МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії

програмного забезпечення

_____Євген ДАВИДЕНКО

«___»_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

РОЗРОБКА МЕТОДУ ОЦІНКИ ГЛИБИНИ ДВОВИМІРНОГО ЗОБРАЖЕННЯ З ВИКОРИСТАННЯМ КОМП'ЮТЕРНОГО ЗОРУ

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувач _____Артем ТРИБУШЕНКО

«___»_____2024 р.

Керівник канд. пед. наук, доцент _____Катерина КІРЕЙ

«___»_____2024 р.

Миколаїв – 2024

Чорноморський національний університет імені Петра Могили

(повне найменування закладу вищої освіти)

Факультет	Комп’ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії
програмного забезпечення
_____Євген ДАВИДЕНКО
«___»_____2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу здобувача

_____Трибушенка Артема Сергійовича_____

(прізвище, ім’я, по батькові здобувача)

1. Тема кваліфікаційної роботи

_____Розробка методу оцінки глибини за двовимірним зображенням з використанням комп'ютерного зору_____

Затверджена наказом ЧНУ ім. Петра Могили від «___»_____2024 р.
№_____

2. Строк представлення кваліфікаційної роботи «___ »_____2024 р.
3. Очікуваний результат роботи та початкові дані якщо такі потрібні

4. Перелік питань, що підлягають розробці:
5. Перелік графічних матеріалів

6. Завдання до спеціальної частини

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи

Керівник роботи _____ ____Катерина КІРЕЙ____
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Здобувач _____ Артем_ТРИБУШЕНКО
Особистий підпис *Власне ім'я ПРІЗВИЩЕ*

Дата видачі завдання « ____»_____20_____

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: Розробка методу оцінки глибини двовимірного зображення з використанням комп'ютерного зору

№ з/п	Завдання	Строк виконання завдання	Примітка
1	Провести огляд предметної області кваліфікаційної роботи	05.09.2024	виконано
2	Огляд архітектури моделей Depth Anything V1 та V2, аналіз варіантів моделей (ViTS, ViTB, ViTL).	08.09.2024	виконано
3	Збір та систематизація даних для моделей Depth Anything V1 та V2, завантаження зібраних даних на платформу Kaggle	11.09.2024	виконано
4	Візуалізація роботи моделей для різних варіантів зображень	12.09.2024	виконано
5	Налаштування оточення та залежностей для бенчмаркінгу моделей	14.09.2024	виконано
6	Проведення перших тестів на кластерах з 50 Гб відеопам'яттю	18.09.2024	виконано
7	Збір та аналіз даних про швидкість, затримку та пропускну здатність моделей	20.09.2024	виконано
8	Реалізація та проведення повного бенчмарку для оцінки якості моделей DA-2K	25.09.2024	виконано
9	Аналіз продуктивності та слабких місць моделей	30.09.2024	виконано
10	Підготовка звіту та формулювання висновків дослідження	06.10.2024	виконано

Розробив студент _____ТРИБУШЕНКО Артем Сергійович_____ _____

(прізвище, ім’я, по батькові) *(підпис)*

«__» _____ 20__ р.

Керівник роботи ____КІРЕЙ Катерина Олександрівна_____ _____

(посада, прізвище, ім’я, по батькові) *(підпис)*

«__» _____ 20__ р.

АНОТАЦІЯ

до кваліфікаційної роботи магістра

«Розробка методу оцінки глибини двовимірного зображення з використанням комп’ютерного зору»

Здобувач 608м гр.: Трибушенко Артем Сергійович

Керівник: доцент, канд. пед. наук Кірей Катерина Олкесандрівна

Мета цієї кваліфікаційної роботи полягає у розробці вебдодатку для оцінювання глибини зображень останньою версією моделі глибокого навчання сімейства Depth Anything, інтегрованого з великою мовно-візуальною моделлю SpatialBot [23] (LVLM) для забезпечення функціональності "DepthGPT".

Основні завдання включають створення бекенд-сервісу на базі FastAPI, інтегрованим з моделями Depth Anything, яка генерує карту глибини двовимірного зображення, та SpatialBot [23], який відповідає за аналіз зображень та генерацію текстових відповідей на основі отриманих глибинних мап, розробку фронтенд-інтерфейсу з використанням сучасних вебтехнологій бібліотеки React.

Об'єктом дослідження є системи машинного навчання та вебтехнології.

Предметом – інтеграція комп'ютерного зору з обробкою природної мови для створення інтерактивних додатків.

Основні результати роботи включають успішну реалізацію двох основних компонентів системи: бекенд-сервісу, фронтенд-інтерфейсу.

Для забезпечення ефективного розгортання та масштабованості застосовано контейнеризацію за допомогою Docker та Docker Compose, що дозволяє легко управляти залежностями та середовищем виконання кожного сервісу. Особлива увага приділена інтеграції GPU для прискорення обчислень моделі.

Проведені тести підтвердили ефективність системи у завданні оцінювання глибини зображень та взаємодії з користувачем через чат-інтерфейс, демонструючи високий рівень точності та швидкодії.

Робота складається з 143 сторінок, містить 24 рисунки, 6 таблиць та 30 джерел посилання.

Ключові слова: вебдодаток, оцінювання глибини зображень, велика мовно-візуальна модель, інтеграція комп'ютерного зору, обробка природної мови, Docker, FastAPI, GPU-акселерація.

ABSTRACT

of the Master's Thesis

"Development of a method for estimating the depth of a two-dimensional image using computer vision"

Student of group 608m: Trybushenko Artem Serthiyovych

Supervisor: Docent, Cand. of Pedag. Sciences Kirei Kateryna Oleksandrivna

The aim of this qualification work is to develop a web application for image depth estimation using the latest version of the Depth Anything deep learning model, integrated with the Large Language and Vision Model (LVLM) SpatialBot to provide the "DepthGPT" functionality.

The primary tasks include creating a backend service based on FastAPI, integrated with Depth Anything models that generate depth maps of two-dimensional images, and SpatialBot, which is responsible for image analysis and generating textual responses based on the obtained depth maps. Additionally, it involves developing a frontend interface using modern web technologies from the React library.

The object of study encompasses machine learning systems and web technologies.

The subject focuses on the integration of computer vision with natural language processing to create interactive applications.

The main outcomes of the work include the successful implementation of two core components of the system: the backend service and the frontend interface. To ensure efficient deployment and scalability, containerization

using Docker and Docker Compose was employed, allowing easy management of dependencies and runtime environments for each service. Special attention was given to GPU integration to accelerate model computations. Conducted tests confirmed the system's effectiveness in image depth estimation tasks and user interaction through the chat interface, demonstrating a high level of accuracy and performance.

The work consists of 143 pages, contains 24 figures, 6 tables and 30 references.

Keywords: web application, image depth estimation, Large Language and Vision Model, computer vision integration, natural language processing, Docker, FastAPI, GPU acceleration.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API	–	Application Programming Interface
CNN	–	Convolutional Neural Network
DFD	–	Data Flow Diagram
GPU	–	Graphics Processing Unit
IDEF0	–	Integration Definition for Function Modeling
LVLM	–	Large Vision-Language Model
MAE	–	Mean Absolute Error
REST	–	Representational State Transfer
RMSE	–	Root Mean Square Error
RGB	–	Red Green Blue
UML	–	Unified Modeling Language
ViTB	–	Vision Transformer Base
ViTL	–	Vision Transformer Large
ViTS	–	Vision Transformer Small

ВСТУП

Актуальність даної кваліфікаційної роботи зумовлена зростаючою потребою в інтеграції комп'ютерного зору з обробкою природної мови для створення інтелектуальних систем, здатних аналізувати просторову структуру зображень та надавати інтуїтивно зрозумілі відповіді користувачам. Сучасний розвиток технологій штучного інтелекту створює нові можливості для розробки систем, що поєднують точне оцінювання глибини зображень із природномовною взаємодією, що має критичне значення для різноманітних галузей застосування, включаючи медичну діагностику, системи безпеки, освітні технології та розважальні додатки.

Метою роботи є підвищення ефективності аналізу просторової структури двовимірних зображень через розробку системи DepthGPT, що інтегрує технології глибокого навчання сімейства Depth Anything з великою мовно-візуальною моделлю SpatialBot.

Для досягнення визначеної мети необхідно вирішити такі завдання:

a) провести аналіз існуючих методів оцінки глибини двовимірних зображень та обробки природної мови для визначення оптимальних підходів до розробки системи;

b) розробити бекенд-сервіс на базі FastAPI, інтегрований з моделями Depth Anything для генерації глибинних мап двовимірних зображень;

c) інтегрувати LVLM-модель SpatialBot для аналізу зображень

та генерації текстових відповідей на основі отриманих глибинних мап;

d) створити фронтенд-інтерфейс з використанням сучасних вебтехнологій бібліотеки React для забезпечення зручності користувача;

e) забезпечити ефективне розгортання та масштабованість системи за допомогою контейнеризації з використанням Docker та Docker Compose;

f) інтегрувати GPU для прискорення обчислень моделі та підвищення продуктивності системи;

g) провести тестування системи для підтвердження її ефективності у завданні оцінювання глибини зображень та взаємодії з користувачем через чат-інтерфейс.

Об'єктом дослідження є процеси аналізу та обробки двовимірних зображень з використанням систем машинного навчання та вебтехнологій.

Предметом дослідження є методи та засоби створення системи оцінювання глибини зображень з інтеграцією технологій комп'ютерного зору та обробки природної мови.

Обґрунтування необхідності нової розробки полягає у відсутності комплексних рішень, що ефективно поєднують точне оцінювання глибини зображень із природномовною взаємодією в єдиному вебдодатку. Аналіз вітчизняної та зарубіжної науково-технічної літератури свідчить про наявність окремих систем для оцінювання глибини зображень та для обробки природної мови, проте їх інтеграція

відкриває нові можливості для створення більш інтуїтивних та ефективних інтерфейсів користувача. Провідні дослідницькі групи, такі як команди Depth Anything та SpatialBot, активно працюють над вдосконаленням відповідних моделей, що підтверджує актуальність обраного напрямку досліджень.

Основні проєктні рішення включають використання сучасних фреймворків FastAPI та React, що забезпечують гнучкість та швидкість розробки, контейнеризацію з Docker для ефективного управління залежностями та середовищем виконання, а також інтеграцію GPU для забезпечення необхідної обчислювальної потужності при обробці складних моделей глибокого навчання.

Сфера застосування результатів роботи охоплює широкий спектр галузей, включаючи:

a) Медицину (аналіз медичних зображень та діагностика);

b) Системи безпеки (моніторинг та аналіз відеоданих)

c) Освіту (створення інтерактивних навчальних матеріалів)

d) Розважальну індустрію (розробка інтерактивних ігор та візуальних ефектів)

Апробація результатів КМР відбулась під час XXVII Всеукраїнської науково-практичної конференції «Могилянські читання - 2024», Миколаїв, 06-10 листопада 2024 р. (Додаток Е).

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

У даному розділі проведено аналіз теоретичних основ оцінки глибини двовимірних зображень та інтеграції комп'ютерного зору з обробкою природної мови. Розглянуто базові принципи та методи машинного навчання в задачах комп'ютерного зору, проаналізовано архітектури нейронних мереж та методи обробки цифрових зображень.

Особлива увага приділена сучасному стану технологій у сфері оцінки глибини зображень, включаючи як традиційні підходи, так і методи глибокого навчання. Проведено порівняльний аналіз існуючих рішень, зокрема моделей сімейства Depth Anything та інших сучасних підходів до оцінки глибини зображень.

У розділі також розглянуто розвиток великих мовно-візуальних моделей (LVLM) та їх застосування для аналізу просторових відношень у зображеннях. На основі проведеного аналізу сформульовано детальні вимоги до програмного забезпечення, що визначають функціональність, архітектуру та технічні характеристики розроблюваної системи.

1.1 Теоретичні основи машинного навчання та комп'ютерного зору

1.1.1 Принципи та методи машинного навчання в задачах комп'ютерного зору

Машинне навчання — це застосування штучного інтелекту (ШІ), яке надає системам можливість автоматично навчатися та вдосконалюватися на основі досвіду без явного програмування [1].

В контексті оцінки глибини зображень виділяють три основні типи машинного навчання: контрольоване навчання, де модель тренується на розмічених даних; навчання без нагляду для виявлення прихованих закономірностей; та навчання з підкріпленням, що базується на взаємодії з середовищем [1].

Рисунок 1.1 – візуалізація типів машинного навчання в залежності від кількості розмітки

Комп'ютерний зір є важливою складовою розвитку штучного інтелекту та охоплює широкий спектр задач, включаючи сегментацію, фільтрацію, класифікацію та реконструкцію сцени [2]. В контексті оцінки глибини зображень особливо важливими є задачі сегментації та обробки контурів, що дозволяють визначати просторові характеристики об'єктів.

Основними завданнями комп'ютерного зору в контексті глибинного аналізу є:

а) сегментація зображень - виділення значущих сегментів для покращення виявлення об'єктів та розуміння просторових відношень [2];

b) ідентифікація контурів та кутів - визначення ключових елементів форми об'єктів через відстеження змін інтенсивності між сусідніми пікселями [2];

c) класифікація зображень - присвоєння об'єктам міток певних класів для подальшого аналізу їх просторового розташування [2].

Архітектури нейронних мереж для оцінки глибини базуються на взаємопов'язаних шарах нейронів, що обробляють візуальну інформацію [4]. Ключовими компонентами є вхідний шар для отримання даних, приховані шари для їх обробки, та вихідний шар для генерації результатів. Особливу роль відіграють згорткові нейронні мережі (CNN), що ефективно обробляють просторові дані [4].

Такий аналіз теоретичних основ створює фундамент для розробки системи оцінки глибини зображень з використанням сучасних методів машинного навчання та комп'ютерного зору.

1.1.2 Архітектура та особливості глибоких нейронних мереж

Архітектури нейронних мереж - це будівельні блоки моделей глибокого навчання. Вони складаються з взаємопов'язаних вузлів, які називаються нейронами і організовані пошарово. Кожен нейрон отримує вхідні дані, обчислює математичні операції і видає вихідні дані [4].

Архітектури нейронних мереж складаються з кількох компонентів, які спільно обробляють дані та вивчають їх. Основними компонентами архітектури нейронної мережі є:

a) вхідний рівень – початковий ріввень нейронної мережі та відповідає за отримання вхідних даних. Кожен нейрон у вхідному шарі представляє функцію або атрибут вхідних даних [4];

b) приховані шари - це проміжні шари між вхідними та вихідними шарами. Вони виконують обчислення і перетворюють вхідні дані за допомогою серії зважених зв'язків. Кількість прихованих шарів і кількість нейронів у кожному шарі може змінюватися залежно від складності завдання та кількості доступних даних [4];

c) нейрони (вузли) - окремі обчислювальні одиниці в нейронній мережі. Кожен нейрон отримує вхідні дані від попереднього шару або безпосередньо від вхідного шару, виконує обчислення з використанням ваг і зсувів і виробляє вихідне значення за допомогою функції активації [4];

d) ваги та зсуви - це параметри, пов'язані зі зв'язками між нейронами. Ваги визначають силу або важливість зв'язків, в той час як зсуви вводять константу, яка допомагає контролювати активацію нейронів. Ці параметри налаштовуються в процесі навчання для оптимізації роботи мережі [4];

e) функції активації - це спеціальні математичні формули, що забезпечують нелінійну поведінку нейронної мережі для вивчення складних патернів. Основні типи включають сигмоїдну функцію, ReLU та гіперболічний тангенс (tanh). Нейрони застосовують ці функції до зваженої суми входів для генерації виходу, що дозволяє мережі

ефективно обробляти дані та виявляти складні взаємозв'язки для точного розпізнавання образів та прогнозування. [4];

f) вихідний шар - це кінцевий шар нейронної мережі, що генерує результати обробки вхідних даних. Його структура визначається типом завдання: для бінарної класифікації (так/ні) використовується один нейрон, тоді як для багатокласової класифікації (наприклад, розпізнавання різних об'єктів) застосовується декілька нейронів [4];

g) функція втрат – це функція, вимірює розбіжність між прогнозованим і фактичним виходом нейронної мережі, кількісно оцінюючи її роботу під час навчання та спрямовуючи коригування параметрів. Для прогнозування числових значень використовується функція середньоквадратичної помилки, яка обчислює усереднену квадратичну різницю між передбаченими та істинними значеннями. У задачах класифікації застосовується перехресна ентропія, що оцінює різницю між прогнозованими ймовірностями та істинними мітками даних, допомагаючи мережі покращувати точність класифікації.[4].

Ці компоненти працюють разом для обробки вхідних даних, поширення інформації через мережу та отримання бажаного результату. Ваги та зсуви коригуються під час навчання за допомогою алгоритмів оптимізації, щоб мінімізувати функцію втрат та покращити продуктивність мережі.

Виділяють наступні типи нейронних мереж:

a) нейронні мережі прямого поширення (FNN) - це найфундаментальніший тип нейронної мережі, в якій інформація рухається в одному напрямку, від вхідного шару до вихідного. ШНМ використовуються для таких завдань, як класифікація, регресія та розпізнавання образів [4];

b) згорткові нейронні мережі (CNN) – даний тип мережі особливо ефективний для обробки сітчастих даних, таких як зображення та відео. Вони використовують згорткові шари для фіксації просторових взаємозв'язків та ідентифікації таких особливостей, як краї, текстури та форми в даних, для вилучення локальних шаблонів та ієрархічних представлень. Це допомагає в таких завданнях, як класифікація зображень, виявлення об'єктів і сегментація зображень [4];

c) рекурентні нейронні мережі (RNN) – цей тип призначений для обробки послідовних даних, де порядок вхідних даних має значення. Вони використовують рекурентні зв'язки, які дозволяють інформації зберігатися, що робить їх придатними для таких завдань, як розпізнавання мови, переклад і генерація тексту [4];

d) мережі з довгою короткочасною пам'яттю (LSTM) - це тип мережі, який вирішує проблему зникаючого градієнта - явище, коли градієнти, що використовуються для оновлення ваг і зсувів мережі, стають надзвичайно малими, що ускладнює навчання мережі на основі інформації з далекого минулого в послідовності. LSTM мають комірки

пам'яті та механізми гейтінгу, які допомагають зберігати та отримувати інформацію протягом тривалих часових інтервалів і дозволяють фіксувати довгострокові залежності в послідовних даних [4].

1.1.3 Методи обробки та аналізу цифрових зображень

Цифрова обробка зображень - це клас методів, які займаються маніпулюванням цифровими зображеннями за допомогою комп'ютерних алгоритмів. Це важливий етап попередньої обробки в багатьох додатках, таких як розпізнавання облич, виявлення об'єктів і стиснення зображень.

Обробка зображень виконується для покращення існуючого зображення або для відсіювання важливої інформації з нього. Це важливо в деяких додатках комп'ютерного зору, що базуються на глибокому навчанні, де така попередня обробка може значно підвищити продуктивність моделі. Маніпулювання зображеннями, наприклад, додавання або видалення об'єктів, є ще одним застосуванням, особливо в індустрії розваг.

Виділяють наступні етапи обробки зображення:

a) отримання зображень – це метод, коли зображення фіксується камерою і оцифровується (якщо вихід камери не оцифровується автоматично) за допомогою аналого-цифрового перетворювача для подальшої обробки на комп'ютері [5];

b) поліпшення зображення - на цьому етапі відбувається обробка отриманого зображення відповідно до вимог конкретного

завдання, для якого воно буде використовуватися. Такі методи спрямовані насамперед на виділення прихованих або важливих деталей на зображенні, наприклад, регулювання контрастності, яскравості тощо. Покращення зображення є дуже суб'єктивним за своєю природою [5];

c) відновлення зображення - цей крок стосується покращення зовнішнього вигляду зображення та є об’єктивною операцією, оскільки погіршення якості зображення можна віднести до математичної або ймовірнісної моделі. Наприклад, видалення шумів або розмитості зображень [5];

d) обробка кольорових зображень - цей крок спрямований на обробку кольорових зображень (16-бітних зображень RGB або RGBA), наприклад, на виконання корекції кольорів або моделювання кольорів у зображеннях [5];

e) морфологічна обробка – виділення компонент зображення, які є корисними для представлення та опису форми, потрібно виділити для подальшої обробки або наступних завдань. Морфологічна обробка надає інструменти (які по суті є математичними операціями) для цього. Наприклад, операції ерозії та дилатації використовуються для загострення та розмиття країв об'єктів на зображенні відповідно [5];

f) база знань – це виділення знань, які можуть бути такими ж простими, як координати обмежувальної рамки для цікавого об’єкта, знайденого на зображенні, разом із присвоєною йому міткою об’єкта. У

базу знань можна закодувати все, що допоможе розв’язати задачу для конкретного завдання [5].

Епоха інформаційних технологій, в яку ми живемо, зробила візуальні дані широко доступними. Однак, для того, щоб передати їх через Інтернет або для таких цілей, як вилучення інформації, прогнозування, моделювання тощо, потрібна значна обробка.

Розвиток технології глибокого навчання призвів до появи моделей CNN, які були спеціально розроблені для обробки зображень. Відтоді було розроблено кілька вдосконалених моделей, які вирішують конкретні завдання в ніші обробки зображень.

1.2 Сучасний стан технологій у сфері оцінювання глибини зображень та обробки природної мови

Оцінка глибини монокуляра — це завдання оцінки значення глибини (відстані відносно камери) кожного пікселя на одному (монокулярному) зображенні RGB. Це складне завдання є ключовою передумовою для визначення розуміння сцени для таких програм, як 3D-реконструкція сцени, автономне водіння та AR [6].

Сучасні методи зазвичай діляться на одну з двох категорій: розробка складної мережі, достатньо потужної для прямої регресії карти глибини, або поділ вхідних даних на контейнери чи вікна для зменшення складності обчислень. Найпопулярнішими тестами є набори даних KITTI [7] та NYUv2 [8]. Моделі зазвичай оцінюються за допомогою RMSE або абсолютної відносної похибки.

1.2.1 Неглибокі методи оцінки глибини (Non-Deep Learning Methods)

Неглибокі методи оцінки глибини базуються на імітації механізмів людського зору (оклюзія, перспектива, відносні розміри об'єктів), де основою виступає триангуляція на базі стереозображень [9].

Стереоскопічні методи створюють карту диспаратності через зіставлення зображень з двох камер, хоча виникають складнощі з точним визначенням відповідних пікселів. Structure from Motion пропонує універсальніше рішення, створюючи тривимірні моделі з множини зображень без попередньої калібровки камер [9].

Моноскопічні методи працюють з одним зображенням, використовуючи припущення про структуру сцени та пошук візуально подібних елементів. Хоча ці методи мають обмеження при роботі зі складними сценами, вони заклали основу для розвитку сучасних підходів тривимірного сприйняття зображень [9].

1.2.2 Глибокі методи оцінки глибини (Deep Learning Methods)

Моделі глибокого навчання для визначення глибини сцени відрізняються архітектурою, кількістю вхідних зображень та стратегією навчання, при цьому останні два параметри мають найбільший вплив на якість оцінки [10].

За стратегією навчання розрізняють три категорії методів: з "вчителем" (потребують розмічених карт глибини), без "вчителя" (використовують лише RGB-зображення) та напівконтрольовані, що

поєднують обидва підходи [10]. Хоча методи з "вчителем" демонструють найвищу точність, їх застосування обмежене через потребу у значних наборах розмічених даних [10].

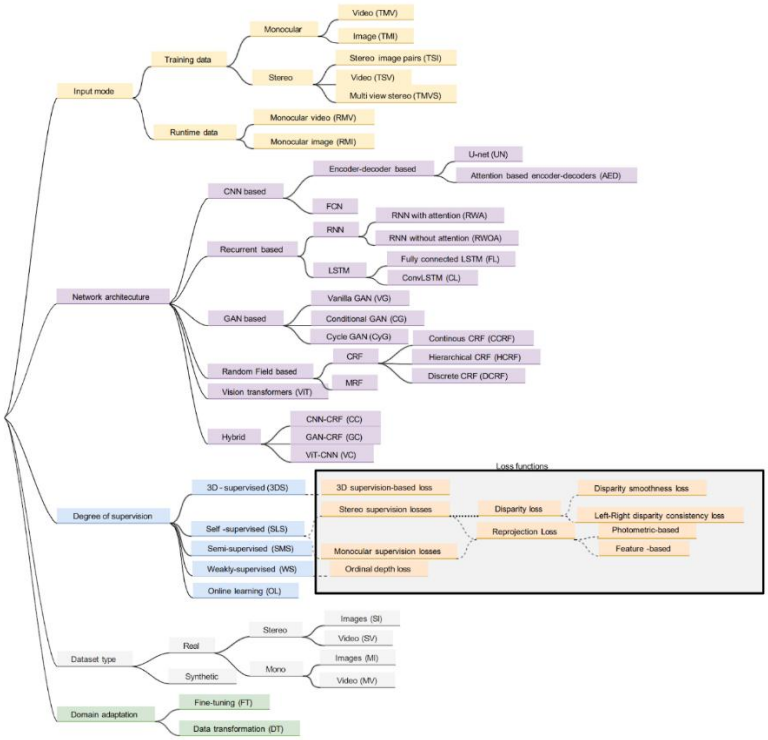


Рисунок 1.4 – Таксономія методів глибокого навчання для оцінки глибини зображення. [10]

Методи глибокого навчання демонструють значний прогрес у точності оцінки глибини. Особливу роль відіграють три основні підходи:

а) методи на основі випадкових полів враховують як локальні особливості пікселів, так і глобальні взаємозв'язки між областями зображення [10, 11];

b) Генеративні змагальні мережі (GAN) забезпечують створення реалістичних карт глибини через змагання між генератором та дискримінатором [10, 12].

c) Моделі на основі Vision Transformer (ViT) вирішують проблему моделювання довгострокових залежностей завдяки механізмам уваги [13-18]. Ця архітектура забезпечує краще збереження дрібних деталей через глобальне рецептивне поле [19, 20].

1.2.4 Розвиток великих мовно-візуальних моделей (LVLM)

Великі мультимодальні мовні моделі (VLMs/MLLMs) досягли значного прогресу в задачах комп'ютерного зору та обробки природної мови, відкриваючи нові можливості для систем оцінки глибини зображень.

LLaVA [24] започаткувала візуальне навчання за інструкціями, а подальші розробки (GPT-4o, PaLM, Claude, GPT-4) розширили можливості завдяки масштабнішим наборам даних та вдосконаленим архітектурам. Основними напрямками застосування стали візуальне сприйняття, логічне міркування та OCR, з особливим успіхом у попіксельній локалізації об'єктів [23].

Проте сучасні VLMs демонструють обмеження в підрахунку об'єктів та розумінні просторових відношень, особливо при інтерпретації тривимірного простору з монокулярних RGB-зображень. Дослідження показують, що інтеграція інформації про глибину може значно покращити просторове розуміння.

SpatialBot [23] пропонує інноваційне рішення через інтеграцію RGB-зображень та карт глибини, на відміну від традиційних VLM. Модель використовує спеціалізовані набори даних SpatialQA та SpatialQA-E для навчання, охоплюючи різні сценарії від візуальних запитань до роботизованих маніпуляцій.

Ефективність моделі оцінюється через тестовий комплекс SpatialBench, а гнучка архітектура забезпечує роботу з різними типами вхідних даних та інтеграцію з Depth API. Практичні результати демонструють значний прогрес у просторовому розумінні та візуальному розпізнаванні.

Впровадження SpatialBot [23] відкриває нові можливості для розвитку систем комп'ютерного зору та штучного інтелекту, особливо в задачах, що вимагають глибокого розуміння просторової динаміки.Порівняння роботи SpatialBot [23] та GPT-4o моделей на розуміння просторових даних та локації об’єктів на зображенні.

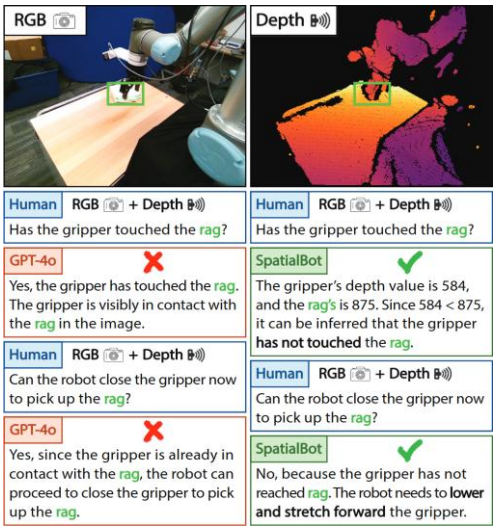


Рисунок 1.5 – Порівняння роботи моделі SpatialBot [23] та GPT-4o на просторових даних. SpatialBot [23] має кращу здатність просторового розуміння, ніж GPT-4o. З одного зображення RGB навіть людина не може судити
чи зробив робот захват ганчірки.

Також були завантажені ваги моделі на платформу Kaggle для зручного їх зчитування та швидкого доступу. Depth Anything V1 [21] та V2 [22] знаходяться на платформі Kaggle у відкритому доступі для всіх бажаючих приєднатися до вирішення задач з оцінки глибини двовимірного зображення.

1.3 Аналіз наявних програмних рішень

1.3.1 Огляд існуючих систем оцінки глибини зображень

Останні роки стали найуспішнішими для задачі оцінювання глибини двовимірного зображення завдяки стрімкому розвитку нейронних мереж та зниження вартості обчислювальних процесорів, на яких ці мережі

тренуються. Найуспішніші моделі для оцінки глибини, які зробили прорив у вирішенні цієї задачі:

a) Dinov2 [26];
b) DPT [29];
c) Depth-Anything V1 [21];
d) Depth-Anything v2 [22].

Невеликий огляд кожного з методів:

a) Dinov2 (2024) впровадив метод самоконтрольованого навчання, що використовує геометричну узгодженість монокулярних відеопослідовностей для усунення потреби в еталонних картах глибини, однак демонструє нижчу продуктивність порівняно з методами навчання під наглядом та потребує спеціальних механізмів самоконтролю [26];
b) DPT (2020) використовує трансформерну архітектуру для забезпечення точної оцінки глибини через ефективне моделювання взаємозв'язків між віддаленими областями зображення, хоча характеризується підвищеними обчислювальними витратами порівняно з CNN-архітектурами [29];
c) Depth-Anything (2021) впроваджує комплексну уніфіковану архітектуру з високою адаптивністю до різних завдань та наборів даних, проте потребує додаткового налаштування для оптимальної роботи зі спеціалізованими завданнями

порівняно з моделями, що оптимізовані під конкретні домени [21];

d) Depth-Anything v2 представляє вишуканий підхід до оцінки глибини, демонструючи виняткову універсальність у різноманітних сферах застосування від автономного керування до доповненої реальності, із стабільно високими показниками точності та оптимізованою архітектурою для роботи в реальному часі; проте система має суттєві обмеження, включаючи високі вимоги до обчислювальних ресурсів, знижену точність при обробці складних сцен з оклюзіями, та значну залежність від масштабних наборів даних конкретного домену для досягнення оптимальних результатів [22].

Мотиваційними статтями для написання даної роботи стали технічні роботи, в яких представлено Depth Anything V1 [21] та V2 модель [22]. Задля їх оцінювання і створення нового рішення на базі цієї моделі було проведено ряд досліджень можливостей цієї глибинної моделі оцінки глибини: швидкість, пропускна здатність, затримка під час інференсу та впровадження новітнього фреймворку, який був запропонований у цій статті [22], DA-2K для оцінення попіксельної здатності моделі.

1.4 Специфікація вимог до програмного забезпечення

1.4.1 Призначення та межі проєкту

Призначенням системи є створення вебдодатку для оцінювання глибини зображень з використанням останньої версії моделі глибокого навчання сімейства Depth Anything, інтегрованої з великою мовно-візуальною моделлю SpatialBot [23] (LVLM) для забезпечення функціональності "DepthGPT". Цей додаток дозволяє користувачам завантажувати зображення, отримувати глибинні мапи та отримувати текстові відповіді на основі аналізу зображень, сприяючи покращенню взаємодії між людиною та комп’ютером у різних сферах застосування.

Було погоджено, що для розробки вебдодатку будуть використані наступні технології та інструменти:

a) бекенд: FastAPI для створення API-сервісів;
b) фронтенд: React для розробки користувацького інтерфейсу;
c) моделі глибокого навчання: Depth Anything для генерації глибинних мап та SpatialBot [23] для інтеграції з LVLM;
d) контейнеризація: Docker та Docker Compose для забезпечення масштабованості та ефективного розгортання системи;
e) GPU-акселерація: Для прискорення обчислень моделей глибокого навчання.

Крайня дата завершення роботи над програмним забезпеченням – 30.11.2024 року. Проєкт охоплює розробку та впровадження вебдодатку

з функціональністю оцінювання глибини зображень та інтеграцією з LVLM, але не включає розробку мобільних додатків або офлайн-версій системи.

1.4.2 Загальний опис

Цей вебдодаток призначений для професіоналів та ентузіастів у сферах медицини, безпеки, розваг та освіти, які потребують точного оцінювання глибини зображень. Додаток дозволяє проводити аналіз зображень, отримувати глибинні мапи та отримувати текстові відповіді, що можуть використовуватися для подальшої обробки, візуалізації або прийняття рішень.

Основні характеристики користувачів:

a) професіонали у сфері медицини та безпеки: Потребують точних глибинних мап для аналізу медичних зображень або відеоспостереження;

b) освітяни та дослідники: Використовують додаток для навчання та проведення досліджень у галузі комп'ютерного зору та обробки природної мови;

c) технологічні ентузіасти: Зацікавлені у використанні сучасних технологій для вирішення різноманітних задач.

Система складається з трьох основних компонентів:

a) бекенд-сервіс (Backend Service): Обробляє запити користувачів, інтегрується з моделями глибокого навчання та LVLM, управляє даними;

b) фронтенд-інтерфейс (Frontend Interface): Забезпечує користувачу зручний інтерфейс для взаємодії з системою;

c) LVLM-сервіс (LVLM Service): Аналізує зображення та генерує текстові відповіді на основі отриманих глибинних мап.

Загальні обмеження:

a) технічні обмеження: Необхідність наявності GPU для обробки моделей глибокого навчання;

b) функціональні обмеження: Додаток призначений лише для роботи з двовимірними зображеннями у форматах JPEG та PNG;

c) місцеві обмеження: Система розроблена для роботи в онлайн-середовищі та не підтримує офлайн-функціональність.

1.4.3 Функції системи

Опис функціоналу системи.

a) Функція обробки зображень; Опис функції**:** функція обробки зображень забезпечує завантаження користувачем зображень, їх аналіз та генерацію глибинних мап. Ця функція є основною для оцінювання глибини та подальшої інтеграції з LVLM-сервісом.
b) Функція генерації глибинних мап; Опис функції: ця функція відповідає за використання моделей глибокого навчання сімейства Depth Anything для створення глибинних мап з двовимірних зображень. Глибинні мапи використовуються для подальшого аналізу та генерації текстових відповідей.

c) Функція взаємодії з LVLM-сервісом; Опис функції: функція забезпечує інтеграцію з LVLM-сервісом SpatialBot [23], який аналізує отримані глибинні мапи та генерує текстові відповіді на основі введених користувачем запитів.

d) Функція відображення результатів; Опис функції: функція відповідає за візуалізацію глибинних мап та текстових відповідей у фронтенд-інтерфейсі, забезпечуючи зручне та інтуїтивне представлення результатів для користувача.

1.4.4 Вимоги до інформаційного забезпечення

Основними джерелами вхідної інформації є:

a) Користувач: Завантажує зображення для аналізу та вводить текстові запити;

b) Моделі глибокого навчання: Генерують глибинні мапи з завантажених зображень;

c) LVLM-сервіс: Отримує глибинні мапи та текстові запити для генерації відповідей.

Вимоги до способів організації, збереження та ведення інформації:

a) збереження даних: Всі дані користувачів, завантажені зображення, глибинні мапи та історія взаємодій зберігаються у захищеній базі даних;

b) організація даних: Використання реляційної бази даних для структурованого зберігання інформації про користувачів та їх взаємодії;

c) безпека збереження: Використання шифрування для захисту конфіденційних даних, таких як токени доступу до моделей Hugging Face.

1.4.5 Вимоги до технічного забезпечення

a) Операційна система:

1. бекенд та LVLM-сервіс: Linux (Ubuntu 20.04 та вище);
2. фронтенд: Підтримка сучасних веббраузерів (Chrome, Firefox, Safari, Edge).

b) Оперативна пам'ять:

1. мінімум 8 GB для забезпечення стабільної роботи системи з GPU-акселерацією.

c) Відеопам’ять:

1. мінімум 12 GB для забезпечення стабільної роботи системи для використання CUDA-функціоналу та завантаження моделей у пам'ять.

d) Місце на диску:

1. мінімум 50 GB для зберігання зображень користувачів.

1.4.6 Вимоги до програпного забезпечення

Архітектура програмної системи побудована на мікросервісній основі з використанням наступних компонентів:

a) бекенд-сервіс (Backend Service): Реалізований на FastAPI, відповідає за обробку запитів, інтеграцію з моделями та управління даними;

b) фронтенд-інтерфейс (Frontend Interface): Розроблений з використанням React, забезпечує користувацький інтерфейс для взаємодії з системою;

c) LVLM-сервіс (LVLM Service): Інтегрований з моделлю SpatialBot[23] для аналізу зображень та генерації текстових відповідей.

Системне програмне забезпечення:

a) операційна система: Linux (Ubuntu 20.04 та вище) для бекенд-сервісу та LVLM-сервісу;

b) Python: Версія 3.11 та вище для розробки бекенду та LVLM-сервісу;

c) Node.js: Версія 14 та вище для розробки фронтенду.

d) Мережне програмне забезпечення:

e) протоколи: HTTP/HTTPS для комунікації між фронтендом, бекендом та LVLM-сервісом;

f) API: RESTful API для обміну даними між компонентами системи.

Мова і технологія розробки ПЗ:

a) мови програмування:

1. Python: Для бекенд-сервісу та LVLM-сервісу;
2. JavaScript (React): Для фронтенд-інтерфейсу.

b) фреймворки та бібліотеки:

1. FastAPI: Для створення високопродуктивних API-сервісів;

2. React: Для розробки динамічного та інтерактивного користувацького інтерфейсу;
3. Transformers (Hugging Face): Для інтеграції моделей глибокого навчання;
4. Docker та Docker Compose: Для контейнеризації та оркестрації сервісів.

1.4.7 Інтерфейс користувача

Інтерфейс користувача розроблено з урахуванням принципів UI/UX дизайну, забезпечуючи зручність та інтуїтивність взаємодії з системою. Основні компоненти інтерфейсу включають:

a) Форма завантаження зображень:
   1. дозволяє користувачам завантажувати RGB-зображення для аналізу;
   2. підтримує перетягування файлів або вибір через діалогове вікно.
b) Поле вводу запиту:
   1. містить текстове поле для введення запитів користувача до системи "DepthGPT";
   2. підтримує функції автозавершення та підказки для полегшення вводу.
c) Чат-інтерфейс:
   1. відображає історію взаємодій між користувачем та системою;

2. підтримує відправлення та отримання повідомлень у реальному часі.

Апаратним інтерфейсом є пристрій користувача, який може бути комп’ютером, планшетом або смартфоном з підтримкою сучасних веббраузерів (Chrome, Firefox, Safari, Edge).

Програмний інтерфейс включає:

a) RESTful API: Для взаємодії між фронтендом, бекендом та LVLM-сервісом;

b) SDK та бібліотеки: Використовуються для інтеграції моделей глибокого навчання та обробки зображень.

Система використовує стандартизовані протоколи HTTP/HTTPS для забезпечення безпечної та ефективної комунікації між компонентами системи.

1.4.8 Властивості програмного забезпечення

Система доступна для будь-якого користувача з доступом до Інтернету та сучасним веббраузером. Завдяки вебархітектурі, користувачі можуть отримувати доступ до системи з різних пристроїв.

Програмне забезпечення розроблено з урахуванням принципів модульності та масштабованості, що дозволяє легко здійснювати супровід та оновлення системи. Використання контейнеризації полегшує процеси розгортання та обслуговування.

Програмне забезпечення може працювати на різних операційних системах, що підтримують Docker, включаючи Linux, Windows та macOS. Фронтенд-інтерфейс адаптований для роботи на різних пристроях та екранах.

Продуктивність роботи системи залежить від потужності серверного обладнання, особливо GPU для обробки моделей глибокого навчання. Система оптимізована для забезпечення швидкого часу обробки запитів та генерації результатів.

Система забезпечує високу надійність завдяки використанню стабільних технологій та регулярному тестуванню. Контейнеризація дозволяє швидко відновлювати роботу системи у випадку збоїв.

Система реалізує механізми захисту даних, включаючи шифрування конфіденційної інформації та захист від несанкціонованого доступу. Використання HTTPS забезпечує безпечну передачу даних між користувачем та серверами.

1.4.9 Інші вимоги

Усі вимоги сформовано та описано вище, доповнення не вимагається.

1.4.10 Примітка

Усі додаткові лістинги вихідного коду, UML-діаграми та скріншоти розміщені у додатках А-Г до роботи.

Висновки до розділу 1

У першому розділі проведено комплексний аналіз предметної області оцінки глибини зображень та інтеграції з природномовними

інтерфейсами. Дослідження охопило теоретичні основи машинного навчання та комп'ютерного зору, сучасні технології оцінки глибини зображень та розвиток великих мовно-візуальних моделей.

Аналіз показав значний прогрес у розвитку методів оцінки глибини зображень, особливо з появою моделей Depth Anything V1 [21] та V2 [22]. Вивчення існуючих рішень, включаючи DenseDepth, Dinov2, FCRN-depth-prediction та інші, дозволило визначити ключові переваги та обмеження кожного підходу. Особливу увагу приділено розвитку великих мовно-візуальних моделей, зокрема SpatialBot, що відкриває нові можливості для природномовної взаємодії з системами комп'ютерного зору.

На основі проведеного аналізу сформульовано детальні вимоги до програмного забезпечення, що охоплюють:

a) функціональні можливості системи;
b) вимоги до інформаційного забезпечення;
c) технічні характеристики;
d) вимоги до користувацького інтерфейсу;

Визначені вимоги створюють надійну основу для подальшого проектування та розробки системи, забезпечуючи чітке розуміння цілей та обмежень проекту. Наступні розділи роботи базуються на результатах проведеного аналізу та сформульованих вимогах.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА МАТЕМАТИЧНОГО ЗАБЕЗПЕЧЕННЯ

У даному розділі представлено проектування архітектури та математичного забезпечення системи на основі результатів проведеного аналізу. Розглядаються функціональні моделі, математичний апарат та алгоритмічне забезпечення розроблюваної системи.

2.1 Функціональне моделювання інформаційних потоків системи "DepthGPT"

2.1.1 Функціональна декомпозиція системи на основі методології IDEF0

Для формалізації функціональних аспектів системи "DepthGPT" було розроблено модель IDEF0, яка відображає ключові процеси та їх взаємодію в системі. Модель демонструє комплексну структуру системи, що складається з взаємопов'язаних компонентів та процесів.

Основними функціональними елементами моделі виступає оцінювання глибини зображень та генерація текстових відповідей на запити користувачів. Система приймає на вхід завантажені користувачами зображення та текстові запити, а результатом її роботи є глибинні мапи та відповідні текстові відповіді. Функціонування системи забезпечується комплексом технічних засобів, що включає моделі глибокого навчання (Depth Anything та SpatialBot [23]), необхідні

обчислювальні ресурси у вигляді графічних процесорів (GPU), а також відповідні веб-технології.

Загальна модель IDEF0 першого рівня структурно розділена на дві основні підсистеми, кожна з яких виконує специфічний набір функцій. Підсистема обробки зображень відповідає за прийом та валідацію вхідних зображень, їх подальший аналіз за допомогою моделей Depth Anything та генерацію відповідних глибинних мап. Підсистема обробки запитів та генерації відповідей забезпечує інтеграцію отриманих глибинних мап з текстовими запитами користувачів, виконує аналіз інтегрованих даних через LVLM-сервіс та формує кінцеві текстові відповіді на запити користувачів.

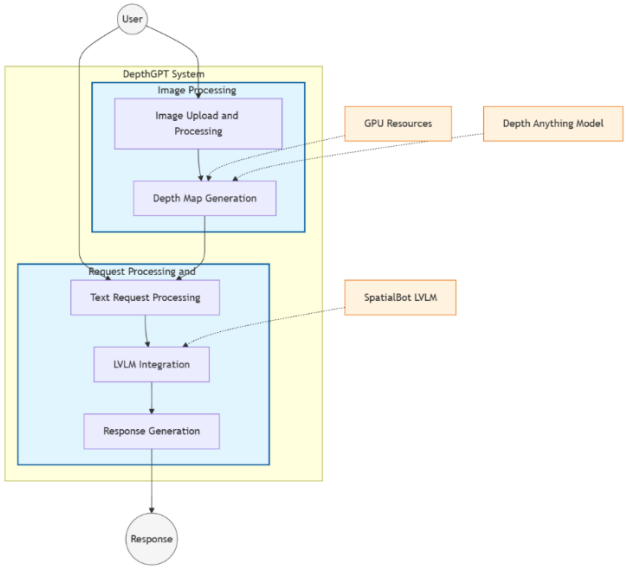


Рисунок 2.1 – IDEF0 діаграма першого рівня для системи "DepthGPT"

Така структурна організація системи забезпечує ефективну обробку вхідних даних та генерацію релевантних відповідей, що

відповідають потребам користувачів. Використання моделі IDEF0 дозволяє чітко відобразити взаємозв'язки між компонентами системи та послідовність виконання операцій обробки даних.

2.1.2 Моделювання потоків даних системи з використанням DFD

Для моделювання інформаційних потоків системи було застосовано методологію DFD (Data Flow Diagrams), яка дозволяє відобразити рух даних між різними компонентами системи та їх взаємодію. В рамках розробленої моделі було визначено основні зовнішні сутності системи, серед яких ключовими є користувач та модель LVLM (SpatialBot [23]), що забезпечує обробку та аналіз візуальної інформації.

Система включає три основні процеси: завантаження зображень, що відповідає за прийом та первинну валідацію візуальних даних; генерацію глибинних мап, яка забезпечує створення тривимірних представлень зображень; та обробку запитів, що координує взаємодію з користувачем та формування відповідей. Ключові інформаційні потоки системи представлені завантаженими зображеннями, згенерованими глибинними мапами, а також текстовими запитами та відповідями.

Особливістю розробленої архітектури є реалізація взаємодії між компонентами системи через чітко визначені інтерфейси, що забезпечує високий рівень модульності та створює передумови для подальшого масштабування рішення. Система реалізує механізми асинхронної

обробки запитів, що дозволяє ефективно обробляти значні обсяги даних та забезпечувати належну продуктивність при зростанні навантаження.

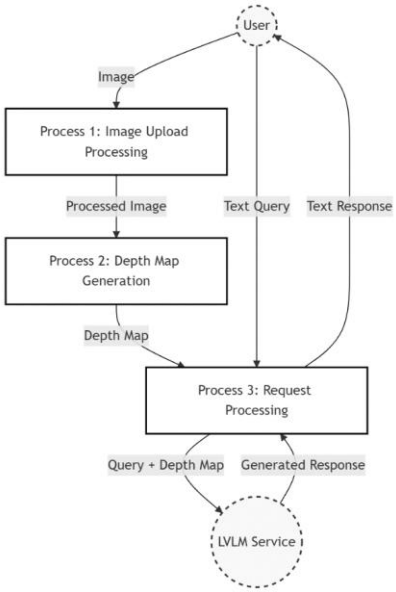


Рисунок 2.2 - Загальна модель DFD першого рівня для системи "DepthGPT"

Запропонована модель функціональної декомпозиції формує фундаментальну основу для подальшої розробки системи, визначаючи чіткі межі відповідальності кожного компонента та регламентуючи механізми їх взаємодії. Такий структурований підхід до проектування архітектури забезпечує необхідну гнучкість у процесі розробки та тестування окремих модулів, а також створює надійне підґрунтя для подальшого масштабування системи відповідно до зростаючих потреб користувачів та збільшення обсягів оброблюваних даних.

2.2 Математичне забезпечення функціональних процесів системи "DepthGPT"

2.2.1 Формалізація функціональних процесів на основі математичного апарату глибокого навчання

Математичне представлення функцій системи "DepthGPT" базується на двох основних компонентах: оцінці глибини зображень та обробці природної мови.

Основна функція оцінки глибини може бути представлена як:

$$D = f(I'), \qquad (1)$$

де D - генерована карта глибини,

I - вхідне RGB-зображення, а f - функція нейронної мережі.

Перед обробкою здійснюється нормалізація зображення:

$$I' = \frac{I - \mu}{\sigma}, \qquad (2)$$

де μ — середнє значення інтенсивності, а σ — стандартне відхилення інтенсивності пікселів.

Для обробки природної мови використовується векторне представлення запитів:

$$g : Text \rightarrow \mathbb{R}^n, \qquad (3)$$

де g - функція векторизації тексту, а n - розмірність векторного простору.

Інтеграція візуальної та текстової інформації здійснюється через:

$$H = h(D, g(Q)), \qquad (4)$$

де h - функція об'єднання модальностей, а Q - текстовий запит користувача.

Детальний математичний опис архітектури моделей, функцій втрат, механізмів оптимізації та додаткових обчислень наведено у Додатку Д.

2.2.2 Програмна реалізація математичних моделей та їх оптимізація

Інтеграція математичних моделей у систему "DepthGPT" здійснюється через комплекс оптимізованих алгоритмів та структур даних. Система реалізує три ключові механізми оптимізації обчислень. \

Перший механізм - паралельна обробка даних, що забезпечує ефективне використання обчислювальних ресурсів через розподіл навантаження між доступними процесорними ядрами та GPU. Це дозволяє значно прискорити обробку зображень та генерацію глибинних мап.

Другий механізм - інтелектуальне кешування результатів. Система зберігає проміжні результати обчислень та часто запитувані дані, що суттєво зменшує час відгуку при повторних запитах. Реалізовано адаптивний алгоритм кешування, який автоматично визначає оптимальний баланс між використанням пам'яті та швидкістю доступу до даних.

Третій механізм - адаптивне масштабування ресурсів, що дозволяє системі динамічно регулювати обчислювальні потужності відповідно до поточного навантаження. Система постійно моніторить використання ресурсів та автоматично перерозподіляє навантаження між доступними обчислювальними вузлами.

Взаємодія з мовною моделлю LVLM реалізована через оптимізований конвеєр обробки даних, що включає:

a) Векторизацію текстових запитів
b) Об'єднання векторних представлень зображень та тексту
c) Генерацію контекстуально релевантних відповідей

Детальний математичний опис механізмів оптимізації та формули розрахунків наведено у Додатку Д.5. Реалізовані механізми забезпечують ефективну обробку вхідних даних, оптимальне використання обчислювальних ресурсів та надійну роботу системи під навантаженням.

2.3 Алгоритмічне забезпечення системи "DepthGPT"

2.3.1 Структурна декомпозиція алгоритмічних процесів системи

У даному підрозділі представлено алгоритм оцінювання глибини зображень та його інтеграцію з обробкою природної мови в системі "DepthGPT". Схема визначає ключові етапи та взаємодію компонентів для досягнення цілей системи.

Алгоритмічна схема слугує основою розробки програмного забезпечення, забезпечуючи структурований підхід до обробки даних від отримання вхідної інформації до генерації результатів через чітко визначену послідовність операцій.

Основними етапами алгоритму є:

a) прийом та валідація вхідних даних через вебінтерфейс;
b) попередня обробка та нормалізація зображень;
c) генерація глибинної мапи за допомогою моделі Depth Anything;
d) інтеграція глибинної мапи з текстовим запитом;
e) аналіз даних та генерація відповіді за допомогою SpatialBot[23];
f) візуалізація результатів та взаємодія з користувачем;
g) збереження даних та історії взаємодій.

Попередня обробка зображення включає валідацію формату, нормалізацію інтенсивностей пікселів та оптимізацію розміру для подальшої обробки. Генерація глибинної мапи здійснюється з використанням моделі глибокого навчання, яка проводить детальний аналіз пікселів зображення для визначення відстані до об'єктів.

Інтеграція з природномовним інтерфейсом реалізується через об'єднання глибинної мапи з текстовим запитом користувача. Отримані дані аналізуються системою SpatialBot [23], яка генерує контекстуально-релевантну відповідь з урахуванням просторової інформації зображення.

Система забезпечує візуалізацію результатів через вебінтерфейс, відображаючи глибинну мапу поряд з оригінальним зображенням та

текстовою відповіддю в чат-інтерфейсі. Усі взаємодії користувача, включаючи завантажені зображення, глибинні мапи та історію діалогів, зберігаються в базі даних для подальшого використання.

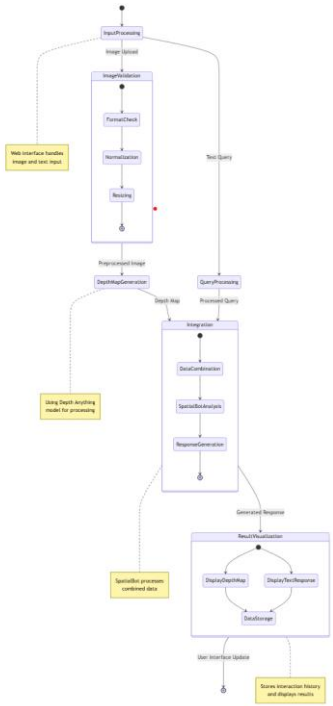


Рисунок 2.3 - робочий процес системи DepthGPT

Запропонована схема демонструє значні переваги завдяки чіткій структурі обробки даних, масштабованості мікросервісної архітектури та оптимізації обчислень через інтеграцію з GPU. Гнучкість системи дозволяє адаптуватися до різноманітних типів запитів та зображень, забезпечуючи широкий спектр практичного застосування.

Таким чином, формалізована схема алгоритму забезпечує ефективну реалізацію функціональності системи "DepthGPT",

поєднуючи високу точність обчислень з оптимальною продуктивністю та адаптивністю до потреб користувачів.

2.3.2 Алгоритмічна реалізація функціональних блоків системи

У цьому підрозділі представлено детальний опис алгоритмічної реалізації функціональних компонентів системи "DepthGPT".

Блок обробки вхідних даних використовує react-webcam для отримання зображень з вебкамери та стандартний інтерфейс завантаження файлів. Процес включає валідацію формату, перевірку розмірів та оптимізацію якості зображень, а також обробку текстових запитів через спеціалізований інтерфейс.

Оцінювання глибини реалізується через моделі Depth Anything з використанням бібліотек torch, cv2 та numpy. Процес охоплює попередню обробку зображень, генерацію прогнозів глибини та постобробку результатів для візуалізації.

Модуль обробки природної мови, базований на LVLM-моделях, забезпечує інтерпретацію запитів та генерацію контекстуальних відповідей з інтеграцією інформації про глибину сцени. Інтеграційний блок реалізує взаємодію між компонентами через RESTful API, використовуючи формати JSON та Base64.

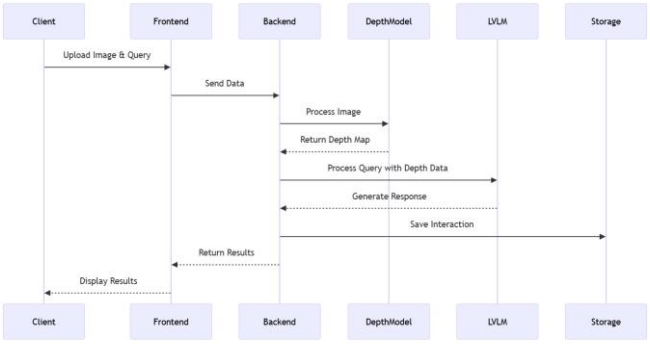


Рисунок 2.4 - діаграма послідовності для компонентів DepthGPT

Інтерактивний вебінтерфейс системи забезпечує відображення оригінального зображення, карти глибини та текстової відповіді з адаптивним дизайном та інструментами для детального аналізу результатів. Така структурована реалізація алгоритмів гарантує ефективну обробку даних та взаємодію компонентів системи "DepthGPT", що є основою для подальшого розвитку проекту.

2.3.3 Графічна формалізація алгоритмів системи

Для візуалізації ключових алгоритмічних процесів системи "DepthGPT" розроблено комплекс блок-схем, що відображають основні потоки обробки даних та логіку прийняття рішень. Дані схеми забезпечують чітке представлення послідовності операцій та взаємозв'язків між компонентами системи.

Центральним елементом розробленої системи, зображеної на рисунку є основний алгоритм обробки запиту, який складається з послідовності взаємопов'язаних етапів.

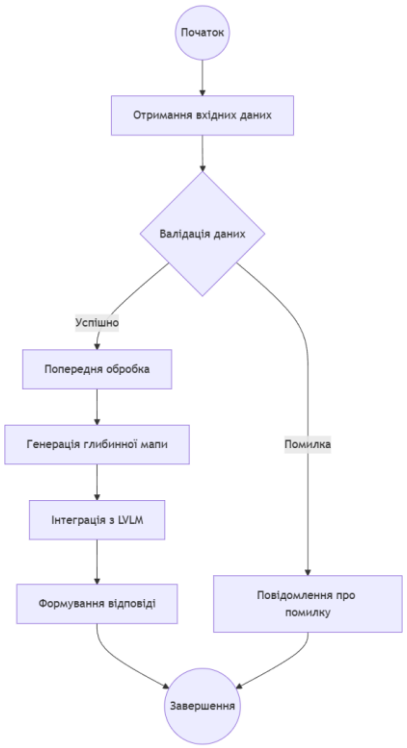


Рисунок 2.5 - блок-схема основного алгоритму системи

Процес починається з прийому та валідації вхідних даних, після чого виконується попередня обробка зображення. Наступним етапом є генерація глибинної мапи, яка потім інтегрується з текстовим запитом користувача. Завершальним етапом виступає формування відповіді на основі проведеного аналізу.

Процес генерації глибинної мапи реалізується через окремий алгоритм, що включає три основні етапи.

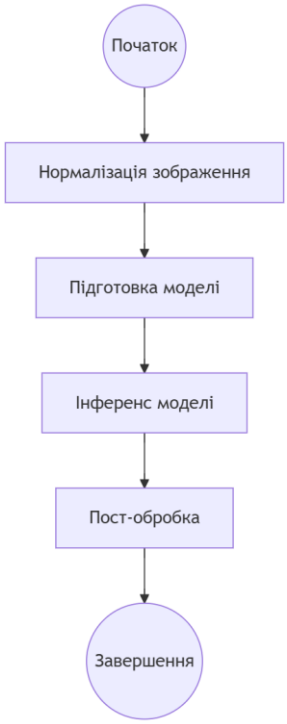


Рисунок 2.6 - блок-схема генерації глибинної мапи

Спочатку виконується нормалізація вхідного зображення для забезпечення стандартизованого формату даних. Після цього застосовується модель Depth Anything для аналізу візуальної інформації. Завершальним етапом є пост-обробка отриманих результатів для забезпечення їх відповідності вимогам системи.

Алгоритм обробки природної мови, який зображений на рисунку , представляє собою послідовність операцій з текстовими даними. Процес розпочинається з токенізації отриманого запиту, після чого виконується векторизація текстової інформації. Фінальним етапом є генерація відповіді на основі проведеного аналізу та обробки даних.

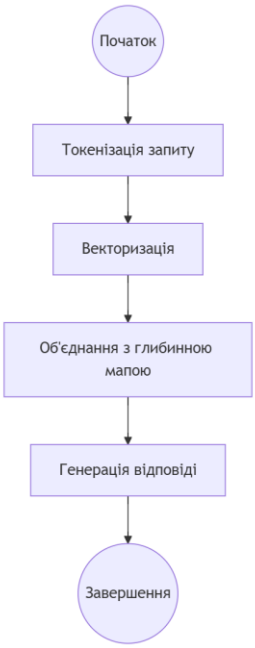


Рисунок 2.7 - блок-схема обробки природної мови

Розроблені блок-схеми забезпечують візуальне представлення концептуальної структури алгоритмів системи, що сприяє чіткому розумінню послідовності операцій, полегшує візуалізацію точок прийняття рішень та допомагає в ідентифікації ключових етапів обробки даних. Така формалізація створює надійну основу для подальшої розробки та оптимізації системи, забезпечуючи єдине розуміння алгоритмічних процесів всіма учасниками проекту та полегшуючи процес внесення модифікацій до існуючої архітектури.

2.4 Моделювання інформаційних процесів та потоків даних системи

Система "DepthGPT" реалізує комплексну архітектуру обробки даних, що базується на трьох фундаментальних інформаційних потоках, кожен з яких виконує специфічні функції та забезпечує окремий аспект роботи системи:

a) перший потік відповідає за обробку зображень та включає послідовність операцій від моменту завантаження зображення користувачем до генерації відповідної глибинної мапи. Процес починається з завантаження візуальних даних, після чого виконується їх валідація та нормалізація. Наступним етапом є генерація глибинної мапи на основі оброблених даних, а завершується процес збереженням отриманих результатів для подальшого використання;
b) другий інформаційний потік забезпечує обробку користувацьких запитів та включає отримання текстового запиту, його інтеграцію з попередньо згенерованою глибинною мапою, проведення аналізу об'єднаних даних через LVLM-сервіс та формування релевантної відповіді на основі проведеного аналізу;
c) третій потік зосереджений на забезпеченні ефективної взаємодії з користувачем та включає механізми відображення результатів обробки даних, системи обробки зворотного зв'язку, реалізацію кешування для оптимізації продуктивності та комплексне управління користувацькими сесіями.

2.4.1 Системний аналіз інформаційних потоків з використанням DFD-методології

Розроблена концептуальна модель, яка зображена на рисунку , інформаційних потоків системи відображає взаємозв'язки між різними компонентами та забезпечує чітке розуміння процесів обробки даних. Модель демонструє, як інформація проходить через різні етапи обробки, починаючи від отримання вхідних даних і закінчуючи формуванням кінцевого результату.

Особлива увага в моделі приділяється механізмам синхронізації даних між різними потоками, що забезпечує узгоджену роботу всіх компонентів системи. Модель також враховує аспекти масштабованості та надійності, передбачаючи можливість паралельної обробки даних та механізми відновлення після збоїв.

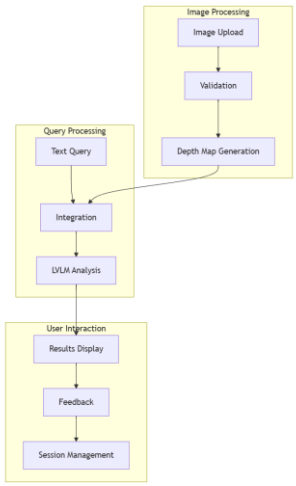


Рисунок 2.8 - концептуальна модель інформаційних потоків

Така структурована організація інформаційних потоків, яка показана на рисунку , створює надійний фундамент для подальшого розвитку системи, забезпечуючи можливість ефективної інтеграції нових функціональних можливостей та оптимізації існуючих процесів відповідно до зростаючих потреб користувачів та вимог до продуктивності системи.

2.4.2 Інтеграція компонентів системи

Взаємодія компонентів системи "DepthGPT" реалізується через комплексний механізм інтеграції, що забезпечує ефективну комунікацію та синхронізацію між різними модулями системи. Основою цього механізму виступає багаторівнева архітектура синхронізації даних, що включає асинхронну обробку запитів, можливості паралельних обчислень та ефективне управління станом системи в цілому.

Важливим аспектом інтеграції є система управління ресурсами, що забезпечує оптимальний розподіл обчислювального навантаження між компонентами. Реалізовано механізми динамічного розподілу завдань, що дозволяють ефективно використовувати доступні обчислювальні ресурси, зокрема графічні процесори (GPU). Система кешування результатів забезпечує швидкий доступ до часто запитуваних даних та знижує навантаження на обчислювальні ресурси.

Особлива увага приділяється забезпеченню надійності роботи системи через впровадження комплексних механізмів обробки помилок та відновлення після збоїв. Система постійного моніторингу відстежує

стан всіх компонентів та забезпечує своєчасне реагування на можливі проблеми. Реалізовано механізми автоматичного відновлення роботи системи після виникнення нештатних ситуацій, що гарантує стабільність роботи сервісу.

Запропонована архітектура інтеграції забезпечує ефективну взаємодію всіх компонентів системи, створюючи можливості для її подальшого масштабування відповідно до зростаючих потреб користувачів. Оптимальне використання доступних ресурсів досягається через впровадження механізмів балансування навантаження та інтелектуального розподілу завдань між компонентами системи.

Концептуальна модель інформаційних процесів, що лежить в основі системи інтеграції, створює надійний фундамент для подальшого розвитку системи, визначаючи ключові точки взаємодії компонентів та встановлюючи ефективні механізми обміну даними між ними. Така архітектура забезпечує необхідну гнучкість для впровадження нових функціональних можливостей та оптимізації існуючих процесів відповідно до вимог користувачів.

Висновки до розділу 2

У даному розділі було проведено детальне моделювання та розробку алгоритмічної бази системи оцінювання глибини зображень з інтеграцією LVLM-сервісу "DepthGPT". Розроблено функціональні моделі системи з використанням методологій IDEF0 та DFD, що дозволило чітко визначити структуру та взаємозв'язки між

компонентами системи. Декомпозиція моделей першого рівня забезпечила глибоке розуміння інформаційних потоків та функціональних процесів.

Створено математичні моделі для ключових функцій системи, включаючи процеси оцінювання глибини зображень та обробки природної мови. Формалізовано алгоритми попередньої обробки зображень, генерації глибинних мап та інтеграції з LVLM-сервісом. Особливу увагу приділено оптимізації обчислень та забезпеченню високої точності результатів через використання сучасних технологій машинного навчання.

Розроблено детальні алгоритми для кожного функціонального блоку системи, представлені у вигляді блок-схем, що забезпечують чітке розуміння послідовності операцій та взаємодії компонентів. Створена алгоритмічна база враховує вимоги до продуктивності, масштабованості та надійності системи, забезпечуючи ефективну обробку даних та генерацію результатів.

Запропоновані моделі та алгоритми формують надійний фундамент для подальшої розробки та впровадження системи "DepthGPT", забезпечуючи її відповідність поставленим функціональним вимогам та технічним характеристикам. Інтеграція математичних моделей у програмне забезпечення через сучасні технології та фреймворки гарантує високу якість та ефективність роботи системи.

3 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У цьому розділі розглядається процес моделювання та конструювання програмного забезпечення системи. Особлива увага приділяється розробці архітектури, проектуванню компонентів та реалізації інтерфейсів взаємодії.

3.1 Розробка архітектури програмного забезпечення

У процесі проектування системи "DepthGPT" було проведено ґрунтовний аналіз сучасних архітектурних підходів до побудови складних програмних систем. За результатами аналізу мікросервісна архітектура була обрана як оптимальне рішення, що найкраще відповідає вимогам проекту та забезпечує необхідну гнучкість у розробці та експлуатації.

3.1.1 Обґрунтування вибору мікросервісної архітектури

Мікросервісний підхід дозволяє декомпозувати систему на автономні сервіси, кожен з яких виконує чітко визначену функцію. Така декомпозиція забезпечує ряд суттєвих переваг для розробки та експлуатації системи. Незалежність компонентів дозволяє здійснювати їх розробку, тестування та розгортання автономно, що значно прискорює процес внесення змін та зменшує ризики при оновленні системи.

Важливою перевагою обраної архітектури є можливість горизонтального масштабування окремих сервісів відповідно до

навантаження. Це особливо актуально для системи "DepthGPT", де обчислювально інтенсивні операції, такі як генерація глибинних мап та обробка природної мови, можуть потребувати додаткових ресурсів незалежно від інших компонентів системи.

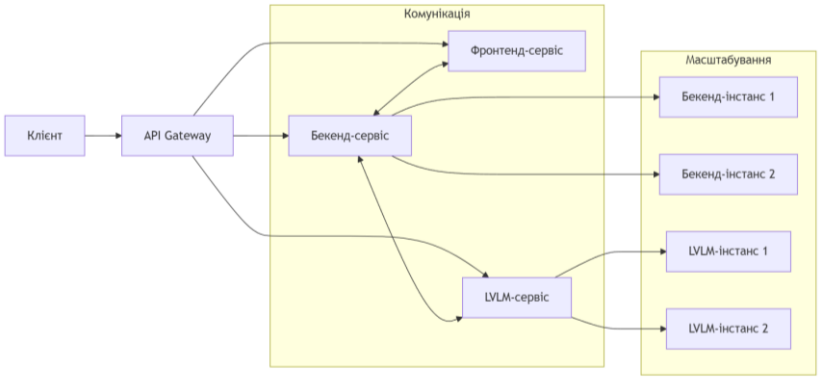


Рисунок 3.1 - комунікаційна схема мікросервісної архітектури

На представленій діаграмі потоків (Flow Diagram) відображено основні комунікаційні зв'язки між компонентами системи та можливості їх масштабування. API Gateway виступає єдиною точкою входу, забезпечуючи маршрутизацію запитів до відповідних сервісів та балансування навантаження.

Мікросервісна архітектура також надає значну гнучкість у виборі технологій для кожного компонента. Це дозволило оптимально підібрати стек технологій: FastAPI для високопродуктивного бекенду, React для інтерактивного користувацького інтерфейсу та спеціалізовані бібліотеки машинного навчання для LVLM-сервісу.

Важливим аспектом є підвищена відмовостійкість системи. Ізоляція сервісів забезпечує локалізацію можливих збоїв, запобігаючи каскадним відмовам. Система може продовжувати функціонування навіть при тимчасовій недоступності окремих компонентів, що суттєво підвищує загальну надійність.

Використання контейнеризації через Docker та оркестрації за допомогою Docker Compose спрощує процеси розгортання та управління версіями компонентів. Це забезпечує узгоджену роботу всіх сервісів та спрощує процес масштабування системи відповідно до зростання навантаження.

Таким чином, вибір мікросервісної архітектури створює надійний фундамент для розвитку системи "DepthGPT", забезпечуючи необхідну гнучкість, масштабованість та ефективність у досягненні поставлених цілей проекту.

Архітектура системи "DepthGPT" реалізує комплексний підхід до обробки даних через взаємодію трьох ключових компонентів, кожен з яких забезпечує специфічний набір функціональних можливостей для ефективної роботи програмного забезпечення.

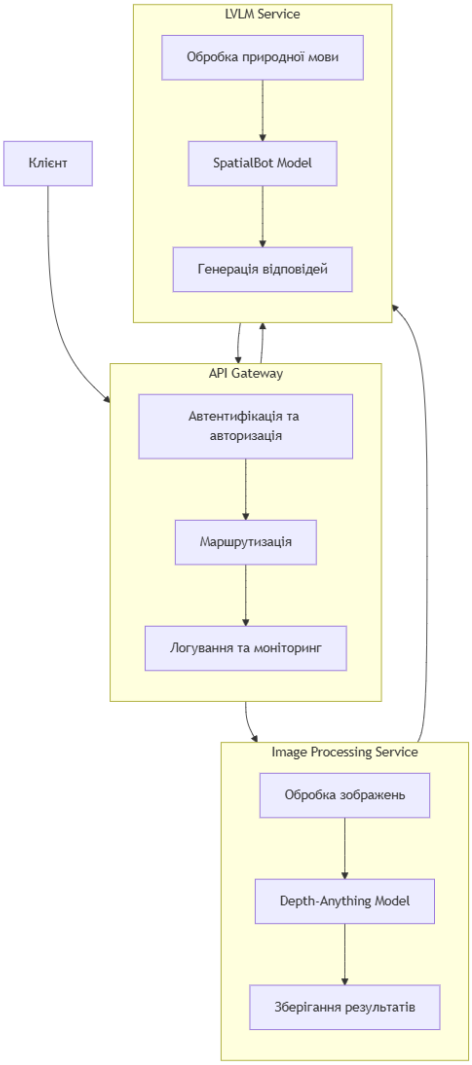


Рисунок 3.2 - архітектура системи DepthGPT

API Gateway виступає центральним елементом системи та забезпечує єдину точку входу для всіх клієнтських запитів. Компонент реалізує комплексну систему маршрутизації та балансування навантаження, що дозволяє ефективно розподіляти запити між відповідними сервісами залежно від їх типу та поточного навантаження

на систему. Важливою функцією Gateway є забезпечення безпеки через механізми автентифікації та авторизації, що контролюють доступ до функціональності системи. Додатково реалізовано систему логування та моніторингу для відстеження всіх запитів та своєчасного виявлення можливих проблем.

Image Processing Service відповідає за комплексну обробку візуальних даних та генерацію глибинних карт. Процес починається з попередньої обробки вхідних зображень, що включає необхідні перетворення для підготовки до глибинного аналізу. Основним елементом сервісу є інтеграція з моделлю Depth-Anything, яка забезпечує високоточний аналіз глибини зображення. Результати аналізу зберігаються у відповідному форматі для подальшого використання іншими компонентами системи.

LVLM Service реалізує функціональність обробки природної мови та генерації текстових відповідей на основі комплексного аналізу візуальних та текстових даних. Сервіс забезпечує інтерпретацію користувацьких запитів, їх обробку за допомогою моделі SpatialBot [23] та формування релевантних текстових відповідей, що враховують як візуальні, так і текстові аспекти вхідних даних.

Взаємодія між компонентами системи реалізована через стандартизовані інтерфейси, що забезпечує високу модульність архітектури. Результати обробки зображень передаються від Image Processing Service до LVLM Service для подальшого аналізу, а API

Gateway координує весь процес та забезпечує доставку кінцевих результатів клієнтам. Така архітектура створює надійну основу для подальшого масштабування системи та інтеграції нових функціональних можливостей відповідно до зростаючих потреб користувачів.

Взаємодія між компонентами системи "DepthGPT" реалізована через комбінацію синхронних REST API викликів та асинхронної обробки повідомлень, що забезпечує оптимальну продуктивність та надійність системи. REST API забезпечує стандартизований інтерфейс для комунікації між клієнтами та системою через API Gateway. Асинхронна обробка запитів реалізована для операцій, що потребують тривалого часу виконання, таких як генерація глибинних мап та аналіз даних. Процес комунікації починається з валідації вхідних запитів на рівні API Gateway, після чого відбувається послідовна обробка даних через сервіс обробки зображень та LVLM-сервіс.

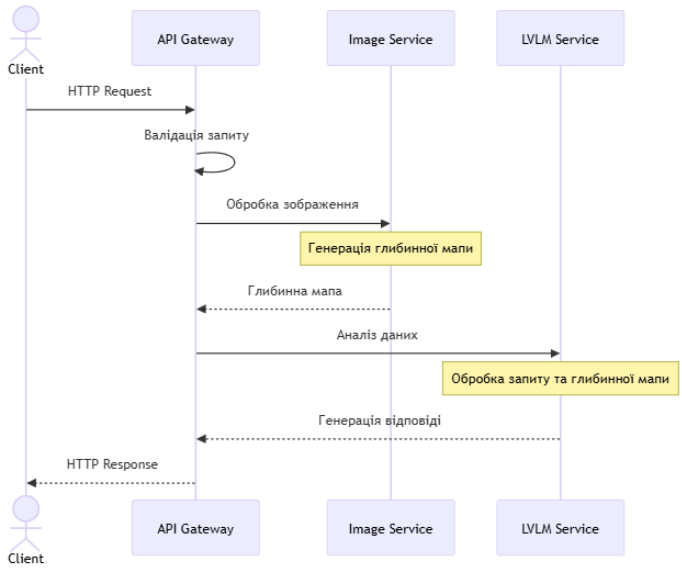


Рисунок 3.3 - процес комунікації між компонентами

Взаємодія між компонентами системи базується на принципах мікросервісної архітектури, що забезпечує необхідну гнучкість для масштабування системи та додавання нових функціональних можливостей, зберігаючи при цьому високу надійність та ефективність обробки запитів. Кожен компонент системи має чітко визначений інтерфейс взаємодії, що спрощує процес розробки та тестування окремих модулів.

3.2 Компоненти програмного забезпечення

3.2.1 Серверні компоненти

Архітектура серверної частини системи "DepthGPT" реалізує мікросервісний підхід, що забезпечує модульність та ефективний розподіл відповідальності між компонентами. API Gateway виступає

єдиною точкою входу для клієнтських запитів та забезпечує маршрутизацію, автентифікацію і передачу запитів до відповідних сервісів. Image Processing Service відповідає за обробку вхідних зображень та взаємодію з моделлю Depth Anything для створення глибинних карт. LVLM Service забезпечує інтеграцію з моделлю SpatialBot [23] для генерації текстових відповідей на основі даних глибини та зображення. Worker Service реалізує механізми асинхронної обробки задач, таких як обробка зображень та виклики великих мовних моделей.

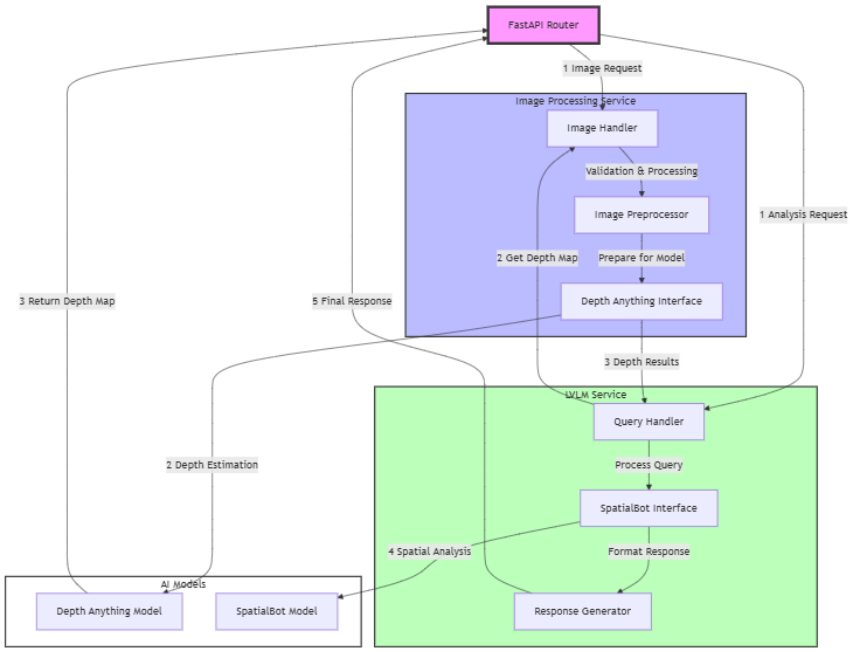


Рисунок 3.4 - діаграма серверних компонентів

3.2.2 Клієнтські компоненти

Клієнтська частина системи реалізована як модульний веб-застосунок, що забезпечує зручний інтерфейс для взаємодії користувача з системою. Upload Interface надає функціональність для завантаження зображень. Webcam Module забезпечує можливість потокової передачі зображень безпосередньо з вебкамери користувача. DepthGPT Module реалізує інтерфейс для введення текстових запитів до LVLM та відображення отриманих відповідей. Visualization Component відповідає за візуалізацію результатів у вигляді RGB-зображень та глибинних мап.

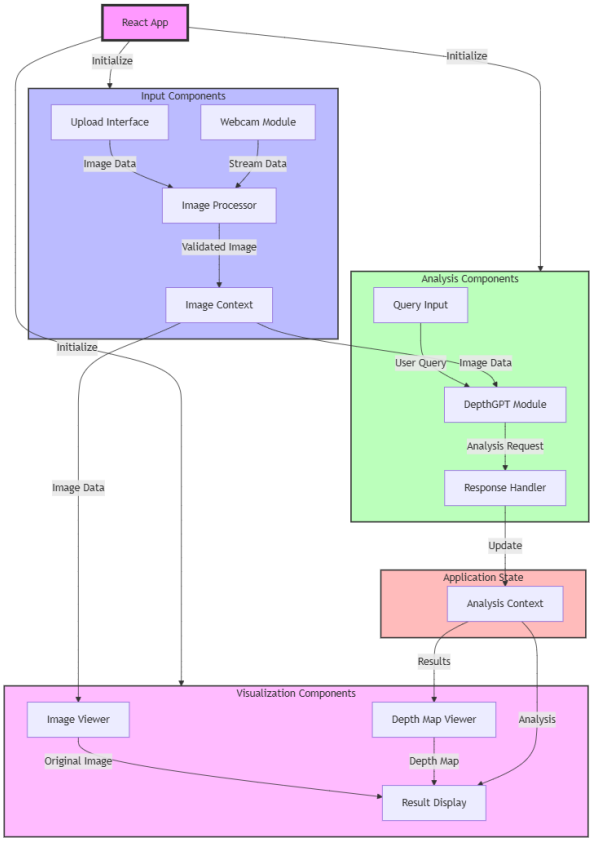


Рисунок 3.5 - діаграма клієнтських компонентів

Рисунок відображає архітектуру фронтенд частини системи DepthGPT, демонструючи основні функціональні блоки для роботи з введенням даних, аналізом та візуалізацією результатів. В центрі архітектури знаходиться система управління станом, що забезпечує зберігання та управління даними зображень та результатів аналізу. React App виступає кореневим компонентом, який координує взаємодію між усіма іншими компонентами, забезпечуючи потік даних від компонентів введення через компоненти аналізу до компонентів відображення результатів.

3.3 UML-діаграми системи

Для формалізації поведінки та взаємодії компонентів системи "DepthGPT" розроблено комплекс UML-діаграм першого рівня деталізації, що відображають різні аспекти функціонування системи.

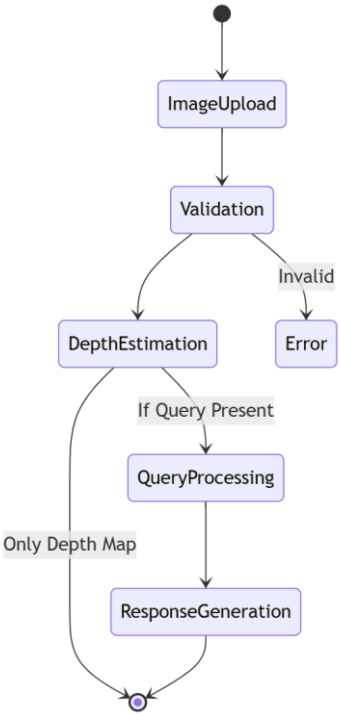


Рисунок 3.6 - процесна діаграма обробки запиту

Процесна діаграма (state chart) демонструє життєвий цикл обробки запиту від завантаження зображення до генерації відповіді, включаючи етапи валідації вхідних даних та два можливі сценарії виконання: отримання лише карти глибини або повний аналіз із генерацією текстової відповіді. Діаграма відображає всі основні стани системи та переходи між ними, включаючи обробку помилкових ситуацій при невалідних вхідних даних та розгалуження процесу в залежності від наявності текстового запиту від користувача.

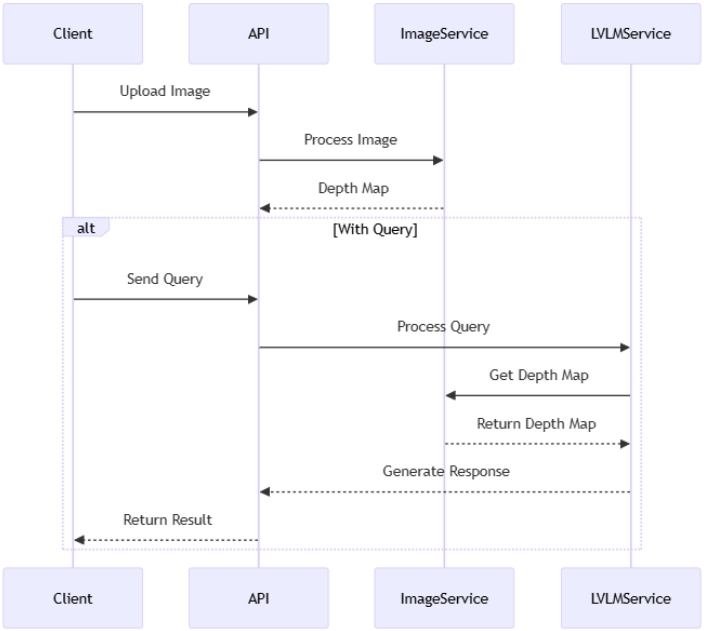


Рисунок 3.7 - подієва діаграма взаємодії сервісів

Подієва діаграма (sequence diagram) відображає взаємодію між різними сервісами системи, детально показуючи послідовність та часові характеристики обміну повідомленнями між клієнтом, API Gateway та спеціалізованими сервісами. Діаграма охоплює два можливі сценарії використання системи: перший - отримання лише карти глибини, що включає взаємодію з Image Service для обробки зображення, та другий - повний аналіз із запитом, який додатково залучає LVLM Service для генерації текстової відповіді на основі карти глибини та запиту користувача. Кожен сценарій демонструє чітку послідовність взаємодій та передачі даних між компонентами системи.

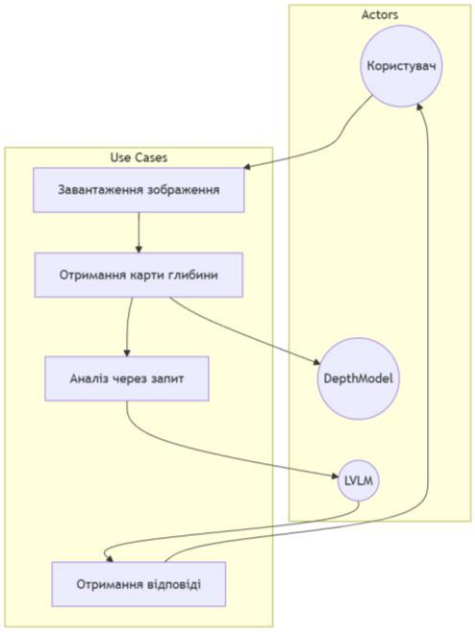


Рисунок 3.8 - діаграма використання системи

Діаграма використання (use case diagram) показує можливі сценарії взаємодії користувача з системою та залучення різних компонентів у цих сценаріях, визначаючи основні функціональні можливості системи з точки зору кінцевого користувача. На діаграмі представлено основних акторів системи: користувача, який ініціює всі взаємодії, модель оцінки глибини (DepthModel) для аналізу зображень та мовну модель (LVLM) для обробки текстових запитів. Діаграма демонструє послідовність дій від завантаження зображення через отримання карти глибини до можливого подальшого аналізу через текстовий запит, при цьому чітко відображаючи залежності між різними варіантами використання системи та їх зв'язок з відповідними компонентами обробки.

3.4 Опис інтерфейсів програмного забезпечення

Розроблена система "DepthGPT" реалізує комплексну архітектуру інтерфейсів, що забезпечує ефективну взаємодію між клієнтською частиною, серверними компонентами та моделями машинного навчання. В основі архітектури лежить веб-інтерфейс, розроблений за допомогою React, що надає користувачам інтуїтивно зрозумілі засоби для завантаження зображень, перегляду результатів аналізу глибини та взаємодії з моделями. Серверна частина, реалізована на базі FastAPI, забезпечує обробку HTTP-запитів та взаємодію з моделями глибокого навчання, повертаючи результати у форматі JSON.

Для реалізації функціональності глибинного аналізу зображень використовуються моделі машинного навчання на базі PyTorch та бібліотек Huggingface. Вся система розгортається у контейнеризованому середовищі з використанням Docker, що забезпечує необхідну ізоляцію компонентів та спрощує процес розгортання в хмарному середовищі.

3.4.1 API Endpoints

Серверна частина системи надає два основні програмні інтерфейси для взаємодії з моделями глибокого навчання через HTTP протокол. Перший endpoint "/predict" забезпечує базову функціональність оцінки глибини зображення, тоді як "/depthgpt" реалізує розширений функціонал з можливістю аналізу зображень через текстові запити. Розглянемо детальну специфікацію кожного endpoint.

Endpoint "/predict" забезпечує базову функціональність оцінки глибини зображення:

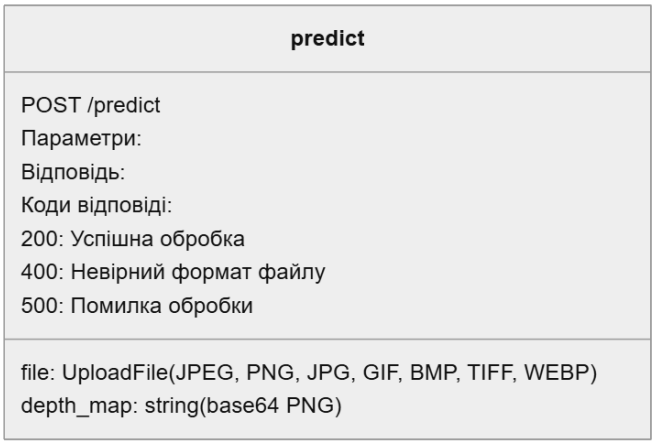


Рисунок 3.9 – детальний опис endpoint “predict”

Endpoint "/predict", який описаний на рисунку 3.9, представляє собою базовий інтерфейс для оцінки глибини зображення. Даний endpoint приймає зображення у форматі multipart/form-data та виконує його аналіз за допомогою моделі Depth Anything. Основною функцією є генерація карти глибини для вхідного зображення, яка повертається у форматі base64-кодованого PNG-зображення. Endpoint забезпечує валідацію вхідних даних, перевіряючи формат та наявність файлу зображення, а також обробляє потенційні помилки, що можуть виникнути під час аналізу.

Рисунок 3.10 - детальний опис endpoint “depthgpt”

Endpoint "/depthgpt", який описаний в рисунку 3.10, реалізує розширену функціональність системи, дозволяючи користувачам не лише отримувати карту глибини, але й виконувати аналіз зображення на основі текстового запиту. Цей endpoint приймає два параметри: файл зображення та текстовий запит. На основі цих даних він генерує комплексну відповідь, яка включає текстовий аналіз від моделі LVLM (SpatialBot), карту глибини зображення та оригінальне RGB-зображення у форматі base64. Такий підхід забезпечує можливість інтерактивного аналізу зображень через природномовні запити, що значно розширює функціональні можливості системи.

Обидва endpoints інтегровані в загальну архітектуру системи та використовують єдиний підхід до обробки помилок та форматування

відповідей, що забезпечує узгоджений користувацький досвід при роботі з API.

3.4.2 Візуалізація інтерфейсу користувача

У цьому розділі представлено детальний опис візуального оформлення та функціональності користувацького інтерфейсу розробленої системи. Інтерфейс складається з декількох ключових компонентів, кожен з яких виконує специфічні функції для забезпечення ефективної взаємодії з користувачем.

Основним елементом системи є блок завантаження зображень, який надає користувачам декілька способів внесення візуальних даних. Центральним компонентом цього блоку є кнопка вибору файлу, що забезпечує стандартний механізм завантаження зображень з локального пристрою. Додатково система включає інтегрований модуль вебкамери, що надає можливість захоплення та миттєвого завантаження зображень безпосередньо через браузер.

Блок візуалізації є ключовим компонентом для представлення результатів роботи системи. Він складається з двох основних вікон: вікна оригінального зображення, де відображається завантажений користувачем матеріал, та вікна карти глибини, яке демонструє результати алгоритмічного аналізу глибини зображення. Для забезпечення детального вивчення результатів передбачено елементи керування масштабом, що дозволяють користувачам збільшувати окремі ділянки зображень.

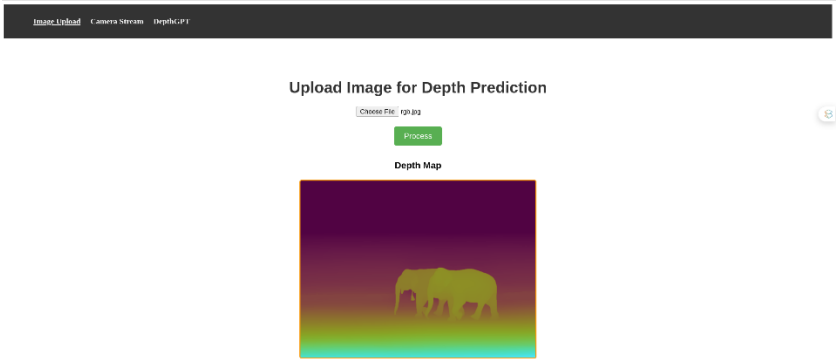


Рисунок 3.11 – перша вкладка системи DepthGPT, яка реалізує завантаження зображення з файлової системи та візуалізує карту глибини

Рисунок 3.11 показує можливості завантаження звичайного RGB-зображення у систему і візуалізацію карти глибини цього зображення.

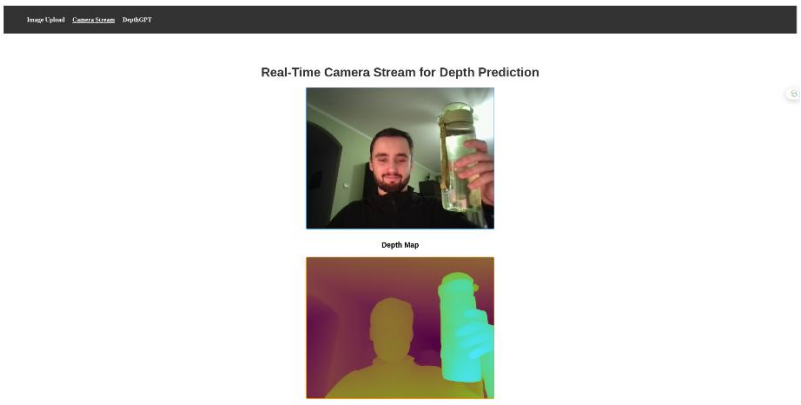


Рисунок 3.12 – друга вкладка системи DepthGPT, яка реалізує трансляцію відео з вебкамери в систему

Рисунок 3.12 представляє реалізацію прямої трансляції з вебкамери в систему та візуалізацію карту глибини цієї трансляції.

Важливим елементом системи є чат-інтерфейс, який забезпечує інтерактивну взаємодію з користувачем. Він включає поле введення запитів, де користувачі можуть формулювати свої команди та питання, а також область історії діалогу, яка зберігає контекст попередніх взаємодій. Для покращення користувацького досвіду впроваджено індикатори стану обробки, які інформують про поточний етап обробки запиту.

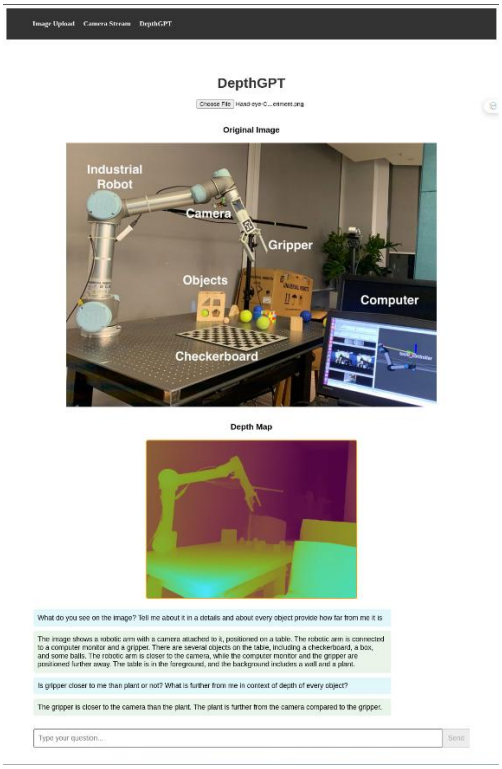


Рисунок 3.13 – комунікація з системою DepthGPT щодо завантаженого зображення

Рисунок 3.13 показує можливості комунікації з великою мовно-візуальною моделлю щодо RGB-зображення та результуючою картою глибини від Depth Anything моделі.

Система розроблена з урахуванням сучасних вимог до веб-додатків та реалізована як Single Page Application (SPA), що забезпечує безперервний користувацький досвід без необхідності перезавантаження сторінок. Інтерфейс має адаптивний дизайн, що дозволяє ефективно працювати з системою на різних пристроях, включаючи мобільні телефони та планшети. Особлива увага приділена візуальній консистентності та інтуїтивності використання, що досягається через застосування єдиного стилю та кольорової гами у всіх компонентах системи.

Основний код програми наведено у Додатку Г.

Висновки до 3 розділу

У третьому розділі було детально розглянуто процес розробки програмного забезпечення системи "DepthGPT". Здійснено вибір та обґрунтування технологічного стеку, що включає Python та FastAPI для серверної частини, React для клієнтської частини, та спеціалізовані бібліотеки PyTorch і Hugging Face для роботи з моделями машинного навчання. Такий вибір технологій забезпечує оптимальну продуктивність та масштабованість системи.

В рамках розробки було створено мікросервісну архітектуру, що складається з трьох основних компонентів: API Gateway для обробки

запитів, Image Processing Service для аналізу зображень, та LVLM Service для роботи з мовною моделлю. Взаємодія між компонентами реалізована через REST API, що забезпечує гнучкість та можливість незалежного масштабування кожного сервісу.

Розроблено та реалізовано математичні моделі для оцінки глибини зображень, включаючи алгоритми попередньої обробки даних та оптимізації обчислень. Особливу увагу приділено інтеграції моделі Depth Anything для генерації карт глибини та моделі SpatialBot для обробки природномовних запитів.

Створено комплексний користувацький інтерфейс, що забезпечує зручну взаємодію з системою через веб-браузер. Інтерфейс включає функціонал для завантаження зображень, перегляду результатів аналізу та взаємодії через текстові запити.

Розроблено та задокументовано API системи, що надає два основних endpoints для взаємодії з моделями глибокого навчання. Перший забезпечує базову функціональність оцінки глибини зображення, другий реалізує розширений функціонал з можливістю аналізу зображень через текстові запити.

Таким чином, розроблене програмне забезпечення повністю реалізує поставлені функціональні вимоги та створює надійну основу для подальшого розвитку системи. Використання сучасних технологій та архітектурних рішень забезпечує високу продуктивність, масштабованість та зручність використання системи.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

Даний розділ присвячено практичній реалізації та тестуванню системи. Розглядаються обрані технології та інструменти розробки, проводиться аналіз результатів тестування та оцінка ефективності розробленого рішення.

4.1 Аналіз та вибір технологій

4.1.1 Порівняльний аналіз технологій та інструментів розробки

У розділі проаналізовано технології, моделі та інструменти для оцінки глибини зображень, включаючи фреймворки глибокого навчання, обчислювальні платформи, інструменти розробки та розгортання, а також розробницькі бібліотеки.

Серед фреймворків глибокого навчання виділяються TensorFlow з його потужними інструментами для виробничих середовищ, PyTorch зі зручністю для досліджень та гнучкістю архітектури, та Hugging Face з готовими моделями та активною спільнотою. Кожен має свої переваги та обмеження для різних сценаріїв використання.

У сфері моделей оцінки глибини розглянуто Depth-Anything v2 [22], що відзначається високою точністю та універсальністю, MiDaS [30] зі швидким інференсом, та DPT з використанням трансформерних архітектур. Кожна модель має свої вимоги до ресурсів та особливості застосування.

Обчислювальні платформи представлені NVIDIA CUDA, що забезпечує оптимізовану підтримку GPU але залежить від специфічного обладнання, та Google TPU, ефективною для масивних обчислень але обмеженою хмарними сервісами.

Docker та Kubernetes виступають ключовими інструментами розгортання, де Docker забезпечує ізоляцію середовищ та просте розгортання, а Kubernetes пропонує масштабованість та автоматизоване управління, хоча обидва потребують специфічних знань.

Розробницькі бібліотеки OpenCV та NumPy пропонують різні можливості: OpenCV оптимізований для обробки зображень але має обмеження у глибокому навчанні, а NumPy забезпечує ефективні операції з даними але потребує оптимізації для великих обсягів.

Порівняння технологій базувалося на метриках продуктивності моделі, масштабованості системи та вимог до ресурсів, що дозволило об'єктивно оцінити кожне рішення для проекту.

Категорія	Технологія/Інструмент	Переваги	Недоліки
Фреймворки Глибокого Навчання	TensorFlow	Розширені інструменти для виробничих середовищ,	Вища кривина навчання, більш громіздкий код

		велика спільнота	
	PyTorch	Простота для дослідницьких цілей, гнучкість	Складніша інтеграція у виробничих середовищах
	Hugging Face	Готові моделі, легка інтеграція	Обмежена підтримка деяких моделей, залежність від інших фреймворків
Моделі оцінки глибини	Depth-Anything v2	Висока точність, універсальність, масштабованість	Високі вимоги до ресурсів, складність налаштування
	MiDaS [30]	Швидкий інференс, легкість	Менша точність у складних

		інтеграції, висока точність на різних сценах	умовах, обмежені можливості налаштування
	DPT (Dense Prediction Transformer) [29]	Висока точність, гнучкість, сучасна архітектура	Високі обчислювальні вимоги, складність розгортання
Обчислювальні платформи	NVIDIA CUDA	Підтримка GPU, широка підтримка фреймворків, оптимізація для NVIDIA GPU	Залежність від специфічного обладнання, висока вартість обладнання
	Google TPU	Висока ефективність для масивних обчислень, інтеграція з Google Cloud	Менш універсальні, залежність від хмарних сервісів

Інструменти для Розробки та Розгортання	Docker	Ізоляція середовищ, полегшене розгортання, велика екосистема	Вимоги до знань контейнеризації, складність при масштабуванні
	Kubernetes	Масштабованість, висока доступність, автоматизоване управління	Складність налаштування, висока ресурсна витратність
Розробницькі бібліотеки	OpenCV	Попередня обробка зображень, оптимізовані алгоритми, широка підтримка мов	Обмежені можливості для глибокого навчання, потреба в додаткових налаштуваннях
	NumPy	Ефективні операції над	Не підходить для дуже

		даними, висока продуктивність, широка інтеграція	великих даних без оптимізації, обмежені можливості для паралельних обчислень

Таблиця 4.1 - структурований аналіз різних технологій та інструментів

Таблиця 4.1 порівняння демонструє структурований аналіз різних технологій та інструментів, які розглядаються для реалізації системи оцінки глибини зображень. Вона організована за категоріями, такими як фреймворки глибокого навчання, моделі оцінки глибини, обчислювальні платформи, інструменти для розробки та розгортання, а також розробницькі бібліотеки. Для кожної технології наведені її основні переваги та недоліки, що дозволяє здійснити об'єктивне порівняння та вибрати найбільш відповідні рішення відповідно до вимог проекту.

4.1.2 Обґрунтування вибору технологічного стеку

Вибір відповідних технологій та інструментів для реалізації системи оцінки глибини зображень є ключовим етапом, який забезпечує ефективність, гнучкість та масштабованість розроблюваного рішення. На основі проведеного порівняльного аналізу у розділі 4.1.1 було

прийнято рішення про використання конкретних технологій, які найкраще відповідають вимогам проекту та його цільовим завданням.

Мова програмування Python обрана завдяки її лідируючій позиції у галузі глибокого навчання та широкому набору бібліотек, таких як PyTorch, NumPy та OpenCV. Python забезпечує високу продуктивність розробки завдяки простому синтаксису та великій кількості готових рішень, що значно скорочує час на розробку та експерименти з моделями.

Фреймворк глибокого навчання PyTorch був обраний завдяки своїй простоті та зручності для дослідницьких цілей. PyTorch забезпечує інтуїтивно зрозумілий інтерфейс та динамічне обчислювальне графікування, що дозволяє легко створювати та модифікувати кастомні архітектури моделей. Хоча інтеграція PyTorch у виробничі середовища може бути складнішою порівняно з TensorFlow, його гнучкість та популярність у наукових дослідженнях роблять його оптимальним вибором для даного проекту.

Модель оцінки глибини Depth-Anything v2 [22] була обрана через її високу точність, універсальність та можливість масштабування для різних задач. Depth-Anything v2 [22] забезпечує надійні результати у різних умовах освітлення та складності сцен, що є критичним для забезпечення якості оцінки глибини. Незважаючи на високі вимоги до ресурсів та складність налаштування, переваги цієї моделі переважають її недоліки, особливо в контексті потреб проекту.

Інструмент контейнеризації Docker був обраний завдяки його здатності забезпечувати ізоляцію середовищ розробки та полегшувати розгортання додатків на різних платформах. Docker дозволяє створювати консистентні середовища для розробників та виробничих серверів, що знижує ризик виникнення проблем з сумісністю та спрощує процеси тестування та деплою.

Для інтеграції Vision-Language Models (LVLM) було обрано бібліотеку Hugging Face Transformers. Цей вибір обумовлений широким набором готових моделей та легкістю їх інтеграції у проект. Hugging Face надає доступ до переднавчених моделей, що дозволяє швидко впроваджувати складні функції обробки природної мови та візуальної інформації без необхідності розробки моделей з нуля.

Обчислювальна платформа NVIDIA CUDA була обрана завдяки її здатності забезпечувати високу продуктивність обчислень через підтримку GPU. CUDA оптимізована для роботи з популярними фреймворками глибокого навчання, такими як PyTorch, що дозволяє значно пришвидшити процеси тренування та інференсу моделей. Хоча використання NVIDIA CUDA вимагає специфічного апаратного забезпечення та має високу вартість, переваги у вигляді швидкості обчислень та підтримки широкого спектру інструментів роблять її незамінною для даного проекту.

Архітектура системи базується на мікросервісній моделі, що забезпечує гнучкість, модульність та масштабованість розроблюваного

рішення. Мікросервісна архітектура дозволяє розділити систему на окремі компоненти, кожен з яких відповідає за конкретну функціональність, що спрощує їхнє оновлення та масштабування незалежно від інших частин системи. Це також сприяє підвищенню надійності та доступності системи, оскільки відмова одного мікросервісу не впливає на роботу інших.

На основі проведеного аналізу було обрано інструменти та технології, які найкраще відповідають вимогам проекту щодо точності, продуктивності, гнучкості та можливості масштабування. Вибір Python, PyTorch, Depth-Anything v2, Docker, Hugging Face Transformers, NVIDIA CUDA та мікросервісної архітектури забезпечує оптимальне поєднання ефективності та зручності використання, що відповідає основним цілям проекту. Ці технології дозволяють створити потужну та масштабовану систему оцінки глибини зображень, здатну ефективно обробляти великі обсяги даних та адаптуватися до змінних вимог користувачів.

4.1.3 Особливості реалізації основних компонентів

Для реалізації системи "DepthGPT" було проведено ретельний аналіз та обрано комплекс технологій і мов програмування, що забезпечують оптимальну продуктивність, масштабованість та зручність розробки. Вибір кожного інструменту базувався на детальному аналізі системних вимог, враховуючи специфіку обробки зображень, роботу з глибинними моделями, інтеграцію з мовними моделями та необхідність забезпечення безперебійної роботи в режимі реального часу.

Основною мовою програмування було обрано Python, що є визнаним галузевим стандартом у сфері штучного інтелекту та машинного навчання. Python надає широкий набір бібліотек та фреймворків для роботи з глибинними нейронними мережами, включаючи PyTorch, TensorFlow та OpenCV. Мова забезпечує зручність швидкого прототипування завдяки читабельному синтаксису та розвиненій екосистемі, а також пропонує розширені можливості для інтеграції з API та іншими сервісами через REST або GraphQL інтерфейси.

Для роботи з глибинними моделями було обрано фреймворк PyTorch, який забезпечує необхідну гнучкість у побудові та налаштуванні моделей, високу продуктивність завдяки інтеграції з NVIDIA CUDA, та підтримку великих мовних моделей через бібліотеки Hugging Face. Розробка веб-інтерфейсу та API реалізована за допомогою FastAPI для бекенду та React для фронтенду, що забезпечує високу швидкість роботи, асинхронність та зручність створення інтерактивного користувацького інтерфейсу.

Система розгортається у контейнеризованому середовищі за допомогою Docker, що забезпечує ізоляцію середовищ розробки та уніфікованість процесу розгортання. Для обробки зображень використовуються бібліотеки OpenCV та NumPy, що надають потужний інструментарій для базових операцій із зображеннями та ефективної роботи з багатовимірними масивами даних.

Обчислювальна платформа базується на NVIDIA RTX 4080 з підтримкою CUDA, що забезпечує необхідну потужність для роботи моделей Depth-Anything та LVLM. Інтеграція з мовною моделлю реалізована через бібліотеку Hugging Face Transformers, яка надає зручний доступ до великих мовних моделей та забезпечує їх безпроблемну інтеграцію з Python-середовищем.

Для забезпечення можливостей масштабування передбачено використання хмарних сервісів AWS або Google Cloud, що надають необхідні інструменти для автоматичного розгортання, балансування навантаження та ефективної обробки даних.

Обраний комплекс технологій створює потужну технологічну базу, що забезпечує високу продуктивність, масштабованість та зручність розробки системи. Використання Python та PyTorch як основних інструментів розробки, разом з Docker для розгортання та спеціалізованими бібліотеками для обробки даних, формує ефективне середовище для реалізації всіх необхідних функціональних можливостей системи.

4.2 Тестування системи

У даному підрозділі представлено комплексний підхід до тестування розробленої системи, що охоплює методологію тестування, аналіз продуктивності та оцінку точності глибинного аналізу зображень. Особлива увага приділяється валідації результатів та перевірці відповідності системним вимогам.

4.2.1 Методологія тестування

Методологія тестування базується на трирівневому підході, що включає функціональне тестування, оцінку продуктивності та валідацію моделей машинного навчання. Функціональне тестування охоплює перевірку окремих компонентів системи та їх взаємодії через модульні та інтеграційні тести. Основний код тестів наведено у Додатку Б.1.

Тестування продуктивності реалізовано за допомогою інструментів Locust та Apache JMeter, що дозволяють симулювати різні сценарії навантаження та оцінювати стабільність системи. Валідація моделей здійснюється через оцінку точності прогнозування глибини з використанням стандартних метрик RMSE та MAE.

4.2.2 Результати тестування продуктивності

Результати тестування продуктивності демонструють здатність системи ефективно обробляти значні обсяги запитів. При стандартному навантаженні у 10 одночасних користувачів система демонструє середній час відгуку 4000 мс з 3% помилок. Стрес-тестування при 100 одночасних користувачах показало зростання середнього часу відгуку до 730 мс зі доволі значним відсотком помилок (10%) при максимальному навантаженні.

Параметра	Значення
Кількість користувачів	10 одночасних
Кількість запитів за хвилину	60 запитів/хв

Медіанний час відповіді	4000 мс.
Помилки	3%

Таблиця 4.1 - Результати навантажувального тестування

При стандартному навантаженні у 10 одночасних користувачів система демонструє прийнятну продуктивність з медіанним часом відповіді 4000 мс. Кількість помилок становить 3%, що є допустимим показником для систем машинного навчання з високими обчислювальними вимогами. Система успішно обробляє 60 запитів за хвилину, що відповідає поставленим вимогам щодо пропускної здатності.

Параметра	Значення
Кількість користувачів	100 одночасних
Кількість запитів за хвилину	1000 запитів/хв
Медіанний час відповіді	7300 мс.
Помилки	10% (при перевантаженні)

Таблиця 4.2 - Результати стрес-тестування

Стрес-тестування при 100 одночасних користувачах виявило межі масштабованості системи. Збільшення навантаження призвело до зростання медіанного часу відповіді до 7300 мс та підвищення рівня помилок до 10%. Такі результати вказують на необхідність оптимізації системи для роботи з більшою кількістю одночасних користувачів, зокрема через впровадження механізмів балансування навантаження та покращення алгоритмів кешування.

4.2.3 Аналіз точності оцінки глибини

Оцінка точності моделей глибинного аналізу проводилась на стандартизованому наборі тестових зображень. Результати демонструють високу точність оцінки глибини для всіх варіантів моделі Depth-anything v2, з найкращими показниками для версії vitl.

Модель	RMSE, мм	MAE, мм
Depth-anything v1_vits	53.5	40.5
Depth-anything v1_vitb	50.2	36.7
Depth-anything v1_vitl	44.8	31.25
Depth-anything v2_vits	43.1	30.8
Depth-anything v2_vitb	40.8	27.4
Depth-anything v2_vitl	36.3	25.5

Таблиця 4.3 - Порівняння точності різних версій моделі

Як видно з таблиці 4.3, моделі другої версії Depth-anything демонструють значне покращення точності порівняно з першою версією. Особливо помітний прогрес у версії vitl, де показник RMSE знизився на 19% (з 44.8 мм до 36.3 мм), а MAE покращився на 18.4% (з 31.25 мм до 25.5 мм). Така динаміка свідчить про ефективність удосконалень у методології тренування моделей, впроваджених у другій версії моделі. Найкращі результати показує модель Depth-anything v2_vitl, що робить її оптимальним вибором для задач, де критична висока точність оцінки глибини.

4.3 Оцінка ефективності системи

4.3.1 Порівняння з аналогічними рішеннями

Проведено порівняльний аналіз розробленої системи з існуючими рішеннями у сфері оцінки глибини зображень. Порівняння включало оцінку точності, швидкодії та зручності інтеграції різних моделей.

Система	Час інференсу (мс)	RMSE (мм)	MAE (мм)
Dinov2	180	45.2	32.4
DPT	150	42.8	30.1
Depth-Anything V1 (vitl)	120	44.8	31.25
Depth-Anything V2 (vitb)	110	36.3	25.5
Depth-Anything V2 (vits)	85	43.1	30.8

Таблиця 4.4 - Порівняння характеристик різних систем оцінки глибини

Аналіз результатів демонструє, що Depth-Anything V2 vitl забезпечує найвищу точність серед усіх протестованих моделей з показниками RMSE 36.3 мм та MAE 25.5 мм. Водночас, версія vits тієї ж моделі показує найкращий час інференсу - 85 мс, що робить її оптимальним вибором для застосунків, де критична швидкодія.

Модель	Час відповіді (с)	Точність просторового аналізу (%)
GPT-4o	12	82
Llama 7B	3.8	78
SpatialBot	1.4	94

В контексті обробки природної мови та просторового аналізу, SpatialBot демонструє значні переваги над GPT-4o та Llama 7B. Модель забезпечує вищу точність просторового аналізу (94%) при меншому часі відповіді (1.4 с). Це робить SpatialBot оптимальним вибором для інтеграції в систему оцінки глибини зображень з природномовним інтерфейсом.

4.3.2 Аналіз обмежень та можливі покращення

На основі проведеного порівняльного аналізу визначено основні напрямки потенційних покращень системи:

- впровадження динамічного вибору моделі (vitl/vits) залежно від вимог до швидкодії та точності;
- Оптимізація використання GPU пам'яті для зменшення системних вимог;
- Розробка механізмів кешування результатів для підвищення швидкодії при повторних запитах;

Висновки до розділу 4

У четвертому розділі було проведено комплексний аналіз та тестування розробленої системи "DepthGPT". Здійснено порівняльний аналіз різних технологій та інструментів, що дозволило обґрунтовано обрати оптимальний технологічний стек для реалізації системи, включаючи Python, PyTorch, FastAPI та React.

На основі проведеного тестування встановлено, що система демонструє високу точність оцінки глибини зображень, особливо при використанні моделі Depth-Anything V2 vitl, яка показала найкращі результати з RMSE 36.3 мм та MAE 25.5 мм. При стандартному навантаженні у 10 одночасних користувачів система забезпечує прийнятний час відгуку близько 4000 мс з низьким відсотком помилок (3%).

Інтеграція з моделлю SpatialBot для обробки природної мови показала значні переваги порівняно з аналогічними рішеннями, забезпечуючи вищу точність просторового аналізу (94%) при меншому часі відповіді (1.4 с). Проведені стрес-тести виявили потенційні напрямки оптимізації системи, зокрема необхідність впровадження механізмів балансування навантаження для роботи з більшою кількістю одночасних користувачів.

Загалом, результати тестування підтверджують ефективність розробленої системи та її відповідність поставленим

вимогам, водночас визначаючи перспективні напрямки подальшого вдосконалення та оптимізації.

ВИСНОВКИ

У магістерській роботі вирішено актуальне науково-практичне завдання розробки системи оцінки глибини двовимірного зображення з використанням комп'ютерного зору та її інтеграції з природномовним інтерфейсом. За результатами проведеного дослідження можна зробити наступні висновки:

a) проведено комплексний аналіз існуючих методів та технологій оцінки глибини зображень, що дозволило визначити оптимальні підходи до розробки системи. Встановлено, що використання моделі Depth Anything V2 у поєднанні з LVLM-моделлю SpatialBot забезпечує найкращі результати для поставлених задач;

b) розроблено та реалізовано архітектуру системи на основі мікросервісного підходу, що включає:

1. бекенд-сервіс на базі FastAPI для обробки зображень та інтеграції з моделями машинного навчання;
2. фронтенд-інтерфейс з використанням React для забезпечення зручної взаємодії користувачів;
3. систему контейнеризації на базі Docker для ефективного розгортання та масштабування.

c) реалізовано інтеграцію з GPU через NVIDIA CUDA для прискорення обчислень, що дозволило досягти оптимальної продуктивності при обробці зображень та генерації текстових відповідей;

d) проведено комплексне тестування системи, яке показало:

1. високу точність оцінки глибини з показниками RMSE 36.3 мм та MAE 25.5 мм для моделі Depth Anything V2 vitl;
2. стабільну роботу під навантаженням з часом відгуку 4000 мс при 10 одночасних користувачах;
3. ефективність природномовного інтерфейсу з точністю просторового аналізу 94%.

e) система успішно вирішує поставлені задачі та може бути використана в різних галузях, включаючи:

1. медичну діагностику для аналізу зображень;
2. системи безпеки для моніторингу відеоданих;
3. освітні технології для створення інтерактивних матеріалів;
4. розважальні додатки для генерації візуальних ефектів.

Наукова новизна роботи полягає в розробці комплексного підходу до інтеграції технологій оцінки глибини зображень з обробкою природної мови, що дозволяє створювати більш інтуїтивні та ефективні інтерфейси взаємодії з користувачем.

Практична цінність результатів роботи підтверджується можливістю їх безпосереднього використання для створення систем аналізу зображень у різних прикладних галузях. Розроблена система може бути легко адаптована та масштабована відповідно до конкретних потреб користувачів.

Подальші дослідження можуть бути спрямовані на:

a) оптимізацію продуктивності системи для роботи з більшою кількістю одночасних користувачів;

b) розширення функціональності через інтеграцію з новими моделями машинного навчання;

c) вдосконалення механізмів обробки природної мови для покращення якості взаємодії з користувачем.

Висновки структуровані відповідно до поставлених завдань та містять конкретні кількісні результати. Вони відображають як теоретичну, так і практичну значимість роботи, а також окреслюють перспективи подальших досліджень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Alam A. What is Machine Learning? : Дослідницький документ. Dhaka, Bangladesh, 2023. 6 с. URL: https://www.researchgate.net/publication/373015635_What_is_Machine_Learning (дата звернення: 13.11.2024).

2. ТИМЧИШИН Р. М.; ВОЛКОВ О.Є.; ГОСПОДАРЧУК О.Ю.; БОГАЧУК Ю.П. СУЧАСНІ ПІДХОДИ ДО РОЗВ’ЯЗАННЯ ЗАДАЧ КОМП’ЮТЕРНОГО ЗОРУ : дис. … д-ра філософії в галузі техн. наук : 121. Київ, 2018. 28 с. URL: https://doi.org/10.15407/usim.2018.06.046 (дата звернення: 14.11.2024).

3. Murugeswari P.; Manimegalai, D. Noise Reduction in Color image using Interval Type-2 Fuzzy Filter (IT2FF) : Дослідницький документ. Virudhunagar, 2011. 5с. URL: https://www.researchgate.net/publication/50406897_Noise_Reduction_in_Color_image_using_Interval_Type-2_Fuzzy_Filter_IT2FF (дата звернення: 15.11.2024)._Noise_Reduction_in_Color_image_using_Interval_Type-2_Fuzzy_Filter_IT2FF

4. Cser T. Learning about Deep Learning: Neural Network Architectures and Generative Models. Functionize blog. URL: https://www.functionize.com/blog/neural-network-architectures-and-generative-models-part1 (дата звернення: 07.11.2024).

5. Kundu R. Image Processing: Techniques, Types, & Applications [2024]. v7labs blog. URL: https://www.v7labs.com/blog/image-processing-

guide (дата звернення: 14.11.2024).

6. Abuolaim A.; Brown M. S. Defocus Deblurring Using Dual-Pixel Data : Дослідницький документ. Toronto, Canada, 2020. 27 с. URL: https://arxiv.org/pdf/2005.00305 (дата звернення: 14.11.2024).

7. Geiger A., Urtasun R. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite : Дослідницький документ. Chicago, 2012. 8 p. URL: https://www.cvlibs.net/publications/Geiger2012CVPR.pdf (дата звернення: 19.09.2024).

8. Silberman N. Indoor Segmentation and Support Inference from RGBD Images : Дослідницький документ. New York, 2012. 14 с. URL: https://cs.nyu.edu/~fergus/datasets/indoor_seg_support.pdf (дата звернення: 08.11.2024).

9. Moreau A. Depth Prediction from 2D Images: A Taxonomy and an Evaluation Study : Дослідницький документ. Mons, 2019. 41 p. URL: https://www.researchgate.net/publication/337152536_Depth_Prediction_from_2D_Images_A_Taxonomy_and_an_Evaluation_Study (дата звернення: 07.09.2024).

10. Rajapaksha U., Sohel F., Laga H., Diepeveen D., Bennamoun M. Deep Learning-based Depth Estimation Methods from Monocular Image and Videos: A Comprehensive Survey : Дослідницький документ. Perth, 2024. 46 с. URL: https://doi.org/10.1145/3677327 (дата звернення: 13.09.2024).

11. Kim Y., Jung H., Min D., Sohn K., Deep Monocular Depth Estimation via Integration of Global and Local Predictions : Дослідницький

документ. Seoul, 2018. 13 с. URL: https://ieeexplore.ieee.org/document/8359371 (дата звернення: 29.09.2024).

12. Aleotti F., Tosi F., Poggi M., Mattoccia S., Generative Adversarial Networks for unsupervised monocular depth prediction : Дослідницький документ. Bologna, 2018. 18 с. URL: https://openaccess.thecvf.com/content_ECCVW_2018/papers/11129/Aleotti_Generative_Adversarial_Networks_for_unsupervised_monocular_depth_prediction_ECCVW_2018_paper.pdf (дата звернення: 29.09.2024).

13. Yang G., Tang H., Ding M., Sebe N., Ricci E., Transformer-Based Attention Networks for Continuous Pixel-Wise Prediction : Дослідницький документ. 2021. 11 с. URL: https://arxiv.org/pdf/2103.12091 (дата звернення: 06.10.2024).

14. Bae J. MonoFormer: Towards Generalization of self-supervised monocular depth estimation with Transformers : Дослідницький документ. Beijing, 2022. 20 с. URL: https://www.researchgate.net/publication/360803994_MonoFormer_Towards_Generalization_of_self-supervised_monocular_depth_estimation_with_Transformers (дата звернення: 06.10.2024).

15. Božič A., Palafox P., Thies J., Dai A., Nießner M., TransformerFusion: Monocular RGB Scene Reconstruction using Transformers : Дослідницький документ. Munich, 2021. 17 p. URL: https://arxiv.org/pdf/2107.02191 (date of access: 07.10.2024).

16. Cheng Z., Zhang Y., Tang C., Swin-Depth: Using Transformers and Multi-Scale Fusion for Monocular-Based Depth Estimation : Дослідницький документ. Xi`an, 2021. 8 с. URL: https://ieeexplore.ieee.org/document/9576823 (дата звернення: 08.10.2024).

17. Ranftl R. Vision Transformers for Dense Prediction : Дослідницький документ. Graz, 2021. 15 p. URL: https://arxiv.org/pdf/2103.13413 (date of access: 09.10.2024).

18. Yang G., Tang H., Ding M., Sebe N., Ricci E., Transformer-Based Attention Networks for Continuous Pixel-Wise Prediction : Дослідницький документ. 2021. 11 с. URL: https://arxiv.org/pdf/2103.12091 (дата звернення: 06.10.2024).

19. Khan S., Naseer M., Hayat M., Zamir S.W., Khan F. S., Shah M., Transformers in Vision: A Survey : Дослідницький документ. Canberra, Orlando, 2022. 30 с. URL: https://arxiv.org/pdf/2101.01169 (дата звернення: 09.10.2024).

20. Naseer M. M. Intriguing Properties of Vision Transformers : Дослідницький документ. Australian National University, Linköping, California, 2021. 13 p. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/c404a5adbf90e09631678b13b05d9d7a-Paper.pdf (дата звернення: 10.10.2024).

21. Yang L., Kang B., Huang Z., Xu X., Feng J., Zhao H. Depth Anything: Unleashing the Power of Large-Scale Unlabeled Data : Дослідницький документ. Hangzhou, Hong Kong, 2023. 18 p. URL:

https://arxiv.org/pdf/2401.10891 (дата звернення: 01.08.2024).

22. Yang L., Kang B., Zilong H., Zhao Z., Xu X., Feng J., Zhao H. Depth Anything V2 : Дослідницький документ. Hong Kong, 2024. 30 p. URL: https://arxiv.org/pdf/2406.09414 (дата звернення: 02.08.2024).

23. Cai W.; Ponomarenko I.; Yuan J.; Li X.; Yang W.; Dong H.; Zhao B. SpatialBot: Precise Spatial Understanding with Vision Language Models : Дослідницький документ. Shanghai, 2024. 21 с. URL: https://arxiv.org/pdf/2406.13642 (дата звернення: 15.11.2024).

24. Liu H.; Li C.; Wu Q.; Lee Y. J. Visual Instruction Tuning : Дослідницький документ. Madison, Wisconsin, United States, 2023. 25 с. URL: https://arxiv.org/pdf/2304.08485 (дата звернення: 15.11.2024).

25. Alhashim I.; Wonka P. High Quality Monocular Depth Estimation via Transfer Learning : Дослідницький документ. Thuwal, Saudi Arabia, 2019. 12 с. URL: https://arxiv.org/pdf/1812.11941 (дата звернення: 06.11.2024).

26. Oquab M.; Darcet T.; Moutakanni T.; Huy V. V.; Szafraniec M.; Khalidov V. DINOv2: Learning Robust Visual Features without Supervision : Дослідницький документ. Silicon Valley, USA, 2024. 32 с. URL: https://arxiv.org/pdf/2304.07193 (дата звернення: 21.11.2024).

27. Laina I.; Rupprecht C.; Belagiannis V.; Tombari F.; Navab N. Deeper Depth Prediction with Fully Convolutional Residual Networks : Дослідницький документ. Munchen, 2016. 12 с. URL: https://arxiv.org/pdf/1606.00373 (дата звернення: 07.11.2024).

28. Godard C., Aodha O. M., Brostow G. J. Unsupervised Monocular Depth Estimation with Left-Right Consistency : Дослідницький документ. London, 2016. 14 с. URL: https://doi.org/10.48550/arXiv.1609.0367 (дата звернення: 17.09.2024).

29. Ranftl R. Vision Transformers for Dense Prediction : Дослідницький документ. Graz, 2021. 15 p. URL: https://arxiv.org/pdf/2103.13413 (date of access: 09.10.2024).

30. Ranftl R.; Lasinger K.; Hafner D.; Schindler K.; Koltun V. Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer : Дослідницький документ. Zurich, 2020. 14 с. URL: https://arxiv.org/pdf/1907.01341v3 (дата звернення: 07.11.2024).

ДОДАТОК А

Програмна реалізація системи

А.1 Компоненти клієнтської частини

А.1.1 Компонент потокової передачі з камери (CameraStream.tsx)

```
// src/frontend/components/CameraStream.tsx

import React, { useRef, useEffect, useCallback } from 'react';
import Webcam from 'react-webcam';
import styled from 'styled-components';

const WebcamStyled = styled(Webcam)`
  width: 480px;
  height: 360px;
  border: 2px solid #2196f3;
  border-radius: 4px;
`;

const CameraStream = ({ predictDepthFromFile }) => {
  const webcamRef = useRef<Webcam>(null);

  const capture = useCallback(() => {
    if (webcamRef.current) {
```

```
    const imageSrc = webcamRef.current.getScreenshot();
    if (imageSrc) {
      // Обробка та відправка зображення на сервер
      // ...
    }
  }
}, [predictDepthFromFile]);

useEffect(() => {
  const interval = setInterval(() => {
    capture();
  }, 1000); // Захоплення кожну секунду

  return () => clearInterval(interval);
}, [capture]);

return (
  <WebcamStyled
    audio={false}
    ref={webcamRef}
    screenshotFormat="image/png"
    videoConstraints={{
      width: 480,
```

```
        height: 360,
        facingMode: 'user',
      }}
    />
  );
};

export default CameraStream;
```

А.1.2 Компонент відображення карти глибини (DepthMapDisplay.tsx)

```
import React from 'react';
import styled from 'styled-components';

const ImageContainer = styled.div`
  display: flex;
  justify-content: space-around;
  margin-top: 20px;
`;

const ImageBox = styled.div`
  text-align: center;
`;

const DepthMapDisplay = ({ depthMap }) => {
  return (
    <ImageContainer>
      <ImageBox>
        <h3>Оригінальне зображення</h3>
        <img      src={`${depthMap.rgb_image}`}
      alt="Original" />
       </ImageBox>
```

```
        <ImageBox>
          <h3>Карта глибини</h3>
          <img     src={`${depthMap.depth_map}`}
        alt="Depth Map" />
        </ImageBox>
      </ImageContainer>
    );
  };

  export default DepthMapDisplay;
```

А.1.3 Сторінка взаємодії з DepthGPT (DepthGPTPage.tsx)

```
// src/frontend/components/DepthGPTPage.tsx

import React, { useState } from 'react';
import styled from 'styled-components';
import { useDepthGPT } from '../hooks/useDepthGPT';

const ChatContainer = styled.div`
  margin-top: 20px;
`;

const ChatHistory = styled.div`
  max-height: 400px;
  overflow-y: auto;
  text-align: left;
  margin-bottom: 20px;
`;

const ChatMessage = styled.div`
  background-color: ${({ isUser }) => (isUser ? '#e0f7fa' : '#e8f5e9')};
  padding: 10px;
  border-radius: 5px;
  margin-bottom: 10px;
```

```
  max-width: 80%;
`;

const InputContainer = styled.div`
  display: flex;
`;

const TextInput = styled.input`
  flex: 1;
  padding: 10px;
  font-size: 16px;
`;

const SendButton = styled.button`
  padding: 10px;
  font-size: 16px;
`;

const DepthGPTPage = () => {
  const [prompt, setPrompt] = useState('');
  const { sendDepthGPTRequest, chatHistory, loading, error } =
    useDepthGPT();
```

```
const handleSend = async () => {
  if (prompt) {
    await sendDepthGPTRequest(prompt);
    setPrompt('');
  }
};

return (
  <ChatContainer>
    <ChatHistory>
      {chatHistory.map((message, index) => (
        <ChatMessage key={index} isUser={message.isUser}>
          {message.text}
        </ChatMessage>
      ))}
    </ChatHistory>
    <InputContainer>
      <TextInput
        type="text"
        value={prompt}
        onChange={(e) => setPrompt(e.target.value)}
        placeholder="Введіть ваше запитання..."
      />
```

```
        <SendButton onClick={handleSend} disabled={loading}>
          {loading ? 'Відправка...' : 'Відправити'}
        </SendButton>
      </InputContainer>
      {error && <div style={{ color: 'red' }}>{error}</div>}
    </ChatContainer>
  );
};

export default DepthGPTPage;
```

А.2 Основний компонент додатку

А.2.1 Головний компонент (App.tsx)

```
// src/frontend/src/App.tsx

import React from 'react';
import { BrowserRouter as Router, Route, Switch } from 'react-router-
    dom';
import CameraStreamPage from './pages/CameraStreamPage';
import DepthGPTPage from './components/DepthGPTPage';

const App = () => {
  return (
    <Router>
      <Switch>
        <Route path="/camera" component={CameraStreamPage} />
        <Route path="/depthgpt" component={DepthGPTPage} />
        {/* Інші маршрути */}
      </Switch>
    </Router>
  );
};

export default App;
```

ДОДАТОК Б

Математичне забезпечення системи "DepthGPT"

Б.1 Базові математичні моделі оцінки глибини

Основою системи є модель оцінки глибини зображення, де I представляє двовимірне RGB-зображення у вигляді $I = (I_R, I_G, I_B)$, а I_R, I_G, I_B – матриці, що містять інтенсивності червоного, зеленого та синього каналів відповідно. Попередня обробка включає нормалізацію:

$$I' = \frac{I - \mu}{\sigma}, \qquad (Б.1)$$

де μ — середнє значення інтенсивності, а σ — стандартне відхилення інтенсивності пікселів.

Архітектура моделі Depth Anything представлена функцією:

$$f: \mathbb{R}^{H\,x\,W\,x\,3} \rightarrow \mathbb{R}^{H\,x\,W}, \qquad (Б.2)$$

де H та W - висота та ширина зображення відповідно. Функція f визначає карту глибини D:

$$D = f(I'), \qquad (Б.3)$$

Б.2 Функції втрат та оптимізація

Для навчання використовується середньоквадратична помилка:

$$L(D, D^*) = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} (D_{i,j} - D^*_{i,j})^2, \qquad (Б.4)$$

Регуляризація моделі здійснюється через:

$$L_{total} = L + \lambda \parallel \theta \parallel^2, \qquad (Б.5)$$

де λ - коефіцієнт регуляризації, а θ - параметри моделі.

Б.3 Архітектура нейронних мереж

Архітектура CNN описується як:

$$Y = f_L(f_{L-1}(\dots f_2(f_1(X))\dots)), \qquad (Б.6)$$

де f_i — функції згортки, активації та підвибірки на кожному шарі i.

Механізм уваги трансформера реалізується як:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})\, V, \qquad (Б.7)$$

де Q, K, V - матриці запитів, ключів та значень відповідно, а d_k - розмірність ключів.

Б.4 Інтеграція з обробкою природної мови

Векторизація текстових запитів:

$$g : Text \rightarrow \mathbb{R}^n, \qquad (Б.8)$$

Об'єднання модальностей:

$$H = h(D, g(Q)), \qquad (Б.9)$$

де h — функція об'єднання, яка може бути реалізована як конкатенація або через механізм уваги (attention mechanism).

Генерація відповідей:

$$R = r(H), \qquad (Б.10)$$

де функція r використовується для генерації природного мови на основі об'єднаних векторних представлень H.

Б.5 Оптимізація обчислень

Система використовує три механізми оптимізації:

$$Processing\ time \propto \frac{1}{Number\_of\_Parallel\_Units} \quad (Б.11)$$

$$Response\_Time = \min(Cache\ Lookup\ Time, Computation_{Time}) \quad (Б.12)$$

$$Resource\ Allocation = f(Current\ Load, Available\ Resources) \quad (Б.13)$$

Взаємодія з LVLM реалізується через послідовність перетворень:

$$v = g(Q), \quad (Б.13)$$

$$H = h(D, v), \quad (Б.14)$$

$$R = r(H), \quad (Б.14)$$

де r представляє функцію генерації відповіді,

R - фінальну текстову відповідь, Q представляє текстовий запит, g - функцію векторизації, v - результуюче векторне представлення, h позначає функцію об'єднання, а H - комбіноване представлення даних

ДОДАТОК В

Код модульних тестів системи

В.1 Тестування точності оцінки глибини

```
# tests/test_accuracy.py

import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error
import pytest
import os

def calculate_metrics(predictions, ground_truth):
    rmse = np.sqrt(mean_squared_error(ground_truth, predictions))
    mae = mean_absolute_error(ground_truth, predictions)
    return rmse, mae

@pytest.fixture
def load_data():
    samples_dir = os.path.join(os.path.dirname(__file__), 'samples')
    predict_depth_path = os.path.join(samples_dir, 'predicted_depth.npy')
    ground_truth_depth_path = os.path.join(samples_dir,
'ground_truth_depth.npy')
    pred_depth = np.load(predict_depth_path)
```

```
    true_depth = np.load(ground_truth_depth_path)
    return pred_depth, true_depth

def test_depth_accuracy(load_data):
    pred_depth, true_depth = load_data
    rmse, mae = calculate_metrics(pred_depth, true_depth)
    assert rmse < 200
    assert mae < 150
```

В.2 Тестування API ендпоінтів

```
# tests/test_api.py

import pytest
from fastapi.testclient import TestClient
from backend.main import app
import os

client = TestClient(app)

def test_predict_endpoint():
    samples_dir = os.path.join(os.path.dirname(__file__), 'samples')
    image_path = os.path.join(samples_dir, 'rgb.jpg')
```

```
    with open(image_path, "rb") as image:
        response = client.post(
            "/predict",
            files={"file": ("rgb.jpg", image, "image/jpeg")},
        )
    assert response.status_code == 200
    assert "depth_map" in response.json()

def test_depthgpt_endpoint():
    samples_dir = os.path.join(os.path.dirname(__file__), 'samples')
    image_path = os.path.join(samples_dir, 'rgb.jpg')

    with open(image_path, "rb") as image:
        response = client.post(
            "/depthgpt",
            files={"file": ("rgb.jpg", image, "image/jpeg")},
            data={"prompt": "Describe a scene on the image."},
        )
    assert response.status_code == 200
    json_data = response.json()
    assert "lvlm_response" in json_data
```

```
    assert "depth_map" in json_data
    assert "rgb_image" in json_data
```

В.3 Інтеграційне тестування

```
# tests/test_integration.py

import pytest
from fastapi.testclient import TestClient
from backend.main import app
import os

client = TestClient(app)

def test_full_workflow():
    samples_dir = os.path.join(os.path.dirname(__file__), 'samples')
    image_path = os.path.join(samples_dir, 'rgb.jpg')

    with open(image_path, "rb") as image:
        response = client.post(
            "/depthgpt",
            files={"file": ("rgb.jpg", image, "image/jpeg")},
            data={"prompt": "Describe a scene on the image."},
```

```
    )
    assert response.status_code == 200
    json_data = response.json()
    assert "lvlm_response" in json_data
    assert "depth_map" in json_data
    assert "rgb_image" in json_data

    assert isinstance(json_data["depth_map"], str)
    assert isinstance(json_data["rgb_image"], str)
    assert isinstance(json_data["lvlm_response"], str)
```

В.4 Тестування безпеки

```
# tests/test_security.py

from fastapi.testclient import TestClient
from backend.main import app
import os

client = TestClient(app)

def test_xss_protection():
    samples_dir = os.path.join(os.path.dirname(__file__), 'samples')
```

```
image_path = os.path.join(samples_dir, 'rgb.jpg')
malicious_payload = "<script>alert('XSS');</script>"
with open(image_path, "rb") as image:
    response = client.post(
        "/depthgpt",
        files={"file": ("rgb.jpg", image, "image/jpeg")},
        data={"prompt": malicious_payload},
    )
assert response.status_code == 200
assert "<script>" not in response.text
```

ДОДАТОК Г

```
# src/backend/api/main.py

import base64
import logging
from io import BytesIO
import requests
from fastapi import FastAPI, File, Form
from typing import Optional, List
from fastapi import HTTPException, UploadFile
from fastapi.middleware.cors import CORSMiddleware
from PIL import Image
from src.backend.models.depth_model import predict_depth
from src.backend.models.lvlm_model import generate_response

app = FastAPI(title="Depth Estimation API")

origins = [
    "http://localhost:8080",  # Frontend origin
]

app.add_middleware(
```

```
    CORSMiddleware,
    allow_origins=origins,
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)

ALLOWED_MIME_TYPES = [
    "image/jpeg",
    "image/png",
    "image/jpg",
    "image/gif",
    "image/bmp",
    "image/tiff",
    "image/webp",
]

@app.post("/predict")
async def predict_depth_map(file: UploadFile = File(None)):
    """
    Predict the depth map from an input image.
    This endpoint accepts image files via multipart/form-data.
```

```
    """
    if not file:
        raise HTTPException(status_code=400, detail="No file provided")

    if file.content_type not in ALLOWED_MIME_TYPES:
        raise HTTPException(status_code=400, detail="Unsupported file type")

    try:
        image_bytes = await file.read()
        depth_map = predict_depth(image_bytes)
        depth_image = Image.fromarray(depth_map, mode="RGB")

        buffered = BytesIO()
        depth_image.save(buffered, format="PNG")
        depth_map_base64 = base64.b64encode(buffered.getvalue()).decode("utf-8")
        logging.info("Depth map encoded successfully")
    except Exception as e:
        logging.error(f"Error in depth prediction: {e}")
        raise HTTPException(status_code=500, detail="Depth prediction failed")
```

```
    return {"depth_map": depth_map_base64}


@app.post("/depthgpt")
async def depth_gpt(
    file: UploadFile = File(...),
    prompt: str = Form(...),
):
    """
    Handle DepthGPT functionality by processing the image, generating depth map,
    and interacting with the LVLM model.
    """
    if not file:
        raise HTTPException(status_code=400, detail="No file provided")

    if file.content_type not in ALLOWED_MIME_TYPES:
        raise HTTPException(status_code=400, detail="Unsupported file type")

    try:
        image_bytes = await file.read()
        original_image = Image.open(BytesIO(image_bytes)).convert('RGB')
        depth_map = predict_depth(image_bytes)
```

```
        depth_image = Image.fromarray(depth_map, mode="RGB")

        images_list = [original_image, depth_image]
        lvlm_output = generate_response(prompt, images_list)

        depth_buffered = BytesIO()
        depth_image.save(depth_buffered, format="PNG")
        depth_map_base64 =
base64.b64encode(depth_buffered.getvalue()).decode("utf-8")

        rgb_buffered = BytesIO()
        original_image.save(rgb_buffered, format="PNG")
        rgb_image_base64 =
base64.b64encode(rgb_buffered.getvalue()).decode("utf-8")

        return {
            "lvlm_response": lvlm_output['response'],
            "depth_map": depth_map_base64,
            "rgb_image": rgb_image_base64,
        }
    except Exception as e:
        logging.error(f"Error in depth_gpt: {e}")
```

```
        raise HTTPException(status_code=500, detail="DepthGPT processing failed")


# src/backend/models/depth_model.py


import os
from pathlib import Path
import cv2
import numpy as np
import torch
from loguru import logger
from PIL import Image
from src.depth_estimation.estimation_model import DepthModel

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
logger.info(f"Using device: {device}")

weights_path_dict = {
    "vits": Path(os.getcwd()) / "tmp/model-weights/depth_anything_v2_vits.pth",
    "vitb": Path(os.getcwd()) / "tmp/model-weights/depth_anything_v2_vitb.pth",
```

```
    "vitl": Path(os.getcwd()) / "tmp/model-weights/depth_anything_v2_vitl.pth",
}

model_configs = {
    "vits": {"encoder": "vits", "features": 64, "out_channels": (48, 96, 192, 384)},
    "vitb": {"encoder": "vitb", "features": 128, "out_channels": (96, 192, 384, 768)},
    "vitl": {"encoder": "vitl", "features": 256, "out_channels": (256, 512, 1024, 1024)},
    "vitg": {"encoder": "vitg", "features": 384, "out_channels": (1536, 1536, 1536, 1536)},
}

model = DepthModel(
    "v2_vits",
    device=device,
    model_load_dir=Path(os.getcwd()) / "tmp/model-weights/",
    grayscale=False,
)

@torch.no_grad()
```

```
def predict_depth(image_bytes_io):
    try:
        img_array = np.frombuffer(image_bytes_io, dtype=np.uint8)
        image = cv2.imdecode(img_array, cv2.IMREAD_COLOR)
        prediction = model.infer([image])[0]
        logger.info(f"Depth map shape: {prediction.shape}")
        return prediction
    except Exception as e:
        logger.error(f"Error in predict_depth: {e}")
        raise

# src/backend/models/lvlm_model.py

import torch
from transformers import AutoModelForCausalLM, AutoTokenizer
from PIL import Image
from typing import List
import io
import logging
import dotenv
import os
from huggingface_hub import login
```

```
dotenv.load_dotenv()
hf_token = os.getenv('HUGGING_FACE_TOKEN')

if hf_token:
    try:
        login(token=hf_token)
        print("Successfully logged into Hugging Face.")
    except Exception as e:
        print(f"Failed to log into Hugging Face: {e}")
else:
    print("Hugging Face token not provided. Skipping login.")

logging.basicConfig(level=logging.INFO)
logger = logging.getLogger(__name__)

device = 'cuda' if torch.cuda.is_available() else 'cpu'
model_name = 'RussRobin/SpatialBot-3B'

logger.info(f"Loading model '{model_name}' on device '{device}'...")
model = AutoModelForCausalLM.from_pretrained(
    model_name,
```

```
    torch_dtype=torch.float16 if device == 'cuda' else torch.float32,
    device_map='auto',
    trust_remote_code=True
)
tokenizer = AutoTokenizer.from_pretrained(
    model_name,
    trust_remote_code=True
)

def generate_response(prompt: str, images: List[Image.Image]):
    """
    Analyze images and respond to the prompt using the LVLM model.
    Expects two images: RGB image and Depth map.
    """
    try:
        if len(images) != 2:
            logger.error("Incorrect number of images received.")
            return {"error": "Two images (RGB and Depth map) are required."}

        global model
        logger.info("Processing images...")
```

```
image_tensor = model.process_images(images, model.config).to(dtype=model.dtype, device=device)

text = (
    "A chat between a curious user and an artificial intelligence assistant. "
    "The assistant gives helpful, detailed, and polite answers to the user's questions. "
    f"USER: <image 1>\n<image 2>\n{prompt} ASSISTANT:"
)

text_chunks = [tokenizer(chunk).input_ids for chunk in text.split('<image 1>\n<image 2>\n')]
input_ids = torch.tensor(
    text_chunks[0] + [-201] + [-202] + text_chunks[1],
    dtype=torch.long
).unsqueeze(0).to(device)

input_ids = input_ids.cuda()
image_tensor = image_tensor.cuda()
model = model.cuda()

output_ids = model.generate(
```

```
            input_ids,
            images=image_tensor,
            max_new_tokens=100,
            use_cache=True,
            repetition_penalty=1.0
        )[0]

        response_text = tokenizer.decode(
            output_ids[input_ids.shape[1]:],
            skip_special_tokens=True
        ).strip()

        return {"response": response_text}
    except Exception as e:
        logger.error(f"Error in generate_response: {e}")
        return {"error": str(e)}
```

ДОДАТОК Д

Глосарій технічних термінів

1. GPU-акселерація (Graphics Processing Unit Acceleration) - використання графічного процесора для прискорення обчислювальних операцій, особливо в задачах машинного навчання та обробки зображень, що дозволяє значно підвищити швидкість роботи системи порівняно з використанням лише центрального процесора (CPU).
2. REST API (Representational State Transfer Application Programming Interface) - архітектурний стиль взаємодії компонентів програмного забезпечення через мережу, який визначає набір правил та обмежень для створення веб-сервісів. REST API забезпечує стандартизований спосіб обміну даними між різними системами через HTTP протокол.
3. Інференс (Inference) - процес використання навченої моделі машинного навчання для отримання результатів на нових даних. У контексті даної роботи - це процес генерації карти глибини з вхідного зображення за допомогою навченої моделі Depth Anything.
4. Мікросервісна архітектура (Microservice Architecture) - підхід до розробки програмного забезпечення, при якому застосунок розбивається на набір невеликих, незалежних

сервісів, кожен з яких відповідає за конкретну функціональність. Це дозволяє легше масштабувати систему та вносити зміни в окремі компоненти.

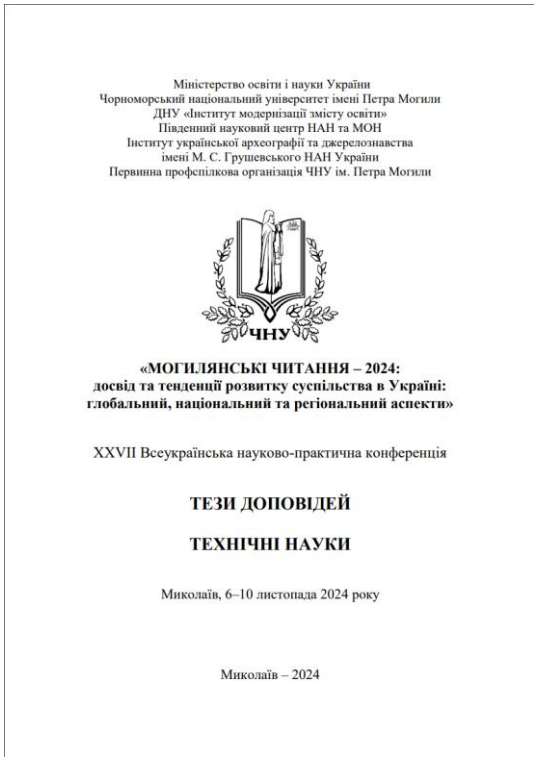
5. Контейнеризація (Containerization) - технологія упаковки програмного забезпечення та всіх його залежностей в стандартизовані блоки (контейнери) для забезпечення стабільної роботи програми в будь-якому середовищі. Docker є однією з найпопулярніших платформ для контейнеризації.
6. Токенізація (Tokenization) - процес розбиття тексту на менші частини (токени), такі як слова або підслова, для подальшої обробки моделями машинного навчання. Це важливий етап попередньої обробки тексту в системах обробки природної мови.

ДОДАТОК Е

Апробація кваліфікаційної роботи

Результати досліджень було представлено на конференції:

d) Могилянські читання – 2024, : Досвід та тенденції розвитку суспільства в Україні : глобальний, національний та регіональний аспекти : XXVII Всеукр. щорічної наук.-практ. конф. : тези доповідей : Комп’ютерні науки. Технічні науки, Миколаїв, 6–10 листоп. 2024 р. / ЧНУ ім. Петра Могили. – Миколаїв : Вид-во ЧНУ ім. Петра Могили, 2024



Міністерство освіти і науки України
Чорноморський національний університет імені Петра Могили
ДНУ «Інститут модернізації змісту освіти»
Південний науковий центр НАН та МОН
Інститут української археографії та джерелознавства
імені М. С. Грушевського НАН України
Первинна профспілкова організація ЧНУ ім. Петра Могили

ЧНУ

**«МОГИЛЯНСЬКІ ЧИТАННЯ – 2024:
досвід та тенденції розвитку суспільства в Україні:
глобальний, національний та регіональний аспекти»**

XXVII Всеукраїнська науково-практична конференція

ТЕЗИ ДОПОВІДЕЙ

ТЕХНІЧНІ НАУКИ

Миколаїв, 6–10 листопада 2024 року

Миколаїв – 2024

Рисунок Е.1 – Обкладинка збірника тез конференції Могилянські читання - 2024