МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно- обчислювальних систем і управління

ШТИНДА Віктор Володимирович

Веб-базована система розпізнавання їжі по фото засобами комп'ютерного зору / Web-based System for Food Recognition from Photos Using Computer Vision

спеціальність: 122 - Комп'ютерні науки
освітньо-професійна програма - Комп’ютерні науки

Кваліфікаційна робота

Виконав студент групи КН-42
В. В. Штинда

Науковий керівник:
к.т.н., доцент, П. Є. Биковий

Кваліфікаційну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
М. П. Комар

ТЕРНОПІЛЬ - 2024

Факультет комп'ютерних інформаційних технологій
Кафедра інформаційно-обчислювальних систем і управління
Освітній ступінь «бакалавр»
Спеціальність: 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ М.П. Комар
«____» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
ШТИНДА Віктор Володимирович
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи: Веб-базована система розпізнавання їжі по фото засобами комп'ютерного зору / Web-based System for Food Recognition from Photos Using Computer Vision
керівник роботи Биковий Павло Євгенович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 12 грудня 2023 р. №753.
2. Строк подання студентом закінченої кваліфікаційної роботи 15 травня 2024 р.
3.Вихідні дані до кваліфікаційної роботи: завдання на кваліфікаційну роботу студента, наукові статті, технічна література.
4. Основні питання, які потрібно розробити:
- провести опис предметної області;
- провести аналіз існуючих рішень;
- зробити постановку задачі дослідження;
- розробити загальну структуру проєктованої системи;
- розробити загальний алгоритм роботи системи;
- зібрати за підготувати набір даних;
- навчити модель розпізнавання їжі;
- розробити інтуїтивно зрозумілу веб-базовану систему;
- провести розпізнавання зображень їжі .
- провести тестування розробленої веб-базованої системи.

5. Перелік графічного матеріалу в роботі:
- структурна схема системи;
- діаграма потоків даних;
- блок-схема алгоритму роботи системи.

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 12 грудня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Затвердження теми кваліфікаційної роботи, ознайомлення з літературними джерелами та складання плану роботи	до 01.01.2024 р.	
2	Написання 1 розділу кваліфікаційної роботи	до 01.03.2024 р.	
3	Написання 2 розділу кваліфікаційної роботи	до 01.04.2024 р.	
4	Написання 3 розділу кваліфікаційної роботи	до 01.05.2024 р.	
5	Представлення попереднього варіанту кваліфікаційної роботи, перевірка та внесення змін керівником	до 15.05.2024 р.	
6	Опрацювання зауважень та представлення завершеного варіанту кваліфікаційної роботи. Підготовка супроводжуючих документів.	до 20.05.2024 р.	
7	Перевірка кваліфікаційної роботи на оригінальність тексту у системі "Unicheck"	до 10.06.2024 р.	
8	Оформлення кваліфікаційної роботи та отримання допуску до захисту	до 14.06.2024 р.	
9	Подання кваліфікаційної роботи до захисту на засіданні атестаційної комісії	до 14.06.2024 р.	

Студент _____ В.В. Штинда
(підпис)

Керівник проекту _____ П.Є. Биковий
(підпис)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Веб-базована система розпізнавання їжі по фото засобами комп'ютерного зору» на здобуття освітнього ступеня «бакалавр» зі спеціальності 122 «Комп’ютерні науки» освітньої програми «Комп’ютерні науки» написана обсягом в 64 сторінки і містить 14 ілюстрацій, дві таблиці, два додатки та 29 використаних джерел.

Метою роботи є створення веб-базованої системи, яка дасть можливість автоматично ідентифікувати їжу на зображеннях, використовуючи методи комп’ютерного зору, глибокого навчання та нейронних мереж.

Методи та технології використані у роботі, включають комп'ютерний зір, глибоке навчання з використанням згорткових нейронних мереж (CNN) та методу YOLO для розробки моделей розпізнавання їжі, методи підготовки зображень до аналізу, такі як масштабування, нормалізація та сегментація. Для інтеграції з веб-технологіями було використано HTML, CSS та JavaScript для створення зручного інтерфейсу. Для реалізації моделей машинного навчання та роботи з зображеннями використовувалися бібліотеки Python (TensorFlow, Keras, OpenCV).

У ході виконання кваліфікаційної роботи розроблено веб-систему для розпізнавання їжі по фото засобами комп'ютерного зору. Проведено аналіз сучасних технологій та алгоритмів розпізнавання їжі, розроблено на навчено модель розпізнавання. Інтегровано систему з веб-інтерфейсом, використовуючи HTML, CSS та JavaScript. Проведено функціональне тестування, результати якого підтвердили високу точність розпізнавання їжі, що підтверджує ефективність системи.

Результати роботи можна використовувати в ресторанах для автоматизації обробки замовлень та контролю інгредієнтів, у додатках для здорового харчування для аналізу раціону користувачів, у дослідженнях харчових звичок.

Ключові слова: РОЗПІЗНАВАННЯ ЇЖІ, МАШИННЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ, НЕЙРОННІ МЕРЕЖІ, КОМП'ЮТЕРНИЙ ЗІР, ОБРОБКА ЗОБРАЖЕНЬ.

ANNOTATION

Qualification work on the topic "Web-based food recognition system using computer vision" for obtaining a bachelor's degree in the specialty 122 "Computer Science" of the educational program "Computer Science" is written in 64 pages and contains 14 illustrations, two tables, two appendices, and 29 references.

The goal of the work is to create a web-based system that allows automatic identification of food in images using computer vision, deep learning, and neural networks.

The methods and technologies used in the work include computer vision, deep learning with convolutional neural networks (CNN) and the YOLO method for developing food recognition models, image preparation methods for analysis such as scaling, normalization, and segmentation. HTML, CSS, and JavaScript were used for integration with web technologies to create a user-friendly interface. Python libraries (TensorFlow, Keras, OpenCV) were used to implement machine learning models and image processing.

During the execution of the qualification work, a web-based system for food recognition in photos using computer vision was developed. An analysis of modern technologies and food recognition algorithms was carried out, and a recognition model was developed and trained. The system was integrated with a web interface using HTML, CSS, and JavaScript. Functional testing was conducted, and the results confirmed the high accuracy of food recognition, demonstrating the system's effectiveness.

The results of the work can be used in restaurants to automate order processing and ingredient control, in health food applications to analyze users' diets, and in studies of eating habits.

Keywords: FOOD RECOGNITION, MACHINE LEARNING, CLASSIFICATION, NEURAL NETWORKS, COMPUTER VISION, IMAGE PROCESSING.

ЗМІСТ

Зміст

ВСТУП

У сучасному світі зростає інтерес до застосування технологій штучного інтелекту для вирішення повсякденних завдань. Одним з таких завдань є розпізнавання їжі на фотографіях за допомогою комп'ютерного зору. Це стає особливо актуальним через популярність мобільних додатків для здорового харчування, дієт та фотографування їжі.

Актуальність теми роботи полягає в необхідності автоматизації процесу розпізнавання їжі на зображеннях. В сучасних умовах, коли швидкість та точність аналізу даних мають критичне значення, автоматизація цього процесу дозволить значно підвищити ефективність і точність аналізу.

Метою даної кваліфікаційної роботи є створення веб-базованої системи, яка буде автоматично розпізнавати різні види їжі на зображеннях за допомогою комп'ютерного зору. Система повинна аналізувати фотографії їжі та точно визначати, яка їжа зображена на них.

Щоб виконати поставлене завдання потрібно:

- провести опис предметної області;
- провести аналіз існуючих рішень;
- зробити постановку задачі дослідження;
- розробити загальну структуру проєктованої системи;
- розробити загальний алгоритм роботи системи;
- зібрати та підготувати набір даних;
- навчити модель розпізнавання їжі;
- розробити інтуїтивно зрозумілу веб-базовану систему;
- провести розпізнавання зображень їжі;
- провести тестування розробленої веб-базованої системи.

Об'єктом дослідження є веб-базовані системи розпізнавання об'єктів на зображеннях. Це включає всі аспекти, пов'язані з аналізом, обробкою та інтерпретацією зображень за допомогою технологій комп'ютерного зору та машинного навчання.

Предметом дослідження є конкретні методи та алгоритми, які використовуються для розпізнавання їжі на зображеннях, включаючи алгоритми машинного навчання та глибокого навчання, обробку зображень та інтеграцію веб-технологій.

Методи дослідження включають: методи комп'ютерного зору, а саме аналіз гістограм кольорів, сегментацію зображень, використання ознак і дескрипторів. Методи глибокого навчання: згорткові нейронні мережі (CNN), метод YOLO (You Only Look Once). Методи підготовки зображень: масштабування, нормалізація, сегментація.

Функціональність системи буде полягати в автоматичному розпізнаванні різних видів їжі на фотографіях. Система буде використовувати моделі глибокого навчання для аналізу зображень, що дозволить користувачам швидко ідентифікувати їжу на фотографіях. Інтерактивний веб-інтерфейс дозволить користувачам завантажувати зображення та отримувати результати розпізнавання в реальному часі.

Можливі сфери застосування результатів включають системи контролю харчування, де автоматичне розпізнавання їжі допомагає створювати харчові щоденники для контролю раціону та підтримки здорового способу життя. У медицині такі системи можуть використовуватися для моніторингу дієти пацієнтів, що важливо для лікування хронічних захворювань, як-от діабет. У сфері обслуговування автоматизація замовлень за допомогою розпізнавання їжі на фотографіях може зменшити час обслуговування клієнтів та підвищити точність замовлень.

Результати роботи були апробовані на міжнародній науково-практичній конференції: «ПЕРСПЕКТИВНІ НАПРЯМКИ РОЗВИТКУ ЕКОНОМІКИ, ОБЛІКУ, УПРАВЛІННЯ ТА ПРАВА: ТЕОРІЯ І ПРАКТИКА» та подані до друку в «Перспективні напрямки розвитку економіки, обліку, управління та права: теорія і практика: збірник тез доповідей міжнародної науково-практичної конференції (Ізмаїл, 18 травня 2024 р.). Ізмаїл: ЦФЕНД, 2024. 63 с.».

1 АНАЛІЗ СИСТЕМ РОЗПІЗНАВАННЯ ЇЖІ ПО ФОТО

1.1 Опис предметної області

Розпізнавання їжі по фото є сучасною технологією, яка знаходить широке застосування в різних сферах, серед них є дієтологія, ресторанний бізнес, фітнес-індустрія та медицина. Основною метою цієї технології є автоматичне розпізнавання їжі на фотографіях та надання інформації про її склад. Це дозволяє користувачам більш свідомо контролювати своє харчування і приймати обґрунтовані рішення щодо своєї дієти, а також може бути корисним для людей, які страждають від харчових алергій та потребують контролювати склад свого харчування.

Система розпізнавання їжі по фотографій повинна вирішувати кілька основних завдань. Перше завдання, вона має вміти точно розпізнавати різні види їж на фотографіях, включаючи як окремі фрукти і овочі, так і складні страви.

Для реалізації цих завдань застосовуються технології, комп’ютерного зору, машинного навчання та штучного інтелекту, зокрема згорткові нейронні мережі (CNN), які використовуються для обробки зображень та розпізнавання об'єктів.

На рисунку 1.1 зображено архітектуру згорткових нейронних мереж.

Рисунок 1.1 – Архітектура CNN

Методи розпізнавання їжі можна поділити на дві великі категорії: класичні алгоритми обробки зображень та методи, що використовують глибоке навчання.

Класичні алгоритми обробки зображень включають аналіз гістограми кольорів, сегментацію зображень та використання ознак і дескрипторів. Аналіз

гістограми кольорів передбачає дослідження розподілу кольорів на зображенні для виявлення характерних ознак їжі. Наприклад, якщо ми маємо зображення фруктів, гістограма кольорів може допомогти визначити різницю між яблуком і бананом на основі їхніх кольорових характеристик. Гістограми можуть будуватися для кожного каналу кольору (R, G, B в RGB просторі), а також для інших колірних просторів, таких як HSV. Гістограма кольорів також може бути використана для порівняння різних зображень та виявлення подібності або відмінностей між ними.

Сегментація зображень полягає у розділенні зображення на сегменти, які відповідають різним об'єктам, використовуючи такі методи, як k-means кластеризація або алгоритм водорозділу. Цей процес є важливим для виявлення і розпізнавання об'єктів на зображенні. Наприклад, сегментація може бути використана для відділення фону від переднього плану або для виявлення конкретних об'єктів, таких як фрукти на столі.

K-means кластеризація: Цей метод групує пікселі на основі їх кольорових або текстурних характеристик. Спочатку вибирається кількість кластерів (k), і кожному кластеру призначаються початкові центри. Потім пікселі призначаються до найближчого центру кластера, і центри оновлюються на основі середнього значення пікселів в кожному кластері. Процес повторюється до досягнення конвергенції.

Алгоритм водорозділу: Цей метод базується на концепції топографічної карти, де інтенсивність пікселя інтерпретується як висота. Зображення сприймається як ландшафт, і водорозділи розділяють області на основі мінімумів в градієнтному зображенні. Це дозволяє виділяти об'єкти, які мають чіткі границі.

Використання ознак і дескрипторів включає в себе ідентифікацію ключових точок на зображенні і опис їхніх характеристик для подальшого розпізнавання або порівняння. Класичні підходи до виділення ознак включають використання алгоритмів SIFT (Scale-Invariant Feature Transform) і SURF (Speeded-Up Robust Features).

Метод SIFT виявляє ключові точки, які є стійкими до масштабування, обертання та змін освітлення. Для кожної ключової точки обчислюється вектор дескриптора на основі локальних градієнтів інтенсивності.

Більш швидкий метод SURF порівняно з SIFT, який також виявляє і описує ключові точки, але використовує апроксимації для прискорення обчислень.

Такі методи є фундаментальними для багатьох додатків комп'ютерного зору, включаючи розпізнавання об'єктів, відстеження руху, 3D реконструкцію та багато інших. Вигляд роботи сегментації показано на рисунку 1.2

Рисунок 1.2 – Сегментація їжі на фото [27]

Ознаки та дескриптори, такі як SIFT, SURF або HOG, використовуються для ідентифікації ключових точок на зображенні і опису їх властивостей для подальшого розпізнавання об'єктів [25].

Методи глибокого навчання, такі як згорткові нейронні мережі (CNN), значно покращили точність і ефективність розпізнавання зображень.

CNN автоматично витягують ознаки з зображень і вчаться класифікувати їх на основі великих наборів даних. Вони використовують шари згорток для виділення локальних ознак, таких як контури, текстури та інші важливі деталі. Основні компоненти CNN включають згорткові шари, які виконують згортку вхідного зображення з набором фільтрів, що витягують різні ознаки; шари підвибірки, які зменшують розмір активаційних карт, зберігаючи найбільш важливу інформацію; та повнозв'язані шари, які з'єднують усі нейрони з попереднього шару з кожним нейроном наступного шару і виконують класифікацію на основі витягнутих ознак.

Один з популярних методів у цій категорії — YOLO (You Only Look Once), який виконує одночасно розпізнавання і локалізацію об'єктів на зображенні. YOLO має унікальний підхід до детекції об'єктів. Зображення поділяється на сітку, де кожна клітинка відповідає за виявлення об'єктів, що знаходяться в ній. Кожна клітинка передбачає межі (bounding boxes) і ймовірності класів для об'єктів, що знаходяться в її межах. Кожна межа визначається координатами центра, шириною, висотою та ймовірністю наявності об'єкта. На відміну від традиційних методів, які виконують спочатку розпізнавання, а потім локалізацію, YOLO виконує ці завдання одночасно, що забезпечує високу швидкість і точність.

Переваги YOLO включають високу швидкість обробки завдяки одноразовій обробці зображення, що робить його придатним для реальних додатків, де важлива швидкість.

Крім того, завдяки своїй структурі, YOLO здатний досягати високої точності в розпізнаванні і локалізації об'єктів, що робить його одним з найпопулярніших методів у цій галузі[26]. Схематичний вигляд роботи методу YOLO зображено на рисунку 1.3.

Основні переваги класичних методів включають їхню простоту і менші вимоги до обчислювальних ресурсів. Однак, їх точність і здатність обробляти складні зображення обмежені. Методи глибокого навчання, хоча і потребують значних обчислювальних ресурсів та великих обсягів даних для тренування, значно

перевершують класичні методи в плані точності та здатності розпізнавати складні об'єкти і деталі на зображеннях.

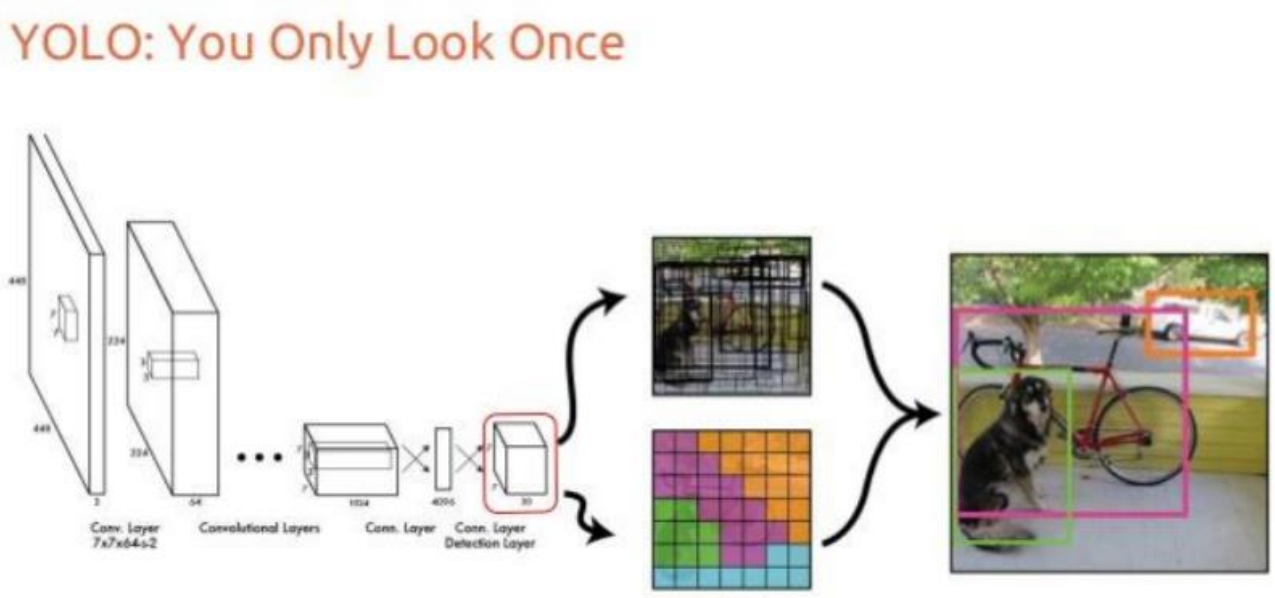

Рисунок 1.3 – Схематичний вигляд роботи методу YOLO[26]

Технології комп'ютерного зору допомагають аналізувати фотографії та виділяти на них їжу, ігноруючи фонові елементи. Використання великих баз даних зображень їжі для тренування моделей розпізнавання забезпечує високу точність та надійність системи.

Розпізнавання їжі по фото використовується у мобільних додатках, які допомагають користувачам відстежувати своє харчування, вести щоденник їжі та отримувати рекомендації щодо дієти. У ресторанах та кафе автоматичні системи розпізнавання страв у меню дозволяють швидко визначати склад та калорійність страв, а також пропонувати клієнтам здорові альтернативи. У медичній сфері це допомагає пацієнтам з дієтичними обмеженнями або хронічними захворюваннями, такими як діабет, контролювати своє харчування. Також у фітнес-індустрії ці інструменти допомагають спортсменам та людям, які дотримуються здорового способу життя, точно відстежувати вжиті калорії та поживні речовини.

Розпізнавання їжі по фотографіях є потужним інструментом, який може значно полегшити контроль за харчуванням, зробити його більш інформативним та індивідуальним. В зв’язку з швидким темпом розвитку технологій комп’ютерного

зору, штучного інтелекту та машинного навчання, ці системи стають все більш точними та доступнішими для все більшої кількості користувачів.

1.2 Аналіз існуючих рішень

На сучасному ринку існує багато систем та програм для розпізнавання їжі засобами комп'ютерного зору, з яких було обрано наступні системи та програми.

LogMeal – це Application Programming Interface (API), який надає можливість розпізнавати їжу на основі фотографій [1]. На рисунку 1.4 представлений вигляд сторінки сайту LogMeal API. Перевагою є те, що модель, яка використовується для розпізнавання їжі, демонструє високу ефективність та доволі точно визначає страву. Це досягається завдяки використанню сучасних методів глибокого навчання, таких як згорткові нейронні мережі (CNN), які забезпечують точне розпізнавання різних типів їжі завдяки автоматичному витяганню ознак зображення та класифікації на основі великих наборів даних.

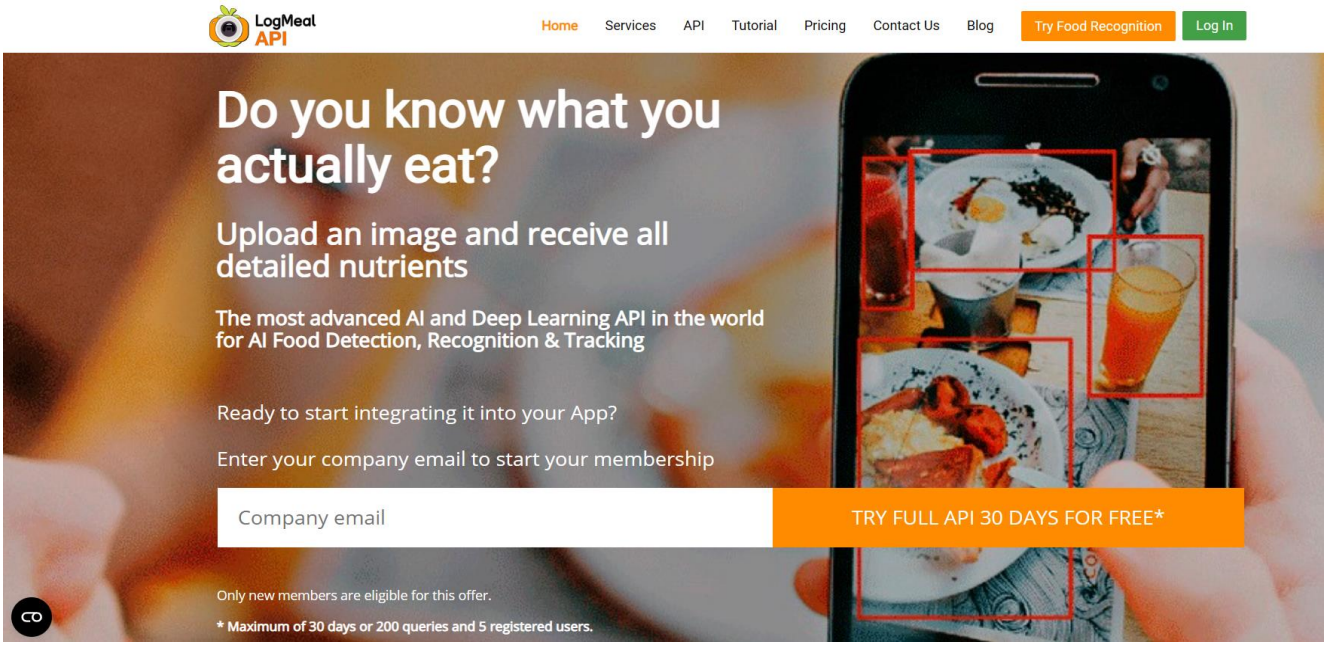

Рисунок 1.4 – Вигляд сайту LogMeal [1]

Головний недолік полягає в тому, що це не є повноцінна система, яку можна з легкістю використовувати самостійно. Для того, щоб скористатися цим API,

необхідно мати програму, яка буде інтегрувати цей інтерфейс прикладного програмування і використовувати його функціональність. Це потребує додаткових технічних знань та ресурсів для розробки і підтримки такої програми. Крім того, цей сервіс є платним, а безкоштовна версія цього API є обмеженою. Користуватися безкоштовно можна за таким планом: 30 днів або 200 запитів та 5 зареєстрованих користувачів на один аккаунт.

Наступна з існуючих систем, яку було обрано для аналізу, – Calorie Mama[2]. Цей сервіс, як і попередній, має добре навчений та ефективний алгоритм розпізнавання, який дозволяє розпізнавати їжу на основі фотографій з високою точністю. Модель, що використовується, базується на глибокому навчанні та інших сучасних методах комп'ютерного зору, що забезпечує швидке та точне розпізнавання різних видів їжі. Вигляд сторінки сервісу зображено на рисунку 1.5

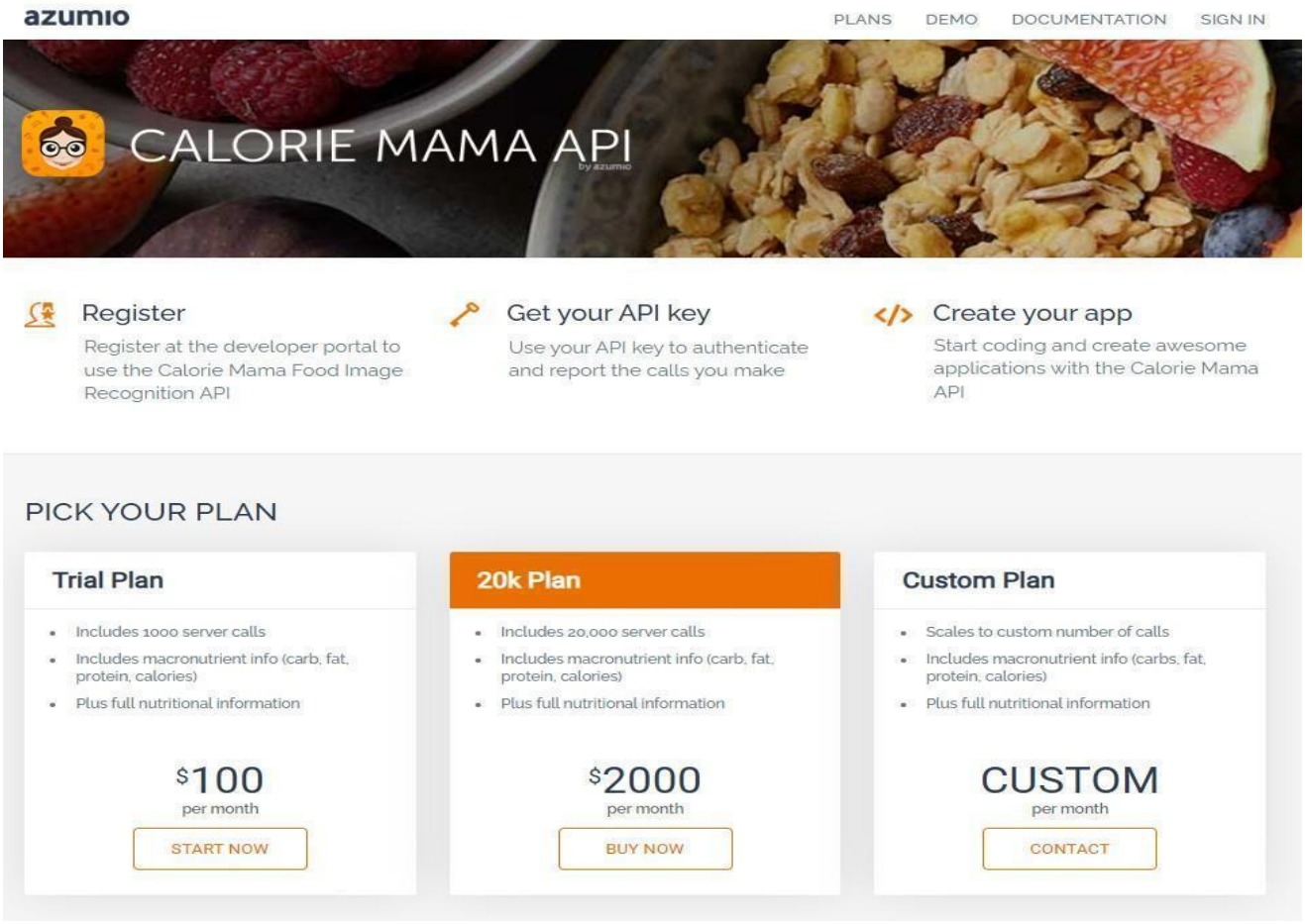

Рисунок 1.5 – Вигляд сайту Calorie Mama [2]

Однак, на відміну від попереднього сервісу, в Calorie Mama доступні тільки платні версії. Це означає, що користувачам необхідно підписатися на план для отримання доступу до повного функціоналу сервісу. Вартість підписки може варіюватися в залежності від обраного плану, але для багатьох користувачів це може стати значною перешкодою. Відсутність безкоштовної версії обмежує можливість ознайомлення з сервісом перед прийняттям рішення про покупку.

1.3 Постановка задачі дослідження

Провівши аналіз існуючих аналогів, можна відзначити, що перевагою в них є добре навчена, швидка та ефективна модель розпізнавання їжі. Ці моделі зазвичай базуються на сучасних методах глибокого навчання, таких як згорткові нейронні мережі (CNN) та YOLO (You Only Look Once). Вони здатні обробляти великі обсяги даних та розпізнавати різноманітні види їжі з високою точністю. Наприклад, CNN дозволяє моделі витягувати важливі ознаки з зображень автоматично, що значно покращує точність розпізнавання.

Однак, основним недоліком таких сервісів є відсутність безкоштовної версії, що обмежує доступ до технології для широкого кола користувачів. Крім того, деякі з них можуть мати обмежений функціонал навіть у платних версіях, що може не відповідати всім потребам користувачів. Також часто відсутній інтерфейс, який був би зручним та інтуїтивно зрозумілим для користувача, що ускладнює використання сервісів не технічними спеціалістами.

Судячи з проведеного аналізу існуючих аналогів, для того щоб розробити систему, яка буде задовольняти користувача, необхідно забезпечити кілька ключових аспектів. Перш за все, інтерфейс має бути інтуїтивно зрозумілим. Це дозволить користувачам швидко ознайомитись з програмою та ефективно використовувати її функції. Другим важливим аспектом є висока точність розпізнавання, щоб мінімізувати можливі помилки і забезпечити надійні результати. Не менш важливим є швидка обробка даних, яка є критично важливою

для забезпечення комфортного користування системою. Швидкість обробки можна покращити, використовуючи оптимізацію алгоритмів та апаратні прискорювачі, такі як GPU та TPU.

Для того щоб виконати поставлене завдання, будуть зроблені наступні кроки: буде навчена модель на наборі даних, використовуючи методи машинного навчання. Це включатиме збір та анотацію великої кількості зображень їжі, а також їх попередню обробку для підготовки до тренування моделі, другим кроком, буде розроблена веб-система, де користувачі зможуть завантажувати свої фотографії їжі та отримувати результати розпізнавання. Ця система буде мати інтуїтивно зрозумілий інтерфейс, що забезпечить зручність використання навіть для не технічних користувачів, третім кроком, буде проведено тестування системи з різними типами їжі, щоб оцінити її точність та швидкість обробки. На основі результатів тестування алгоритм буде вдосконалений при необхідності для підвищення його ефективності та надійності.

Таким чином, впровадження цих кроків дозволить створити систему, яка відповідатиме вимогам користувачів та забезпечуватиме високу якість розпізнавання їжі, швидкість обробки та зручність використання.

2 АЛГОРИТМІЧНЕ ТА ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

2.1 Загальна структура проектованої системи

Для реалізації запропонованої системи було обрано структурно-орієнтований підхід.

Основна ідея цього підходу полягає в розділенні даних та логіки, де дані і функції, які їх обробляють, чітко розділені. Це забезпечує більш модульну та організовану структуру системи, полегшуючи її розробку та обслуговування. Система організовується у вигляді ієрархії модулів та підсистем, що дозволяє легко відстежувати зв'язки між різними компонентами та спрощує налагодження. Модулі є незалежними та відповідають за виконання окремих функцій, що полегшує розробку, тестування та повторне використання коду.

Система складається з декількох модулів, кожен з яких виконує свою специфічну роль.

Основні модулі включають модуль завантаження зображення, модуль обробки зображення, модуль розпізнавання їжі та модуль відображення результатів.

Модуль завантаження зображення приймає зображення від користувача, перевіряє його розмір, зберігає та передає його для подальшої обробки. Цей модуль відповідає за початкову перевірку вхідних даних, забезпечуючи, щоб зображення мали прийнятний розмір для подальшої обробки.

Модуль обробки зображення здійснює попередню обробку зображення. Це включає зміну розміру зображення та нормалізацію, що підготовлює зображення для аналізу. Зміна розміру забезпечує стандартизацію розмірів вхідних зображень, тоді як нормалізація приводить піксельні значення до певного діапазону, що полегшує подальший аналіз.

Модуль розпізнавання їжі є серцем системи, оскільки виконує основну функцію розпізнавання об'єктів на зображеннях, цей модуль використовує алгоритми машинного навчання для розпізнавання їжі на зображенні. Він аналізує

оброблене зображення та повертає результати аналізу, такі як назва їжі та впевненість у розпізнаванні.

Модуль відображення результатів обробляє результати, отримані від модуля розпізнавання, та відображає їх користувачеві у зручному форматі. Цей модуль забезпечує зручний та зрозумілий спосіб представлення інформації.

Структурна схема системи представлена на рисунку 2.1.

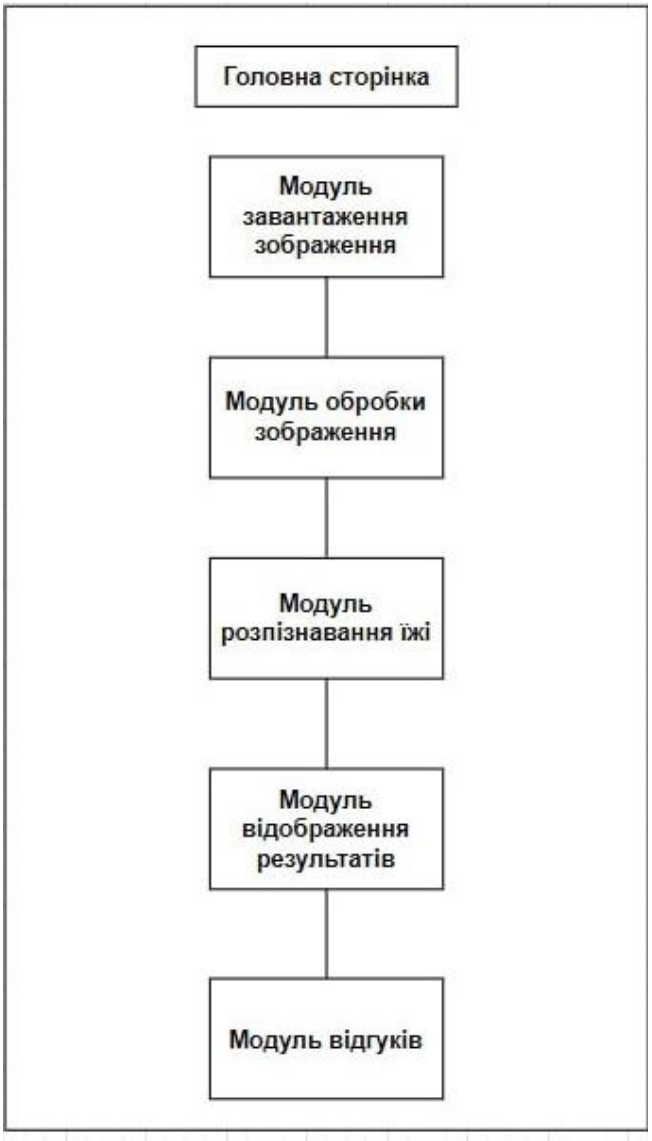

Рисунок 2.1 – Структурна схема системи

Діаграма потоків даних, яка відображає, як дані передаються між модулями, представлена на рисунку 2.2.

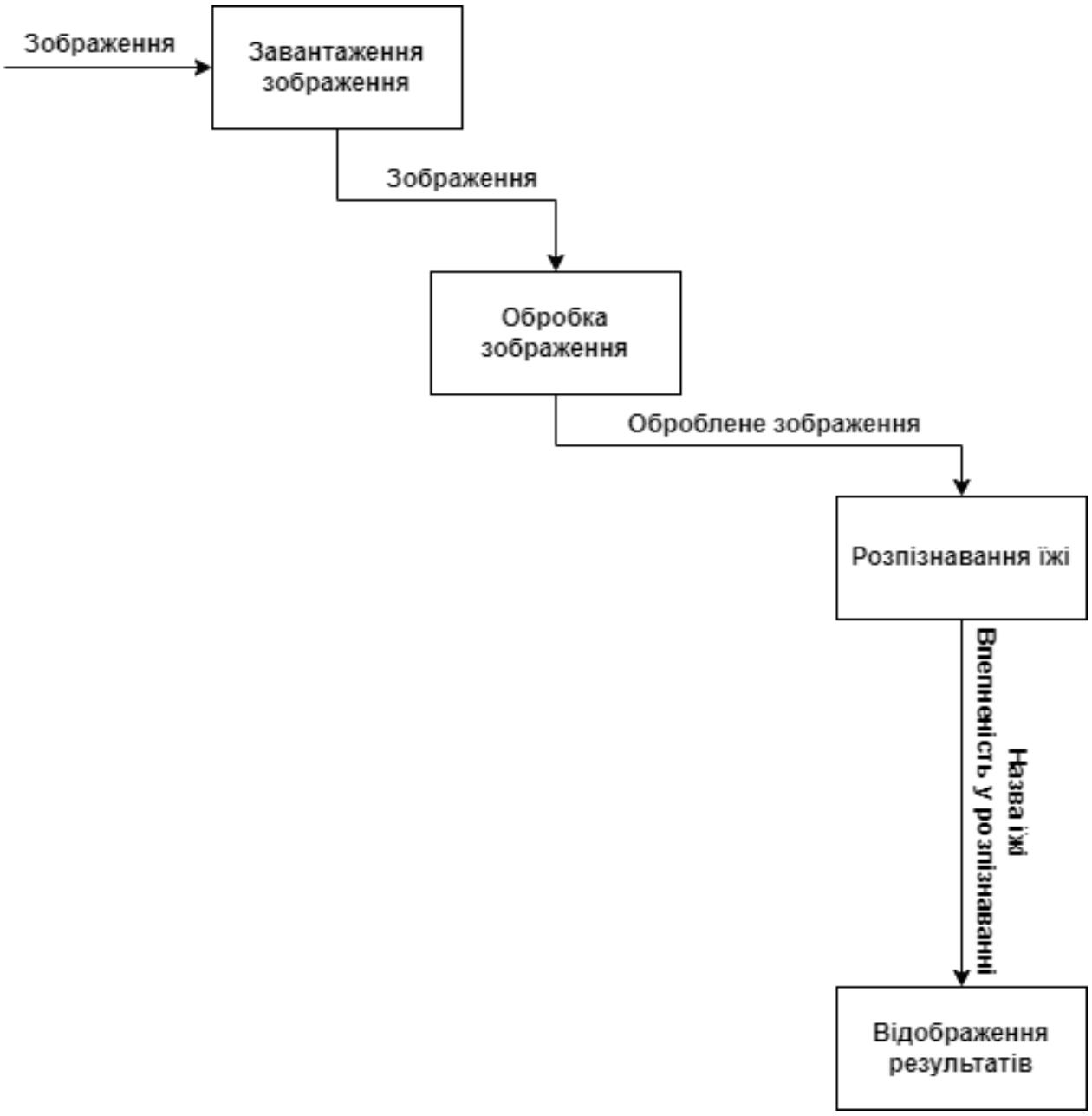

Рисунок 2.2 – Діаграма потоків даних

Процес починається з того, що користувач завантажує зображення в систему через веб-інтерфейс до модуля завантаження зображення, який перевіряє його розмір, щоб переконатися, що зображення відповідає вимогам системи.

Якщо перевірка успішна, модуль завантаження передає зображення до наступного модуля для подальшої обробки. Модуль обробки зображення отримує зображення і починає його попередню обробку. Цей процес включає обрізання та нормалізацію зображення, щоб привести його до стандартного вигляду, придатного

для аналізу. Обрізання дозволяє видалити зайві частини зображення, а нормалізація коригує піксельні значення для покращення якості аналізу.

Після завершення попередньої обробки, модуль обробки зображення передає підготовлене зображення до модуля розпізнавання їжі. Модуль розпізнавання їжі отримує оброблене зображення і аналізує його з використанням алгоритмів машинного навчання. Ці алгоритми дозволяють ідентифікувати їжу на зображенні, визначаючи її назву та рівень впевненості в правильності розпізнавання.

Після завершення аналізу, результати, такі як назва їжі та впевненість у розпізнаванні, передаються до модуля відображення результатів. Модуль відображення результатів обробляє отримані дані та показує їх користувачеві у зручному форматі. Користувач може бачити назву їжі та рівень впевненості в її правильному визначенні, що забезпечує зворотний зв'язок та підвищує зручність користування системою.

Інтерфейс представлення результатів зроблений таким чином, щоб інформація була легко зрозуміла і доступна користувачеві.

Завдяки модульній структурі, кожен модуль може бути розроблений, протестований та оптимізований окремо, що знижує загальну складність системи та підвищує її надійність. Така організація дозволяє легко вносити зміни і додавати нові функції, не впливаючи на роботу інших частин системи, що робить її більш адаптивною до змін та вимог користувачів.

2.2 Алгоритмічне забезпечення

На рисунку 2.3 представлено загальний алгоритм роботи системи. Спочатку користувач відкриває головну сторінку системи, де він може завантажити зображення у веб-базовану систему або вийти з системи.

На головній сторінці користувач бачить кнопку для завантаження зображення та опції для виходу з системи. Після натискання кнопки завантаження користувач обирає файл зі свого пристрою.

Рисунок 2.3 – Блок-схема алгоритму роботи системи

Якщо користувач завантажив зображення, система перевіряє його формат та розмір, щоб переконатися, що вони відповідають вимогам. У разі невдалого завантаження користувач отримує повідомлення про помилку з поясненням, що саме пішло не так, наприклад, що розмір зображення перевищує допустимий ліміт.

Після цього користувачу пропонується можливість повторно завантажити зображення.

Якщо зображення успішно завантажено, система переходить до етапу попередньої обробки зображення. Цей етап включає зміну розміру зображення для приведення його до стандартного розміру, який використовує система, та нормалізацію, яка коригує піксельні значення для покращення якості аналізу.

Після завершення попередньої обробки зображення передається до моделі розпізнавання, яка аналізує його за допомогою алгоритмів машинного навчання. Модель ідентифікує їжу на зображенні та визначає рівень впевненості в розпізнаванні.

Результати аналізу, такі як назва їжі та рівень впевненості, передаються на наступний етап. На наступному етапі результати розпізнавання обробляються та відображаються користувачеві у зручному форматі.

Інтерфейс показує назву їжі та відсоток впевненості в правильності розпізнавання. Після цього користувач має можливість залишити відгук про роботу системи, що може допомагати покращувати якість розпізнавання та взаємодію з користувачами. Алгоритм забезпечує точність та достовірність обчислень шляхом використання перевірених методів машинного навчання та ретельної обробки зображень. Це дозволяє системі надавати надійні результати та забезпечує високу якість розпізнавання, роблячи її корисним інструментом для ідентифікації їжі на зображеннях.

Для реалізації системи була обрана клієнт-серверна архітектура.

Клієнт-серверна архітектура є однією з найбільш поширених архітектурних парадигм у розробці програмного забезпечення. У цій архітектурі функціональність системи розділяється між двома типами програмних компонентів: клієнтами та серверами.

Клієнтська частина програми відповідає за взаємодію з користувачем та відображення інформації. Це може бути веб-браузер у випадку веб-додатків або додаток на мобільному пристрої. Клієнтська частина ініціює запити до сервера та

обробляє отримані відповіді, зазвичай використовуючи HTML, CSS та JavaScript для створення інтерфейсу та забезпечення взаємодії з користувачем.

Серверна частина програми відповідає за обробку запитів від клієнтів та надання відповідей. Вона може виконувати обчислення, доступ до баз даних, обробку файлів тощо. У випадку з клієнт-серверною архітектурою для розпізнавання зображень, серверна частина відповідає за приймання зображень від клієнтів, передачу їх на обробку за допомогою Python коду, отримання результатів та їх повернення клієнту.

Ця архітектура забезпечує ефективне розподілення функціональності між клієнтами та серверами, що дозволяє забезпечити швидку та надійну роботу системи. Крім того, вона спрощує розширення системи, оскільки можна додавати або змінювати сервери без необхідності змінювати клієнтську частину.

На рисунку 2.4 зображено схематичну модель клієнт-серверної архітектури.

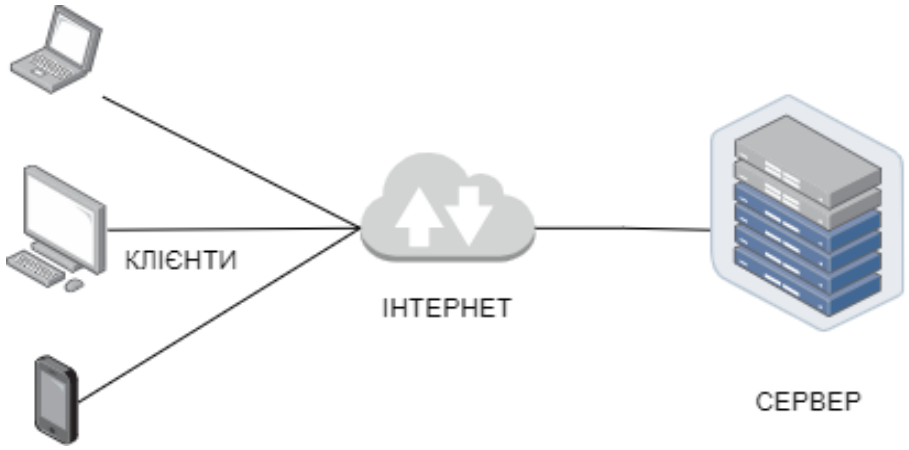

Рисунок 2.4 – Схематична модель клієнт-серверної архітектури

Клієнтська частина веб-системи була реалізована з використанням таких технологій, як HTML, CSS та JavaScript. HTML (HyperText Markup Language) забезпечує структуру веб-сторінки, визначаючи розташування та ієрархію елементів інтерфейсу, таких як форми, кнопки та зображення. CSS (Cascading Style Sheets) відповідає за стилізацію веб-сторінки, надаючи можливість задавати кольори, шрифти, розміри та розташування елементів, що дозволяє створити

привабливий та зручний інтерфейс користувача. JavaScript використовується для додавання динамічної поведінки та інтерактивності на веб-сторінці, забезпечуючи функціонал для завантаження зображень, відправки даних на сервер та обробки отриманих результатів.

Серверна частина системи була реалізована за допомогою PHP скрипта та Python коду. PHP скрипт обробляє запити від клієнтів, керує завантаженням зображень та взаємодією з сервером, тоді як Python код відповідає за безпосереднє розпізнавання їжі на фото. PHP використовується для обробки HTTP-запитів від клієнтів, приймає завантажені зображення, зберігає їх на сервері та готує до обробки Python кодом. Python застосовується для реалізації алгоритмів розпізнавання зображень. Після отримання зображення від PHP скрипта, Python код виконує попередню обробку зображення (зміна розміру, нормалізація) та застосовує модель машинного навчання для визначення категорії їжі.

Взаємодія між клієнтською та серверною частинами реалізована за допомогою HTTP-запитів. Клієнт надсилає запити на сервер для завантаження зображень та отримання результатів розпізнавання. Серверна частина, реалізована на PHP, обробляє ці запити, передає зображення на обробку Python коду, отримує результати розпізнавання та повертає їх клієнту у вигляді JSON-даних. Клієнтська частина обробляє отримані результати та відображає їх користувачеві. Така архітектура забезпечує розподіл навантаження між клієнтською та серверною частинами, що дозволяє ефективно використовувати обчислювальні ресурси та забезпечувати швидку і надійну роботу системи.

2.3 Інформаційне забезпечення

Навчання моделі розпізнавання їжі складається з декількох ключових етапів: збір і підготовка даних, вибір архітектури моделі, тренування моделі та оцінка її продуктивності.

На етапі збору даних важливо зібрати великий та різноманітний набір зображень їжі, кожне з яких повинно мати відповідну мітку, що вказує на категорію їжі. Це дозволяє моделі навчитися розпізнавати різні види їжі з високою точністю.

Для навчання моделі було використано датасет Food-101, який містить 101 000 зображень, розділених на 101 категорію їжі, кожна з яких представлена 1000 зображеннями. Цей датасет є одним із найпопулярніших та найбільш використовуваних у задачах розпізнавання зображень їжі.

Після збору даних наступним кроком є підготовка даних. Цей етап включає кілька важливих підетапів. По-перше, очищення даних: видалення некоректних або дублікатних зображень. По-друге, нормалізація даних: масштабування зображень до однакового розміру та нормалізація пікселів до певного діапазону значень. По-третє, застосування такий технік як обертання, зміна яскравості, відображення та обрізання зображень. Це допомагає збільшити обсяг даних та підвищити стійкість моделі до різних варіацій даних.

Вибір архітектури моделі є наступним критичним етапом. Архітектура моделі визначає, наскільки ефективно вона зможе розпізнавати зображення. Для задач розпізнавання зображень зазвичай використовуються згорткові нейронні мережі (CNN).

Після вибору архітектури модель тренується на зібраних та підготовлених даних. Тренування моделі виконується шляхом багаторазового проходження через набір даних, під час якого коригуються ваги моделі для мінімізації функції втрат. В процесі тренування модель поступово вдосконалюється, навчаючись розпізнавати різні категорії їжі з високою точністю.

Після завершення тренування модель оцінюється на тестовому наборі даних. Оцінка моделі включає визначення її точності та інших метрик продуктивності, таких як точність, повнота та F-міра. Це дозволяє зрозуміти, наскільки добре модель виконує свою задачу на нових, невидимих під час тренування, даних.

Для прискорення тренування моделей глибокого навчання часто використовуються графічні процесори (GPU) та тензорні процесори (TPU). GPU мають велику кількість ядер, що дозволяє їм ефективно обробляти великі обсяги

даних паралельно, значно прискорюючи процес тренування моделей. TPU спеціально розроблені для виконання операцій з тензорами, що робить їх ще більш ефективними для задач глибокого навчання. Використання цих обчислювальних ресурсів дозволяє зменшити час тренування моделей з декількох днів до кількох годин.

Таким чином, навчання моделі розпізнавання їжі включає в себе комплексний процес, що складається з декількох етапів, кожен з яких є важливим для досягнення високої точності та ефективності моделі.

2.3.1 Опис вхідної та вихідної інформації

Вхідна та вихідна інформація подана в таблиці 2.1

Таблиця 2.1 – Вхідна та вихідна інформація

	Інформація	Подробиці
Вхідна інформація	Зображення їжі	Зображення їжі: Завантажуване зображення може бути в будь якому форматі
Вихідна інформація	Зображення їжі	Те саме зображення що було завантажене для розпізнавання в систему
	Назва розпізнаної їжі	Назва розпізнаної їжі: Назви трьох найбільш вірогідних варіантів розпізнаної їжі.
	Відсоток впевненості в розпізнаванні	Відсоток впевненості: Відсотки впевненості трьох розпізнаних типів їжі.

3 ПРОГРАМНО-ТЕХНОЛОГІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Програмні засоби реалізації системи

Для реалізації веб-базованої системи розпізнавання їжі було використано мову програмування Python, яка застосовувалась у процесах навчання моделі розпізнавання їжі, створення скриптів для обробки та аналізу зображень. Мова Python – це зручний і простий інструмент для створення програм. Вона легко засвоюється і має чистий, зрозумілий синтаксис, який полегшує написання коду. Python має велику кількість бібліотек, що дозволяють виконувати різноманітні завдання, такі як обробка даних та розробка штучного інтелекту і так далі. Він також крос-платформений, тобто може працювати на різних операційних системах, таких як Windows, macOS та Linux. Python інтерпретована мова, тобто виконується рядок за рядком без необхідності компіляції, що робить його досить гнучким для розробки та відлагодження програм.[3]

Мова розмітки HTML використовувалась для створення інтерфейсу веб-базованої системи. HTML (Hypertext Markup Language) – це основна мова розмітки для створення веб-сторінок. Вона використовується для визначення структури та вмісту веб-сторінки за допомогою різних елементів та тегів. HTML визначає різноманітні елементи, такі як заголовки, абзаци, списки, посилання, зображення та інші, які сприймаються веб-браузерами та відображаються користувачам. HTML є основою для створення веб-сторінок, а веб-браузери інтерпретують його код для відображення сторінок на екранах користувачів.[4]

Скриптова мова програмування PHP використовувалась для створення серверної частини веб-системи та обробки запитів. PHP – це скриптова мова програмування, яка використовується для створення динамічних веб-систем та веб-додатків. Вона виконується на веб-сервері та генерує HTML-код, який передається до веб-браузера користувача, що дозволяє створювати веб-сторінки, які можуть взаємодіяти з базами даних, обробляти форми, створювати куки та сеанси, і багато іншого. PHP є однією з найпопулярніших мов програмування для веб-розробки, завдяки своїй простоті в освоєнні та потужному набору функцій.[5]

Таблиці каскадних стилів CSS використовувалась для задання стилів для веб-системи. CSS (Cascading Style Sheets) – це опис стилів, який використовується для оформлення веб-сторінок, написаних на HTML або інших мовах розмітки. CSS дозволяє визначати зовнішній вигляд елементів веб-сторінки, такі як кольори, шрифти, розміри, відступи, межі та інші стилізаційні властивості. Використання CSS дозволяє веб-розробникам легко контролювати і стилізувати вигляд веб-сторінок, забезпечуючи їм більш сучасний та привабливий вигляд [6].

Мова програмування JavaScript використовувалась для створення динамічної поведінки у веб-системі. JavaScript – це високорівнева, об'єктно-орієнтована мова програмування, що використовується для реалізації динамічної поведінки на веб-сторінках. Вона широко використовується для створення інтерактивних елементів веб-сторінок, таких як анімації, обробка подій, валідація форм, динамічне оновлення вмісту сторінок без перезавантаження, а також для розробки веб-додатків та інструментів. JavaScript виконується в середовищі веб-браузера та може взаємодіяти з HTML та CSS для створення динамічних та інтерактивних веб-сторінок. Він також використовується для розробки серверних додатків (за допомогою платформи Node.js), мобільних додатків та додатків для різних IoT-пристроїв. Основні функції JavaScript включають у себе роботу з об'єктами, функціональне програмування, роботу з подіями, маніпуляцію DOM (Document Object Model), роботу з сесійними файлами та локальним сховищем, а також роботу з мультимедійними елементами. JavaScript є однією з найпоширеніших та найважливіших мов програмування веб-розробки [7].

При розробці модулів обробки та розпізнавання їжі по фото були використані наступні бібліотеки комп’ютерного зору та моделі для розпізнавання їжі:

- OpenCV (Open Source Computer Vision Library) – це відкрите програмне забезпечення, що надає широкий спектр інструментів для обробки зображень. Вона дозволяє виконувати різноманітні завдання, такі як завантаження та збереження зображень, редагування, обробка та аналіз пікселів, детекція об'єктів та облич, взаємодія з відеопотоками, виконання операцій з фільтрами та перетворення між різними колірними просторами. OpenCV надає зручний та потужний інтерфейс для

розв'язання задач комп'ютерного зору в різних областях, від наукових досліджень до застосувань у промисловості та робототехніці [8].

- os – це вбудована бібліотека мови програмування Python, яка надає функції для взаємодії з операційною системою. Вона дозволяє виконувати різноманітні завдання, такі як створення, перейменування та видалення файлів та каталогів, отримання інформації про поточну робочу директорію, а також виконання системних команд [9].
- shutil – це вбудована бібліотека Python, яка надає функції для виконання операцій з файлами та каталогами. Вона включає в себе інструменти для копіювання, переміщення, перейменування та видалення файлів та каталогів, а також для архівації та розархівації даних [10].
- collections – це вбудована бібліотека Python, яка містить додаткові структури даних порядку в порівнянні зі стандартними типами даних Python. Одна з основних структур даних, яку вона надає, - це defaultdict, який є підкласом словника та автоматично надає значення за замовчуванням для нових ключів [11].
- numpy – це бібліотека для наукових обчислень у мові програмування Python. Вона надає підтримку для масивів та матриць, включаючи векторизовані операції з цими даними. numpy також містить функції для виконання різних математичних операцій, включаючи обчислення статистики, роботу з лінійною алгеброю та обробку зображень [12].
- TensorFlow – це відкрита бібліотека машинного навчання, розроблена компанією Google, яка надає засоби для побудови та тренування різних типів нейронних мереж. Основним призначенням TensorFlow є створення та навчання моделей штучних нейронних мереж для розв'язання різних задач машинного навчання, таких як класифікація зображень, розпізнавання мови та обробка природної мови. Однією з головних переваг TensorFlow є його гнучкість та масштабованість. Вона надає засоби для роботи з різними типами нейронних мереж, включаючи згорткові нейронні мережі (CNN) – тип глибокої нейронної мережі, яка ефективно аналізує візуальні дані[13], рекурентні нейронні мережі (RNN) – тип нейронних мереж, які ефективно працюють з послідовними даними

[14], глибокі нейронні мережі (DNN) – тип нейронних мереж, які складаються з великої кількості шарів між вхідним і вихідним шаром. Вони використовуються для розв'язання складних завдань, таких як розпізнавання образів, обробка природної мови, прогнозування часових рядів та багато інших[15]. А також для роботи з багатьма типами даних, включаючи зображення, текст та числові дані. TensorFlow також надає можливості для розгортання моделей на різних платформах, включаючи мобільні пристрої та вбудовані системи, що робить його популярним інструментом для розробки різних застосунків штучного інтелекту[16].

Для написання та редагування програмного коду було використано Sublime Text. Sublime Text – це інтегроване середовище розробки (ICP), яке призначене для редагування текстових файлів у різних мовах програмування та розмітки. Sublime Text пропонує ряд функцій, які полегшили роботу з кодом, включаючи підсвічування синтаксису, автоматичне доповнення коду, можливість створення маркерів, які дозволяли швидко переходити до певних рядків коду, і багато іншого. Крім того, він має можливість редагування кількох рядків одночасно, що робило процес редагування більш ефективним та швидким [17].

3.2 Реалізація програмного забезпечення

Для розробки веб-базованої системи розпізнавання їжі за допомогою комп'ютерного зору, перш за все, була розроблена модель для розпізнавання. Для цього був створений код, в якому є функція яка називається prepare_data. Ця функція відповідає за підготовку даних для навчання моделі(Додаток А).

Наступний блок коду імпортує необхідні бібліотеки: os для роботи з файловою системою, shutil для виконання високорівневих операцій з файлами, зокрема копіювання, defaultdict з модуля collections для створення словника зі значенням за замовчуванням у формі списку:

```
import os
from shutil import copy
from collections import defaultdict
```

Наступний блок коду – це визначення функції prepare_data, яка приймає три аргументи:

filepath – шлях до текстового файлу, який містить список шляхів до зображень та їхніх класів.

src – шлях до директорії, де розташовані вихідні зображення.

dest – шлях до директорії, куди будуть копіюватися зображення, розсортовані за класами:

```
def prepare_data(filepath, src, dest):
```

Далі створюється порожній словник classes_images, де ключами будуть класи їжі, а значеннями - списки зображень. Використання defaultdict гарантує, що кожен ключ автоматично матиме порожній список як значення:

```
classes_images = defaultdict(list)
```

Наступний блок коду відкриває файл за шляхом filepath для читання ('r'). Він читає всі рядки з файлу та зберігає їх у списку paths, попередньо видаляючи зайві пробіли на початку і в кінці кожного рядка:

```
with open(filepath, 'r') as file:
    paths = [line.strip() for line in file.readlines()]
```

Тут кожен рядок з paths розділяється за символом "/", щоб отримати клас їжі (food_class) та ім'я зображення (image). Потім ім'я зображення (до якого додається розширення .jpg) додається до відповідного списку у словнику classes_images:

```
for path in paths:
    food_class, image = path.split('/')
    classes_images[food_class].append(image + '.jpg')
```

Цей блок коду перебирає всі класи їжі та списки зображень з classes_images. Для кожного класу визначається шлях до теки призначення (dest_folder) шляхом об'єднання шляху dest та назви класу. Якщо така папка ще не існує, вона створюється за допомогою os.makedirs:

```
for food_class, images in classes_images.items():
    dest_folder = os.path.join(dest, food_class)
    if not os.path.exists(dest_folder):
        os.makedirs(dest_folder)
```

Цей блок коду перебирає всі зображення для кожного класу. Він формує шлях до вихідного зображення (src_path) шляхом об'єднання шляху src, назви класу та імені зображення, а також шлях до призначення (dest_path) шляхом об'єднання шляху dest, назви класу та імені зображення. Зображення копіюється з src_path у dest_path за допомогою функції copy з модуля shutil:

```
for image in images:
    src_path = os.path.join(src, food_class, image)
    dest_path = os.path.join(dest, food_class, image)
    copy(src_path, dest_path)
```

Ці рядки коду спершу виводять повідомлення "Copying Done!" у консоль після завершення копіювання всіх файлів. Після чого викликають функцію prepare_data двічі: один раз для підготовки даних для тренувального набору (train), використовуючи список з файлу train.txt, і один раз для тестового набору (test), використовуючи список з файлу test.txt:

```
    print("Copying Done!")

prepare_data('food-101/meta/train.txt', 'food-101/images', 'food-101/train')
prepare_data('food-101/meta/test.txt', 'food-101/images', 'food-101/test')
```

Другим кроком в розробці системи було розроблено код який використовує навчену модель для розпізнавання та класифікації їжі(Додаток А).

Цей блок імпортує необхідні бібліотеки для обробки зображень (cv2), роботи з масивами (numpy) та завантаження попередньо навченої моделі (load_model). Потім він завантажує зображення з вказаного шляху:

```
import cv2
import os
import numpy as np
from tensorflow.keras.models import load_model

image_path = 'uploads/photo.jpg'
image = cv2.imread(image_path)
```

Тут зображення змінюється до розміру 224x224 пікселів, нормалізується (перетворюється на значення в діапазоні [0, 1]) та додається додатковий вимір для сумісності з моделлю нейронної мережі:

```
image_food = cv2.resize(image, (224, 224))
image_food = image_food.astype('float32') / 255.0
image_food = np.expand_dims(image_food, axis=0)
```

Цей блок завантажує попередньо навченої моделі розпізнавання з файлу та використовує її для прогнозування класів на основі підготовленого зображення:

```
model_food = load_model('model/food_recognition_model.h5')
predictions_food = model_food.predict(image_food)
```

В наступному блоці коду створюється пустий список для подальшого заповнення його класами їжі, потім завантажуються назви класів з файлу coco.names, обчислюються індекси трьох найвірогідніших класів, і для кожного з них формується рядок з назвою класу та впевненістю у відсотках:

```
food_clases = []
with open('model/coco.names', 'r') as f:
    food_clases = f.read().splitlines()


result_text = ""
top_n_indices = np.argsort(predictions_food[0])[::-1][:3]
for idx in top_n_indices:
    label_food = food_clases[idx]
    confidence_food = predictions_food[0][idx] * 100
    result_text += f"{label_food}: {confidence_food:.2f}%\n"
```

Останній блок зберігає текстовий результат класифікації у файл result.txt, щоб його можна було переглянути пізніше:

```
with open('result.txt', 'w') as file:
    file.write(result_text)
```

Наступним кроком була розробка інтерфейсу веб-системи. Для цього було розроблено HTML документ, написано CSS стилі до документу, а також створено JavaScript код для реалізації динамічної поведінки на інтерфейсі(Додаток А).

Цей блок визначає основні налаштування HTML-документа: встановлює мову (uk), кодування (UTF-8), параметри перегляду на різних пристроях (viewport), підключає шрифт і CSS-стилі, а також встановлює заголовок сторінки:

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
                                                                    <link
href="https://fonts.googleapis.com/css2?family=Comic+Neue:wght@300;400;700&display=swap"
rel="stylesheet">
  <link rel="stylesheet" href="styles.css">
  <title>Food Recognition</title>
</head>
<body>
```

Цей блок створює заголовок сторінки з текстом "Food Recognition" і починає форму для завантаження зображення страви. Форма має ідентифікатор uploadForm та налаштування для завантаження файлів:

```
<div class="wrapper">
  <!-- header -->
  <header>
    <h1>Food <span class="highlight">Recognition</span></h1>
  </header>

  <!-- main -->
  <main>
    <form id="uploadForm" enctype="multipart/form-data">
      <div class="instruction">
        <p>Load your photo of dish to recognize it!</p>
      </div>
```

Цей блок створює секцію для завантаження та перегляду зображення. Включає елемент img для попереднього перегляду зображення, кнопку для завантаження файлу, текстову область для відображення результату передбачення і кнопку для отримання прогнозу:

```
            </div>
            <div class="upload-section">
              <div class="preview-buttons">
                <div class="image-preview">
                  <img id="preview" src="" alt="Image preview">
                </div>
                <label for="fileInput" class="upload-button">
                  Load photo
                  <input type="file" id="fileInput" name="image" accept="image/*">
                </label>
              </div>
              <div class="preview-buttons">
                <div class="text-preview">
                  <textarea id="predictionText" readonly></textarea>
                </div>
                <label for="getPredictionInput" class="prediction-button">
                  Get Prediction
                        <input type="button" id="getPredictionInput" onclick="getPrediction()"
style="display: none;">
                </label>
              </div>
            </div>
```

Цей блок додає секцію для залишення відгуків. Включає заголовок, текстову область для введення відгуку та кнопку для його відправки. Закінчується форма завантаження та головний контент сторінки:

```
              <div class="feedback">
                <h2>Leave Feedback</h2>
                  <textarea id="feedbackText" rows="4" cols="50" placeholder="Enter your feedback
here..."></textarea>
                <button onclick="submitFeedback()">Submit</button>
              </div>
              </form>
            </main>
          </div>
```

Наступний блок коду містить JavaScript для обробки завантаження та перегляду зображення, а також для відправки зображення на сервер та отримання прогнозу:

```
<script>
  function previewImage(event) {
   const file = event.target.files[0];
   if (file) {
    const reader = new FileReader();
    reader.onload = function(e) {
     const preview = document.getElementById('preview');
     preview.src = e.target.result;
     preview.style.display = 'block';
    };
    reader.readAsDataURL(file);
   }
  }



  function getPrediction() {
   fetch('process_image.php', {
    method: 'POST',
    body: new FormData(document.getElementById('uploadForm'))
   })
   .then(response => response.text())
   .then(data => {
    const predictionText = document.getElementById('predictionText');
    predictionText.value = data;
   })
   .catch(error => {
    console.error('Error:', error);
    alert('An error occurred while getting the prediction. Please try again later.');
   });
  }

  document.getElementById('fileInput').addEventListener('change', previewImage);
</script>
</body>
</html>
```

Функція previewImage відображає попередній перегляд зображення, а функція getPrediction відправляє зображення на сервер та відображає результат у текстовій області. Подія change на елементі fileInput викликає функцію previewImage (Додаток А).

Етап додавання стилів до інтерфейсу описаний нижче. Цей стиль встановлює шрифт тексту, вирівнює вміст по центру екрану та задає фонове зображення. Висота елементу body становить 100% від висоти екрану (100vh). Фонове зображення розтягнуте на весь екран, зберігаючи пропорції (background-size: cover):

```
body {
  font-family: 'Comic Neue', sans-serif;
  margin: 0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  overflow: hidden;
  position: relative;
}
```

Цей стиль "::before" вставляє фонове зображення з ефектом розмиття (blur). Зображення розміщується на всю поверхню сторінки (width: 100%; height: 100%;) і має фіксоване розташування. Використовується фонове зображення з файлу 'images/background.jpg':

```
body::before {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: url('images/background.jpg') no-repeat center center fixed;
  background-size: cover;
  filter: blur(3px);
  z-index: -1;
}
```

Цей блок встановлює стилі для основного контейнера з контентом. Задається фоновий колір з прозорістю, внутрішні відступи, закруглені кути, тінь та розмір контейнера:

```
.wrapper {
  background-color: rgba(255, 255, 255, 0.9);
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
  text-align: center;
  width: 80%;
  max-width: 600px;
  position: relative;
  z-index: 0;
}
```

Цей блок встановлює стилі для заголовка та інструкцій. Заголовок має великий розмір шрифту, а кольором виділяється частина з класом 'highlight'. Текст інструкції має певний розмір шрифту та відступи:

```
header h1 {
  font-size: 2em;
  margin: 0;
}
header .highlight {
  color: #2ECC71;
}
.instruction p {
  font-size: 1.2em;
  margin: 20px 0;
}
```

Цей блок встановлює стилі для секцій завантаження та попереднього перегляду зображення. Елементи знаходяться поруч один з одним, кнопки мають

певні кольори та відступи. Попередній перегляд зображення та текстовий попередній перегляд мають фіксований розмір та вирівнюються по центру:

```
.upload-section, .preview-buttons {
 display: flex;
 flex-direction: row;
 justify-content: space-between;
}
.upload-button, .prediction-button {
 display: inline-block;
 background-color: #2ECC71;
 color: white;
 padding: 10px 20px;
 border-radius: 5px;
 cursor: pointer;
 position: relative;
 margin-top: 10px;
}
.image-preview, .text-preview {
 width: 200px;
 height: 200px;
 background-color: #e0e0e0;
 border-radius: 8px;
 display: flex;
 align-items: center;
 justify-content: center;
 margin: 20px;
}
```

Цей блок коду перевіряє, чи було відправлено файл зображення методом POST і чи існує він у формі з іменем "image". Якщо ці умови виконуються, встановлюється шлях для завантаження файлу у папку "uploads/" з ім'ям "photo.jpg". По-замовчуванню, змінна $uploadOk встановлюється на 1, що означає, що завантаження можливе. Після цього виконується перевірка, чи файл є дійсним

зображенням за допомогою функції getimagesize(), а також перевіряється розмір файлу. Якщо хоча б одна з цих перевірок не пройдена, змінна $uploadOk встановлюється на 0, що означає, що завантаження файлу не допускається:

```
<?php
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_FILES['image'])) {
    $target_dir = "uploads/";
    $target_file = $target_dir . "photo.jpg";
    $uploadOk = 1;

    $check = getimagesize($_FILES["image"]["tmp_name"]);
    if ($check === false) {
        $uploadOk = 0;
    }

    if ($_FILES["image"]["size"] > 500000) {
        $uploadOk = 0;
    }
```

Цей блок коду виконується лише у випадку, якщо користувач відправив файл зображення методом POST через форму на веб-сайт. Він перевіряє, чи завантажене користувачем зображення відповідає всім необхідним критеріям, таким як коректність формату та розмір файлу. Якщо усі перевірки пройдені успішно, зображення зберігається на сервері, а потім викликається Python-скрипт для обробки цього зображення, який розпізнає на ньому їжу. Результат розпізнавання виводиться на веб-сторінку для подальшого використання чи відображення:

```
    if ($uploadOk != 0 && move_uploaded_file($_FILES["image"]["tmp_name"],
$target_file)) {
        $pythonScriptPath = "code/food_recognition.py";
        $command = "python3 $pythonScriptPath 2>&1";

        exec($command, $output, $return_var);
        $result = trim(file_get_contents("result.txt"));
        echo htmlspecialchars($result);
    }
}
?>
```

3.3 Тестування розробленої системи

Для функціонального тестування розробленої системи було створено таблицю (див. таблиця 3.1).

Таблиця 3.1 – Функціональне тестування

Наявність елементів системи	Chrome (Пройшло Так/Ні)	Коментарі
Вивід результатів	Так	
Кнопки	Так	Відображаються правильно обидві кнопки “Load Photo” та “Get Prediction”
Зображення в рамці	Так	Зображення показується після завантаження його в систему, та залишається після натискання на кнопку отримати результати розпізнавання
Результати розпізнавання	Так	Виводяться у передбачене для цього вікно, виводиться назва розпізнаної їжі та відсотки впевненості Текстове поле доступне тільки для перегляду
Меню відгуків	Так	З’являється після використання системи
Вивід помилки	Так	Якщо перед завантаженням зображення, натиснути на кнопку “Get Prediction” виведе помилку про те що зображення не було завантажене

Вигляд початкової веб-системи зображено на рисунку 3.1

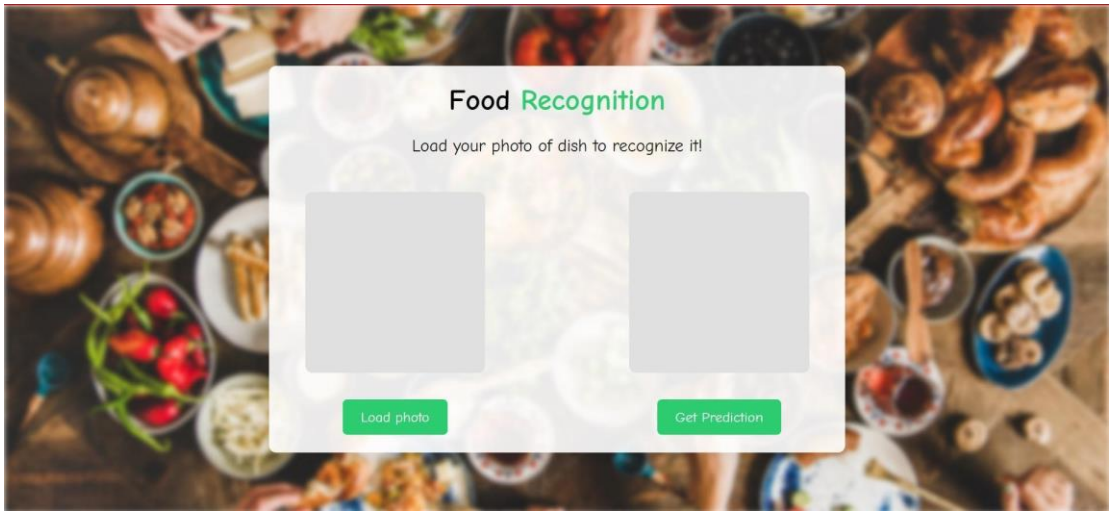

Рисунок 3.1 – Початковий вигляд веб-системи розпізнавання їжі

На початковій сторінці веб-системи розташоване вікно з двома полями, зліва – для зображення, те що справа – текстове поле для виводу результатів розпізнавання, під ними розташовані дві кнопки, зліва “Load Photo”, при натисканні на яку можна буде вибрати та завантажити зображення для розпізнавання, справа – кнопка “Get Prediction”, при натисканні на яку система починає розпізнавати зображення яке вибрав користувач, після закінчення розпізнавання система виведе результати в передбачене для цього текстове поле.

При спробі отримати результат розпізнавання не завантаживши зображення, виведе помилку про те, що його не було завантажено. На рисунку 3.2 зображено вигляд системи при виводі помилки.

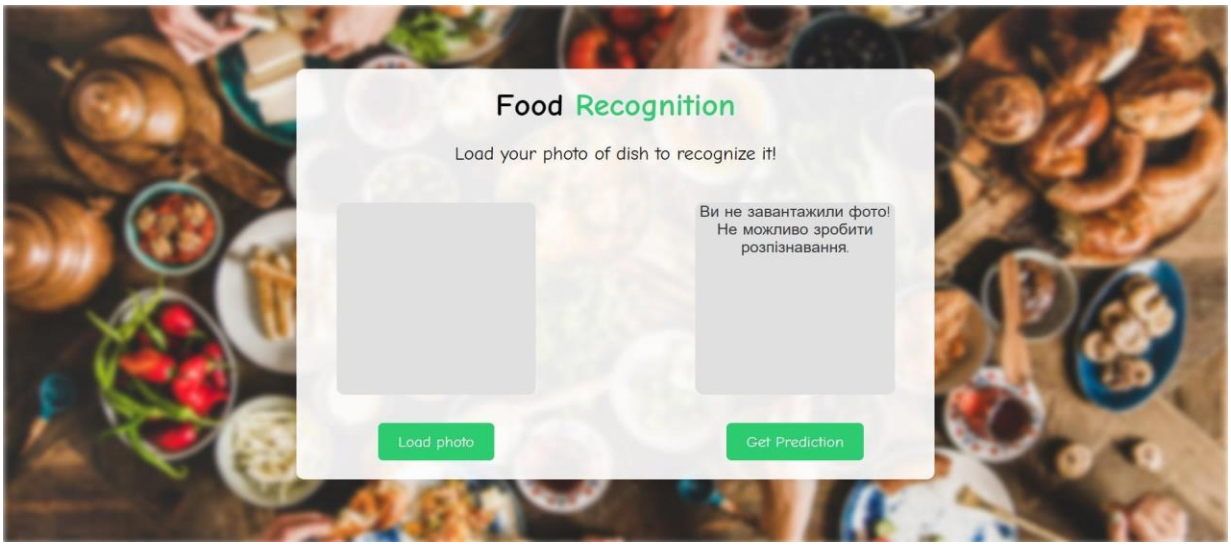

Рисунок 3.2 – Вигляд системи при виводі помилки про зображення

На рисунку 3.3 зображено результат роботи системи при розпізнаванню тістечок.

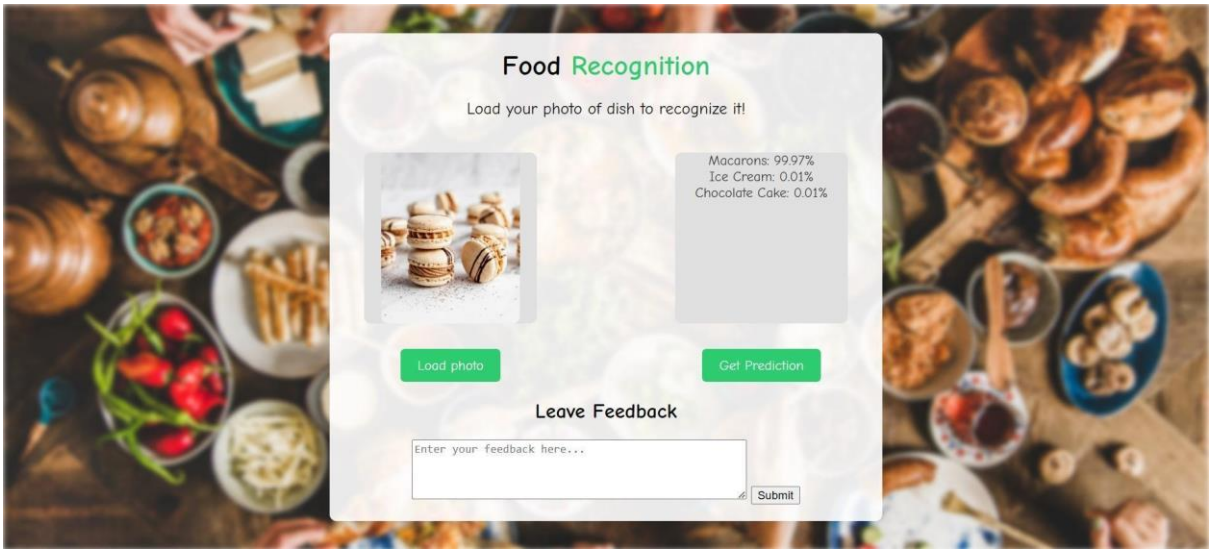

Рисунок 3.3 – Вигляд системи при розпінаних тістечках

На рисунку 3.4 зображено результат роботи системи при розпізнаванні піци.

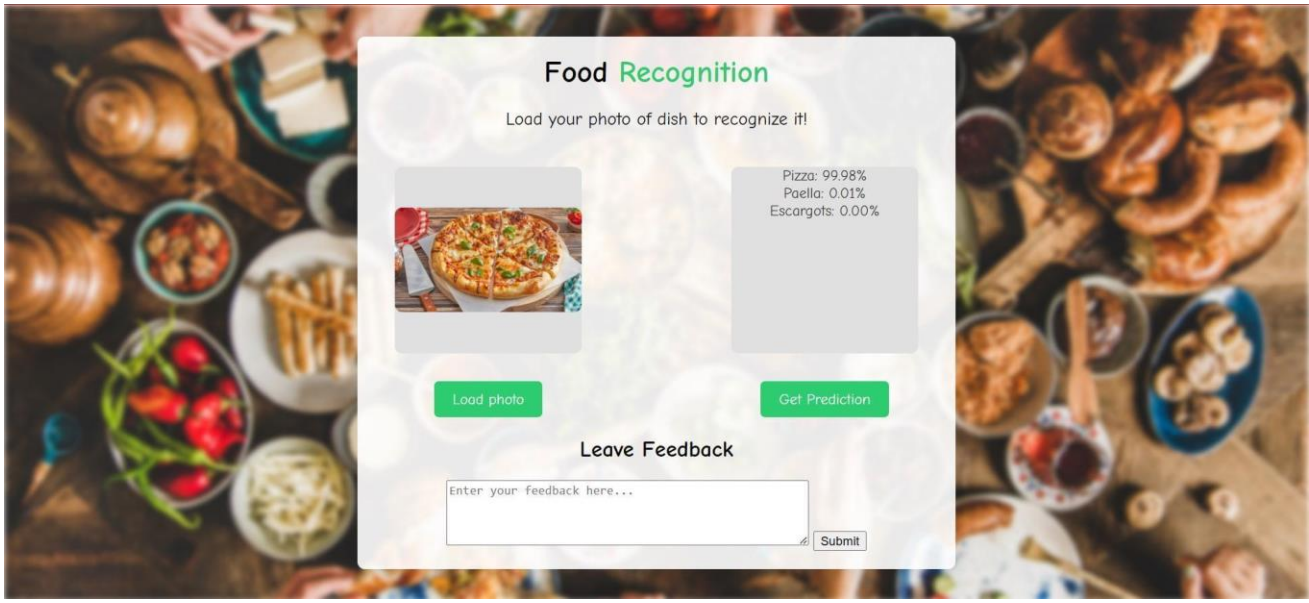

Рисунок 3.4 – Результат роботи системи при розпізнаванню піци

На рисунку 3.5 зображено результат роботи системи при розпізнаванні пончика.

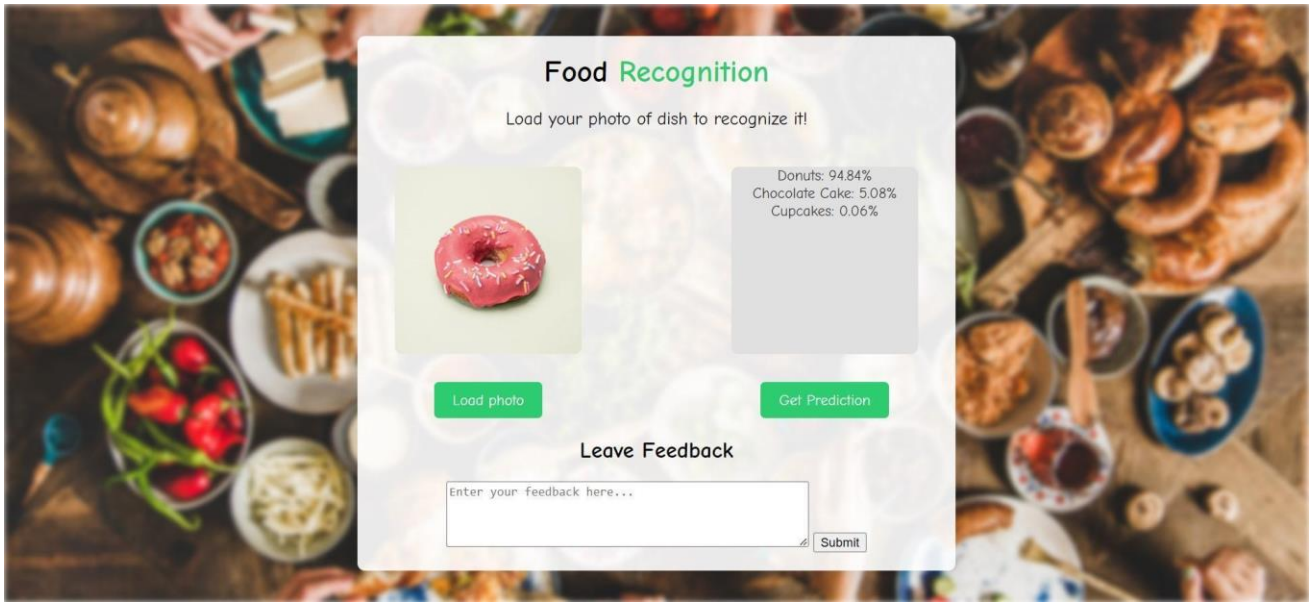

Рисунок 3.5 – Результат роботи системи при розпізнаванні пончика

ВИСНОВКИ

В ході виконання кваліфікаційної роботи, було розроблено веб-систему для розпізнавання їжі по фото засобами комп’ютерного зору.

На етапі аналізу існуючих рішень було проведено огляд сучасних технологій та алгоритмів розпізнавання їжі. Встановлено, що більшість існуючих систем використовують нейронні мережі та глибоке навчання для досягнення високої точності розпізнавання.

Під час вибору алгоритмів та методів для розробки системи, було обрано нейронні мережі через їх високу ефективність в обробці зображень. Зокрема, використовувалися архітектури Convolutional Neural Network (CNN), які показали найкращі результати для задач розпізнавання об'єктів на зображеннях.

Для навчання моделі розпізнавання було використано датасет Food-101, який містить 101 категорії їжі та понад 100 000 зображень . Проведено попередню обробку даних, включаючи нормалізацію та аугментацію зображень, що дозволило покращити якість навчання моделей.

На етапі розробки та навчання моделі було використано бібліотеки TensorFlow та Keras. Моделі були навчальні на оброблених даних з використанням GPU для прискорення процесу навчання. Досягнуто високої точності розпізнавання їжі, що підтверджується результатами тестування.

Для інтеграції моделі з веб-інтерфейсом було застосовано сучасні веб-технології, такі як HTML, CSS та JavaScript. Реалізовано інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам завантажувати зображення їжі та отримувати результати розпізнавання в реальному часі.

Проведено комплексне тестування системи. Результати тестування показали високу точність розпізнавання їжі, що свідчить про ефективність розробленої системи.

Розроблена система має значну практичну цінність для широкого спектру застосувань. Зокрема, вона може бути використана в ресторанах для автоматизації процесів, таких як обробка замовлень та відстеження споживання інгредієнтів. У

додатках для здорового харчування система допомагає користувачам контролювати та аналізувати своє харчування шляхом розпізнавання та підрахунку споживаних страв. Крім того, система знаходить застосування в інших галузях, де важливо швидко і точно ідентифікувати їжу на зображеннях, наприклад, у дослідницьких проектах, пов'язаних з аналізом харчових звичок, або у програмному забезпеченні для кулінарних блогів та соціальних мереж, де користувачі діляться фотографіями їжі.

Рекомендації для подальших досліджень: покращення алгоритмів розпізнавання для ще вищої точності, розширення датасету для включення більшої кількості категорій їжі, інтеграція додаткових функцій, таких як розрахунок калорійності та харчової цінності страв.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. LogMeal Food AI - Image API for Food Detection Food Tracking based on the most Advanced Deep Learning. URL: https://logmeal.com/api
2. Calorie Mama Food Recognition API. URL: https://dev.caloriemama.ai/
3. Python 3.12.3 documentation. URL: https://docs.python.org/3/
4. W3Schools HTML Tutorial. URL: https://www.w3schools.com/html/
5. PHP Manual. URL: https://www.php.net/manual/en/
6. W3Schools CSS Tutorial. URL: https://www.w3schools.com/css/
7. JavaScript.info. URL: https://javascript.info/
8. Документація OpenCV. URL: https://docs.opencv.org/
9. Бібліотека os (стандартна бібліотека Python). URL: https://docs.python.org/3/library/os.html
10. Бібліотека shutil (стандартна бібліотека Python). URL: https://docs.python.org/3/library/shutil.html
11. Модуль collections (стандартна бібліотека Python). URL: https://docs.python.org/3/library/collections.html
12. NumPy. URL: https://numpy.org/
13. Convolutional Neural Networks (CNNs) by Stanford University. URL: https://cs231n.github.io/convolutional-networks/
14. Recurrent Neural Networks (RNN) by Andrej Karpathy. URL: http://karpathy.github.io/2015/05/21/rnn-effectiveness/
15. Deep neural networks. URL: https://en.wikipedia.org/wiki/Deep_learning#Deep_neural_networks
16. Документація TensorFlow. URL: https://www.tensorflow.org/api_docs
17. Sublime Text. URL: https://www.sublimetext.com/
18. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

19. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

20. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

21. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

22. Singla, A., Yuan, L., Ebrahimi, T. (2016). Food/non-food image classification and food categorization using pre-trained googlenet model. In Proceedings of the 2nd International Workshop on Multimedia Assisted Dietary Management (pp. 3-11).

23. Liu, C., Cao, Y., Luo, Y., Chen, G., & Vokkarane, V. (2016). DeepFood: Deep learning-based food image recognition for computer-aided dietary assessment. Journal of Visual Communication and Image Representation, 46, 412-419.

24. Chen, X., Zhang, H., & Xu, X. (2016). Image-based food recognition using deep convolutional network and compressed fisher vector. In Proceedings of the ACM International Conference on Multimedia (pp. 99-103).

25. The Food Recognition Benchmark: Using Deep Learning to Recognize Food in Images. URL: https://www.frontiersin.org/articles/10.3389/fnut.2022.875143/full

26. YOLO: Real-Time Object Detection. URL: https://pjreddie.com/darknet/yolo/

27. GourmetNet: Food Segmentation Using Multi-Scale Waterfall Features with Spatial and Channel Attention. URL: https://www.mdpi.com/1424-8220/21/22/7504

28. Клієнт-серверна архітектура. URL: https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0

29. Комар М.П., Саченко А.О., Васильків Н.М., Гладій Г.М., Коваль В.С., Ліп’яніна-Гончаренко Х.В. Методичні рекомендації до виконання кваліфікаційної роботи з освітньо-професійної програми «Комп’ютерні науки» спеціальності 122 «Комп’ютерні науки» за першим (бакалаврським) рівнем вищої освіти. Тернопіль: ЗУНУ, 2024. 52 с.

30. Перспективні напрямки розвитку економіки, обліку, управління та права: теорія і практика: збірник тез доповідей міжнародної науково-практичної конференції (Ізмаїл, 18 травня 2024 р.). Ізмаїл: ЦФЕНД, 2024. 63 с.

ДОДАТОК А

Код системи

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link
href="https://fonts.googleapis.com/css2?family=Comic+Neue:wght@300;400;700&display=
swap" rel="stylesheet">
  <link rel="stylesheet" href="styles.css">
  <title>Food Recognition</title>
</head>
<body>
  <div class="wrapper">
    <!-- header -->
    <header>
      <h1>Food <span class="highlight">Recognition</span></h1>
    </header>

    <!-- main -->
    <main>
      <form id="uploadForm" enctype="multipart/form-data">
        <div class="instruction">
          <p>Load your photo of dish to recognize it!</p>
        </div>
        <div class="upload-section">
          <div class="preview-buttons">
            <div class="image-preview">
              <img id="preview" src="" alt="Image preview">
            </div>
```

```
    <label for="fileInput" class="upload-button">
     Load photo
     <input type="file" id="fileInput" name="image" accept="image/*">
    </label>
   </div>
        <div class="preview-buttons">
         <div class="text-preview">
          <textarea id="predictionText" readonly></textarea>
         </div>
         <label for="getPredictionInput" class="prediction-button">
          Get Prediction
          <input type="button" id="getPredictionInput" onclick="getPrediction()"
style="display: none;">
         </label>
        </div>
       </div>
       <div class="feedback">
        <h2>Leave Feedback</h2>
        <textarea id="feedbackText" rows="4" cols="50" placeholder="Enter your feedback
here..."></textarea>
        <button onclick="submitFeedback()">Submit</button>
      </div>
     </form>
    </main>
   </div>
<script>
    function previewImage(event) {
     const file = event.target.files[0];
     if (file) {
      const reader = new FileReader();
      reader.onload = function(e) {
       const preview = document.getElementById('preview');
       preview.src = e.target.result;
```

```
        preview.style.display = 'block';
      };
      reader.readAsDataURL(file);
    }
  }


  function getPrediction() {
    fetch('process_image.php', {
      method: 'POST',
      body: new FormData(document.getElementById('uploadForm'))
    })
    .then(response => response.text())
    .then(data => {
      const predictionText = document.getElementById('predictionText');
      predictionText.value = data;
    })
    .catch(error => {
      console.error('Error:', error);
      alert('An error occurred while getting the prediction. Please try again later.');
    });
  }

  document.getElementById('fileInput').addEventListener('change', previewImage);
</script>
</body>
</html>

 body {
   font-family: 'Comic Neue', sans-serif;
   margin: 0;
   display: flex;
   justify-content: center;
   align-items: center;
```

```
  height: 100vh;
  overflow: hidden;
  position: relative;
}

body::before {
  content: "";
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background: url('images/background.jpg') no-repeat center center fixed;
  background-size: cover;
  filter: blur(3px);
  z-index: -1;
}

.wrapper {
  background-color: rgba(255, 255, 255, 0.9);
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 0 20px rgba(0, 0, 0, 0.1);
  text-align: center;
  width: 80%;
  max-width: 600px;
  position: relative;
  z-index: 0;
}
header h1 {
  font-size: 2em;
  margin: 0;
}
```

```
header .highlight {
  color: #2ECC71;
}

.instruction p {
  font-size: 1.2em;
  margin: 20px 0;
}

.upload-section {
  display: flex;
  flex-direction: row;
  justify-content: space-between;
}

.preview-buttons {
  display: flex;
  flex-direction: column;
  align-items: center;
}

.upload-button, .prediction-button {
  display: inline-block;
  background-color: #2ECC71;
  color: white;
  padding: 10px 20px;
  border-radius: 5px;
  cursor: pointer;
  position: relative;
  margin-top: 10px;
}
```

```
.upload-button input[type="file"] {
  position: absolute;
  left: 0;
  top: 0;
  opacity: 0;
  width: 100%;
  height: 100%;
  cursor: pointer;
}

.image-preview, .text-preview {
  width: 200px;
  height: 200px;
  background-color: #e0e0e0;
  border-radius: 8px;
  display: flex;
  align-items: center;
  justify-content: center;
  margin: 20px;
}

.image-preview img {
  max-width: 100%;
  max-height: 100%;
  border-radius: 8px;
  display: none;
}

.text-preview textarea {
  width: 100%;
  height: 100%;
  border: none;
```

```
  resize: none;
  background: transparent;
  text-align: center;
  color: #333;
  font-size: 1em;
  font-family: 'Comic Neue', sans-serif;
  pointer-events: none;
}


<?php
if ($_SERVER["REQUEST_METHOD"] == "POST" && isset($_FILES['image'])) {
    $target_dir = "uploads/";
    $target_file = $target_dir . "photo.jpg";
    $uploadOk = 1;

    $check = getimagesize($_FILES["image"]["tmp_name"]);
    if ($check === false) {
        $uploadOk = 0;
    }
    if ($_FILES["image"]["size"] > 500000) {
        $uploadOk = 0;
    }
    if ($uploadOk != 0 && move_uploaded_file($_FILES["image"]["tmp_name"],
$target_file)) {
        $pythonScriptPath = "code/food_recognition.py";
        $command = "python3 $pythonScriptPath 2>&1";

        exec($command, $output, $return_var);
        $result = trim(file_get_contents("result.txt"));
        echo htmlspecialchars($result);
    }
}
?>
```

```
import cv2
import os
import numpy as np
from tensorflow.keras.models import load_model

image_path = 'uploads/photo.jpg'
image = cv2.imread(image_path)

image_food = cv2.resize(image, (224, 224))
image_food = image_food.astype('float32') / 255.0
image_food = np.expand_dims(image_food, axis=0)

model_food = load_model('model/food_recognition_model.h5')
predictions_food = model_food.predict(image_food)

food_clases = []
with open('model/coco.names', 'r') as f:
    food_clases = f.read().splitlines()

result_text = ""
top_n_indices = np.argsort(predictions_food[0])[::-1][:3]
for idx in top_n_indices:
    label_food = food_clases[idx]
    confidence_food = predictions_food[0][idx] * 100
    result_text += f"{label_food}: {confidence_food:.2f}%\n"

with open('result.txt', 'w') as file:
    file.write(result_text)
```

```
import os
from shutil import copy
from collections import defaultdict

def prepare_data(filepath, src, dest):
    classes_images = defaultdict(list)

    with open(filepath, 'r') as file:
        paths = [line.strip() for line in file.readlines()]
        for path in paths:
            food_class, image = path.split('/')
            classes_images[food_class].append(image + '.jpg')

    for food_class, images in classes_images.items():
        dest_folder = os.path.join(dest, food_class)
        if not os.path.exists(dest_folder):
            os.makedirs(dest_folder)
        for image in images:
            src_path = os.path.join(src, food_class, image)
            dest_path = os.path.join(dest, food_class, image)
            copy(src_path, dest_path)

    print("Copying Done!")

prepare_data('food-101/meta/train.txt', 'food-101/images', 'food-101/train')
prepare_data('food-101/meta/test.txt', 'food-101/images', 'food-101/test')
```

ДОДАТОК Б

Апробовані результати

УДК 004.9

Штинда В. В.
студент групи КН-42
Західноукраїнський національний університет
Вітенко В. В.
аспірант
Західноукраїнський національний університет

ВЕБ-БАЗОВАНА СИСТЕМА РОЗПІЗНАВАННЯ ЇЖІ ПО ФОТО ЗАСОБАМИ КОМП'ЮТЕРНОГО ЗОРУ

Розпізнавання їжі є важливим аспектом для осіб, які контролюють свою вагу, мають діабет або інші захворювання, що вимагають ретельного моніторингу споживання їжі. Завдяки широкому поширенню смартфонів, розробляється все більше систем, що використовують звичайну камеру зі смартфону для автоматичного розпізнавання об'єктів, зокрема для розпізнавання зображення їжі [1].

Метою даного проекту є розробка веб-базової системи, що повинна автоматично розпізнавати різні види їжі на фото за допомогою комп'ютерного зору. Основною метою є створення інструменту, який дозволить користувачам швидко та ефективно отримувати інформацію про страви на основі фотографій.

Розпізнавання їжі по фото є складною задачею через велику кількість страв, їх розмірів та кольорів. Для розпізнавання об'єктів на зображеннях запропоновано використання методів комп'ютерного зору та глибокого навчання, зокрема згорткові нейронні мережі (CNN), які можуть бути навчені розпізнавати різні види їжі на зображеннях. Саме CNN користуються великою популярністю як засіб для класифікації та категоризації зображень [2], основний принцип їх роботи полягає у використанні фільтрів (ядра) для виявлення різних особливостей на зображеннях, таких як границі, текстури та форми. Після цього застосовується операція, яка допомагає зменшити розмір зображення та спростити обробку. CNN можуть аналізувати деталі на різних рівнях складності, що дозволяє їм ефективно впізнавати об'єкти на зображеннях різного виду. Також вони використовуються для різних завдань, таких як класифікація зображень, визначення об'єктів та глибоке навчання. CNN є дуже ефективним інструментом для розпізнавання об'єктів на зображеннях, тому їх часто використовують у системах розпізнавання їжі за фотографіями [3].

Для взаємодії з користувачем запропонована розробка веб-системи з використанням веб-технологій HTML, CSS та JavaScript які дозволять завантажувати фотографії страв, передавати їх на сервер, де за допомогою алгоритмів обробки зображень та аналізу даних буде визначатись вміст страви та передаватись результат користувачу [4]. Така система стане хорошим помічником в сучасному дієтологічному менеджменті та харчовій промисловості, пропонуючи підходи, що сприяють здоров'ю та харчовій безпеці.

Список літератури

1. F. Zhu, M. Bosch, N. Khanna, C. J. Boushey and E. J. Delp, "Multiple Hypotheses Image Segmentation and Classification With Application to Dietary Assessment," in IEEE Journal of Biomedical and Health Informatics, vol. 19, no. 1, pp. 377-388, Jan. 2015, doi: 10.1109/JBHI.2014.2304925.

2. ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012) - [Електронний ресурс] - https://image-net.org/challenges/LSVRC/2012/index.php

3. Umberto Michelucci. Advanced Applied Deep Learning Convolutional Neural Networks and Object Detection, 2019. 285 c.

4. Chollet, François. Deep Learning with Python. Manning Publications, 2018. 384 p.