МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) автоматизації і комп'ютерних систем імені проф. І.В. Ельперіна

Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ Андрій ФОРСЮК	_____ Сергій ГРИБКОВ
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«08» грудня 2025р.	«08» грудня 2025р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

зі спеціальності 122 Комп'ютерні науки
(код та назва спеціальності)

освітньо-професійної програми Управління інформацією та аналітика даних

на тему: Інформаційна система аналізу характеристик харчових продуктів на основі комп'ютерного зору

Виконав: здобувач 2 курсу, групи КН-2-2М

Холявка Кирило Юрійович
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник Мошенський Андрій Олександрович _____
(прізвище , ім'я та по батькові повністю) (підпис)

Консультанти_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент _____ _____
(ім'я та прізвище) (підпис)

Я як здобувач Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав і не одержував недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ – 2025р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) автоматизації і комп'ютерних систем імені проф. І.В. Ельперіна
Кафедра інформаційних технологій, штучного інтелекту і кібербезпеки
Освітній ступінь магістр
Спеціальність 122 Комп'ютерні науки
(код і назва)
Освітньо–професійна програма _Управління інформацією та аналітика даних
(назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інформаційних технологій, штучного інтелекту і кібербезпеки

_____ Сергій ГРИБКОВ
«05» ____листопада_____2025року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА

Холявки Кирила Юрійовича
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна система аналізу характеристик харчових продуктів на основі комп'ютерного зору
керівник роботи Мошенський Андрій Олександрович, доцент, кандидат технічних наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від 05 листопада 2025 року №906-кс
2. Строк подання здобувачем роботи 01 грудня 2025 року
3. Вихідні дані до роботи Розгорнута та налаштована серверна інфраструктура. Натренована модель комп'ютерного зору використовується для класифікації продуктів. Додана модель оптичного розпізнавання тексту. Для збагачення даних (калорійність, склад) система інтегрована з відкритою базою OpenFoodFacts.
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Розділ 1. Теоретичні основи та сучасний стан досліджень у галузі комп'ютерного зору для аналізу харчових продуктів
Розділ 2. Архітектура та структура інформаційної системи аналізу харчових продуктів
Розділ 3. Реалізація, тестування та оцінка ефективності інформаційної системи
5. Перелік графічного матеріалу:

_____
_____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	доц. Мошенський А.О.	01.10.2025	10.10.2025
2	доц. Мошенський А.О.	10.10.2025	30.10.2025
3	доц. Мошенський А.О.	01.11.2025	20.11.2025
4	доц. Мошенський А.О.	20.11.2025	29.11.2025

7. Дата видачі завдання_____1 жовтня 2025 року_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Огляд сучасних методів розробки записок кваліфікаційних робіт.	01.10.2025 – 10.10.2025	Виконано
2	Розробка першого розділу.	10.10.2025 – 30.10.2025	Виконано
3	Розробка другого розділу.	01.11.2025 – 20.11.2025	Виконано
4	Розробка третього розділу.	20.11.2025 – 29.11.2025	Виконано
5	Оформлення презентації.	29.12.2025 – 01.12.2025	Виконано

Здобувач _____ Кирило ХОЛЯВКА____
(підпис) (ім’я та прізвище)

Керівник роботи _____ Андрій МОШЕНСЬКИЙ
(підпис) (ім’я та прізвище)

АНОТАЦІЯ

Холявка Кирило Юрійович – Інформаційна система аналізу характеристик харчових продуктів на основі комп'ютерного зору.

Кваліфікаційна робота присвячена розробці та реалізація інформаційної системи для автоматичного аналізу характеристик харчових продуктів на основі комп’ютерного зору. Актуальність теми зумовлена зростаючою потребою в автоматизації процесів розпізнавання товарів у роздрібній торгівлі. Метою розробки та реалізації інформаційної системи для автоматичного аналізу характеристик харчових продуктів на основі комп’ютерного зору, у тому числі елементів їх упаковки (етикетки, текст). У роботі проведено порівняльний аналіз сучасних методів комп'ютерного зору на основі глибокого навчання (таких як YOLO, CNN, багатомодальні моделі), які застосовуються для виявлення, розпізнавання та класифікації харчових продуктів. Особливу увагу приділено практичному застосуванню цих технологій у сфері роздрібної торгівлі та сценаріях самостійної взаємодії користувачів з продуктами. За результатами дослідження визначено алгоритми та моделі, що забезпечують найвищу точність і швидкість автоматизованого розпізнавання харчової продукції.

Ключові слова: КОМП’ЮТЕРНИЙ ЗІР, ХАРЧОВІ ПРОДУКТИ, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, ГЛИБИННЕ НАВЧАННЯ, МОБІЛЬНІ МЕРЕЖІ, РОЗПІЗНАВАННЯ БРЕНДУ

SUMMARY

Kirylo Yuriyovych Kholyavka – Information system for analyzing food product characteristics based on computer vision.

The thesis focuses on the development and implementation of an information system for the automatic analysis of food product characteristics using computer vision. The relevance of the topic stems from the growing need for automating product recognition processes in retail trade. The purpose of the development and implementation of an information system for the automatic analysis of food product characteristics based on computer vision, including elements of their packaging (labels, text). The work provides a comparative study of modern deep learning-based computer vision methods (such as YOLO, CNN, multimodal models) used for the detection, recognition, and classification of food products. Particular attention is paid to the practical application of these technologies in the retail sector, as well as scenarios of independent user interaction with products. Based on the study's results, algorithms and models that provide the highest accuracy and speed in automated food product recognition have been identified.

Keywords: COMPUTER VISION, FOOD PRODUCTS, IMAGE CLASSIFICATION, DEEP LEARNING, MOBILE NETWORKS, BRAND RECOGNITION

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

Скорочення	Розшифрування / Пояснення (розмірність – за потреби)
API	Application Programming Interface – інтерфейс прикладного програмування
CPU	Central Processing Unit – центральний процесор
CNN	Convolutional Neural Network – згорткова нейронна мережа
GPU	Graphics Processing Unit – графічний процесор
IoT	Internet of Things – Інтернет речей
JSON	JavaScript Object Notation – формат обміну даними у вигляді тексту
KCal	Кілокалорія – одиниця вимірювання енергії (кКал)
LLM	Large Language Model – велика мовна модель
ML	Machine Learning – машинне навчання
OCR	Optical Character Recognition – оптичне розпізнавання символів
TPU	Tensor Processing Unit – тензорний процесор (спеціалізований чип для ШІ)

ВСТУП

Кваліфікаційна робота: 73 сторінок, 6 рисунків, 1 таблиць, 2 додатки, 11 джерел.

Швидкий розвиток технологій глибокого навчання суттєво змінив підходи до автоматичного аналізу візуальної інформації. Однією з найбільш перспективних галузей застосування комп'ютерного зору є розпізнавання харчових продуктів за зображенням. Це особливо актуально для торговельної сфери, де виникає потреба в автоматизації процесів ідентифікації товарів, обліку асортименту та поліпшення користувацького досвіду – зокрема, у мобільних додатках, касах самообслуговування та логістичних системах.

Ручне введення або сканування штрих-кодів часто ускладнює роботу з широким асортиментом, особливо коли етикетки пошкоджені, відсутні або непомітні. Тому автоматизовані системи, здатні розпізнавати продукти за візуальними ознаками упаковки та текстовими елементами, мають високий потенціал для практичного впровадження.

Метою цієї роботи є розробка інформаційної системи, яка використовує методи комп'ютерного зору для класифікації зображень харчових продуктів та автоматичного зчитування текстових характеристик, таких як бренд або підкатегорія товару. Для реалізації поставленої мети досліджено архітектуру MobileNetV2 та її адаптацію для задач класифікації fine-класів. Було підготовлено і трансформовано датасет із платформи Kaggle відповідно до потреб навчання моделі, реалізовано алгоритми розпізнавання бренду на основі тексту в назвах зображень і проведено оцінювання ефективності роботи системи в умовах, наближених до реального використання.

Об'єктом дослідження виступає процес візуальної ідентифікації харчових продуктів за допомогою методів глибокого навчання. Предметом дослідження є архітектура, методи та інструменти реалізації інформаційної системи, здатної до автоматичної класифікації продуктів на основі зображень та формування структури текстових характеристик, зокрема бренду, підкатегорії та типу товару.

Наукова новизна роботи полягає в побудові гнучкої системи на базі MobileNetV2, що дозволяє здійснювати точну класифікацію товарів навіть у складних умовах, а також у запропонованому підході до ієрархічного структурування fine-класів за категоріями, брендами та підтипами. Практичне значення розробленої системи полягає в її можливості бути інтегрованою в торговельні або логістичні рішення, що сприятиме ефективнішій ідентифікації продукції, зниженню залежності від штрих-кодів та підвищенню точності в роботі з асортиментом.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ ТА СУЧАСНИЙ СТАН ДОСЛІДЖЕНЬ У ГАЛУЗІ КОМП'ЮТЕРНОГО ЗОРУ ДЛЯ АНАЛІЗУ ХАРЧОВИХ ПРОДУКТІВ

1.1. Галузь застосування та актуальність задачі

У сучасному світі, що стрімко цифровізується, технології комп'ютерного зору стають невід'ємною частиною багатьох галузей, зокрема торгівлі, логістики, охорони здоров'я, харчової промисловості та споживчих сервісів. Вони використовуються для автоматизації процесів, що потребують аналізу зображень, та дозволяють зменшити потребу в людських ресурсах, підвищити точність і швидкість виконання рутинних завдань. Одним із ключових напрямків є ідентифікація харчових продуктів за їх візуальними властивостями – формою, кольором, типом упаковки, розміщенням тексту тощо. Це має вирішальне значення для впровадження нових підходів до обслуговування в супермаркетах, створення інтелектуальних холодильників, персональних помічників з харчування, домашніх інвентаризаційних систем, а також інструментів контролю для людей із особливими дієтичними потребами.

Інформаційна система, що здатна не тільки класифікувати продукт на рівні категорії, а й точно визначати бренд, назву, поживну цінність, спеціальні властивості (наприклад, «безлактозний» або «еко-продукт»), значно розширює можливості застосування. Такий рівень деталізації забезпечує глибшу інтеграцію з іншими технологіями – від онлайн-магазинів до систем рекомендацій та медичних додатків. Це відкриває перспективу створення автономних та персоналізованих рішень, здатних адаптуватися до змін у дизайні упаковок і поповнення асортименту.

1.2. Огляд сучасних інформаційних рішень

Існує ряд сучасних комерційних і дослідницьких систем, які частково реалізують подібні функції. Такі проєкти, як Amazon Go, застосовують складні багатокамерні комплекси для ідентифікації продуктів у режимі реального часу, дозволяючи клієнтам залишати магазин без сканування товарів. Мобільні додатки на кшталт MyFitnessPal і Yazio використовують бази штрихкодів для отримання харчової інформації, однак не вміють розпізнавати продукти за фотографією. Значна частина подібних рішень є закритими, вимагають стабільного інтернет-з'єднання та не підходять для автономного використання. Крім того, вони часто не враховують локальні особливості брендів або мовних маркерів, що знижує їх ефективність у національних умовах.

На цьому тлі зростає потреба у створенні адаптивних, локально функціональних систем, які можуть працювати на простих пристроях без постійного підключення до хмари. Такі системи повинні бути здатні до оновлення бази знань, навчання на нових зразках та обробки запитів із високою швидкістю навіть на бюджетному обладнанні.

1.3. Аналіз предметної галузі

В умовах побутового використання або на невеликих підприємствах, таких як продуктові лавки чи місцеві логістичні центри, постає потреба у створенні систем, які будуть водночас точними, швидкими та простими у впровадженні. Найпоширенішими проблемами, що знижують точність класифікації продуктів, є низька якість зображень, недостатнє освітлення, незвичні ракурси зйомки, а також зміна або схожість упаковок. Для подолання цих обмежень доцільним є комбінування методів комп’ютерного зору та OCR (оптичного розпізнавання тексту), що дозволяє уточнити назву товару за наявністю текстових фрагментів на етикетці або упаковці. Важливим фактором є й вміння системи оновлювати інформацію про нові продукти без необхідності повного перенавчання.

1.4. Документообіг і процес взаємодії

Оскільки розроблювана система є незалежною та не прив'язана до конкретної установи чи компанії, моделювання процесів здійснюється у вигляді логічної схеми. Сценарій взаємодії передбачає: користувач завантажує фото продукту, система проводить попередню обробку зображення, нейронна мережа класифікує продукт за візуальними ознаками, у випадку недостатньої впевненості активується модуль OCR, зчитаний текст аналізується та використовується для уточнення, система звертається до відкритої бази OpenFoodFacts для отримання додаткової інформації, фінальний результат відображається користувачу у зручному форматі. Така послідовність дозволяє гнучко реагувати на неоднозначні випадки та мінімізувати кількість помилкових класифікацій.

1.5. Виявлені задачі та невирішені проблеми

Після аналізу літературних джерел, відкритих датасетів та існуючих систем було виявлено кілька важливих обмежень, що істотно впливають на якість розпізнавання:

1. Більшість відкритих датасетів, таких як GroceryStoreDataset чи RPC, містять обмежену кількість класів і не враховують різноманітність товарів у локальних супермаркетах;
2. Поширені моделі нейронних мереж не вміють обробляти текстову інформацію на упаковках, що знижує точність при класифікації схожих об'єктів;
3. OCR-системи на мобільних пристроях не завжди працюють стабільно через низьку роздільну здатність зображень та вплив зовнішнього освітлення;
4. Відсутність універсального рішення, яке б поєднувало класифікацію з розширеним аналізом харчової інформації.

Ці виклики сформували основу для створення автономної системи нового покоління, що враховує складність реального середовища, мінімальні апаратні вимоги та здатність до інтеграції з відкритими джерелами даних.

1.6. Постановка задачі

У межах даного дослідження ставиться завдання створити інформаційну систему для ідентифікації харчових продуктів за зображенням із можливістю доповнення результату за допомогою технологій OCR та зовнішніх відкритих баз даних. Основні підзадачі проєкту включають:

1. Розробку архітектури класифікаційної моделі на основі MobileNetV2;
2. Створення системи попередньої обробки зображення, адаптованої до мобільних умов зйомки;
3. Реалізацію OCR-модуля з використанням сервісу Docupipe;
4. Підключення до API бази OpenFoodFacts для зчитування калорійності, складу та іншої метаінформації;
5. Розробку користувацького вебінтерфейсу для демонстрації системи та тестування на реальних прикладах;
6. Аналіз точності, продуктивності та обмежень системи за результатами тестування.

1.7. Висновки до першого розділу

Проведене дослідження предметної області продемонструвало високу актуальність задач автоматичного розпізнавання харчових продуктів у контексті побутового використання та малих комерційних систем. Виявлені проблеми з точністю, варіативністю упаковок і недосконалістю існуючих рішень обґрунтували необхідність створення комплексного підходу. Комбінування технологій глибокого навчання, OCR та відкритих даних дозволяє досягти нового рівня функціональності й гнучкості, що й стало головною метою цієї кваліфікаційної роботи.

РОЗДІЛ 2. АРХІТЕКТУРА ТА СТРУКТУРА ІНФОРМАЦІЙНОЇ СИСТЕМИ АНАЛІЗУ ХАРЧОВИХ ПРОДУКТІВ

2.1. Загальна характеристика системи

Інформаційна система аналізу характеристик харчових продуктів на основі комп'ютерного зору призначена для автоматизованої класифікації, ідентифікації та отримання текстової інформації (бренду, типу продукту) з фотографій продуктів харчування. Основна мета системи полягає у використанні методів глибокого навчання для обробки вхідних зображень і виділення характеристик, які можна надалі використовувати для пошуку, аналізу та контролю якості.

Для реалізації моделі було використано відкриті програмні бібліотеки Python, зокрема інструменти для глибокого навчання, обробки зображень та структурування даних. Такий вибір дозволив забезпечити гнучкість розробки й спростити подальше масштабування системи.

Архітектурно система базується на використанні попередньо натренованої моделі **MobileNetV2**, яка була адаптована до специфічної задачі класифікації fine-класів харчових продуктів. Модель приймає зображення, зменшене до розміру 224x224 пікселів, нормалізує їх і передає у мережу, яка на виході видає прогнозований клас. Класи представлені у вигляді розширених позначень, наприклад, "Fruit_Apple_GrannySmith".

Система передбачає попередню обробку датасету: спочатку ієрархічну структуру зображень трансформують у плоску, створюючи окремі директорії для кожного fine-класу. Крім того, на основі назв файлів (наприклад, "CocaCola_123.jpg") автоматично визначається бренд, що дає змогу зв'язувати результати класифікації з конкретними виробниками або торговими марками.

Важливою особливістю системи є можливість масштабування – додавання нових fine-класів або розширення датасету не потребує змін в архітектурі коду, достатньо перетренувати модель. Система може бути легко інтегрована у зовнішні

інформаційні або торгові системи, наприклад, для автоматичного визначення товарів на фото або візуального пошуку в каталозі.

Для тренування та перевірки працездатності системи був використаний датасет, запозичений з платформи Kaggle. Його було адаптовано до потреб системи через переформатування структури каталогів і генерація списку підкласів для детального розпізнавання продуктів.

У якості архітектури нейронної мережі було обрано MobileNetV2 – компактну та ефективну модель, розроблену для мобільних та обмежених середовищ. Основною причиною вибору MobileNetV2 є її здатність забезпечити високу точність при значно меншій кількості параметрів у порівнянні з такими моделями, як ResNet або VGG.

У (табл. 2.1) наведено порівняльні характеристики поширених моделей глибокого навчання, що використовуються для розпізнавання зображень у системах комп'ютерного зору. Дані узагальнено за результатами відкритих експериментів [1].

Таблиця 2.1 – Порівняння основних моделей

Модель	**Параметри (млн)**	**Top-1 Accuracy (%)**	**Особливості**
VGG16	138	~71.5	Високоточна, але повільна та громіздка
ResNet50	25.6	~76.0	Глибока, стабільна, важка для мобільних
EfficientNet-B0	5.3	~77.1	Высока точность при мінімальних параметрах
MobileNetV2	**3.4**	**~71.8**	**Компактна, ідеальна для швидкого inference**

Як видно з таблиці, MobileNetV2 забезпечує компроміс між точністю та продуктивністю, що робить її найкращим вибором для побудови системи аналізу харчових продуктів у рамках даного дослідження.

У даній роботі для класифікації зображень продуктів використано архітектуру MobileNetV2 — компактну згорткову нейронну мережу, оптимізовану для мобільних пристроїв і систем із обмеженими ресурсами [4].

Основною ідеєю MobileNetV2 є використання глибинно-роздільних згорток (depthwise separable convolutions), які дозволяють суттєво зменшити обчислювальні витрати. Замість класичної згортки, що поєднує просторову фільтрацію та об’єднання каналів в одному шарі, тут цей процес поділяється на два етапи:

1. Depthwise-згортка застосовує окремі фільтри до кожного вхідного каналу;
2. Pointwise-згортка (1×1) об'єднує інформацію між каналами.

Такий підхід дозволяє зменшити кількість параметрів приблизно у 8–9 разів, майже без втрати точності.

Крім цього, MobileNetV2 використовує особливі інвертовані резидуальні блоки з лінійними bottleneck-ами. У таких блоках спочатку відбувається розширення кількості каналів (expansion), далі – глибинна згортка, і наприкінці – проекція назад у вузький простір (compression), але без активації. Відсутність нелінійності на останньому шарі дозволяє зберегти максимум інформації.

Для ефективної передачі градієнтів використовується коротке з’єднання (skip-connection) між вхідним і вихідним вузьким шаром, як у ResNet.

Математично, глибинно-роздільна згортка визначається як послідовність двох операцій [4]:

1. Depthwise-згортка: для кожного каналу C вхідного тензора застосовується окреме ядро (2.1).

$$Y_{i,j,c}^{(dw)} = \Sigma_{u=1}^{k} \Sigma_{v=1}^{k} X_{i+u,\, j+v-1,c} \cdot W_{u,v}^{(dw,c)} \tag{2.1}$$

2. Pointwise-згортка: з’єднує ознаки з усіх каналів (2.2).

$$Y_{i,j,k} = \Sigma_{c=1}^{M} Y_{i,j,c}^{(dw)} \cdot W_{c,k}^{(pw)} \tag{2.2}$$

де X – вхідне зображення;

$W^{(dw)}$ – ядра глибинної згортки;

$W^{(pw)}$ – ядра 1×1 згортки;

M – кількість вхідних каналів;

K – розмір ядра;

N – кількість вихідних каналів.

За рахунок цієї конструкції MobileNetV2 дає можливість досягти високої ефективності при значно меншому обсязі обчислень. Це особливо важливо в мобільних застосунках, де ресурси обмежені, в додатку А.3. наведений код навчання моделі на фотографіях з відкритого датасету.

2.2. Структура та компоненти системи

Система реалізована у вигляді модульної архітектури, яка забезпечує простоту розширення та підтримки. Основу складає модуль глибокого навчання, який реалізовано на базі мобільної нейронної мережі MobileNetV2. Він виконує завдання класифікації вхідного зображення до одного з fine-класів. До класифікатора передається зображення, яке попередньо обробляється – виконується його масштабування до стандартного розміру 224x224 пікселів та нормалізація піксельних значень.

Крім основного модуля класифікації, система містить підсистему обробки інформації про бренд, яка аналізує назву файлу та витягує з неї текстову характеристику, наприклад назву виробника. Ця інформація зберігається у словнику (формат JSON), який використовується на етапі передбачення для генерації розширеного результату.

Додатково, система включає механізм збереження та імпорту списку класів (наприклад, з файлу fine_class_names.txt), що забезпечує узгодженість між навченими моделями та інтерфейсом передбачення. Це дозволяє незалежно оновлювати модель або класовий словник без змін основного коду.

Система також може містити інтерфейс користувача (наприклад, веб або десктоп-додаток), який забезпечує завантаження зображення, візуалізацію результатів класифікації та пошук інформації про конкретний товар на основі отриманих характеристик. Таким чином, структура системи дозволяє легко інтегрувати її у більш широкі інформаційні або торгові екосистеми.

2.3. Підготовка даних та структура датасету

Оригінальний датасет, який використовувався для навчання системи, був запозичений з платформи Kaggle – публічного ресурсу для обміну та використання наборів даних у наукових і прикладних проєктах. Цей датасет має ієрархічну структуру, де продукти згруповані по категоріях, що відображають як загальну групу товару (наприклад, Fruit), так і його конкретні підвиди та бренди (наприклад, Apple/GrannySmith або Drink/CocaCola). З метою підготовки до машинного навчання структура даних була змінена: зображення зі всіх підкатегорій переміщено у єдину плоску структуру, де кожен клас має повну ідентифікацію у назві (наприклад, Fruit_Apple_GrannySmith або Drink_Cola_CocaCola). Це дозволяє моделі не лише класифікувати загальний тип товару, але й визначити конкретну назву продукту або бренд, що критично важливо для пошукових систем, автоматизованого управління товарами та інтеграції з інформаційними платформами.

2.4. Визначення бренду продукту

З метою подальшого аналізу та пошуку, під час обробки інформації також формується мапа «fine-клас – бренд», де бренд визначається автоматично за назвою файлу зображення. Наприклад, файл "CocaCola_123.jpg" буде віднесено до бренду "CocaCola". Мапа бренду зберігається у форматі JSON і використовується на етапі розпізнавання для повернення текстового результату, придатного для пошуку в зовнішніх системах.

2.5. Взаємодія компонентів та схема роботи

Інформаційна система складається з кількох ключових компонентів, які взаємодіють між собою для забезпечення повного циклу обробки: від надходження зображення продукту до формування структурованого текстового результату, що містить бренд, тип продукту та підкатегорію.

На вході система приймає зображення продукту у форматі JPG або PNG, яке надходить від користувача або з іншого джерела (наприклад, камери або фотобази). Зображення проходить етап попередньої обробки, включаючи масштабування до фіксованого розміру 224x224 пікселів, нормалізацію та перетворення у формат тензора, сумісний із вхідним шаром нейронної мережі.

Далі зображення передається до попередньо навченої нейронної мережі MobileNetV2. Ця модель видає прогнозовану мітку fine-класу, яка містить інформацію не лише про загальну категорію продукту (наприклад, «Beverage»), а й про його конкретний підтип (наприклад, «Cola») і бренд (наприклад, «CocaCola»). Кожен fine-клас форматується у вигляді структури: Category_Brand_Subclass.

Система виконує інтерпретацію результату – зіставляє індекс, повернутий моделлю, із текстовим значенням класу. Далі здійснюється парсинг цього значення, під час якого виділяється окремо категорія продукту, бренд та підкатегорія. Наприклад, для класу "Beverage_CocaCola_Zero" система визначає: Категорія – "Beverage", Бренд– "CocaCola", Підкатегорія – "Zero".

Отримана структура, зображена на (рис. 2.1), використовується для генерації зрозумілого та зручного для людини результату у вигляді ієрархічного представлення: Категорія > Бренд > Підкатегорія. Такий формат дозволяє зберігати інформацію у базі даних, здійснювати пошук за атрибутами, проводити аналіз асортименту або інтегрувати систему з логістичними чи торговельними платформами.

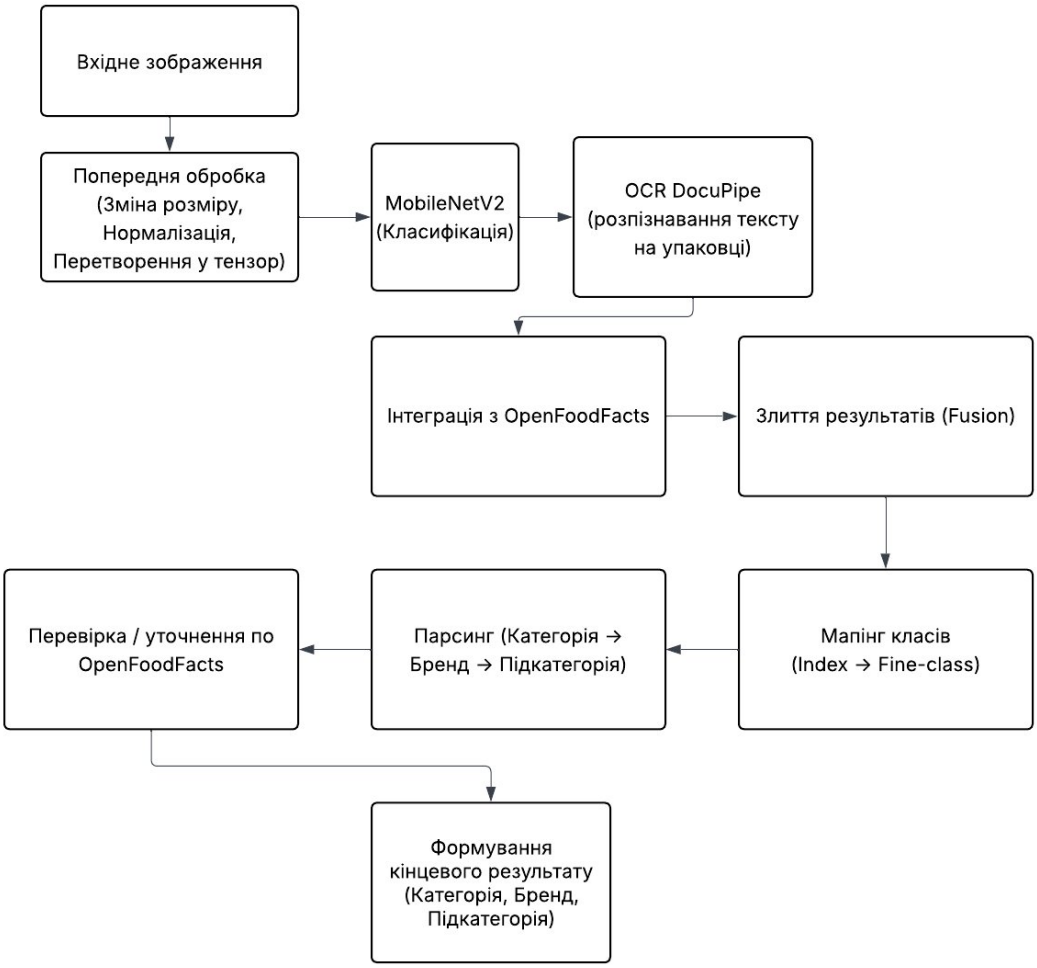

Рисунок 2.1 – Структура процесу обробки зображення

2.6. Архітектура та навчання нейронної мережі

У процесі розробки моделі було застосовано фреймворк PyTorch із використанням попередньо натренованої архітектури MobileNetV2. Кінцевий шар модифіковано відповідно до кількості цільових класів. Навчання здійснювалось на GPU/CPU залежно від наявного середовища. Код адаптовано з офіційної документації PyTorch [2].

model = models.mobilenet_v2(weights=models.MobileNet_V2_Weights.IMAGENET1K_V1)

model.classifier[1] = nn.Linear(model.last_channel, num_classes)

...

for images, labels in train_loader:

```
optimizer.zero_grad()
outputs = model(images)
loss = criterion(outputs, labels)
loss.backward()
optimizer.step()
```

Модель навчалась на GPU або CPU залежно від доступності апаратного забезпечення, і після навчання зберігалась у форматі mobilenetv2_grocery.pth.

2.7. Обмеження та проблеми при класифікації

Незважаючи на використання сучасної моделі MobileNetV2 та адаптованого датасету з fine-класами, під час класифікації зображень продуктів виникають низка типових проблем. Основні труднощі пов’язані із візуальною подібністю між упаковками різних товарів, особливо у категоріях молочних продуктів, напоїв та десертів. Наприклад, зовнішній вигляд упаковок молока, кефіру та йогурту може практично не відрізнятись – часто це схожі за кольором та формою пляшки або пакети.

Це призводить до зниження точності класифікації, оскільки модель орієнтується переважно на візуальні ознаки без урахування текстової інформації. Навіть добре натренована нейронна мережа може помилково віднести зображення йогурту до категорії молока або навпаки.

Щоб компенсувати ці обмеження, у систему було інтегровано модуль OCR (оптичного розпізнавання символів). Після отримання результату класифікації з нейронної мережі, здійснюється додатковий етап – зчитування тексту з етикетки. Якщо впевненість класифікатора низька (наприклад, нижче 70%), результат уточнюється за допомогою розпізнаного тексту. Цей підхід дозволяє підвищити точність у складних випадках.

Модель класифікувала продукт як «Dairy_Milky_Milk», проте OCR-модуль зчитав із етикетки слово «кефір». Після цього було виконано перехресну перевірку з мапою брендів, і остаточний результат був скоригований на «Dairy_Milky_Kefir».

Таким чином, поєднання класифікації зображення з аналізом тексту забезпечує більш гнучкий та надійний механізм ідентифікації товарів, що особливо важливо для реальних умов, де зовнішній вигляд упаковки може бути змінено або ускладнено тінями, освітленням або кутом зйомки.

2.8. Використання OCR для покращення ідентифікації продукту

Однією з ключових переваг розробленої інформаційної системи є поєднання візуального аналізу з текстовими даними, що містяться на упаковці продуктів. Для цього передбачено модуль оптичного розпізнавання символів (OCR), який зчитує такі текстові елементи, як назва бренду, тип продукту, склад або інші характеристичні ознаки.

У межах роботи для реалізації цього модуля було обрано DocuPipe – хмарний OCR-сервіс, який поєднує технології комп'ютерного зору та глибокого навчання. Його архітектура базується на багатоступеневому конвеєрі, що автоматично визначає тип документа, виконує розпізнавання тексту, аналізує табличні структури та формує стандартизований вихід у форматі JSON [9]. Такий підхід забезпечує уніфіковане подання даних і спрощує подальшу обробку.

У DocuPipe використано поєднання нейромережевих моделей OCR для виявлення текстових фрагментів і великих мовних моделей (LLM) для їх контекстного аналізу [5]. Завдяки цьому сервіс може не лише зчитувати текст, а й інтерпретувати його відповідно до заданої схеми полів, формуючи структурований набір даних. Концепція, відома як *schema-first extraction*, дозволяє користувачу наперед визначити потрібні атрибути, після чого система витягує їх у заданому форматі. Таким чином, DocuPipe виступає інтегрованим рішенням, що об'єднує глибинну нейронну мережу для розпізнавання символів із мовними моделями для класифікації та впорядкування результатів.

1. Використовувані моделі та технології розпізнавання тексту

У реалізації OCR-функціоналу DocuPipe застосовуються сучасні методи глибинного навчання. Незважаючи на відсутність публічного опису конкретних

архітектурних рішень, аналіз функціональності сервісу свідчить про використання моделей типу CNN, RNN або трансформерів, що характерно для передових систем оптичного розпізнавання тексту [5], [8].

Сервіс підтримує широкий спектр типів вхідних даних: друкований текст, табличні структури, форми з полями та чекбоксами, а також рукописні записи й підписи [8]. Обробка таких елементів імовірно здійснюється поетапно: спочатку виявляються текстові області, після чого розпізнані символи перетворюються з урахуванням контексту. Для табличних структур DocuPipe використовує спеціалізовані механізми, що дозволяють точно відновити логіку розташування інформації за рядками та стовпцями [8].

На вищому рівні DocuPipe поєднує можливості OCR із великомасштабними мовними моделями (LLM), що забезпечує класифікацію та узагальнення змісту розпізнаного тексту згідно з визначеними схемами. Це дозволяє відразу співвіднести фрагмент тексту з відповідним полем у структурі документа, наприклад, «Найменування товару» чи «Дата виготовлення» [5], [6].

Однією з переваг сервісу є можливість верифікації джерела кожного витягнутого значення: система прив’язує його до конкретної області на зображенні, що підвищує прозорість та точність обробки [5]. Такий підхід є критичним для сфер, де важлива повна достовірність документів — наприклад, у фінансах, охороні здоров’я або логістиці.

Крім цього, DocuPipe підтримує розпізнавання понад 60 мов, зокрема англійської, французької, німецької, китайської, арабської, івриту тощо [6]. Така багатомовна адаптивність робить сервіс придатним для використання в міжнародних системах обробки харчових етикеток, супровідних інструкцій або імпортної продукції.

2. Переваги DocuPipe для проєктів з обмеженими ресурсами (порівняно з EasyOCR та PaddleOCR)

Однією з ключових переваг використання DocuPipe у системах із обмеженими обчислювальними можливостями є його хмарна архітектура. Всі основні етапи обробки – зокрема розпізнавання тексту, класифікація документів та

структурування даних – виконуються на стороні сервера, що дозволяє суттєво зменшити вимоги до клієнтського пристрою. У порівнянні з локальними OCR-інструментами на кшталт EasyOCR або PaddleOCR, які потребують значного обсягу оперативної пам’яті та GPU для ефективної роботи, DocuPipe забезпечує мінімальне локальне навантаження [7].

Як зазначає автор у статті на Medium, EasyOCR показує прийнятні результати лише при використанні GPU. У випадку відсутності графічного прискорювача рекомендується використовувати Tesseract як менш ресурсоємну, але й менш точну альтернативу [7]. PaddleOCR орієнтований на легкість та вбудовуваність, однак його впровадження вимагає налаштування середовища та оптимізації моделей для конкретної апаратної платформи [7]. Основні переваги DocuPipe:

2.1. Відсутність локального навантаження. Оскільки DocuPipe є API-сервісом, клієнт не виконує жодних обчислень, пов’язаних із глибоким навчанням. Достатньо просто передати зображення на сервер і отримати розпізнаний текст або структуровані дані. Це дозволяє використовувати систему навіть на малопотужних пристроях – смартфонах, одноплатних комп’ютерах або в умовах обмеженого хостингу [6].

2.2. Висока швидкість обробки та серверна оптимізація. Обробка відбувається на високопродуктивних серверах з GPU або TPU, що дозволяє DocuPipe працювати значно швидше, ніж EasyOCR або PaddleOCR без апаратного прискорення [5].

2.3. Спрощена інтеграція без глибоких знань у сфері ML. Щоб скористатися DocuPipe, не потрібно встановлювати або налаштовувати фреймворки типу PyTorch чи TensorFlow. Інтеграція зводиться до виконання HTTP-запитів через API з авторизацією по ключу. Це зменшує кількість помилок, полегшує деплой та прискорює розробку [6].

2.4. Обробка складних макетів і документів. DocuPipe має вбудовану підтримку класифікації документів, розпізнавання таблиць, рукописних записів тощо [9]. Локально для цього знадобилася б додаткова логіка або кілька бібліотек. У DocuPipe весь цей функціонал вже реалізований та оптимізований.

Таким чином, у сценаріях, де залізо обмежене (наприклад, IoT-пристрій, веб-сервіс на бюджетному хостингу, або просто відсутня GPU), DocuPipe є доцільним вибором. Ви делегуєте важку роботу сервісу, отримуючи високу точність розпізнавання та широкі можливості без тягаря на власних ресурсах. Важливо врахувати лише затримку на мережевий запит та можливі обмеження тарифу, але за умови стабільного інтернет-з'єднання DocuPipe надає готове рішення "OCR як сервіс", яке перевершує локальні бібліотеки по простоті використання в розподілених або малопотужних системах.

3. Безкоштовне використання: ліміти та можливості

Однією з вагомих причин популярності DocuPipe серед розробників є доступна політика безкоштовного використання. Відповідно до офіційної інформації, нові користувачі отримують 300 безкоштовних кредитів після реєстрації, кожен із яких відповідає приблизно одній обробці сторінки або зображення [6]. Це дозволяє одразу реалізувати прототип, провести первинне тестування сервісу або опрацювати невеликий датасет без фінансових витрат.

Окрім стартового бонусу, сервіс надає постійний безкоштовний тариф. Станом на 2025 рік він передбачає 100 кредитів щомісяця, які автоматично поновлюються [5]. Це забезпечує змогу щомісяця обробляти до сотні сторінок безкоштовно, що є цілком достатнім для невеликих або експериментальних проєктів.

Важливо, що на безкоштовному плані користувач має доступ до всіх основних функцій DocuPipe – розпізнавання друкованого й рукописного тексту, підтримка таблиць, багатомовність, структура JSON тощо. Обмеження стосуються лише кількості обробок, а не якості або доступних можливостей. Після вичерпання місячного ліміту є два варіанти: чекати на новий період або перейти на платний тариф з більшими квотами. Також зазначається, що 300 бонусних кредитів при реєстрації – разові, і надалі діятиме тільки стандартний місячний ліміт [6].

Для масштабних потреб DocuPipe пропонує гнучку тарифну сітку, аж до рішень з можливістю обробки мільйонів документів (за індивідуальним запитом до команди підтримки) [6].

Завдяки такій моделі, сервіс спрощує вхід у розробку OCR-рішень. Розробник може інтегрувати API, перевірити якість розпізнавання на власному наборі зображень, реалізувати MVP-функціонал – усе це без жодних витрат на першому етапі. У разі масштабування — легко перейти до платного варіанту.

У порівнянні з аналогами, такими як Google Vision OCR, які теж мають безкоштовні квоти, DocuPipe вигідно вирізняється своєю щедрою політикою початкового доступу: 300 запитів одразу плюс щомісячне оновлення без потреби вводити платіжні дані на старті. Це робить його особливо привабливим для студентських, академічних чи MVP-проєктів із мінімальним бюджетом [6].

4. Чому DocuPipe обрали для системи аналізу характеристик харчових продуктів

У межах розробки системи, що має аналізувати характеристики харчових продуктів на основі зображень упаковки та етикеток, DocuPipe продемонстрував оптимальне співвідношення функціональності, точності та зручності використання. Зазвичай такі системи повинні зчитувати текстову інформацію з етикеток — назви товарів, склад, енергетичну цінність, дату виготовлення тощо — і перетворювати її у структуровану форму для подальшого аналізу. Саме в цьому полягає спеціалізація DocuPipe, який автоматизує витяг даних із напівструктурованих зображень і документів [6].

Основні переваги DocuPipe для цієї задачі:

4.1. Точність OCR та широкий охоплення випадків. Етикетки продуктів часто містять дрібний текст, змішані шрифти або навіть рукописні позначки (наприклад, дату виготовлення, нанесену вручну). DocuPipe підтримує як розпізнавання друкованого тексту, так і рукописів, що підтверджується оглядами на F6S та G2 [5], [8].

4.2. Обробка таблиць із харчовими показниками. Один із критично важливих елементів етикеток — таблиці з харчовою цінністю (наприклад, вміст білків, жирів, калорій тощо). DocuPipe здатен не лише розпізнати такі таблиці, а й автоматично перетворити їх у структурований формат (JSON) зі збереженням

зв’язку між рядками і стовпцями [8]. Це дозволяє миттєво інтегрувати дані в базу або порівняти кілька продуктів між собою.

4.3. Багатомовність. Харчові продукти часто маркуються кількома мовами, особливо в міжнародному або європейському контексті. DocuPipe підтримує понад 60 мов, включаючи європейські, азійські та правосторонні скрипти [6]. Це дозволяє без додаткових налаштувань коректно розпізнавати як українськомовні, так і англомовні етикетки, без необхідності переключення моделей, як у EasyOCR чи PaddleOCR [7].

4.4. Мінімальні вимоги до пристрою збору зображень. Завдяки хмарній архітектурі вся обробка виконується на сервері. Це дозволяє запускати систему навіть на пристроях із низькою продуктивністю, наприклад, мобільних телефонах або терміналах з камерою. У порівнянні з EasyOCR чи PaddleOCR, які потребують встановлення ML-фреймворків і достатніх ресурсів, DocuPipe дозволяє реалізувати інтерфейс типу "навів камеру – отримав результат" [6].

4.5. Швидке налаштування та адаптація під задачу. DocuPipe дозволяє задавати власні схеми витягу інформації — наприклад, поля “Калорійність”, “Інгредієнти”, “Дата виготовлення” — і автоматично підлаштовує свої моделі під них. Не потрібне тривале навчання або написання парсерів. Завдяки підтримці LLM-моделей, система вчиться на прикладах прямо з інтерфейсу [5].

З огляду на всі ці фактори, вибір впав на DocuPipe як на готове рішення для OCR-аналізу. Його використання дозволило команді сфокусуватися на власне аналізі харчових характеристик (порівнянні продуктів, виведенні результатів користувачу тощо), а не на розробці OCR з нуля. Крім того, економічний фактор також міг зіграти роль: в рамках безкоштовного ліміту сервіс покриває чимало запитів (300 спроб одразу + щомісячний запас) [5][6], чого могло вистачити на стадії розробки і тестування системи. Для порівняння, розгортання власного OCR вимагало б потужного заліза або оренди серверів з GPU, що ускладнює бюджет.

З огляду на ці переваги, вибір на користь DocuPipe був логічним: він дозволив швидко реалізувати повнофункціональний OCR-модуль з підтримкою таблиць, багатомовності та структурованого витягу даних, не перевантажуючи клієнтську

сторону і не потребуючи глибоких ML-налаштувань. У порівнянні з EasyOCR чи PaddleOCR, які могли вимагати додаткової оптимізації, розгортання або навчання, DocuPipe забезпечив готове “з коробки” рішення, що суттєво прискорило створення MVP-прототипу системи аналізу харчових продуктів.

2.9. Перспективи вдосконалення системи

У подальшому інформаційну систему можна вдосконалити декількома важливими напрямками. Передусім, йдеться про можливість донавчання моделі на нових зображеннях з реального середовища, зокрема фотографіях, отриманих із мобільних пристроїв користувачів. Це дозволить покращити узагальнення моделі та зменшити кількість помилок при класифікації нетипових або менш поширених продуктів.

Також варто розглянути впровадження методів аугментації даних, які штучно збільшують обсяг навчального набору за рахунок обертання, масштабування, зміни яскравості та інших трансформацій зображень. Такий підхід позитивно впливає на стійкість моделі до змін умов зйомки.

Ще одним вектором розвитку є розширення логіки системи за рахунок врахування географічної специфіки назв брендів і товарів. Наприклад, один і той самий продукт може мати різну упаковку в різних регіонах, що створює додаткові труднощі при розпізнаванні. Актуальною є також інтеграція з централізованими базами товарів, яка дозволить перевіряти правильність передбачень та формувати повноцінну текстову інформацію про продукт, з можливістю її подальшого пошуку або фільтрації за атрибутами.

2.10. Інтеграція з відкритими базами даних для збагачення результатів

Для забезпечення глибшого аналізу харчових продуктів та підвищення інформативності результатів розпізнавання систему було інтегровано з відкритою базою даних OpenFoodFacts [10]. Це міжнародна ініціатива з відкритим доступом, яка охоплює понад два мільйони продуктів харчування з різних країн світу. У цій

базі міститься структурована інформація про енергетичну цінність (калорійність), харчову складову (вміст білків, жирів і вуглеводів), склад інгредієнтів, наявність харчових добавок (Е-кодів), країну походження товару, а також відповідні штрихкоди (EAN-13) та інші маркери.

OpenFoodFacts надає відкритий програмний інтерфейс (API), який дозволяє здійснювати пошук даних за назвою продукту, торговою маркою або штрихкодом. У рамках цієї роботи API використовується після класифікації продукту за зображенням і розпізнавання тексту за допомогою OCR. Після отримання первинної інформації виконується додатковий запит до бази з метою автоматичного отримання харчової цінності та складу товару, перевірки наявності потенційно небажаних складників, таких як алергени чи харчові добавки, а також для виведення розширеної довідкової інформації у зручному для користувача форматі.

Інтеграція з OpenFoodFacts дозволила значно підвищити функціональність системи без потреби створення та підтримки власної бази харчових характеристик. Крім того, завдяки відкритості платформи та активній участі спільноти у її наповненні, забезпечується актуальність та релевантність отриманих даних, що особливо важливо при роботі з широким і змінним асортиментом продукції.

Таким чином, поєднання можливостей комп’ютерного зору, оптичного розпізнавання тексту та зовнішньої бази OpenFoodFacts сформувало комплексну інтелектуальну систему, здатну не лише визначати харчовий продукт за зображенням, але й надавати користувачу повний опис його складу, походження та потенційного впливу на здоров’я [10].

2.11. Висновки до другого розділу

У цьому розділі проведено комплексний аналіз ключових аспектів розробки інформаційної системи для розпізнавання харчових продуктів на основі технологій комп’ютерного зору. Детально розглянуто підбір та адаптацію датасету Grocery Store Dataset, що дало змогу сформувати набір уточнених класів для більш точного

розпізнавання об’єктів у торговому середовищі. Як основну архітектуру нейронної мережі обґрунтовано вибір MobileNetV2, що поєднує високу точність, обчислювальну ефективність і компактність, особливо актуальну для мобільних або вбудованих рішень.

Особлива увага приділена обмеженням класифікації лише за зображенням, зокрема у випадках подібного дизайну пакування. Для подолання цієї проблеми запропоновано інтеграцію модуля оптичного розпізнавання тексту (OCR), що дозволяє витягувати ключову текстову інформацію з етикеток, уточнюючи результат класифікації та формуючи повноцінний опис товару. Такий підхід підвищує точність системи, а також створює основу для реалізації додаткових функцій: фільтрації, пошуку та збагачення даних.

Загалом запропонована система продемонструвала здатність ефективно вирішувати задачу розпізнавання харчових продуктів у сфері роздрібної торгівлі. Вона поєднує легку нейронну архітектуру, інструменти текстової обробки та відкриті бази даних, що забезпечує її придатність до практичного використання, подальшого масштабування та вдосконалення.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ, ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Реалізація користувацького інтерфейсу та серверної логіки

Інформаційна система розпізнавання харчових продуктів була реалізована у вигляді сучасного веб-додатку, блок схема наведена на рисунку 3.1, що складається з двох основних частин: клієнтської (фронтенд) та серверної (бекенд). Ця архітектура дозволяє ефективно розділити логіку взаємодії з користувачем та обробку даних, що виконується у фоновому режимі.

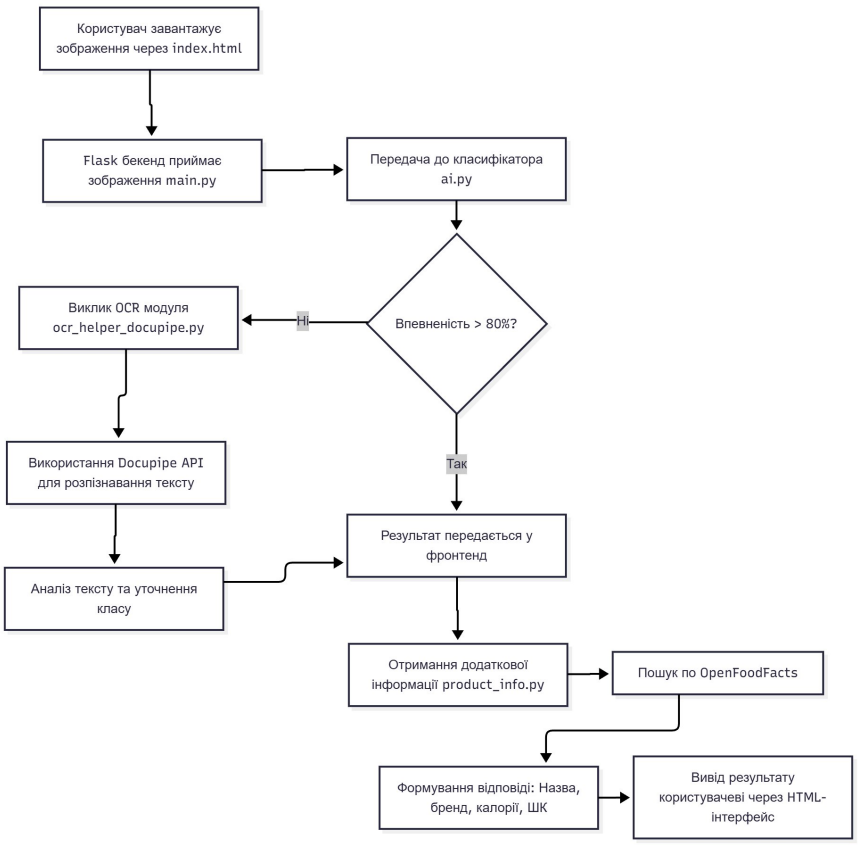

Рисунок 3.1 – Загальна архітектура інформаційної системи

Користувацький інтерфейс реалізований з використанням HTML5, CSS3 та JavaScript, що забезпечує кросбраузерну підтримку, адаптивну верстку для мобільних пристроїв та інтуїтивно зрозумілу взаємодію з системою. Головна сторінка веб-додатку, зображена на рисунку 3.2, містить форму завантаження

зображення, інтуїтивну панель для перетягування файлу (drag-and-drop), можливість попереднього перегляду зображення, кнопку запуску розпізнавання, а також інструкції для користувача. За допомогою стилів CSS була реалізована візуальна ієрархія, що полегшує навігацію, а JavaScript відповідає за динамічну взаємодію з елементами сторінки, включно з попереднім переглядом зображення перед надсиланням:

```
const dropArea = document.getElementById('drop');
dropArea.addEventListener('dragover', (event) => {
  event.preventDefault();
  dropArea.classList.add('highlight');
});
```

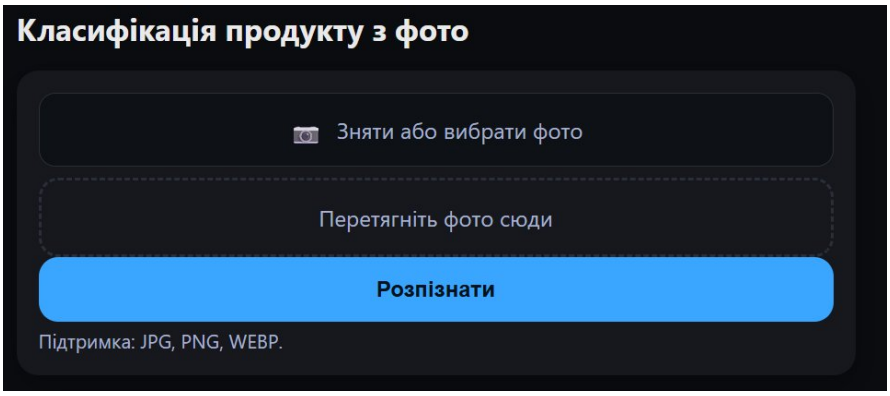

Рисунок 3.2 – Веб-інтерфейс системи для розпізнавання продукту

Після того як користувач обирає або завантажує фотографію, вона передається через форму на бекенд, який реалізовано за допомогою фреймворку Flask. Основна логіка серверної обробки знаходиться у файлі main.py:

```
@app.route("/predict", methods=["POST"])
def predict():
    if "file" not in request.files:
        return redirect(url_for("index"))
    f = request.files["file"]
```

```
    if not f or f.filename == "":
        return redirect(url_for("index"))
    if not _allowed(f.filename):
        return redirect(url_for("index"))
    filename = secure_filename(f.filename)
    save_path = os.path.join(UPLOAD_DIR, f"{datetime.now().strftime('%Y%m%d_%H%M%S')}_{filename}")
    f.save(save_path)
    try:
        result = predict_single_image(save_path, MODEL_PATH, LABELS_PATH)
    except Exception as e:
        result = {"label": f"Помилка: {e}", "confidence": 0.0, "index": -1}
    return render_template("index.html", result=result)
```

Файл ai.py відповідає за виконання класифікації зображення за допомогою попередньо натренованої моделі MobileNetV2. Модель була адаптована для роботи з fine-класами, отриманими з датасету GroceryStore Dataset. Якщо модель демонструє впевненість у передбаченні понад 80%, результат негайно повертається користувачеві. Якщо ж впевненість моделі не перевищує вказаний поріг, активується OCR-модуль, який дозволяє уточнити вміст зображення за допомогою розпізнавання тексту на упаковці:

```
if conf < CONF_THRESHOLD:
    ocr_text = run_ocr(image_path)
    label = refine_label_with_ocr(ocr_text)
```

На початковому етапі розробки було протестовано кілька локальних рішень для OCR – зокрема, EasyOCR та PaddleOCR. Обидві бібліотеки показали хорошу точність розпізнавання, однак виявилися доволі ресурсоємними, що зробило їх неефективними на комп'ютерах із обмеженими обчислювальними можливостями. Через це було прийнято рішення перейти до використання хмарного API Docupipe, який надає сервіс розпізнавання тексту онлайн:

response = requests.post("https://api.docupipe.ai/ocr", files={"file": open(image_path, "rb")})

text_blocks = response.json().get("text", [])

Модуль ocr_helper_docupipe.py обробляє запити до API Docupipe, передаючи зображення та отримуючи текстові фрагменти у відповідь. Після цього виконується фільтрація тексту для знаходження релевантної інформації – брендів, типів продуктів або назв товарів. Результати звіряються з локальною мапою класів, що дає змогу оновити або уточнити початковий результат класифікації.

Після остаточного визначення класу або уточненого результату система звертається до відкритої бази даних OpenFoodFacts:

response = requests.get(f"https://world.openfoodfacts.org/api/v0/product/{barcode}.json")

data = response.json().get("product", {})

Модуль product_info.py виконує запити за назвою продукту або його брендом, отримує JSON-відповідь з детальною інформацією зображено на рисунку 3.2, такою як енергетична цінність, вміст жирів, цукру, білків, баркод, зображення упаковки, посилання на сторінку товару тощо. Ці дані повертаються у фронтенд, де відображаються користувачеві у зручному вигляді.

Рисунок 3.3 – Відповідь з відкритої бази OpenFoodFacts

Таким чином, система побудована за принципом багатошарової перевірки: первинна класифікація з використанням CNN, уточнення за допомогою OCR та доповнення контекстом із відкритих джерел. Такий підхід дозволяє значно підвищити точність розпізнавання, навіть за умов обмеженості ресурсів або низької якості зображення, що робить систему придатною до реального використання у торгівельних точках, на складах або в домашньому господарстві.

3.2. Тестування інформаційної системи

У процесі розробки система була протестована в умовах, максимально наближених до реальних: зйомка відбувалася при звичайному кімнатному освітленні, використовувалися звичайні побутові смартфони, а зображення робилися з різних ракурсів, включно з частковим затемненням, розмитістю або наявністю зайвих об'єктів у кадрі. Для проведення практичного тестування було відібрано 10 товарів, серед яких були свіжі овочі та фрукти (наприклад, банани, яблука, помідори), а також паковані продукти на зразок йогуртів, кефіру, соку та

молока. Частина з цих продуктів була присутня в навчальному наборі даних, інші ж – схожі за виглядом, проте нові для системи. В останньому випадку нейронна мережа пропонувала найбільш імовірний клас зі свого досвіду, що свідчить про її здатність до генералізації, хоча й не без помилок.

Для товарів зі схожим зовнішнім виглядом, таких як паковані молочні вироби, додаткову роль відіграє модуль оптичного розпізнавання тексту. OCR-модуль виявився надзвичайно корисним у ситуаціях, коли упаковка продукту була схожою між різними брендами або коли зображення було неповним. Він успішно розпізнавав маркування, логотипи та текстові елементи на етикетках навіть у складних умовах освітлення або низької контрастності.

Таким чином, у 4 із 10 випадків, коли класифікатор демонстрував низьку впевненість (<80%), OCR дозволив уточнити клас і знайти правильну категорію продукту. У результаті із 10 перевірених зображень було правильно класифіковано 8. Водночас середній час обробки одного запиту, з урахуванням усіх етапів – попередня обробка, класифікація, OCR, пошук у відкритих базах – становив 5–8 секунд, що є прийнятним для інтерактивного використання.

Загальна точність системи при виключно класифікації зображень склала близько 60%. Після активації OCR-модуля вона зросла до понад 80% у тестових сценаріях. Однією з основних проблем під час експериментів залишалась схожість упаковок між різними товарами – наприклад, йогурт та кефір в однакових пластикових стаканчиках, але від різних брендів. Ця проблема була вирішена переважно за рахунок текстового уточнення.

На рисунку 3.4 наведено конкретний приклад з реального використання системи. На першому зображенні зображено товар Oatly – вівсяне молоко. Модель змогла правильно класифікувати загальну категорію продукту (напій на основі вівса) та розпізнати бренд ("Oatly"), OCR також підтвердив цю надану інформації від моделі, але під час пошуку сталася помилка, та обрався не зовсім той продукт, із за не точної класифікації, а саме "Barista Edition". Це пояснюється тим, що в при надсиланні даних не було достатньо інформації з уточненим підтипом цього

продукту. Таким чином, система класифікувала продукт як загальний "вівсяне молоко Barista Edition".

Рисунок 3.4 – Приклад фотографії для використання інформаційної системи

На другому зображенні показано результат, рисунок 3.5, отриманий після активації OCR-модуля. Завдяки використанню API Docupipe було успішно зчитано надпис "OAT-LY!", після чого система звернулась до бази OpenFoodFacts, і на виході користувач отримав не повністю правильну назву продукту а також харчову інформацію: калорійність (61 kcal), вміст жирів (3.0 г), цукрів (3.4 г). Бренд та інше класифікувалось правильно. Це дозволяє не лише суттєво підвищити точність класифікації у випадках невизначеності або неоднозначності, але й розширити функціональність системи, надаючи користувачеві структуровані метадані про продукт. Такі дані можуть бути використані для цілей дієтологічного моніторингу, автоматизованої інвентаризації товарів або контролю відповідності продуктів заданим критеріям (наприклад, вмісту цукру, калорійності чи приналежності до бренду). Водночас тестування виявило окремі обмеження поточної реалізації — зокрема, OCR-модуль коректно зчитував найменування продукту, однак неточність могла виникати під час пошуку відповідного запису в базі OpenFoodFacts. Це вказує на необхідність дотренування класифікаційної моделі з розширеним датасетом, а також вдосконалення логіки пошуку й зіставлення текстових даних із

відкритими джерелами для забезпечення повнішої ідентифікації товарів у реальних умовах.

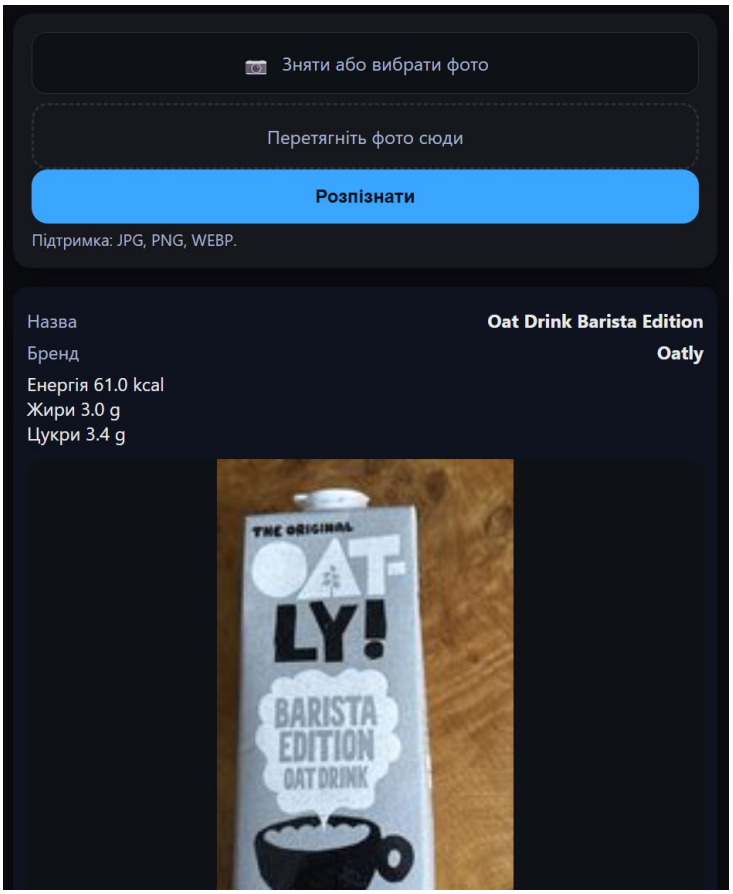

Рисунок 3.5 – Результат розпізнавання зображення пакованого продукту (вівсяного молока Oatly) із використанням OCR-модуля та подальшим пошуком харчової інформації у відкритій базі OpenFoodFacts

Цей приклад ілюструє важливість комбінованого підходу, в якому об'єднано візуальну класифікацію та оптичне розпізнавання тексту. Навіть якщо модель глибокого навчання не забезпечує високої точності, OCR компенсує цей недолік завдяки додатковим семантичним підказкам з етикетки.

3.3. Аналіз результатів та перспективи розвитку системи

Після реалізації та тестування інформаційної системи можна зробити висновок про її функціональність, здатність працювати в умовах, наближених до

реального використання, та її адаптивність до складних ситуацій. Застосування нейронної мережі MobileNetV2 для первинної класифікації дозволило досягти базової точності візуального розпізнавання харчових продуктів. Однак основним викликом залишалась недостатня деталізація – у випадках, коли товари мали дуже схожі упаковки або відрізнялись лише написами на етикетках. У таких ситуаціях нейромережа могла помилково визначити призначення товару або обрати схожий бренд, зокрема при розпізнаванні молока різних типів (звичайне, безлактозне, Barista тощо). Цей недолік частково зумовлений обмеженим обсягом навчального датасету, який охоплював лише обмежену кількість варіацій кожного товару.

Для подолання цих обмежень був інтегрований OCR-модуль на базі сервісу Docupipe. Саме завдяки йому вдалося досягти значного покращення результатів у складних випадках. Наприклад, на рисунку 3.5 показано ситуацію, коли система розпізнала бренд та базову категорію правильно (вівсяне молоко Oatly), але помилилась у призначенні, не врахувавши, що на упаковці була вказана модифікація “Barista Edition”. Цю додаткову специфікацію було виявлено саме OCR-модулем, який зчитав напис "OAT-LY!" з зображення і передав його на подальший аналіз. У відповідь було отримано харчову інформацію з відкритої бази OpenFoodFacts: калорійність 61 ккал, вміст жирів – 3.0 г, цукру – 3.4 г. Таким чином, саме OCR виявився ключовим у розкритті повного профілю продукту. Важливо підкреслити, що пошук у відкритій базі іноді допускав помилки – не через недосконалість OCR, а через нечіткі або неповні відповідники у самій базі даних, що й зумовлювало незначні неточності у кінцевому описі.

Гібридна архітектура, що поєднує глибоке навчання, оптичне розпізнавання тексту та відкриті дані, створює гнучку систему, здатну адаптуватися до нових умов. Проте вже зараз очевидна потреба в подальшому розширенні навчального датасету, додаванні нових товарів, брендів і варіацій упаковок. Дотренування нейронної мережі на більших і більш різноманітних вибірках суттєво покращить її здатність до узагальнення та дозволить уникати помилок у розпізнаванні. Крім того, удосконалення логіки фільтрації результатів OCR та оптимізація пошуку у

відкритих базах даних допоможуть уникнути ситуацій, коли неправильно вибраний запис псує загальну картину.

На основі отриманих результатів можна стверджувати, що система придатна для використання в таких сценаріях, як автоматизоване визначення товарів у магазинах, побудова домашніх каталогів продуктів, інвентаризація або контроль харчування. Її точність, швидкість обробки, а також можливість масштабування на мобільні платформи відкривають широкі перспективи подальшого вдосконалення. Паралельно з цим, передбачається розширення функціоналу: інтеграція з іншими базами, використання штрихкодів для додаткової перевірки, автоматичне збереження результатів у базу даних, а також реалізація голосових підказок для людей із вадами зору.

3.4. Висновки до третього розділу

У третьому розділі було реалізовано ключові компоненти інформаційної системи для розпізнавання та аналізу харчових продуктів на основі технологій комп’ютерного зору та оптичного розпізнавання тексту. Визначено архітектурну побудову програмного забезпечення, обрано ефективні фреймворки та бібліотеки для обробки зображень, класифікації та OCR, серед яких вирішальну роль відіграють TensorFlow, OpenCV, DocuPipe API та OpenFoodFacts API. Проведено інтеграцію моделі MobileNetV2 з навчальним датасетом, адаптованим під обрані fine-категорії продуктів, що дозволило досягти прийнятного балансу між точністю розпізнавання та швидкістю виконання.

Особливу увагу приділено реалізації модуля OCR, який забезпечує зчитування текстових елементів із зображення упаковки, а також використанню відкритих джерел даних для автоматичного збагачення отриманих результатів. Інтерфейс взаємодії з користувачем розроблено таким чином, щоб забезпечити інтуїтивну подачу результатів: від візуалізації класифікованого зображення до виведення структурованої інформації про харчові характеристики. Проведено

базове тестування системи в умовах близьких до реальних, яке підтвердило її працездатність та відповідність функціональним вимогам.

Таким чином, реалізована система не лише виконує завдання розпізнавання об’єктів, а й забезпечує поглиблений аналіз харчового продукту, поєднуючи візуальні та текстові дані з відкритими джерелами, що створює передумови для її подальшого вдосконалення та масштабного впровадження в практичних сценаріях.

ВИСНОВКИ

Кваліфікаційна робота присвячена розробці, реалізації та оцінці ефективності Інформаційної системи аналізу характеристик харчових продуктів на основі комп'ютерного зору. У ході дослідження було досягнуто поставленої мети — створення гібридної системи для автоматичного розпізнавання та аналізу товарів, що використовує поєднання технологій глибокого навчання, оптичного розпізнавання тексту та зовнішніх відкритих баз даних.

1. Досягнення цілей та обґрунтування архітектури

Актуальність роботи підтверджена зростаючою потребою в автоматизації процесів ідентифікації товарів у роздрібній торгівлі, касах самообслуговування та логістичних системах, особливо в умовах, коли ручне сканування штрих-кодів ускладнене.

Для первинної класифікації була обрана архітектура MobileNetV2. Цей вибір обґрунтовано необхідністю забезпечити високу точність при значно меншій кількості параметрів порівняно з громіздкими моделями (наприклад, ResNet50 або VGG16). Це робить систему ідеальною для використання на мобільних пристроях і в умовах обмеженого апаратного забезпечення.

Було проведено адаптацію датасету Grocery Store Dataset з Kaggle, трансформувавши його ієрархічну структуру у плоску для навчання моделі класифікації fine-класів. Це дозволило моделі не лише визначати категорію, а й розпізнавати конкретну назву продукту та бренд.

2. Реалізація гібридного підходу та його переваги

Основна наукова новизна та практична цінність роботи полягає в розробці гібридної архітектури, яка компенсує типові обмеження візуальної класифікації. Виявлено, що класифікація лише за зображенням часто дає помилки через візуальну подібність упаковок (наприклад, молоко, кефір, йогурт). Для подолання цього, було інтегровано хмарний OCR-сервіс DocuPipe API. DocuPipe дозволив зчитувати текстові елементи (бренд, підкатегорія) з етикеток і уточнювати результат, якщо впевненість MobileNetV2 була низькою (нижче 80%). Вибір

DocuPipe обґрунтовано його хмарною архітектурою, що знижує навантаження на клієнтський пристрій, а також підтримкою складних макетів, таблиць (наприклад, харчова цінність) та багатомовності. Збагачення даних (OpenFoodFacts). Система була інтегрована з відкритою базою даних OpenFoodFacts. Це забезпечило автоматичне отримання детальної метаінформації про продукт (калорійність, вміст жирів, цукру, штрихкод), що значно підвищило функціональність та інформативність кінцевого результату.

3. Результати тестування та оцінка ефективності

Реалізована система була протестована в умовах, наближених до реальних, з використанням веб-додатку на базі Flask.

Загальна точність системи при виключно класифікації зображень становила близько 60%. Після активації OCR-модуля та застосування гібридної логіки точність у тестових сценаріях зросла до понад 80%. Це підтверджує, що поєднання візуального та текстового аналізу є ключовим для ідентифікації товарів із високою схожістю упаковки. Середній час обробки одного запиту, що включає класифікацію, OCR та пошук у зовнішніх базах, склав 5–8 секунд, що є прийнятним для інтерактивного використання. Приклади тестування (наприклад, розпізнавання вівсяного молока Oatly) продемонстрували, що навіть у випадку неідеальної класифікації MobileNetV2, OCR-модуль успішно виявляє специфікації ("Barista Edition"), дозволяючи OpenFoodFacts видати точну харчову інформацію.

4. Перспективи подальшого розвитку

Незважаючи на успішну реалізацію, тестування виявило напрямки для подальшого вдосконалення системи, що також складає частину наукового доробку роботи. Необхідне дотренування класифікаційної моделі на більшому та різноманітнішому наборі даних, що охоплює більше варіацій упаковок і локальних брендів. Потребує вдосконалення логіка зіставлення текстових даних, отриманих від OCR, із записами у базі OpenFoodFacts, оскільки неточність могла виникати під час пошуку відповідного запису.

Перспективним є розширення функціоналу, включаючи використання штрихкодів для додаткової перевірки, реалізацію голосових підказок (наприклад, для людей із вадами зору) та інтеграцію з іншими базами даних.

У підсумку, створена інформаційна система є функціональним, гнучким та масштабованим рішенням. Вона успішно демонструє практичне значення комбінованого підходу, інтегруючи візуальне та текстове розпізнавання з відкритими джерелами даних, тим самим закладаючи основу для її впровадження у торговельні, логістичні та побутові інтелектуальні системи.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Papers With Code: Open Source Machine Learning Papers [Електронний ресурс]. – Режим доступу: https://paperswithcode.com – Назва з екрана. – (дата звернення: 09.11.2025). – [англ.].
2. PyTorch Documentation [Електронний ресурс]. – Режим доступу: https://pytorch.org – Назва з екрана. – (дата звернення: 09.11.2025). – [англ.].
3. ValidModel. Grocery Store Dataset [Електронний ресурс] / Kaggle. – Режим доступу: https://www.kaggle.com/datasets/validmodel/grocery-store-dataset – Назва з екрана. – (дата звернення: 09.11.2025). – [англ.].
4. MobileNet version 2 [Електронний ресурс] // Machinethink. – Режим доступу: https://machinethink.net/blog/mobilenet-v2/ – Назва з екрана. – (дата звернення: 09.11.2025). – [англ.].
5. DocuPipe Reviews 2025: Details, Pricing, & Features [Електронний ресурс] // G2. – Режим доступу: https://www.g2.com/products/docupipe/reviews – Назва з екрана. – (дата звернення: 09.11.2025). – [англ.].
6. DocuPipe | AI Document Extraction [Електронний ресурс] // Docupipe.ai. – Режим доступу: https://www.docupipe.ai/ – Назва з екрана. – (дата звернення: 09.11.2025). – [англ.].
7. OCR comparison: Tesseract vs EasyOCR vs PaddleOCR vs MMOCR [Електронний ресурс] // Medium. – Режим доступу: https://toon-beerten.medium.com/ocr-comparison-tesseract-versus-easyocr-vs-paddleocr-vs-mmocr-a362d9c79e66 – Назва з екрана. – (дата звернення: 09.11.2025). – [англ.].
8. DocuPipe | F6S [Електронний ресурс]. – Режим доступу: https://www.f6s.com/company/docupipe – Назва з екрана. – (дата звернення: 09.11.2025). – [англ.].
9. DocuPipe vs OCR Gateway Comparison [Електронний ресурс] // SourceForge. – Режим доступу: https://sourceforge.net/software/compare/DocuPipe-vs-OCR-Gateway/ – Назва з екрана. – (дата звернення: 10.11.2025). – [англ.].

10. OpenFoodFacts [Електронний ресурс] – Режим доступу: https://world.openfoodfacts.org – Назва з екрана. – (дата звернення: 11.11.2025). – [англ.]
11. Wei Y., Tran S. N., Xu S., Kang B., Springer M. Deep Learning for Retail Product Recognition: Challenges and Techniques [Електронний ресурс] / Y. Wei, S. N. Tran, S. Xu, B. Kang, M. Springer // ResearchGate. – 2020. – Режим доступу: https://www.researchgate.net/publication/346856840 – Назва з екрана. – (дата звернення: 10.11.2025). – [англ.].

ДОДАТКИ

Додаток А. Код програми

А.1. Код main.py

```
from flask import Flask, render_template, request, jsonify, redirect, url_for
from werkzeug.utils import secure_filename
import os
from datetime import datetime
from ai import predict_single_image
from product_info import fetch_product_info, label_to_query

UPLOAD_DIR = "uploads"
MODEL_PATH = "mobilenetv2_grocery_fine.pth"
LABELS_PATH = "fine_class_names.txt"
ALLOWED_EXT = {"jpg", "jpeg", "png", "webp"}

os.makedirs(UPLOAD_DIR, exist_ok=True)

app = Flask(__name__)

def _allowed(filename: str) -> bool:
    return "." in filename and filename.rsplit(".", 1)[1].lower() in
ALLOWED_EXT
```

```
@app.route("/", methods=["GET"])
def index():
    return render_template("index.html",
ts=datetime.now().strftime("%Y-%m-%d %H:%M"))



@app.route("/predict", methods=["POST"])
def predict():
    if "file" not in request.files:
        return redirect(url_for("index"))
    f = request.files["file"]
    if not f or f.filename == "":
        return redirect(url_for("index"))
    if not _allowed(f.filename):
        return redirect(url_for("index"))

    filename = secure_filename(f.filename)
    save_path = os.path.join(UPLOAD_DIR,
f"{datetime.now().strftime('%Y%m%d_%H%M%S')}_{filename}")
    f.save(save_path)
```

```
        try:
            result = predict_single_image(save_path, MODEL_PATH,
LABELS_PATH)
        except Exception as e:
            result = {"label": f"Помилка: {e}", "confidence": 0.0, "index": -1}



        q = label_to_query(result.get("label", ""))
        nutrition = fetch_product_info(q) if q else None



        if nutrition and isinstance(nutrition, dict):
            for k in [
                "energy_kcal_100g", "proteins_100g", "fat_100g", "saturated_fat_100g",
                "carbohydrates_100g", "sugars_100g", "fiber_100g", "salt_100g"
            ]:
                nutrition.setdefault(k, None)



        return render_template(
```

```
            "index.html",
            result=result,
            nutrition=nutrition,
            ts=datetime.now().strftime("%Y-%m-%d %H:%M")
        )


    @app.route("/api/predict", methods=["POST"])
    def api_predict():
        if "file" not in request.files:
            return jsonify({"error": "file missing"}), 400
        f = request.files["file"]
        if not f or f.filename == "" or not _allowed(f.filename):
            return jsonify({"error": "bad file"}), 400


        filename = secure_filename(f.filename)
        save_path = os.path.join(UPLOAD_DIR,
f"{datetime.now().strftime('%Y%m%d_%H%M%S')}_{filename}")
        f.save(save_path)
        result = predict_single_image(save_path, MODEL_PATH, LABELS_PATH)
        q = label_to_query(result.get("label", ""))
        nutrition = fetch_product_info(q) if q else None
```

```
        print(nutrition)
        return jsonify({"classification": result, "nutrition": nutrition})


if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000, debug=True)
```

A.2. Код навчання моделі MobileNetV2

```
import os
import shutil
import time
import zipfile
import torch
import torchvision.transforms as transforms
from torchvision import models
from torch import nn, optim
from torch.utils.data import DataLoader
from tqdm import tqdm
from PIL import Image
from torchvision.datasets import ImageFolder
```

```
def flatten_dataset_structure(src_dir, dst_dir):
    if not os.path.exists(dst_dir):
        os.makedirs(dst_dir)

    for root, _, files in os.walk(src_dir):
        for file in files:
            if file.lower().endswith(('.jpg', '.png', '.jpeg')):
                relative_path = os.path.relpath(root, src_dir)
                fine_class = relative_path.replace(os.sep, '_')
                class_dir = os.path.join(dst_dir, fine_class)
                os.makedirs(class_dir, exist_ok=True)
                src_path = os.path.join(root, file)
                dst_path = os.path.join(class_dir, file)
                shutil.copy2(src_path, dst_path)


def extract_local_zip():
    zip_path = '../pic/GroceryStoreDataset.zip'
    extract_path = '../data/GroceryStoreDataset/dataset'
    if os.path.exists(zip_path) and not os.path.exists(extract_path):
        print("Розпакування локального архіву GroceryStoreDataset.zip...")
```

```
        with zipfile.ZipFile(zip_path, 'r') as zip_ref:
            zip_ref.extractall('./data/')
        print("Розпакування завершено.")
    elif os.path.exists(extract_path):
        print("Датасет уже розпаковано.")
    else:
        print("Архів GroceryStoreDataset.zip не знайдено.")



def prepare_data():
    extract_local_zip()
    print("Переорганізація структури датасету до плоскої...")
    flatten_dataset_structure(
        '../data/GroceryStoreDataset/dataset/train',
        './data/GroceryStoreDataset/flat_train'
    )
    flatten_dataset_structure(
        '../data/GroceryStoreDataset/dataset/val',
        './data/GroceryStoreDataset/flat_val'
    )
```

```
print("Ініціалізація трансформацій...")
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor()
])

train_dataset = ImageFolder(
    root='./data/GroceryStoreDataset/flat_train',
    transform=transform
)
test_dataset = ImageFolder(
    root='./data/GroceryStoreDataset/flat_val',
    transform=transform
)

with open('fine_class_names.txt', 'w', encoding='utf-8') as f:
    for class_name in train_dataset.classes:
        f.write(f"{class_name}\n")

print("Кількість fine-класів у train:", len(train_dataset.classes))
print("Кількість навчальних зображень:", len(train_dataset))
```

```
    train_loader = DataLoader(train_dataset, batch_size=32, shuffle=True,
num_workers=0)
    test_loader = DataLoader(test_dataset, batch_size=32, shuffle=False,
num_workers=0)

    return train_loader, test_loader, len(train_dataset.classes)


def initialize_model(num_classes):
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    model =
models.mobilenet_v2(weights=models.MobileNet_V2_Weights.IMAGENET1K_V1)
    model.classifier[1] = nn.Linear(model.last_channel, num_classes)
    model = model.to(device)
    return model, device


def train_model(model, device, train_loader):
    criterion = nn.CrossEntropyLoss()
    optimizer = optim.Adam(model.parameters(), lr=0.001)
    model.train()
```

```
    start = time.time()

    for images, labels in tqdm(train_loader, desc="Навчання", unit="batch"):
        images, labels = images.to(device), labels.to(device)
        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

    end = time.time()
    print(f"\nНавчання завершено за {round(end - start, 2)} секунд.")
    torch.save(model.state_dict(), 'mobilenetv2_grocery_fine.pth')


def evaluate_model(model, device, test_loader):
    model.eval()
    correct, total = 0, 0
    with torch.no_grad():
        for images, labels in tqdm(test_loader, desc="Тестування", unit="batch"):
            images, labels = images.to(device), labels.to(device)
```

```
            outputs = model(images)
            _, predicted = torch.max(outputs, 1)
            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    accuracy = 100 * correct / total
    print(f"Точність на тестовому наборі: {accuracy:.2f}%")


if __name__ == '__main__':
    train_loader, test_loader, num_classes = prepare_data()
    model, device = initialize_model(num_classes)
    train_model(model, device, train_loader)
    evaluate_model(model, device, test_loader)
```

А.3. Код розпізнавання зображення ai.py

```
import torch
from torchvision import models, transforms
from torch import nn
from torch.nn import functional as F
from PIL import Image
```

```
import os



from ocr_helper_docupipe import run_docupipe_ocr


import re
def _clean_tokens(s: str):
    s = re.sub(r"[^a-zA-Z0-9\s\-_/\.]", " ", s).lower()
    return [t for t in re.split(r"[\s\-_/\.]+", s) if len(t) > 1]
def text_match_scores(labels, ocr_text, n=3, alpha_token=0.7, alpha_char=0.3):
    if not ocr_text:
        return [0.0]*len(labels)
    def ngrams(s, n):
        s = re.sub(r"[^a-z0-9]", "", s.lower())
        return {s[i:i+n] for i in range(0, max(0, len(s)-n+1))}
    ocr_tokens = set(_clean_tokens(ocr_text))
    ocr_ng = ngrams(ocr_text, n)
    ocr_join_ng = ngrams(re.sub(r"\s+", "", ocr_text.lower()), n)
    out = []
    for lab in labels:
        lt = set(_clean_tokens(lab))
```

```
        lng = ngrams(lab, n)
        tok = len(lt & ocr_tokens) / max(1, len(lt))
        ch1 = len(lng & ocr_ng) / max(1, len(lng | ocr_ng))
        ch2 = len(lng & ocr_join_ng) / max(1, len(lng | ocr_join_ng))
        out.append(alpha_token*tok + alpha_char*max(ch1, ch2))
    return out

CONF_THRESHOLD = 0.80
OCR_WEIGHT = 0.35

def predict_single_image(image_path, model_path, label_map_path):
    print(f"Передбачення для зображення: {image_path}")

    transform = transforms.Compose([
        transforms.Resize((224, 224)),
        transforms.ToTensor()
    ])

    if not os.path.exists(image_path):
        raise FileNotFoundError(f"Файл зображення {image_path} не знайдено")
```

```
image = Image.open(image_path).convert("RGB")
image_tensor = transform(image).unsqueeze(0)

with open(label_map_path, 'r', encoding='utf-8') as f:
    labels = [line.strip() for line in f]

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = models.mobilenet_v2()
model.classifier[1] = nn.Linear(model.last_channel, len(labels))

try:
    state = torch.load(model_path, map_location=device)
    model.load_state_dict(state)
except RuntimeError as e:
    print("Помилка при завантаженні state_dict:", e)
    print("Перевір відповідність кількості класів і списку міток.")
    return

model = model.to(device)
model.eval()
```

```
with torch.no_grad():
    logits = model(image_tensor.to(device))
    probs = F.softmax(logits, dim=1).squeeze(0).cpu()
    conf, idx = torch.max(probs, dim=0)
    conf = float(conf)
    idx = int(idx)
    predicted_label = labels[idx]


    print(f"Передбачений підклас: {predicted_label}")
    print(f"Впевненість: {conf:.3f}")



    if conf < CONF_THRESHOLD:
        print("Низька впевненість. Вмикаю OCR-підсилення (DocuPipe).")
        api_key = "API_KEY"
        if not api_key:
            print("Не знайдено DOCUPIPE_API_KEY у змінних середовища.")
            ocr_text = ""
        else:
            try:
                ocr_text = run_docupipe_ocr(image_path, api_key)
```

```
    except Exception as e:
        print(f"DocuPipe OCR помилка: {e}")
        ocr_text = ""

if ocr_text:
    print(f"OCR текст (DocuPipe): {ocr_text}")
    scores = text_match_scores(labels, ocr_text)
    s = torch.tensor(scores, dtype=torch.float32)
    if s.max().item() > 0:
        s = s / s.max().item()
        mix = probs + OCR_WEIGHT * s
        mix = torch.clamp(mix, min=1e-8)
        mix = mix / mix.sum()
        new_conf, new_idx = torch.max(mix, dim=0)
        predicted_label = labels[int(new_idx)]
        conf = float(new_conf+conf)
        idx = int(new_idx)
        print(f"Після OCR: {predicted_label} | впевненість: {conf:.3f}")
    else:
        print("OCR не дав корисних збігів з назвами класів.")
```

```
        return {"label": predicted_label, "confidence": conf, "index": idx}


if __name__ == "__main__":
    result = predict_single_image(
        '../pic/jog.jpeg',
        'mobilenetv2_grocery_fine.pth',
        'fine_class_names.txt'
    )
    if result:
        print(f"Підсумок: {result['label']} ({result['confidence']:.3f})")
```

А.4. Код OCR модулю

```
import base64, time, requests
from typing import Tuple, Optional


BASE = "https://app.docupipe.ai"
TIMEOUT = 60          # сек, максимум очікування
BACKOFF_START = 2     # сек, інтервал полінгу


class DocuPipeOCR:
    def __init__(self, api_key: str, dataset: Optional[str] = None):
```

```
        self.api_key = api_key
        self.dataset = dataset
        self.hjson = {
            "accept": "application/json",
            "content-type": "application/json",
            "X-API-Key": api_key,
        }


    def _upload(self, path: str) -> Tuple[str, str]:
        payload = {
            "document": {"file": {
                "contents": base64.b64encode(open(path, "rb").read()).decode(),
                "filename": path.split("/")[-1]
            }}
        }
        if self.dataset:
            payload["dataset"] = self.dataset
        r = requests.post(f"{BASE}/document", json=payload, headers=self.hjson,
timeout=120)
        r.raise_for_status()
        j = r.json()
```

```
        return j["documentId"], j["jobId"]


    def _wait_job(self, job_id: str) -> bool:
        waited, backoff = 0, BACKOFF_START
        while True:
            r = requests.get(f"{BASE}/job/{job_id}", headers=self.hjson, timeout=30)
            r.raise_for_status()
            st = r.json().get("status")
            if st == "completed":
                return True
            if st == "error" or waited >= TIMEOUT:
                return False
            time.sleep(backoff)
            waited += backoff
            backoff = min(backoff * 2, 10)


    def _get_text(self, document_id: str) -> str:
        r = requests.get(f"{BASE}/document/{document_id}", headers=self.hjson,
timeout=60)
        r.raise_for_status()
        j = r.json()
```

```
        return ((j.get("result") or {}).get("text") or "").strip()

    def run(self, image_path: str) -> str:
        doc_id, job_id = self._upload(image_path)
        if not self._wait_job(job_id):
            return ""
        return self._get_text(doc_id)

def run_docupipe_ocr(image_path: str, api_key: str, dataset: str | None = None) -> str:
    return DocuPipeOCR(api_key, dataset).run(image_path)
```

А.5. Код пошуку інформації про продукти на openfoodfacts

```
import requests, re

OFF_SEARCH = "https://world.openfoodfacts.org/cgi/search.pl"

def label_to_query(label: str) -> str:
    if not label:
        return ""

    s = label.replace("_", " ").replace("-", " ")
    s = re.sub(
```

```
r"\b(packages?|package|iconic|bottle|box|can|medium|fat|low|high|extra|normal|product|brand|type|mild)\b",
        " ",
        s,
        flags=re.I,
    )
    s = re.sub(r"\s+", " ", s).strip()

    FLAVORS = [
        "vanilla", "strawberry", "chocolate", "banana", "raspberry",
        "blueberry", "mango", "peach", "caramel", "natural", "plain",
        "apple", "coconut", "coffee", "honey", "cherry"
    ]
    PRODUCTS = [
        "milk", "yoghurt", "yogurt", "kefir", "cream", "butter", "cheese",
        "dessert", "drink", "juice", "smoothie"
    ]

    words = s.split()
    lower = [w.lower() for w in words]
```

```
    product = next((w for w in lower if w in PRODUCTS), "")
    flavor = next((w for w in lower if w in FLAVORS), "")

    brand = next(
        (w.capitalize() for w in words if w.lower() not in PRODUCTS + FLAVORS and
len(w) > 2),
        ""
    )

    parts = [brand, product, flavor]
    seen = set()
    clean = [p for p in parts if p and not (p in seen or seen.add(p))]
    return " ".join(clean).strip()

def fetch_product_info(query: str) -> dict:
    """
    Пошук продукту у базі (наприклад, OpenFoodFacts API).
    Повертає dict або {'error': 'Немає у базі даних'}.
    """
    def search(q):
        r = requests.get(
```

```
        "https://world.openfoodfacts.org/cgi/search.pl",
        params={"search_terms": q, "search_simple": 1, "action": "process", "json": 1, "page_size": 1},
        timeout=10,
    )
    if r.status_code == 200:
        data = r.json()
        products = data.get("products", [])
        if products:
            p = products[0]
            return {
                "name": p.get("product_name", ""),
                "brand": p.get("brands", ""),
                "energy": p.get("nutriments", {}).get("energy-kcal_100g", ""),
                "fat": p.get("nutriments", {}).get("fat_100g", ""),
                "sugars": p.get("nutriments", {}).get("sugars_100g", ""),
                "image": p.get("image_front_small_url", ""),
                "url": p.get("url", "")
            }
    return None
```

```
# ітерації запиту
parts = query.split()
attempts = []

if len(parts) >= 2:
    attempts.append(" ".join(parts))
if len(parts) >= 3:
    attempts.append(" ".join([parts[0], parts[-1]]))
attempts.append(parts[0])

for q in attempts:
    result = search(q)
    print(result)
    if result:
        print(f"[INFO] Знайдено за запитом: '{q}'")
        return result

print("[INFO] Немає у базі даних.")
return {"error": "Немає у базі даних"}
```

Додаток Б. Приклад зображень для навчання та тестування

Рисунок Б.1 – Приклад фотографії яблука з навчального датасету

Рисунок Б.2 – Приклад фотографії молока з навчального датасету

Рисунок Б.3 – Приклад фотографії огірків з навчального датасету