.filename)
        return render_template('index.html', list=klass, image="/images/" + f.filename + "new.jpeg")

@app.route('/main/<string:img_klass>')
def image_klass(img_klass):
    ans = storage.read_blob_data(img_klass)
    print(ans)
    new_ans = list()
    for i in ans:
        ans = i.split('/')
        new_ans.append(ans[-1])
    return render_template('answer.html', list=new_ans)
```

Рисунок 5 – Функції image_main, image_klass

Компонент storage. Відповідає за взаємодію з базою даних image_kod та таблицею images, яка використовується для зберігання зображень. Для цього використовується бібліотека sqlite3 (рис. 3.6).

```python
def convert_to_binary_data(filename):
    with open(filename, 'rb') as file:
        blob_data = file.read()
    return blob_data

def insert_blob(image, photo):
    try:
        sqlite_connection = sqlite3.connect('image_kod')
        cursor = sqlite_connection.cursor()
        print("Connected к SQLite")

        sqlite_insert_blob_query = """INSERT INTO Images
                                      (image, kod) VALUES (?, ?)"""

        emp_photo = convert_to_binary_data(photo)
        data_tuple = (image, emp_photo)
        cursor.execute(sqlite_insert_blob_query, data_tuple)
        sqlite_connection.commit()
        cursor.close()
```

Рисунок 3.6 – Функція для підключення до бази даних.

Цей компонент також дозволяє отримати всі фотографії певної категорії за допомогою відповідної функції (рис. 3.7).

```python
def read_blob_data(img_id):
    im = list()
    try:
        sqlite_connection = sqlite3.connect('image_kod')
        cursor = sqlite_connection.cursor()
        print("Connected to SQLite")

        sql_fetch_blob_query = """SELECT * from Images where image = ?"""
        cursor.execute(sql_fetch_blob_query, (img_id,))
        record = cursor.fetchall()
        for row in record:
            print("Id = ", row[0], "Name = ", row[1])
            name = row[1]
            photo = row[2]

            print("save on disk \n")
            photo_path = "static/images/" + name + str(random.randint(1,10000)) + ".jpg"
            write_to_file(photo, photo_path)
            im.append(photo_path)
        cursor.close()
```

Рисунок 3.7 – Функція read_blob_data()

Останній компонент додатку – це main. Цей компонент використовує бібліотеку ImageAI, яка базується на PyTorch для виявлення об’єктів на зображеннях. Він завантажує модель застосовує її для ідентифікації об’єктів на зображенні, ім’я якого передається як аргумент. Для кожного виявленого об'єкта код перевіряє, чи належить він до певних категорій (наприклад, ‘cat’, ‘car’, ‘dog’, ‘cow’ або ‘horse’). Якщо об'єкт відповідає одній з цих категорій, зображення додається до бази даних SQLite з відповідною міткою. Деталі цього процесу представлені на рисунку 3.8.

```python
from imageai.Detection import ObjectDetection
import os
from storage import insertBLOB

def classification(images_name):
    images_name1 = images_name

    detector = ObjectDetection()
    detector.setModelTypeAsRetinaNet()
    detector.setModelPath("coco_resnet_50_map_0_335_state_dict.pt")
    detector.loadModel()
    detections = detector.detectObjectsFromImage("static/images/" + images_name1, "static/images/" + images_name1 + "new.jpeg")

    klass = list()

    for eachObject in detections:
        insertBLOB(eachObject["name"], "static\\images\\" + images_name1)
        klass.append(eachObject["name"] + " : " + str(eachObject["percentage_probability"]) + ", " + str(eachObject['box_points']));
        print(eachObject["name"] , " : ", eachObject["percentage_probability"], " , ", eachObject["box_points"] )

    return klass
```

Рисунок 3.8 – Компонент main.

Для створення інтерфейсу використовуємо html (рис. 3.9):

```
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href = "{{ url_for('static', filename='css/main.css')}}">
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    {% block body %}{% endblock %}
</body>
</html>
```

Рисунок 3.9 – Розмітка головної сторінки.

Інтерфейс має доволі простий функціонал – дві кнопки для додавання та аналізу зображення (рис. 3.10).

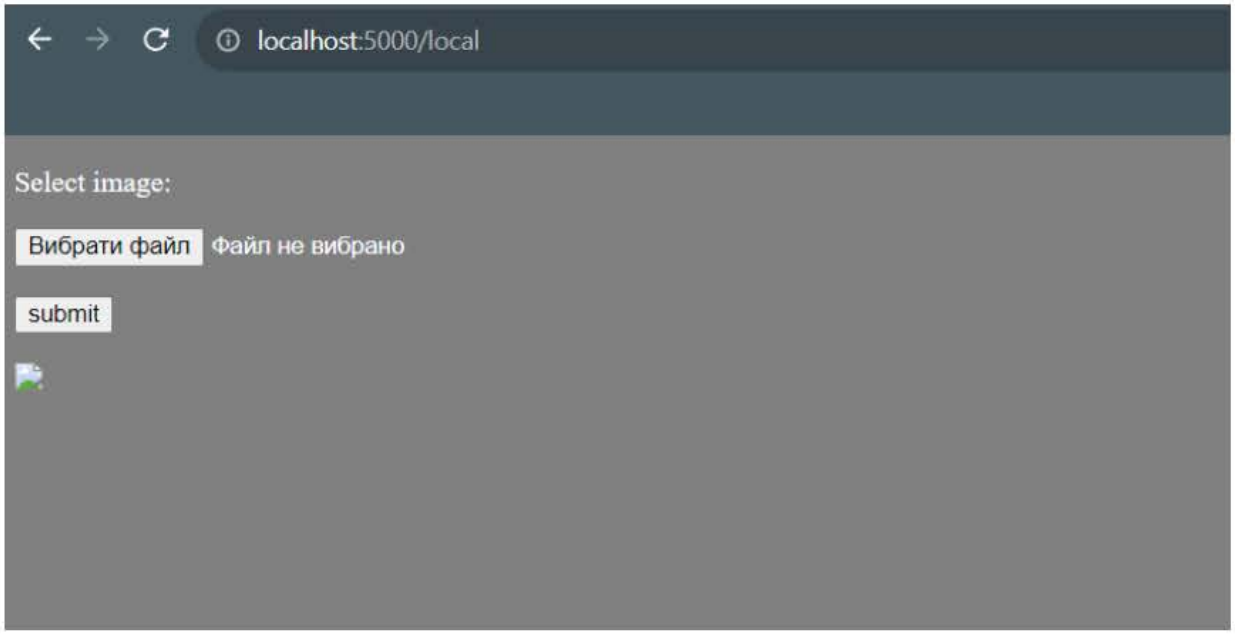

Рисунок 3.10 – Інтерфейс додатку.

Останнім етапом розробки додатку є створення бази даних та збереження інформації до неї. Для цього було створено базу даних image_kod. Таблиця images використовується для зберігання зображень. Щоб зберегти

зображення, його потрібно конвертувати у двійковий формат (BLOB) перед додаванням до таблиці (рис. 3.11).

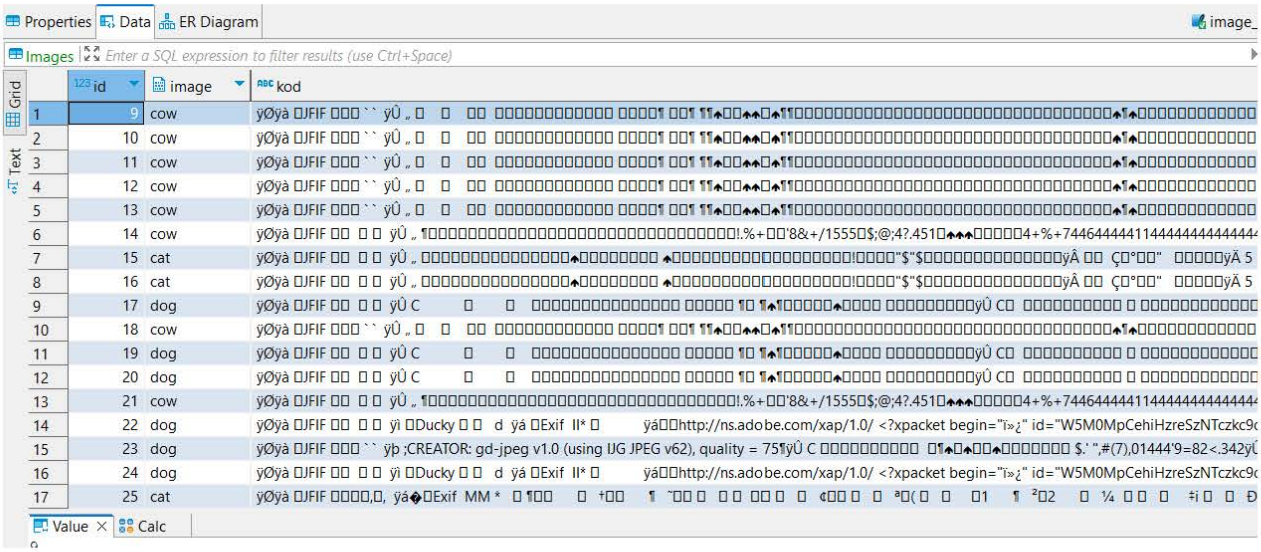

Рисунок 3.11 – Таблиця Images бази даних image_kod

Отже в результаті розробки, отримано додаток, що дозволяє аналізувати зображення, визначати його категорію та зберігати в базу даних.

3.4. Тестування розробленого програмного забезпечення.

Після завершення розробки програми необхідно провести тестування, перевіривши функціональність додатку на коректність його роботи та аналізу графічної інформації. Для початку запустимо головний скрипт web.py.

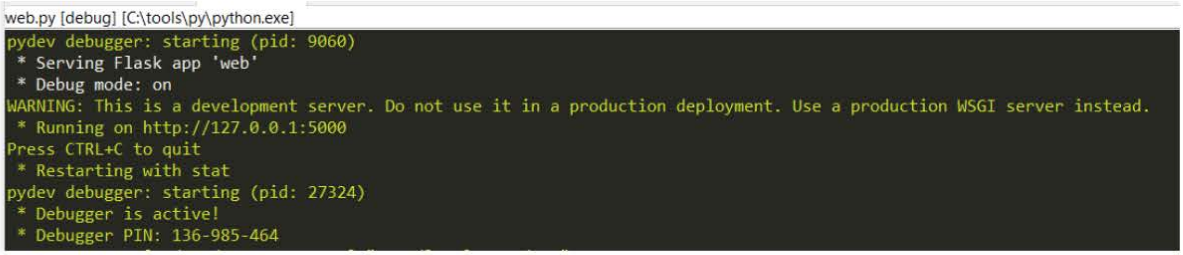

Рисунок 3.12 – Запуск web.py

Після запуску цього скрипта в консолі буде відображено адресу, за якою доступний веб-додаток: http://127.0.0.1:5000.

Тепер введіть цю адресу в адресний рядок браузера: http://127.0.0.1:5000/local. Це проілюстровано на рисунку 3.13.

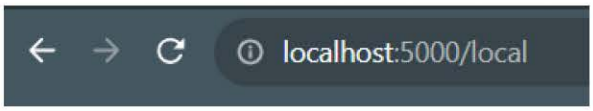

Рисунок 13 – Введення адреси

В результаті завантажилася наступна сторінка:

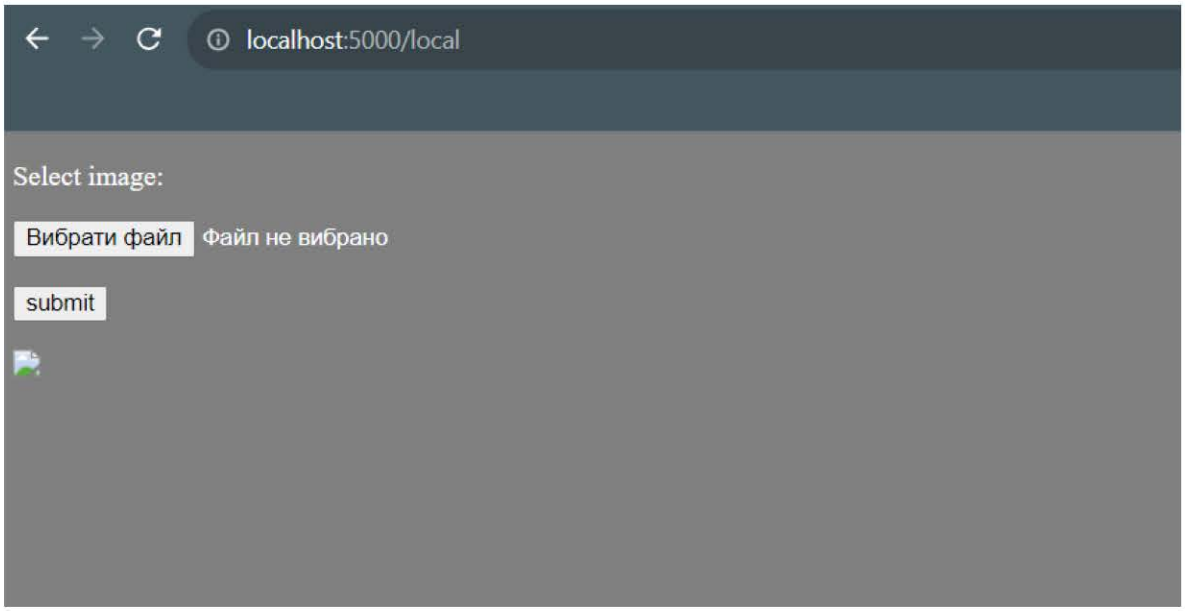

Рисунок 3.14 – Інтерфейс додатку

Далі необхідно вибрати фотографію для перевірки. Зверніть увагу, що модель може розпізнавати такі категорії: cat, car, dog, cow. Для тестування пропонується використати наступне зображення:

Рисунок 3.15 – Приклад зображення

Вибираємо дане зображення на сторінці (рис. 3.16):

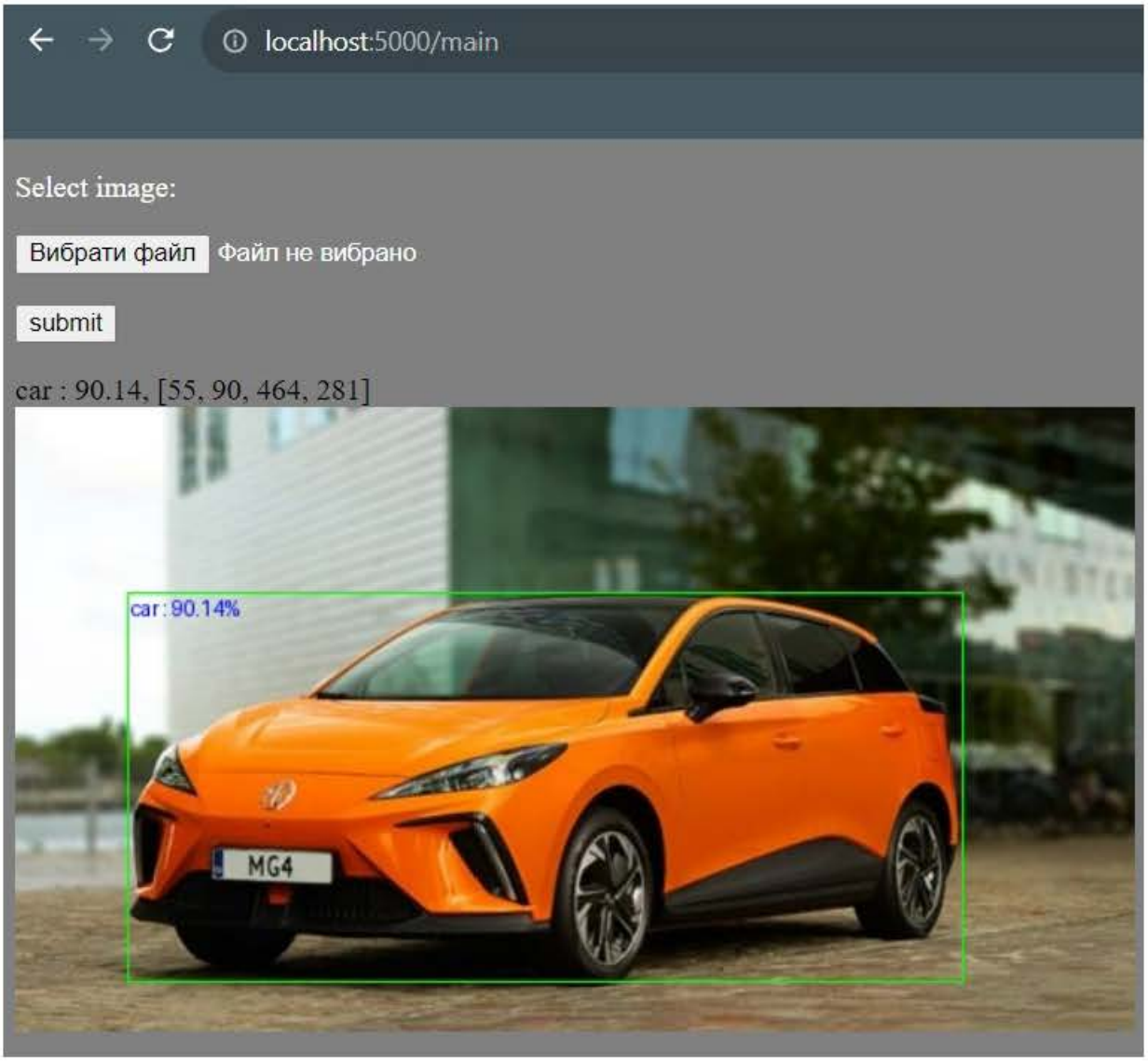

Рисунок 3.16 – Збережене зображення

Після натискання кнопки "Submit", програма проаналізує зображення та визначить його категорію. Це можна перевірити, переглянувши базу даних:

Рисунок 3.17 – Додавання фото в базу даних.

Далі введемо в рядок наступну адресу: http://localhost:5000/main/car.

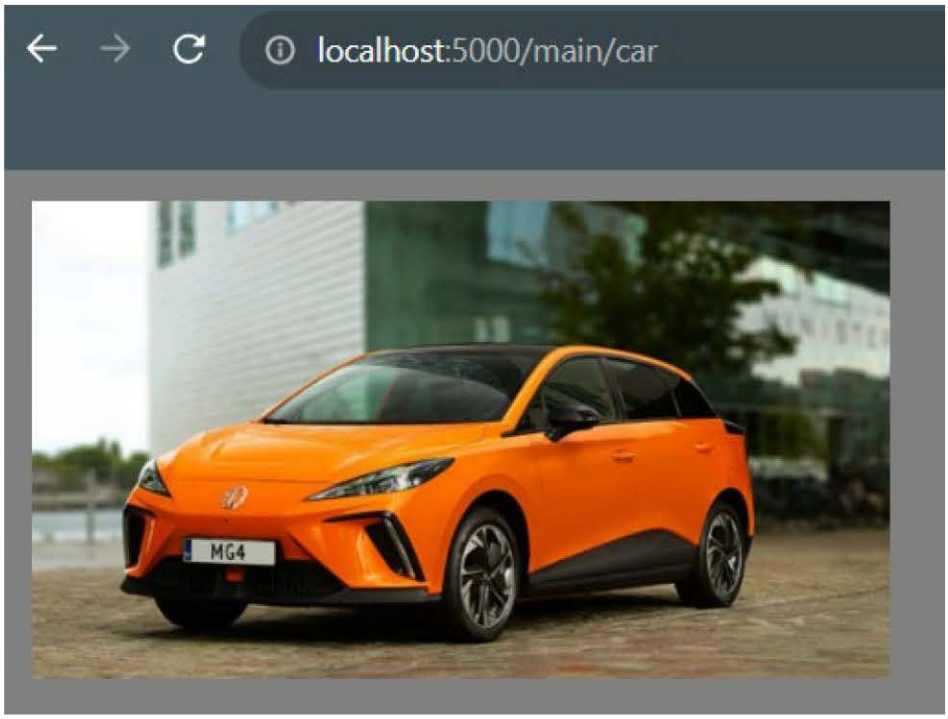

Рисунок 3.18 – Виведення всіх зображень категорії «машина»

Також можна спробувати інші запити. Наприклад http://localhost:5000/main/cow

Рисунок 3.19 – Зображення які система розпізнала як «cow»

На рисунку 3.19 видно, що за запитом «cow» програма зробила помилковий висновок. Це показовий приклад того що система не завжди забезпечує 100% точність результатів. У таких випадках важливо враховувати можливість помилки. Отже, внаслідок отримання зображення з відповідної категорії, можна впевнитися, що програма працює вірно та без помилок настільки наскільки добре натренована нейронна мережа.

3.5. Висновки до третього розділу

У третьому розділі було розроблено систему пошуку зображень, яка включає розробку загальної архітектури програмного забезпечення, опис обраних технологій, розробку графічної та функціональної частини додатку, а також тестування програмного засобу.

При виборі архітектурного стилю необхідно враховувати цілі та вимоги вашого проекту, а також його розмір та обмеження. Описане програмне забезпечення – система, яка аналізує зображення, класифікує їх за категоріями та додає до бази даних.

Було обрано тришарову архітектуру для розробки програмного забезпечення, що забезпечує розділення системи на інтерфейсний шар, бізнес-логіку та шар даних. Це спрощує розробку, підтримку та масштабування системи. Також було розглянуто мікросервісну архітектуру, яка дозволяє забезпечити більшу гнучкість та швидкість розробки, але потребує більших ресурсів.

Для розробки програмного забезпечення використовувалася мова програмування Python з бібліотеками PyTorch та ImageAI, що дозволило ефективно реалізувати аналіз та класифікацію зображень. База даних SQLite обрана для зберігання зображень та результатів їхнього аналізу, що забезпечує легкість та продуктивність у роботі з даними. SQLite – це база даних з відкритим вихідним кодом, написана на мові C і доступна для запитів за допомогою звичайного SQL. Має широке розповсюдження серед розробників

Було розроблено інтерфейс користувача за допомогою HTML та Flask, що забезпечило зручну взаємодію з системою. Графічний та функціональний компоненти додатку були інтегровані для забезпечення повного циклу обробки зображень, від завантаження до класифікації та збереження результатів у базі даних.

Тестування програмного забезпечення показало, що система працює коректно, визначаючи категорії зображень та зберігаючи їх у базі даних.

Однак, результати також вказали на можливість помилкових класифікацій. Тобто система не завжди забезпечує 100% точність результатів. У таких випадках важливо враховувати можливість помилки. Внаслідок отримання зображення з відповідної категорії, можна впевнитися, що програма працює вірно та без помилок настільки, наскільки добре натренована нейронна мережа. Тестування розробленого програмного забезпечення підкреслює важливість подальшого вдосконалення моделей та алгоритмів класифікації для досягнення вищої точності.

Отже в результаті розробки, отримано додаток, що дозволяє аналізувати зображення, визначати його категорію та зберігати в базу даних.

Таким чином, розроблена система пошуку зображень демонструє високу ефективність та продуктивність, а також забезпечує гнучкість у використанні та подальшому вдосконаленні.

ВИСНОВОК

Кваліфікаційна робота присвячена розробці системи пошуку зображень з використанням нейронних мереж, що є актуальним завданням у контексті сучасного розвитку технологій машинного навчання та комп'ютерного зору. Метою роботи було створення ефективного інструменту для класифікації та пошуку зображень, що забезпечить користувачам швидкий та точний аналіз візуальної інформації.

У першому розділі роботи було проведено детальний аналіз предметної області, зокрема дослідження сучасних методів та підходів до розпізнавання зображень. Особливу увагу було приділено аналізу існуючих методів, таких як SIFT, SURF, ORB та AKAZE, а також їх застосуванню у задачах комп'ютерного зору. Встановлено, що згорткові нейронні мережі (CNN) є найбільш ефективними для задач класифікації зображень завдяки їхній здатності автоматично визначати та використовувати важливі ознаки у вхідних даних.

У другому розділі роботи було обрано оптимальні програмні засоби для реалізації системи. Серед сучасних бібліотек та фреймворків для глибокого навчання були розглянуті TensorFlow, PyTorch та інші. Для реалізації проекту було обрано мову програмування Python та бібліотеку PyTorch, яка забезпечила високу гнучкість та продуктивність розробки. Було визначено архітектуру системи, яка включає підготовку даних, навчання моделей та їх подальше використання для класифікації зображень.

У третьому розділі було детально описано процес розробки системи пошуку зображень. Реалізовано основні компоненти, такі як архітектура програмного забезпечення, алгоритми обробки зображень та інтеграція з базою даних SQLite. Було розроблено та протестовано ключові елементи системи, включаючи завантаження зображень, їх аналіз та збереження результатів у базі даних. Проведене тестування показало, що система ефективно обробляє зображення та забезпечує високу точність класифікації.

Розроблена система демонструє високу ефективність та продуктивність, а також відповідає сучасним вимогам у галузі комп'ютерного зору та машинного навчання. Використання згорткових нейронних мереж дозволило досягти високої точності у класифікації зображень, що підвищує цінність розробленої системи для практичного застосування. Результати роботи можуть бути корисними для інших дослідників та розробників у сфері штучного інтелекту, які прагнуть створювати інноваційні продукти для обробки візуальної інформації.

Розробка цієї системи підкреслює важливість використання сучасних методів машинного навчання для вирішення складних задач комп'ютерного зору, забезпечуючи високу продуктивність і точність аналізу даних. Подальші дослідження та вдосконалення моделей і алгоритмів можуть ще більше підвищити ефективність і точність системи, розширюючи її застосування у різних галузях, таких як медицина, безпека, автоматизація виробничих процесів та інші.

В цілому, результати кваліфікаційної роботи підтверджують, що використання згорткових нейронних мереж є перспективним напрямком у розпізнаванні зображень і може значно покращити якість та швидкість обробки візуальної інформації, що є важливим для розвитку сучасних інформаційних технологій.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Van den Berg C. A., van den Boomgaard R., Worring M., Koelma D., Smeulders A. Horus: Integration of image processing and database paradigms//Proceedings of the First Int. Workshop of Image Databases and MultiMedia Search.–1996.–p. 226–234.

2. Smith J. R., Chang S. – F. Tools and Techniques for Color Image Retrieval// In Symposium on Electronic Imaging: Science and Technology//Storage & Retrieval for Image and Video Databases. – 1996. – № 4, vol. 2670. – p. 426–437.

3. Stricker M., Swain M. The capacity of color histogram indexing// Computer Vision and Pattern Recognition. Proceedings CVPR'94. IEEE Computer Society Conference. – 1994. – p. 704-708.

4. Viisage Products Portfolio. http://www.viisage.com/ww /en/pub/ viisage_products_new.htm

5. Комп`ютерний зір. [Електронний ресурс]/Режим доступу:https://www.amazon.com/Computer-Vision-Modern-Approach2nd/dp/013608592

6. Siamese Neural Network for One-Shot learning. [Електронний ресурс]/Режим доступу:https://medium.com/@subham.tiwari186/siamese-neural-network-for-one-shotimage-recognition-paper-analysis-44cf7f0c66cb

7. Contrastive and Triplet losses. [Електронний ресурс]/Режим доступу: https://gombru.github.io/2019/04/03/ranking_loss

8. Grewenig S. Cyclic Schemes for PDEBased Image Analysis / S. Grewenig, J. Weickert, C. Schroers, A. Bruhn // International Journal of Computer Vision, 2013.

9. ORB: an efficient alternative to SIFT or SURF, Computer Vision / [E. Rublee, V. Rabaud, K. Konolige, G. Bradski]; (ICCV), IEEE International Conference. – 2011. – C. 2564–2571.

10. Yang X. LDB: An ultra-fast feature for scalable augmented reality / X. Yang, K. T. Cheng // In IEEE and ACM Intl. Sym. on Mixed and Augmented Reality (ISMAR). – 2012. – С. 49–57.

11. Shapiro L. Computer vision // L. Shapiro, D. Stockmann - M .: Binom. Laboratory of knowledge, 2006. — 752 с

12. Python [Електронний ресурс] – Режим доступу: https://docs.python.org/3/

13. SQLite [Електронний ресурс] – Режим доступу: https://www.sqlite.org/doclist.html

14. Flask [Електронний ресурс] – Режим доступу: https://flask.palletsprojects.com/en/3.0.x/