МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

Другого (магістерського) рівня вищої освіти

на тему: “ Інформаційно-аналітична система розпізнавання розташування місцевості на зображенні з використанням нейронних мереж ”

Виконав: студент 6 курсу групи Іт-62
Спеціальності 126 – „Інформаційні системи та технології”
(шифр і назва)

Бугай Андрій Ігорович
(Прізвище та ініціали)

Керівник: к.т.н., в.о. доц. Падюка Р.І.
(Прізвище та ініціали)

ДУБЛЯНИ-2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ

ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Другий (магістерський) рівень вищої освіти
Спеціальність 126 "Інформаційні системи та технології"

"ЗАТВЕРДЖУЮ"
Завідувач кафедри_____
д.т.н., проф. А.М. Тригуба
"____"_____2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Бугаю Андрію Ігоровичу

1. Тема роботи: «Інформаційно-аналітична система розпізнавання розташування місцевості на зображенні з використанням нейронних мереж»

Керівник роботи Падюка Роман Іванович, к.т.н., в.о. доцента.
Затверджені наказом по університету від 12 вересня 2024 року № 616/к-с
2. Строк подання студентом роботи 10.01.2025 р.
3. Початкові дані до роботи: 1. Методики навчання штучних нейронних мереж; 2) Типові архітектури навчання мережі; 3) Набори даних для навчання; 4) Методика визначення показників економічної ефективності розробки ПЗ; 5) ДСТУ прикладної статистики.

4. Зміст розрахунково-пояснювальної записки:
1. Аналіз стану питання в практиці та теорії
2. Обґрунтування, вибір та реалізація інструментарію вирішення задачі
3. Результати вирішення задачі
4. Охорона праці та безпека в надзвичайних ситуаціях
5 Визначення ефективності розробки інформаційно-аналітичної системи
Висновки та пропозиції.
Бібліографічний список.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень):Тема, автор, керівник магістерської роботи; Мета,завдання, об'єкт, предмет дослідження; Типова топологія штучних нейронних мереж; Вибір математичних методів навчання нейронних мереж; Вибір архітектури для навчання мережі; Вибір наборів даних для вирішення задачі; Архітектура системи розпізнавання розташування місцевості на зображенні; Метод кодування географічних координат місцевості

Опис реалізації інформаційно-аналітичної системи; Визначення ефективності розробки інформаційно-аналітичної системи;

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 5	*Падюка Р.І., в.о. доцента кафедри інформаційних технологій*		
4	*Городецький І.М., доцент кафедри інженерії та безпеки виробництва*		

7. Дата видачі завдання 16 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Приміт-ка
1	*Написання першого розділу та означення головних завдань роботи*	16.09.-30.09.24	
2	*Виконання другого розділу та формування головних показників для розрахунків*	01.10.-16.10.24	
3.	*Виконання третього розділу та формування початкових даних*	16.10.-30.10.24	
4.	*Виконання розділу охорони праці та безпеки в надзвичайних ситуаціях*	01.11.-07.11.24	
6.	*Виконання п'ятого розділу та узагальнення отриманих результатів магістерської роботи*	08.11.-16.11.24	
7.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів графічної частини*	17.11.-25.11.24	
8.	*Завершення роботи в цілому*	26.11.-10.01.25	

Студент _____ Бугай А.І.
(підпис)

Керівник роботи _____ Падюка Р.І.
(підпис)

УДК 658.51:631.1

Інформаційно-аналітична система розпізнавання розташування місцевості на зображенні з використанням нейронних мереж – Бугай А.І. Магістерська робота. Кафедра ІТ. – Дубляни, Львівський НУП, 2025.

64 с. текст. част., 28 рис., 1 табл., 10 арк. графічної частини, 20 літ. джерел.

Проведено аналіз загальних відомостей з проблематики розпізнавання об’єктів на місцевості, означено принципи використання нейронних мереж в процесі розпізнавання розташування місцевості на зображенні та здійснено загальний опис існуючих математичних методів навчання нейронних мереж.

Обґрунтовано основні принципи моделювання архітектури для навчання нейронної мережі та використання методів автоматичного диференціювання та максимальної правдоподібності для визначення параметрів мережі та їх оцінки. Здійснено вибір методу навчання мережі та вибір наборів даних.

Розроблено архітектуру інформаційної системи розпізнавання розташування місцевості на зображенні та обґрунтовано метод кодування географічних координат місцевості. Здійснено опис реалізації розробленої інформаційно-аналітичної системи.

Визначено економічну ефективність розробки інформаційно-аналітичної системи, здійснено розрахунок витрат, що пов'язані з розробкою програмного забезпечення та аналіз потенційного ринку збуту розробленої інформаційно-аналітичної системи.

Запропоновано заходи з охорони праці та безпеки в надзвичайних ситуаціях.

ЗМІСТ

ПЕРЕДМОВА

Визначення географічного положення об'єкта на зображенні без будь-якої попередньої інформації є серйозною проблемою через величезну кількість потенційних зображень, зроблених у всьому світі. Такі фактори, як час доби, розташування об'єкта або налаштування камери, ще більше ускладнюють цю спробу. Крім того, багато зображень, як правило, неоднозначні, пропонуючи мінімальні візуальні підказки щодо правильного розташування об'єкта. Отже, більшість сучасних методів спрощують процес геолокалізації об'єктів, обмежуючи його обмеженим діапазоном зображень, таких як добре відомі пам'ятки, міста або природні середовища, такі як ліси та гори. Лише кілька систем вирішують ширше завдання без залежності від конкретних зображень чи попередніх припущень. Ці методи особливо виграють від сучасних досягнень глибокого навчання та збільшення доступності загальнодоступних наборів даних і зображень із геотегами на таких платформах, як Instagram, Facebook, Imgur і Pinterest.

Проблеми, пов'язані зі складною природою проблем, у поєднанні з нерівномірним розподілом місць, де знімаються фотографії по всьому світу, роблять методи на основі згорткової нейронної мережі (CNN) ефективними для розгляду геолокації об'єктів на зображеннях як проблеми класифікації. Це передбачає сегментування Землі на географічні комірки, які містять узгоджену мінімальну кількість зображень. Тим не менш, навіть найдосконаліші CNN намагаються зберегти візуальні характеристики всіх регіонів Землі, одночасно справляючись із завданням визначення типу сцени зображення. Поділ землі на географічні комірки створює дилему. Більш детальний розподіл підвищує точність на рівні місцевого міста (з помилками менше 1 км), тоді як ширший розподіл осередків підвищує продуктивність у глобальному масштабі (близько 1000 км). І навпаки, категорії природних об'єктів, такі як гори та галявини або внутрішні приміщення, як правило, більш точно характеризуються особливостями, пов'язаними з флорою та фауною або природою внутрішнього

простору, відповідно. Тому можна стверджувати, що процес локалізації об’єкта на зображенні може значно покращитися, якщо його доповнити інформацією щодо опису сцени, оскільки цей підхід може помітно зменшити дисперсію вибірки даних і усунути сторонній шум. Ця методологія застосовна для вирішення проблем, пов’язаних із криміналістичною експертизою соціальних медіа, включаючи маніпуляції вмістом і метаданими. Завдяки швидкому розширенню платформ соціальних медіа медіа-файли, які публікуються онлайн, можуть швидко охопити мільйони користувачів. Одночасно люди можуть без особливих зусиль розповсюджувати підроблені зображення з шкідливими намірами, такими як розпалювання паніки чи вплив на громадську думку.

Отже, метою цієї роботи є розробка системи, яка підвищує точність визначення географічних координат об’єктів на зображенні. Для досягнення поставленої мети необхідно вирішити наступні основні завдання:

- Вивчити прийоми визначення географічних координат об'єктів, зафіксованих на фотографіях;
- Створити програмне забезпечення, яке оцінює ймовірність розташування об'єктів на зображеннях;
- Обґрунтувати взаємозв'язок похідного рішення та обчислити ключові показники навчання.

Ефективність навчання нейронної мережі вимірюється кількома ключовими показниками, включаючи оцінку F1, точність і повноту тестового набору даних.

РОЗДІЛ 1
АНАЛІЗ ТЕМИ ТА ПОСТАНОВКА ЗАДАЧІ

1.1. Аналіз загальних відомостей з проблематики розпізнавання об'єктів на місцевості

Наскільки складно визначити GPS-координати об'єктів, зображених на зображенні? Хоча завдання може здатися надзвичайно складним, зображення часто містять достатньо даних, таких як орієнтири, візерунки хмар, типи трави, дорожні знаки або архітектурні деталі, які можуть вказати місце розташування фотографії. Раніше цю проблему вирішували за допомогою методів пошуку зображень або шляхом аналізу існуючих метаданих.

Завдання розпізнавання - це класифікаційне завдання, що сегментує поверхню Землі на географічні комірки за допомогою унікальної кривої заповнення простору. Модель навчається з використанням тисяч географічних регіонів різного масштабу. У цій роботі представлено кілька методів глибокого навчання, які використовують складні методології географічних даних і багатозадачне навчання. Також рекомендується враховувати контекст сцени — незалежно від того, чи було знято зображення на вулиці, у приміщенні, у природних умовах чи в міських районах тощо.

Отже, додаткова інформація з різноманітною просторовою роздільною здатністю та більш адаптованими функціями для різних середовищ інтегрується в процес навчання згорткової нейронної мережі. Значення, отримані для цільових показників, демонструють ефективність запропонованого готового методу, який використовує допоміжну мережу для об'єднання двох наборів даних.

Класифікація міток типу сцени відбувається в межах набору даних, який містить лише координати GPS. Ця модель не залежить від методів пошуку [1][2][3], які потребують значних обчислювальних ресурсів, натомість вибирає імовірнісний підхід. Через широке поширення численних ресурсів в Інтернеті

одним із найпростіших способів знайти об’єкт є пошук порівнянних зображень за допомогою Google або TinEye. Виявлення назви місця, де розташований об’єкт, може спростити завдання ідентифікації цього об’єкта. Однак такий підхід не завжди ефективний. Як наслідок, вивчення альтернативних методів може бути доволі корисним. Наприклад, вивчення елементів, які не видно на фотографії, може дати цінну інформацію щодо самого зображення.

Метадані, включаючи дані EXIF, містять різну інформацію, наприклад:

- мітка часу створення зображення;
- інформація про місцезнаходження;
- параметри зображення та модель камери (витримка, діафрагма тощо);
- відомості про особу, якій належить зображення.

Ця інформація може бути цінною для вивчення двох ключових елементів: часу та місця зйомки фотографії, а також ступеня та характеру будь-яких змін, внесених до зображення. Зокрема, дані геолокації, якщо вони присутні в метаданих, можуть надати точні відомості про те, де було зроблено фотографію. Однак на доступність цієї геолокаційної інформації впливає кілька факторів. По-перше, це залежить від пристрою, який використовується для захоплення зображення; так деякі камери або мобільні пристрої можуть не мати GPS-датчика, здатного записувати координати.

Крім того, користувачі мобільних пристроїв можуть вимкнути геолокацію з міркувань конфіденційності або для збереження заряду акумулятора. Нарешті, на наявність таких даних також впливає платформа, на якій публікується фотографія, оскільки соціальні мережі, такі як Facebook, Twitter або Instagram, часто видаляють метадані із зображень після їх завантаження в ці служби.

1.2. Використання нейронних мереж в процесі розпізнавання розташування місцевості на зображенні

Машинне навчання — це підгалузь штучного інтелекту в інформатиці,

яка зазвичай використовує статистичні методи, щоб дозволити комп'ютерам «навчатися» з даних (тобто поступово підвищувати продуктивність у певному завданні) без необхідності явного програмування. Розвинулася ця галузь з досліджень розпізнавання образів і теорії обчислень.

Вивчаючи штучний інтелект, машинне навчання досліджує вивчення та побудову алгоритмів, які можуть навчатися та передбачати на основі доступних даних. Такі алгоритми долають необхідність слідувати строгим інструкціям статичного програмування та стають керованими даними, створюючи моделі з вибіркових прогнозів або рішень. Машинне навчання використовується в багатьох обчислювальних завданнях, де складно або неможливо розробити явні алгоритми з хорошою продуктивністю, наприклад, фільтрація електронної пошти, виявлення мережевих хакерів або зловмисників, які намагаються викрасти дані людини, оптичне розпізнавання символів (OCR). , рейтингове навчання та комп'ютерний зір [6][7].

Згорткова нейронна мережа — це тип глибокої штучної нейронної мережі прямого поширення, яка використовується в аналізі зображень. Штучні нейронні мережі використовують багатошаровий персептрон, розроблений для мінімальної обробки даних та містять характеристики інваріантності на основі їх спільної вагової архітектури та паралельної передачі. Порівняно з іншими алгоритмами класифікації зображень, SNM використовує відносно невелику попередню обробку. Це означає, що мережа навчається за допомогою фільтрів, тоді як фільтри в традиційних алгоритмах потрібно розробляти вручну. Ця незалежність від попередніх знань і людських зусиль при побудові елементів є великою перевагою. SNM складається з вхідного та вихідного шарів і кількох прихованих шарів. Прихований рівень SNM зазвичай складається з згорткового рівня, рівня агрегації, повністю зв'язаного рівня та рівня нормалізації. [8]

Згорткові шари застосовують операції згортки до вхідних даних і передають результати на наступний рівень. Згортка моделює реакцію одного нейрона на візуальний стимул. Хоча повністю зв'язані нейронні мережі

прямого поширення можна використовувати для вивчення функцій і класифікації даних, застосовувати цю архітектуру до зображень непрактично. [9]

Навіть у неглибоких шарах потрібна велика кількість нейронів, оскільки вхідний розмір, пов'язаний із зображенням, дуже великий, де кожен піксель є відповідною змінною. Наприклад, повністю пов'язаний шар для (маленького) зображення розміром 100 × 100 має 10 000 параметрів. Операція згортки вирішує цю проблему, оскільки вона зменшує кількість вільних параметрів, дозволяючи мережі бути глибшою з меншою кількістю параметрів.

Наприклад, незалежно від розміру зображення, кожне має однакову загальну вагу і вимагає лише 25 вільних параметрів. Таким чином, це вирішує проблему зникнення або розширення градієнтів під час навчання традиційних багатошарових нейронних мереж із використанням зворотного поширення. Малюнок. 1.1 описує такий рівень нейронної мережі.

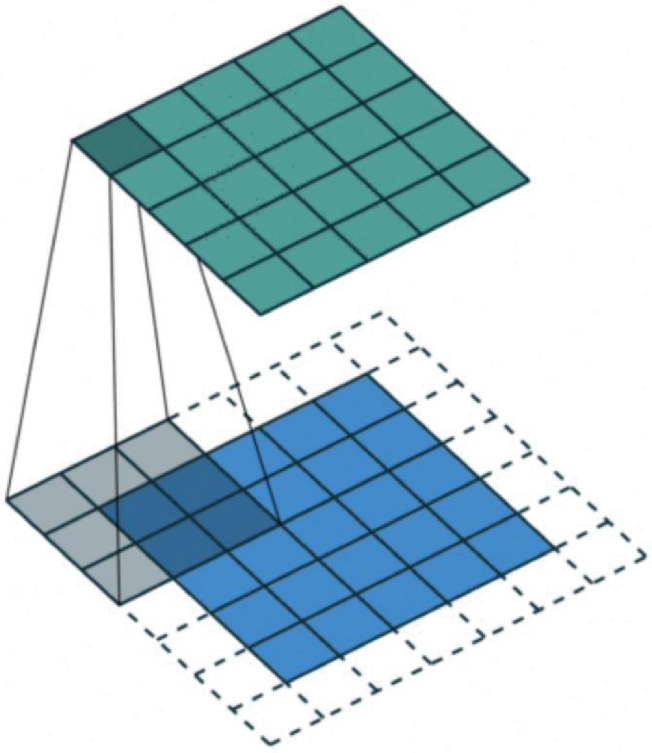

Рис. 1.1. Візуалізація операції згортки

Згорткові мережі можуть включати локальні або глобальні рівні агрегації, які поєднують виходи нейронних кластерів від одного рівня до іншого. Наприклад, агрегація за максимумом використовує максимальне значення кожного кластера нейронів у попередньому шарі.

Іншим прикладом є середня агрегація, яка використовує середнє значення кожного кластера нейронів у попередньому шарі. [10].

Повністю зв'язаний рівень з'єднує кожен нейрон одного шару з кожним нейроном наступного шару. В принципі, це те саме, що традиційна багатошарова нейронна мережа персептронів. SNM використовує загальні ваги в згорткових шарах, що означає, що той самий фільтр (значення ваги) використовується для кожного рецептивного поля рівня; це зменшує необхідний обсяг пам'яті та покращує продуктивність.

1.3. Загальний опис існуючих математичних методів навчання нейронних мереж.

Нами використано метод градієнтного спуску для навчання нейронної мережі, яка розроблялася в рамках цієї кваліфікаційної роботи.

Градієнтний спуск — це ітеративний алгоритм оптимізації першого порядку, який виконує дії, пропорційні градієнту (або приблизному градієнту) функції в поточній точці, щоб знайти локальний мінімум функції. І навпаки, якщо робляться кроки, пропорційні значенню самого градієнта, відбуватиметься наближення до локального максимуму функції.

Стандартна реалізація цього методу не є оптимальною для навчання глибоких мереж, тому в цій роботі використовується модифікована версія методу градієнтного спуску під назвою Адам.

Адам — це алгоритм оптимізації, який може замінити класичний процес стохастичного градієнтного спуску та ітеративно оновлювати вагові коефіцієнти мережі на основі навчальних даних. [11].

Переваги використання Адама в задачах неопуклої оптимізації

полягають у наступному:

- простий у реалізації;
- малі вимоги до пам'яті;
- інтуїтивно зрозумілі гіперпараметри;
- інваріантність до змін градієнтів напрямку.

Цей метод розраховує індивідуальні швидкості адаптивного навчання для різних параметрів на основі оцінок першого та другого моментів градієнта. На рис. 1.2 показаний псевдокод алгоритму. [12].

Algorithm: Generalized Adam

S0. Initialize $m_0 = 0$ and x_1

For $t = 1, \dots, T$, do

S1. $m_t = \beta_{1,t} m_{t-1} + (1 - \beta_{1,t}) g_t$

S2. $\hat{v}_t = h_t(g_1, g_2, \dots, g_t)$

S3. $x_{t+1} = x_t - \frac{\alpha_t m_t}{\sqrt{\hat{v}_t}}$

End

Рис. 1.2. Псевдокод алгоритму

Адам є популярним алгоритмом глибокого навчання, оскільки він дуже швидко досягає хороших результатів.

Емпіричні результати показують, що Адам добре працює на практиці. Якість алгоритму було емпірично продемонстровано в оригінальній статті, показуючи, що збіжність функції витрат відповідає очікуванням теоретичного аналізу.

Адам використовувався для навчання логістичної регресії на наборах даних розпізнавання символів MNIST і IMDB, багатошарових нейронних

мереж на наборі даних MNIST і згорткових нейронних мереж на наборі даних розпізнавання зображень CIFAR-10.

1.4. Постановка задачі кваліфікаційної роботи

Метою даної роботи є створення системи, що використовує багатозадачні нейронні мережі для визначення географічних координат об’єктів, зображених на фотографіях.

Для досягнення поставленої мети необхідно вивчити метод розрахунку розподілу температурного поля в ізотропних тілах обертання, чисельні методи в інформатиці, методи моделювання фізичних процесів теплопровідності, методи регіональної дискретизації для розрахунку розподілу температури.

Після завершення дослідження зібрану інформацію та дані необхідно проаналізувати і на основі цього розробити програмне забезпечення для визначення географічного розташування об’єктів за зображеннями.

РОЗДІЛ 2
ОБҐРУНТУВАННЯ, ВИБІР ТА РЕАЛІЗАЦІЯ ІНСТРУМЕНТАРІЮ ВИРІШЕННЯ ЗАДАЧІ

2.1. Основні принципи моделювання архітектури ResNet для навчання мережі

Багато сучасних мереж починаються з вибору архітектури для конкретного завдання. Для вирішення проблем класифікації архітектура ResNet є розумним вибором, оскільки її впровадження не дороге та дозволяє досягти високих значень метрики класифікації та вирішити проблему затухання градієнта.

ResNet - це абревіатура Residual Network. Коли глибша нейронна мережа починає процес навчання, виникає проблема: зі збільшенням глибини мережі точність спочатку зростає, а потім швидко падає. Зниження точності навчання показує, що не всі глибокі мережі легко оптимізувати. Щоб подолати цю проблему, Microsoft представила структуру глибокого «залишкового» навчання. Замість того, щоб кожні кілька послідовних шарів відповідали безпосередньо бажаній основній ідеї, вони явно дозволяють цим шарам відповідати «решті». Цю ідею можна реалізувати за допомогою нейронних мереж зі з’єднаннями швидкого доступу. на рисунку. 2.1 наведена структурна схема цього з’єднання.

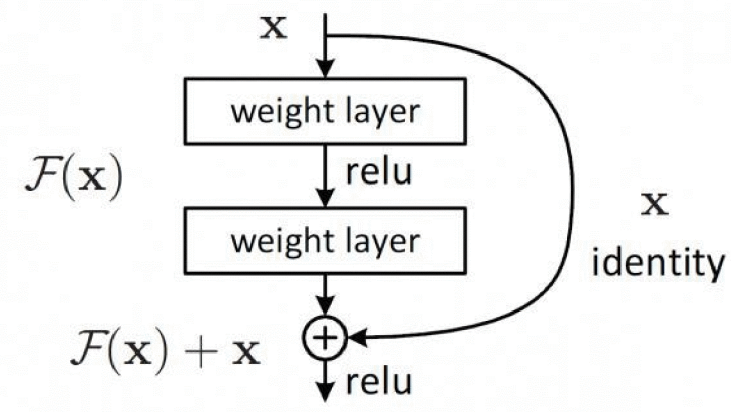

Рис. 2.1. З'єднання для швидкого доступу

Як видно з рисунку, проміжні виходи ранніх рівнів використовують такі з'єднання для передачі інформації на наступні рівні. Для визначення географічних координат об’єктів у цій роботі була обрана конкретна версія архітектури під назвою ResNet50. [14].

На рисунку. 2.2 Кожен рівень архітектури ResNet50 детально описаний і включає такі основні елементи:

Згортка з розміром ядра 7 * 7 і 64 диференціальних ядра, всі з кроком 2, утворюючи 1 шар;

Наступним шаром є MaxPooling із кроком 2;

Наступне ядро згортки становить 1 * 1,64, наступне за ним ядро згортки - 3 * 3,64, а останнє ядро - 1 * 1256. Ці три шари повторюються 3 рази, таким чином отримуючи 9 ідентичних шарів на цьому кроці;

Далі використовується ядро 1 * 1.128, потім ядро 3 * 3.128 і, нарешті, ядро 1 * 1.512, цей крок повторюється 4 рази, створюючи таким чином ще 12 шарів на цьому етапі;

Після цього є одне ядро 1 * 1.256 і ще два ядра 3 * 3.256 і 1 * 1.1024, повторюється 6 разів, це створює ще 18 шарів;

Потім ще одне ядро 1 * 1,512 і ще два 3 * 3,512 і 1 * 1,2048, повторювані 3 рази, створюючи загалом 9 шарів;

Після цього, виконуючи операцію AveragePooling, ми додаємо останній рівень мережі, який є повністю пов’язаним шаром із 1000 нейронами, і, нарешті, функцію softmax, тож у нас є ще 1 рівень.

Загалом мережа містить 50 рівнів, за винятком функцій активації як окремих рівнів.

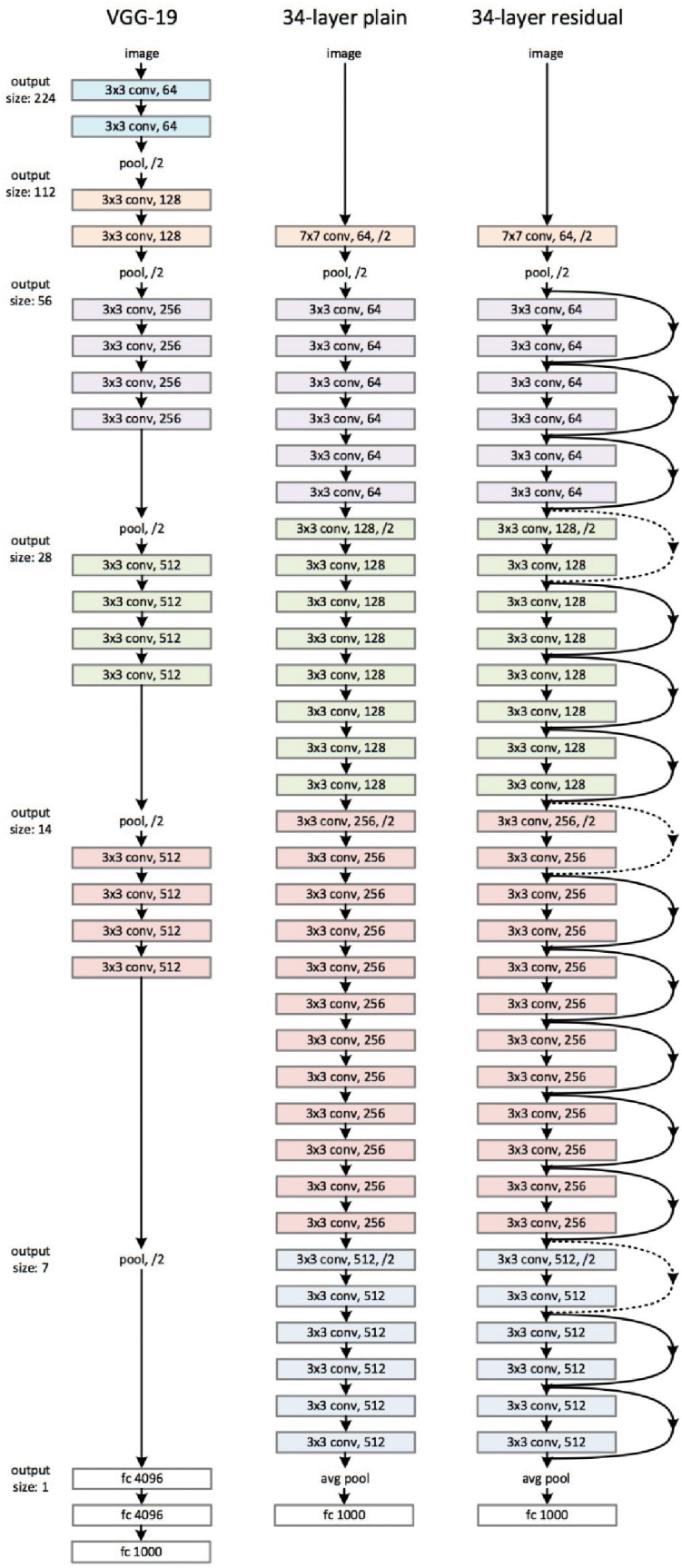

Рис. 2.2. Візуалізація архітектури нейронної мережі ResNet50

2.2. Використання методу автоматичного диференціювання для визначення параметрів мережі

Щоб визначити параметри мережі, які використовуються для визначення географічних координат, необхідно обчислити часткові похідні, що є дуже ресурсною операцією, тому нами було використано метод, який називається автоматичним диференціюванням. [15].

Цей метод використовує той факт, що будь-яка функція комп'ютерної програми все одно використовуватиме арифметичні операції (+, -, *, /) і основні функції стандартної бібліотеки (exp, log, sin, cos тощо). Застосовуючи правило ланцюга, похідну будь-якого порядку можна обчислити із заданою точністю за кількість операцій, пропорційну кількості операцій самої функції обчислення. Водночас автоматичне диференціювання не є ні символьним, ні числовим.

Символьне диференціювання не завжди є ефективним, оскільки деякі функції важко виразити в одному виразі, тоді як числове диференціювання призводить до появи помилок округлення та дискретизації. Обидва методи незручні для обчислення похідних високого порядку, а похибка і складність значно зростають. Крім того, обидва методи працюють повільно під час обчислення часткових похідних функцій із кількома параметрами. [16]. Автоматичне диференціація вирішує всі ці проблеми, але вводить додаткові залежності програмного забезпечення. Для цього на рисунку 2 представлено спеціальний «обчислювальний» граф. 2.3.

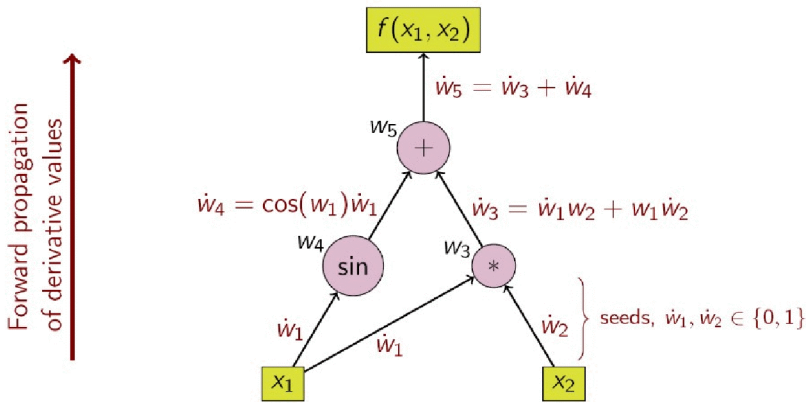

Рис. 2.3. Граф обчислень

Граф обчислень — це граф, який містить компоненти рівняння. Він є формою спрямованого графа, що представляє математичні вирази. Дуже поширеним прикладом є підрахунок суфіксів, інфіксів або префіксів. Кожен вузол на графіку може містити операцію, змінну або саме рівняння. Ці графи з'являються в більшості обчислень, які виконуються в комп'ютерах. Структура такого графа показана на рисунку нижче.

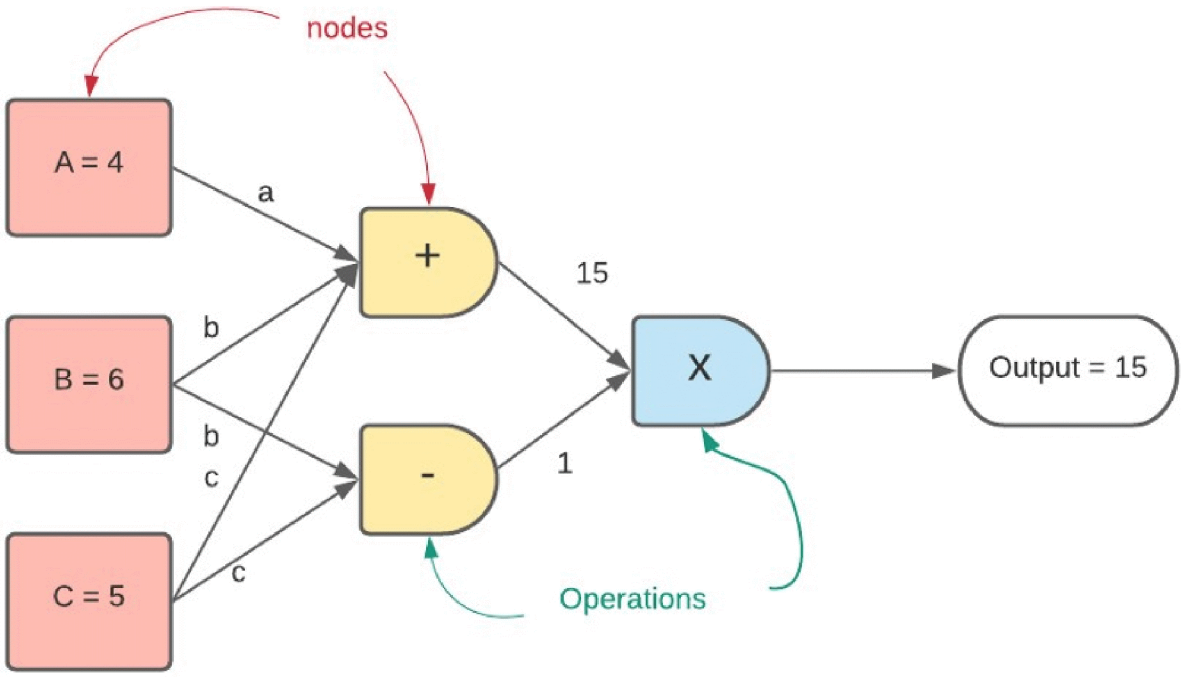

Рис. 2.4. Граф обчислень для простих операцій

Усі нейронні мережі є обчислювальними графами. Не тільки вони, навіть такі алгоритми, як лінійна регресія, можуть бути представлені графіками. Основна відмінність традиційних графів від нейронних мереж полягає в реалізації. Нейронні мережі часто навчені імітувати обчислювальні графіки, але не можуть працювати з графоподібними даними. Для роботи їм потрібні структуровані дані. [17].

2.3. Використання методу максимальної правдоподібності для оцінки значень параметрів моделі

Цей метод заснований на припущенні, що вся інформація про статистичну вибірку міститься в цій функції. Метод максимальної правдоподібності проаналізовано, рекомендовано та значно узагальнено Р. Фішером між 1912 і 1922 роками (хоча раніше його використовували Гаусс, Лаплас та ін.). Оцінка максимальної правдоподібності є популярним статистичним методом побудови статистичних моделей на основі даних і надання оцінок параметрів моделі. [18].

Метод максимальної правдоподібності відповідає багатьом відомим методам оцінки в галузі статистики. Наприклад, припустимо, що метою є аналіз зростання жителів України. Припустімо, що у вас є дані про зріст певної кількості людей, а не всього населення. Крім того, передбачається, що зріст є нормально розподіленою змінною з невідомою дисперсією та середнім. Вибіркове середнє значення та дисперсія зросту максимально наближені до середнього значення та дисперсії всієї сукупності. [19].

Для фіксованого набору даних і базової ймовірнісної моделі метод максимальної правдоподібності використовується для пошуку значень параметрів моделі, які роблять дані «ближчими» до справжніх значень. Оцінка максимальної правдоподібності забезпечує унікальний і простий метод визначення рішення у випадку нормального розподілу.

2.4. Вибір методу навчання нейронної мережі

Квантування — це метод виконання обчислень і збереження тензорів у типах даних із меншою точністю, ніж точність із плаваючою комою. Квантувані моделі виконують деякі або всі операції над тензорами за допомогою цілих чисел замість використання типів даних з плаваючою комою. Це дозволяє створювати більш компактні представлення моделі та використовувати високопродуктивні операції векторизації на багатьох апаратних платформах. Цей підхід особливо корисний під час навчання моделі запуску, оскільки він може заощадити багато обчислювальних витрат без

шкоди для надто високої точності обчислень. Наразі такі основні фреймворки глибокого навчання, як TensorFlow і PyTorch, підтримують квантування. [20].

На малюнку 2.5 наведені значення параметрів, записані під час навчання мережі.

Original Values	Sign Bit	2^{3}	2^{2}	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	8 Bit Binary Rep	Quantized Value
														Power of 2 Bins	
0.03125															
-0.250															
0.250															
0.500															
1.000															
2.100															
-2.125															
8.250															
16.250															

Рис. 2.5. Перший крок квантизації

Важливо знайти оптимальне двійкове представлення для кожного записаного значення параметра. Крайній лівий біт двійкового слова відомий як старший біт (MSB), який має найбільший вплив на числове значення. На рис. 2.6 MSB для кожного значення позначено жовтим кольором.

Original Values	Sign Bit	2^{3}	2^{2}	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}	8 Bit Binary Rep	Quantized Value
										MSB				Power of 2 Bins	
0.03125										1	0	0	0		
-0.250	✓						1	0	0	0	0	0	0		
0.250							1	0	0	0	0	0	0		
0.500						1	0	0	0	0	0	0	0		
1.000					1	0	0	0	0	0	0	0	0		
2.100				1	0	0	0	0	1	1	0	0	1		
-2.125	✓			1	0	0	0	1	0	0	0	0	0		
8.250			1	0	0	0	0	1	0	0	0	0	0		
16.250		1	0	0	0	0	0	1	0	0	0	0	0		

Рис. 2.6. Другий крок квантизації

Вирівнювання двійкових слів дозволяє чітко бачити розподіл бітів, пов'язаних із зареєстрованими значеннями параметрів. На малюнку 2.7 старший біт (MSB) у кожному стовпці, позначений зеленим кольором, дає уявлення про записані значення. [22].

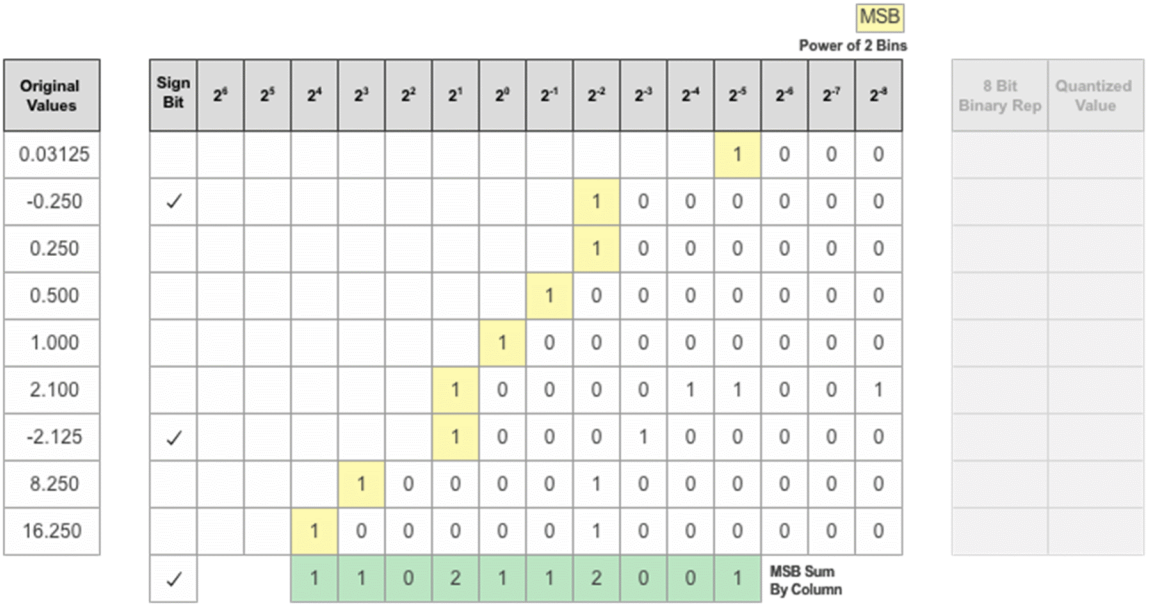

Рис. 2.7. Третій крок квантизації

Теплова карта показує кількість MSB для кожного біта. У цій візуалізації області з більш темним синім кольором вказують на більшу кількість MSB у розташуванні бітів, представленому на рис. 2.8.

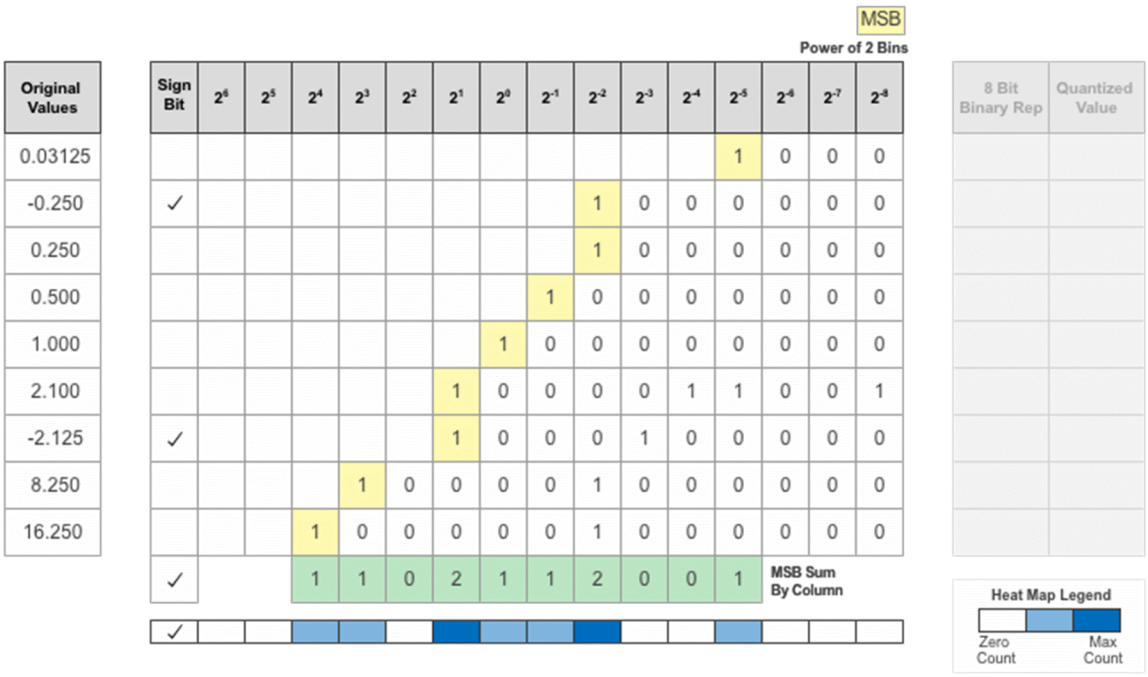

Рис. 2.8. Четвертий крок квантизації

Наступним кроком алгоритму є призначення типу даних, який запобігатиме переповненню та охоплюватиме необхідний діапазон. Для представлення значення зі знаком потрібен додатковий знаковий біт. [23].

Після визначення типу даних будь-які біти, що виходять за межі цього типу даних, усуваються. Коли менший тип даних фіксованої довжини присвоюється значенням, які він не може адекватно представити, це може призвести до втрати точності або переповнення. Наприклад, значення 0,03125 відчуває відтінок, що призводить до квантованого значення 0. Тим часом квантове значення для 2,1 становить 2,125. Крім того, значення 16,250 перевищує максимально представлене значення типу даних, що спричиняє переповнення, і, таким чином, квантоване значення встановлюється на 15,874.

Рисунок 2.9 ілюструє результат квантування, демонструючи, що тепер кожне число використовує максимум 8 біт інформації.

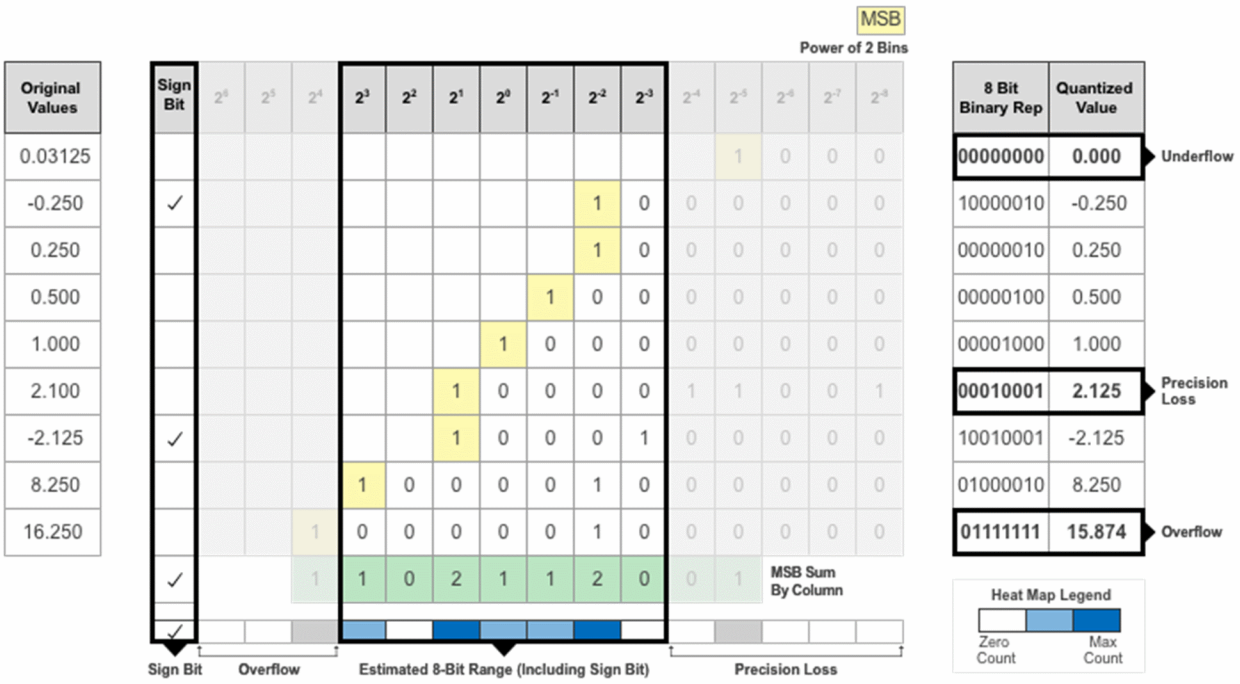

Рис. 2.9. Фінальний крок квантизації

2.5 Вибір наборів даних для вирішення задачі

Як найбільш придатні набори даних для вирішення задачі були обрані набори Places2 і Im2GPS, які дозволяють отримати достатню кількість балів для навчання багатозадачної мережі. [13].

Набір даних Places2 розроблено на основі принципів людського візуального пізнання. Ця колекція створює ядро візуальних знань, які можна використовувати для навчання штучних систем для виконання розширених завдань візуального розуміння, таких як контекст сцени, розпізнавання об’єктів, передбачення дій і подій, а також теорія мислення. Місця в цьому наборі даних визначаються їхньою функціональністю: мітки представляють базові рівні контексту. Щоб проілюструвати це, набір даних містить різні категорії спалень, вулиць тощо. На рисунку 2.10. представлено кілька класів таких візуалізацій.

Рис. 2.10. Вибірка зображень з набору даних

Загалом Places2 містить понад 10 мільйонів зображень, у тому числі понад 400 унікальних категорій сцен. Набір даних містить від 5000 до 30000 навчальних зображень на клас, подібних до реального світу. Набір даних Places2 використовує згорткову нейронну мережу (CNN), яка дозволяє вивчати глибокі функції для налаштувань зображення для різноманітних завдань класифікації.

Дуже корисною функцією набору даних є розподіл класів за частотою, оскільки баланс навчальних вибірок впливає на якість згенерованої моделі. На рисунку 2.11 відображено розподіл кількості зображень у базі даних, відсортованих за категоріями. Places2 містить 10624928 зображень із 434 категорій. Назви категорій відображаються кожні 6 інтервалів. [14].

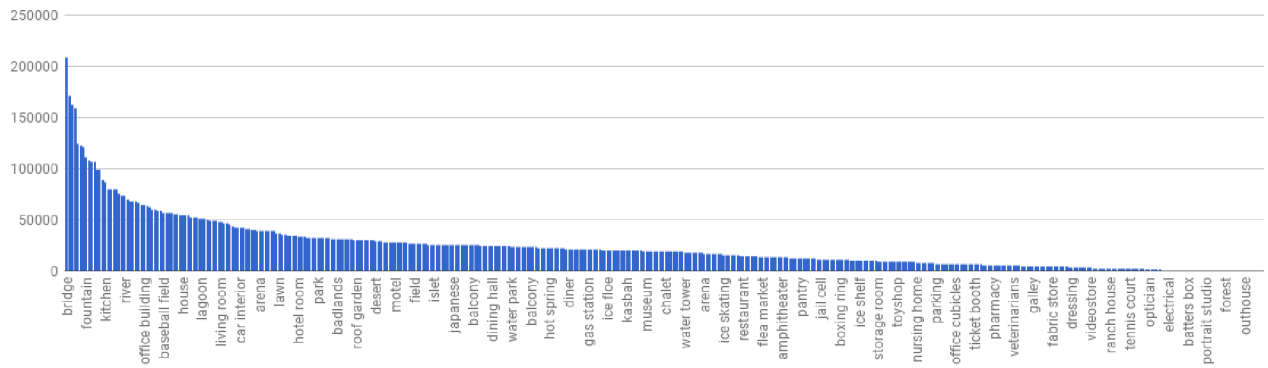

Рис. 2.11. Кількості міток для кожного класу

Наступний набір даних містить набір зображень і відповідні GPS-координати. Оскільки оцінка географічної інформації із зображень є цікавою та складною проблемою комп'ютерного зору високого рівня, наявність великої кількості геокаліброваних даних зображень є важливою причиною використання комп'ютерного зору в усьому світі! Автори цього набору даних пропонують простий алгоритм для оцінки географічного розподілу окремих зображень, використовуючи підхід до картографування сцени на основі даних. Для цього завдання було використано набір даних із понад 6 мільйонів зображень із тегами GPS, зібраних з Інтернету. Im2GPS — це найбільший набір даних такої якості. Рисунок 2.22 показує розподіл даних, присутніх у наборі.

Рис. 2.22. Розподіл даних Im2GPS на земній кулі

Знання розподілу можливих географічних розташувань зображення може надати велику кількість додаткових кліматичних метаданих, середньої температури в певний день, індексу рослинності, які разом можуть дуже чітко визначити потрібне місце.

РОЗДІЛ 3
РЕЗУЛЬТАТИ ВИРІШЕННЯ ЗАДАЧІ

3.1. Розробка архітектури системи розпізнавання розташування місцевості на зображенні

Спочатку розглядається високорівнева архітектура системи, яка охоплює фази поділу землі на комірки та обробки даних з різних наборів (рис. 3.1).

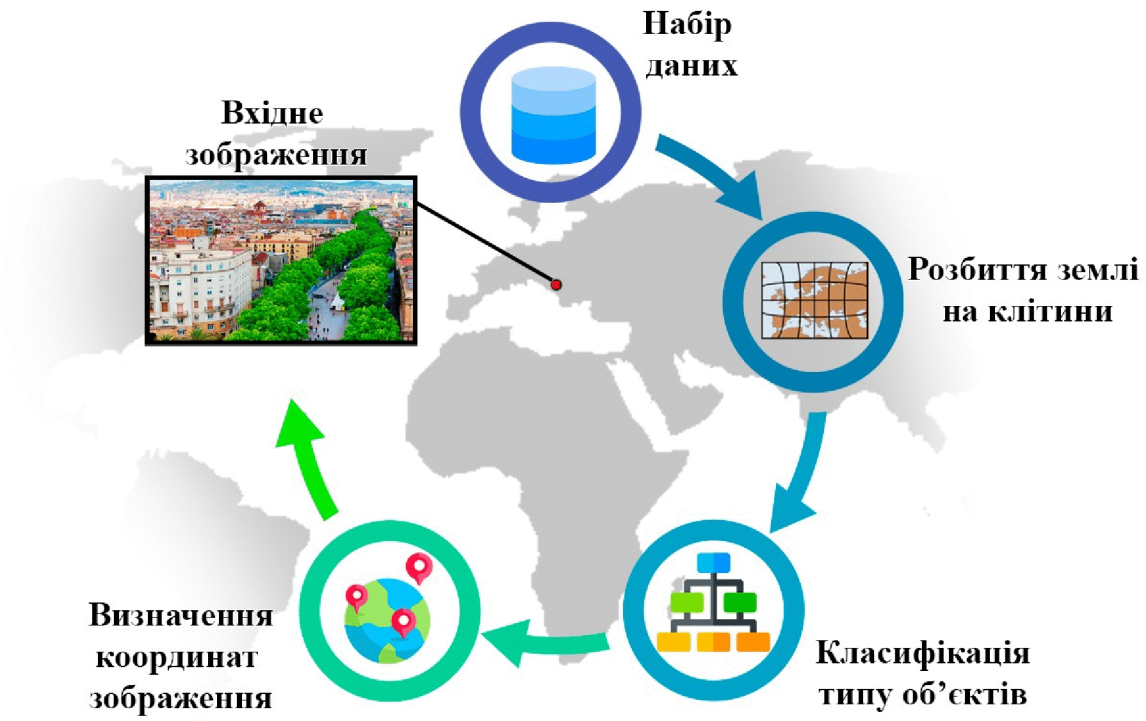

Рис. 3.1. Архітектура системи

Нейронна мережа навчається для завдання класифікації за допомогою завантажених наборів даних і вирішальним фактором у цьому процесі є вибір класів, які мережа використовуватиме. Щоб досягти цього необхідно визначити відповідний спосіб поділу земної кулі на клітини.

Для точної обробки різної кількості зображень використовуються різні

розширення для їх розподілу на комірки. На рисунку нижче (рис. 3.2) показано процедуру навчання, поділу та агрегування ймовірностей.

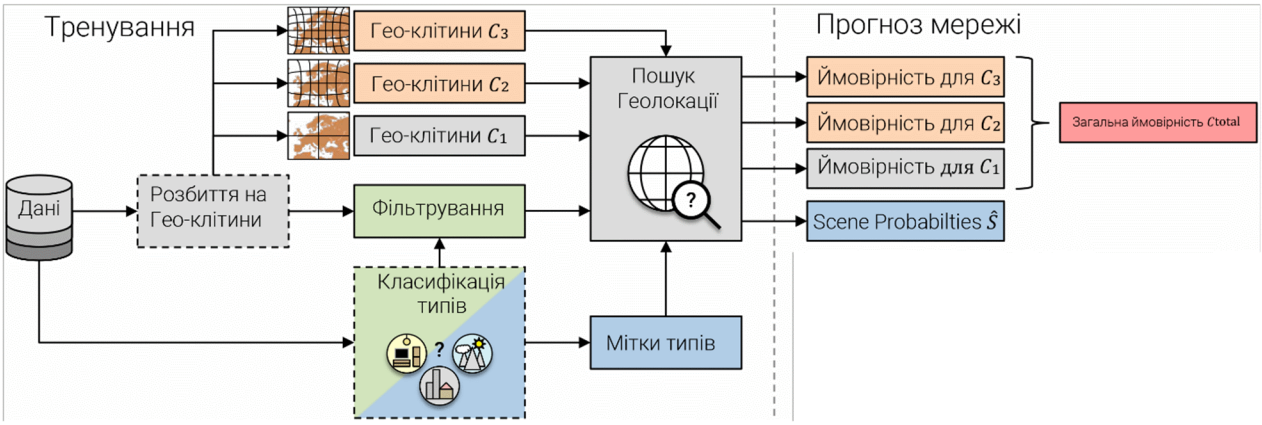

Рис. 3.2. Архітектура використання системи

Багатозадачна нейронна мережа: Щоб усунути неефективність вищезгаданого методу геолокації, який бореться з великою різноманітністю зображень навколишнього середовища в різних регіонах, пропонується двозадачна архітектура навчання. Це передбачає додавання повністю зв'язаного рівня після рівня «глобального об'єднання» в архітектурі згорткової мережі ResNet50. Кількість нейронів у цьому додатковому шарі відповідає кількості категорій для типів зображень |S|. Параметри класифікатора для обох завдань ідентичні. Для класифікації типу крос-ентропія була обрана як функція витрат. Загальна функція витрат L_{total} для мережі виходить із суми L_{scene} і L_{geo} для двох завдань.

$$L_{total} = L_{scene} + L_{geo} \tag{3.1}$$

У представленій роботі обидві функції втрат мають однакове значення. Кожен компонент обчислюється незалежно за допомогою категоріальної крос-ентропії з урахуванням відповідної кількості класів.

$$L_{CCE} = -\sum_{i=1}^{n} y_i * \log \hat{y}_i \quad (3.2)$$

Такий підхід дозволяє нейронній мережі отримувати завдання у форматі, який сприяє узагальненню цільової функції.

3.2. Метод кодування географічних координат місцевості

Для ефективного вирішення задачі класифікації множини визначення географічних координат об'єктів необхідно вибрати оптимальний спосіб поділу земних одиниць.

Для цього використовується бібліотека геометрії S2. Це допомагає створити кілька клітин C, які не перекриваються. Більш детально, поверхня Землі проектується на описаний куб, причому шість граней представляють початкові одиниці (рис. 3.3).

Потім виконується адаптивний ієрархічний поділ граней на основі GPS-координат у наборі даних Im2GPS таким чином, що кожна комірка є вузлом квадродерева. Починаючи з кореневого вузла, відповідне квадродерево рекурсивно розбивається до тих пір, поки всі комірки не містять не більше τ_{max} зображень.

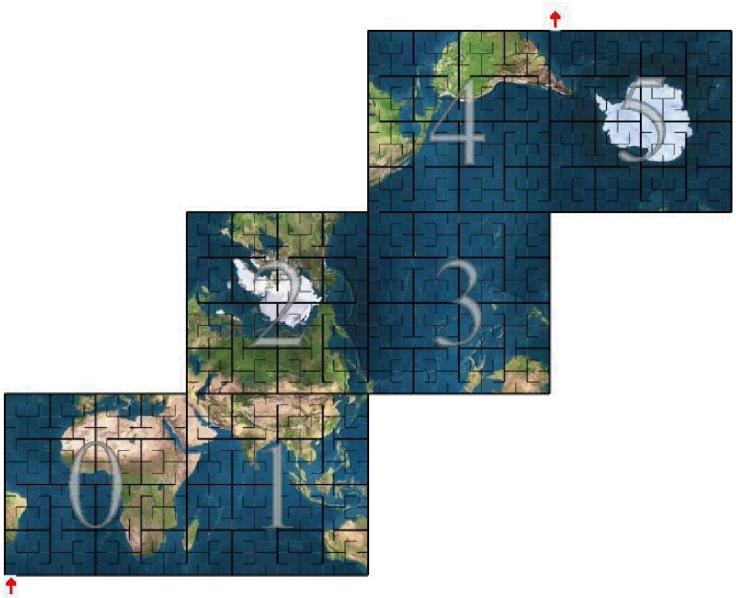

Рис. 3.3. Архітектура використання проектованої системи

Традиційна картографія базується на картографічних проекціях, які є простими функціями, що відображають точки земної поверхні на точки на плоскій карті. Картографічні проекції викликають спотворення, оскільки форма Землі не дуже відповідає формі площини. Наприклад, добре відома проекція Меркатора розривна вздовж 180-градусного меридіана, має великомасштабні спотворення на високих широтах і взагалі не може представляти північний і південний полюси. Інші проекції роблять різні компроміси, але жодна плоска проекція не відображає добре всю поверхню Землі.

S2 вирішує цю проблему, використовуючи виключно сферичну проекцію. Як випливає з назви, сферична проекція відображає точки на поверхні Землі на ідеальну математичну сферу. Звичайно, такі відображення все ж створюють певні спотворення, оскільки Земля не є ідеальною кулею, але виявляється, що Земля ближче до сфери, ніж до плоскої поверхні. За допомогою сферичної проекції можна апроксимувати всю поверхню Землі з максимальним спотворенням 0,56%. Можливо, що більш важливо, сферична проекція зберігає правильну топологію Землі — немає сингулярностей або розривів, з якими потрібно мати справу.

Крім того, одиниці S2 впорядковуються послідовно вздовж кривої заповнення простору (фрактальний тип). Конкретна крива, яка використовується для розщеплення S2, називається кривою S2 і складається з шести кривих Гільберта, з'єднаних разом, щоб утворити єдину безперервну петлю, що охоплює всю сферу. На рис. 3.4 показана крива S2 після 5-ступеневої сепарації.

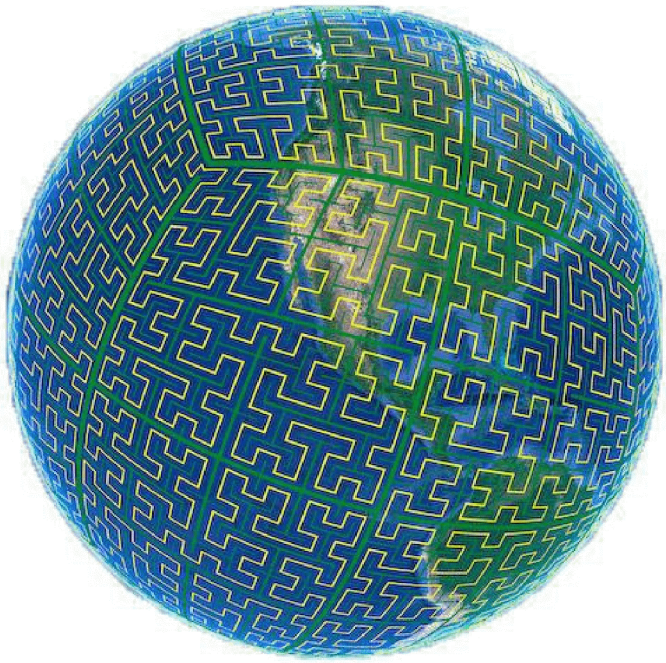

Рис. 3.4. Крива заповнення простору

3.3. Особливості використання обраних технологій та мов програмування під час вирішення задачі

Інструменти, які використовувалися для розробки, включали мову програмування Python, JavaScript, менеджер пакетів pip та інтегроване середовище розробки PyCharm. Крім того, Docker служив набором інструментів для обробки ізольованих контейнерів Linux.

Важливим аспектом, який слід враховувати, є вибір мови програмування. Для цього проекту Python дуже підходить, оскільки він пропонує можливості та швидкість, порівняні з C++ у цьому контексті, а також його простіше використовувати для розробки серверної логіки та керування базами даних.

Загалом ця мова фокусується на швидкій і якісній розробці додатків, зберігаючи міцну кодову базу (що дозволяє інтегрувати різні модулі, адаптовані до конкретних завдань). Цей додаток може використовувати графічний процесор (GPU) для підвищення продуктивності нейронних мереж.

Крім того, оскільки Python є інтерпретованою мовою з міцною структурою, необхідний обсяг коду буде значно меншим порівняно з

програмою, розробленою на скомпільованій мові, як-от C++. Однак одним недоліком є те, що недосвідченим людям може бути складно вивчити різні модулі цієї мови програмування та зрозуміти принципи інтерпретатора та їх взаємодії. Python підтримує поєднання процедурних та об'єктно-орієнтованих технік, а також узагальнене програмування та метапрограмування як у статичному, так і в динамічному стилях.

Pip служить системою керування пакетами, розробленою для встановлення та керування програмними пакетами, створеними на Python. Численні пакети доступні через індекс пакетів Python.

PyCharm служить інтегрованим середовищем розробки, спеціально розробленим для мови програмування Python. Він пропонує різноманітні інструменти для аналізу коду, засіб перегляду графіки, інструмент модульного тестування та підтримку веб-розробки Django. Це програмне забезпечення створено компанією JetBrains і базується на IntelliJ IDEA.

Django — це високорівнева структура з відкритим вихідним кодом на Python, призначена для створення веб-систем. PyTorch, бібліотека з відкритим кодом для машинного навчання, створена на основі бібліотеки Torch, використовується для таких завдань, як комп'ютерне бачення та обробка природної мови. Випущене за модифікованою ліцензією BSD, це безкоштовне програмне забезпечення з відкритим кодом. Хоча основна увага при розробці зосереджена на відшліфованому інтерфейсі Python, PyTorch також надає зовнішній інтерфейс для C++. Ця бібліотека дозволяє швидко реалізувати архітектуру обробки природної мови для оцінки коментарів користувачів (рис. 3.5).

JavaScript (JS) — це об'єктно-орієнтована динамічна мова програмування на основі прототипу. Він заснований на стандарті ECMAScript і в основному використовується для розробки сценаріїв для веб-сторінок. Це дозволяє взаємодіяти з користувачами на стороні клієнта, контролювати браузер, асинхронний обмін даними з сервером і модифікувати структуру та візуальні елементи веб-сторінки.

CSS — це унікальна мова стилів, призначена для визначення вигляду сторінок. Ці сторінки створено за допомогою мов розмітки даних. За допомогою CSS можна застосовувати до документа стилі (рис. 3.6). Хоча CSS в основному використовується для візуального відображення сторінок, створених у HTML і XHTML, його також можна використовувати з різними типами XML-документів.

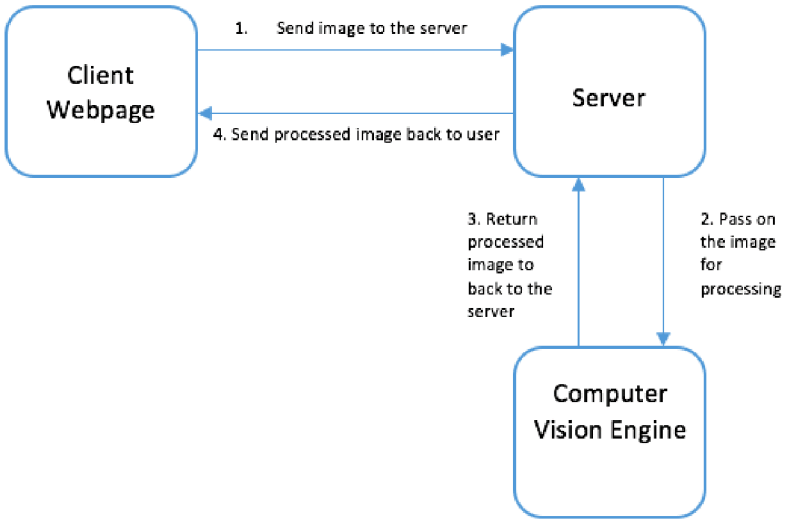

Рис. 3.5. Архітектура системи обробки ображень

Консорціум World Wide Web відповідає за створення та постійний розвиток специфікацій CSS.

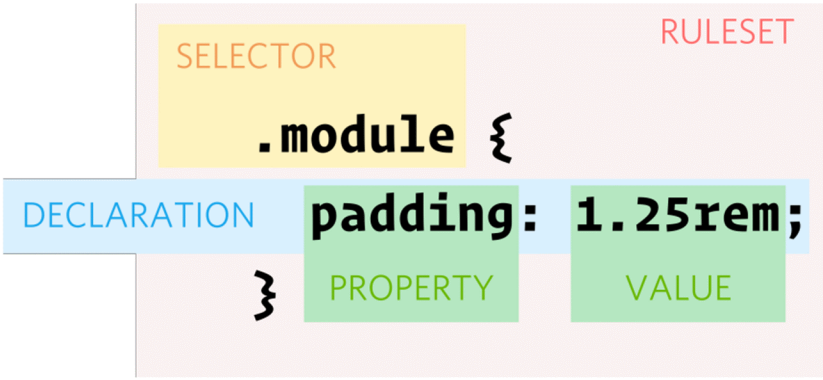

Рис. 3.6. Візуалізація CSS-стилю

CSS складається з різних рівнів і профілів. Кожен наступний рівень базується на попередніх, додаючи додаткові функції або вдосконалюючи

поточні. Ці рівні ідентифікуються як CSS1, CSS2 і CSS3. Профілі являють собою набір правил CSS з одного або кількох рівнів, розроблених для конкретних пристроїв або інтерфейсів.

HTML, що розшифровується як HyperText Markup Language, є мовою розмітки, яка використовується для створення гіпертекстових документів, призначених для Інтернету. Веб-браузери отримують документи HTML або з веб-сервера, або з локального сховища, а потім передають ці документи на мультимедійні веб-сторінки. Семантично HTML описує структуру веб-сторінки (див. рис. 3.7.) і початково містить інструкції щодо візуального представлення документа.

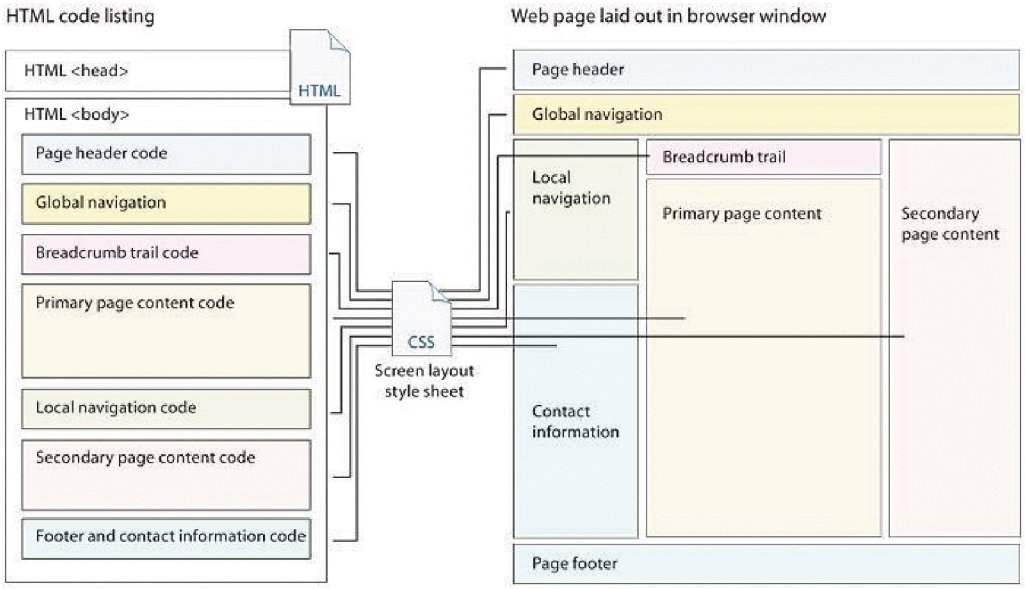

Рис. 3.7. Аналіз послідовності HTML-коду на складній веб-сторінці разом із відповідним макетом, створеним для кожного сегмента коду.

Інакше кажучи, HTML служить мовою розмітки, по суті, методом зберігання інформації. Використовуючи HTML, текст анотується, сигналізуючи веб-браузеру, як інтерпретувати позначений текст. Так само на жорсткому диску дані організовані в блоки, кластери, сектори та доріжки; саме цей структурований формат дозволяє комп'ютеру розрізняти, що важливо, а

що слід ігнорувати.

У HTML теги використовуються для позначення тексту. Кожен HTML-документ складається з набору елементів, кожен з яких визначається певними тегами, які вказують на їх початок і кінець (хоча деякі елементи не потребують кінцевого тегу). Тег складається з імені елемента, поміщеного в кутові дужки (< >).

Кожний HTML-тег має окрему назву з певним синтаксисом, що складається з латинських літер і не залежить від регістру. З 1997 року World Wide Web Consortium (W3C), організація, відповідальна за нагляд за стандартами HTML і CSS, просуває використання CSS замість прямого презентаційного HTML.

HTML забезпечує спосіб створення структурованого документа шляхом позначення структурних елементів тексту, таких як заголовки, абзаци, списки, таблиці, цитати тощо. Це розробка інтерактивних форм, а саме включення в текст зображень, аудіо, відео та різних інших елементів..

Docker служить набором інструментів, розробленим для керування ізольованими контейнерами Linux. Це дозволяє автоматизувати розгортання веб-сайту. Крім того, можна використовувати різні інтерфейси для доступу до віртуалізації ядра операційної системи (рис. 3.10), а також можливість працювати з різними дистрибутивами.

Вихідний код Docker розроблено на мові програмування Go та доступний за ліцензією Apache 2.0. Цей набір інструментів використовує стандартні методи ізоляції, які інтегровані в ядро Linux, зокрема з використанням просторів імен і контрольних груп (cgroups). Контейнери створюються за допомогою скриптів lxc. Щоб налаштувати контейнер, потрібно просто завантажити базовий образ середовища за допомогою команди (docker pull base). Після цього можна виконувати різні програми в ізольованих середовищах; наприклад, щоб запустити bash, ви можете виконати команду docker run -i -t base/bin/bash.

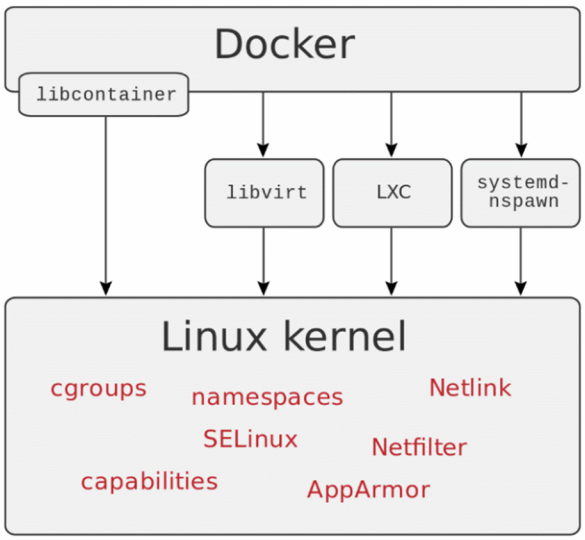

Рис. 3.8. Доступ до засобів віртуалізації ядра Linux через різні інтерфейси.

3.4. Опис реалізації розробленої інформаційно-аналітичної системи

Для технічних засобів сервера рекомендована така конфігурація:

- процесор серії Intel® Xeon з тактовою частотою 2,4 ГГц;
- оперативна пам’ять 4 x DIMM DDR4 ECC 32 GB;
- SSD диск 1x512 GB SATA3 або PCI-e x3
- Жорсткий диск 3x 1000 ГБ SATA 3, 7200 RPM;
- Доступ до мережі Інтернет.

Для технічних засобів клієнта рекомендована така конфігурація:

- процесор серії Intel® Core з тактовою частотою 3 ГГц;
- оперативна пам’ять 2 x DIMM DDR4 16 GB;
- SSD диск 1x512 GB SATA3 або PCI-e x3
- Жорсткий диск 1 x 250 ГБ SATA 2 16 МБ буфера, 7200 RPM;

- рідкокристалічний монітор з діагоналлю не менше 17 дюймів;

- доступ до мережі Інтернет;
- клавіатура;
- маніпулятор «Мишка».

Вищевказані технічні характеристики є рекомендованими, тобто при наявності технічних засобів не нижче заданих розроблена інформаційно-аналітична система буде працювати відповідно до вимог надійності, швидкості обробки даних та безпеки, запропонованих замовником.

У реалізації проекту використовуються мови програмування Python та Javascript. Крім того, веб-браузер, який підтримує технологію JavaScript, необхідний для належного функціонування розробленої програми.

Для головного комп'ютера рекомендоване передвстановлене ПЗ:

– Операційні системи включають Windows (10,11) або Linux;

– HTTP-сервер nginx для розгортання на клієнтському комп'ютері: – сумісний з операційними системами Windows (10,11) та Linux;;

– Веб-браузери, такі як Firefox, Google Chrome і Opera.

Після розгортання веб-програми на сервері наступним кроком є перехід до домену, де буде розміщуватися програма. Сайт завантажуватиметься та працюватиме через веб-браузер, який підтримує JavaScript.

Інтерфейс користувача складається зі сторінки, яка оновлюється в режимі реального часу.

Щоб отримати доступ до веб-програми, перейдіть до вказаного доменного імені http://127.0.0.1:5000/ (за умови, що веб-програма працює локально). Основний інтерфейс програми служить головною сторінкою. (Див. рис. 3.9.) Ця сторінка містить кілька функціональних можливостей, включаючи кнопку для завантаження зображення для обробки, карту, яка показує результат, і приклади малюнків.

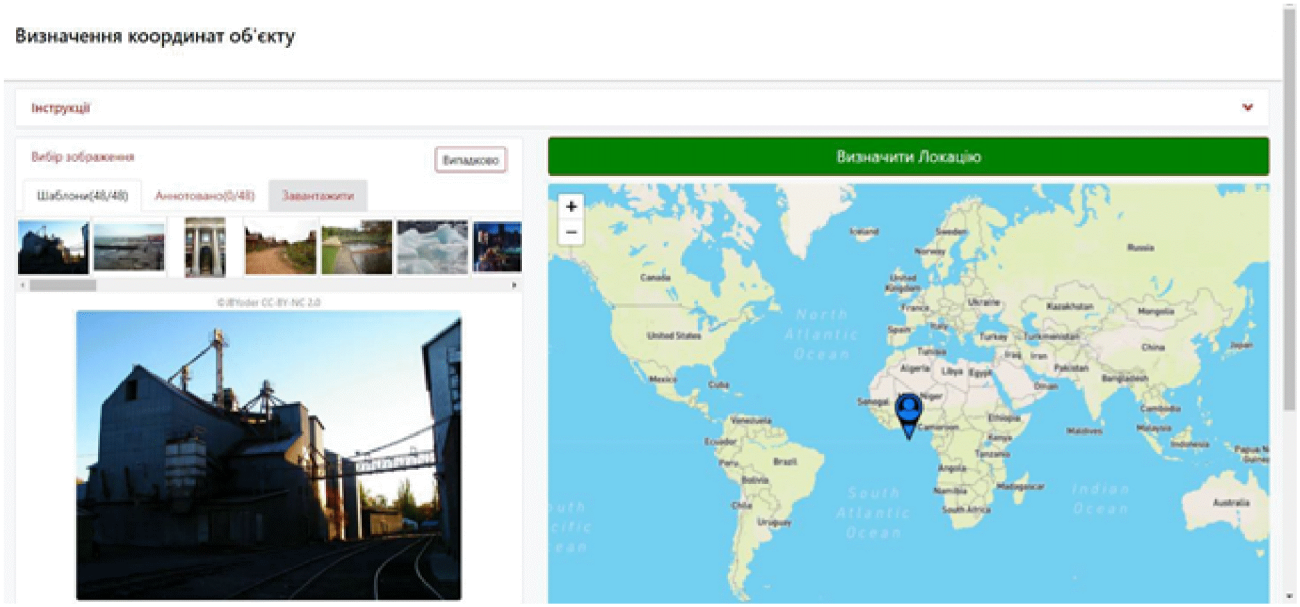

Рис. 3.9. Інтерфейс запроектованої системи розпізнавання зображень

Було оцінено різноманітні зображення, що призвело до візуалізації географічних координат об’єктів, визначених на цих зображеннях. Для цього тестування були обрані особисті фотографії подорожей. Аналіз зосереджується на результатах, створених мережею на основі вхідних даних зображення. На малюнку 3.10 зображено колесо огляду в Лондоні, яке демонструє точність мережі у визначенні потенційних координат об’єкта виключно за зображенням (жовтий маркер вказує справжнє розташування, тоді як білий маркер представляє найбільш імовірне місце розташування системи). На малюнках 3.12 і 3.13 показано ще один об’єкт, знятий з пляжу в Барселоні, демонструючи, що навіть за допомогою такої фотографії місце розташування можна точно визначити в радіусі 100 метрів.

Рис. 3.10. Вхідне зображення 1

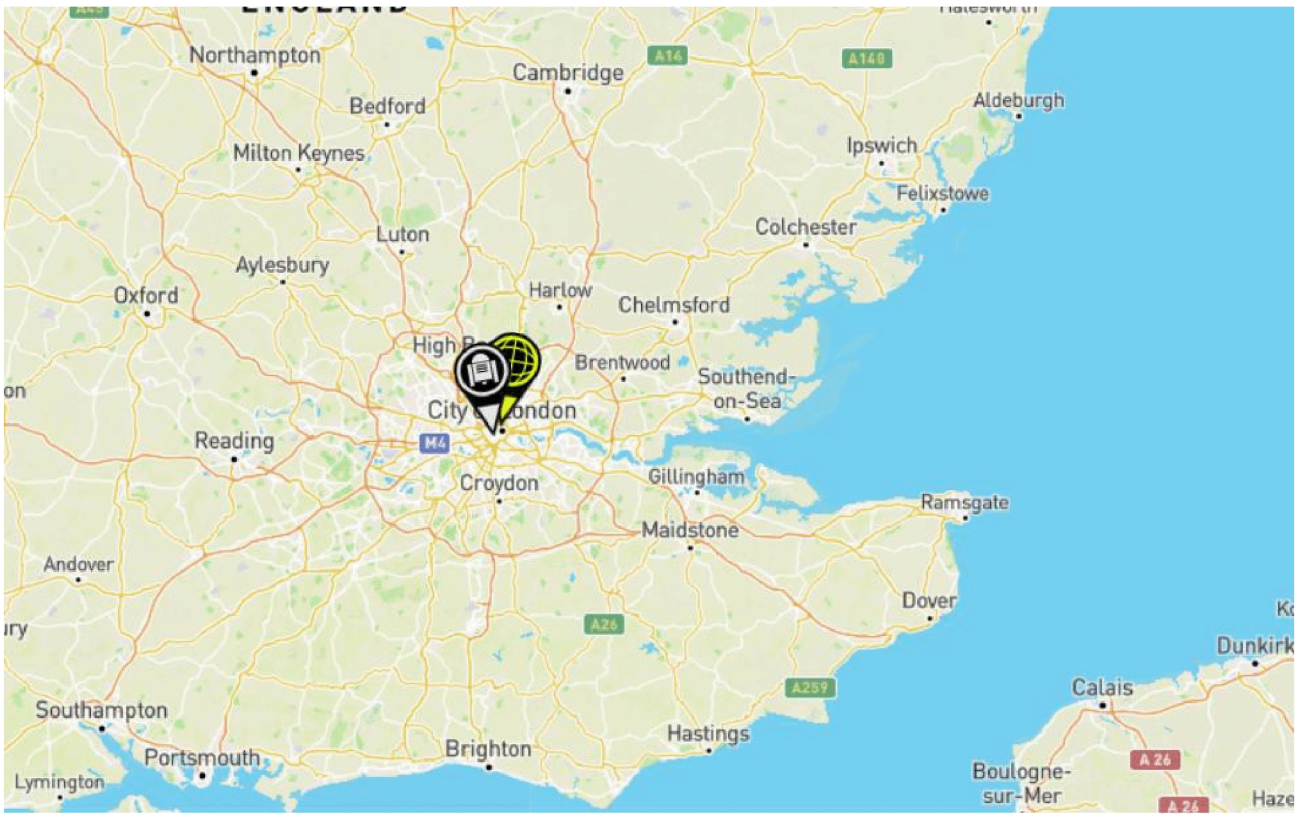

Рис. 3.11. Вихід мережі 2

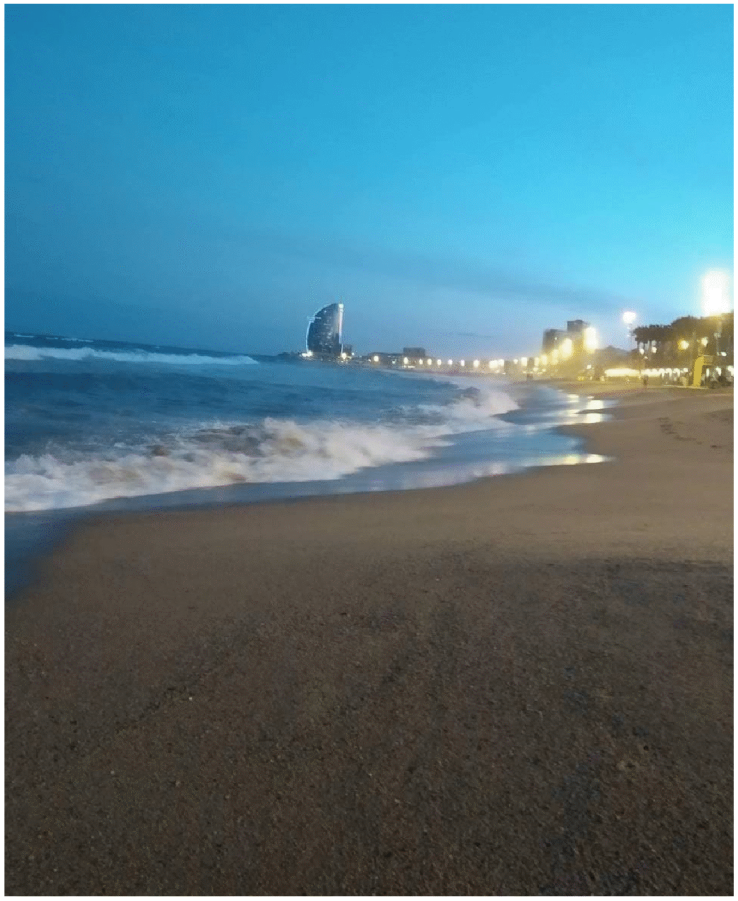

Рис. 3.12. Вхідне зображення 2

Рис. 3.13. Вихід мережі 2

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Розробка логіко-імітаційної моделі виникнення травм і аварій

Методикою оцінки рівня небезпеки робочих місць, машин, виробничих процесів та окремих виробництв передбачено пошук об'єктивного критерію рівня небезпеки для конкретного об'єкта [20]. Таким показником вибрана ймовірність виникнення аварії, травми залежно від явища, що досліджується.

Для побудови логіко-імітаційної моделі процесу, формування і виникнення аварії та травми в процесі створення мікрокліматичних умов у приміщенні оцінюють відповідні небезпечні події. Кожній іх них присвоїмо ймовірність виникнення:

Шифр	Назва події	Ймовірність
P_1	Відсутність захисного заземлення	0,02
P_2	Пошкодження захисного заземлення	0,04
P_3	Спрацювання складових захисту	0,1
P_4	Неправильна експлуатація захисту	0,02
P_5	Відсутність профілактичних заходів	0,2
P_6	Відсутність захисного щита	0,12
P_7	Недотримання правил вибору взуття	0,15
P_8	Незнання правил техніки безпеки	0,1
P_9	Відсутність засобів індивідуального захисту	0,2
P_{10}	Легковажність	0,08

На основі наведених подій будуємо матрицю логічних взаємозв'язків між окремими пунктами, графічна інтерпретація якої зображено на рис. 4.1.

Розрахуємо ймовірності виникнення подій, що формують логіко-імітаційну модель процесів створення мікрокліматичних умов. Розглянемо травмонебезпечну ситуацію, що виникає за умови роботи працівників із електронебезпекою.

Підставивши дані ймовірностей базових подій у формулу, отримаємо ймовірність події 13: $P_{13} = 0{,}2 + 0{,}4 - 0{,}2 \cdot 0{,}4 = 0{,}0592$.

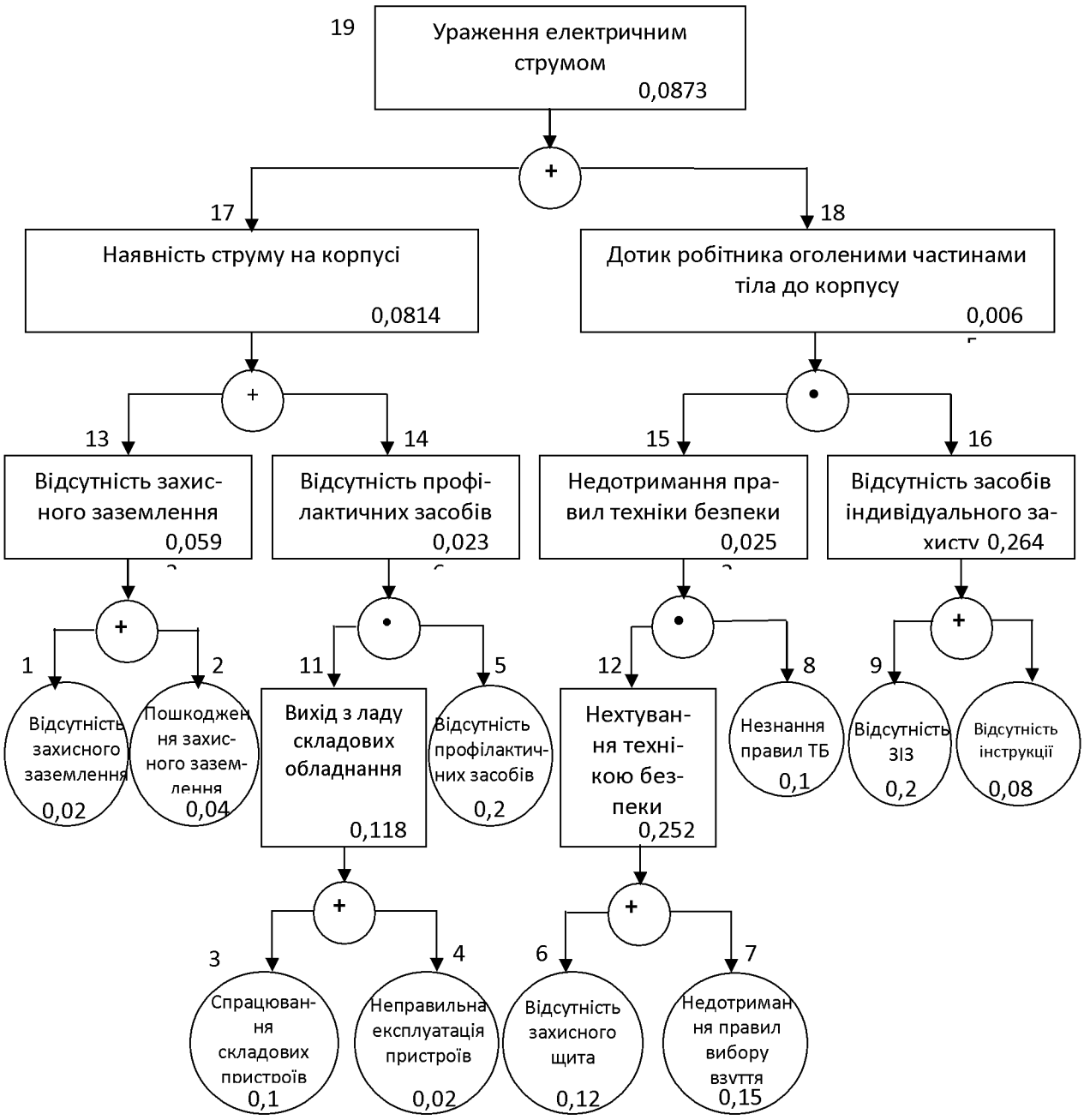

Рис. 4.1. Матриця логічних взаємозв'язків між окремими подіями травмонебезпечної ситуації [ ]

Аналогічно визначаємо ймовірність інших подій:

$$P_{11} = P_4 + P_5 - P_4P_5 = 0,3 + 0,4 - 0,3 \cdot 0,4 = 0,118.$$

$$P_{12} = P_6 + P_7 - P_6P_7 = 0,3 + 0,5 - 0,3 \cdot 0,5 = 0,252.$$

$$P_{16} = P_9 + P_{10} - P_9P_{10} = 0,2 + 0,15 - 0,2 \cdot 0,15 = 0,264.$$

$$P_{14} = P_{11} \cdot P_5 = 0,118 \cdot 0,2 = 0,0236.$$

$$P_{15} = P_{12} \cdot P_8 = 0,252 \cdot 0,1 = 0,0252.$$

$$P_{17} = P_{13}+P_{14} - P_{13}\cdot P_{14} = 0{,}592+0{,}0236-0{,}0592\cdot 0{,}0236 = 0{,}0814.$$

$$P_{18} = P_{15}\cdot P_{16}=0{,}264\cdot 0{,}0252=0{,}0065.$$

$$P_{19} = P_{17}+P_{18} - P_{17}\cdot P_{18} = 0{,}0065+0{,}0814 - 0{,}0065\cdot 0{,}0814 = 0{,}0873.$$

Таким чином, ймовірність перекидання машини та наслідкового виникнення травми працівника є досить мала і становить – $P_{19} = 0,0873$.

4.2. Планування заходів із покращення умов праці

До заходів щодо покращення умов праці належать всі види діяльності, спрямовані на попередження, нейтралізацію або зменшення негативної дії шкідливих і небезпечних виробничих факторів на працівників.

Рівень умов праці оцінюють порівнянням за фактичними і нормативними значеннями узагальнених (групових) показників.

Заходи щодо поліпшення умов праці здійснюють з метою створення безпечних умов праці шляхом:

- доведення до нормативного рівня показників виробничого середовища за елементами умов праці;
- захисту працівників від дії небезпечних і шкідливих виробничих факторів.

До показників ефективності заходів щодо поліпшення умов праці належать:

а) зміни стану умов праці:

- зміна кількості засобів виробництва, приведених у відповідність до вимог стандартів безпеки праці;
- покращання санітарно-гігієнічних показників;
- покращання психофізичних показників, зменшення фізичних і нервово-психічних навантажень, в т.ч. монотонних умов праці;
- покращання естетичних показників, раціональне компонування робочих місць і впорядкування робочих приміщень;

б) соціальні результати заходів:

- збільшення кількості робочих місць, що відповідають нормативним вимогам;
- зниження рівня виробничого травматизму;
- зменшення кількості випадків професійних захворювань;
- зменшення плинності кадрів через незадовільні умови праці;
- престиж та задоволення працею.

Отже, на покращення охорони праці потрібно виділити кошти на відновлення вентиляційних систем у ремонтних майстернях, естетично оформити приміщення офісу, відновити кабінет з охорони праці, поновити протипожежний інвентар.

4.3. Безпека в надзвичайних ситуаціях

Актуальність проблеми природно-техногенної безпеки для населення і території, зумовлена зростанням втрат людей, що спричиняється небезпечними природними явищами, промисловими аваріями та катастрофами. Ризик надзвичайних ситуацій природного та техногенного характеру невпинно зростає, тому питання захисту цивільного населення від надзвичайних ситуацій на сьогодні є дуже важливе [20].

У системі цивільної оборони окремого господарства необхідно забезпечити захист населення таким чином:

Укриття в захисних спорудах, якому підлягає усе населення відповідно до приналежності, досягається створенням фонду захисних споруд.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об'єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення його у позаміській зоні.

Медичний захист проводиться для зменшення ступеня ураження людей, своєчасного надання допомоги постраждалим та їх лікування, забезпечення

епідеміологічного благополуччя в районах надзвичайних ситуацій.

Радіаційний і хімічний захист включає заходи щодо виявлення і оцінки радіаційної та хімічної обстановки, організацію і здійснення дозиметричного та хімічного контролю, розроблення типових режимів радіаційного захисту, забезпечення засобами індивідуального захисту, організацію і проведення спеціальної обробки.

Евакуаційні заходи, які проводяться в містах та інших населених пунктах, які мають об'єкти підвищеної небезпеки, а також у воєнний час, основним способом захисту населення є евакуація і розміщення у позаміській зоні [20].

5. ВИЗНАЧЕННЯ ЕФЕКТИВНОСТІ РОЗРОБКИ ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ

5.1. Оцінка складності розробки програмного забезпечення

Ключові етапи розробки програмного забезпечення включають оцінку складності проекту та оцінку витрат, пов'язаних із створенням програмного продукту.

Початкові дані для розрахунку економічної ефективності:

1. Прогнозована кількість операторів – 600.
2. Коефіцієнт складності програми – 1,5;
3. Коригувальний коефіцієнт для програми при її розробці – 0,1;
4. Погодинна оплата програміста 320 грн./год.
5. Коефіцієнт збільшення витрат на оплату праці через неадекватний опис завдання – 1,3;
6. Кваліфікаційний коефіцієнт програміста, виходячи з досвіду роботи в даній сфері, становить 0,8;
7. Витрати на одну годину роботи комп'ютера становлять 10 грн./год.

Стандартизації праці в розробці програмного забезпечення значною мірою заважають творчі аспекти завдань програміста. У результаті можна оцінити складність розробки програмного забезпечення за допомогою модельної системи, яка пропонує різні рівні точності оцінки на основі такої формули:

$$t = t_o + t_u + t_a + t_n + t_{\text{відл}} + t_{\text{д}}, \text{ людино-годин} \qquad (4.1),$$

де t_o відноситься до витрат праці, пов'язаних із підготовкою та плануванням завдання (з нормативним значенням 50);

t_u −витрати праці на дослідження алгоритму рішення;

t_a – вказує витрати праці на створення блок-схеми алгоритму;

t_n – витрати на оплату праці для програмування;

$t_{\text{відл}}$ – витрати на оплату праці, пов'язані з відлагодженням програми на комп'ютері.

$t_д$ – позначає витрати праці, пов'язані з підготовкою документації.

Ці складові витрати на оплату праці встановлюються на основі умовної кількості операторів у програмному забезпеченні та обчислюються за такою формулою:

$$Q = q * C * (1 + p), \tag{4.2}$$

де q – очікувана кількість операторів;

C – коефіцієнт складності програми;

p – коригуючий коефіцієнт програми під час її розробки.

Таким чином отримуємо такий розрахунок:

$$Q = 600 * 1{,}5 * (1 + 0{,}1) = 1089 \text{ людино-годин.}$$

Витрати праці на розбір опису завдання встановлюються виходячи з деталей опису та кваліфікації програміста:

$$t_u = \frac{Q*B}{(75...85)*k} \tag{4.3}$$

де B - збільшення витрат на робочу силу внаслідок неадекватних описів задачі.

k - коефіцієнт, що відображає кваліфікацію програміста, виходячи з досвіду роботи. У цій сфері діє коефіцієнт 0,8 за наявності у спеціаліста стажу роботи до 2 років.

$$t_u = \frac{1089*1,3}{80*0,8} = 22,2$$ людино-години,

Витрати на збірку програми на основі заздалегідь розробленої блок-схеми:

$$t_n = \frac{Q}{(20...25)*k}$$, людино-годин.

$$t_n = \frac{1089}{25*0,8} = 54,5$$, людино-годин.

Витрати, пов'язані з оплатою праці по встановленню програми на комп'ютері:

$$t_{\text{відл}} = \frac{Q}{(4 \dots 5) * k}$$

$$t_n = \frac{1089}{5*0,8} = 272,3$$, людино-годин.

- за умови повного відлагодження завдання:

$$t_{\text{відл}}^{\text{п}} = 1,5 * t_{\text{відл}}$$

$$t_{\text{відл}}^{\text{п}} = 1,5 * 272,3 = 408,4$$, людино-годин.

Формула для розрахунку трудовитрат, пов’язаних з підготовкою документації, визначається як

$$t_{\text{д}} = t_{\text{др}} + t_{\text{до}}$$, людино-годин.

У цьому контексті $t_{\text{др}}$ означає час, необхідний для підготовки як

матеріалів, так і рукопису:

$$t_{\text{др}} = \frac{Q}{(15...20)*k}, \text{ людино-годин}$$

$t_{\text{до}}$ – трудозатрати на редагування, друк та ведення записів:

$$t_{\text{до}} = 0{,}75 * t_{\text{др}}, \text{ людино-годин.}$$

Замінивши відповідні значення, отримаємо:

$$t_{\text{др}} = \frac{1089}{15*0{,}8} = 90{,}8, \text{ людино-годин.}$$

$$t_{\text{до}} = 0{,}75 * 55 = 68{,}1, \text{ людино-годин.}$$

$$t_{\text{д}} = 90{,}8 + 68{,}1 = 159 \text{ людино-годин.}$$

Застосовуючи формулу (4.1), можна отримати комплексну оцінку складності розробки програмного забезпечення.

$$t = 50 + 22.2 + 68.1 + 54.5 + 272.2 + 159 \approx 627 \text{ людино-годин.}$$

4.2. Витрати пов'язані з розробкою програмного забезпечення

Витрати на розробку програмного забезпечення $K_{пз}$ складаються із заробітної плати розробника $З_{зп}$ та часу, необхідного для встановлення та відлагодження програми на комп’ютері.

$$K_{пз} = З_{зп} + З_{мв} \qquad (4,5)$$

Заробітна плата виконавця розраховується за формулою:

$$З_{зп} = t * C_{ПР}\ , \qquad (4.6),$$

де t – загальна трудомісткість, виміряна в людино-годинах, а $C_{ПР}$ – середньогодинна оплата праці програміста в гривнях за годину. .

$$З_{зп} = 626 * 320 = 200320 \text{ грн.}$$

Витрати, пов'язані з машинним часом, необхідним для відлагодження програми:

$$З_{мв} = t_{відл} * С_{мг}\ , \text{ грн} \qquad (4.7)$$

де, $t_{відл}$ – складність встановлення програми на комп'ютері, год. $С_{мг}$ – це вартість однієї години використання комп'ютера, грн/год.

Розраховані таким чином витрати, пов'язані з розробкою програмного забезпечення, включаються до одноразових капітальних витрат на створення АСУП.

$$З_{мв} = 272{,}3 * 10 = 2723 \text{ грн.}$$

Таким чином, витрати, пов'язані з розробкою програмного забезпечення $К_{ПЗ}$, складають:

$$К_{пз} = 200320 + 2723 = 203043 \text{ грн.}$$

Орієнтовні терміни розробки ПЗ:

$$Т = \frac{t}{B_k \cdot F_p}\ , \text{ міс}, \qquad (4.8)$$

де B_k – кількість виконавців, F_p – місячний фонд робочого (за 40-годинного робочого тижня $F_p = 176$ годин.

Таким чином, на основі зроблених розрахунків були визначені такі показники:

- Трудомісткість на розробку ПЗ – 627 людино-годин;
- Витрати на реалізацію ПЗ – 203043 грн.;
- Орієнтовна тривалість розробки ПЗ – 3,6 місяці.

5.3. Аналіз потенційного ринку збуту розробленої інформаційно-аналітичної системи

Обґрунтування створення програмного продукту, який вирішує завдання ідентифікації GPS-координат об'єктів на фотографіях, підтверджується його значенням для журналістів і медіа-криміналістів у визначенні та дослідженні ймовірних місць розташування об'єкта на зображенні. Цією програмою можна поділитися з молоддю та завзятими мандрівниками, які можуть використовувати її, щоб відкривати нові та інтригуючі місця, опубліковані в Інтернеті, навіть за відсутності геотегів.

Унікальність результатів кваліфікаційної роботи полягає в розробці та впровадженні сучасних методів комп'ютерного зору.

Програмне забезпечення, призначене для моделювання та навчання нейронних мереж, фокусується на класифікації регіонів Землі, що дозволяє ідентифікувати географічні координати для різних об'єктів. Практичне значення цих результатів полягає у вирішенні та вирішенні проблем, пов'язаних із медіакриміналістикою, що дозволяє використовувати більш точний, автоматизований і зручний метод визначення місцезнаходження конкретних об'єктів, показаних на різних вхідних зображеннях. Крім того,

велике значення має використання нового підходу до поділу поверхні Землі на клітини.

Сфера використання. Розроблене програмне забезпечення можна використовувати для вирішення проблем медіакриміналістики, включаючи маніпуляції вмістом і метаданими.

Важливість роботи та знахідки. Створена система дозволяє ефективно ідентифікувати цільові об'єкти на знімках, значно скорочуючи як матеріальні, так і часові витрати, підвищуючи точність визначення координат.

Практична корисність програмного забезпечення випливає з методів і моделей, викладених у цій роботі, які дозволяють ефективніше визначати географічні координати об’єктів на фотографіях. Цей прогрес принесе значну користь таким галузям, як журналістські розслідування та медіа-криміналістика, завдяки підвищенню точності та зниженню витрат, пов’язаних із перевіркою гіпотез. Крім того, використання розробленого програмного забезпечення допоможе скоротити час, необхідний для вирішення завдань визначення потенційних місць розташування.

Основним конкурентом розробленого програмного забезпечення є пошукова система Google; однак він не є таким точним і здебільшого шукає схожі зображення, а не надає координати показаного місця. Програмне забезпечення, про яке йде мова, є універсальним і дозволяє користувачам отримувати координати об’єктів, зображених на фотографіях, не вимагаючи глибоких знань у нейронних мережах чи математиці. Ця доступність дозволяє менш кваліфікованим ресурсам брати участь у процесі, що зрештою призводить до скорочення витрат.

ЗАГАЛЬНІ ВИСНОВКИ ТА РЕКОМЕНДАЦІЇ

Робота над даною магістерською дисертацією призвела до розробки програмного забезпечення, призначеного для визначення географічних координат об'єктів на зображенні, використовуючи методи, які забезпечують найвищу точність зображення. Потім цільові значення метрики програмного забезпечення порівнювалися зі значеннями аналогічних систем.

Це дослідження розробляє надійну архітектуру, спрямовану на прогнозування геолокації об'єкта на конкретному зображенні, використовуючи лише піксельні дані. Методика навчання та архітектура показали точність у досягненні поставленої мети.

Крім того, було введено новий підхід, який розглядає проблему як проблему класифікації, використовуючи криву заповнення простору S2 для зображення кожного сегмента земної кулі як клітини.

Ця система розроблена для ефективної роботи в різних налаштуваннях, використовуючи оптимальний класифікатор для оцінки географічного розташування. Майбутні дослідження мають на меті дослідити, яка додаткова контекстна інформація може підвищити точність моделі на основі зібраних даних.

У ході виконання роботи було визначено трудомісткість розробки програмного забезпечення (627 людино-годин), був проведений підрахунок витрат на створення програми (203043 гривень) та визначено очікуваний період створення ПЗ (3,6 місяці). Проведено маркетинговий аналіз ринку збуту програмного продукту та визначено соціальний ефект від впровадження програмного продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chisman J.A. The clustered traveling salesman problem // Computers & Operations Research. – Volume 2. – Issue 2. – September 1975. – P. 115-119.

2. Bektas T. The multiple traveling salesman problem: an overview of formulations and solution procedures // Omega. – Elsevier. – № 34. – 2006. – P. 209-219.

3. Gutin G., Punnen A. P. The traveling salesman problem and its variations // Springer. – USA. – 2006. – 830 p.

4. Current J.R., Schilling D.A. The covering salesman problem // Transportation science. – Volume 23. – № 3. – 1989. – P. 151-229.

5. Nygard K.E., Yang C.H. Genetic algorithm for the traveling salesman problem with time windows // Computer Science and Operations Research: New Developments in their Interfaces. – Elsevier. – 2014. – P. 411-423

6. Mauricio Resende G.C., Ribeiro Celso C. A short tour of combinatorial optimization and computational complexity // Optimization by GRASP. – 2016. – P. 13-39.

7. Korte B., Vygen J. Combinatorial optimations: Theory and algorithms // Algorithms and Combinatorics. – Springer. – 2011. – Volume 21. – 659 p.

8. Ebert T., Fischer T., Belz J., Heinz T. O., Kampmann G., Nelles O. Extended Deterministic Local Search Algorithm for Maximin Latin Hypercube Designs // 2015 IEEE Symposium Series on Computational Intelligence. – 2015. – P. 375-382.

9. Kirkpatrick S., Gelatt C. D., Vecchi M. P. Optimization by Simulated Annealing // Science. – Volume 220. - № 4598. – 1983. – P.671-680.

10. Kheiri A., Özcan E., Parkes A. J. A stochastic local search algorithm with adaptive acceptance for high-school timetabling // Annals of Operations Research. – Springer. – Volume 239(1). – 2016. – P.135-151.

11. Geng X., Chen Zh., Yang W., Shi D., Zhao K. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy

search // Applied Soft Computing. – Elsevier. – Volume 11. – 2011. – P.3680.

12. Holland J. H. Genetic algorithms and the optimal allocation of trials // SIAM Journal on Computing. – Volume 2. - №2. – 1973. – P.88-105.

13. Yu W., Li B., Jia H., Zhang M., Wang D. Application of multi- objective genetic algorithm to optimize energy efficiency and thermal comfort in building design // Energy and Buildings. – Volume 88. – 2015. – P.135-143.

14. Merz P., Freisleben B. Memetic Algorithms for the Traveling Salesman Problem // Complex Systems. – 2001. - № 13. – P. 297-345.

15. Cattaruzza D., Absi N., Feillet D., Vidal T. A memetic algorithm for the multi trip vehicle routing problem // European Journal of Operational Research. Volume 236. – Issue 3. – 2014. – P.833-848.

16. Colorni A., Dorigo M., Maniezzo V. Distributed optimization by ant colonies // Proceeding of ECAL91. – Elsevier Publishing. – P.134-142.

17. Liao T., Stützle T., Oca M. A. M. de, Dorigo M. A unified ant colony optimization algorithm for continuous optimization // European Journal of Operational Research. – Volume 234. – 2014. – P.597-609.

18. Wang Z., Xing H., Li T., Yang Y., Qu R., Pan Y. A modified ant colony optimization algorithm for network coding resource minimization // IEEE Transactions on Evolutionary Computation. – Volume 20. – Issue 3. – 2016. – P.325-342.

19. Гуляницький Л.Ф., Мулеса О.Ю. До класифікації метаевристик // XXI Всеукраїнська наукова конференція «Сучасні проблеми прикладної математики та інформатики». – Львів. – 2015. – С.139-142.

20. Охорона праці: практикум /І.П. Пістун, Ю.В. Кіт, А.П.Березовецький – Суми: Університетська книга, 2000. –205с

Додаток

ФГАГМЕНТ КОДУ ПРОГРАМИ

```
import os
import sys
import re
from math import ceil
from typing import Dict, List, Tuple, Union
from io import BytesIO
import random
from pathlib import Path
from multiprocessing import Pool

import pandas as pd
from PIL import
Image import
torchvision import
torch
import msgpack

class
MsgPackIterableDatasetMultiTargetWithDynLabels(torch.utils.data.IterableDatas
et)
:
    """
    Data source: bunch of msgpack files
    Target values are generated on the fly given a mapping (id->[target1, target, ...])
    """

    def  init  (
        self, path:
        str,
```

```
        target_mapping: Dict[str, int],
        key_img_id: str = "id",
        key_img_encoded: str = "image",
        transformation=None,
        shuffle=True,
        meta_path=None,
        cache_size=6 * 4096,
        lat_key="LAT",
        lon_key="LON",
):

        super(MsgPackIterableDatasetMultiTargetWithDynLabels, self).__init__()
        self.path = path
        self.cache_size = cache_size
        self.transformation = transformation
        self.shuffle = shuffle
        self.seed = random.randint(1, 100)
        self.key_img_id = key_img_id.encode("utf-8")
        self.key_img_encoded = key_img_encoded.encode("utf-8")
        self.target_mapping = target_mapping

        for k, v in self.target_mapping.items():
            if not isinstance(v, list):
                self.target_mapping[k] = [v]
        if len(self.target_mapping) == 0:
            raise ValueError("No samples found.")

        if not isinstance(self.path, (list, set)):

            self.path = [self.path]

        self.meta_path = meta_path
```

```
        if meta_path is not None:
            self.meta = pd.read_csv(meta_path, index_col=0)
            self.meta = self.meta.astype({lat_key: "float32", lon_key:
"float32"})
self.lat_key = lat_key self.lon_key =
lon_key

        self.shards = self.  init_shards(self.path) self.length =
        len(self.target_mapping)

    @staticmethod
    def   init_shards(path: Union[str, Path]) -> list: shards = []
        for i, p in enumerate(path):
            shards_re = r"shard_(\d+).msg"
            shards_index = [
                int(re.match(shards_re,x).group(1)) for x
                in os.listdir(p)
                if re.match(shards_re, x)
            ]
            shards.extend(
                [
                    {
                        "path_index": i,
                        "path": p, "shard_index":
                        s,
                        "shard_path": os.path.join(p, f"shard_{s}.msg"),
                    }
                    for s in shards_index
                ]
            )
        if len(shards) == 0:
            raise ValueError("No shards found")
```

```
        return shards

    def _process_sample(self, x):
        # prepare image and target value

        # decode and initial resize if necessary
        img = Image.open(BytesIO(x[self.key_img_encoded]))
        if img.mode != "RGB":
            img = img.convert("RGB")

        if img.width > 320 and img.height > 320:
            img = torchvision.transforms.Resize(320)(img)

        # apply all user specified image transformations img =
        self.transformation(img)
        if self.meta_path is None: return
            img, x["target"]
        else:
            _id = x[self.key_img_id].decode("utf-8") meta
            = self.meta.loc[_id]
            return img, x["target"], meta[self.lat_key], meta[self.lon_key]

    def __iter__(self):

        shard_indices = list(range(len(self.shards))) if
        self.shuffle:

            random.seed(self.seed)
            random.shuffle(shard_indices)

        worker_info = torch.utils.data.get_worker_info() if
        worker_info is not None:
```

```
            def split_list(alist, splits=1): length =
                len(alist)
                return [
                    alist[i * length // splits : (i + 1) * length // splits] for i in
                    range(splits)
                ]

            shard_indices_split = split_list(shard_indices,
worker_info.num_workers)[
                worker_info.id
            ]

        else:
            shard_indices_split = shard_indices

        cache = []

        for shard_index in shard_indices_split: shard =
            self.shards[shard_index]

            with open(
                os.path.join(shard["path"],
f"shard_{shard['shard_index']}.msg"),"rb"
            ) as f:
                        unpacker =
                    msgpack.Unpacker(
                    f, max_buffer_size=1024 * 1024 * 1024, raw=True
                )
                for x in unpacker: if
                    x is None:
                        continue
```

```
                    # valid dataset sample?
                    _id = x[self.key_img_id].decode("utf-8") try:
                        # set target value dynamically
                        if len(self.target_mapping[_id]) == 1: x["target"] =
                            self.target_mapping[_id][0]
                        else:
                            x["target"] = self.target_mapping[_id]
                    except KeyError:
                        # reject sample
                        # print(f'reject {_id} {type(_id)}')
                        continue

                    if len(cache) < self.cache_size: cache.append(x)

                    if len(cache) == self.cache_size:
            if self.shuffle:
            if self.shuffle: random.shuffle(cache)
while cache:
    yield self._process_sample(cache.pop())
                random.shuffle(cache)

            while cache:
                yield self._process_sample(cache.pop())

        def   len   (self): return
            self.length


 class FiveCropImageDataset(torch.utils.data.Dataset): def
        init   (
            self,
            meta_csv: Union[str, Path, None],
```

```
        image_dir: Union[str, Path], img_id_col:
        Union[str, int] = "img_id",
        allowed_extensions: List[str] = ["jpg", "jpeg", "png"]
    ):
        if isinstance(image_dir, str):
            image_dir = Path(image_dir)
        self.image_dir = image_dir self.img_id_col
        = img_id_col self.meta_info = None
        if meta_csv is not None:
            print(f"Read {meta_csv}")
            self.meta_info = pd.read_csv(meta_csv)
            self.meta_info.columns = map(str.lower, self.meta_info.columns)
            # rename column names if necessary to use existing data if
            "lat" in self.meta_info.columns:
                self.meta_info.rename(columns={"lat": "latitude"}, inplace=True)
            if "lon" in self.meta_info.columns:
                self.meta_info.rename(columns={"lon":"longitude"},
inplace=True)
            self.meta_info["img_path"] = self.meta_info[img_id_col].apply(
                lambda img_id: str(self.image_dir / img_id)
            )
        else:
            image_files = []
            for ext in allowed_extensions:
                image_files.extend([str(p) for p in
self.image_dir.glob(f"**/*.{ext}")])
            self.meta_info = pd.DataFrame(image_files, columns=["img_path"])
            self.meta_info[self.img_id_col] = self.meta_info["img_path"].apply(
                lambda x: Path(x).stem
            )
        self.tfm = torchvision.transforms.Compose( [
                torchvision.transforms.ToTensor(), torchvision.transforms.Normalize(
```

```
                    (0.485, 0.456, 0.406), (0.229, 0.224, 0.225)
                ),
            ]
        )

    def __len__(self):
        return len(self.meta_info.index)

    def __getitem__(self, idx) -> Tuple[torch.Tensor, dict]: meta =
        self.meta_info.iloc[idx]
        meta = meta.to_dict()
        meta["img_id"] = meta[self.img_id_col]

        image = Image.open(meta["img_path"]).convert("RGB")
        image = torchvision.transforms.Resize(256)(image)
        crops = torchvision.transforms.FiveCrop(224)(image)
        crops_transformed =
```