

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

СИТНИК Назар Дмитрович

Програмний модуль визначення характеристик продуктів з допомогою аналізу
зображень / Software module for determining product characteristics using image
analysis

спеціальність: 123 – Комп'ютерна інженерія
освітньо–професійна програма – Комп'ютерна інженерія

Кваліфікаційна робота

Виконав: студент групи КІ–41
СИТНИК Назар Дмитрович

Науковий керівник
к.т.н. доц. Піцун О.Й.

ТЕРНОПІЛЬ – 2026

АНОТАЦІЯ

Ситник Н.Д. Мобільний додаток для автоматичного визначення калорійності продуктів харчування за фотографією. – Рукопис.

Кваліфікаційна робота на здобуття освітнього ступеня «бакалавр» за спеціальністю 123 «Комп'ютерна інженерія», освітньо-професійна програма «Комп'ютерна інженерія». Тернопільський національний економічний університет, Тернопіль, 2025. Обсяг роботи: 62 сторінки, 14 рисунків, 5 таблиць, 4 додатки, 30 джерел.

Метою роботи є проектування та реалізація мобільного комплексу для автоматичного визначення калорійності та КБЖВ продуктів за фотографією за допомогою методів глибокого навчання та комп'ютерного зору.

Розроблено клієнт-серверну систему, що включає серверну частину (Python/FastAPI, PostgreSQL) та мобільний клієнт (React Native). Для розпізнавання їжі навчено модель EfficientNet-B0 методом Transfer Learning на датасеті Food-101 (101 клас). Створено REST API для JWT-аутентифікації, розпізнавання зображень (із попередньою обробкою засобами OpenCV), пошуку продуктів та ведення щоденника харчування. Реалізовано автоматичний розрахунок денної норми КБЖВ за формулою Маффіна-Джеора на основі антропометричних даних користувача. Серверна частина розгорнута за допомогою Docker.

Навчена модель EfficientNet-B0 досягла точності класифікації на тестовій вибірці близько 73% (Top-1) та 92% (Top-5). Мобільний клієнт містить п'ять основних екранів: головний екран із прогресом норми калорій, камеру для сканування їжі, результати розпізнавання з КБЖВ, щоденник харчування та профіль.

Ключові слова: КБЖВ, РОЗПІЗНАВАННЯ ЇЖІ, МОБІЛЬНИЙ ДОДАТОК, ГЛИБОКЕ НАВЧАННЯ, EFFICIENTNET, TRANSFER LEARNING, FOOD-101, FASTAPI, REACT NATIVE, POSTGRESQL, OPENCV, DOCKER.

ANNOTATION

Sytnyk N.D. Mobile application for automatic food calorie recognition from photographs. – Manuscript.

Bachelor's thesis for the degree of "Bachelor" in specialty 123 "Computer Engineering", educational and professional programme "Computer Engineering". Ternopil National Economic University, Ternopil, 2025. Thesis scope: 69 pages, 10 figures, 5 tables, 4 appendices, 30 references.

The objective of this thesis is the design and implementation of a mobile application for automatic recognition of food caloric content and nutritional values (macros) from photographs using deep learning and computer vision.

A client-server architecture was developed comprising a Python/FastAPI backend with PostgreSQL and a mobile client built on React Native. For food recognition, an EfficientNet-B0 model was trained using Transfer Learning on the Food-101 dataset. A REST API was implemented for JWT authentication, image recognition (with OpenCV preprocessing), product search, and nutrition diary management. Automatic calculation of daily nutritional norms using the Mifflin-St Jeor formula based on user anthropometric data is implemented. The backend is containerised using Docker.

The trained EfficientNet-B0 model achieved a Top-1 classification accuracy of ~73% and Top-5 accuracy of ~92% on the test set. The mobile app features five main screens: a home screen with daily calorie progress, a camera for food scanning, recognition results, a nutrition diary, and a user profile.

Keywords: NUTRITIONAL VALUES, FOOD RECOGNITION, MOBILE APPLICATION, DEEP LEARNING, EFFICIENTNET, TRANSFER LEARNING, FOOD-101, FASTAPI, REACT NATIVE, POSTGRESQL, OPENCV, DOCKER.

ЗМІСТ

Перелік умовних скорочень	9
Вступ.....	10
1. Аналіз предметної області та технології	12
1.1 Аналіз існуючих систем контролю харчування.....	12
1.2 Аналіз засобів обробки зображень.....	16
1.3 Первинний аналіз засобів обчислення калорій.....	20
2. Алгоритм роботи системи обчислення калорій.....	25
2.1 Узагальнений алгоритм роботи системи обчислення	25
2.2 Архітектура нейронної мережі	29
2.3 Алгоритм попередньої обробки зображень	33
3 Реалізація програмного комплексу	38
3.1 Загальна архітектура програмного комплексу.....	38
3.2 Реалізація серверної частини	39
3.3 Реалізація модуля машинного навчання.....	43
3.4 Реалізація мобільного клієнта.....	45
3.5 Наповнення бази даних та розгортання.....	50
Висновки.....	52
Список використаних джерел	54
Додаток А Світлокопії публікацій.....	57
Додаток Б Техніко–економічне обґрунтування	59
Додаток В Лістинг коду	68

					КР.КІ.10659264/22.00.00.000 ПЗ			
Змн.	Лист	№ докум.	Підпис	Дата				
Розробив	Ситник Н.Д.				ПРОГРАМНИЙ МОДУЛЬ ВИЗНАЧЕННЯ ХАРАКТЕРИСТИК ПРОДУКТІВ З ДОПОМОГОЮ АНАЛІЗУ ЗОБРАЖЕНЬ	Літ.	Арк.	Акрушів
Перевір.	Піцун О.Й.						8	85
Консульт.	Савка Н.Я.					ЗУНУ,ФКІТ, КІ-41		
Н. Контр.	Дубчак ЛО.							
Затвердив	Дубчак ЛО.							

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API	–	Application Programming Interface
ASGI	–	Asynchronous Server Gateway Interface
BMR	–	Basal Metabolic Rate
CNN	–	Convolutional Neural Network
CORS	–	Cross-Origin Resource Sharing
CPU	–	Central Processing Unit
GPU	–	Graphics Processing Unit
HTTP	–	Hypertext Transfer Protocol
JSON	–	JavaScript Object Notation
JWT	–	JSON Web Token
КБЖВ	–	Калорії, Білки, Жири, Вуглеводи
ML	–	Machine Learning
ORM	–	Object-Relational Mapping
REST	–	Representational State Transfer
SDK	–	Software Development Kit
SQL	–	Structured Query Language
TDEE	–	Total Daily Energy Expenditure
TFLite	–	TensorFlow Lite
UI	–	User Interface
URL	–	Uniform Resource Locator
БД	–	База даних
ВКР	–	Випускна кваліфікаційна робота
ПЗ	–	Програмне забезпечення

ВСТУП

У сучасному світі умови праці також і темп життя приводять нас до нехтування правилами щодо здорового харчування що, в свою чергу, погано впливає на наш з вами загальний стан здоров'я. Хочеться згадати що ми живемо у високорозвинену епоху інформаційних технологій, у сучасному світі дуже швидко розвиваються технології та пристрої для їх використання, тому наявність зручного інструменту який допоможе контролювати та коригувати раціон є потрібним для покращення нашого життя та запобігати захворюванням.

В Україні згідно з даних Міністерства охорони здоров'я, близько 60% дорослих мають ознаки ожиріння і в 30% з них воно є вираженим [27]. Це призводить до таких хвороб як діабет, рак, серцево - судинні хвороби та інші.

Основним джерелом для контролю харчування, доступним для усіх користувачів на теперішній час є мобільні додатки які допомагають проводити аналіз та коригувати його за потреби, оскільки зараз не усі мають змогу вести паперові щоденники або постійно вручну шукати потрібні продукти та їх енергетичну цінність.

Тому створення додатку який зможе автоматизовано розрахувати калорії шляхом розпізнавання фото є актуальною задачею, тому що це дозволяє значно спростити процес відстеження харчування що, в свою чергу, дозволяє мінімізувати час потрібний для цього та мінімізувати помилки користувачів.

Основною метою даної бакалаврської роботи розробка програмного забезпечення для автоматичного визначення калорійності продуктів харчування на основі зображень із використанням сучасних алгоритмів машинного навчання та комп'ютерного зору.

Актуальність роботи полягає у реалізації якісного та швидкого процесу контролю за харчуванням завдяки використанню сучасних алгоритмів машинного навчання та комп'ютерного зору.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		10

Об'єктом дослідження є процес автоматичного визначення калорійності та поживної цінності продуктів харчування за фотографією з використанням методів глибокого навчання.

Предметом дослідження є методи та засоби розпізнавання зображень їжі на основі згорткових нейронних мереж, архітектура програмного комплексу клієнт-сервер та алгоритми обробки зображень.

Основними завданнями дослідження є: аналіз існуючих систем контролю харчування та методів розпізнавання їжі за фотографією; вибір та обґрунтування технологічного стеку для реалізації системи; навчання моделі EfficientNet-B0 на датасеті Food-101 методом Transfer Learning; розробка серверної частини на FastAPI з REST API та базою даних PostgreSQL; реалізація мобільного клієнта на React Native з функціями фотографування та ведення щоденника харчування; розгортання системи у Docker-контейнерах та тестування функціональності.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ТЕХНОЛОГІЇ

1.1 Аналіз існуючих систем контролю харчування

Здорове харчування це одне із найголовніших частин у нашому житті для підтримки не тільки фізично а також психологічного здоров'я. Саме харчування впливає на рівень нашої енергії що, в свою чергу, впливає на працездатність, також імунітет та і загалом на нашу якість життя.

Поняття мобільного здоров'я або mHealth - це використання мобільних пристроїв та бездротових технологій для підтримки досягнення цілей у галузі охорони здоров'я. Інформацію, яку обробляють такими додатками, розташовується на серверах, де в свою чергу встановлено спеціальне програмне забезпечення за допомогою якого користувач отримує необхідну інформацію про продукти харчування практично миттєво.

Значна частина таких варіантів виконана у варіанті із мобільним додатком-щоденником. У таких додатках користувачеві дозволяється вести моніторинг спожитої їжі, проте більшість з них вимагають ручного введення даних.

Перевагою використання автоматизованої системи для розпізнавання (фото та штрих-коди) у порівнянні із ручним введенням є швидкість та зручність. Користувачу не обов'язково шукати назву продукту в базі вручну - достатньо навести камеру.

Додатків у сфері Health & Fitness з кожним днем стає все більшим. Такі додатки розробляють не лише незалежні розробники, а також і великі компанії та корпорації (наприклад, MyFitnessPal чи Samsung Health).

Для зручності аналізу, існуючі рішення на ринку можна класифікувати за методом введення даних:

а) ручні (Manual entry) - найпростіші та найпоширеніші. У таких додатках користувачеві потрібно самому вводити назву продукту його масу. Цей спосіб найбільш універсальним хоча і займає певну кількість часу та існує ймовірність помилок;

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		12

б) напівавтоматичні (Barcode scanners) - цей спосіб дозволяє автоматизувати введення даних для готової упакованої продукції за допомогою зчитування даних через штрих-код, після сканування якого система отримує інформацію з бази даних. Це вже спрощує ведення обліку спожитої продукції за день, однак має обмеження у визначенні продуктів а саме здатний розпізнавати лише фасовану їжу, до прикладу домашня їжа не може бути визначена оскільки на них немає штрих-коду;

с) інтелектуальні (AI-based) - зараз найбільш перспективний напрям розвитку систем для контролю харчування. Він використовує комп'ютерний зір і машинне навчання для визначення типу продукту за фотографією. за допомогою такого підходу можна мінімізувати участь користувача в процесі пошуку та введення необхідних даних що в свою чергу зробить взаємодію із додатком більш зручною. Основною проблемою такого методу є складність її реалізації а саме забезпечення високої точності розпізнавання оскільки потрібні великі датасети для машинного навчання.

Ринок додатків для контролю харчування щодня збільшується, проте серед них найбільше виділяються MyFitnessPal, FatSecret та Yazio.

Одним із найбільш відомих додатків у сфері контролю харчування є додаток MyFitnessPal. Він надає можливість користувачу отримати доступ до великої бази даних продуктів а також дозволяє вести детальний щоденник харчування. Основний акцент у даному додатку зроблений на ручному введенні інформації, хоча також підтримується сканування штрих-кодів. Автоматичне розпізнавання їжі за фотографіями реалізоване обмежено та не є основною функцією системи. На рисунку 1.1 наведено приклад інтерфейсу MyFitnessPal.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		13

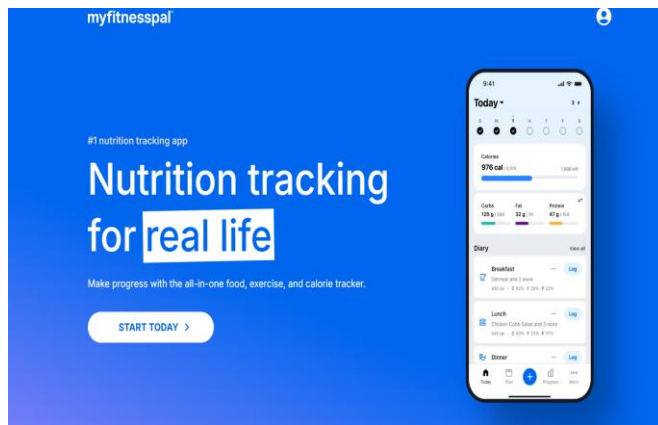


Рисунок 1.1 – MyFitnessPal

В свою чергу додаток FatSecret орієнтований на користувачів, які прагнуть безкоштовного доступу до базових функцій контролю харчування. Він надає можливість ручного введення даних і сканування штрих-кодів, але має менш розвинені аналітичні можливості порівняно з іншими аналогами. Також варто зазначити що інтерфейс додатку який показаний на рисунку 1.2 може бути менш інтуїтивним для нових користувачів.

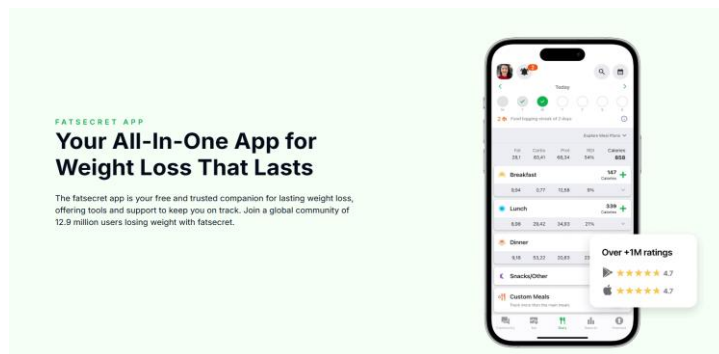


Рисунок 1.2 - FatSecret

Yazio також є популярним рішенням, у якому поєднано функції підрахунку калорій та планування раціону. Додаток пропонує персоналізовані рекомендації, проте значна частина функціоналу доступна лише у платній версії. Функції автоматичного розпізнавання страв реалізовані частково та не завжди забезпечують високу точність результатів. На рисунку 1.3 приклад інтерфейсу додатка Yazio.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		14

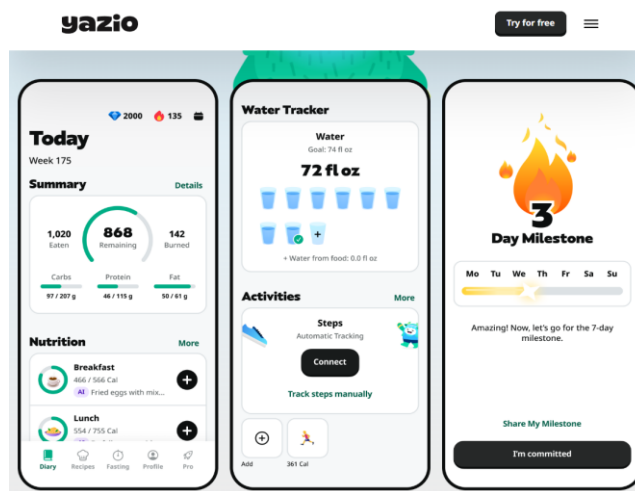


Рисунок 1.3 – Yazio

Для більш наочної характеристики розглянутих додатків наведено таблицю 1.1.

Таблиця 1.1 - Порівняння систем для ведення моніторингу харчування

Функції	MyFitnessPal	FatSecret	Yazio
Ручне введення даних	Так	Так	Так
Сканування штрих-кодів	Так	Так	Так
Розпізнавання за фото	обмежено	Ні	частково
Безкоштовний доступ	частково	Так	частково
Рівень автоматизації	Середній	Низький	Середній

Незважаючи на різноманіття систем для контролю харчування серед них можна виділити декілька спільних недоліків:

- залежність від введення інформації вручну;
- платний доступ до повного функціоналу програми;

- помилки в базі даних, відображається не той продукт що був потрібний користувачу.

Через ці обмеження та недоліки, процес користування додатком користувачем може негативно вплинути на оцінку системи самим користувачем.

Отже, у цьому підрозділі було проведено аналіз деяких існуючих систем для контролю харчування і в результаті встановлено, що більша частина рішень не забезпечує повної автоматизації процесу обліку спожитої їжі. Найперспективнішим напрямом розвитку є використання інтелектуальних алгоритмів розпізнавання за зображенням які значно спрощують взаємодію між системою та користувачем а також підвищує точність визначення КБЖВ продуктів.

1.2 Аналіз засобів обробки зображень

Для розробки та реалізації додатку автоматичного визначення КБЖВ (калорії, білки, жири, вуглеводи) продуктів за фотографією треба вибрати відповідні технології для цього. У цьому підрозділі було розглянуто технології для компонентів системи: клієнтська частина, база даних, алгоритми машинного навчання, комп'ютерного зору та серверної частини.

Мова програмування Python являється однією із найбільших мов програмування у напрямку машинного навчання та розробки серверних застосунків. Її широке використання у промисловій розробці та наукових дослідженнях обумовлено багатьма суттєвими перевагами.

В першу чергу, це простота та зрозумілість синтаксичної структури, яка в свою чергу дає можливість швидко реалізувати складний алгоритм без зайвого навантаження для синтаксичної частини коду. По-друге, бібліотеки Python мають одну з найбільших екосистем порівняно з іншими мовами програмування для роботи із: TensorFlow, PyTorch, Keras, scikit-learn, OpenCV та інші - усі мають

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		16

офіційну підтримку саме для python або були написанні для нього [20]. По-третє, цю мову програмування активно використовують у таких галузях як, обробка зображень та комп'ютерного зору, що в цьому випадку є ключовою функцією додатку. Велика кількість готових рішень, туторіалів та наукових публікацій у цій сфері робить Python гарним вибором для реалізації серверної логіки та моделей машинного навчання.

Для розробки REST API серверної частини використовується фреймворк FastAPI, він швидкий, сучасний та підтримує обробку запитів в асинхронному режимі [5]. Також FastAPI сам генерує документацію на основі OpenAPI-специфікації, яка значно спростить тестування та майбутню інтеграцію з мобільним клієнтом.

Головним елементом системи є модуль який відповідає за автоматичне розпізнавання продуктів харчування за фотографією. Щоб вирішити ці задачі з класифікації зображень використовують згорткові нейронні мережі (Convolutional Neural Networks, CNN) – це клас архітектури глибокого навчання який був розроблений спеціально для обробки зображень. CNN складається з декількох шарів а саме: convolutional layers (згорткових шарів), pooling layers (шарів підвибірки) та fully connected layers (повнозв'язних шарів). Згорткові шари виконують функцію згортки вхідного файлу із набором фільтрів, це дозволить системі в подальшому виявляти локальні ознаки - форму, край, текстуру. Далі шари підвибірки зменшують просторову розмірність даних та зберігають їх найбільш значущі ознаки. Повнозв'язні шари вкінці на виході мережі виконують класифікацію на основі набору виявлених раніше ознак. У цьому завданні можна застосувати метод перенесення навчання (Transfer Learning) - це підхід у якому попередньо навчена модель на великому датасеті (до прикладу, ImageNet) використовується як основа, а потім донавчається на спеціалізованому датасеті продуктів харчування [2]. За допомогою цього підходу можна досягти високої точності класифікації не дивлячись на обмежений обсяг навчальних даних а також скоротити час навчання моделі.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		17

Щоб реалізувати та начити модель яка буде класифікувати зображення, часто використовують два головних фреймворки глибокого навчання: TensorFlow від компанії Google та PyTorch від компанії Facebook (Meta).

TensorFlow - це open source бібліотека для числових обчислень та машинного навчання, вона підтримує побудову та навчання нейронних мереж різної складності. Разом з Keras - високорівневим API для TensorFlow - вона дозволяє будувати, навчати та розгортати моделі з мінімальним обсягом коду. Головною перевагою TensorFlow є підтримка формату TensorFlow Lite (TFLite), який дозволяє оптимізувати та розгортати навчені моделі безпосередньо на мобільних пристроях, що є важливим для даного проєкту.

PyTorch - це динамічний фреймворк для глибокого навчання, який завоював досить широку популярність у наукових дослідженнях саме завдяки своїй гнучкості та зручності налагодження. PyTorch дозволяє будувати обчислювальний граф у режимі реального часу (динамічний граф), це значно спрощує розробку і тестування нових архітектур. Для виробничого розгортання PyTorch моделей використовується формат ONNX (Open Neural Network Exchange), він забезпечує сумісність між різними фреймворками.

OpenCV (Open Source Computer Vision Library) - це відкрита бібліотека комп'ютерного зору та машинного навчання, вона містить в собі понад 2500 оптимізованих алгоритмів для роботи із зображеннями та відео. Ця бібліотека є кросплатформенною і підтримується у Python через інтерфейс opencv-python.

У контексті даного проєкту OpenCV використовується для попередньої обробки (preprocessing) зображень, отриманих з камери мобільного пристрою, перед їх подачею на вхід до нейронної мережі [10]. До основних операцій попередньої обробки входять: зміна розміру зображення до фіксованого розміру входу мережі 224×224 пікселів (стандарт EfficientNet-B0), нормалізація значень пікселів до діапазону [0, 1], корекція яскравості та контрастності, а також видалення шуму за допомогою фільтрів розмиття. Некоректна або відсутня нормалізація вхідних даних може призвести до значного зниження точності класифікації навіть при використанні якісно навченої нейронної мережі. На

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		18

рисунку 1.4 показано структуру бібліотеки OpenCV.

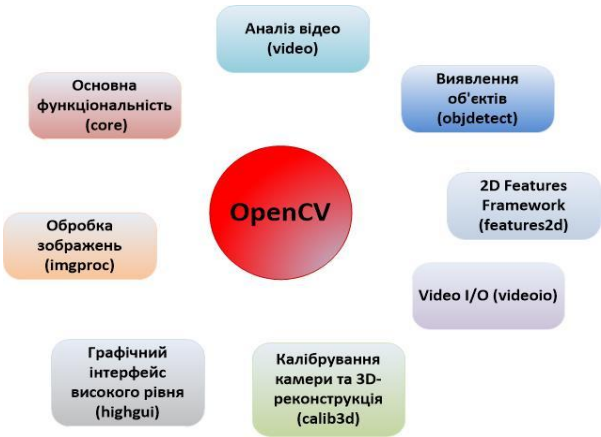


Рисунок 1.4 - Структура бібліотеки OpenCV

Для таких проектів де використовується камера для отримання вхідних даних якісна попередня обробка зображень є критично важливою для забезпечення стабільної та точної роботи моделі розпізнавання.

Щоб зберігати дані в системі буде використана така база даних як PostgreSQL. Це потужна відкрита реляційна СУБД, вона підтримує стандарт SQL і надає широкі можливості для зберігання структурованих даних [9]. У даній системі PostgreSQL використовується для зберігання бази даних продуктів харчування з їх КБЖВ-характеристиками, бази користувачів, а також журналу прийомів їжі. PostgreSQL забезпечує надійність, транзакційність та масштабованість зберігання даних. Взаємодія з базою даних здійснюється через ORM-бібліотеку SQLAlchemy, яка дозволяє описувати структуру таблиць у вигляді Python-класів та виконувати запити без написання сирого SQL. SQLAlchemy дозволяє описувати структуру таблиць у вигляді Python-класів і виконувати запити без написання сирого SQL. У системі реалізовано три основні таблиці: users (дані користувачів та їх цілі КБЖВ), foods (база продуктів харчування з характеристиками на 100г) та diary_entries (щоденник прийомів їжі). Аутентифікація реалізована через JWT-токени за допомогою бібліотеки python-jose з хешуванням паролів через bcrypt.

Якість навченої моделі для класифікації продуктів харчування безпосередньо залежить від обсягу та різноманітності навчального датасету. У сфері розпізнавання їжі існує декілька широко використовуваних відкритих датасетів.

Food-101 - це один із найбільш популярних датасетів у сфері розпізнавання їжі [3]. Він містить 101 000 зображень, розподілених рівномірно по 101 класу страв (по 1000 зображень на кожен клас). Датасет є стандартним бенчмарком для оцінки якості моделей розпізнавання їжі та широко використовується у наукових дослідженнях.

ETHZ Food-101 та UEC Food-256 - розширені датасети, які містять більшу кількість категорій (до 256 класів) та зображень, зокрема страви азійської та японської кухні. Їх використання дозволяє навчити модель розпізнавати ширший спектр продуктів та страв.

У даному проєкті використовується датасет Food-101, на якому навчено модель EfficientNet-B0 методом двофазного Transfer Learning: на першій фазі навчалась лише класифікаційна голова моделі при замороженій базі, на другій - виконувалось донавчання верхніх шарів (fine-tuning) з меншим learning rate.

Для розширення навчальної вибірки без необхідності збирання нових зображень застосовується аугментація даних (data augmentation) - техніка, що полягає у штучному збільшенні різноманітності навчальних зображень шляхом їх випадкового повороту, масштабування, відображення, зміни яскравості та контрасту. Це дозволяє підвищити узагальнювальну здатність моделі та знизити ризик перенавчання

1.3 Первинний аналіз засобів обчислення калорій

Визначення калорійності продуктів харчування є важливим інструментом контролю раціону людини. Енергетична цінність їжі визначає кількість енергії,

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		20

яку організм отримує після її споживання. Для підтримки здорового способу життя, контролю ваги та профілактики різних захворювань важливо правильно визначати кількість калорій, білків, жирів та вуглеводів у щоденному раціоні.

Разом із розвитком інформаційних технологій з’явилася велика кількість різних інструментів для підрахунку калорійності продуктів. Вони можуть бути реалізовані у вигляді мобільних додатків, веб-сервісів або спеціалізованого програмного забезпечення. Такі системи дозволяють людям швидко отримувати інформацію про поживну цінність продуктів та вести облік спожитої їжі. У сучасному світі існує декілька основних підходів до визначення калорійності продуктів харчування.

Першим і найпростішим методом є використання довідкових таблиць харчової цінності продуктів. У таких таблицях міститься інформація про кількість калорій, білків, жирів та вуглеводів на 100 грамів певного продукту. Користувач повинен самостійно визначити масу продукту та виконати відповідні математичні розрахунки для визначення правильної калорійності порції. Незважаючи на простоту реалізації, цей метод містить в собі ряд недоліків. Зокрема, процес розрахунку є досить трудомістким та займає велику кількість часу, а також існує висока ймовірність помилок під час введення даних або виконання обчислень. На рисунку 1.5 наведено приклад такої таблиці.

Крупи та каші (54)

Амарант Крупи та каші 371 ккал 14 білки 7 жири 65 вугл.	Борошно пшеничне біле Крупи та каші 364 ккал 10.3 білки 1 жири 76.3 вугл.	Борошно цілнозернове Крупи та каші 340 ккал 13.2 білки 2.5 жири 71.2 вугл.	Булгур Крупи та каші 342 ккал 12.3 білки 1.3 жири 75.9 вугл.
Вівсяна каша на воді Крупи та каші 68 ккал 2.4 білки 1.4 жири 12 вугл.	Вівсяна каша на молоці Крупи та каші 102 ккал 3.2 білки 4.1 жири 14 вугл.	Вівсянка Крупи та каші 342 ккал 12.3 білки 6.1 жири 59.5 вугл.	Вівсянка швидкого приготування Крупи та каші 367 ккал 12 білки 6 жири 68 вугл.

Рисунок 1.5 - Приклад довідкові таблиці калорійності продуктів

Ще одним підходом для вирахування калорійності є використання електронних калькуляторів калорій. Такі калькулятори зазвичай реалізуються у вигляді веб-додатків або мобільних програм. У такому випадку користувач вводить назву продукту та його масу, після чого система автоматично вираховує калорійність, кількість білку, вуглеводів та жирів на основі внутрішньої бази даних. Перевагою такого підходу є значне спрощення процесу розрахунку, оскільки система виконує всі обчислення самостійно. Проте користувачеві все одно потрібно вручну вводити назву продукту, що займає певний час. Далі на рисунку 1.6 показано калькулятор калорійності.

Мед

Солодощі

Натуральний мед з корисними властивостями.

Додати до порівняння

Додати до щоденника

Поживна цінність на 100 г

329
ккал

0.8
білки (г)

0
жирів (г)

81.5
вуглеводи (г)

0
клітковина (г)

Калькулятор калорій

Кількість:

100

Одиниця виміру:

грам

Швидкий вибір:

50 г

100 г

200 г

1 шт.

1 ст.л.

2 ст.л.

1 ч.л.

Поживна цінність для 100 г:

329
ккал

0.8
білки (г)

0.0
жирів (г)

81.5
вуглеводи (г)

0.0
клітковина (г)

Рисунок 1.6 - Калькулятор калорійності

Також популярним методом є використання технології сканування штрих-кодів. Більша частина сучасних мобільних додатків для контролю раціону підтримують функцію сканування штрих-кодів на упаковці продукту за допомогою камери. Після сканування система отримує інформацію про продукт із бази даних і автоматично визначає його харчову цінність. Цей метод дозволяє значно пришвидшити процес введення інформації та зменшити кількість помилок. Однак він має певні обмеження, оскільки працює лише для фасованих продуктів, що мають штрих-код.

Найбільш сучасним та перспективним методом є використання алгоритмів

комп'ютерного зору та машинного навчання для визначення продуктів за фотографією. У таких додатках користувач робить фотографію страви або продукту, після чого нейронна мережа аналізує зображення та визначає тип їжі. Після чого система отримує інформацію про харчову цінність із бази даних та виконує розрахунок калорійності. Основною перевагою такого підходу є високий рівень автоматизації та зручність для користувача, оскільки необхідність ручного введення інформації мінімізується максимально. Проте разом з тим, цей метод визначення калорійності має певні труднощі реалізації. Однією з основних проблем є забезпечення високої точності розпізнавання продуктів. Для цього необхідно використовувати великі набори навчальних даних та складні алгоритми машинного навчання. Крім того, досить складно визначити точну масу продукту тільки лише за фотографією, що може впливати на точність розрахунку калорійності.

Таким чином було проведено аналіз який показує, що традиційні методи обчислення калорійності продуктів мають ряд недоліків, пов'язаних із необхідністю ручного введення даних користувачем або обмеженим функціоналом. Найбільш перспективним напрямом розвитку є використання алгоритмів машинного навчання і комп'ютерного зору, які дозволяють автоматизувати процес визначення продуктів та їх калорійності на основі зображення. Використання такого підходу може значно підвищити зручність використання систем контролю харчування та зменшити кількість помилок під час введення даних.

В загальному у цьому розділі було проведено аналіз предметної області контролю харчування а також сучасних інформаційних технологій, які використовуються для реалізації систем підрахунку калорійності продуктів.

У підрозділі 1.1 було розглянуто існуючі системи контролю харчування та мобільні додатки, що використовуються для ведення щоденника харчування. Проведений аналіз показав, що більшість сучасних рішень, таких як MyFitnessPal, FatSecret та Yazio, надають користувачам можливість контролювати раціон, вести облік спожитих продуктів та отримувати

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		23

інформацію про їх харчову цінність. Проте значна частина таких систем все ще залежить від ручного введення даних або обмежена функціями сканування штрих-кодів, що знижує зручність використання та швидкість роботи.

У підрозділі 1.2 було проаналізовано основні технології, які можуть бути використані для реалізації системи автоматичного визначення калорійності продуктів. Було розглянуто використання мови програмування Python, фреймворку FastAPI для реалізації серверної частини, бази даних PostgreSQL для зберігання інформації, а також бібліотек машинного навчання TensorFlow та PyTorch. Особливу увагу приділено алгоритмам комп'ютерного зору та згортковим нейронним мережам, які дозволяють виконувати класифікацію зображень продуктів харчування.

Отже, результати проведеного аналізу підтверджують актуальність розробки програмної системи для автоматичного визначення калорійності продуктів харчування за фотографією. Використання сучасних технологій машинного навчання та комп'ютерного зору дозволить підвищити рівень автоматизації процесу підрахунку калорій, зменшити кількість помилок під час введення даних та зробити систему більш зручною для користувачів.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

2. АЛГОРИТМИ РОБОТИ СИСТЕМИ ОБЧИСЛЕННЯ КАЛОРІЙ

2.1 Узагальнений алгоритм роботи системи обчислення

Система яка автоматично визначає калорійність продуктів за фотографією є багатокомпонентним програмним комплексом, у якому тісно взаємодіють мобільний клієнт, серверна частина на базі FastAPI, модуль машинного навчання та реляційна база даних PostgreSQL. Узагальнений алгоритм роботи системи включає повний ланцюжок обробки - від моменту отримання зображення до відображення результату і збереження запису у щоденнику харчування. Розуміння того як працює цей ланцюжок є необхідною передумовою для детального розгляду окремих підсистем у наступних підрозділах.

Загальну архітектуру взаємодії компонентів системи показано на рисунку 2.1. Між клієнтом і сервером організовано комунікацію через REST API, специфікацію якого автоматично генерує фреймворк FastAPI у форматі OpenAPI. Це забезпечує чітко визначений, задокументований та незалежний від платформи інтерфейс обміну даними. Сервер, у свою чергу, взаємодіє з модулем машинного навчання та базою даних через внутрішні виклики. Такий підхід до архітектури забезпечує модульність системи: кожен компонент може розвиватись і оновлюватись незалежно від інших за умови збереження інтерфейсу взаємодії.

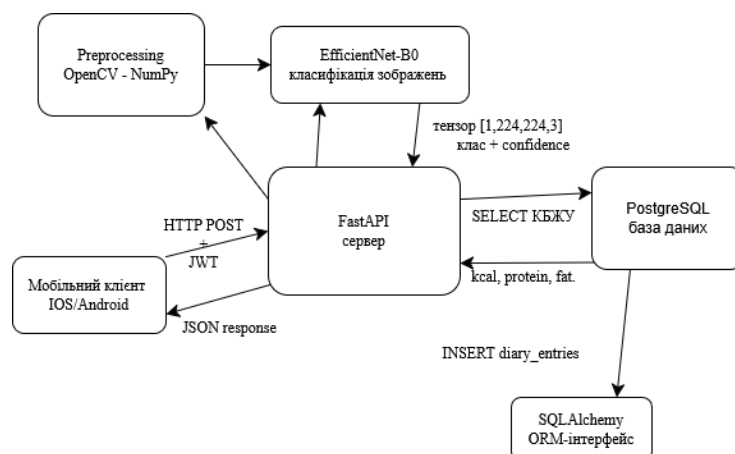


Рисунок 2.1 - Узагальнена схема взаємодії компонентів системи

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		25

Алгоритм роботи системи складається з шести послідовних етапів, які разом формують єдиний конвеєр обробки запиту.

Етап 1 - Отримання зображення від користувача.

Взаємодія між користувачем та системою розпочинається з отримання фотографії продукту або готової страви. Користувач робить фото за допомогою камери мобільного пристрою або вибирає раніше зроблене зображення з галереї. Мобільний клієнт виконує первинну перевірку перед відправкою: перевіряється відповідність формату файлу (JPEG, PNG), мінімальний розмір зображення (не менше 100×100 пікселів) та максимальний розмір файлу (не більше 10 МБ). Якщо зображення не відповідає цим критеріям, клієнт повідомляє користувачу про помилку без відправки запиту на сервер, це в свою чергу дозволить зменшити навантаження на мережу. Зображення, що пройшло перевірку, передається на сервер у форматі multipart/form-data або як рядок у кодуванні base64. Разом із зображенням передається токен аутентифікації JWT, це дозволить серверу ідентифікувати користувача та прив'язати результат до його облікового запису.

Етап 2 - Попередня обробка зображення.

Після виконання першого етапу сервер передає отримане зображення до спеціалізованого модуля попередньої обробки фото. Цей модуль реалізований засобами бібліотеки OpenCV і виконує стандартизований набір перетворень, необхідних для коректної та стабільної роботи нейронної мережі. Основна мета обробки полягає у приведенні зображення, знятого в довільних умовах (різне освітлення, кути, відстань та якість камери), до єдиного налаштованого вигляду, відповідного до вхідних вимог навченої моделі. До операцій обробки належать: декодування та валідація файлу, конвертація кольорового простору, зміна розміру, фільтрація шуму, нормалізація значень пікселів та формування пакетного тензора. Хороша підготовка вхідних даних є критично важливою умовою для досягнення стабільної точності класифікації - неправильна або взагалі відсутня нормалізація може суттєво знизити точність навіть дуже добре навченої моделі. Детальний опис кожного кроку попередньої обробки наведено у підрозділі 2.3.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		26

Етап 3 - Класифікація зображення нейронною мережею.

Підготовлений тензор зображення подається на вхід до навченої моделі EfficientNet-B0. У моделі виконується прямий прохід (forward pass) через усі шари і повертає вихідний вектор розмірністю 101, кожен елемент якого відповідає належності вхідного зображення до одного з 101 класів продуктів харчування з датасету Food-101. Клас у якому максимальна кількість значень ймовірності визначається як результат класифікації. Поряд із назвою класу система зазначає числовий показник впевненості моделі, який відображає максимальне значення у вихідному векторі після функції активації Softmax. Якщо значення confidence буде нижчим за встановлений пороговий рівень (стандартом використовується 0.5), система повідомить користувача про низьку впевненість у результаті та запропонує повторити фотографування з кращим освітленням чи кутом зйомки. За допомогою цього механізму можна уникнути передачі хибних результатів в наступні етапи обробки та підвищити надійність системи.

Етап 4 - Запит до бази даних продуктів.

Після того як ми отримуємо назви розпізнаного класу сервер формує запит до бази даних PostgreSQL. Запит виконується до таблиці foods за відповідністю назви класу продукту. У разі успішного знаходження запису система отримує значення калорійності, вмісту білків, жирів та вуглеводів на 100 грамів потрібного продукту. В іншому випадку коли потрібний запис відсутній у базі даних або confidence score є недостатньо високим, сервер повертає клієнту повідомлення та пропонує користувачеві ввести дані про продукт вручну або скористатися пошуком по базі. Така реалізація забезпечує відмовостійкість системи та можливість поступового розширення бази продуктів без змін у логіці класифікації. База даних містить записи для всіх 101 класів датасету Food-101 з актуальними значеннями КБЖВ відповідно до таблиць харчової цінності.

Етап 5 - Розрахунок КБЖВ порції.

На цьому етапі після отримання нормованих значень КБЖВ на 100 грам знайденого в системі продукту вона переходить до розрахунку поживної цінності

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		27

спожитої порції. На превеликий жаль визначити масу страви чи продукту лише за однією фотографією є технічно досить складним завданням, таке завдання потребує додаткових даних про глибину сцени, у даній системі застосовується один з двох підходів: використання стандартного розміру порції, попередньо визначеного для кожного класу продуктів, або введення маси порції користувачем вручну через відповідний інтерфейс після отримання результату розпізнавання. Розрахунок фактичної калорійності та поживної цінності порції виконується за формулою 1.1:

$$КБЖВ_{порції} = \frac{КБЖВ_{на\ 100г} \times маса_{порції}}{100} \quad (1.1)$$

де $КБЖВ_{на\ 100г}$ - відповідне значення калорій, білків, жирів та вуглеводів на 100 грамів продукту згідно за базою даних; $маса_{порції}$ - це маса фактично спожитої порції у грамах. Розрахунок виконується для чотирьох показників: калорійності (ккал), вмісту білків (г), жирів (г) та вуглеводів (г). Отримані значення будуть основою для формування добової та тижневої статистики харчування для користувача.

Етап 6 - Формування відповіді та збереження запису.

В кінцевому етапі сервер формує структуровану відповідь у форматі JSON файла, який містить: розпізнану назву продукту, значення confidence score, а також розраховані показники КБЖВ для зазначеної порції. Ця відповідь передається через REST API до мобільного клієнта та відображається користувачеві в зрозумілому інтерфейсі. Паралельно з цим сервер зберігає дані про прийом їжі у таблиці diary_entries бази даних. У кожному записі міститься ідентифікатор користувача, мітка дати та часу, назва розпізнаного продукту, введена маса порції та розраховані значення КБЖВ. Записи які були збереження використовуються для подальшого формування добової та тижневої статистики споживання поживних речовин, порівняння фактичного харчування з

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		28

персональними цільовими показниками та встановленими користувачем у профілі.

Для забезпечення взаємодії між клієнтом і сервером у системі реалізовано набір REST API ендпоінтів. Специфікацію основних ендпоінтів із зазначенням HTTP-методу, призначення та формату відповіді наведено у таблиці 2.1.

Таблиця 2.1 - Основні ендпоінти REST API системи

Ендпоінт	Метод	Формат відповіді JSON
/api/recognize	POST	product, confidence, kcal, protein, fat, carbs
/api/foods	GET	масив об'єктів продуктів
/api/foods	GET	id, name, kcal, protein, fat, carbs
/api/diary	POST	id, user_id, timestamp
/api/diary	GET	масив записів із сумарним КБЖВ
/api/users/me	GET	user_id goals

Алгоритм роботи системи який був описаний вище забезпечує повний цикл обробки запитів - від отримання фотографії до збереження результату. Наочне представлення роботи цього алгоритму у вигляді блок-схеми дозволяє деталізувати кожен етап. Розподіл відповідальності між компонентами системи спростить її тестування, масштабування і подальший розвиток.

2.2 Архітектура нейронної мережі

В даному проекті було використано архітектуру EfficientNet-B0 для класифікації зображень продуктів харчування. Дана архітектура була вибрана оскільки являється однією з найбільш ефективних архітектур згорткових нейронних мереж. EfficientNet-B0 була запропонована дослідниками в компанії

Google у 2019 році під час роботи над "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks" і засноване на принципі комплексного масштабування мережі (compound scaling). Суть цього принципу полягає в одночасному рівномірному масштабуванні трьох ключових характеристик мережі - глибини, ширини та роздільної здатності вхідного зображення - за допомогою єдиного коефіцієнта масштабування (ϕ). На відміну від попередніх підходів, які масштабують лише одну із цих характеристик, compound scaling дозволяє досягати вищої точності класифікації при значно меншій кількості параметрів і обчислювальних операцій. Для прикладу, EfficientNet-B0 з ~5.3 мільйонами параметрів досягає точності, яку можна зрівняти з ResNet-50 (~25 млн параметрів) на датасеті ImageNet. На рисунку 2.2 візуалізовано архітектуру EfficientNet-B0.

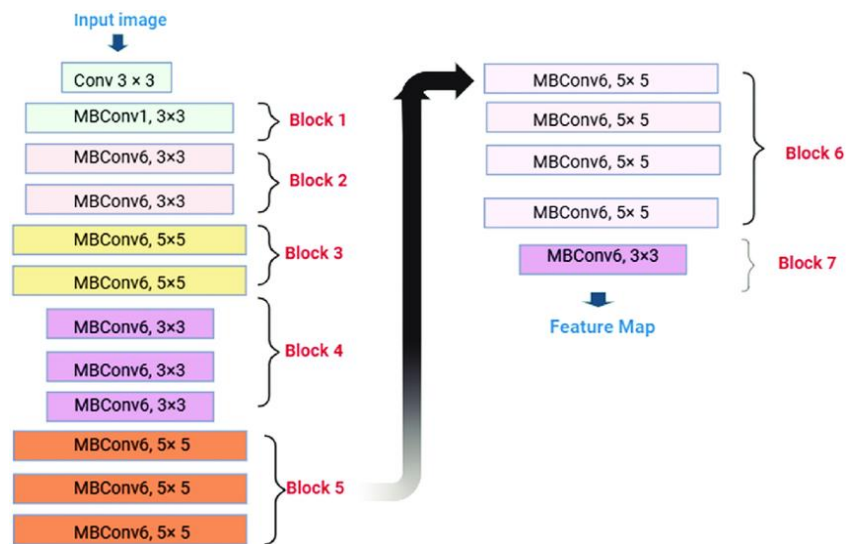


Рисунок 2.2 - Архітектура EfficientNet-B0

Основними причинами вибору алгоритму EfficientNet-B0 для даної системи були: по-перше, невелика кількість параметрів порівняно з іншими алгоритмами яка дозволяє розгортати модель як на сервері, так і потенційно на мобільних пристроях у форматі TensorFlow Lite. По-друге, висока ефективність архітектури яка забезпечує швидкий час виконання inference, а це є важливим для комфортної роботи мобільного додатку в режимі реального часу. По-третє,

EfficientNet показує одні з найкращих результатів на задачах класифікації зображень їжі при навчанні методом Transfer Learning на датасеті Food-101.

Базова модель EfficientNet-B0 приймає на вхід зображення за розміром 224×224 пікселі і складається з таких структурних компонентів.

Стемова згортка (Stem Convolution) - перший шар мережі виконує початкове просторове перетворення вхідного зображення. Він складається зі шару на який наложено 32 фільтра розміром 3×3 з кроком (stride) 2, нормалізацією по пакету (Batch Normalization) і функцією активації SiLU (Sigmoid Linear Unit або Swish). Завдяки кроку 2 стемова згортка зменшує просторові розміри з 224×224 до 112×112 пікселів, збільшуючи при цьому кількість каналів з 3 (RGB) до 32. Функція SiLU, яка визначається як $f(x) = x \cdot \sigma(x)$, σ - сигмоїдна функція, забезпечує кращу точність порівняно з традиційною ReLU для архітектур EfficientNet.

MBConv блоки (Mobile Inverted Bottleneck Convolution) - це основний будівельний блок архітектури EfficientNet, який був взятий з MobileNetV2 і вдосконалений механізмом каналної уваги. Кожен MBConv блок послідовно виконує такі операції як: розширювальна згортка 1×1 (expansion convolution) збільшує кількість каналів у певне задане число разів (expansion factor, зазвичай задають 6); глибинно-роздільна згортка (depthwise convolution) розміром 3×3 або 5×5 виконує просторову фільтрацію незалежно для всіх каналів, що в свою чергу суттєво зменшує кількість параметрів порівняно зі звичайними згортками; механізм Squeeze-and-Excitation (SE) виконує глобальне усереднення ознак по просторовій розмірній сітці, після чого двошаровий повнозв'язний блок обчислює вектор вагових коефіцієнтів для кожного каналу та застосовує їх, це дозволить моделі динамічно акцентувати увагу на найбільш інформативних каналах; проективна згортка 1×1 повертає кількість каналів до цільового значення. Якщо в кінці будуть явні розбіжності між вхідними та вихідними розмірами тоді використовується залишкове з'єднання (residual connection).

Класифікаційна голова (Classification Head) - являється завершальним блоком в мережі яка виконує задачу класифікації на основі вже сформованого

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		31

раніше набору ознак. Він складається зі згорткового шару 1×1 з 1280 фільтрами і функцією активації SiLU, операції глобального середнього пулінгу (Global Average Pooling), яка перетворює просторові ознаки у вектор фіксованої розмірності, шар регуляризацій Dropout з коефіцієнтом для відсіву 0.2 для запобігання перенавчанню, та звязаного шару Dense з виходом на 101 - відповідно до кількості класів Food-101. Фінальна функція активації Softmax перетворює вихідний вектор у вже нормований розподіл ймовірностей.

Навчання моделі методом Transfer Learning - виконується у два етапи які виконуються послідовно. На першому етапі (feature extraction) базова частина EfficientNet-B0, попередньо навчена вже на датасеті ImageNet (в якому понад 1.2 мільйона зображень та 1000 класів), повністю заморожується. Навчається виключно нова голова для виконання класифікації з вихідним шаром на 101 клас. Також використовується оптимізатор Adam зі значенням кроком навчання $1e-3$ та функція втрат Categorical Cross-Entropy. Навчання виконується протягом 10–15 епох до стабілізації точності у валідаційній вибірці. За допомогою цього етапу з'являється можливість швидко адаптувати низькорівневі візуальні ознаки (краї, текстури, форми), вивчені на ImageNet, до більш специфічної задачі розпізнавання їжі. В другому етапі (fine-tuning) розморожуються верхні шари базової моделі (десь приблизно останні 20 шарів) та виконується донавчання вже всієї мережі зі значно зменшеним кроком для навчання $1e-5$. Зменшений крок для навчання є дуже важливим для збереження вже навчених ознак: занадто великий крок навчання може спричинити «забування» корисних ознак, які були отримані на ImageNet. Для запобігання перенавчанню системи під час обох фаз застосовується аугментація навчальних зображень.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		32

2.3 Алгоритм попередньої обробки зображень

Одним із головних факторів є якість та стандарти вхідних даних, це визначає точність та стабільну роботу систем на основі нейронних мереж. Зображення які були отримані з камери пристрою у реальних умовах, відрізняються за параметрами: роздільною здатністю, яскравістю та контрастом сцени, колірною температурою, наявністю розмиття через рух, а також рівнем цифрового шуму матриці камери. Нейронна мережа, очікує нормалізоване зображення на вході, воно має бути певного розміру з нормою значення пікселів. Тому між отриманням зображення від клієнта та його подачею до класифікатора потрібно виконати набір певних перетворень - конвеєр попередньої обробки (preprocessing pipeline), що перетворює довільне вхідне зображення на стандартизований тензор.

У даному проєкті функція попередньої обробки реалізований на серверній стороні мовою Python за використанням бібліотеки OpenCV та засобів NumPy. Розміщення логіки попередньої обробки на сервері, а не на клієнті, забезпечує однаковість обробки незалежно від типу пристрою, а також спрощує майбутні оновлення конвеєру без необхідності оновлення клієнтської частини. Конвеєр складається з шести послідовних кроків, які наведених у таблиці 2.3.

Таблиця 2.3 - Кроки алгоритму попередньої обробки зображень

№	Операція	Інструмент	Мета
1	Декодування та валідація	cv2.imdecode()	Перевірка коректності та цілісності файлу
2	Конвертація BGR → RGB	cv2.cvtColor()	Відповідність колірного формату вагам моделі
3	Зміна розміру (224×224)	cv2.resize(), INTER_LINEAR	Стандартизація розміру вхідного тензора

Продовження таблиці 2.3

№	Операція	Інструмент	Мета
4	Видалення шуму	cv2.GaussianBlur(), ядро 3×3	Зменшення впливу цифрових артефактів
5	Нормалізація [0, 1]	numpy: / 255.0	Масштабування значень пікселів для мережі
6	Розширення розмірності	np.expand_dims(axis=0)	Формування пакетного тензора [1,224,224,3]

Першим кроком є декодування та валідація - вхідне зображення яке надходить до модуля попередньої обробки у вигляді байтового коду, отриманого з multipart/form-data, або як рядок у форматі base64. На цьому кроці виконується декодування байтового потоку у матрицю NumPy за допомогою функції cv2.imdecode(). Одночасно відбувається перевірка коректності отриманих даних: матриця не повинна бути порожньою, зображення повинно містити три кольорових канали і мати мінімальний розмір необхідний для подальшої роботи з ним. У разі виявлення будь-якої помилки модуль генерує виключення з описом помилки, що дозволяє серверу повернути клієнту інформативне повідомлення про некоректний формат вхідних даних.

Наступним кроком є функція cv2.imdecode() яка за замовчуванням повертає зображення у колірному просторі BGR (Blue-Green-Red) - порядок каналів, який прийнятий у бібліотеці OpenCV з метою сумісності зі старими системами обробки відео. Проблема в тому що модель EfficientNet-B0 була навчена на зображеннях у стандартному колірному просторі RGB (Red-Green-Blue). Невідповідність каналів призводить до систематичної помилки класифікації, оскільки модель інтерпретує значення каналів некоректно. Для виправлення цього виконується конвертація: cv2.cvtColor(image, cv2.COLOR_BGR2RGB). Ця операція є швидкою і не змінює самих значень пікселів - лише переставляє канали у правильному порядку.

Кроком №3 є зміна розміру зображення. EfficientNet-B0 вимагає щоб зображення фіксованого розміру 224×224 пікселі. Проте реальні зображення з камер сучасних пристроїв зазвичай мають значно більшу роздільну здатність (від 1080×1080 до 4032×3024 пікселів). Зміна розміру виконується за допомогою функції `cv2.resize(image, (224, 224), interpolation=cv2.INTER_LINEAR)`. Метод білінійної інтерполяції `INTER_LINEAR` обраний як оптимальний компроміс між якістю масштабування і швидкістю виконання. При зменшенні роздільної здатності він визначає середню кількість значень пікселів у блоці, що природно зглажує шум. Слід зазначити, що при даному підході пропорції зображення не зберігаються - з причини виконання примусового масштабування до квадрату. Це може призводити до геометричних спотворень для деяких предметів, однак для більшості продуктів харчування ця проблема є несуттєвою завдяки тому, що класифікатор орієнтується переважно на текстурні, а не геометричні ознаки.

Кроком чотири є видалення шуму на зображенні, особливо характерного для зображень, знятих при слабкому освітленні або на камери низької якості, тут застосовується розмиття за Гаусом: `cv2.GaussianBlur(image, (3, 3), 0)`. Ядро розміром 3×3 пікселів обрано свідомо невеликим - воно ефективно усуває дрібний шум, не розмиваючи при цьому значущих деталей зображення: країв об'єктів, текстур поверхні та кольорних переходів, оскільки вони є ключовими ознаками для класифікатора. Застосування більшого ядра (5×5 або 7×7) призводить до надмірного розмиття що в свою чергу призводить до втрати інформації про деталі продукту, що негативно вплине на точність класифікації.

Наступний крок №5 відбувається нормалізація значень пікселів. Значення пікселів зображення після попередніх кроків знаходяться у цілочисельному діапазоні від 0 до 255. Для правильної роботи нейронної мережі необхідно перевести їх до числового діапазону з рухомою комою. Спочатку тип масиву змінюється на `float32`: `image = image.astype(np.float32)`, після чого виконується ділення на 255.0 для отримання значень у діапазоні $[0.0, 1.0]$. Додатково може застосовуватись поканальна стандартизація за середніми значеннями та стандартними відхиленнями датасету ImageNet: `mean = [0.485, 0.456, 0.406]`, `std`

					КР.KI.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		35

= [0.229, 0.224, 0.225]. Формула 2.2 - це формула стандартизації для кожного каналу c має вигляд:

$$x'_c = \frac{x_c - mean_c}{std_c} \quad (2.2)$$

де x'_c - це нормалізоване значення пікселя каналу у діапазоні [0.0, 1.0]; $mean_c$ та std_c - це середнє значення та стандартне відхилення для певного каналу в датасеті ImageNet. Ця операція забезпечує максимальну сумісність вхідних даних із попередньо навченими вагами EfficientNet-B0.

Останнім кроком є - розширення розмірності. Нейронна мережа EfficientNet-B0, реалізована за допомогою засобів TensorFlow/Keras, приймає на вхід тензор з чотирма вимірами у форматі [batch_size, height, width, channels]. Після минулих кроків оброблене зображення має форму [224, 224, 3]. Для подачі одного зображення на вхід до мережі нам необхідно додати вимір пакету: `image = np.expand_dims(image, axis=0)`, що перетворює тензор у форму [1, 224, 224, 3]. Вже сформований тензор передається до функції `model.predict()`, яка виконує класифікацію і повертає вихідний вектор ймовірностей форми [1, 101].

Таким чином, реалізований конвеєр попередньої обробки з шести послідовних кроків забезпечує перетворення нестартизованого вхідного зображення з камери пристрою на стандартизований тензор, придатний для подачі до класифікатора EfficientNet-B0. Кожен крок вирішує конкретну проблему невідповідності між реальними даними та очікуваним форматом вхідних даних нейронної мережі, що в сукупності дозволяє мінімізувати вплив умов зйомки на кінцеву точність розпізнавання.

І підсумовуючи у другому розділі було детально описано алгоритми роботи системи автоматичного визначення калорійності продуктів харчування за фотографією.

У підрозділі 2.1 розглянуто узагальнений шестистапний алгоритм роботи системи: отримання та первинна валідація зображення на стороні клієнта,

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		36

попередня обробка на сервері, класифікація нейронною мережею з механізмом порогового confidence score, запит до бази даних PostgreSQL, розрахунок КБЖВ порції за встановленою формулою та збереження результату у щоденнику харчування. Описано архітектуру REST API системи та основні ендпоінти з форматами запитів і відповідей, а також механізм аутентифікації на основі JWT-токенів.

У підрозділі 2.2 детально описано архітектуру згорткової нейронної мережі EfficientNet-B0. Розглянуто принцип compound scaling, структуру стемової згортки, конфігурацію семи стадій MBConv блоків з механізмом Squeeze-and-Excitation та класифікаційну голову. Наведено порівняльну таблицю стадій мережі з параметрами вхідних і вихідних тензорів. Описано двофазний підхід до навчання моделі методом Transfer Learning на датасеті Food-101: фаза feature extraction з замороженою базою та фаза fine-tuning верхніх шарів зі зменшеним learning rate.

Таким чином, запропонований підхід - поєднання ефективної архітектури EfficientNet-B0, методу Transfer Learning та стандартизованого конвеєру попередньої обробки зображень - забезпечує необхідний рівень автоматизації та точності для практичного застосування системи контролю харчування.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		37

3 РЕАЛІЗАЦІЯ ПРОГРАМНОГО КОМПЛЕКСУ

3.1 Загальна архітектура програмного комплексу

Розроблений програмний комплекс реалізований за архітектурним патерном клієнт-сервер і складається з двох основних підсистем: серверної частини на базі Python та FastAPI і мобільного клієнта на React Native з Expo. Взаємодія між підсистемами здійснюється через REST API з автентифікацією на основі JWT-токенів [16].

Серверна частина розгортається у Docker-контейнерах за допомогою docker-compose.yml, який визначає два сервіси: db - контейнер з СУБД PostgreSQL 15, та api - контейнер з FastAPI-застосунком. Файл моделі машинного навчання монтується у контейнер через volume, що дозволяє оновлювати модель без перебудови образу. Така архітектура забезпечує ізоляцію залежностей, відтворюваність середовища та спрощене розгортання [11].

Мобільний клієнт розроблено засобами Expo SDK 52, що дозволяє запускати додаток на Android та iOS з єдиної кодової бази. Для навігації між екранами використовується бібліотека React Navigation з комбінацією нативного стеку (Native Stack) та нижньої панелі вкладок (Bottom Tabs). HTTP-запити виконуються через Axios з автоматичним додаванням JWT-токена до заголовків через interceptor. Токен зберігається у захищеному сховищі пристрою за допомогою Expo Secure Store.

Загальна структура проєкту наведена у таблиці 3.1.

Таблиця 3.1 – Структура файлів програмного комплексу

Файл / директорія	Призначення
backend/app/main.py	Точка входу FastAPI, реєстрація роутерів, lifespan
backend/app/config.py	Pydantic Settings, зчитування змінних середовища
backend/app/database.py	Асинхронне підключення до PostgreSQL через asynpcpg

Продовження таблиці 3.1

Файл / директорія	Призначення
backend/app/models/	SQLAlchemy моделі таблиць: users, foods, diary_entries
backend/app/schemas/	Pydantic схеми валідації запитів та відповідей
backend/app/routers/	Роутери FastAPI: auth, food, diary
backend/app/services/	Бізнес-логіка: auth_service, food_service
backend/app/ml/	ML-модуль: predictor, preprocessor, ваги моделі
backend/docker-compose.yml	Оркестрація контейнерів PostgreSQL та FastAPI
mobile/src/screens/	Екрани React Native: Home, Camera, Result, Diary, Profile
mobile/src/navigation/	Конфігурація навігації між екранами
mobile/src/services/	API-клієнт (axios) та сервіс авторизації
ml_training/	Jupyter notebook для навчання моделі на Kaggle

Проведений аналіз існуючих рішень показав, що більшість популярних систем контролю харчування потребують ручного введення даних або використання сканування штрих-кодів. Повноцінна автоматизація процесу розпізнавання продуктів харчування за фотографією залишається актуальною задачею. Найперспективнішим напрямком для реалізації такої функціональності є застосування методів глибокого навчання, зокрема згорткових нейронних мереж з Transfer Learning на великих датасетах продуктів харчування.

3.2 Реалізація серверної частини

3.2.1 Конфігурація та запуск застосунку

Конфігурація серверної частини реалізована у файлі config.py за допомогою класу Settings бібліотеки pydantic-settings. Клас автоматично зчитує значення змінних із файлу .env та надає їх через атрибути з типовою перевіркою.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		39

Серед параметрів конфігурації: DATABASE_URL для підключення до PostgreSQL через asyncpg, SECRET_KEY та ALGORITHM

M для підпису JWT-токенів, ACCESS_TOKEN_EXPIRE_MINUTES для терміну дії токена, MODEL_PATH та CLASS_NAMES_PATH для шляхів до файлів ML-моделі, IMG_SIZE для розміру вхідного зображення нейронної мережі.

Точка входу застосунку - файл main.py - визначає FastAPI-застосунок з lifespan-менеджером. При старті застосунку lifespan виконує створення таблиць бази даних через SQLAlchemy та ініціалізацію ML-моделі у пам'яті. CORS middleware налаштований для дозволу запитів з мобільного клієнта. Роутери підключаються з префіксом /api/v1/ та тегами для документації Swagger UI.

3.2.2 Моделі бази даних

У системі реалізовано три таблиці. Таблиця users зберігає дані облікових записів: email, username, hashed_password, а також персональні параметри для розрахунку норм харчування - вік, вагу, зріст, стать, рівень активності та цільові значення КБЖВ. Таблиця foods містить базу продуктів харчування: назви українською та англійською мовами, категорію, ml_class_name для відповідності класам датасету Food-101 та значення калорій, білків, жирів, вуглеводів та клітковини на 100 г. Таблиця diary_entries зберігає записи щоденника харчування: посилання на користувача та продукт, тип прийому їжі (breakfast, lunch, dinner, snack), дату, вагу порції та кешовані значення КБЖВ. Кешування значень КБЖВ у записі щоденника є важливим архітектурним рішенням - воно гарантує збереження точних даних навіть у разі подальшої зміни інформації про продукт у базі. У таблиці 3.2 наведено структуру таблиць.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		40

Таблиця 3.2 – Структура таблиць бази даних

Таблиця	Поля
Users	id, email, username, hashed_password, age, weight_kg, height_cm, gender, activity_level, daily_calories_goal, daily_protein_goal, daily_fat_goal, daily_carbs_goal
Foods	id, name_ua, name_en, ml_class_name, category, calories_per_100g, protein_per_100g, fat_per_100g, carbs_per_100g, fiber_per_100g, typical_serving_g
diary_entries	id, user_id, food_id, meal_type, eaten_date, weight_g, calories, protein, fat, carbs, ml_confidence, ml_predicted_class, eaten_at

Взаємодія з базою даних PostgreSQL реалізована через асинхронний SQLAlchemy 2.0 з драйвером asyncpg. Файл database.py створює AsyncEngine та AsyncSessionLocal, а також визначає залежність get_db() для ін'єкції сесії у роутери FastAPI. Використання асинхронного режиму дозволяє серверу одночасно обслуговувати кілька запитів без блокування потоку виконання під час очікування відповіді від бази даних. В додатку Г графічно реалізовано таблиці їх поля та взаємодія між ними.

3.2.3 Реалізація аутентифікації

Аутентифікація реалізована у файлах routers/auth.py та services/auth_service.py. Система підтримує чотири ендпоінти: POST /api/v1/auth/register для реєстрації нового користувача, POST /api/v1/auth/login для входу та отримання токена, GET /api/v1/auth/me для отримання профілю поточного користувача та PATCH /api/v1/auth/me для оновлення персональних даних.

Паролі хешуються алгоритмом bcrypt через бібліотеку passlib. JWT-токени генеруються функцією create_access_token() з бібліотеки python-jose та містять ідентифікатор користувача, ім'я та час закінчення дії. Термін дії токена налаштовується через параметр ACCESS_TOKEN_EXPIRE_MINUTES у файлі

.env. Захищені маршрути використовують залежність `get_current_user()`, яка витягує токен зі схеми OAuth2PasswordBearer, декодує його та повертає об'єкт поточного користувача.

При оновленні профілю через `PATCH /auth/me` система автоматично розраховує денну норму КБЖВ за формулою Маффіна-Джеора [27]. Базальний метаболізм (BMR) обчислюється залежно від статі, а потім множиться на коефіцієнт активності (від 1.2 для малорухливого способу життя до 1.725 для дуже активного) для отримання загального добового споживання енергії (TDEE). На основі TDEE автоматично розраховуються цільові значення білків, жирів та вуглеводів.

3.2.4 Реалізація API продуктів харчування

Роутер `food.py` реалізує чотири ендпоінти для роботи з продуктами. `GET /api/v1/food/search?q=` виконує текстовий пошук по полях `name_ua` та `name_en` через оператор `ILIKE` з підтримкою реєстронезалежного пошуку. `GET /api/v1/food/{food_id}` повертає детальну інформацію про продукт за ідентифікатором.

Основним ендпоінтом є `POST /api/v1/food/recognize`, який приймає фотографію у форматі `multipart/form-data` та параметр `weight_g` з вагою порції. Реалізована перевірка типу файлу (підтримуються `jpeg`, `png`, `webp`, `heic`) та розміру (не більше 10 МБ). Після успішної валідації виконується ML-інференс, результатом якого є назва розпізнаного класу та показник впевненості. Потім система шукає відповідний продукт у базі даних за полем `ml_class_name`. Якщо продукт першого класу відсутній у базі, система послідовно перебирає наступні позиції зі списку `top-5` доти, доки не знайде відповідність. Це забезпечує відмовостійкість при неповноті бази продуктів. Відповідь містить інформацію про розпізнаний клас, рівень впевненості, повний список `top-5` альтернатив та розраховані значення КБЖВ для вказаної ваги порції.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		42

3.2.5 Реалізація щоденника харчування

Роутер `diary.py` реалізує CRUD-операції для щоденника харчування. `POST /api/v1/diary/` приймає `food_id`, `meal_type`, `weight_g` та опціональну `eaten_date` (за замовчуванням поточна дата), перевіряє існування продукту, розраховує КБЖВ для вказаної ваги за формулою $(\text{значення_на_100г} \times \text{вага_порції} / 100)$ і зберігає запис у таблицю `diary_entries`.

`GET /api/v1/diary/` повертає список записів за вказану дату з приєднаними назвами продуктів через `LEFT OUTER JOIN`. `GET /api/v1/diary/stats` повертає агреговану статистику за день: суму калорій, білків, жирів та вуглеводів, кількість записів та відсоток виконання денної норми для кожного показника. `DELETE /api/v1/diary/{entry_id}` видаляє запис після перевірки належності поточному користувачу.

3.3 Реалізація модуля машинного навчання

3.3.1 Навчання моделі

Модель `EfficientNet-B0` навчалась на датасеті `Food-101` у хмарному середовищі `Kaggle` з використанням `GPU Tesla P100`. Навчання виконувалось у два послідовних етапи, що відповідає методу `Transfer Learning`.

На першому етапі (`Feature Extraction`) базова частина `EfficientNet-B0` із попередньо навченими вагами `ImageNet` була заморожена, а навчалась лише класифікаційна голова: шар `GlobalAveragePooling2D`, `BatchNormalization`, два шари `Dropout` з коефіцієнтом 0.4 та 0.3, `Dense(128, relu)` та вихідний `Dense(101, softmax)`. Навчання тривало 8 епох з `learning rate 1e-4` та оптимізатором `Adam` з параметром `clipnorm=1.0` для запобігання вибуху градієнтів.

На другому етапі (`Fine-Tuning`) розморожувались верхні 20 шарів базової моделі та виконувалось донавчання всієї мережі з `learning rate 1e-6`. Значно менший `learning rate` на цьому етапі є критично важливим для збереження

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		43

низькорівневих ознак, вивчених на ImageNet, і запобігання катастрофічному забуванню. Навчання тривало 12 епох. Для обох фаз використовувались callbacks: ModelCheckpoint для збереження найкращої моделі за val_accrasy, ReduceLROnPlateau для автоматичного зменшення learning rate при відсутності покращення та EarlyStopping для запобігання перенавчанню.

Для аугментації навчальних даних застосовувались: горизонтальне відображення (RandomFlip), поворот до $\pm 10^\circ$ (RandomRotation), масштабування $\pm 10\%$ (RandomZoom). Розмір батчу - 32 зображення, розмір вхідного зображення - 224×224 пікселів. Параметри конвеєру підготовки даних було оптимізовано для роботи в умовах обмеженої пам'яті: buffer_size=500 для shuffle та prefetch=2.

3.3.2 Завантаження та ініціалізація моделі

ML-логіка серверної частини реалізована у двох файлах: preprocessor.py та predictor.py. Клас FoodPredictor у файлі predictor.py реалізований за патерном Singleton - модель завантажується в пам'ять один раз при старті сервера через lifespan-менеджер у main.py і використовується для всіх наступних запитів без повторного завантаження. Це суттєво скорочує час відповіді на запити розпізнавання, оскільки завантаження моделі TensorFlow з диску займає декілька секунд.

При ініціалізації FoodPredictor завантажує модель із файлу efficientnet_food101.h5 через tf.keras.models.load_model() та список імен класів із class_names.json. Метод predict() приймає байтовий вміст зображення, передає його до ImagePreprocessor для стандартизації, виконує інференс через model.predict() та повертає об'єкт PredictionResult з полями: class_name (назва розпізнаного класу), confidence (числове значення впевненості від 0 до 1) та top5 (список п'яти найімовірніших класів з їх значеннями впевненості). Метод is_confident() перевіряє чи перевищує confidence пороговий рівень 0.5.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		44

3.3.3 Попередня обробка зображень

Клас `ImagePreprocessor` у файлі `preprocessor.py` реалізує конвеєр попередньої обробки зображень, описаний у розділі 2. Метод `preprocess_bytes()` приймає байтовий вміст файлу зображення та повертає нормалізований тензор форми $[1, 224, 224, 3]$. Конвеєр виконує такі операції: декодування байтів у масив `NumPy` через `cv2.imdecode()`, конвертацію колірного простору `BGR`→`RGB` через `cv2.cvtColor()`, зміну розміру до 224×224 пікселів методом білінійної інтерполяції, гаусове розмиття з ядром 3×3 для видалення шуму, нормалізацію значень пікселів до діапазону $[0.0, 1.0]$ та розширення розмірності для формування батчу. Додатково реалізований метод `enhance_image()` з адаптивним вирівнюванням гістограми (CLAHE) для покращення контрасту зображень, знятих при поганому освітленні.

3.4 Реалізація мобільного клієнта

3.4.1 Структура та навігація

Мобільний клієнт розроблено на `React Native` з `Expo SDK 52` та мовою `TypeScript`. Точка входу - файл `App.tsx` - ініціалізує компонент `Navigation`. Навігаційна структура визначена у `src/navigation/index.tsx` та складається з двох рівнів: зовнішнього `Native Stack Navigator` та внутрішнього `Bottom Tab Navigator`.

Зовнішній стек містить: `LoginScreen` та `RegisterScreen` для неавторизованих користувачів, `Main (Bottom Tabs)` як головну оболонку після входу, `CameraScreen` у режимі повноекранного модального вікна (`fullScreenModal`) та `ResultScreen` для відображення результатів розпізнавання. `Bottom Tab Navigator` об'єднує три основні вкладки: `HomeScreen` (головний екран), `DiaryScreen` (щоденник) та `ProfileScreen` (профіль) з іконками з бібліотеки `@expo/vector-icons`.

Типи навігаційних параметрів визначені у `src/types/index.ts` через `TypeScript`-інтерфейси `RootStackParamList` та `MainTabParamList`, що забезпечує

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		45

типобезпечну передачу параметрів між екранами. Зокрема, ResultScreen отримує об'єкт RecognizeResult через параметр навігації.

3.4.2 Сервіси взаємодії з API

HTTP-клієнт реалізований у файлі `src/services/api.ts` на основі бібліотеки `Axios`. Базова URL-адреса сервера визначається константою `BASE_URL`. `Request interceptor` автоматично витягує JWT-токен із `Expo Secure Store` та додає його до заголовка `Authorization: Bearer` кожного запиту, що позбавляє необхідності передавати токен вручну при кожному виклику. `Timeout` запиту встановлено на 30 секунд з урахуванням часу ML-інференсу на сервері.

API-клієнт структурований у три об'єкти: `authAPI` (`register`, `login`, `getMe`, `updateProfile`), `foodAPI` (`search`, `getById`, `recognize`) та `diaryAPI` (`addEntry`, `getEntries`, `getStats`, `deleteEntry`). Метод `foodAPI.recognize()` формує `multipart/form-data` запит, де зображення передається як об'єкт з полями `uri`, `type` та `name`. Сервіс `auth.ts` реалізує операції збереження та видалення токена через `SecureStore`, а також методи `login()` та `register()`, які виконують запит до API та автоматично зберігають токен після успішної відповіді.

3.4.3 Реалізація екранів

`LoginScreen` та `RegisterScreen` реалізують форми для входу та реєстрації відповідно. Обидва екрани використовують `KeyboardAvoidingView` для коректного відображення при відкритій клавіатурі та виконують клієнтську валідацію введених даних перед відправкою запиту. При успішній авторизації викликається `navigation.replace('Main')` для переходу до головного екрана без можливості повернення до форми входу. На зображенні 3.1 показано екрани входу та реєстрації.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		46

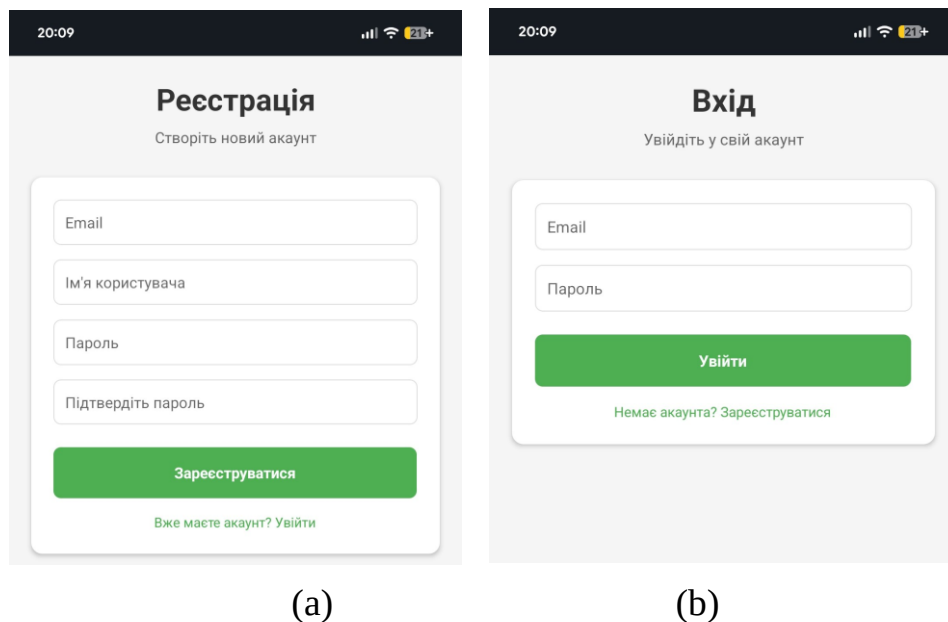


Рисунок 3.1 – (a) приклад екрану реєстрації, (b) приклад екрану входу

HomeScreen відображає персоналізоване привітання, денну статистику КБЖВ через компонент KBZVCard та прогрес-бар досягнення калорійної норми. Дані завантажуються при кожному фокусуванні екрана через хук useFocusEffect, що забезпечує актуальність інформації після додавання нових записів. Кнопка «Сфотографувати їжу» здійснює навігацію до CameraScreen. А кнопка «Знайти вручну» знаходить необхідний запис про продукт чи страву лише за назвою без використання фото. Реалізація екрану наведено на рисунку 3.2.

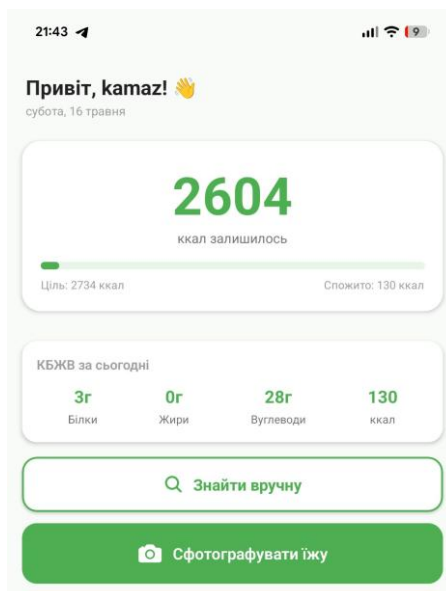


Рисунок 3.2 – Екран головної сторінки

CameraScreen реалізує інтерфейс для отримання зображення. Після перевірки дозволу на використання камери через хук `useCameraPermissions` відображається прев'ю з рамкою наведення та панель управління з полем введення ваги порції, кнопкою зйомки та кнопкою вибору з галереї через `expo-image-picker`. Після отримання зображення викликається метод `recognize()`, який відправляє запит на сервер та перенаправляє на ResultScreen. Приклад реалізації екрану наведено на рисунку 3.3.

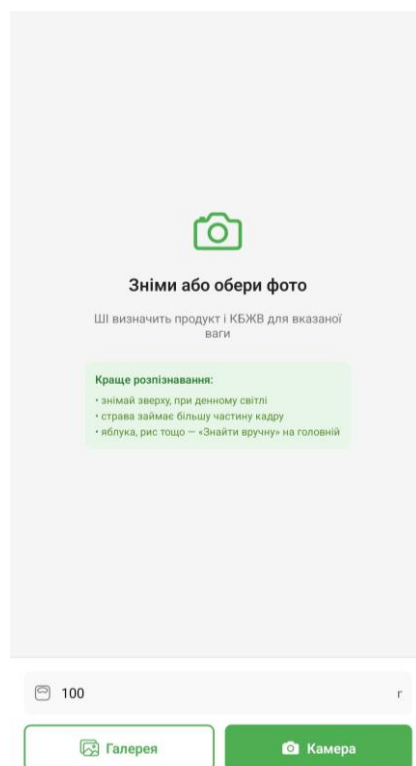


Рисунок 3.3 – Екран отримання зображень для подальшої роботи

ResultScreen відображає результати розпізнавання: назву продукту, показник впевненості моделі з кольоровою індикацією (зелений при $\text{confidence} \geq 70\%$, помаранчевий при нижчому значенні), значення КБЖВ через KBZVCard та список топ-5 альтернатив. Користувач вибирає тип прийому їжі (сніданок, обід, вечеря, перекус) та зберігає запис у щоденнику через кнопку «Додати до щоденника». На рисунку 3.4 наведено графічний інтерфейс екрану з результатами.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		48

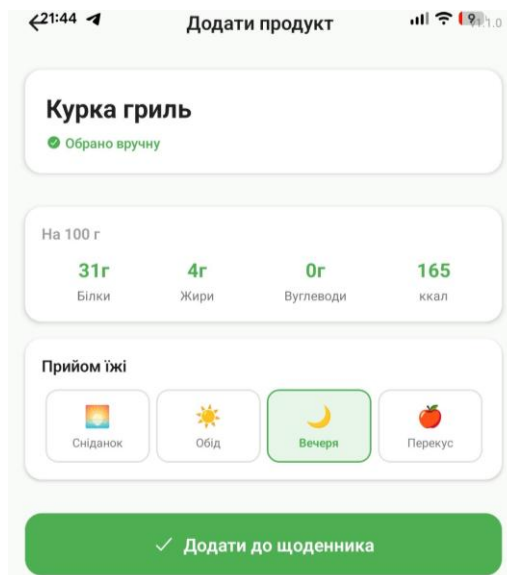


Рисунок 3.4 – Екран з інформацією про страву

DiaryScreen відображає записи щоденника за вибрану дату з навігацією по датах через кнопки перемотування. На рисунку 3.5 показано реалізацію графічного інтерфейсу щоденника.

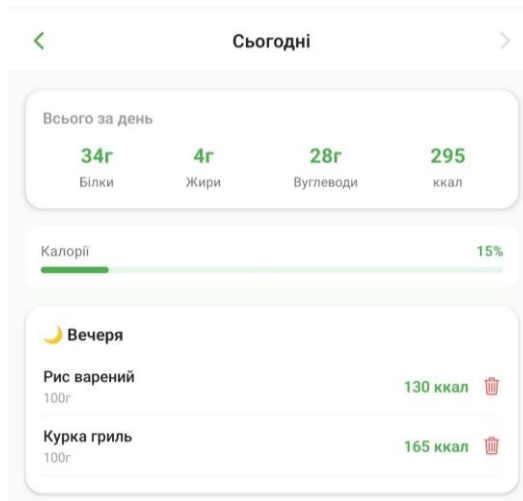


Рисунок 3.5 – Приклад щоденника із записами про їжу

Записи згруповані за типом прийому їжі. Статистика дня показує сумарні значення КБЖВ та прогрес виконання денної норми. При перевищенні норми прогрес-бар змінює колір на червоний. Кожен запис можна видалити через свайп або кнопку з підтвердженням. На рисунку 3.5 показано реалізацію графічного інтерфейсу щоденника.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		49

ProfileScreen відображає аватар з першою літерою імені, дані облікового запису та блок денної норми. У режимі редагування відображається форма для введення персональних параметрів: вік, вага, зріст, стать та рівень активності. При збереженні сервер автоматично розраховує денну норму КБЖВ. Кнопка виходу з підтвердженням видаляє токен зі SecureStore та перенаправляє на екран входу. Далі на рисунку 3.6 показано як реалізовано інтерфейс ProfileScreen.

The screenshot displays the ProfileScreen interface. At the top, there is a green circular avatar with the letter 'K', the username 'kamaz', and the email 'nazarsitnik4@gmail.com'. Below this, a section titled 'Денна норма' (Daily norm) shows four values: 2734 ккал (Calories), 123 г (g) Білки (Proteins), 76 г (g) Жири (Fats), and 389 г (g) Вугл. (Carbs). The 'Особисті дані' (Personal data) section includes a note about automatic calculation of daily norms and input fields for Age (21), Weight (77 kg), Height (175 cm), Gender (Male), and Activity Level (Moderately active, which is selected with a green checkmark). A bottom navigation bar contains icons for Home, Shopping, and Profile, with a green bar labeled 'Зберегти' (Save) above the Profile icon.

Рисунок 3.6 – Приклад екрану з профілем користувача та його даними

3.5 Наповнення бази даних та розгортання

3.5.1 Початкове наповнення бази даних

Скрипт `seed_foods.py` виконує початкове наповнення таблиці `foods`. Скрипт містить масив `FOODS` з даними понад 50 продуктів харчування, для кожного з яких вказані: назва українською та англійською, `ml_class_name` для

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		50

відповідності до класів датасету Food-101, категорія (dish, meat, vegetable, fruit, grain, dairy, snack, drink), значення калорій, білків, жирів, вуглеводів та клітковини на 100 г, а також типовий розмір порції.

З 50 продуктів 25 мають визначений `ml_class_name`, що дозволяє системі автоматично знаходити їх у базі після ML-розпізнавання. Решта продуктів доступна через текстовий пошук. Скрипт перевіряє наявність даних перед вставкою та пропускає наповнення якщо таблиця вже містить записи, що запобігає дублюванню при повторному запуску.

3.5.2 Розгортання системи

Серверна частина розгортається за допомогою Docker Compose. Файл `docker-compose.yml` визначає сервіс `db` на базі образу `postgres:15-alpine` з `healthcheck` для перевірки готовності СУБД та сервіс `api`, який будується з `Dockerfile` проєкту. Сервіс `api` запускається лише після успішного проходження `healthcheck` бази даних через директиву `depends_on` з умовою `service_healthy`. Файл моделі монтується у контейнер як `read-only volume`, що дозволяє замінювати модель без перебудови Docker-образу.

`Dockerfile` базується на образі `python:3.11-slim` та встановлює системну залежність `libgl1`, необхідну для коректної роботи `OpenCV`. Застосунок запускається через `uvicorn` на порту 8000 з параметром `--reload` для режиму розробки. Конфігурація зчитується з файлу `.env` через `env_file` директиву в `docker-compose.yml`.

Перед початком розгортання системи необхідно переконатися, що на цільовій машині встановлено платформу контейнеризації Docker з плагіном Docker Compose, а також середовище Node.js для роботи мобільного клієнта.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		51

ВИСНОВКИ

В результаті виконання бакалаврської роботи розроблено мобільний програмний комплекс для автоматичного визначення калорійності та КБЖВ продуктів харчування за фотографією. Мета роботи досягнута - реалізована система дозволяє користувачу сфотографувати страву і миттєво отримати розраховані значення калорій, білків, жирів та вуглеводів без ручного введення даних.

В ході виконання роботи досягнуто таких результатів:

1. Проведено аналіз існуючих систем контролю харчування (MyFitnessPal, FatSecret, Yazio), визначено їх основні недоліки - відсутність автоматичного розпізнавання їжі за фотографією та необхідність ручного введення даних. Обґрунтовано актуальність розробки власного рішення з використанням методів глибокого навчання.

2. Обрано та обґрунтовано технологічний стек: Python 3 з FastAPI для серверної частини, React Native з Expo для мобільного клієнта, PostgreSQL для зберігання даних, TensorFlow/Keras для навчання нейронної мережі та OpenCV для попередньої обробки зображень.

3. Розроблено архітектуру нейронної мережі на базі EfficientNet-B0 з методом Transfer Learning. Описано алгоритм двофазного навчання: на першій фазі навчається лише класифікаційна голова при замороженій базовій моделі, на другій - виконується донавчання верхніх 20 шарів з меншим значенням learning rate.

4. Описано алгоритм попередньої обробки зображень, що включає декодування байтів через OpenCV, конвертацію колірного простору BGR→RGB, масштабування до розміру 224×224 пікселів, нормалізацію значень пікселів та стандартизацію за параметрами ImageNet.

5. Навчено модель EfficientNet-B0 на датасеті Food-101 (101 клас, 101 000 зображень) у хмарному середовищі Kaggle з використанням GPU Tesla P100.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		52

Застосовано аугментацію даних (горизонтальне відображення, поворот, масштабування) для підвищення узагальнювальної здатності моделі.

6. Реалізовано серверну частину на FastAPI з асинхронною взаємодією з PostgreSQL через SQLAlchemy 2.0 та asyncpg. Розроблено REST API з ендпоінтами для аутентифікації (JWT + bcrypt), розпізнавання їжі за фотографією, текстового пошуку продуктів та ведення щоденника харчування.

7. Реалізовано автоматичний розрахунок денної норми КБЖВ для кожного користувача за формулою Маффіна-Джеора на основі антропометричних даних (вага, зріст, вік, стать) та рівня фізичної активності.

8. Розроблено мобільний клієнт на React Native (Expo SDK) з п'ятьма основними екранами: головний екран з прогресом калорій, екран камери для фотографування їжі, екран результатів розпізнавання з КБЖВ, щоденник харчування по днях та профіль з налаштуваннями.

9. Розгорнуто серверну частину у Docker-контейнерах за допомогою docker-compose з автоматичним запуском PostgreSQL та перевіркою готовності бази даних перед стартом API-сервера.

10. Проведено техніко-економічне обґрунтування розробки (додаток Б). Загальна кошторисна вартість склала 16 043,0 грн. Розраховано, що завдяки автоматичному розпізнаванню за фотографією час введення одного прийому їжі скорочується у 4 рази.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		53

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Chollet F. Deep Learning with Python / François Chollet. — Shelter Island: Manning Publications, 2021. — 504 p.
2. Tan M., Le Q. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks / Mingxing Tan, Quoc V. Le // Proceedings of the 36th International Conference on Machine Learning. — 2019. — P. 6105–6114.
3. Bossard L., Guillaumin M., Van Gool L. Food-101 – Mining Discriminative Components with Random Forests / Lukas Bossard, Matthieu Guillaumin, Luc Van Gool // European Conference on Computer Vision. — 2014. — P. 446–461.
4. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems [Електронний ресурс]. — Режим доступу: <https://www.tensorflow.org>. — Дата звернення: 10.04.2025.
5. FastAPI — Modern, Fast Web Framework for Building APIs [Електронний ресурс]. — Режим доступу: <https://fastapi.tiangolo.com>. — Дата звернення: 10.04.2025.
6. React Native — Learn once, write anywhere [Електронний ресурс]. — Режим доступу: <https://reactnative.dev>. — Дата звернення: 10.04.2025.
7. Expo Documentation [Електронний ресурс]. — Режим доступу: <https://docs.expo.dev>. — Дата звернення: 15.04.2025.
8. SQLAlchemy — The Database Toolkit for Python [Електронний ресурс]. — Режим доступу: <https://www.sqlalchemy.org>. — Дата звернення: 10.04.2025.
9. PostgreSQL: The World's Most Advanced Open Source Relational Database [Електронний ресурс]. — Режим доступу: <https://www.postgresql.org>. — Дата звернення: 10.04.2025.
10. OpenCV: Open Source Computer Vision Library [Електронний ресурс]. — Режим доступу: <https://opencv.org>. — Дата звернення: 12.04.2025.
11. Docker Documentation [Електронний ресурс]. — Режим доступу: <https://docs.docker.com>. — Дата звернення: 15.04.2025.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		54

12. Kaggle: Your Machine Learning and Data Science Community [Електронний ресурс]. — Режим доступу: <https://www.kaggle.com>. — Дата звернення: 01.03.2025.

13. Goodfellow I. Deep Learning / Ian Goodfellow, Yoshua Bengio, Aaron Courville. — Cambridge: MIT Press, 2016. — 800 p.

14. Howard A., Sandler M. et al. Searching for MobileNetV3 / Andrew Howard, Mark Sandler et al. // IEEE/CVF International Conference on Computer Vision. — 2019. — P. 1314–1324.

15. He K. et al. Deep Residual Learning for Image Recognition / Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun // IEEE Conference on Computer Vision and Pattern Recognition. — 2016. — P. 770–778.

16. Pydantic Documentation [Електронний ресурс]. — Режим доступу: <https://docs.pydantic.dev>. — Дата звернення: 10.04.2025.

17. Bradski G. The OpenCV Library / Gary Bradski // Dr. Dobb's Journal of Software Tools. — 2000. — Vol. 25. — P. 120–125.

18. Axios — Promise based HTTP client for the browser and Node.js [Електронний ресурс]. — Режим доступу: <https://axios-http.com>. — Дата звернення: 20.04.2025.

19. React Navigation Documentation [Електронний ресурс]. — Режим доступу: <https://reactnavigation.org>. — Дата звернення: 20.04.2025.

20. Python Software Foundation. Python Language Reference, version 3.11 [Електронний ресурс]. — Режим доступу: <https://www.python.org>. — Дата звернення: 10.04.2025.

21. Keras Documentation [Електронний ресурс]. — Режим доступу: <https://keras.io>. — Дата звернення: 10.04.2025.

22. JSON Web Token Introduction [Електронний ресурс]. — Режим доступу: <https://jwt.io/introduction>. — Дата звернення: 12.04.2025.

23. passlib — Password Hashing Library for Python [Електронний ресурс]. — Режим доступу: <https://passlib.readthedocs.io>. — Дата звернення: 12.04.2025.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		55

24. ВООЗ. Звіт про стан здоров'я у Європейському регіоні ВООЗ 2018 [Електронний ресурс]. — Режим доступу: <https://www.euro.who.int>. — Дата звернення: 05.03.2025.

25. Шутіков В. М. Методи комп'ютерного зору для класифікації зображень / В. М. Шутіков // Вісник НТУУ «КПІ». Серія: Інформатика, управління та обчислювальна техніка. — 2022. — № 74. — С. 45–52.

26. Литвин В. В. Методи глибокого навчання в задачах розпізнавання об'єктів / В. В. Литвин, Н. Б. Шаховська // Матеріали X Міжнародної конференції з комп'ютерних наук та інформаційних технологій. — Львів, 2021. — С. 112–117.

27. Mifflin M. D., St. Jeor S. T. A new predictive equation for resting energy expenditure in healthy individuals / M. D. Mifflin, S. T. St Jeor // American Journal of Clinical Nutrition. — 1990. — Vol. 51, № 2. — P. 241–247.

28. Uvicorn — ASGI Web Server Implementation for Python [Електронний ресурс]. — Режим доступу: <https://www.uvicorn.org>. — Дата звернення: 15.04.2025.

29. TypeScript Documentation [Електронний ресурс]. — Режим доступу: <https://www.typescriptlang.org>. — Дата звернення: 20.04.2025.

30. Expo Secure Store API [Електронний ресурс]. — Режим доступу: <https://docs.expo.dev/versions/latest/sdk/securestore>. — Дата звернення: 20.04.2025.

					КР.КІ.10659264/22.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		56