НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.89:664](043)

«До захисту допущено»
В.о. завідувача кафедри
_____ Едуард ЖАРІКОВ
«___»_____2021 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Програмне та алгоритмічне забезпечення для інтелектуального визначення та аналізу складу продуктів»

Виконав (-ла):
студент (-ка) ІІ курсу, групи ІТ-з03мп
Козак Олена Сергіївна _____

Керівник:
проф. д.т.н., проф., засл. діяч
Синєглазов Віктор Михайлович _____

Рецензент:
начальник управління СПІНМД МВС України,
д.т.н., академік ІРА
Дорогий Ярослав Юрійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2021 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних систем»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Едуард ЖАРІКОВ

«___»_____2021р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Козак Олені Сергіївні

1. Тема дисертації «Програмне та алгоритмічне забезпечення для інтелектуального виявлення та аналізу складу продуктів», науковий керівник дисертації Синєглазов Віктор Михайлович проф. д.т.н., проф., засл. діяч, затверджені наказом по університету від «25» жовтня 2021 р. № 3575-с

2. Термін подання студентом дисертації «6» грудня 2021 р.

3. Об'єкт дослідження – програмне та алгоритмічне забезпечення для інтелектуального визначення та аналізу складу продуктів.

4. Предмет дослідження – алгоритми та методи визначення та розпізнавання текстової інформації на зображеннях етикеток харчових продуктів.

5. Перелік завдань, які потрібно розробити – провести аналіз проблеми та існуючих рішень даної проблеми та наявних аналогів; створити та реалізувати алгоритми та методи, які, не вдаючись до використання хмарних обчислень, розв'язують задачі: детекції тексту, видалення нетекстових областей, вирівнювання тексту, групування в слова та рядки, розпізнавання символів на зображеннях; розробити інтелектуальну систему для оцінки шкідливості складу; розробити бібліотеку для розпізнавання тексту на зображеннях для мобільних систем; дослідити ефективність розробленої програмної бібліотеки та проаналізувати результати роботи системи.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу: діаграма використання системи, схема структурна роботи системи, схема структурна бази даних.
7. Орієнтовний перелік публікацій: Sineglazov V.M., Kozak O.S., «Intelligent system for assessing the harmfulness of food products based on the processing of textual and graphic information» // Матеріали до XX Міжнародної науково-технічної конференції «Штучний інтелект та інтелектуальні системи» (Київ, 2021 р.);
Sineglazov V.M., Kozak O.S., «System for detectiong and analyzing textual information of product composition» // «Electronics and Control Systems» N 2(68): 18-25, 2021;
Sineglazov V.M., Kozak O.S., «Intelligent system for determining the harmfulness of food composition» // Матеріали першої всеукраїнської науково-практичної конференції молодих вчених та студентів «Інженерія програмного забезпечення та і передові інформаційні технології» (SoftTech-2021) – м. Київ. НТУУ «КПІ ім. І. Сікорського», 22-26 листопада.

8. Консультанти розділів дисертації

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

9. Дата видачі завдання «30» вересня 2020 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Проаналізувати вимоги до програмного забезпечення інтелектуального визначення складу продуктів	09.09.2021	
2	Дослідити існуючі методи створення програмного та алгоритмічного забезпечення для інтелектуальної системи визначення та аналізу складу продуктів	10.09.2021	
3	Розробити алгоритмічне забезпечення для виявлення текстових регіонів	17.09.2021	
4	Підібрати архітектуру нейронної мережі та навчити її на згенерованому навчальному наборі	30.09.2021	
5	Проєктування програмного забезпечення для системи інтелектуального визначення та аналізу складу продуктів	16.10.2021	
6	Розробка програмного забезпечення	29.10.21	
7	Виконання експериментальних досліджень	10.11.2021	
8	Оформлення пояснювальної записки	12.11.2021	
9	Подання дисертації на попередній захист	22.11.2021	
10	Подання дисертації на захист	6.12.2021	

Студент Олена КОЗАК

Науковий керівник Віктор СИНЄГЛАЗОВ

РЕФЕРАТ

До магістерської дисертації Козак Олени Сергіївни на тему: «Програмне та алгоритмічне забезпечення для інтелектуального визначення та аналізу складу продуктів».

Актуальність теми: Сьогодні харчові продукти можуть містити велику кількість хімічних інгредієнтів, які є шкідливими для здоров'я людей, складають індивідуальну загрозу, виступають алергенами, або належать до непереносимостей. Більшість людей нехтують вивченням складу продуктів, через відсутність розуміння, небажання чи брак часу, а тому споживачі потребують автоматизованого рішення, яке дозволяє, при необхідності, швидко і на місці надати оцінку продукту. Відтак, створення інтелектуальної системи, яка може виявляти текстову інформацію складу на зображеннях та аналізувати її, сьогодні є дуже актуальною та соціально значущою.

Наразі існує два підходи до виявлення текстової інформації на зображеннях мобільними пристроями: серверні обчислення та локальні обчислення на мобільних пристроях. Перший підхід дозволяє досягти великої продуктивності, але має проблему масштабованості та потребує підключення до мережі. Існуючі рішення, які відповідають другому підходу не підтримують кирилицю або є надто ресурсовитратними для мобільних пристроїв, а тому працюють повільно. Отже, постає необхідність створення власного механізму розпізнавання тексту, що наділений достатньою швидкодією, щоб працювати на мобільних пристроях, та має підтримку обробки українського тексту.

Мета дослідження: підвищення швидкості визначення складу продуктів харчування шляхом створення програмного та алгоритмічного забезпечення для інтелектуального виявлення та аналізу складу продуктів; підвищення швидкості розпізнавання тексту мобільними пристроями.

Для реалізації поставленої мети були сформульовані та вирішені наступні **завдання**:

- проаналізувати існуючі методи для інтелектуального визначення тексту на зображеннях живих сцен;
- створити та реалізувати алгоритми та методи, які, не вдаючись до використання хмарних обчислень, розв'язують задачі: детекції тексту, видалення нетекстових областей, вирівнювання тексту, групування в слова та рядки, розпізнавання символів на зображеннях;
- запропонувати архітектурне рішення, яка забезпечує систему можливістю обміну даними з зовнішнім хмарним сховищем для систем, які містять автономні компоненти;
- розробити інтелектуальну систему для оцінки шкідливості складу;
- розробити бібліотеку для розпізнавання тексту на зображеннях живих сцен для мобільних систем мовою програмування Java;
- дослідити ефективність програмної бібліотеки та проаналізувати результати роботи системи.

Об’єкт дослідження: програмне та алгоритмічне забезпечення для інтелектуального визначення та аналізу складу продуктів.

Предмет дослідження: алгоритми та методи визначення та розпізнавання текстової інформації на зображеннях етикеток харчових продуктів.

Наукова новизна. Найбільш суттєвими науковими результатами магістерської дисертації є:

- запропоновано алгоритмічне забезпечення інтелектуального визначення та аналізу складу продуктів харчування, яке включає алгоритм виявлення контрастних областей, алгоритм видалення нетекстових регіонів, алгоритм групування слів, алгоритм підготовки даних для розпізнавання, а також використання апарату згорткових нейронних мереж, що в результаті дозволяє зменшити час вивчення складу продуктів в середньому на 80%;

- розроблено програмну бібліотеку для розпізнавання тексту на зображеннях з підтримкою обробки українського тексту для пристроїв з обмеженими обчислювальними ресурсами, яка, завдяки використанню гібридного підходу, тобто, використанню алгоритмічного забезпечення та апарату штучних нейронних мереж, дозволяє зменшити час розпізнавання тексту в середньому на 75%, в порівнянні з існуючим рішенням, а також робить систему незалежною від мережі Інтернет;
- адаптовано клієнт-серверну архітектуру для побудови інтелектуальних систем, яке дає можливість оновлення інтелектуального програмного модуля, за рахунок наявності зовнішнього репозиторію для моделей нейронних мереж, а також можливість обміну даними з зовнішнім хмарним середовищем без постійного доступу до сервера.

Практична цінність отриманих результатів: розроблена інтелектуальна система може використовуватись індивідуальними користувачами для отримання персонального аналізу шкідливості харчових продуктів; розроблена програмна бібліотека може використовуватись розробниками для розпізнавання текстової інформації без необхідності використання системою серверних обчислень.

Методи дослідження застосовані у даній роботі, базуються апараті штучних нейронних мереж та методах оптимізації навчання.

Зв'язок роботи з науковими програмами, планами, темами: робота виконувалась на кафедрі Інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут ім. Ігоря Сікорського» в рамках теми «Програмне та алгоритмічне забезпечення для інтелектуального визначення та аналізу складу продуктів».

Апробація: основні положення роботи доповідались і обговорювались на XX Міжнародній науково-технічній конференції «Штучний інтелект та інтелектуальні системи» Artificial intelligence and intelligent systems (AIIS'2021) (Київ, 2021 р.), а також на Першій Всеукраїнській науково-практичній конференції молодих вчених

та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2021).

Публікації:

1) Sineglazov V.M., Kozak O.S. «Intelligent system for determining the harmfulness of food composition» / Sineglazov V.M., O.S. Kozak // Перша Всеукраїнська науково-практична конференція молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2021). Секція кафедри інформатики та програмної інженерії. Матеріали конференції. – Київ. – 2021. 22–26 листопада 2021р. – С.124-127;
2) Sineglazov V.M., Kozak O.S. «System for detectiong and analyzing textual information of product composition» / V.M. Sineglazov, O.S. Kozak // «Electronics and Control Systems» N 2(68): 2021 – pp. 18-25;
3) Sineglazov V.M., Kozak O.S. «Intelligent system for assessing the harmfulness of food products based on the processing of textual and graphic information» / Sineglazov V.M., Kozak O.S. // XX Міжнародна науково-технічна конференція «Штучний інтелект та інтелектуальні системи». Матеріали конференції. – Київ. – 2021. 27 листопада 2021 р.

Магістерська дисертація складається зі вступу, 5 розділів, висновку та додатків, містить 43 рисунки, 28 таблиць та 95 джерел. Повний обсяг магістерської дисертації складає 158 сторінок, з яких додатки – 29 сторінок.

ABSTRACT

Actuality. Today, food can contain hundreds of harmful to human health chemical ingredients. They pose an individual threat, act as allergens, or are intolerance. Most people neglect studying the composition of food products due to reluctance, lack of time or understanding. Therefore consumers need an automated solution that allows, if necessary, quick and on the spot evaluating of the product. For this reason, the creation of an intelligent system that can detect textual composition in images and analyze it is important and socially significant today.

There are currently two approaches to detecting textual information on images by mobile devices: server computing and local computing on mobile devices. The first approach allows you to achieve high performance but has a problem of scalability and requires a stable network connection. Existing solutions that fit the second approach do not support the Cyrillic alphabet or are too resource-intensive for mobile devices, and therefore work slowly. Thus, there is a need to create our own text recognition mechanism, endowed with sufficient speed to work on mobile devices, and has Ukrainian text processing capabilities.

The aims of the study are to introduce the algorithms and software for intelligent detection and analysis of product composition; to increase the speed of text recognition by mobile devices by developing a library for text extraction.

It is necessary to perform the following tasks to achieve these **goals:**

- analyze existing methods for an intelligent recognition of text in scene images;
- create and implement algorithms and methods that, without resorting to the use of cloud computing, solve problems: text detection, deleting non-text areas, text alignment, grouping into words and lines, character recognition in images;
- develop an architectural solution that provides the system with the ability to exchange data with external cloud storage for systems that contain autonomous components;
- develop an intelligent system for assessing the harmfulness of the composition;

- develop a library for text recognition in scene images for mobile systems in the Java programming language;
- investigate the effectiveness of the software library and analyze the results of the system.

Object of study. Algorithms and software for intelligent determination and analysis of product composition. **The subject of the research** is algorithms and methods of detection and recognition of textual information and images.

Scientific novelty of the obtained results. The most significant scientific results of the master's thesis are:

- an algorithmic software for intelligent determination and analysis of food composition was proposed, which includes algorithm for detecting contrasting regions, algorithm for removing non-text regions, word grouping algorithm, algorithm of data preparation for recognition, as well as the use of convolutional neural networks, which reduce the time of study products on average by 80%;
- a software library for text recognition on images with Ukrainian text processing capabilities for devices with limited computing resources was developed, which, thanks to the use of a hybrid approach, ie, the use of algorithmic software and artificial neural networks, can reduce the time of text recognition by 75% compared to existing solution, and makes the system independent of the Internet;
- adapted client-server architectural solution for building intelligent systems, which allows updating the intelligent software module due to the presence of an external repository for neural network models, as well as the ability to exchange data with the external cloud environment without permanent access to the server.

The practical significance of the results. The developed intelligent system can be used by individual users to obtain a personal analysis of the harmfulness of food; the developed software library can be used by developers to recognize textual information without the need to use a server computing system.

Research methods used in this work are based on the apparatus of artificial neural networks and methods for optimizing learning.

Relationship with working with scientific programs, plans, topics. The work was performed at the Department of Informatics and Software Engineering of the Faculty of Computer Engineering of The National Technical University of Ukraine «Igor Sikorsky Kyiv Polytechnic Institute» within the work «Algorithms and Software for Intelligent Determination and Analysis of Product Composition».

Testing: the main provisions of the work were reported and discussed at the the First All-Ukrainian scientific-practical conference of young scientists and students «Software engineering and advanced information technology» (SoftTech-2021) and at the XX International Scientific and Technical Conference «Artificial Intelligence and Intelligent Systems» (Kyiw, 2021).

Publications.

1) Sineglazov V.M., Kozak O.S. «Intelligent system for determining the harmfulness of food composition» / Sineglazov V.M., O.S. Kozak // The first All-Ukrainian scientific-practical conference of young scientists and students “Software engineering and advanced information technology” (SoftTech-2021). Section of the Department of Informatics and Software Engineering. Conference proceedings. – Kyiw. – 2021. the 22–26 of November 2021p. – C.124-127;
2) Sineglazov V.M., Kozak O.S. «System for detectiong and analyzing textual information of product composition» / V.M. Sineglazov, O.S. Kozak // «Electronics and Control Systems» N 2(68). – Kyiw, 2021 – pp. 18-25. – doi:10.18372/1990-5548.68.16087;
3) Sineglazov V.M., Kozak O.S. «Intelligent system for assessing the harmfulness of food products based on the processing of textual and graphic information» / Sineglazov V.M., Kozak O.S. // XX International Scientific and Technical Conference «Artificial intelligence and intellectual systems». Conference proceedings. – Kyiw. – 2021. the 27 of November 2021 p.

The master's dissertation consists of an introduction, 5 chapters, conclusion and appendices, contains 43 figures, 28 tables and 95 sources. The full volume of the master's dissertation is 158 pages, of which the appendices are 29 pages.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 13

ВСТУП 14

1 АНАЛІЗ ПРОБЛЕМИ ВИЗНАЧЕННЯ СКЛАДУ ПРОДУКТІВ 16

1.1 Проблема визначення складу харчової продукції за друкованою інформацією виробника 16

1.2 Державні стандарти та форми представлення текстової інформації на маркуваннях харчових продуктів 18

1.3 Аналіз існуючого програмного забезпечення для виявлення та аналізу складу продуктів 20

1.4 Постановка вимог до системи для виявлення та аналізу текстової інформації складу продуктів та обґрунтування її створення 23

Висновки до розділу 25

2 ДОСЛІДЖЕННЯ ІНТЕЛЕКТУАЛЬНИХ МЕТОДІВ ОБРОБКИ ТЕКСТОВОЇ ІНФОРМАЦІЇ 26

2.1 Постановка задачі обробки візуальної інформації 26

2.2 Аналіз методів вирішення поставлених задач 28

2.2.2 Методи детекції тексту на зображеннях зі складом 28

2.2.3 Методи розпізнавання тексту на зображеннях зі складом 30

2.3 Згорткові нейронні мережі *31*

2.3.1 Архітектура згорткової нейронної мережі 34

2.3.2 Навчання згорткової нейронної мережі 36

Висновки до розділу *41*

3 АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ 42

3.1 Структура алгоритмічного забезпечення 42

3.2 Алгоритм MSER для виділення текстових регіонів 42

3.3 Алгоритм для видалення нетекстових регіонів 48

3.2.1 Алгоритм визначення ширини штрихових лінії та евристична фільтрація 49

3.2.2 Алгоритм для видалення дублікатів та областей, створених внутрішньою частиною інших регіонів використанням метрик перетину над об'єднанням 53

3.3 Розпізнавання виявлених регіонів символів 57

3.4 Алгоритм складання слів 59

3.5 Алгоритм аналізу та оцінки шкідливості харчових продуктів за його складом 64

Висновки до розділу 68

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 69

4.1 Архітектура інтелектуальної системи визначення та оцінки шкідливості складу 69

4.2 Структура розробленої інтелектуальної системи 73

4.2.1 Рівень представлення 73

4.2.2 Рівень бізнес-логіки та платформи 75

4.2.3 Рівень даних 76

4.2.3.1 Аналіз предметної області 77
4.2.3.2 Інфологічна та даталогічна моделі 78
4.2.4 Рівень інфраструктури 79
4.4 Програмні засоби для реалізації методів інтелектуального виявлення тексту 82
4.5 Дослідження ефективності 85
4.5.1 Апаратне забезпечення для проведення тестування 85
4.5.3 Визначення точності обробки тексту 86
4.5 Інтерфейс користувача інтелектуальної системи «FoodUnderstood» 87
Висновки до розділу *92*
5 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ 93
5.2 Технологічний аудит ідеї проекту 96
5.3 Аналіз ринкових можливостей запуску стартап-проекту 97
5.4 Розроблення ринкової стратегії проекту 106
5.5 Розроблення маркетингової програми стартап-проєкту 110
Висновок до розділу 115
ВИСНОВКИ 116
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 118
ДОДАТОК А 130
ДОДАТОК Б 131
ДОДАТОК В 132
ДОДАТОК Б 133
ДОДАТОК Ж 138

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ЗНМ – згорткова нейронна мережа

CNN (convolutional neural network) – згорткова нейронна мережа

MSER (maximaly stable extremal region) – максимально стабільний екстремальний регіон

IoU (intersection over union) – перетин над об’єднанням

MVVM – Model View ViewModel

ANN (artificial neural networks) – штучні нейронні мережі

ML – machine learning

ПЗ – програмне забезпечення

СУБД – системи управління базами даних

ВСТУП

На сьогодні, велика увага приділяється якості харчових продуктів. Це в першу чергу пов’язано із зацікавленістю компаній знизити свої витрати шляхом підміни складових готових продуктів з метою зменшення витрат, запобігання псуванню продуктів додаванням хімічних харчових домішок, які є небезпечними для здоров’я людей, а також використанням інгредієнтів, які можуть бути алергенами. Більшість людей нехтують вивченням складу продуктів, через відсутність розуміння, небажання чи брак часу, а тому споживачі потребують автоматизованого рішення, яке дозволяє, при необхідності, швидко і на місці надати оцінку продукту. Відтак, створення інтелектуальної системи, яка може виявляти текстову інформацію складу на зображеннях та аналізувати її, сьогодні є дуже актуальною та соціально значущою.

Така система, з одного боку, повинна бути мобільною та займати небагато місця, а з іншого боку, вона повинна бути спроможна обробляти великі обсяги інформації, в тому числі текстово-графічну інформацію на матриці зображення. З точки зору технічної реалізації таким вимогам відповідає мобільний телефон. З іншого боку система повинна бути інтелектуальною (реалізованою з допомогою використання штучних нейронних мереж, які своєю чергою висувають вимоги до параметрів телефону, його операційної системи).

Наразі існує два підходи до виявлення текстової інформації на зображеннях мобільними пристроями: серверні обчислення та локальні обчислення на мобільних пристроях. Перший підхід дозволяє досягти великої продуктивності, але має проблему масштабованості та потребує підключення до мережі. Існуючі рішення, які відповідають другому підходу не підтримують кирилицю або є надто ресурсовитратними для мобільних пристроїв, а тому працюють повільно. Отже, постає необхідність створення власного механізму розпізнавання тексту, що наділений достатньою швидкодією, щоб працювати на мобільних пристроях, та має підтримку обробки українського тексту.

Мета дослідження: підвищення швидкості визначення складу продуктів харчування шляхом створення програмного та алгоритмічного забезпечення для інтелектуального виявлення та аналізу складу продуктів; підвищення швидкості розпізнавання тексту мобільними пристроями.

Для реалізації поставленої мети були сформульовані та вирішені наступні **завдання**:

- проаналізувати існуючі методи для інтелектуального визначення тексту на зображеннях живих сцен;
- створити та реалізувати алгоритми та методи, які, не вдаючись до використання хмарних обчислень, розв'язують задачі: детекції тексту, видалення нетекстових областей, вирівнювання тексту, групування в слова та рядки, розпізнавання символів на зображеннях;
- запропонувати архітектурне рішення, яке забезпечує систему можливістю обміну даними з зовнішнім хмарним сховищем для систем, які містять автономні компоненти;
- розробити інтелектуальну систему для оцінки шкідливості складу, для якої створити бібліотеку для розпізнавання тексту на зображеннях живих сцен для мобільних систем;
- дослідити ефективність програмної бібліотеки та проаналізувати результати роботи системи;

Об'єкт дослідження: програмне та алгоритмічне забезпечення для інтелектуального визначення та аналізу складу продуктів.

Предмет дослідження: алгоритми та методи визначення та розпізнавання текстової інформації на зображеннях етикеток харчових продуктів.

Магістерська дисертація складається зі вступу, 5 розділів, висновку та додатків, містить 43 рисунки, 28 таблиць та 95 джерел. Повний обсяг магістерської дисертації складає 158 сторінок, з яких додатки – 29 сторінок.

1 АНАЛІЗ ПРОБЛЕМИ ВИЗНАЧЕННЯ СКЛАДУ ПРОДУКТІВ

1.1 Проблема визначення складу харчової продукції за друкованою інформацією виробника

Сьогодні харчові продукти мають складну структуру та велику кількість інгредієнтів, різного хімічного складу. Від властивостей інгредієнтів складу харчових продуктів залежить якість всього харчового продукту. Велику частину шкідливих для здоров'я інгредієнтів складають отримані шляхом хімічного синтезу синтетичні нехарчові речовини – харчові домішки, які є природними або синтезованими речовинами, які спеціально додаються та вводяться до харчових продуктів за технологічною доцільністю [90].

Сучасний розвиток харчової та хімічної промисловості дає можливість вводити харчові домішки в склад харчові продукти не лише за технологічної потреби, як от подовження терміну зберігання, але й для регулювання смаку та кольору продукту, покращення зовнішнього вигляду, здешевлення продукту. Через розвиток харчових технологій кількість харчових домішок зросла до 2800 одиниць [85].

Консерванти, стабілізатори та емульгатори, ароматизатори вживаються щодня та накопичуються в організмі людини протягом великого періоду життя. Всесвітня організація охорони здоров'я зазначає, що, наприклад, ароматизатори в неправильній кількості викликають гіперчутливість до певних продуктів [86]. Підсолоджувачі, такі як цикламати (Е952), аспартам (Е951), сахарин (Е954) мають допустиму кількість вживання на добу та не рекомендуються для споживання протягом довгого періоду [91]. Також, відома пальмова олія, яка і досі використовується в харчових продуктах (маргарин, певні недорогі сири, шоколад, шоколадні солодощі та ін.) через велику кількість насичених жирних кислот спричиняє захворювання серцево-судинної системи, та використовується задля здешевленя продукції [60, 94]. Про шкоду він насичених жирів, які містяться в пальмовій олії, а також в м'ясі та молочній продукції зазначається у доповіді,

представленій Всесвітньою організацією охорони здоров'я [19]. Соєвий лецитин (Е471), який є поширеним емульгатором та стабілізатором, та який використовується навіть у хлібі, протипоказаний людям, які мають проблеми з печінкою [22].

На жаль, не лише харчові домішки є причиною проблем зі здоров'ям – велика кількість людей страждає на харчові алергії. Згідно з інформаційною запискою про харчові алергії Всесвітньої організації охорони здоров'я, харчові алергії можуть викликати понад 70 продуктів, а єдиним способом запобігання алергіям є уникнення вживання продуктів алергенів, серед яких знайомі кожному зернові продукти, які містять глютен, молочні продукти, какао, арахіс та інші види горіхів, яйця та риба, гірчиця, селера та ін. [83]. Алергії проявляються у вигляді шкірних захворювань, розладів шлунково-кишкового тракту, проблем з диханням тощо.

Іншою проблемою багатьох людей є непереносимість лактози – нездатність організму до перетравлення молочного цукру лактози, який міститься у всіх, без виключення, молочних продуктах, хлібі, виробах з борошна, шинці та ковбасах, солодощах, шоколаді та какао, заправках для салатів та кетчупі тощо. За показниками Міністерства охорони здоров'я понад 75%, населення страждають від даної непереносимості [95, 57]. Єдиним виходом для людей з непереносимістю лактози є повна відмова від продуктів, які мають у своєму складі молочні продукти, зокрема сухе молоко.

Розглянемо склад звичайного твердого сиру, який має середню ціну та одразу помітимо такі інгредієнти як: агент твердіння хлорид кальцію (спричиняє хвороби кишківника та виразки), барвник екстракт аннато (спричиняє харчові алергії [3]), калій азотнокислий, або інша назва нітрат калію, який також використовується у виробництві скла та ракетного палива (середня токсичність, спричиняє головні болі та запаморочення, запалення нирок та проблеми з диханням у людей, які страждають на астму [21]), хімозин (може викликати алергічні реакції, не рекомендований вагітним). З усього наведеного вище зрозуміло, що звичайний сир, популярної марки є шкідливим для життя.

З погляду на те, що сьогодні існує величезна кількість харчових домішок, які несуть потенційну загрозу здоров'ю людей, особливо, тих, у кого є алергії на певні інгредієнти, харчові обмеження, або проблеми зі здоров'ям, споживачі повинні бути обов'язково ознайомлені зі складом продуктів. Та попри це, результати опитувань, проведені серед європейців показали, що 52,2 % споживачів не читають та не звертають уваги на склад на упаковках товарів [34]. Серед споживачів, які перевіряють склад харчових продуктів, 57,7% споживачів з усього світу не розуміють склад продуктів, не знайомі з інгредієнтами, представленими у складі та не знають, для чого вони використовуються, тоді як 39,7% розуміють лише частково [34]. Складність розуміння та відсутність бажання читати склад продуктів спричиняють нововведені терміни, або харчові домішки, які є незнайомими та незрозумілими для людей.

Враховуючи той факт, що велика частина населення не читає склад продуктів взагалі, а та частина, яка читає, взагалі не розуміється на ньому, випливає необхідність забезпечення споживачів автоматизованим та мобільним інструментом, який буде надавати споживачеві оцінку шкідливості складу продуктів. Зважаючи на вимогу до мобільності, основним способом реалізації системи є створення програмного забезпечення у вигляді мобільної системи.

1.2 Державні стандарти та форми представлення текстової інформації на маркуваннях харчових продуктів

Етикетки та упаковки харчових продуктів є основними носіями інформації про товар, важливої для споживача. Згідно зі ст. 1 Закону Україні «Про інформацію для споживачів щодо харчових продуктів», маркуванням є будь-яка текстово-графічна інформація, представлена на етикетці виробника [90]. Основними вимогами до маркування є достатність та достовірність. Згідно із законодавством України виробники харчових продуктів зобов'язані вказувати повний перелік інгредієнтів (склад), а також наявні у складі харчові домішки, які використовувались при виробництві або під час приготування харчового продукту, на упаковці своїх

продуктів [90]. Ці стандарти щодо маркування харчових продуктів представленні у Кодексі Аліментаріус, який визначає загальноприйняті стандарти на маркування фасованих продуктів, впроваджені в багатьох країнах, в тому числі й в Україні, та зобов'язує виробників продуктів вказувати достовірну інформацію про перелік інгредієнтів, наявність харчових домішок та іншої інформації на упаковках продуктів [84].

Маркування на етикетках харчових продуктів представлене у текстово-графічному вигляді. Як видно на рисунку 1.1, друкована інформація має блочну форму представлення інформації на етикетці, яка включає блок зі складом, блок з харчовою цінністю, блок маси продукту, блок умов зберігання, блок зі штрих-кодом та ін.

Рисунок 1.1 – Етикетка твердого сиру «Голандія»

Найважливішу інформацію, яка дозволяє споживачу робити аналіз продукту несе блок складу харчової продукції, а саме інгредієнти, перераховані після слова «Склад» або «Інгредієнти», розташовані у верхній частині етикетки.

Форма представлення інформації для обробки мобільною системою, у вигляді текстового блоку на зображені, обумовлює створення інтелектуальної системи, яка обробляє етикетку харчового продукту, виявляє на ньому інформацію зі складом та аналізує цю інформацію.

1.3 Аналіз існуючого програмного забезпечення для виявлення та аналізу складу продуктів

Проведемо аналіз подібних існуючих автоматизованих систем оцінки шкідливості складу продуктів. За останні роки технології мобільного здоров'я (mHealth), включаючи програмне забезпечення для мобільних телефонів для лікування захворювань, пов'язаних з їжею [71], таких як харчова алергія та непереносимість, зросли у попиті у всьому світі. Зручності додатків в управлінні охороною здоров'я віддають перевагу приблизно 59% населення світу [32]. Додатки, які дозволяють споживачам визначати харчові алергени в продуктах і продуктах, є доступними, хоча більшість із них пропонують нерелевантний вміст або є незручними у користуванні [14], оскільки більшість мобільних систем представлені у вигляді репозиторіїв, які містять інформацію про продукт, яку можна знайти за його штрих-кодом [28], серед яких, наприклад, «Натурометр», «Open Food Facts».

В основі принципу роботи додатку «Натурометр» лежить сканування штрих-коду камерою телефону, в результаті чого надається перелік інгредієнтів, які входять до складу. Додаток призначений лише для російського ринку продуктів, що видно по відсутності українських продуктів в базі даних. Розробники заохочують самостійне додавання продуктів разом з наявними складниками для створення «народної» бази даних продуктів. Вигляд додатку представлений на рис. 1.2.

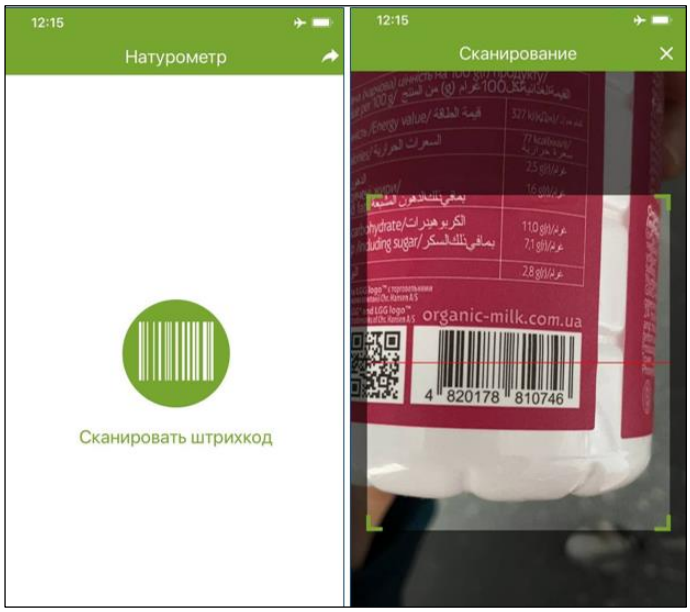


Рисунок 1.2 – «Натурометр»

Принцип його роботи додатку «Open Food Facts» (рис. 1.3) базується на скануванні штрих-коду продуктів. «Open Food Facts» є відкритою базою даних продуктів, тобто, за відсутності продукту користувач додає його в базу [59]. Застосунок «Open Food Facts» надає перелік інгредієнтів, присутніх у складі разом із визначеною оцінкою, а відтак, отримання персоналізованої оцінки є неможливим.

Рисунок 1.3 – «Open Food Facts»

Існуючі системи, що надають функціонал розпізнавання складу продукту та проводять оцінку шкідливості, мають суттєву відмінність реалізації у порівнянні з

розробленим програмним та алгоритмічним комплексом. Перш за все, їх підхід заснований на пошуку та читанні штрих-кодів, що розміщені на етикетках товарів.

Такий підхід є підвладний переліку вад:

- ідентифікатор, що нанесений на штрих-коді, має обов'язково міститись у базі даних, інакше сканування буде завершене невдачею;
- сховище даних для рішення завдання такого роду має містити записи для мільйонів товарів що можна знайти на полицях магазинів, що лишає слід на об'ємі спожитої пам'яті дискового простору.

Очевидно така база даних не може бути розміщена на мобільному пристрої кожного користувача, через обмежені ресурси таких платформ. А також сховище даних такого масштабу неодмінно вимагає постійного адміністрування (видалення застарілих продуктів, внесення нових, редагування існуючих). Сукупність цих чинників примушують розробників вдаватись до застосування клієнт-серверної архітектури програмного забезпечення. Що, і собі, додає перелік слабких місць рішення: наявність єдиного вузького місця у системі у вигляді серверу, а також залежність користувачів від наявності швидкого та стабільного з'єднання з мережею Інтернет.

Саме тому було вирішено запропонувати програмне рішення, яке не полягається на штрих-коди етикеток, а, натомість, розпізнає текст та позначки рівно так, як це робить око людини. Таким програмним рішенням є інтелектуальна система, яка, використовуючи апарат штучних нейронних мереж, обробляє текстову інформацію на зображені етикетки товару для подальшої роботи з нею.

Найпопулярнішим мобільним застосунком, який розпізнає текстову інформацію складу на зображенні є «Food Scanner» (рис. 1.4).

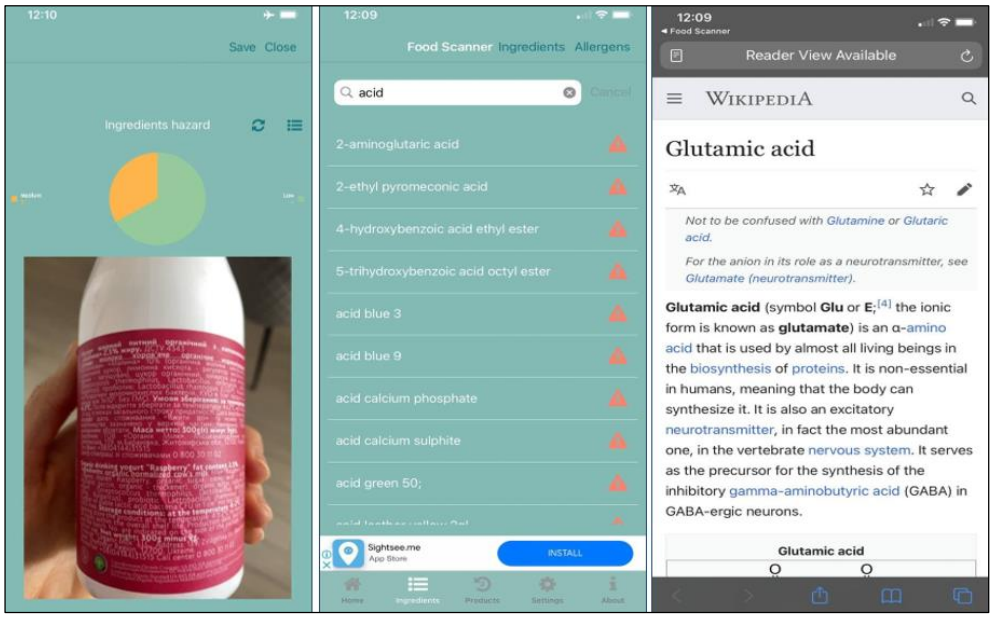


Рисунок 1.4 – «Food Scanner»

Мобільний застосунок «Food Scanner» є інтелектуальною системою та розпізнає склад продуктів англійською мовою; має ручний пошук продуктів з посиланнями на статті з вікіпедії. Недоліком такого додатку є незручний та не інтуїтивний інтерфейс, важкість розуміння оцінки шкідливості та описаних інгредієнтів, неможливість роботи при відсутності з'єднання з мережею.

Отже, провівши короткий огляд представлених вище додатків, можна зробити висновок, що наразі не існує зручної інтелектуальної системи, яка могла б розпізнавати склад продуктів, представлених на українському ринку, надавати характеристики продуктів та оцінку шкідливості продукту.

1.4 Постановка вимог до системи для виявлення та аналізу текстової інформації складу продуктів та обґрунтування її створення

З наведеного вище випливає необхідність створення системи, яка повинна:

1) бути швидкою, зручною та доступною для споживачів;
2) обробляти текстову інформацію на зображеннях зі складом продуктів та надаватиме користувачу перелік інгредієнтів;
3) надавати користувачеві можливість вводити свої особливі дані щодо бажаних або не бажаних інгредієнтів у складі (обмежень, алергенів);

4) мати компонент для аналізу шкідливості кожного інгредієнту, який на базі аналізу складу та з урахуванням обмежень користувача, зможе надавати оцінку про даний харчовий продукт;
5) мати велику базу даних продуктів;
6) мати можливість працювати за відсутності з'єднання з мережею;
7) мати зручний та інтуїтивний компонент представлення оцінки шкідливості та переліку всіх інгредієнтів разом з їх характеристиками.

Предметним середовищем розробленої мобільної системи виступають зображення, які містять текст складу продуктів. А відтак:

1) мобільний застосунок буде використовувати камеру телефону та робити чіткі зображення з текстом складу продукту (на прикладі рисунку, представленого нижче), щоб зменшити витрати часу на ручний пошук інгредієнтів та їх властивостей;
2) з попереднього пункту витікає те, що система повинна мати програмний модуль для обробки текстової інформації зображень, вилучення, перетворення текстової інформації на зображенні в текст, тобто, мати модуль для інтелектуального виявлення та розпізнавання тексту із зображення;
3) додаток містить компонент «особистий кабінет», в якому користувач додаватиме свої обмеження та інгредієнти-алергени;
4) додаток містить програмний модуль, який займається аналізом інгредієнтів, отриманих після розпізнавання, та, за їхніми характеристиками, надавати оцінку шкідливості всьому продукту;
5) додаток має компонент представлення отриманих результатів;
6) з попереднього пункту випливає:
   а) необхідність наявності розгорнутої бази даних продуктів, в якій будуть зберігатись всі представлені інгредієнти продуктів з їхніми характеристиками, описом;

b) необхідність створення та використання алгоритму підтримки прийняття рішень, який робитиме аналіз текстової інформації складу продуктів та, на основі характеристики кожного інгредієнту, даватиме загальну оцінку продукту харчування, тобто, оцінку шкідливості даного інгредієнту для конкретного споживача.

Висновки до розділу

У даному розділі було виявлено, що через наявність сьогодні великої кількості шкідливих інгредієнтів у складі продуктів, є висока необхідність у системі інтелектуального, персоналізованого аналізу складу продуктів, адже саме через відсутність автоматизації процесу безліч покупців ігнорує склад товарів та вживає шкідливі, або навіть загрозливі продукти харчування. На відміну від існуючих рішень, що надають відгук про продукт лише за наявності ідентифікатору у базі даних, що отримується скануванням штрих-коду, було прийнято рішення розробити систему, яка розпізнає саме текстову інформацію на зображеннях зі складом продуктів.

2 ДОСЛІДЖЕННЯ ІНТЕЛЕКТУАЛЬНИХ МЕТОДІВ ОБРОБКИ ТЕКСТОВОЇ ІНФОРМАЦІЇ

2.1 Постановка задачі обробки візуальної інформації

У ході створення структури програмного забезпечення для аналізу та оцінки складу харчових продуктів та її реалізації виникла задача створення незалежного перевикористовуваного програмного модуля, який може постачатись у вигляді окремої бібліотеки для швидкого розпізнавання тексту. Завдання створення такого програмного компонента обумовлено необхідністю вилучення тексту із зображення, зроблених з допомогою камери мобільного телефону. Даний програмний модуль буде вбудований в клієнтську частину мобільного додатку, а тому накладає обмеження на використання методів розпізнавання тексту такі, як швидкість роботи, обсяги пам’яті.

Виявлення тексту на зображеннях упаковок харчових продуктів, як і загалом на зображеннях в живих сценах (англ. scene images) відрізняється від виявлення тексту на зображеннях документів і є однією з найскладніших задач через велику кількість недоліків зображення таких як: якість, шуми, викривлення, малоконтрастний фон, нахил, відблиск, зашумленість та розтягнення, шрифт та розмір тексту на таких зображеннях (рис. 2.1) [45].

Рисунок 2.1 – Зображення зі складом продуктів

Вилучення та розпізнавання тексту на зображеннях упаковки зі складом харчового продукту для подальшого його аналізу реалізується шляхом вирішення наступних підзадач:

- на вхід програмному модулю розпізнавання тексту поступає багатоканальне зображення етикетки продукту харчування, яке повинне бути конвертована до одноканального та приведене до бінаризованого вигляду;
- необхідно використати математичні алгоритми вилучення множини прямокутників, які представляють текстові регіони, за характерними ознаками (наприклад, контрастність пікселів матриці, кутові точки тощо);
- вилучені регіони, які місять текстову інформацію, необхідно погрупувати у регіони слів та рядків;
- кожен регіон-кандидат символу з множини виявлених регіонів, який представлений матрицею пікселів, повинен оброблятись окремо, відтак, матричне представлення зображення символу необхідно інтелектуально розпізнати до текстового варіанту символу. Для цього необхідно підібрати структуру нейронної мережі, яка могла б швидко працювати на мобільному пристрої, створити навчальний сет зображень символів, навчити її та імпортувати в мобільний додаток;
- символи, які були розпізнанні, необхідно об’єднати у слова, що ми будемо робити базуючись на центрі маси символу в обмежувальному прямокутнику, його математичних характеристиках;
- у отриманих словах, з допомогою метрик Левенштейну, необхідно заповнити пропущенні або не розпізнані букви, або ж виправити помилки у словах, після чого на виході отримується текст переліку інгредієнтів харчового продукту, який надалі буде оброблятись алгоритмом підтримки прийняття рішень для оцінки шкідливості продукту.

2.2 Аналіз методів вирішення поставлених задач

Перш за все необхідна попередня обробка вхідного зображення: конвертація багатоканального (кольорового) зображення в одноканальне (чорно-біле) та бінаризація зображення.

Основною причиною, чому для методів вилучення тексту використовують одноканальне зображення замість безпосередньої роботи з кольоровими зображеннями, є те, що відтінки сірого дають можливість обробки лише одного каналу зображення, тому зменшують вимоги до обчислень. У нашому випадку колір не несе велику користь, та натомість, багатоканальні зображення несуть додаткову непотрібну інформацію, що збільшує кількість навчальних даних, необхідних для досягнення високої точності [43].

Поєднання аналізу одноканального зображення з аналізом бінарного зображення використовується через більш простий аналіз в останньому випадку [58]. Бінаризація процесом перетворення матриці зображення в двійкове з допомогою порогових підходів [77]. Кожен піксель зображення класифікується як білий для фону, а чорний для тексту. З допомогою бінаризації полегшується виявлення ознак символьних регіонів (рис. 2.2).

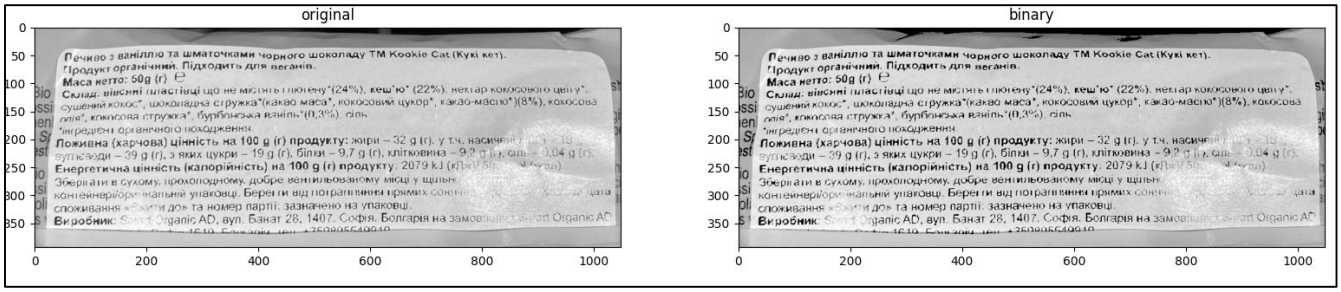


Рисунок 2.2 – Представлення бінаризованого зображення

2.2.2 Методи детекції тексту на зображеннях зі складом

Наступною задачею обробки зображень зі складом харчових продуктів є виявлення текстових регіонів-кандидатів. Вся складність виявлення тексту на таких зображеннях полягає у складному фоні, варіації шрифту, розміру, кольору та орієнтації тощо.

Існуючі алгоритми детекції тексту поділяються на дві групи: методи на основі регіонів та методи на основі зв’язних компонентів [82]. Методи на основі регіонів роблять аналіз текстури та на основі цього виявляють області з текстом [82], в той час, як методи на основі зв’язних компонентів виділяють символи-кандидати на основі виявлення країв або кластерного аналізу кольорів. Після чого нетекстові елементи фільтруються з допомогою підходящого евристичного методу або алгоритму [69]. Методи на основі регіонів повільні, оскільки зображення потрібно обробляти в декількох масштабах. Процес виконання цієї багаторазової операції, завдає збитків швидкості, особливо, якщо всі обчислення відбуваються на мобільному пристрої [69]. Відтак, оскільки наше програмне забезпечення є вбудованим у клієнтську частину систему обраний метод повинен бути швидким та маловитратним [81].

Аналізуючи підходи виявлення символів на етикетках харчових продуктів, було виявлено, що такі об’єкти зображення мають, здебільшого, однакову контрастність, інтенсивність, а отже об’єкти мають однакові ознаки для їх виявлення, немає необхідності використовувати методи на базі регіонів, які будуть повторно обробляти зображення та для швидкості вимагати великих обчислювальних витрат. А відтак, для розв’язання задачі вилучення тексту із зображення етикеток товарів було обрано групу методі на базі зв’язних компонентів [82].

Серед методів на базі зв’язних компонентів, наприклад, метод Такахаші та ін. витягує текстові області-компоненти за допомогою детектора краю Canny (англ. CannyEdge detector), після чого вилучені компоненти аналізуються з допомогою графу суміжності регіонів [68]. Метод, представлений Чжу та ін., працює з використанням нелінійного алгоритму локальної бінаризації для вилучення зв’язних компонентів на основі декількох типів виняткових рис компоненту, які включають особливі риси таких, як геометричні риси, риси контрастних країв, характеристики співрозміщення деталей, статистику штрихових особливостей, закономірності форми тощо [74].

Одним з найбільш підходящих методів для детекції тексту на зображеннях етикеток є метод виявлення максимально стабільних екстремальних регіонів (англ. MSER), представлений Матасом та ін.. Метод працює аналізує контрастність пікселів зображення для знаходження регіонів зображення, представлених у вигляді набору пікселів, які можна виявити з високою повторюваністю, оскільки мають певну відмінність (від нетекстових регіонів), інтенсивність та стійкість [52], які отримали назву екстремальних регіонів; на зображеннях етикеток харчових – текстові контрастні регіони (див. розділ 3.2).

2.2.3 Методи розпізнавання тексту на зображеннях зі складом

Наступною задачею, після отримання із загального зображення окремих зображень з символами, є розпізнавання літери на зображенні. Дана задача вирішується методами класифікації символів на зображенні. Сьогодні найпопулярнішими методами класифікації зображень є методи машинного навчання, зокрема штучні нейронні мережі та ін.

До методів машинного навчання належать метод k-найближчих сусідів (англ. k-Nearest Neighbor), метод випадковий ліс (англ. Random Forest), метод дерева прийняття рішень (англ. Decision Tree) та ін. [41]. Найвідомішим методом машинного навчання, який добре проявив себе при вирішенні задач розпізнавання символів, є метод опорних векторів (англ. support vector machine). Його головна ідея полягає в тому, щоб перетворити вхідні дані у багатовимірний простір об’єктів, та знайти гіперплощину, яка лінійно розділяє на класи, використовуючи якомога більший відступ (геометричний зазор) між крайніми точками обох класів (опорними векторами); гіперплощина проводиться в найцентральнішій позиції геометричного зазору [12].

Нейромережеві методи використовують архітектуру штучних нейронних мереж (англ. artificial neural networks) [75]. Багатошаровий персептрон є найпоширенішою повнозв’язною нейронною мережею для розпізнавання символів. [75]. При класифікації зображень символів на першому шарі відбувається

перетворення зображення у шар нейронів, який складається з k*m нейронів, де k – це висота зображення, а m – ширина. [75].

Кожен нейрон обчислює зважену суму виходів нейронів попереднього шару, після чого пропускає їх через функцію активації. Отримане значення, тобто вихід попереднього шару, активує або не активує нейрони наступного шару. Вагові коефіцієнти, пов’язані з кожним вузлом, коригуються на навчальних вибірках, поки помилка між істинним та бажаним виходом не стане граничною або нульовою [12].

Спочатку обчислювальна складність пошуку ваг, пов’язаних з нейронами, заважала застосуванню нейронних мереж, та з появою глибоких нейронних архітектур, вони зарекомендували себе як один з найкращих методів класифікації для завдань розпізнавання символів на зображеннях [41].

Сьогодні, для вирішення задач розпізнавання тексту на зображеннях дослідники зосередилися насамперед на архітектурах глибокого навчання, до якого належать рекурентні нейронні мережі, зокрема архітектура довга короткочасна пам'ять (англ. long short-term memory) та згорткові нейронні мережі (CNN) [48].

Для вирішення задачі розпізнавання символів було вирішено використовувати згорткові нейронні мережі через доведену високу точність та ефективність розпізнавання зображень згортковими нейронними мережами.

2.3 Згорткові нейронні мережі

Запропонована Яном Лекуном згорткова нейронна мережа (англ. convolutional neural network), або скорочено CNN [35], натхненна біологічними особливостями зорової кори, та є особливим типом моделі нейронної мережі, яка обробляє дані, які мають сіткову топологію – матрицю [33], відтак вона ефективна в задачах обробки візуальної інформації.

Вхідні значення для нейронів шару ЗНМ отримуються розбиттям попереднього шару на регіони, утворюючи так звані рецептивні поля. Таким чином досягається менша зв'язність шарів у порівнянні з повнозв'язними нейронними мережами, а також більша сфокусованість кожного нейрону, адже він обробляє

близькі за розташуванням (а, отже, значенням) елементи попереднього шару [33]. Набір ваг що відповідають з'єднанням елементів рецептивного поля з нейроном називають ядром (англ. kernel) або фільтром (англ. filter). Згорткову нейронну мережу визначає використання згортки принаймні в одному з шарів нейронної мережі [3]. Принцип роботи операції згортки виглядає наступним чином: нехай нейрон розташований у рядку i, колонці j з'єднаний з виходами нейронів у попередньому рівні що розміщені в рядках від i до $i + H - 1$, колонках від j до $j + W - 1$, де H та W – висота та ширина рецептивного поля.

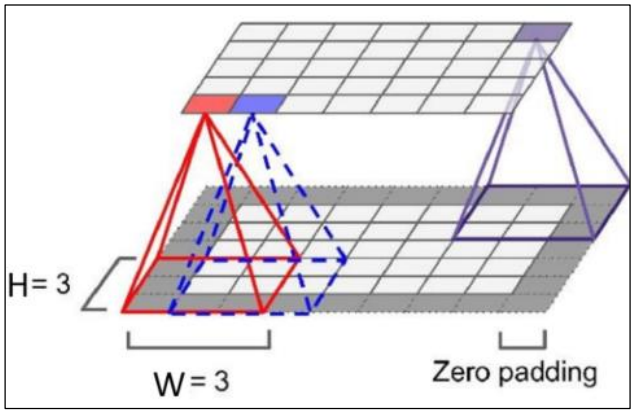


Рисунок 2.3 – Приклад проставлення нулями для операції згортки

Для збереження розмірності шарів застосовують підхід проставлення з нулями (англ. zero padding), коли рецептивне поле виходить за рамки попереднього шару, області, що бракують значень заповнюються нулями [16]. Кроком зсуву (англ. stride) рецептивного поля можливо досягти регулювання розміру вихідного шару щодо до вхідного [2].

Ядро згортки нейрона, розміщеного у рядку i, колонці j утворюється з'єднаннями з нейронами попереднього рівня, що розташовані у рядках від $i * s_h$ до $i * s_h + H - 1$, колонках від $j * s_w$ до $j * s_w + W - 1$, де s_h, s_w — значення горизонтального та вертикального зсуву [31].

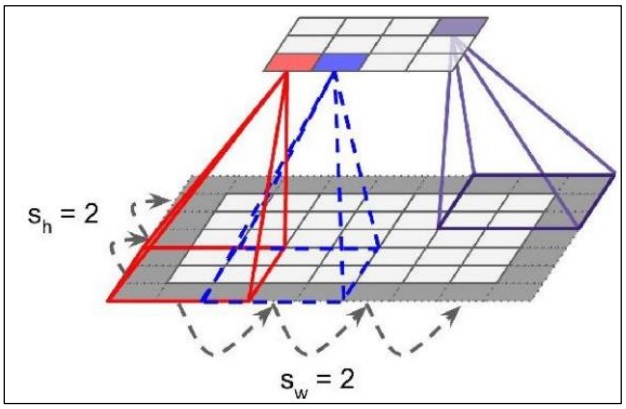


Рисунок 2.4 – Операція згортки

Таким чином можна отримати вихідне значення (карту ознак) нейрону розміщеного у рядку i, колонці j ($x_{i,j}^{l}$) за формулою 2.5.

$$x_{i,j}^{l} = f\left(\sum_{k=i*s_h^l}^{i*s_h^l+H^l-1}\ \sum_{m=j*s_w^l}^{j*s_w^l+\mathrm{W}^l-1} x_{k,m}^{l-1} \times w_{k,m}^{l}\right) \tag{2.5}$$

де l позначає поточний шар,

f – функція активації шару нейронів,

$\mathrm{x}_{\mathrm{k,m}}^{\mathrm{l-1}}$ – результуюче значення вхідного шару,

$w_{k,m}^{l}$ – ваги відповідного з'єднання у матриці ядра згортки.

Для кожного нейрону сума поелементного добутку значень рецептивного поля і ядра подається на вхід функції активації нейрону, що утворює результуючі значення шару або карти ознак (англ. feature map) [33].

Глибина згорткових нейронних мереж вважається важливим фактором для досягнення високоточних рішень. Глибина нейронної мережі є кількістю її шарів [25]. Ширина стосується кількості нейронів шару, зокрема, у згорткових нейронних мережах відповідає розмірності шару.

2.3.1 Архітектура згорткової нейронної мережі

Архітектура ЗНМ складаються з послідовності та чергування різного типу шарів, що визначає точність роботи ЗНМ. Для побудови ЗНМ зазвичай використовуються наступні шари: згортковий шар, шар субдискритизації, повнозв'язний шар. Типова структура нейронної мережі представлена на наступному рисунку:

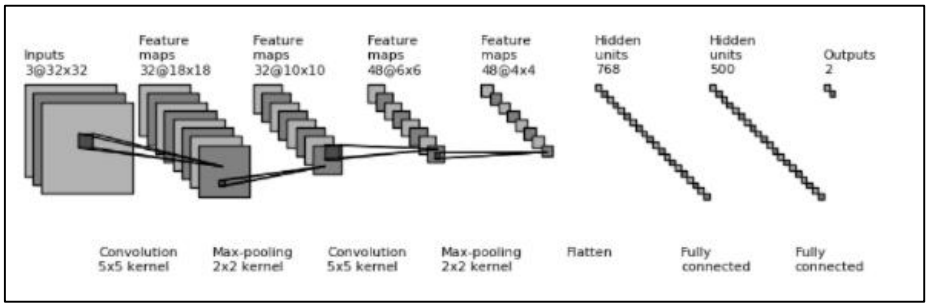


Рисунок 2.5 – Архітектура згорткової нейронної мережі

Математична модель згорткової нейронної мережі представлена наступним чином: нехай $l \in [1; L]$ – це поточний шар нейронної мережі, а кількість шарів отримаємо за формулою: $L = 2a + 2$, де $a \in Z^+$, тобто, 2а шари – як мінімум один згортковий та один субдискритизаційний та 2 – повнозв'язний та вихідний. n^l позначатиме кількість карт ознак шару l, а n-нною картою ознак на шарі l змінна x_n^l [92].

Основою згорткових нейронних мереж є згортковий шар, у якому проводиться операція згортки. Він призначений для вилучення локальних шаблонів [39]. Згортковий шар визначає вихід нейронів, з'єднаних з локальними ділянками входу, шляхом обчислення скалярного добутку між їх вагами та областю, з'єднаною з вхідним об'ємом.

В згортковій нейронній мережі шар згортки прийнято використовувати на не парній позиції $l = 2a + 1$, починаючи з першого. Його математична модель представлена формулою 2.5.

Шар субдискретизації (англ. pooling layer) шляхом агрегування карт ознак дозволяє оперувати розмірністю – зменшити розмірність представлення, а отже, кількість параметрів, які впливають і на обчислювальну складність моделі. Шар

субдискретизації обробляє кожну карту активації вхідних даних і масштабує її розмірність за допомогою відповідної функції [53]. Середня субдискритизація враховує всі активації в регіоні об'єднання. Максимальна субдискритизація фіксує лише найсильнішу активацію та ігнорує всі інші об'єднання в регіоні [53]. І хоча на максимальну субдискретизацію може впливати шум, при цьому вона використовується частіше серед архітектур ЗНМ.

В згортковій нейронній мережі шар субдискритизації прийнято використовувати на парній позиції $l = 2a$. Розглянемо математичну модель шару субдискритизації: нехай за шаром l слідує шар субдискритизації, пряме поширення можна описати як:

$$a_j^{(l+1)} = Pool\left(a_1^{(l)}, a_2^{(l)}, .., a_i^{(l)}, \dots, a_n^{(l)}\right), i \in R_j^{(l)} \quad (2.6)$$

де $Pool$ () позначає функцію субдискритизації (наприклад, max $(a_1^{(l)}, a_2^{(l)}, .., a_n^{(l)})$); $R_j^{(l)}$ – регіон j субдискритизації l [53].

А шар субдискритизації l (n-нна карта ознак шару l) описується наступною формулою [53]:

$$x_n^l = f(w^l * Pool\left(a_1^{(l)}, a_2^{(l)}, .., a_n^{(l)}\right) + b^l) \quad (2.7)$$

Повнозв'язані шари у згортковій нейронній мережі необхідні тому, що у неглибокій моделі ЗНМ ознаки, отримані від останнього згорткового шару, відповідають частині вхідного зображення, оскільки його рецептивне поле не охоплює зображення у його повному просторовому вимірі. Через це додавання кількох повнозв'язаних шарів є обов'язковими у цьому випадку.

Математична модель повнозв'язного шару має наступний вигляд: нехай l – це вихідний шар, що складається з одиничних нейронів, тоді n^l – кількість нейронів

даного шару. b^l – коефіцієнти зсуву шару l [92]. Тоді значення вихідного нейрону j дорівнюватиме:

$$x_n^l = f(\sum_i^{n^{l-1}} x_i^{l-1} * w_i^{l-1} + b^l) \quad (2.8)$$

де f() – це функція активації;

w_i^l – матриця вагових коефіцієнтів шару l;

x_n^l – активований нейрон шару l [50].

На вихідному шарі міститься n кількість нейронів, де n відповідає кількості класів, а x_n^l – значення вихідного нейронну.

Архітектура нейронної мережі для мобільного додатку повинна бути невеликою, але при цьому ефективною у точності розпізнавання. Серед архітектур, які використовуються на мобільних пристроях існують наступні: Factorized Networks [51], Xception network [13], Inception V3 [46] та ін. Найшвидшою та при цьому ефективною на сьогодні є MobileNetV2. Для вирішення даної задачі та з врахування клієнтської архітектури додатку було обрано архітектуру саме цієї нейронної мережі. Замість використання стандартної згортки MobileNetV2 використовує спеціальну згортку, яка називається глибинно роздільною згорткою, відтак, мережа потребує приблизно однієї восьмої обчислювальних витрат і має зовсім невелике падіння точності [56] (детальніше див. розділ 3.3).

2.3.2 Навчання згорткової нейронної мережі

Ключовим компонентом згортки є ядро, яке є набором певних параметрів. Значення параметрів визначають, які особливості та шаблони ідентифікує операція згортки. Вибір параметрів має вирішальне значення для досягнення високої точності розпізнавання зображення. Навчання ЗНМ проводиться за допомогою градієнтного спуску та зворотного поширення. Цей метод дозволяє автоматично

налаштовувати параметри за допомогою позначених навчальних даних. Функція втрат (англ. loss function) обчислює, наскільки добре працює мережа на одному прикладі входу у порівнянні з істинним виходом. На основі цього можна розрахувати градієнт функції втрат, оцінений на навчальному прикладі, щодо параметрів або ваг у кожному шарі [16]. Критерій помилки у задачах класифікації найчастіше вираховується функцією крос-ентропії, яка має наступний вигляд:

$$E = -\sum_{i=1}^{n}\sum_{j=1}^{m} t_i * \log(y_i) \tag{2.9}$$

де t – істинні виходи нейронів;

y – передбачені виходи нейронів;

Σ_i^n– сума по навчальній вибірці;

Σ_i^m– сума по класах [9];

Навчання відбувається шляхом коригування ваг таким чином, щоб y_i був максимально близьким або дорівнював відповідному t_i. У класичному алгоритмі зворотного поширення помилки змінюються відповідно до напрямку градієнтного спуску функції помилки E. Тоді, метою навчання мережі є налаштування параметрів для зведення до мінімуму E.

Відтак, градієнт функції втрат від матриці ваг виглядає наступним чином:

$$\nabla \mathrm{E(W)} = [\frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n}] \tag{2.10}$$

де $\frac{\partial E}{\partial w}$ – часткова похідна функції помилки по коефіцієнту ваги нейрону;

n – представляє кількість всіх коефіцієнтів ваг на всіх шарах [16].

Використовуючи ланцюгове правило диференціювання, цей градієнт можна розкласти на частини, що відповідають кожному шару, що показує наступна формула:

$$\frac{\partial E}{\partial w_{i,j}} = \frac{\partial E}{\partial y_i} * \frac{\partial yi}{\partial s_i} * \frac{\partial s_j}{\partial w_{i,j}} \quad (2.11)$$

де $\frac{\partial E}{\partial w_{i,j}}$ – це значення похідної функції помилки по коефіцієнту ваг $w_{i,j}$, між нейронами i та j;

$\frac{\partial E}{\partial y_i}$ – значення помилки нейрона j(також позначається δ);

$\frac{\partial \mathrm{yi}}{\partial \mathrm{s_i}}$ – значення похідної функції активації по її аргументу для нейрона j;

$\frac{\partial s_j}{\partial w_{i,j}}$ – вихід i нейрона попереднього шару (по відношенню до нейрону j) [9].

Для розрахування помилки внутрішніх шарів, застосовується принцип зворотного поширення помилки для обчислення градієнта відносно ваг у кожному шарі – напрямок найкрутішого підйому функції втрат щодо кожної ваги. Оновлюючи вагу в негативному напрямку градієнта, роблячи «крок» відповідно до параметра розміру кроку, втрата на виході зменшується. Тоді, оновлення ваг можна записати наступним чином:

$$\theta_i = \theta_i - \eta \frac{\partial E}{\partial \theta_i} \quad (2.12)$$

де θ_i – є параметром, а η – розміром кроку [39].

Принцип градієнтного спуску і застосовується до ваг ядра, так само як і до ваг у традиційному шарі нейронної мережі – тобто, для згорткового шару вираховування дельти відбувається таким самим чином: необхідно вирахувати $\frac{\partial E}{\partial w_{i,j}}$, що можна інтерпретувати як вимірювання того, як зміна одного пікселя $w_{i,j}$ у ядрі впливає на функцію втрат E.

Через збільшення розмірів нейронних мереж, які навчаються методом оберненого поширення похибки (англ. back propagation), збільшується і кількість параметрів, відтак моделі схильні до перенавчання. Перенавчання — це явище, при якому модель завчає приклади надані при навчанні та втрачає здатність до узагальнення, що призводить до зниження показників ефективності мережі.

Дієвим шляхом визначення наявності перенавчання є порівняння значень точності на навчальних та тестових наборах даних. При перенавчанні можна прослідкувати високий рівень точності при роботі з навчальним набором та значне погіршення цього показника на даних тестового набору [38].

Далі наведено декілька шляхів боротьби з перенавчанням:

Послаблення ваг (англ. weight decay). Менша кількість параметрів – це лише один із способів запобігти надмірній складності нашої моделі. Розв'язання складних проблем вимагає складніших нелінійних моделей. Проте водночас постає необхідність контролю при нарощенні складності. Отже, варто накласти свого роду штраф на складність, таким чином модель зможе мати значну кількість параметрів, проте матиме обмеження на шляху до надлишкової складності й перенавчання.

Одним зі способів знеохочення нарощення складності є додавання всіх параметрів до функції втрат. Через те, що параметри можуть бути як позитивними, так і негативними, ми зацікавленні у їх квадратичному значенні — w^2. Для боротьби з ситуацією, коли функція втрат набуватиме дуже великих значень через додані параметри, необхідно ввести коефіцієнт ослаблення ваг ($0 < d << 1$), що буде застосований до суми квадратів. Наприклад, отримаємо функцію втрат, до якої застосовано підхід ослаблення ваг:

$$Loss = MSE(\hat{y}, y) + d \times \sum w^2 \tag{2.13}$$

де $MSE(\hat{y}, y)$ – середньоквадратична похибка нейронної мережі,

d – коефіцієнт ослаблення ваг,

w – значення параметру нейронної мережі.

При оновленні ваг підчас градієнтного спуску отримуємо вплив оновленої функції втрат:

$$w^l = w^{l-1} - \eta \times \left(\frac{\partial Loss}{\partial d}\right) \tag{2.14}$$

де η – гіперпараметр мережі «швидкість навчання» (англ. learning rate).

Тепер коли функція втрат складається з двох членів, отримаємо похідну другого члену по змінній w:

$$\frac{\partial}{\partial w}(d \times w^2) = 2 \times d \times w \tag{2.15}$$

А отже відтепер, при навчанні ваги мережі виконується віднімання не тільки значення кроку за градієнтним спуском, а ще й оригінальне значення ваги помножене на константу. Що і призвело найменування підходу методом ослаблення ваг.

Рання зупинка. Для уникнення перенавчання на тренувальному наборі при виконанні навчання на заданих інтервалах проводиться аналіз ефективності мережі на тестовому наборі та зберігається копія мережі. По завершенню процесу навчання, проводиться аналіз проміжних мереж та обирається та, що досягла найкращих показників [31].

Метод виключення (англ. dropout). Ключовою ідеєю даного методу є випадкове видалення елементів (разом з їх з'єднаннями) з нейронної мережі під час процесу навчання. Це запобігає надмірному проявові явища спів-адаптування та значно підвищує ефективність нейронних мереж, що розв'язують задачі комп'ютерного зору, розпізнавання мовлення, класифікації тощо [20].

Розширення навчального набору (англ. data augmentation). Класичним підходом до розширення навчального набору є застосування маніпуляцій над кольором та геометричними якостями початкового навчального набору,

утворюючи нові входження навчального набору. Застосовують такі операції як дзеркальне відображення, обрізання та розтягування зображення, зміна палітри кольорів тощо [61].

Перевикористання параметрів (англ. paramenter sharing). ЗНМ є ідеальним кандидатом на застосування перевикористання параметрів, адже згорткові шари мережі складаються з набору нейронів, кожен з яких має рецептивне поле та асоційоване ядро з формалізованою структурою, що і є головною ціллю для використання підходу. Зменшення кількості параметрів досягається їх перевикористанням, тобто, деякі два або більше окремих незалежних параметрів об'єднуються у один єдиний, значення якого застосовується там, де раніше застосовувались значення окремих, цільових параметрів [73].

Висновки до розділу

У даному розділі були поставлені задачі до розробки бібліотеки інтелектуального визначення складу продуктів харчування, було розглянуто необхідність конвертації кольорового зображення в одноканальне та його бінаризації. Серед методів детекції тексту таких для детекції тексту було обрано алгоритм максимально стабільних екстремальних регіонів, який є типом методів на базі зв’язних компонентів. Також було виявлено, що найбільш ефективними методами класифікації зображень є використання згорткових нейронних мереж. Було обрано архітектуру нейронної мережі MobileNetV2 для поставленої задачі розпізнавання тексту, яка завдяки наявним інвертованим залишковим блокам, а також використанню роздільним по глибині згорткам вимагає менше часу на обробку зображення, а тому ідеально підходить для вирішення задачі розпізнавання тексту мобільною системою.

3 АЛГОРИТМІЧНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Структура алгоритмічного забезпечення

Алгоритмічне забезпечення даної системи представляє собою набір алгоритмів для виявлення регіонів з символами, групування символів у рядки, вираховування оцінки шкідливості інгредієнтів тощо, та має наступні компоненти:

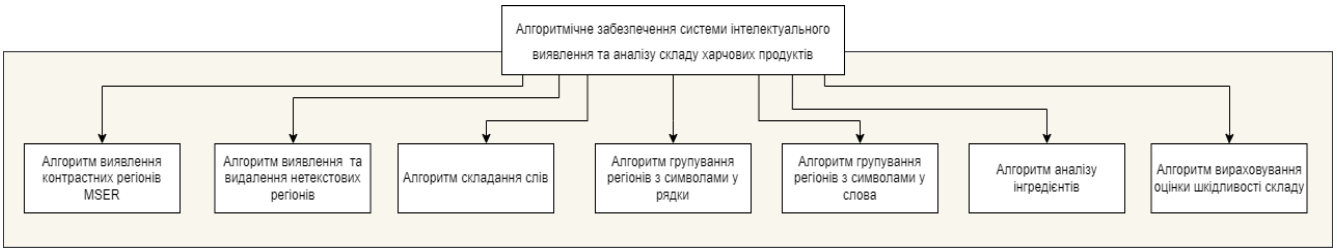


Рисунок 3.1 – Компоненти алгоритмічного забезпечення

Розроблене алгоритмічне забезпечення має на меті вирішити перелік наступних задач:

- виділити на матриці зображення контрастні регіони – потенційні регіони з текстом;
- виявити та видалити серед виявлених регіонів-кандидатів нетекстові регіони;
- текстові регіони згрупувати у слова та рядки;
- побудувати сховище даних для зберігання всієї необхідної інформації про інгредієнт: опис, характеристики, рівень загрози;
- розпізнати регіони згортковою нейронною мережею;
- проаналізувати отримані в результаті попередніх кроків слова-інгредієнти та вирахувати оцінку шкідливості складу харчового продукту.

Необхідність побудови алгоритмічного забезпечення обумовлена наявністю, окрім інтелектуального визначення тексту, математичних алгоритмів, наведених далі.

3.2 Алгоритм MSER для виділення текстових регіонів

Алгоритм MSER у представленому алгоритмічному забезпеченні використовується для виділення текстових регіонів на матриці вхідного

зображення. Оскільки один символ на матриці зображення складу продуктів має схожий колір пікселів, а їх інтенсивність відрізняється від фону, алгоритм MSER може ефективно знайти текстові області, які представлені групою поєднаних між собою за інтенсивністю та контрастністю пікселів. Детектор вибирає видимі області серед екстремальних областей зображення, що визначається екстремальною властивістю функції інтенсивності зображення на зовнішній межі області. Серед цих областей мінімальні (відповідно максимальні) екстремальні області – це такі пов'язані області зображення, що всі елементи на зовнішній межі мають суттєво більшу (або ж, відповідно, меншу) інтенсивність, ніж усі суміжні елементи області [11].

Метод MSER зазвичай використовується для обчислення стабільних регіонів зображень на основі ідеї методу водоподілу (англ. watershed algorithm) – розглядає значення пікселів як локальну висоту. Стабільні області визначаються виключно екстремальною властивістю функції інтенсивності і області та його зовнішній границі. Нехай регіон Q є підмножиною пікселів зображення, а ∂Q є границею регіону Q, а відтак, підмножиною пікселів, яка суміжна, принаймні з одним пікселем регіону Q, але не належить йому [52].

На першому кроці алгоритму до матриці зображення застосовується ряд порогових значень – порогове значення зображення на кожному рівні, щоб створити набір чорно-білих зображень (набір рівнів зображення). Нехай $D \subset \mathrm{R}^2$ є областю значень, в якій визначено зображення у відтінках сірого як $\mathrm{I}: \mathrm{D} \rightarrow [0,1]$. Тоді кожне зображення I буде представлено як колекція наборів його рівнів. Набір рівнів зображення I, за деяким заданим $\eta \in [0,1]$, є множиною $\{\mathrm{d} \in \mathrm{D}: \mathrm{I}(\mathrm{d}) = \eta\}$ [5].

Для зображення із заданою інтенсивністю припустимо, що його діапазон рівня сірого дорівнює $[0, 1]$; n рівноінтервальних порогів рівня сірого встановлюються як $\{\eta_i \mid \eta_{i+1} = \eta_i + \Delta, \eta_i \in [0,1], i = 1,2 \dots, n\}$, де Δ (дельта) означає інтервал, а $\eta_i - i$-ий поріг. Для певного порогу, якщо рівень сірого пікселя на зображенні є менше, ніж поріг, то рівень сірого пікселя встановлюється на нуль; інакше він встановлюється як один. Таким чином, отримуємо n двійкових

зображень, що були отримані за допомогою застосування порогів. Область, що утворена межею, яка проходить між регіоном з інтенсивністю вищою за рівень порогу та регіоном, що має інтенсивність нижче порогу в кожному бінарному зображенні називається екстремальним регіоном [67].

Екстремальний регіон є зв'язаною областю, що відповідає певному порогу рівня сірого, і рівні сірого всіх пікселів у цій області більші за поріг, тоді як рівні сірого за межами цієї області всі менші за поріг. Відношення суміжності двох точок визначається як A ⊂ D×D, при чому використовуються 4 суміжні пікселі, як приклад точки p, q ∈ D суміжні (pAq), тільки при умові, що $\sum_{i=1}^{d} |p_i - q_i|$. Відтак, екстремальний регіон $\mathcal{Q}$, це регіон зображення, де для всіх $p \in \mathcal{Q}, q \in \partial\mathcal{Q}$: $I(p) > I(q)$ (максимально інтенсивна область) або $I(p) < I(q)$ (мінімально інтенсивна область), де p – це певна точка регіону Q, q – певна точка границі регіону $\partial\mathcal{Q}$ [52]. Тобто всі значення пікселів у областях є або суворо темнішими, або суворо яскравішими за значення на межі, де поріг інтенсивності дорівнює η.

Стабільність регіону визначається за наступною формулою (3.1):

$$\Psi_1(\mathcal{Q}_{\eta_i}) = \frac{S\left(\mathcal{Q}_{\eta_i}\right)}{\frac{d}{d\eta} S\left(\mathcal{Q}_{\eta_i}\right)} \tag{3.1}$$

де $S\left(\mathcal{Q}_{\eta_i}\right)$ позначає площу регіону $\mathcal{Q}_{\eta_i}$.

Область вважається стабільною, якщо її площа незначно змінюється зі зміною порога η. Регіон $\mathcal{Q}_{\eta_i}$ називається максимально стабільним, якщо $\Psi_1(\mathcal{Q}_{\eta_i})$ має локальний максимум при η [5].

Екстремальні регіони повинні бути максимально стабільними, тобто такими площа яких є максимально незмінною при зміні порогового значення. Екстремальні регіони вважаються максимально стабільними, якщо виконуються наступні умови (3.2):

$$\begin{cases} S\left(\mathcal{Q}_{\eta_i}\right) \in [a,b] \\ q_{\eta_i} = \dfrac{|S(\mathcal{Q}_{\eta_i+1}) - (\mathcal{Q}_{\eta_i})|}{(\mathcal{Q}_{\eta_i})} < \varepsilon \end{cases} \tag{3.2}$$

де $S\left(\mathcal{Q}_{\eta_i}\right)$ представляє площу регіону;

[a, b] – діапазон розміру області;

q_{η_i} – ступінь зміни площі екстремальної області $\mathcal{Q}_{\eta_i}$;

ε – верхня межа ступеня зміни площі [67]. Інтервал [a, b] представляє собою параметр для налаштування розміру бажаної для захоплення області.

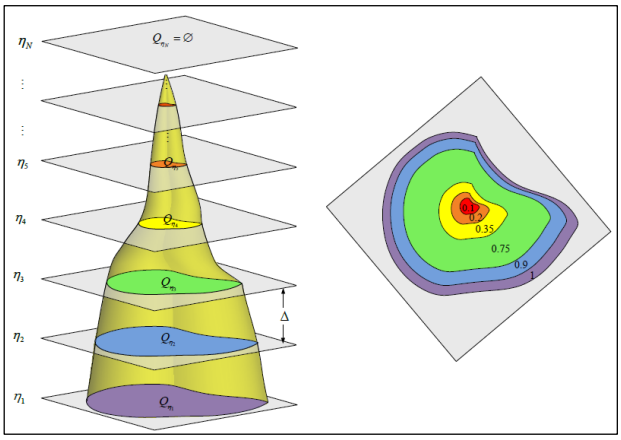


Рисунок 3.2 – Генерація набору екстремальних регіонів

На рисунку 3.2 (зліва) представлена генерація набору екстремальних регіонів, які представлені як $\mathcal{Q}_{\eta_1} \supset \mathcal{Q}_{\eta_2} \supset \mathcal{Q}_{\eta_3} \supset \cdots \mathcal{Q}_{\eta_{n-1}} \supset \mathcal{Q}_{\eta_n}$, а справа показані ступені зміни площі екстремального регіону [67]. Підбір параметрів впливає на швидкість та ефективність виявлення максимально стабільних екстремальних регіонів, які відповідають областям з символами на зображеннях зі складу харчових продуктів. Дельта Δ не повинна бути дуже великою, щоб отримати точний ступінь q_{η_i}, та дуже малою, оскільки це збільшить обчислювальні витрати. Отож, для компромісу між ефективністю обчислення та точністю, дельта повинна бути у рамках між $\frac{1}{32}$ і $\frac{1}{16}$.

Якщо верхня межа ступеня зміни площі ε є занадто великою, тоді велика кількість регіонів-кандидатів, які несуть корисну інформацію (тобто, потенційно є символами), відсіюється, що зменшує точність виявлення символів. Якщо верхня

межа ступеня зміни площі ε є занадто малою, тоді на зображені буде виявлена велика кількість дефектних регіонів, які не позиціонуються як регіони з символами. А тому дану межу варто встановити значенням від 0,2 до 0,4, що було перевірено на значній кількості зображень зі складом харчових продуктів.

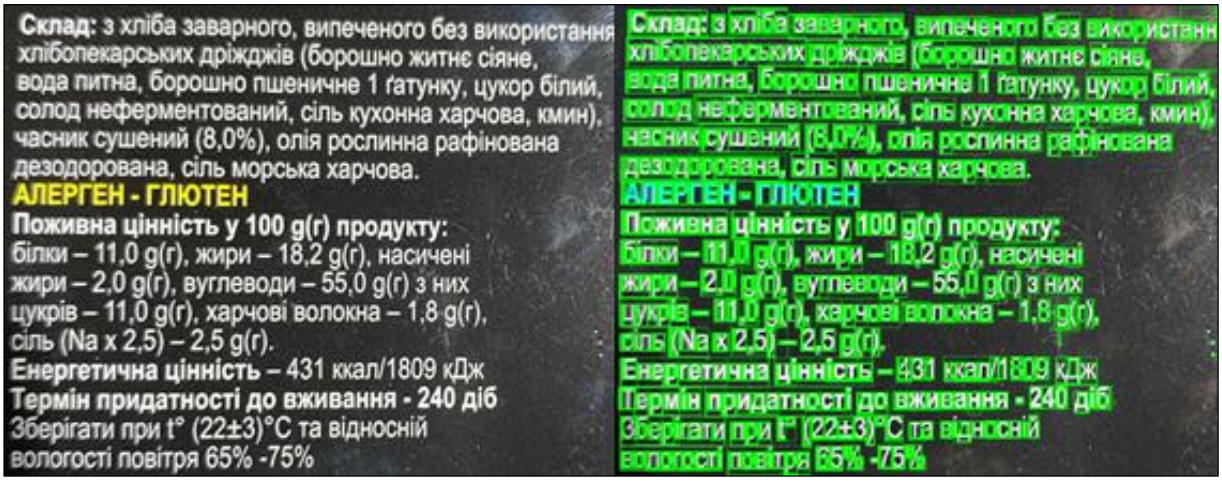


Рисунок 3.3 – Результат роботи алгоритму для виявлення текстових регіонів на зображенні етикетки харчового продукту

Відтак, чим менший ступінь зміни площі області, тим стабільнішою є область. Максимально стабільна екстремальна область визначається як екстремальна область, яка мінімізує ступінь зміни площі. Максимально стабільний регіон може бути математично представлений як (3.3):

$$Q_* = \arg\min_{Q_{\eta_i}} \{q_{\eta_i}, i = 1,2, \dots\} \tag{3.3}$$

Тобто, з набору $\{Q_{\eta_i}, i = 1,2, \dots\}$, обирається значення Q_{η_i}, яке має відповідне q_{η_i}, що є найменшим у наборі. Q_* найкраще репрезентує максимально стабільний екстремальний регіон, що представлений набором $\{Q_{\eta_i}, i = 1,2, \dots\}$ та приховує деталі реалізації, конвертуючи набір регіонів утворених застосуванням порогів у скалярне значення [67].

Загальний процес виявлення контрастних областей з тексом на зображення алгоритмом MSER має наступний вигляд:

1) на **вхід** поступає матриця зображення;
2) зображення обробляється із застосуванням різних порогів інтенсивності від чорного до білого з певним кроком порогу, для визначення наборі рівнів зображення;
3) вилучаються зв'язні компоненти – екстремальні області, де контрастність пікселів є більшою порівняно з суміжними;
4) знаходиться ступінь зміни площі екстремальної області, при якому екстремальна область є максимально стабільною, тобто знайти локальний мінімум відносного зростання її площі;
5) знайдені максимально стабільні екстремальні регіони обрамляються з допомогою обмежувальних прямокутників;
6) на **виході** отримується множина прямокутників, які містять в собі максимально стабільні екстремальні області, тобто кандидати-символи.

Оскільки метод MSER вимагає лише невеликої кількості обчислювальних ресурсів, обчислювальну ефективність виявлення текстових регіонів можна значно підвищити. Складність алгоритму виявлення максимально стабільних екстремальних областей представлена як $O(pN)$, де p позначає кількість використовуваних каналів. У нашому випадку, вхідні дані для алгоритму представлені матрицею, попередньо конвертованого в одноканальне, зображення.

Для ще більшої оптимізації застосуємо наступний принцип: нехай величина r є величиною зменшення зображення, а матриця M_r утворена застосуванням коефіцієнта r до початкової матриці. Тоді, виходячи з того, що алгоритм MSER має складність $O(N)$ отримуємо лінійне зменшення кількості необхідних обчислень, що пропорційна r. Виявлені координати регіонів з текстом перемножуються на початкове відношення обернене до r, тобто, $\frac{1}{r}$ та переносяться на вхідне зображення. В результаті, швидкодія алгоритму збільшується, а якість отриманих регіонів залишається первинною.

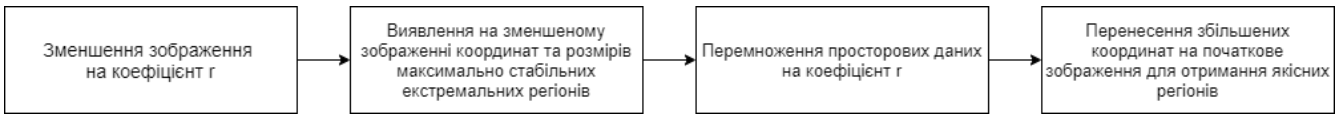


Рисунок 3.4 – Принцип збільшення швидкодії застосування алгоритму MSER для задачі розпізнавання тексту на зображенні

Алгоритм MSER виявляє та вилучає більшу частину тексту, але заразом виявляє багато інших стабільних регіонів на зображенні, які не є текстом. Спробуємо об’єднати використання декількох підходів для створення алгоритму для виявлення та видалення нетекстових областей, присутніх на зображенні.

3.3 Алгоритм для видалення нетекстових регіонів

Виявлені алгоритмом MSER нетекстові області, які необхідно відфільтрувати, можна поділити на такі групи: дефектні контрастні області, які мають велике співвідношення висоти до ширини, або ширини до висоти; регіони з геометричними формами, які не характерні символьній інформації; регіони поза основною областю з текстом; контрастні регіони створені внутрішньою частиною інших регіонів; області, які мають подібні обмежувальні рамки (дублікати).

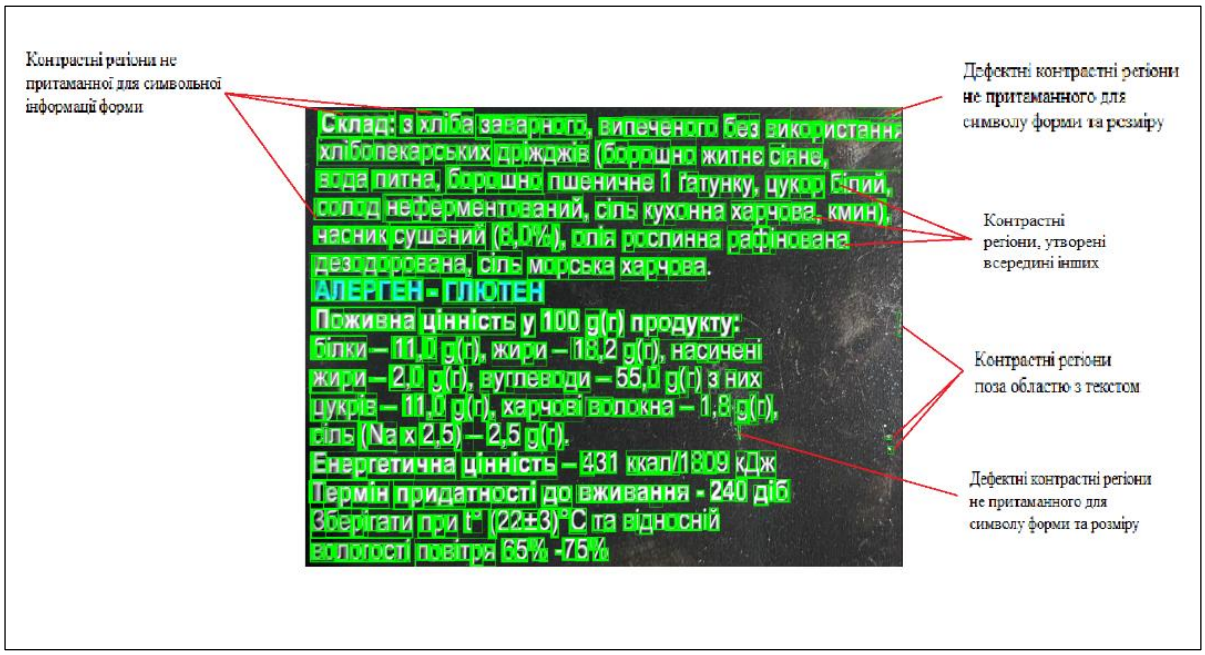


Рисунок 3.5 – Представлення нетекстових областей після виявлення контрастних регіонів алгоритмом MSER

А відтак, для виявлення та видалення нетекстових областей необхідно вирішити певні проблеми, а саме:

- виявити та видалити дефектні контрастні області та контрастні регіони з геометричними формами, які не притаманні символьній інформації;
- виявити та видалити контрастні області, створені внутрішньою частиною інших регіонів;
- виявити та видалити області, які мають подібні обмежувальні рамки (дублюючі регіони);
- виявити регіони, які не належать основній області з текстом.

Алгоритм для виявлення та видалення нетекстових областей виглядає наступним чином:

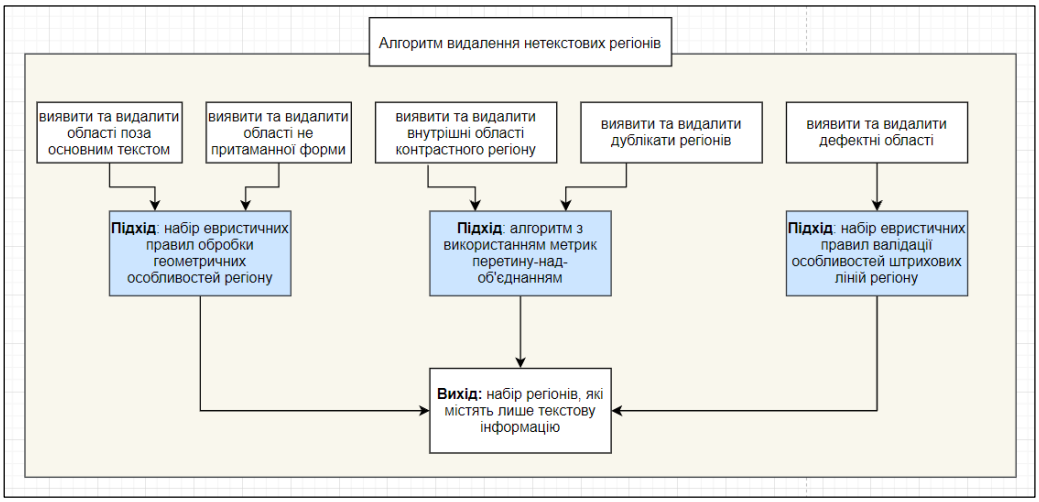


Рисунок 3.6 –Алгоритм видалення нетекстових областей

3.2.1 Алгоритм визначення ширини штрихових лінії та евристична фільтрація

Для вирішення проблеми виявлення та видалення дефектних контрастних областей, регіонів, які не належать основній області з текстом, регіонів, якій мають не притаманні форми використовуються два підходи до розгляду особливостей нетекстових регіонів: виявлення геометричних особливостей регіону та виявлення особливостей варіації штрихових ліній, наявних у контрастному регіоні [80].

Нехай виявлені алгоритмом MSER регіони-кандидати позначаються як $C = \{c_1, c_2, \cdots, c_m\}$, де m є загальною кількістю регіонів. Регіон x визначається як $x \in C$.

Відповідно до особливостей геометрії форми та ліній символьної інформації, встановлюється та застосовується набір евристичних правил. До особливостей штрихових ліній зображення відноситься:

- співвідношення ширини штрихової лінії до висоти регіону (3.4):

$$R_{sh}(x) = sw_x/h_x \tag{3.4}$$

- співвідношення ширини штрихової лінії до ширини регіону (3.5):

$$R_{sw}(x) = sw_x/w_x \tag{3.5}$$

При чому R_{sh} , R_{sw} позначають відповідні відношення штрихових ліній для кожного регіону, а sw_x – ширина штрихової лінії для регіону x.

Ширина штрихової лінії — це значення ширини ліній, які утворюють символ. Символьні області мають невеликі варіації ширини штрихової лінії, тоді як нетекстові області мають більші варіації.

Штриховою лінією (англ. stroke) називається частина зображення, яка утворює смугу постійної ширини [31]. Для знаходження ширини штрихової лінії використовується підхід, принцип роботи якого полягає в наступному.

Спочатку використовується метод потоншення, запропонований Ламом та іншими [54] для виділення скелету символу шириною в один піксель.

Після цього використовується метод отримання точок ознак скелета символу, шляхом обчислення восьми значень пікселів сусідства точок пікселя скелета [49] для отримання кінцевих точок (англ. end points). На наступному рисунку зелена лінія представляє скелет символу, а червоні точки є кінцевими точками.

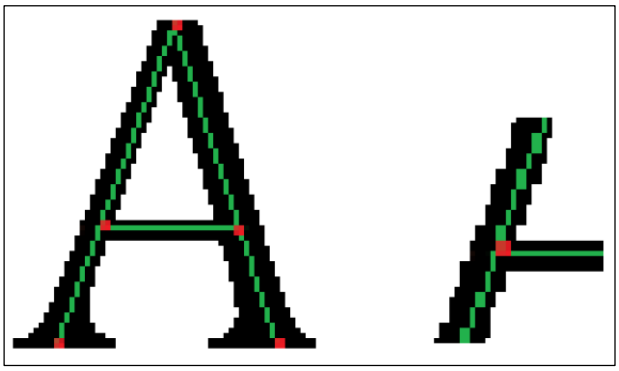


Рисунок 3.8 – Представлення скелету символу з кінцевими точками

Якщо розглядати зображення як бінаризоване, тоді точки скелету символу дорівнюються 1, тоді як всі інші суміжні пікселі мають значення 0. Відтак, точки ознак є: $N(P) = 1$, де $N(P)$ означає, що є лише одна точка пікселя P має значення 1 серед суміжних точок скелету символу є кінцевими точками. Кількість кінцевих точок визначається як n [63].

Нехай на певному сегменті маємо кінцеву точку $P_e(x_e y_e)$, заданий крок e, що є відстаню між точками, утвореними на скелеті та точки, що утворені застосуванням кроку e, починаючи від P_e: $P_1(x_1, y_1)$, $P_2(x_2, y_2)$,…, $P_n(x_n, y_n)$. На наступному зображенні представлені сині точки $\mathrm{P}(x, \mathrm{y})$, червона кінцева точка $\mathrm{P}(x, \mathrm{y})$, ламана, позначена жовтим кольором – скелет символу.

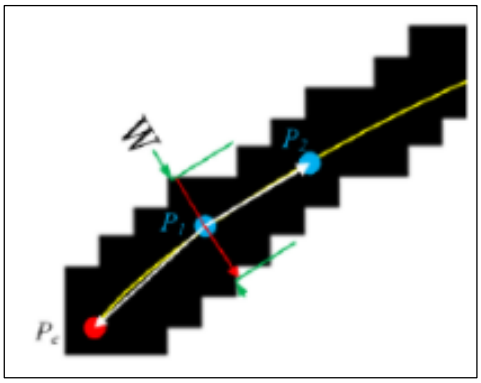


Рисунок 3.10 – Визначення ширини штрихової лінії

Кут точки $P_1(x_1, y_1)$є кутом θ між векторами $P_1P_e = (x_e - x_1, y_e - y_1)$ та $P_1P_2 = (x_2 - x_1, y_2 - y_1)$.

Кут вираховується за наступною формулою [63]:

$$\theta = \arccos\left(\frac{\overrightarrow{P_1P_e} * \overrightarrow{P_1P_2}}{\overrightarrow{|P_1P_e|} * \overrightarrow{|P_1P_2|}}\right) \tag{3.6}$$

Далі вираховується ширина штрихової лінії, яка визначається довжиною відрізка a_i, проведеного через точку P_i перпендикулярно до $\overrightarrow{P_iP_{i+1}}$ та обмеженого контуром штрихової лінії. Довжина відрізка a_i є шириною w_i штрихової лінії на проміжку P_iP_{i+1}. А отже, ширина штрихової лінії конкретного символу визначається за формулою [63]:

$$w = \frac{1}{n}\sum_{i=1}^{n} w_i \tag{3.7}$$

Відтак, вираховуємо середню ширину штрихової лінії кожного регіону, порівнюємо значення ширини штрихової лінії та відкидаємо ті регіони, які мають занадто велике значення ширини штрихової лінії. Вирахувавши ширину штрихової лінії встановлюємо обмеження: $\frac{1}{6} < R_{sh}(x) < \frac{1}{9}$, $0.3 < R_{sw}(x) < 3$.

Наступним кроком є використання евристичного фільтра відповідно до геометричних особливостей символьного регіону. Відтак, до геометричних особливостей відносяться співвідношення сторін, співвідношення загальної площі до площі обмежувального прямокутника. Вузькі та довгі регіони навряд чи відповідають геометрії символу, а тому такі компоненти повинні відсіюватись [23].

Якщо співвідношення сторін визначається як $R_{hw}(x) = h_x/w_x$, тоді для фільтрації нетекстових регіонів встановлюємо значення співвідношення як $0.1 < R_{hw}(x) < 10$.

Співвідношення між діаметром регіону, який визначається як $d = \sqrt{w^2 + h^2}$ та середнім значенням ширини штрихової лінії визначається як $R_{sd}(x) = sw_x/d_x$, встановлюється в межах $R_{sd}(x) < 10$.

Дані алгоритми для фільтрації дефектних регіонів та регіонів неправильної форми є ефективними для даних задач, та залишають проблему наявності повторюваних обмежувальних прямокутників (дублікатів), а тому було прийнято рішення скористатись метриками перетину над об’єднанням для видалення дублюючих регіонів.

3.2.2 Алгоритм для видалення дублікатів та областей, створених внутрішньою частиною інших регіонів використанням метрик перетину над об’єднанням

Перетин над об’єднанням (англ. Interception over Union) – це метрика, яка використовується для отримання індексу подібності між двома довільними формами об’єктів, також відома як індекс Жаккара. IoU на базі властивостей форми об'єктів, що порівнюються, наприклад ширини, висоти та розташування двох обмежувальних прямокутників у властивості регіону, обчислює нормалізовану міру, яка зосереджується на їхніх площах (або об’ємах) [30].

Проаналізувавши рис. та отримавши множину всіх контрастних регіонів, які були розпізнані алгоритмом MSER**,** було виявлено, що контрастні регіони, які мають подібні обмежувальні прямокутники, мають велику площу перетину S_I та площу об’єднання S_O, а отже визначивши показник подібності IoU двох регіонів можна видалити дублюючий регіон.

Рисунок 3.11 – Набір регіонів, які мають подібні обмежувальні прямокутники

Метрика IoU є способом вирішення проблеми прибирання дублікатів, оскільки дає можливість порівняти два виявлені контрастні регіони, на предмет повторюваності, завдяки властивості показника подібності IoU.

Відтак, нехай на координатній площині існують два прямокутники з координатами x_1, y_1 та x_2, y_2 відповідно.

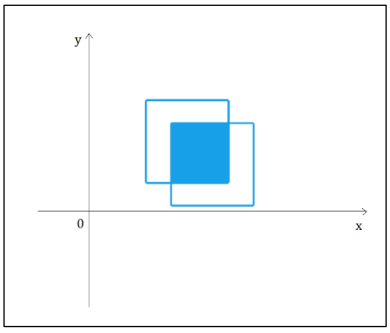


Рисунок 3.12 – Представлення перетину на координатній площині

Тоді координати перетину визначаються як:

$$x_{Ia} = \max(x_1, x_2) \quad (3.8)$$

$$y_{Ia} = \max(y_1, y_2) \quad (3.9)$$

$$x_{Ib} = \min(x_1, x_2) \quad (3.10)$$

$$y_{Ib} = \min(y_1, y_2) \quad (3.11)$$

Площа перетину S_I вираховується за формулою добутку різниці координат перетину:

$$S_I = (x_{Ib} - x_{Ia}) * (y_{Ib} - y_{Ia}) \quad (3.12)$$

Далі знайдемо площу для кожного з прямокутників S_a та S_b як $S = (x_2 - x_1) * (y_2 - y_1)$.

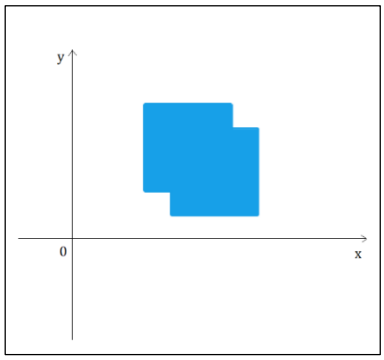


Рисунок 3.13 – Представлення об’єднання на координатній площині

Відтак, формула площі об’єднання S_U має наступний вигляд:

$$S_U = S_a + S_b - S_I \tag{3.13}$$

Показник подібності знаходиться наступною формулою:

$$IoU = \frac{S_I}{S_U} \tag{3.14}$$

де S_I – це A ∩ B, а S_U – A ∪ B [62].

Початковий поріг t є показником бажаної точності видалення дублікатів та порівнюється з вихідним показником подібності. Відтак, вхідними даними буде множина прямокутників та початковий поріг. В нашому випадку, коефіцієнт, який буде вищий за початковий поріг вважатиметься істинним екземпляром літери, а нижчий – дублікатом, який буде відсіюватись. Нехай порогове значення встановлено як $t = 0{,}2$ – це означає, що якщо хоча б 20% площі двох прямокутників перетинається, один з прямокутників буде видаленим.

Алгоритм 1: Видалення дублікатів з використанням метрик IoU.

Вхідні дані: множина прямокутників $C = \{c_1, c_2, \dots, c_n\}$ з координатами x, y та розміром w, h, поріг коефіцієнта t при якому два прямокутники вважаються дублікатами.

Результат: множина прямокутників з видаленими дублікатами.

1) З метою оптимізації, в залежності від кількості прямокутників у вхідній множині розрахуємо параметр m, який визначає кількість рядків та колонок на які необхідно поділити площину, що розміщує прямокутники.
2) Поділимо прямокутники на групи за належністю до комірки (прямокутник що утворюється поділенням площини на n колонок і рядків)
3) Для всіх можливих пар прямокутників в рамках комірки розрахуємо коефіцієнт подібності IoU.
4) Позначимо прямокутник у кожній парі, де коефіцієнт подібності IoU перевищує заданий поріг t, один з прямокутників.
5) Отримаємо результуючу множину прямокутників, відфільтрувавши прямокутники що були відмічені у попередньому кроці.

Вихідними даними буде множина зображень з видаленими дублікатами, тобто єдино істинні символи тексту зображення складу продукту. Встановлюємо пороговий коефіцієнт як $t = 0{,}1$, за якого всі прямокутники, де коефіцієнт подібності IoU вищий, будуть відфільтровані і в результаті маємо лише один прямокутник.

Скомпонуємо всі попередні алгоритми видалення нетекстових областей та отримаємо регіони, які містять лише символьну інформацію, що представлено на наступному зображенні 3.14:

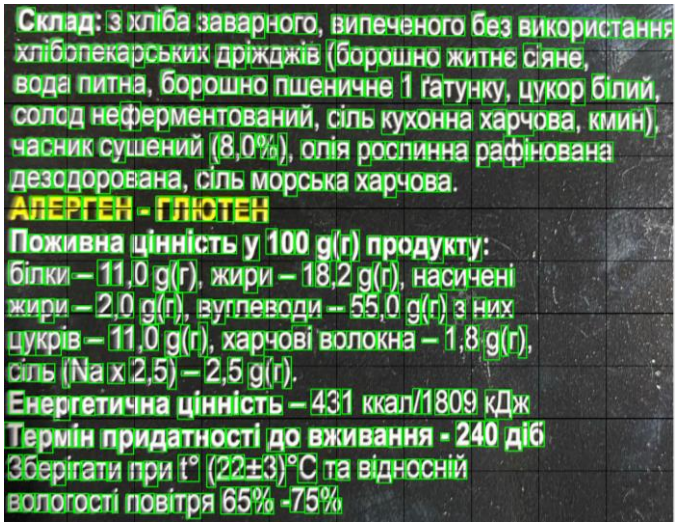


Рисунок 3.14 – Результат роботи алгоритму видалення нетекстових областей

3.3 Розпізнавання виявлених регіонів символів

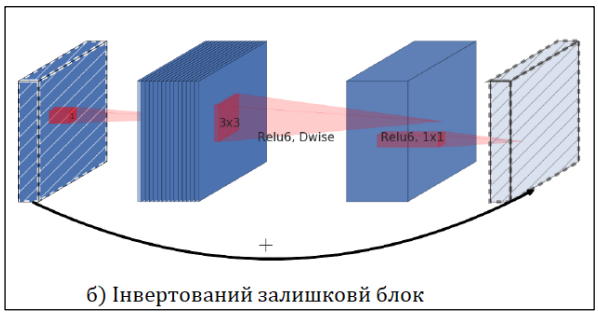
Виявлена множина текстових регіонів є множиною матриць зображень з символами, які, кожен окремо, поступають на вхід нейронній мережі для розпізнавання. Для розпізнавання символів в розробленій мобільній системі використовується архітектура мережі MobileNetV2. Дана нейронна мережа на сьогодні є найкращим варіантом у задачах класифікації зображень мобільними додатками, для яких важливою вимогою є робота за відсутності з'єднання з мережею, а отже, які не мають серверної частини, а отже великих обчислювальних ресурсів.

MobileNetV2 використовує архітектуру попередника MobileNet, яка ґрунтується на ідеї використання легковагових роздільних по глибині згорток (англ. depthwise convolution). MobileNetV2 також використовує роздільні по глибині згортки та розширює свого попередника, вводячи інвертовані залишкові блоки [61].

Основна ідея роздільних по глибині згорток полягає в заміні повного згорткового оператора на факторизовану версію, яка розділяє згортку на два окремі шари. Перший шар називається глибинною згорткою (англ. depthwise), він виконує легку фільтрацію, застосовуючи один згортковий фільтр на вхідний канал. Другий шар — це згортка розміром 1 × 1, яка називається точкової (англ. pointwise) згорткою, яка відповідає за створення нових функцій за допомогою обчислення лінійних комбінацій вхідних каналів [56].

Інвертований залишковий блок, який іноді називають блоком MBConv, є типом залишкового блоку, який використовується для моделей для розпізнавання зображень, який використовує, задля ефективності, інвертовану структуру, запропонований для архітектури MobileNetV2 та оптимізований для мобільних пристроїв [56]. Традиційний залишковий блок має структуру, в якій відбувається чергування широкий-вузький-широкий блоки. Вхід має велику кількість каналів, які згортаються згорткою 1x1. Потім кількість каналів знову збільшується за допомогою згортки 1x1, щоб можна було додати вхід і вихід.

На відміну від цього, інвертований залишковий блок дотримується підходу вузький-широкий-вузький, тобто, інверсії (рис. 3.15). Спочатку йде розширення з допомогою згортки 1x1, після чого згортка 3x3 по глибині (що значно зменшує кількість параметрів), потім – згортка 1x1, для зменшення кількості каналів, щоб можна було додати вхід і вихід [56].



б) Інвертований залишковий блок

Рисунок 3.15 – Інвертований залишковий блок

Архітектура MobileNetV2 містить початковий шар з повною згорткою з 32 фільтрами, за яким слідують 19 залишкових вузьких (англ. bottleneck) шарів (рис. 3.16).

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Рисунок 3.16 – Архітектура MobileNetV2 [61]

На рисунку вище можна побачити, як розташовані блоки вузьких місць. t позначає швидкість розширення каналів, c представляє кількість вхідних каналів, а n, як часто повторюється блок. Перший шар кожної послідовності має крок s, а всі інші використовують крок 1. Усі просторові згортки використовують ядра 3×3, а функцією активації є ReLU6 [56].

Тренування моделі відбувається на навчальному наборі, який складається з 200 тисяч тренувальних зображень та 50 тисяч тестових. Оскільки метою створення системи оцінки шкідливості тексту є сканування складу продуктів, представлених

на українському ринку, навчальний набір повинен містити літери українського алфавіту. У своїй роботі ми використали власний навчальний набір, згенерований власноруч (див. розділ 4). На даному навчальному наборі модель показує точність 99.51%, а отже є ефективною при розпізнаванні символів. На виході отримується масив літер, які були розпізнанні з множини регіонів.

3.4 Алгоритм складання слів

Маючи множину прямокутників, які містять зображення символів, необхідно об’єднати їх у слова. Для вирішення даної задачі необхідно 1) вирівняти зображення таким чином, щоб вирівнявся текст, присутній на даному зображенні; 2) виділити рядки; 3) скласти регіони з символами і окремі регіони слів (рис. 3.17).

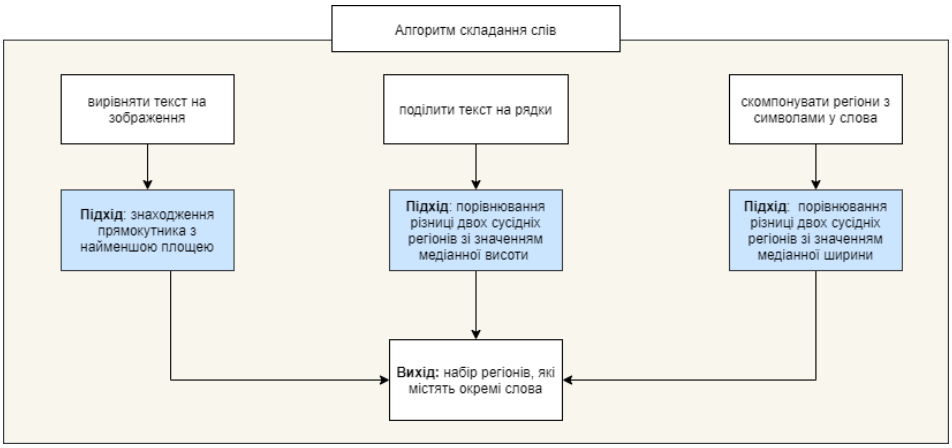


Рисунок 3.17 – Алгоритм складання слів

Вирівнювання тексту на зображення здійснюється використанням методу вирівнювання тексту знаходженням прямокутника з найменшою площею (англ. min area rectangle) [10]. Для цього необхідно визначити блок з текстом на зображенні по крайнім точкам регіонів з символами; вирахувати кут нахилу тексту; повернути зображення на отриманий кут для вирівнювання.

Вираховування прямокутника з найменшою площею (англ. min-area rectangle) визначається обчисленням площі обмежувального прямокутника тексту обертанням його на різний кутами нахилу у великому діапазоні. Обчислення відбувається в два етапи. Під час першої ітерації прямокутний обертається від α_{min}

до α_{max} градусів і з певним розміром кроку $\Delta\alpha$ та обчислюється оцінка орієнтації прямокутника, тобто вираховується площа прямокутника, на даному куті.

Під час другої ітерації знаходиться кінцева орієнтація прямокутника, обчисленням площі прямокутника обертанням його в діапазоні між кутом прямокутника з найменшою площею та другим за найменшою площею сусіднім прямокутником.

Алгоритм виглядає наступним чином:

1) нехай задано α_{min}, α_{max} та $\Delta\alpha$, X – набір точок тоді що утворюють опуклу оболонку (англ. convex hull) набору регіонів, точкою центру координат приймемо точку центра мас опуклої оболонки, що утворена X. Встановимо значення *rotAngle* як початкове (α_{min}), *minArea* дорівнює площі поточного обмежувального прямокутника;
2) для кожного значення α від α_{min} до α_{max} з кроком $\Delta\alpha$ при $\alpha \neq 0$ виконаємо крок 3 та крок 4;
3) нехай існує пряма m ($y = tg(\alpha)x + b_m$) та пряма n ($y = -1/tg(\alpha)x + b_n$) тоді площа обмежувального прямокутника при куті α вираховується за формулою:

$$area = (h_1 - h_2) * (h_3 - h_4) * |cos\alpha * sin\alpha| \quad (3.15)$$

де h_1– це максимум b_m, а h_2 – мінімум b_m, при яких пряма n проходить через x є X, h_3 та h_4 – максимум та мінімум b_n, при яких пряма m проходить через $x \in X$;

4) якщо $area < minArea$, тоді $minArea = area$, а $rotAngle = \alpha$;
5) відтак, кут повороту регіону з текстом дорівнює $rotAngle$ [64].

На рисунку 3.18 представлено знаходження кута, при якому площа прямокутника досягає найменшого значення.

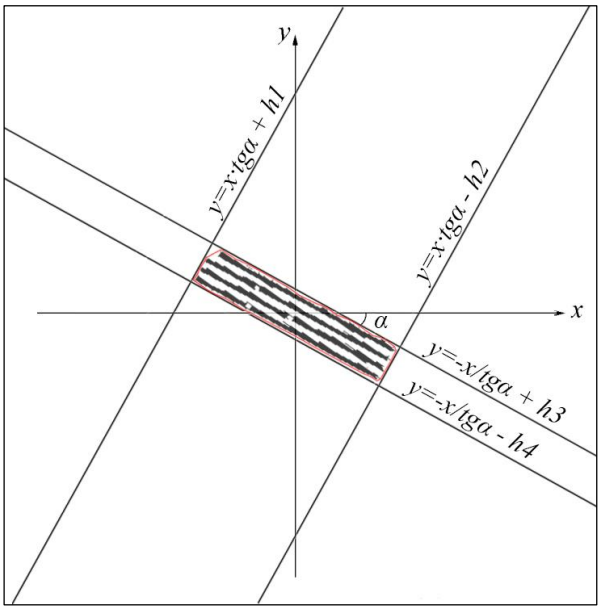


Рисунок 3.18 – Обчислення мінімальної площі прямокутника

Для оцінки точності орієнтація прямокутника знаходиться обчисленням кута нахилу найменшої інерції за наступною формулою:

$$\theta = \frac{1}{2}\tan^{-1}\frac{2\sum_{i=1}^{n}\sum_{j=1}^{m}(x-\bar{x})(y-\bar{y})B[i,j]}{\sum_{i=1}^{n}\sum_{j=1}^{m}(x-\bar{x})^2B[i,j]-\sum_{i=1}^{n}\sum_{j=1}^{m}(y-\bar{y})^2B[i,j]} \quad (3.16)$$

де $\bar{x}$ та $\bar{y}$ є координатами центру мас регіону, $B[i,j]$ є значенням 1 для пікселя на межі регіону, в іншому випадку – 0 [64].

Вирівнявши текст на зображенні, необхідно поділити текст на зображенні на рядки. Якщо подивитись на обмежувальні прямокутники, то можна помітити, що їх висота відрізняється, тому висота рядка математично визначається на основі інформації про розмір кожного обмежувального прямокутника. Для знаходження висоти рядку використаємо метод знаходження половини медіани всіх висот обмежувальних прямокутників. Для цього всі обмежувальні прямокутники сортуються по висоті, після чого знаходиться медіанна висота, як середина всіх висот: $H = \{h_1, h_2 \dots, h_n\}$, де n – кількість всіх висот, тому індекс медіанного значення дорівнює $m = \frac{n}{2}$. Значення медіани v дорівнює $h_m/2$, де m індекс медіанного елементу.

Далі регіони, які містять зображення символу, необхідно згрупувати, враховуючи певний інтервал для координат y (значення половини медіани v). Регіони сортуються по координаті y, після чого обмежувальні прямокутники групуються порівнюючи їх по два по координаті y. Відтак, нехай регіон c_i порівнюється з попереднім, тоді якщо рівність $|c_{i-1} - c_i| \leq v$ виконується регіон групується в поточний рядок, а якщо ні, тобто координата поточного регіону є значно більшою попереднього – цей регіон належить до наступного рядка [36].

Відтак, після групування кожна група регіонів, яка дорівнює рядку перебирається та визначаються мінімальні та максимальні значення: min x, min y, max x, max y, які необхідні для моделювання обмежувального прямокутника, який включає всі обмежувальні прямокутники з регіонами-символами даного рядка. Псевдокод алгоритму наданий нижче.

Алгоритм 2: Групування регіонів з символами у рядки

input: множина обмежувальних прямокутників регіонів з символами

output: множина обмежувальних прямокутників рядків

begin

H = {}

for всіх прямокутників $c \in C$

додати висоту h_c обмежувального прямокутника до H

end for

відсортувати список H за зростанням;

розрахувати $v = h_m/2$, де $m = \frac{n}{2}$;

відсортувати C по координаті y за зростанням;

$L = \{\}$

$l_{current} = \{\}$

foreach всіх прямокутників $c \in C$

if $\left| c_{prev} - c_{current} \right| \leq v$

додати до поточного списку прямокутників рядку $l_{current}$

else

очистити $l_{current}$

додати поточний $c_{current}$ до $l_{current}$

end if

end foreach

for кожної координати списку $l_{current} \in L$

$y_{min} = \min(l_{current})$;

$x_{min} = \min(l_{current})$;

$y_{max} = max(l_{current})$;

$x_{max} = \max(l_{current})$;

$\text{rect} = (x_{min}, y_{min,} x_{max}, y_{max})$

додати $rect$ у список $R = \{\dots\}$;

end for

end

Блок-схема алгоритму представлена у Додатку Б. Результат роботи алгоритму представлений на рис. 3.19:

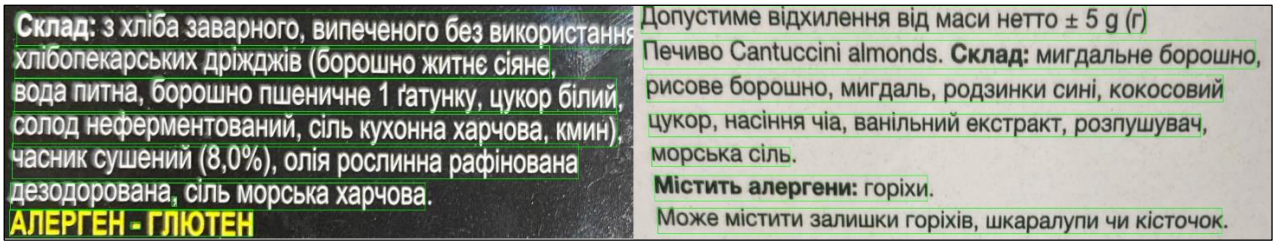


Рисунок 3.19 – Результат роботи алгоритму складання регіонів з символами в рядки

Об’єднання регіонів в слова відбувається схожим чином до Алгоритму 2. Нехай $\mathrm{C} = \{c_1, c_2, \dots, c_n\}$ є множиною регіонів з символами. Слова групуються по математичним властивостям ширини регіону з символом. Для знаходження ширини слова використаємо метод знаходження відстані між обмежувальними прямокутниками: значення четвертини медіани v дорівнює $w_m/4$, де m індекс медіанного елементу. Регіони сортуються по координаті x, після чого, знайшовши значення четвертини медіани ширини, тобто, порогове значення з яким

порівнюється різниця двох сусідніх регіонів, регіон c_i порівнюється з попереднім, і якщо рівність $|c_{i-1} - c_i| \leq v$ виконується, регіон групується в поточний слово, а якщо ні, тобто координата поточного регіону є значно більшою попереднього, що говорить про наявність пробілу і тд. – регіон додається до списку наступного слова. В результаті отримуємо координати та просторові розміри обмежувальних прямокутників, які містять згруповані символи у слова.

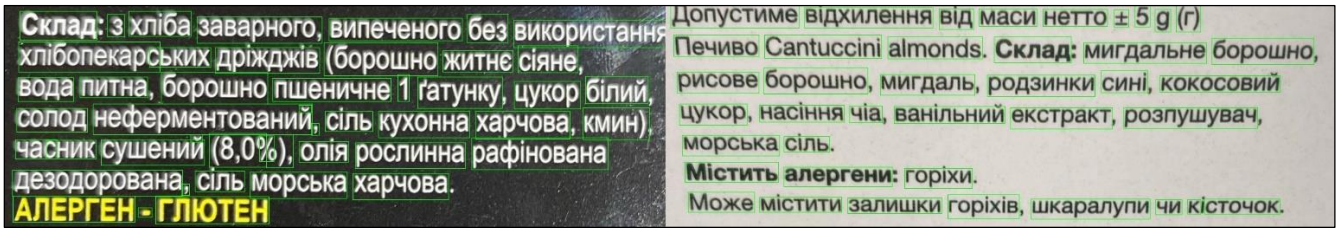


Рисунок 3.20 – Результат роботи алгоритму складання слів

Відтак, як видно з рисунку 3.20, всі символи ідеально погруповані в слова.

3.5 Алгоритм аналізу та оцінки шкідливості харчових продуктів за його складом

Оскільки наше програмне забезпечення має на меті проаналізувати всі інгредієнти складу харчового продукту, які представлені на упаковках різноманітних товарів, найкращим рішенням буде створення такого алгоритму, який міг би на вході брати до уваги всі зауваження щодо кожного з інгредієнтів, а також зауваження користувача системи щодо загроз, обмежень, алергій та непереносимостей та видавати на виході індивідуальну оцінку шкідливості продукту для певного користувача. Блок-схема алгоритму представлена в Додатку А, загальна структура даного алгоритму представлена на рисунку 3.21:

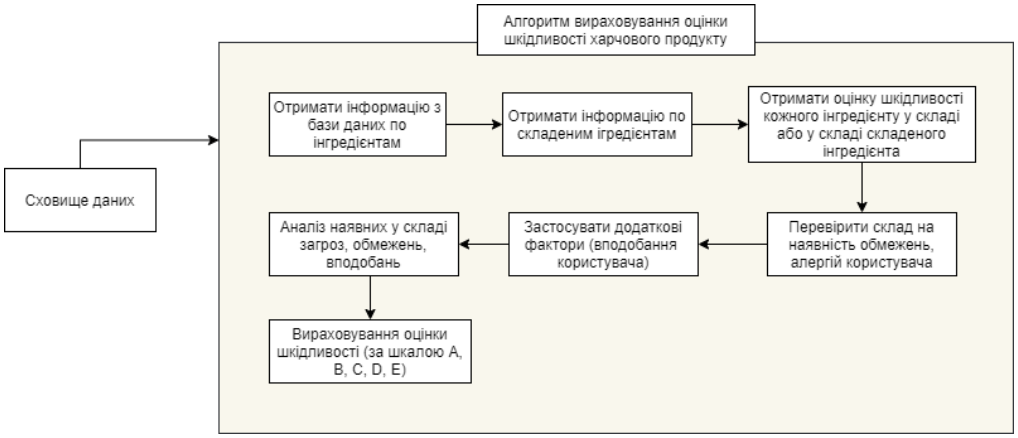


Рисунок 3.21 –Алгоритм аналізу та оцінки шкідливості харчових продуктів за його складом

Перш за все необхідно алгоритмічно отримати з бази даних інформацію про кожен інгредієнт (ми називатимемо елемент складу продукту складником або інгредієнтом): а) загальну інформацію, яка буде представлена для кожного інгредієнту – опис, функцію, тип виробництва, характеристики, тип походження; б) інформація, яка необхідна для формування оцінки – набір загроз даного складнику. Дана інформація повинна бути отримана заразом і по всіх інгредієнтах композитного інгредієнта (якщо такі є). Нехай набір інгредієнтів представлений як $C = \{c_1, c_2, \dots, c_n\}$, тоді кожен інгредієнт, якщо є композитним, тобто такими, до складу яких входять інші інгредієнти, які також мають свій рівень загроз, представляється як граф, де складові інгредієнти представлення як $c_l = \{c_{l_1}, c_{l_2}, \dots, c_{l_n}, c_l \subset C\}$.

Відтак, отримання кожного інгредієнту, який входить до складу композитного представлено наступною блок-схемою алгоритму:

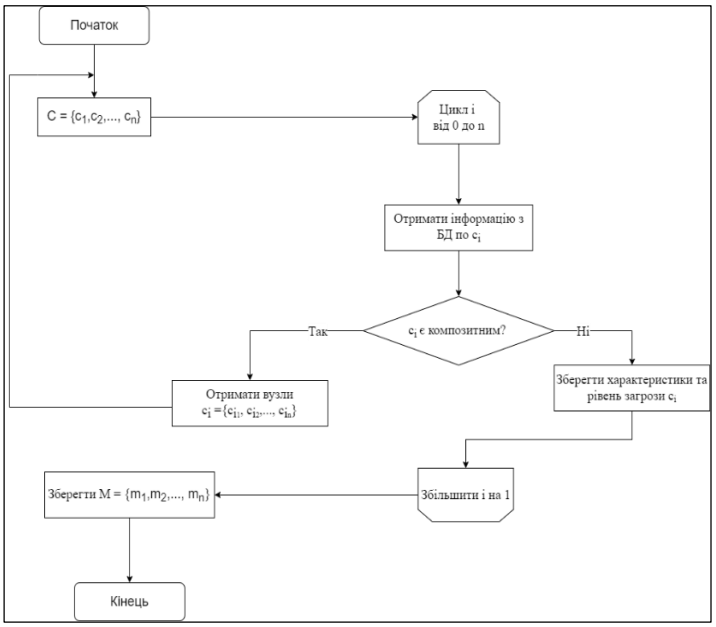


Рисунок 3.23 – Блок-схема алгоритму отримання характеристик інгредієнтів

Отже, якщо інгредієнт є композитним, то необхідно зробити обхід дерева елементів, з якого складається поточний інгредієнт та рекурсивно вичитати всі загрози дерева інгредієнтів. Ці кроки необхідні задля уникнення дублювання інформації на рівні бази даних, а тому для композитного інгредієнту зберігається лише загрози, які утворились при його створенні з дочірніх інгредієнтів. Після цього маємо множину інгредієнтів M, які входять до складу поточного продукту $M = \{m_1, m_2, \ldots, m_n\}$, де m_i – це інгредієнт з його характеристиками (описом та рівнем загроз).

Наступною задачею є перевірка на наявність у наборі інгредієнтів M інгредієнти, які є загрозливими для користувача, або на які у нього є обмеження. Для цього необхідно отримати з бази даних внесені користувачем обмеження $R_c = \{r_1, r_2, \ldots, r_n\}$, де R_c – перелік обмежень, алергій, заборон і т.д. користувача щодо їжі та інгредієнтів.

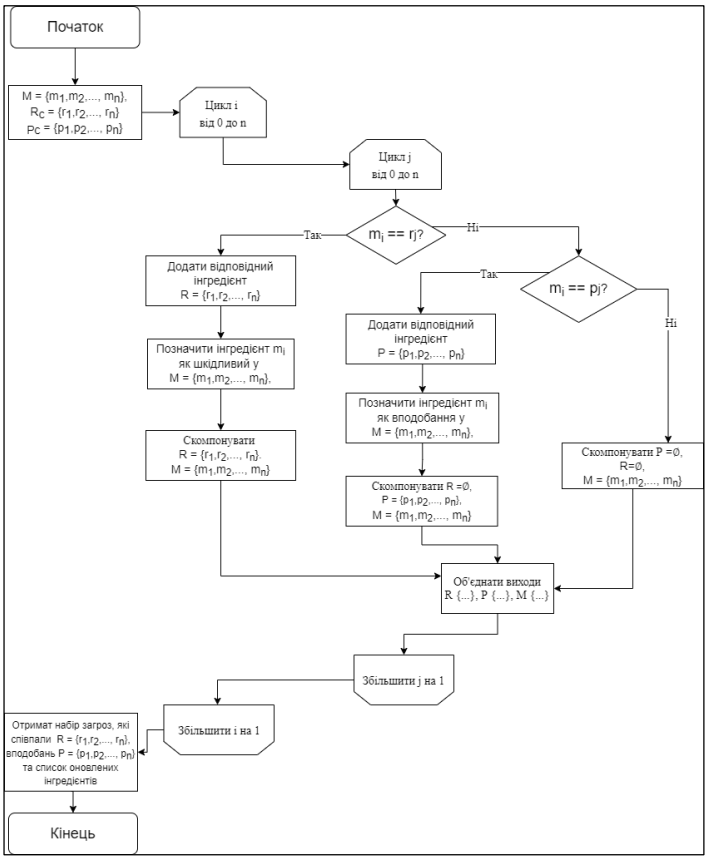


Рисунок 3.24 – Блок-схема перевірки наявності загрозливих для користувача інгредієнтів або вподобань у складі

Загрозливі інгредієнти порівнюються з наявними у складі та зберігаються. Таким чином $R = \{r_1, r_2, \ldots, r_n\}$, будуть містити всі інгредієнти зі складу, які збігаються з обмеженнями користувача $R_c = \{r_1, r_2, \ldots, r_n\}$. Заразом, порівнюються інгредієнти зі списку вподобань користувача та інгредієнтами складу $P_c = \{p_1, p, \ldots, p_n\}$. Поточний інгредієнт m_i, який дорівнює p_j, позначається як такий, що входить до вподобань користувача та зберігаються як $P = \{p_1, p, \ldots, p_n\}$.

Якщо поточний інгредієнт m_i, дорівнює r_j, тоді показнику загрози для користувача $y = 0$ встановлюється найвищий рівень загрози $y = 6$, інгредієнт позначається як шкідливий для користувача.

На виході маємо $R = \{r_1, r_2, \ldots, r_n\}$ або $R = \emptyset$; $P = \{p_1, p, \ldots, p_n\}$ або $P = \emptyset$; оновлений або незмінний список інгредієнтів $M = \{m_1, m_2, \ldots, m_n\}$.

Останній крок – це проходження по всіх рівнях загроз та вираховування оцінки шкідливості. Для оцінки шкідливості ми обрали формулу середнього

арифметичного зваженого, для всіх дійсних чисел x_1, x_2,…x_n з ваговими коефіцієнтами w_1,w_2,…w_n та визначається як:

$$x = \frac{x_1 w_1 + x_2 w_2 + \cdots + x_n w_n}{w_1 + w_2 + \cdots + w_n} \quad (3.17)$$

Було прийнято ваговими коефіцієнтами: $w = 1$ для рівня 1, 2; $w = 2$ для рівня 3, $w = 3$ для рівня 4, $w = 4$ для рівня 5. Блок-схема алгоритму представлена у графічних матеріалах Додатку Б.

На виході ми отримуємо оцінку шкідливості продукту (від одиниці до шести, яка конвертується в буквену, де $1 = A$, $5 = F$) на основі проаналізованих інгредієнтів. Якщо максимальний рівень загрози 6, тоді склад отримує статус «продукт не підходить під вподобання користувача».

Висновки до розділу

Алгоритмічне забезпечення для системи інтелектуального виявлення та аналізу складу харчових продуктів необхідне для збільшення швидкодії інтелектуальної обробки тексту на матриці зображення, а також для аналізу та оцінки шкідливості складу. Алгоритм MSER, який використовується для виявлення контрастних регіонів, базуючись на різниці контрастності області та її границі, а також максимальній стабільності площі екстремального регіону визначає максимально стабільні екстремальні області, які є об’єктами зображення. Розроблений алгоритм для видалення нетекстових регіонів, базується на визначенні ширини штрихової лінії об’єкту регіону, метриці перетину над об’єднанням, використанню евристичних правил. Для розпізнавання виявлених символів-регіонів було застосовано архітектуру нейронної мережі MobileNetV2, яка показала точність розпізнавання на власному навчальному наборі 99.51%. Групування текстових контрастних регіонів у слова та рядки, відбувається шляхом визначення значення медіани висоти та ширини регіону. Також було розроблено алгоритмічне забезпечення для вираховування оцінки шкідливості складу харчового продукту.

4 ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Архітектура інтелектуальної системи визначення та оцінки шкідливості складу

Метою інтелектуальної системи визначення та аналізу складу продуктів є забезпечення користувачів інструментом, який буде інтелектуально визначати текстову інформацію складу та автоматично вираховувати оцінку шкідливості складу.

Розроблена система реалізує гібридну архітектуру, яка поєднує в собі переваги клієнт-серверної та клієнтської архітектури та бореться з недоліками кожної з них.

При використанні методів машинного навчання для обробки будь-якого типу інформації мобільними системами, програмний модуль, який містить в собі компонент обробки інформації розташовується здебільшого на частині серверу для збільшення швидкодії роботи. Попри високу швидкість, використання клієнт-серверної архітектури в розробці мобільних додатків має велику кількість недоліків таких, як проблема постійного доступу до даних, конфіденційність, відсутність доступу до системи при відсутності мобільного з'єднання. Також великою проблемою може стати масштабованість (рис. 4.1).

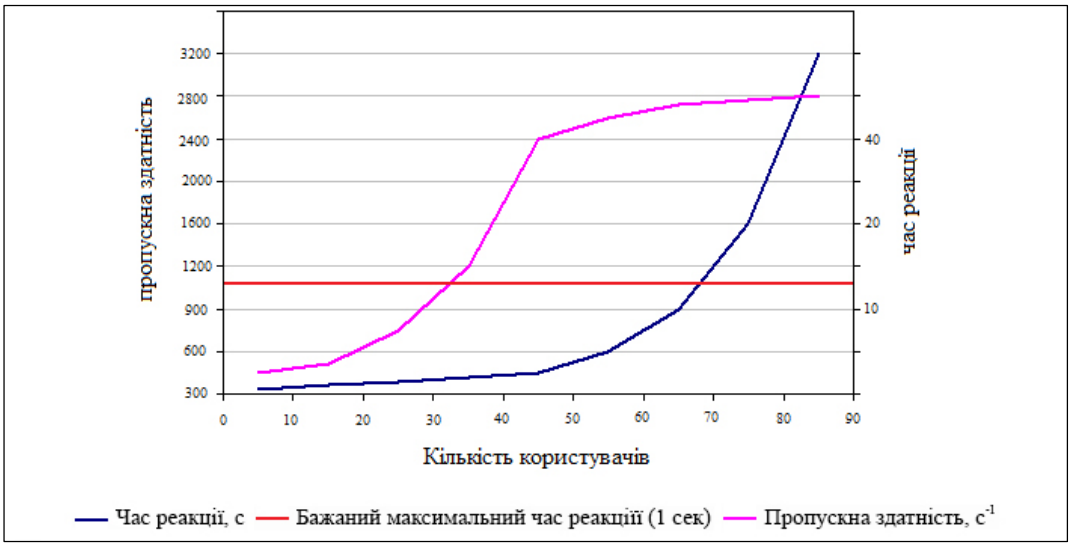


Рисунок 4.1 – Проблема масштабованості клієнт-серверної архітектури, що відображена графіком залежності часу реакції/пропускної здатності від кількості користувачів

На рисунку 4.1, можна бачити, що час реакції має експоненціальну залежність від кількості користувачів, а залежність пропускної здатності, після досягання свого максимуму при 100%-му навантаженні серверу, набуває характеру сталої функції. А, отже, явно проявляється проблема масштабованості, притаманна клієнт-серверній архітектурі.

Вирішити проблему можна перенесенням всіх основних обчислювальних операцій на сторону клієнта. При цьому виникає перелік інших проблем: зменшення швидкодії розпізнавання тексту на зображенні та його аналізу через брак обчислювальних потужностей; відсутність надійного збереження інформації користувача про здійснені ним можливі операції та ін.; можливість оновлення даних локальної бази даних лише при оновленнях всієї системи.

Оскільки наша система дозволяє користувачеві вводити перелік загроз, обмежень чи алергій, тобто, ключові дані користувача, які не хочеться втрачати та вводити знову, є велика необхідність збереження даної інформації при виході з системи тощо. Також, іншою проблемою може стати предметна область, взаємодія якої покладається на використання словникових даних, що вимагають частого оновлення. До прикладу, у розробленій системі оцінки шкідливості складу продуктів, інформація про інгредієнти, які входять до складу, повинна підлягати регулярній підтримці. Дану проблему можна вирішити використанням додаткового зовнішнього сховища даних.

Та при класичному серверному підході оплата за обчислювані ресурси не залежить від навантаження, в той час, у нашій системі навантаження є низьким та не регулярним, нема потреби оплачувати серверний час.

Задля оптимізації матеріальних витрат постає необхідність використання безсерверної (англ. serverless) архітектури для даних цілей [78]. Поєднання використання переваг клієнтської архітектури для обчислень та безсерверної архітектури для збереження даних дає можливість отримати все краще з від двох традиційних архітектур. Архітектура на прикладі реалізації для мобільної інтелектуальної системи представлена на рис 4.2.

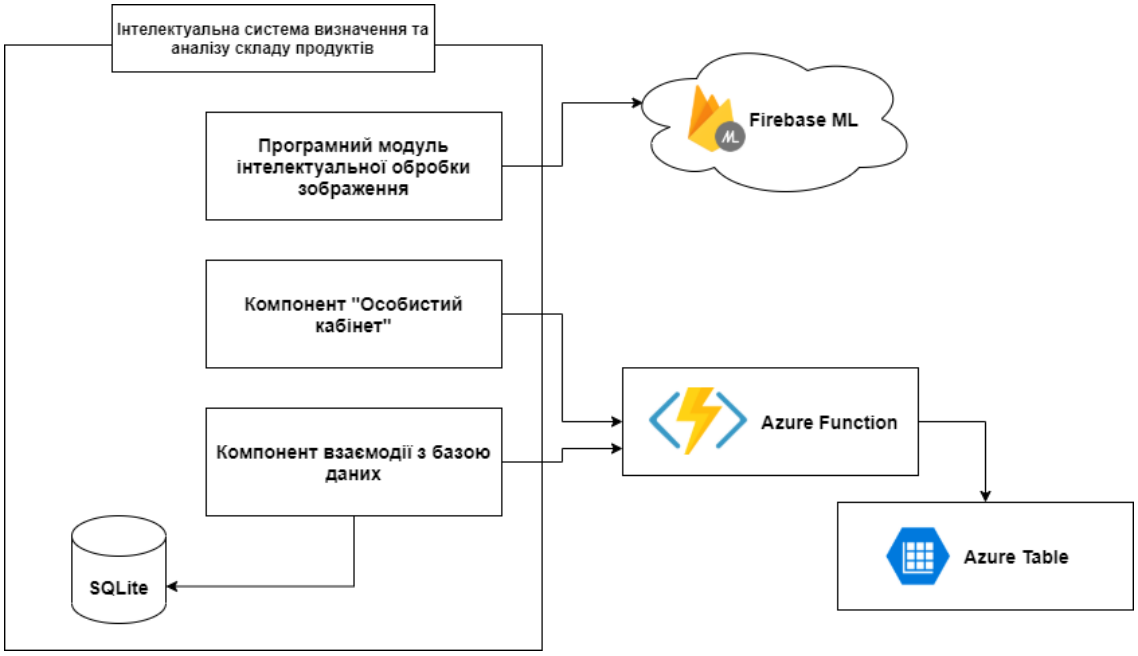


Рисунок 4.2 – Гібридна архітектура мобільної інтелектуальної системи визначення та аналізу складу продуктів

Запропонована гібридна архітектура системи має наступні складові:

- безсерверне середовище, яке дозволяє матеріально ефективно обробляти невеликі навантаження;
- зовнішнє сховище даних для зберігання великої кількості неструктурованих даних;
- клієнт, який може тривалий час підтримувати функціональність, знаходячись у ізоляції (від решти системи, при відсутності мобільного зв'язку тощо);
- зовнішній репозиторій для моделей нейронних мереж, що дозволяє розповсюджувати оновлення моделі, не вимагаючи оновлення всього застосунку.

Одним з типів компонентів безсерверних застосунків є безсерверні функції (англ. serverless functions), які розміщуються та керуються за допомогою типу служби «Функція як послуга» (англ. function as a service) або FaaS. За допомогою FaaS надається можливість написання незалежних сценаріїв коду, відомих як «функції», та завантаження цих функцій в FaaS. Після чого функціонал викликається ініціацією певної події або запускається за розкладом [78]. Безсерверна архітектура дає можливість балансувати навантаження в міру

збільшення або зменшення навантаження на програму. Користувачеві потрібно буде платити лише за ресурси, які споживає запущена програма.

Azure Functions – це хмарна служба типу безсерверних функцій, яка на вимогу надає актуальну інфраструктуру та ресурси, необхідні для забезпечення роботи користувацьких застосунків [7]. Наше програмне забезпечення споживає кількість обчислювальних ресурсів, наданих Azure Function залежно від навантаження та виступають у ролі посередника між клієнтським мобільний додатком та сховищем даних. Таким чином мітигується ризик отримання зловмисниками прямого доступу до сховища даних шляхом вичитування конфігураційних файлів.

Azure Storage — це хмарна служба, керована Microsoft, яка забезпечує високодоступне, безпечне, довговічне, масштабоване та резервне сховище. У Azure є два типи облікових записів зберігання, чотири типи сховища, чотири рівні резервування даних і три рівні для зберігання файлів [8]. Для потреб розробленої системи найкраще підходить тип сховища Azure Table Storage.

Azure Table Storage є службою, яка зберігає структуровані дані NoSQL всередині хмари, створюючи сховище атрибутів із безсхемним дизайном. Оскільки сховище таблиць немає схем, дані легко адаптуються, до змін програми. Доступ до даних сховища таблиць швидкий і рентабельний для кількох типів додатків і зазвичай нижчий, ніж традиційний SQL для подібних обсягів знань.

Використання сховища таблиці в реальному часі включає: зберігання наборів даних, які не вимагають складних об’єднань, зовнішніх ключів або збережених процедур і можуть бути денормалізовані для швидкого доступу; швидкий запит даних за допомогою кластерного індексу; доступ до даних за допомогою протоколу OData і запитів LINQ за допомогою бібліотек .NET [8].

Firebase ML – це інструмент, який спрощує інтеграцію моделей нейронних мереж в мобільний застосунок. Firebase ML дозволяє завантажувати навчену модель у консоль Firebase та зв’язувати з конкретним програмним продуктом. Для розробки нашої системи корисно мати можливість динамічно оновлювати моделі, коли застосунок залишається не оновленим, наприклад, покращення точності.

Заразом, використовуючи Firebase ML з Remote Config, є можливість завантажувати різні моделі для різних сегментів користувачів, після чого з допомогою A/B-тестування проводити експерименти, щоб знайти найкращу модель [26].

4.2 Структура розробленої інтелектуальної системи

У розробленому програмному забезпеченні клієнтом виступає інтелектуальна система для визначення та аналізу складу харчових продуктів. Клієнтська частина в своє чергу має багатошарову архітектуру та складається з наступних рівнів: рівень представлення, рівень бізнес-логіки, рівень інфраструктури, рівень платформи та рівень даних (рис. 4.3).

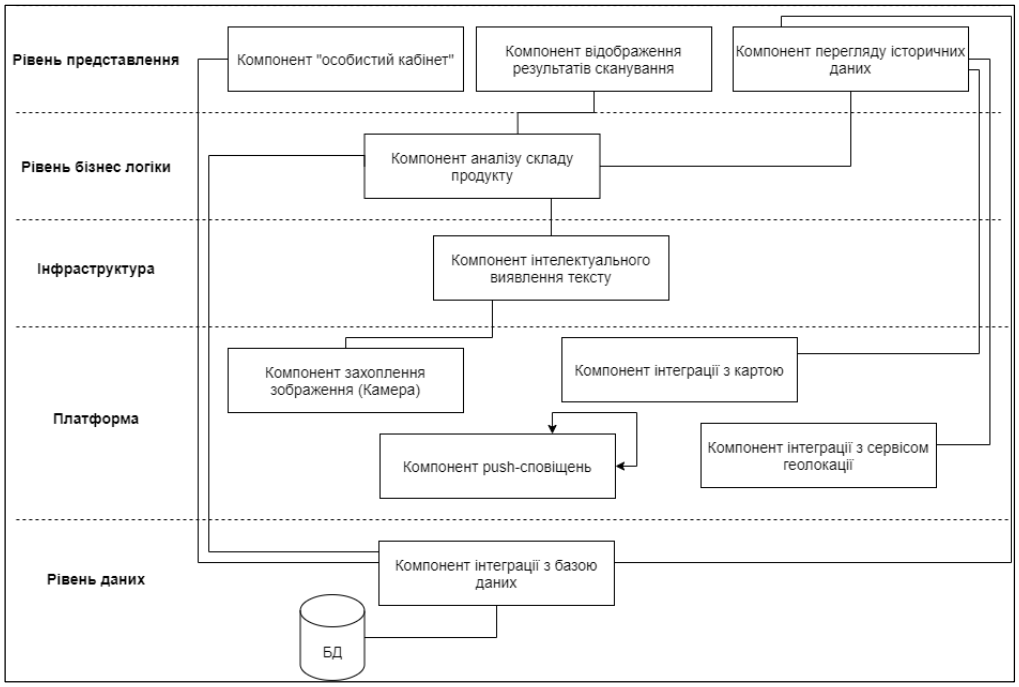


Рисунок 4.3 – Багатошарова структура клієнтської частини

4.2.1 Рівень представлення

Взаємодія рівня презентації відбувається з використанням архітектурного шаблону MVVM (Model-View-ViewModel). MVVM – це архітектура розробки програмного забезпечення, яка має на меті усунути тісний зв'язок між кожним компонентом та розділити застосунок на три незалежні шари. Відтак, елементами керування інтерфейсу користувача та логіка можуть бути чітко розділені для легшого тестування та обслуговування, компоненти не знають явно один про

одного. Компоненти відокремлені один від одного, що дозволяє заміняти компоненти; змінити внутрішню реалізацію, не впливаючи на інше; працювати над компонентами незалежно; можливість ізольованого модульного тестування (рис 4.4). З допомогою механізму зв'язування даних (англ. binding) компоненти взаємодіють між собою [15].

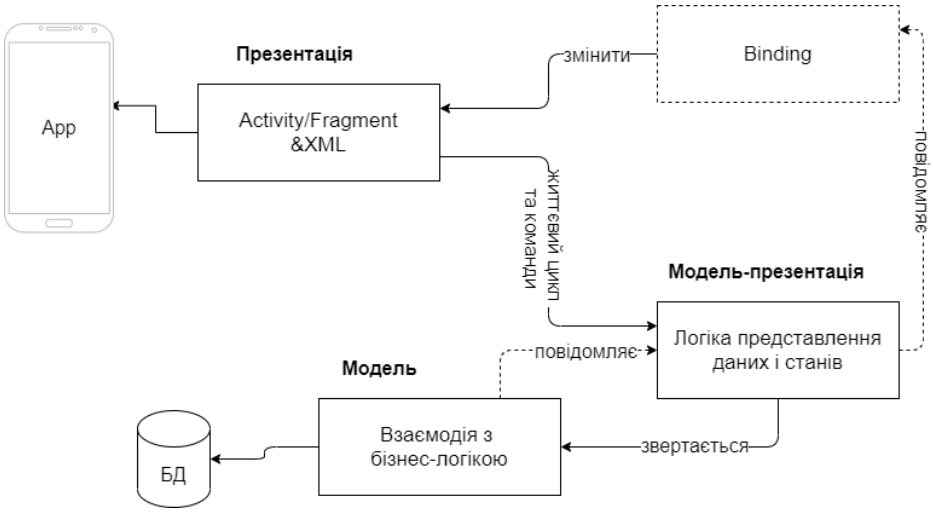


Рисунок 4.4 – Архітектура MVVM

Представлення (View) відповідає за визначення структури, макета та зовнішнього вигляду того, що користувач бачить на екрані (рис. 4.4). Представлення складається з коду інтерфейсу користувача (Activity, Fragment), визначається з допомогою XML та не містить коду бізнес-логіки. Під час виконання представлення змінюється тоді, коли UI елементи реагують на властивості моделі-представлення, викликаючи події сповіщень про зміни [70]. View надсилає інформацію про дії користувача у ViewModel, але не отримує відповідь безпосередньо. Щоб отримати відповідь, представлення повинне бути підписаним на спостережувані елементи, які ViewModel надає йому.

Модель **(**Model) представляє реалізацію моделі предметної області програми, яка включає модель даних разом із бізнес-логікою та логікою валідації. Приклади об'єктів моделі включають репозиторії, бізнес-об'єкти, об'єкти передачі даних (DTOs), а також згенеровані об'єкти та проксі-об'єкти.

Модель-представлення (ViewModel) – це міст між представленням і моделлю (бізнес-логіка), який відповідає за підготовку та управління даними для Activity або

Fragment, а також обробляє взаємодію Activity/Fragment з іншою частиною застосунку (наприклад, комунікує з класами бізнес-логіки) (рис 4.4). ViewModel призначена для зберігання та керування даними, пов’язаними з інтерфейсом користувача, з урахуванням життєвого циклу [76]. ViewModel є ключовою частиною тріади, оскільки саме вона відділяє презентацію від моделі та діє як зв’язок між ними. Модель представлення взаємодіє з моделлю, викликаючи методи в класах моделі. Потім модель представлення надає дані з моделі у формі, яку представлення може легко використовувати. Модель представлення отримує дані з моделі, а потім робить дані доступними для представлення. Модель представлення також забезпечує реалізації команд, які користувач програми ініціює у представленні.

Рівень представлення системи включає три компоненти: компонент «особистий кабінет», компонент відображення результатів сканування. Компонент особистий кабінет надає користувачу можливість для введення особистих даних таких як обмеження, алергії, проблеми зі здоров’ям, вподобання, що буде враховуватись при оцінці шкідливості харчового продукту. Компонент відображення результатів сканування є компонентом представлення, який відображає оцінку шкідливості продукту та перелік продуктів з їх характеристиками. Компонент відображення результатів сканування взаємодіє з компонентом аналізу складу продуктів та відображає результат оцінки шкідливості продукту.

4.2.2 Рівень бізнес-логіки та платформи

Рівень бізнес-логіки включає компонент аналізу складу продуктів та представляє компонент алгоритмічного забезпечення системи – алгоритм прийняття рішень, представлений у розділі алгоритмічного забезпечення (див. розділ 3.5). На базі вилученого тексту із зображення алгоритм робить аналіз інгредієнтів та вираховує оцінку шкідливості продукту.

Рівень платформи включає взаємодію рівнів з платформними механізмами для зручності користування додатком. Компонент push-сповіщень відповідає за надсилання користувачеві сповіщень-нагадувань. Компонент захоплення зображення взаємодіє з камерою та з допомогою камери отримує зображення.

4.2.3 Рівень даних

Рівень даних включає компонент інтеграції з локальним сховищем даних та надає механізми отримання даних. Локальне збереження даних забезпечує кешування відповідних фрагментів даних, для отримання вмісту даних у автономному режимі користувачем при відсутності доступу до мережі.Вибір сховища даних в результаті впав на локальну базу даних SQLite [42]. SQLite — одна з найбільш поширеніших СУБД у світі, що повністю підтримується на ключових мобільних платформах: IOS, Android, IpadOS.

Оскільки SQLite є механізмом реляційної бази даних, більшість фреймворків збереження, розроблених для SQLite, є фреймворками ORM, наприклад, ROOM. Однією з основних функцій такого об’єктно-реляційного відображення (ORM) та об’єктно-орієнтованих структур є вирішення невідповідності об’єктно-реляційного імпедансу між моделлю об’єктно-орієнтованого програмування та моделлю роботи реляційної бази даних [42].

Фреймворк Room – це Java бібліотека, яка забезпечує рівень абстракції над SQLite, щоб забезпечити вільний доступ до бази даних, використовуючи всю потужність SQLite. Зокрема, Room надає такі переваги: перевірка SQL-запитів під час компіляції; зручні анотації, які зводять до мінімуму повторюваний і схильний до помилок шаблонний код; оптимізовані шляхи міграції бази даних [66].

Room включає три основні компоненти:

- клас бази даних, який містить базу даних і слугує основною точкою доступу для базового підключення до постійних даних вашої програми;
- сутності (entity), які представляють таблиці в базі даних системи;

- об’єкти доступу до даних (DAO), які надають методи для запитів, оновлення, вставки та видалення даних у базі даних [66].

4.2.3.1 Аналіз предметної області

База даних створюється для інформаційного обслуговування користувачів мобільного додатку що розпізнає та аналізує склад продуктів харчування.

БД повинна містити дані:

- про інгредієнти продуктів харчування, їх відношення між собою (інгредієнт може складатись з інших інгредієнтів);
- детальну класифікацію інгредієнта (інформація про природу походження інгредієнту, його важливість для підтримки здорового стану організму людини або навпаки рівень шкідливості для людей з певними обмеженнями чи загалом);
- інформацію про захворювання користувача, які змушують уникати певних інгредієнтів чи груп інгредієнтів;
- перелік інгредієнтів, що викликають у користувача алергічну реакцію;
- дані про вподобання та неприязнь користувача по відношенню до певного інгредієнту чи групи інгредієнтів;
- дані, що відповідають за виконані сканування;
- дані що описують місце проведення сканувань.

Відповідно до предметної області система будується з врахуванням наступних особливостей:

- інгредієнти можуть складатись один з одного, утворюючи складні дерева композиції;
- інгредієнти можуть мати більше одного найменування;
- інгредієнти можуть ділитись на підтипи, кожен з яких може мати характеристики притаманні тільки йому;
- користувач має змогу переглянути проведені сканування, що можуть бути згруповані по конкретному відвідуванні магазину;

- користувач має змогу внести перелік інгредієнтів до вподобань, внести свої захворювання, алергії чи інші обмеження на інгредієнт чи їх групу;
- користувач, провівши сканування може отримати детальний звіт про інгредієнти з рекомендаціями, що розраховуються під нього особисто.

Після аналізу предметної області можна виділити ряд сутностей і приступити до проектування інфологічної моделі. В даній предметній області було виділено наступні сутності: Інгредієнт (Ingredient), Органічний інгредієнт (OrganicIngredient), Хімічний інгредієнт (ChemicalIngredient), Група інгредієнтів (IngredientGroup), Загроза (Threat), Група загроз (ThreatGroup), Уподобання (Preference), Обмеження (Restriction), Місце для покупок (ShoppingLocation), Похід за покупками (ShoppingScope), Сканування (Scan).

4.2.3.2 Інфологічна та даталогічна моделі

Як можна побачити з Додатку В, предметна область багата на зв’язки «багато-до-багатьох», а саме: Ingredient – IngredientGroup, Ingredient – Threat, Ingredient – Scan, Thread – ThreadGroup. Для такого роду зв'язків у даталогічній моделі, представленій у додатку Г були створенні розв'язні таблиці IngredientGroupBinding, IngredientThreat, ScanIngredient, ThreatGroupBinding відповідно.

Також предметній області відслідковується наявність ієрархії наслідування: Ingredient (як кореневий елемент), OrganicElement (як дочірній), ChemicalIngredient (як дочірній). Існує кілька способів моделювання ієрархій наслідування в реляційних базах даних.

Для нашого випадку було обрано підхід «таблиця на тип». При такому підході кожна ланка ієрархічного дерева наслідування (тип) має свою окрему таблицю. Кореневий тип містить атрибути притаманні типам всієї ієрархії. Кожен дочірній тип має свою окрему таблицю з атрибутами, що притаманні виключно йому, а також первинний ключ, який одночасно є зовнішнім ключем на батьківську таблицю, що дає змогу зібрати атрибути сутності, а також накладає обмеження цілісності, не дозволяючи створити декілька рядків у дочірній таблиці що

посилаються на один рядок батьківської та обмежує можливість створення фантомних рядків у дочірніх таблицях, що не мають відповідних рядків у батьківській таблиці.

Даталогічна модель бази даних представлена у Додатку Г.

4.2.4 Рівень інфраструктури

Рівень інфраструктури нашої системи, який є компонентом виявлення тексту, представляє розроблена перевикористовувана бібліотека для виявлення та розпізнавання текстової інформації на зображеннях.

Необхідність створення власної бібліотеки для розпізнавання тексту на зображеннях обумовлена низькою швидкістю наявних фреймворків (наприклад, Tesseract OCR). Швидкість розпізнавання тексту складу продуктів з використанням бібліотеки Tesseract досліджувалась на двох різних мобільних пристроях: honor 9 lite (8 ядерний HiSilicon Kirin 659) та oneplus 7t pro (8 ядерний Qualcomm Snapdragon 855 Plus) та представлена у табл. 4.1.

Таблиця 4.1 – Швидкість роботи бібліотеки Tesseract для розпізнавання тексту

Назва процесу	Загальний час обробки 9 lite, с	Загальний час обробки 7t pro, с
Tesseract OCR	8	2.41

Реалізація розробленої бібліотеки дозволяє перепризначити імплементацію класифікатора (нейронної мережі) символів на матриці зображення, що дає розробнику можливість динамічного оновлення моделей, або ж використовувати вбудовану. Опис інтерфейсу бібліотеки представлений у табл. 4.2.

Таблиця 4.2 – Опис інтерфейсу бібліотеки

Назва методу (функції)	Аргументи	Значення, яке повертається	Призначення
ImageUtils::binarize	imageMatrix: Bitmap – 3-канальна матриця зображення	binarizedMatrix: Bitmap – одноканальна бінаризована матриця зображення	Бінаризує зображення застосовуючи метод Оцу
ImageUtils::detectMserRegions	imageMatrix: Bitmap – одноканальна матриця зображення з контрастними регіонами.	contrastRegions: Rect[] - набір регіонів представлених прямокутниками з координатами та розмірами.	Виявляє стабільні контрастні регіони на вхідній матриці зображення.
RegionService::removeDuplicates	regions: Rect[] – набір регіонів представлених прямокутниками iouThreshold: float – поріг подібності прямокутників-дублікатів.	uniqueRegions: Rect[] - набір регіонів представлених прямокутниками з координатами та розмірами.	Виявляє та видаляє регіони-дублікати серед наданого набору прямокутників, застосовуючи метод IoU, методу ширини штриха та евристичних фільтрів.

Продовження таблиці 4.2

Назва методу (функції)	Аргументи	Значення, яке повертається	Призначення
RegionService::deskew	regions: Rect[] – набір регіонів представлених прямокутниками angleStep: float – крок зміни кута при пошуку нахилу регіонів.	skew: float – кут, що утворений нахилом регіонів.	Розраховує кут повороту регіонів по відношенню до осі абсциси координатної площини, застосовуючи метод опуклої оболонки.
RegionService::formTextGroups	regions: Rect[] – набір регіонів представлених прямокутниками	TextRegions: TextLineRegion[] – набір регіонів що згруповані у строки тексту так їх слова.	Групує вхідні регіони по словам та строкам тексту, базуючись на геометричних властивостях.
NetRunner::classifyChar	charImage: Bitmap – матриця зображення, що містить символ.	recognizedChar: char – символ розпізнаний на зображенні.	Розпізнає символ на наданому зображенні застосовуючи нейронну мережу формату .tflite

Кінець таблиці 4.2

Назва методу (функції)	Аргументи	Значення, яке повертається	Призначення
NetRunner::setClassifier	classifier: Classifier – об'єкт, що інкапсулює нейромережеву модель у форматі .tflite. classes: char[] – набір символів, що класифікує classifier.	void	Перепризначає поточний (початковий вбудований) класифікатор символів та набір символів що класифікуються.
TextExtractor::getText	image: Bitmap – матриця зображення з текстовим регіоном для розпізнавання.	text: String – виявлений на матриці зображення текст.	Розпізнає текст на матриці зображення, застосовуючи комп'ютерне бачення.

4.4 Програмні засоби для реалізації методів інтелектуального виявлення тексту

Для створення бібліотеки для виявлення та розпізнавання тексту були використані допоміжні бібліотеки: OpenCv2, TensorFlow, TensorFlow Lite, Keras, Firebase ML Kit.

OpenCv2 – це величезна бібліотека з відкритим вихідним кодом для вирішення різного типу задач комп’ютерного зору, машинного навчання та обробки зображень. Вхідне зображення представляється у матричному вигляді, а робота з вхідним зображенням реалізується методами обробки матричних масивів. У

розробленому програмному модулі бібліотека OpenCv2 використовується, для попередньої обробки зображень, такої як використання методів перетворення в одноканальне зображення та бінаризації, а також для використання алгоритму виявлення контрастних регіонів на зображенні – MSER, механізм використання якого надає дана бібліотека.

Модель нейронної мережі, яка використовується у розробленому програмному забезпеченні, створюється та навчається за допомогою застосунку розробленого мовою програмування Python, що використовує бібліотеку TensorFlow. TensorFlow – величезна бібліотека з відкритим вихідним кодом, розроблена командою Google Brain насамперед для програм глибокого навчання. В основі використання бібліотеки TensorFlow лежить поняття тензору (англ. tensor) – масиви даних, які подаються на вхід нейронним мережам, та які можуть мати різноманітні розміри. Тобто, тензор – це узагальнення матриць, які потенційно можуть мати розмірності високого порядку.

TensorFlow надає декілька високорівневих API, основним з яких є Keras.

Keras – високорівневий API для створення та навчання різного типу моделей. Keras надає інструмент налаштування та навчання моделі нейронної мережі MobileNetV2, яка використовується у розробленій мобільній системі для розпізнавання символьних регіонів [55]. Одним з важливих механізмів є методи ранньої зупинки моделі для запобігання перенавчанню. Також API дозволяє під час навчання зберігати ваги нейронної мережі, отримані на певному етапі або при виконанні певної умови, а тому під час навчання є можливість призупинення навчання та продовження навчання зі збереженого стану, а також повернення до стану, в якому нейронна мережа мала найкращі показники точності.

Оскільки інтелектуальна система обробляє текстову інформацію українською мовою, перш за все виникла необхідність підготовки великого набору навчальних даних. Найкращим рішенням було створити власноруч такий набір даних українських символів через відсутність датасетів у вільному доступі. Для цього було створено генератор навчальних даних, який на базі переданих параметрів

видає набір зображень розмірністю 32*32 пікселі. Генератор датасету, написаний нами мовою програмування Python, є допоміжним компонентом, який генерує набір JPEG зображень з наступними характеристиками:

- розмір зображення – 32*32 пікселів;
- на зображенні на випадковій позиції буде розміщуватись буква;
- кольори фону і букви генерується випадковим чином, але так щоб контрастність двох кольорів дозволяла людському оку побачити, що за буква на зображенні;
- шрифт букви обирається випадково з перечню.

Генератор датасету має функціонал для аугментації даних: розмиття, різні види геометричного спотворення букви, додавання артефактів (таких як шматків інших букв по краях зображення). На рисунку 4.5 представлений приклад згенерованих допоміжним застосунком зображень навчального датасету:

Рисунок 4.5 – Окремі приклади навчальних даних

Для зручності навчання моделі нейронної мережі було створено застосунок для навчання.

TensorFlow Lite — це кросплатформний фреймворк для глибокого навчання з відкритим вихідним кодом (англ. open-source), який використовує попередньо навчену модель у бібліотеці TensorFlow та конвертовану з використанням Python API у спеціальний формат, оптимізований для швидкості зберігання. Модель TF Lite – це модель спеціального формату, ефективна з точки зору точності, а також легка версія, яка буде займати менше місця. Під час процесу перетворення з моделі Tensorflow на модель Tensorflow Lite розмір файлу зменшується [55]. Останнім кроком операції на серверній частині є збереження моделі у плоский буферний файл

Tensorflow Lite (.tflite). Запуск моделі TensorFlow Lite включає завантаження моделі в пам'ять, створення інтерпретатору на основі існуючої моделі, встановити значення вхідного тензора, отримання виходу, зчитування вихідних значень вихідного.

Firebase ML Kit – це мобільний пакет SDK, що надає зручний API, для використання навчених моделей TensorFlow Lite у мобільних системах. Firebase ML Kit спрощує застосування методів машинного навчання у програмах. Для розробленої інтелектуальної системи ML Kit дає можливість динамічно завантажувати моделі, наприклад, які навчені розпізнавати інші мови, та використовувати їх при роботі з системою [26].

4.5 Дослідження ефективності

Дослідження ефективності проводилось за наступними параметрами: оцінка швидкості розпізнавання тексту; оцінка точності розпізнавання тексту; оцінка швидкості роботи алгоритму підтримки прийняття рішень. Для порівняння результатів оцінка ефективності проводилась на апаратному забезпечені з різним

4.5.1 Апаратне забезпечення для проведення тестування

Для тестування використовувались мобільні пристрої, які мають наступні характеристики, представлені у таблиці 4.3:

Таблиця 4.3 – Опис характеристик апаратного забезпечення для тестування

Назва характеристики	Honor 9 Lite	Oneplus 7T Pro	Poco f3
Операційна система	Android 8.0 Oreo	Android 11	Android 11
Процесор	8 ядерний HiSilicon Kirin 659	8 ядерний Qualcomm Snapdragon 855 Plus	8 ядерний Qualcomm Snapdragon 870
Графічний процесор	Mali-T830 MP2	Adreno 640	Adreno 650
RAM	3Gb	8Gb	6Gb

4.5.2 Швидкість розпізнавання тексту

Для виміру швидкості розпізнавання тексту було обрано зображення з середнім за розміром текстом складу об'ємом в 89 слів. Результати швидкодії обробки наведені у таблиці 4.4.

Таблиця 4.4 – Результат швидкодії обробки тексту

Назва процесу	Загальний час обробки honor, с	Загальний час обробки oneplus, с	Загальний час обробки poco f3, с
MSER	0.28	0.11	0.07
Видалення нетекстових регіонів	0.079	0.003	0.003
Групування в рядки та слова	0.021	0.0003	0.0003
Класифікація символів мережею MobileNetV2	0.75	0.15	0.12
Перевірка та заповнення слів	0.152	0.054	0.032

4.5.3 Визначення точності обробки тексту

Критерієм оцінки точності розпізнавання тексту складу запропонованого програмного рішення було обране наступне відношення (представлене відсотком): сума коректно розпізнаних інгредієнтів по відношенню до всіх наявних інгредієнтів разом з сумою всіх хибно розпізнаних інгредієнтів (хибно позитивне розпізнавання) — «відносна точність» та сума кількості нерозпізнаних та хибно розпізнаних — «абсолютна похибка». Результат оцінки точності наведений у таблиці 4.5.

Таблиця 4.5 – Результати оцінки точності

Середня відносна точність	98.33 %
Мінімальна відносна точність	92.14 %
Середня абсолютна похибка	0.87
Максимальна абсолютна похибка	1.30

4.5 Інтерфейс користувача інтелектуальної системи «FoodUnderstood»

Для виявлення та оцінки шкідливості складу продуктів було розроблено інтелектуальну систему у вигляді мобільного додатку. Діаграма використання системи представлена на у Додатку Д.

При відкритті програми користувач потрапляє на головний екран (рис. 4.6 (а)). На головному екрані знаходяться три кнопки «Сканувати склад», «Особистий кабінет» та «Налаштування».

Рисунок 4.6 – Вигляд а) головного Activity системи та б) Activity особистого кабінету

Оскільки оцінка шкідливості є індивідуальною, користувач має можливість ввести свої захворювання або обмеження в особистому кабінеті, натиснувши на кнопку «Особистий кабінет». На екрані особистого кабінету знаходяться дві функціональні кнопки «Загрози», «Вподобання» та «Алергії» (рис. 4.6 (б))

Для введення захворювань, користувачеві необхідно натиснути на кнопку «Загрози», яка веде до екрана введення захворювань (рис. 4.7). На екрані введення захворювань наданий список всіх можливих захворювань. Користувач позначає можливі наявні захворювання зі списку. Наприклад, відмітимо «Непереносимість лактози».

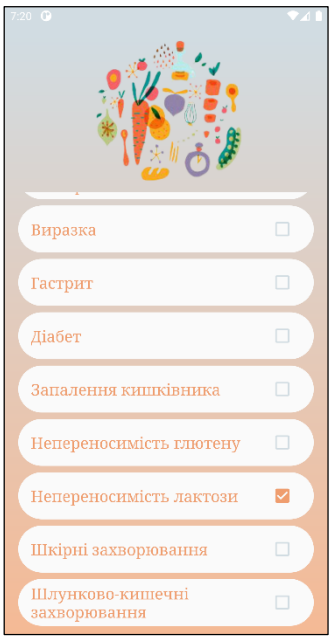


Рисунок 4.7 – Вигляд Activity введення захворювань

При натисненні кнопки «Алергії», користувач потрапляє на екран введення обмежень у вживанні певних інгредієнтів або алергенів, які можуть бути у складі (рис. 4.8). Роботу з вибором інгредієнтів спрощує компонент SearchView для пошуку по загальному списку інгредієнтів.

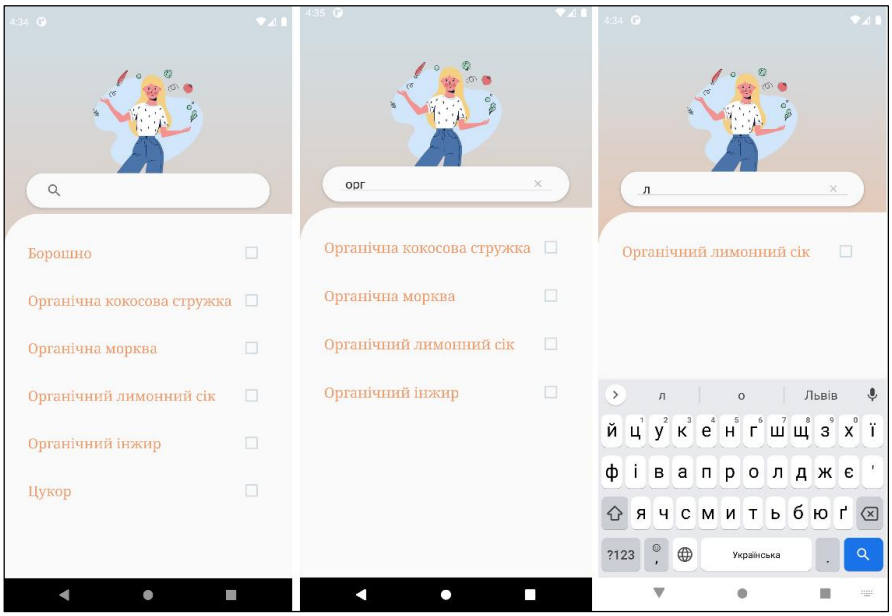


Рисунок 4.8 – Приклад здійснення пошуку по інгредієнтах

Інгредієнти, які є алергенами або представляють загрозу для здоров'я користувача позначаються ним у Checkbox.

Механізм позначення вподобань виглядає таким самим чином, як і позначення інгредієнтів-алергенів. Натиснувши на кнопку «Вподобання», користувач перегодить на Activity вподобання (рис. 4.9).

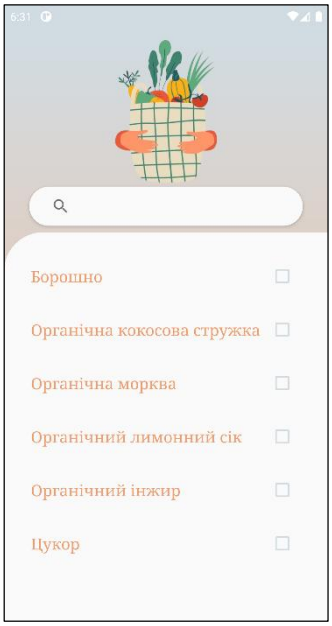


Рисунок 4.9 – Вигляд Activity позначення вподобань

Після введення можливих обмежень користувач виходить на головний екран. Кнопка «Налаштування» необхідна для вибору мови складу, який буде розпізнаватись.

Для початку сканування складу та отримання оцінки шкідливості необхідно натиснути на кнопку «Сканувати склад». Після чого відкривається екран захоплення зображення. Користувач наводить на склад продукту та натискає на кнопку «Зробити знімок» (рис. 4.10 (а)).

Зробивши знімок, користувач потрапляє на екран для обрізання блоку зі складом. Правильно вирівнявши за необхідності зображення та обрізавши його, користувач натискає на кнопку «Застосувати» (рис. 4.10 (б)), яка запускає інтелектуальну обробку зображення та вираховування оцінки шкідливості складу.

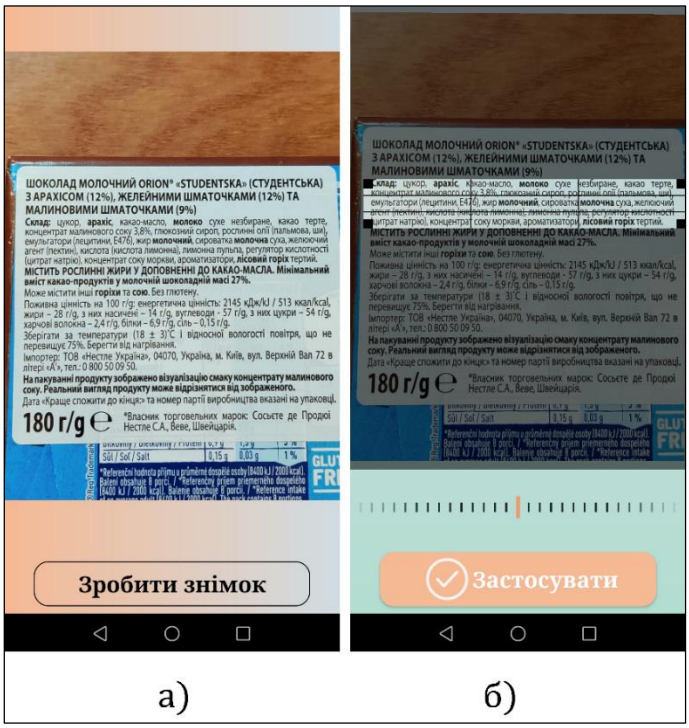


Рисунок 4.10 – Вигляд Activity а) для отримання знімку та б) обрізання блоку зі складом на фото

Після проведення даної операції користувач потрапляє результуючий екран, де він отримує оцінку шкідливості по шкалі ECTS, де А означає «відмінно», а Е – «дуже погано». Складові інгредієнти позначені кольором, де зелений означає повністю безпечний інгредієнт, а червоний – дуже небезпечний. Результат сканування на рис. 4.11.

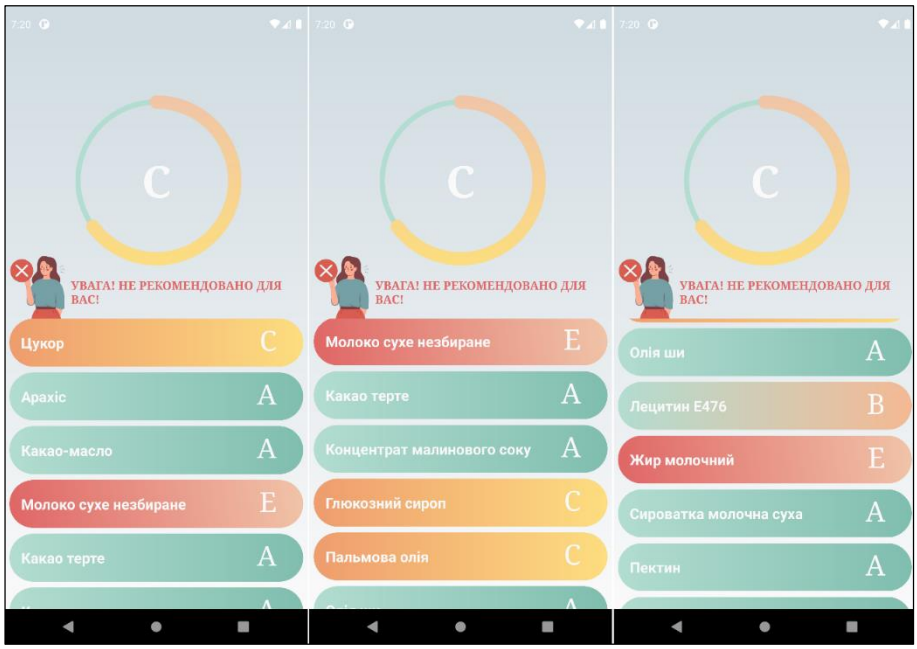


Рисунок 4.11 – Результат сканування для користувача з позначеною загрозою

Якщо оцінка продукту є відмінною, але продукт має у складі інгредієнти, які шкідливі для його здоров’я або є небажаними, то користувач отримує інформацію про наявність такого інгредієнта у складі (рис. 4.12 (б)). Якщо у складі є інгредієнт з групи вподобань користувача, такий інгредієнт позначається відміткою (рис. 4.12(а)).

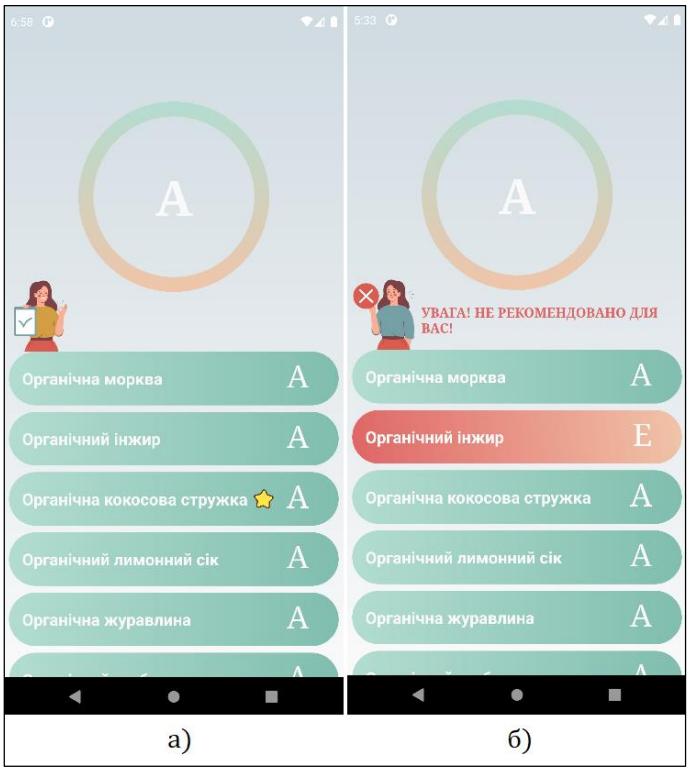


Рисунок 4.12 – Результат сповіщення про небажані інгредієнти

Висновки до розділу

Запропоноване в даному розділі архітектурне рішення підходить для розробки інтелектуальних мобільних систем, які обробляють текстові дані зображення та потребують з можливості оновлення інтелектуального програмного модуля та можливості обміну даними з зовнішнім хмарним середовищем без постійного доступу до сервера, що досягається шляхом використанням безсерверного середовища. Розроблена під час реалізації мобільної системи бібліотека для розпізнавання тексту мобільними пристроями на мові програмування Java використовує апарат штучних нейронних мереж та алгоритмічного забезпечення системи. Оцінка ефективності роботи розробленої програмної бібліотеки показує збільшення швидкості майже у 6 разів, порівняно з існуючими рішеннями. Представлений інтерфейс користувача інтелектуальної системи є простим, зручним та інтуїтивним.

5 МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЄКТУ

5.1 Опис ідеї проекту

В межах даного підпункту послідовно проаналізовано та подано у вигляді таблиць:

- зміст ідеї – розробка додатку, який буде сканувати склад продуктів харчування та впізнавати шкідливі добавки, або інгредієнти, на які може бути алергія у користувача;
- основні вигоди, що отримує користувач;
- чим відрізняється розроблена система від існуючих аналогів.

У таблиці 5.1 описана ідея стартап-проекту, а саме розкритий зміст ідеї, наведені напрямки застосування та сформульовані основні вигоди, що отримує користувач.

Таблиця 5.1 – Опис ідеї стартап проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Розробка додатку, який буде сканувати склад продуктів харчування та впізнавати шкідливі добавки, або інгредієнти, на які може бути алергія у користувача. Користувач буде вводити перелік продуктів-алергенів, або захворювань, а додаток з допомогою камери телефону буде робити фото складу продукту.	1. Скануваня складу продуктів харчування та впізнання шкідливих добавок	Користувач зможе бачити вміст шкідливих добавок, що містяться в продуктах харчування
	2. Виявлення в складі продуктів харчування алергенів, а також добавок, які є дуже шкідливими для конкретного споживача	Користувач зможе виявляти у складі продуктів алергени та добавки, які є шкідливими саме для нього

Кінець таблиці 5.1

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Алергени (вказані користувачем завчасно), а також добавки, які є дуже шкідливими, будуть виділятись червоним кольором. Для реалізації даної ідеї використовуються алгоритми виявлення тексту на зображення та методи розпізнавання тексту нейронними мережами.	3. Надання інформації про кожен з інгредієнтів, що міститься у складі продукту	Користувач зможе отримати інформацію про інгредієнти, які містяться у складі продуктів харчування

Також, проведено аналіз потенційних техніко-економічних переваг ідеї розроблюваного стартап-проекту.

В таблиці 5.2 наведені результати досліджень, а саме:

- визначено перелік техніко-економічних властивостей та характеристик ідеї;
- визначено попереднє коло конкурентів (проектів-конкурентів) або товарів-замінників чи товарів-аналогів, що вже існують на ринку, та проведено збір інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку;
- проведено порівняльний аналіз показників: для власної ідеї визначено показники, що мають:
  1) гірші значення (W, слабкі);
  2) аналогічні (N, нейтральні) значення;
  3) кращі значення (S, сильні).

Для порівняння були обрані два найбільш схожі конкуренти, «Open Food Facts» та «FoodScanner».

Таблиця 5.2 – Переваги продукту

№	Техніко-економічні характеристики ідеї	Продукція конкурентів			W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Мій проєкт	«Open Food Facts»	«Food Scanner»			
1	Сканування текстової інформації складу продуктів	так	ні	так			+
2	Підтримка української мови	так	ні	ні			+
3	Персоналізована оцінка шкідливості	так	ні	ні			+
4.	Вимагає постійної підтримки БД	ні	так	ні		+	
5.	Робота при відсутності мережі	так	ні	ні			+

Провівши аналіз та порівняння з попереднім колом проектів, було визначено, що у розроблюваного застосунку висока конкурентоспроможність, оскільки навіть найближчі аналоги мають не усіх функцій мого проекту.

5.2 Технологічний аудит ідеї проекту

В межах даного підрозділу було проведено аудит технологій, за допомогою яких можна реалізувати ідею проекту. Технологічу здійсненність ідеї проекту було проведено за допомогою аналізу таких складових:

- технологія, за якою буде виготовлено товар згідно ідеї проекту;
- існування технології: технологія вже існує чи її потрібно розробити;
- доступність технології.

Технологічну здійсненність ідеї проекту наведено у таблиці 5.3.

Таблиця 5.3 – Технологічна здійсненність ідеї проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Виявлення текстових регіонів	Алгоритм MSER бібліотеки OpenCv	Наявна	Доступна
2	Видалення нетекстових регіонів	Використання метрик IoU, евристичні правила	Розробити	Розроблено автором проекту
3	Розпізнавання символів нейронною мережею MobileNetV2	Бібліотека keras для створення та навчання моделі	Наявна	Доступна
5	Можливість незалежного озповсюдження оновлень	Firebase ML	Нявна	Для наших цілей технологія безкоштовна

За результатами аналізу таблиці, зроблено висновок, що технологічна реалізація проекту можлива і визначено технологічний шлях, яким доцільно реалізувати ідею проекту.

5.3 Аналіз ринкових можливостей запуску стартап-проекту

В даному підрозділі було визначено ринкові можливості, які можна використати під час ринкового впровадження проекту, та ринкові загрози, які можуть перешкодити реалізації проекту, що дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

5.3.1 Аналіз попиту

У таблиці 5.4 було проведено аналіз попиту, а саме наявність попиту, обсяг та динаміка розвитку ринку.

Таблиця 5.4 – Попередня характеристика потенційного ринку

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	3
2	Загальний обсяг продаж, грн./ум.од	Невідомо
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі або по ринку, %	Дані відсутні

Висновок: враховуючи невелику кількість конкурентів та зростаючу динаміку ринку за попереднім оцінюванням можна зробити висновок, що на даний момент ринок для входження стартап-продукту є привабливим.

5.3.2 Визначення потенційних груп клієнтів

В таблиці 5.5 визначено потенційні групи клієнтів, їх характеристики, та сформовано орієнтовний перелік вимог до товару для кожної групи.

Таблиця 5.5 – Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці цільових груп клієнтів	Вимоги споживачів до товару
1	Сканування складу продуктів харчування та впізнання шкідливих добавок	Фізичні особи Фітнес клуби Підприємства рітейлу	Підприємства та клуби готові заплатити високу ціну за продукт, що задовольняє їх потреби	Якісне розпізнавання складу продуктів, зрозумілий формат результатів розпізнавання інгредієнтів
2	Виявлення в складі продуктів харчування алергенів, а також добавок, які є дуже шкідливими для конкретного споживача	Фізичні особи Лікарі-дієтологи приватних медичних закладів		
3	Надання інформації про кожен з інгредієнтів складу продукту	Фізичні особи Фітнес клуби Підприємства рітейлу		

5.3.3 Аналіз ринкового середовища

Для аналізу ринкового середовища складено таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають. Фактори, що сприяють ринковому впровадженню проекту, наведені у таблиці 5.6.

Таблиця 5.6 – Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Наявність функціоналу, якого немає у конкурентів	Немає конкурентів, у яких є розпізнавання складу, представленого укрїнською мовою, персоналізована оцінка шкідливості	У рекламній кампанії акцентувати увагу на переваги продукту над продуктами конкурентів. Розширення набору існуючих функцій.
2	Зниження довіри до конкурента	Через обмежений функціонал програмних продуктів конкурентів їх клієнтська база зменшується	У рекламній компанії акцентувати увагу на переваги продукту над продуктами конкурентів.
3	Консультації з потенційними клієнтами	Можливість отримання необхідної інформації для вдосконалення продукту	Передбачити можливість отримання даних для відповідної реакції на них з боку команди розробників задля задоволення потреб та бажань кінцевих користувачів

Фактори, що перешкоджають ринковому впровадженню проекту, наведені у таблиці 5.7.

Таблиця 5.7 – Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Поява конкурентів	Поява могутніх компаній-конкурентів	Запропонувати більш досконалий продукт з унікальними характеристиками
2	Недостатня рекламна компанія	Недостатнє розповсюдження реклами та недостатнє зацікавлення цільової аудиторії	Вдосконалити власну рекламну компанію
3	Недостатність коштів для розробки та підтримки продукту	Недостатність фінансування	Залучення нових інвесторів, поетапне впровадження продукту

5.3.4 Аналіз пропозиції

Ступеневий аналіз конкуренції на ринку наведений у таблиці 5.8.

Таблиця 5.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції - монополія	Один головний товаровиробник	Здійснення контролю над цінами. Захищеність від конкуренції
2. За рівнем конкурентної боротьби - локальний рівень	Конкуренти всередині країни	Пошук однодумців лише в Україні

Кінець таблиці 5.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
3. За галузевою ознакою - внутрішньогалузева	Конкуренція спостерігається в сфері охорони здоров’я та здорового харчування	Розробка продукту, що вирішує специфічні задачі
4. Конкуренція за видами товарів - товарно-родова	Конкуренція між товарами різного виду	Додавання нового функціоналу
5. За характером конкурентних переваг -нецінова	Конкуренція проводиться за рахунок вдосконалення якості продукту	Вдосконалення продукту
6. За інтенсивністю - немарочна	Роль торгової марки незначна	Заохочення клієнтів якістю товару, а не брендом.

Аналіз конкуренції в галузі за М. Портером наведений у таблиці 5.9.

Таблиця 5.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Відсутні	Монополія	Розміри поставок	Дешева ціна	Ціна
Висновки:	Конкурентна боротьба відсутня, що зменшує стимул вдосконалення продукту	Можливості виходу на ринок є оскільки не існує аналогів, а товар замінник має значно гірший функціонал	Незначні обсяги замовлень поставок	Доступна ціна	Обмежень немає

Висновок: Проведений аналіз показав, що існують можливості виходу продукту на ринок. Для виходу на ринок продукт повинен мати функціонал, що відсутній у продуктів-аналогів, повинен задовольняти потреби користувачів, мати необхідний та достатній функціонал з конфігурування та підтримку зі сторони розробників.

5.3.5 Обгрунтування переліку факторів конкурентоспроможності

Виходячи з вимог споживачів і характеристик ідей проекту визначимо перелік факторів конкурентоспроможності (таблиця 5.10)

Таблиця 5.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування
1	Підвищення якості	Постійне вдосконалення та оновлення додатку
2	Потреби споживачів	Оцінка та відгуки споживачів додатку впливають на вдосконалення мобільного додатку
3	Застосування новітніх технологій	Використання новітніх технологій штучного інтелекту, що впливає на якість продукції
4	Результативність	Кінцевий результат
5	Відсутність конкурентів	Відсутність варіантів вибору у цільової аудиторії
6	Ціна та собівартість продукції	Конкурентоспроможна ціна

5.3.6 Аналіз сильних та слабких сторін стартап-проекту

Таблиця 5.11 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-замінників у порівнянні зі стартапом						
			–3	–2	–1	0	+1	+2	+3
1	Підвищення якості	15				+			
2	Потреби споживачів	19							+
3	Застосування новітніх технологій	17						+	
4	Результативність	18							+
5	Відсутність конкурентів	17							+
6	Ціна та собівартість продукції	16				+			

2.3.7 SWOT аналіз стартап-проекту

Завершальним етапом ринкового аналізу можливостей реалізації проекту є складання SWOT (матриця аналізу сили та слабкості), аналізу загроз (проблем) та можливостей (таблиця 5.12) на основі обраних ринкових загроз та можливостей, а також сильних та слабких сторін. Перелік загроз та ринкових можливостей базується на аналізі загроз та факторів маркетингового середовища. Ринкові загрози та ринкові можливості – це наслідки (очікувані результати) впливу факторів і, навпаки, вони ще не реалізовані на ринку та мають певну ймовірність реалізації.

Таблиця 5.12 – SWOT-аналіз стартап-проекту

Сильні сторони: даний мобільний додаток дозволяє виявити в харчових продуктах шкідливі домішки та алергенів, та надати персоналізовану оцінку шкідливості продукту.	Слабкі сторони: компанія є «новачком» на ринку, тому не має репутації. Необхідність залучення інвесторів через недолік фінансування.
Можливості: Низька ціна на продукт, Легке налаштування, Наявність максимальної інформації у додатку Великий відсоток розпізнаваних інгредієнтів	Загрози: Малий ринок послуг Недостатність коштів Витіснення конкурентами за рахунок кращих пропозицій

5.3.8 Альтернативні поведінки

На основі SWOT-аналізу були розроблені альтернативи ринкової поведінки для виведення стартап-проекту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проекти конкурентів, що можуть бути виведені на ринок.

Визначені альтернативи аналізуються з точки зору строків та ймовірності отримання ресурсів у таблиці 5.13

Таблиця 5.13 – Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Проведення онлайн зустрічей з дієтологами та спеціалістами по здоровому харчуванню з рекламою додатку	Висока за рахунок того, що товар буде на слуху	1 рік
2	Впровадження безкоштовного пробного періоду користування	Висока за рахунок того, що у клієнта з’явиться бажання купити ліцензію на довгий термін після закінчення пробного періоду	1 рік

Як альтернативу варто обрати комбінацію вищенаведених варіантів – впровадження безкоштовного пробного періоду користування разом з активним просуванням у вигляді проведення онлайн зустрічей з дієтологами та спеціалістами по здоровому харчуванню з рекламою додатку.

5.4 Розроблення ринкової стратегії проекту

5.4.1 Опис цільових груп потенційних користувачів

Таблиця 5.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсив-ність конкуренції в сегменті	Простота входу у сегмент
1.	Фізичні особи	Висока, серед споживачів, що ведуть здоровий спосіб життя або мають алергію на харчові продукти	90% - високий попит на продукцію	Відсутня конкуренція	Легко, бо немає аналогів
2.	Фітнес клуби	Висока, бо компанії зацікавлені покращити свій сервіс і пропагують здоровий спосіб життя	90% - високий попит	Відсутня конкуренція	Легко, бо немає аналогів

Кінець таблиці 5.14

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
3.	Підприємства рітейлу	Середня, не всі підприємства мають спеціалізовані стенди для безпечних продуктів харчування	50% - середній попит на продукцію	Відсутня конкуренція	Важко, так як такі підприємства можуть користуватися іншими джерелами інформації
4.	Лікарі-дієтологи приватних медичних закладів	Середня, такі клієнти можуть користуватися іншими джерелами інформації	50% - середній попит на продукцію	Відсутня конкуренція	Важко, так як лікарі можуть користуватися іншими джерелами інформації
Які цільові групи обрано: фізичні особи, фітнес клуби					

За результатами аналізу потенційних груп споживачів були обрані цільові групи – 1 і 2, для яких буде пропонуватися даний товар, та стратегія захоплення ринку – диференційована маркетингова стратегія (компанія працює з різними сегментами).

5.4.2 Базова стратегія розвитку

Для роботи в обраних сегментах ринку було сформовано базову стратегію розвитку (таблиця 5.15)

Таблиця 5.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспромо жні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Виготовлення додаткового програмного забезпечення для обробки результатів та підбору оптимального набору харчових продуктів, рекомендованих для вживання з урахуванням інвидуальних особливостей та медичного діагнозу	Стратегія диференці-йованого маркетингу	Зручність та легкість у користування, не потребує налаштування	Стратегія диференціації

5.4.3 Вибір стратегії конкурентної поведінки

Здійснений вибір базової стратегії конкурентної поведінки (таблиця 5.16).

Таблиця 5.16 – Визначення базової стратегії конкурентної поведінки

Чи є проєкт «першопрохідцем» на ринку	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, які?	Стратегія конкурентної поведінки
Ні, оскільки є замінники	Забирати існуючих споживачів у конкурентів за рахунок унікальності, а також пошук нових споживачів, які ще не пробували користуватись подібним продуктом	Не буде	Стратегія лідера: наступальна (охопити максимально ринок, забрати користувачів у конкурентів)

5.4.4 Визначення стратегії позиціонування

На основі потреб споживачів з вибраних сегментів до постачальника (стартап-компанії) та товару, а також обраної базової стратегії розвитку та стратегії конкурентної поведінки розробляється стратегія позиціонування, яка полягає у формуванні ринкової позиції, за якою користувачі матимуть змогу безпомилково визначити проект.

Таблиця 5.17 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
1.	Відсутність налаштування програмного продукту	Стратегія диференціації	Не потребує налаштування, готовий для використання після завантаження	Швидкий вхід в систему Доступність для будь-яких користувачів Кросплатформеність
2.	Легкість у використанні та швидкість	Стратегія диференціації	User-friendly інтерфейс	Зручний інтерфейс Адаптованість Взаємодія з клієнтом через підказки

Відповідно до проведеного аналізу можна зробити висновок, що компанія вибирає як базову стратегію розвитку – стратегію диференціації, як базову стратегію конкурентної поведінки – стратегію лідера.

5.5 Розроблення маркетингової програми стартап-проєкту

5.5.1 Маркетингова концепція товару

Першим кроком є формування маркетингової концепції товару. Для цього необхідно узагальнити результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.18 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1.	Зручний інтерфейс	Не потребує адаптації користувача до програми	Сучасний та зрозумілий інтерфейс
2.	Доступність інформації, яку надає додаток	Можливість розпізнати великий відсоток інгредієнтів	Розпізнавання тексту українською мовою, швидкість розпізнавання та постійний доступ до інформації, можливість індивідуальної оцінки шкідливості
3.	Швидка робота при відсутності з'єднання з мережею	Можливість використовувати без підключення до мережі	Коректна робота програми у будь-якому при відсутності доступу до мережі

5.5.2 Маркетингова модель товару

Розроблено трирівневу маркетингову модель товару. Для цього у таблиці 5.19 підсумовано результати попереднього аналізу конкурентоспроможності товару.

Таблиця 5.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Мобільний додаток по скануванню та розпізнаванню складу продуктів харчування та виявлення шкідливих домішок та алергенів		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор*
	1. Зручний інтерфейс 2. Адаптованість 3. Масштабованість	М М М	Тх/Тл/Ор Тх/Ор Тх/Тл

Кінець таблиці 5.19

II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор*
	4. Швидкість інтернету 5.Процесор мобільного телефону	Нм М	Тх/Вр Тх/Вр
	Якість: Додаток відповідає стандартам якості Googleplay. Дизайн: додаток не перевизначає очікувану функцію системного значка (наприклад, кнопку Назад). Додаток не замінює системну піктограму абсолютно іншою, якщо вона запускає стандартну поведінку інтерфейсу користувача. Додаток не перевизначає та не використовує неправильно шаблони інтерфейсу Android, так що піктограми чи поведінка можуть ввести користувачів в оману або заплутати. Функціональність: Додаток вимагає лише абсолютних мінімальних дозволів, необхідних для підтримки основних функціональних можливостей. Стабільність: Додаток не завершує роботу в аварійному режимі, примусово закриває, зависає чи не працює аномально на будь-якому цільовому пристрої.		
	Марка: FoodUnderstood		
III. Товар із підкріпленням	До продажу: Інтелектуальна система для обробки складу продуктів та оцінки шкідливості харчових продуктів		
	Після продажу: видання нових версій з розширенням мов, які буде розпізнавати додаток		
За рахунок чого потенційний товар буде захищено від копіювання: патенту та авторського права			

* *М/Нм – монотонні/немонотонні; Вр/Тх/Тл/Е/Ор – вартісні/ технічні/ технологічні/ ергономічні/ органолептичні*

5.5.3 Визначення меж встановлення ціни

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги, а також аналіз рівня доходів цільової групи споживачів. Аналіз проводиться за допомогою експертного методу. Визначення меж встановлення ціни наведено у таблиці 5.20.

Таблиця 5.20 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів		Верхня та нижня межі встановлення ціни на товар/послугу
0-1000 грн.	Відсутні	Фізичні особи	Фітнес-клуби	80 – 2000 грн./міс
		15000 грн.	50000-500000 грн	

5.5.4 Оптимальна система збуту

Визначаємо оптимальну систему продажів, за якою приймається рішення: здійснювати власні продажі або залучати сторонніх посередників (власна або задіяна система продажів); вибір та обґрунтування оптимальної глибини каналу збуту; вибір та обґрунтування типу посередників.

Таблиця 5.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Клієнти купують продукт на GooglePlayMarket	Реклама додатку Обслуговування	1-Канал першого рівня (виробник безпосередньо продає товар роздрібному продавцю, останній продає клієнту)	Через GooglePlay Market

5.5.5 Розроблення стратегії маркетингових комунікацій

Останньою складовою маркетингової програми є розробка концепції маркетингової комунікації, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів. Концепція маркетингових комунікацій наведена у таблиці 5.22.

Таблиця 5.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
Клієнти дізнаються про товар з листівок, інтернету та соціальних мереж	Інтернет Роздаткова реклама Соціальні мережі	Абсолютно новий продукт	Проінформувати про існування продукту	Споживання здорової та безпечної їжі. Рекламне звернення «FoodUnderstood – розберись зі складом та збережи гроші!»

Висновок до розділу

У даному було сформульовано та описано ідею стартап-проекту з аналізом потенційних техніко-економічних переваг цієї ідеї. Було проведено аудит технології, за допомогою якої можна реалізувати ідею проекту та визначено технологічний шлях, який є доцільним для реалізації даної ідеї. В результаті проведених досліджень було виявлено, що ідея стартап-проекту технічно може бути реалізованою, а також має високу конкурентоспроможність. Також було розроблено ринкову стратегію, а саме було визначено стратегії охоплення ринку, сформовано базову стратегію розвитку визначено стратегію позиціонування. Фінальним етапом даного розділу було розроблення маркетингової програми стартап-проекту.

ВИСНОВКИ

У ході виконання магістерської дисертації було розроблено програмне та алгоритмічне забезпечення для системи інтелектуального виявлення та аналізу складу продуктів харчування.

Для розробки програмного забезпечення для інтелектуальної системи було використано ідеї клієнт-серверної архітектури, які було адаптовано та розширено для розробки інтелектуальних систем. Відтак, у розробленій системі кожен компонент архітектури є автономним, всі обчислювальні операції відбуваються на пристрої, а безсерверне середовище використовується для збереження та синхронізації даних, а також для підтримки актуальності даних.

У першому розділі, на основі даних, отриманих в процесі аналізу, було виявлено високу необхідність у системі інтелектуального, персоналізованного аналізу складу продуктів: саме через відсутність автоматизації процесу безліч покупців ігнорує склад товарів та вживає шкідливі, або навіть загрозливі продукти харчування. На відміну від існуючих рішень, що надають відгук про продукт лише за наявності ідентифікатору, що отримується скануванням штрих-коду, у БД, було прийнято рішення розробити систему, яка розпізнає саме текстову інформацію на зображеннях зі складом продуктів, а отже, було вирішено створити саме інтелектуальну систему, яка, використовуючи апарат нейронних мереж, розпізнаватиме текст складу продуктів харчування.

У другому розділі, зважаючи на необхідність інтелектуального компонента розпізнавання, було проведено аналіз існуючих рішень детекції та розпізнавання тексту та підібрано алгоритм MSER для виявлення текстових регіонів зображення, а також підібрано архітектуру нейронної мережі, а саме MobileNetV2, як найбільш підходящу для розпізнавання тексту на пристроях з обмеженими обчислювальними ресурсами – мобільних телефонах.

У третьому розділі було представлено розроблене алгоритмічне забезпечення для розпізнавання тексту, яке складається з алгоритму виявлення максимально стабільних екстремальних регіонів, алгоритму видалення нетекстових регіонів,

алгоритму вирівнювання тексту та групування слів, алгоритму отримання оцінки шкідливості продукту та використання згорткової нейронної мережі для розпізнавання текстової інформації. Представлені алгоритми підібрані до вирішення задачі обробки зображень зі складом продуктів, оскільки враховуть всі особливості таких зображень (малу контрастність, шуми, штрифт, фон та ін.).

У четвертому розділі, на базі алгоритмічного забезпечення було розроблено програмне забезпечення для інтелектуальної системи, включаючи програмний модуль у вигляді бібліотеки на мові програмування Java для розпізнавання текстової інформації на зображеннях на пристроях з обмеженими обчислювальними ресурсами, яка, порівняно з існуючим рішенням таким, як бібліотека Tesseract OCR, показує перевагу у швидкості у 5 разів на бюджетному мобільному пристрої.

В останньому розділі проведений маркетинговий аналіз стартап-проєкту. Проведено опис ідеї проєкту, технологічний аудит ідеї проєкту, аналіз ринкових можливостей запуску стартап-проєкту. Розроблено ринкову стратегію проєкту та маркетингову програму стартап-проєкту.

Результати роботи над магістерською дисертацією опубліковані в статті «System for detecting and analyzing textual information of product composition» до наукового фахового видання «Електроніка та системи управління» та в статті «Intelligent system for assessing the harmfulness of food products based on the processing of textual and graphic information» до наукового фахового видання «Штучний інтелект». Також матеріали роботи були опубліковані у збірнику тез до першої всеукраїнської науково-технічної конференції молодих вчених та студентів «Інженерія програмного забезпечення і передові інформаційні технології» (SoftTech-2021). 22-26 листопада 2021р, Козак О.С., Синєглазов В.М. «Intelligent system for determining the harmfulness of food composition».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) A Brief Discussion About Client-Server System / Sawan Gupta, Sahil Narang, Shubham Arora, Jindal Shruti // International Journal of Advanced Research in Computer Engineering & Technology. – Volume 6. – № 3. – March 2017. – pp. 338-341. – ISSN: 2278 – 1323.

2) A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospect / Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, Jun Zhou // IEEE Transactions on Neural Networks and Learning Systems. – June 2021. – pp. 1 – 21. – doi: 10.1109/TNNLS.2021.3084827.

3) Anaphylaxis to annatto dye: a case report / W.A. Nish, B. A. Whisman, D. W. Goetz, D. A. Ramirez //Annals of Allergy. – Volume 66. – № 2. – March 1991. – pp. 126-131. – doi:10.1097/MCG.0b013e3181b84517.

4) Approved additives and E numbers. Additives and E numbers for colours, preservatives, antioxidants, sweeteners, emulsifiers, stabilisers, thickeners and other types of additives [Електронний ресурс] – Режим доступу до ресурсу: https://www.food.gov.uk/business-guidance/approved-additives-and-e-numbers.

5) Are MSER features really interesting? / Kimmel Ron, Zhang Cuiping, Bronstein Alexander M, Bronstein Michael // IEEE Transactions on Pattern Analysis and Machine Intelligence. – Volume 33. – № 11. – November 2011. – pp. 2316 - 2320 – ISSN: 0162-8828.

6) Arokia Paul Rajan R. Serverless Architecture – A Revolution in Cloud Computing / Arokia Paul Rajan R // Tenth International Conference on Advanced Computing (ICAC). – Chennai, India, 13-15 December 2018 – pp. 88–93 – ISBN 978-1-7281-0354-9.

7) Azure Functions documentation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/azure/azure-functions/.

8) Azure Storage Types [Електронний ресурс] – Режим доступу до ресурсу: https://www.dremio.com/data-lake/adls/.

9) Backpropagation in Convolutional Neural Networks [Електронний ресурс] – Режим доступу до ресурсу: https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/.

10) Bae Sang Won. On the Minimum-Area Rectangular and Square Annulus Problem / Sang Won Bae // Computational Geometry. – Volume 92. – January 2021. – 22 p. – doi: 10.1016/j.comgeo.2020.101697.

11) Bosilj Petra. Beyond MSER: Maximally Stable Regions using Tree of Shapes / Petra Bosilj, Ewa Kijak, S. Lefèvre // Proceedings of the British Machine Vision Conference (BMVC) ; ed. by Xianghua Xie, Mark W. Jones, and Gary K. L. Tam. – September 2015. – pp. 169–184. – ISBN 1-901725-53-7.

12) Chen Datong. Text Detection and Recognition in Images and Videos / Datong Chen, Herve Bourlard, Jean-Marc Odobez // Pattern Recognition. – Volume 37. - № 3. – March 2004. – pp. 595-608. – doi:10.1016/j.patcog.2003.06.001.

13) Chollet F. Xception: Deep learning with depthwise separable convolutions / François Chollet // IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – USA, November 2017. – pp. 1251–1258. – ISBN 978-1-5386-0458-8.

14) Cuervo-Pardo L. Apps for food allergy: A critical assessment / Lyda Cuervo-Pardo, Maria Angelica Barcena-Blanch, Alexei Gonzalez-Estrada // The Journal of Allergy and Clinical Immunology: In Practice. – August 2015. – Volume 3. – № 6. – P980-981.E1. – doi:10.1016/j.jaip.2015.06.011.

15) Data Binding Library [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/topic/libraries/data-binding.

16) Deep Convolutional Neural Networks. A survey of the foundations, selected improvements, and some current applications [Електронний ресурс] : електронний архів наукових статей: arXiv.org / Lars Ankile, Morgan Heggland, Kjartan Krange // Computer Vision and Pattern Recognition .– November 2020. – 14 p. – Режим доступу до ресурсу: https://arxiv.org/pdf/2011.12960.pdf.

17) Deep Convolutional Neural Networks: A survey of the foundations, selected improvements, and some current applications [Електронний ресурс]: електронний архів наукових статей: arXiv.org / Ankile Lars, Heggland Morgan, Krange Kjartan // Computer Vision and Pattern Recognition (arXiv:2011.12960v1) – 2020р. – 10р. – Режим доступу до ресурсу: https://arxiv.org/pdf/2011.12960.pdf.

18) Deep learning for pedestrians: backpropagation in CNNs [Електронний ресурс] : електронний архів наукових статей : arXiv.org / Bou Laurent. – November 2018. – pp. 44. – Режим доступу до ресурсу: https://arxiv.org/ftp/arxiv/papers/1811/1811.11987.pdf

19) Diet, nutrition and the prevention of chronic diseases / World Health Organization // Report of a joint WHO/FAO expert consultation. – Geneva, 2003. – p. 160. – ISBN 92 4 120916 X.

20) Dropout: A Simple Way to Prevent Neural Networks from Overfitting / Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever [та ін.] // Journal of Machine Learning Research. – Volume 15. – №1. – January 2014. – pp. 1929-1958. – doi/10.5555/2627435.2670313.

21) E252 – Нитрат калия [Електронний ресурс] – Режим доступу до ресурсу: https://dobavkam.net/additives/e252.

22) E471 – Моно- и диглицериды жирных кислот. [Електронний ресурс] – Режим доступу до ресурсу: https://dobavkam.net/additives/e471.

23) Epshtein Boris. Detecting Text in Natural Scenes with Stroke Width Transform / Boris Epshtein, Eyal Ofek, Yonatan Wexler // Proceedings IEEE Computer Society Conference on Computer Vision and Pattern Recognition. – San Francisco, June 2010. – 8 p. – doi: 10.1109/CVPR.2010.5540041.

24) Evaluation of certain food additives and contaminants / World Health Organization and Food and Agriculture Organization // Sixty-ninth report of the Joint FAO/WHO Expert Committee on Food Additives (Meeting 80th). – Italy, 2015. – 109 p. – ISBN 9789241209953. – ISBN 9789240695405 (pdf).

25) Exploring the Ideal Depth of Neural Network when Predicting Question Deletion on Community Question Answering [Електронний ресурс] : електронний архів наукових статей: arXiv.org / Ghosh Souvick, Ghosh Satanu // Information Retrieval. – December 2019. – Режим доступу до ресурсу: https://arxiv.org/abs/1912.03585.

26) Firebase Machine Learning [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/docs/ml.

27) Food additives [Електронний ресурс] – Режим доступу до ресурсу: https://www.who.int/news-room/fact-sheets/detail/food-additives.

28) FoodRepo: An Open Food Repository of Barcoded Food Products / Gianrocco Lazzari, Yannis Jaquet, Djilani Kebaili, Laura Symul, Marcel Salathé // Frontiers in Nutrition. – Jan 2018. – Volume 5. – № 57. – doi:10.3389/fnut.2018.00057.

29) Garrett McGrath. Serverless Computing: Design, Implementation, and Performance / Garrett McGrath, Paul R. Brenner // IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW). – Atlanta, July 2017. – pp. 405 – 410. – doi: 10.1109/ICDCSW.2017.36.

30) Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression [Електронний ресурс] : електронний архів наукових статей: arXiv.org / Hamid Rezatofighi, Nathan Tsoi, Jun Young Gwak, Amir Sadeghian // Computer Vision and Pattern Recognition. – USA, April 2019. – Режим доступу до ресурсу: https://arxiv.org/pdf/1902.09630.pdf.

31) Geron A. Hands-On Machine Learning with Scikit-Learn and TensorFlow / Aurelien Geron / [publisher O'Reilly Media]. – 2nd edition – March 2017. – 581с. – ISBN: 9781491962299.

32) Global digital population as of January 2021 [Електронний ресурс] – Режим доступу до ресурсу: https://www.statista.com/statistics/617136/digital-population-worldwide/#:~:text=How.

33) Goodfellow Ian. Deep Learning (Adaptive Computation and Machine Learning series) / Goodfellow Ian, Bengio Yoshua, Courville Aaron / [publisher The MIT Press] – October 2017. – 800 p. – ISBN 978-0262035613.

34) Goyal R. Food label reading: Read before you eat / Raksha Goyal, Neeta Deshmukh // Journal of education and health promotion. – Volume 7. – № 56. – April 2018. – doi: 10.4103/jehp.jehp_35_17.

35) GradientBased Learning Applied to Document Recognition / LeCun Y., Bottou L., Bengio Y., Haffner P. // Proceedings of the IEEE. – Volume 86. – № 11. – November 1998. – pp. 2278 – 2324. – doi: 10.1109/5.726791.

36) Grouping text lines in freeform handwritten notes / M. Ye, H. Sutanto, S. Raghupathy, C. Li [та ін.] // Eighth International Conference on Document Analysis and Recognition (ICDAR'05). – Jan. 2006. – doi: 10.1109/ICDAR.2005.121.

37) Guideline: Sugars intake for adults and children / [publisher Physical Medicine and Rehabilitation Research Center] // Nutrition and Food Safety ; ed. by World Health Organization. – Iran, March 2015 p. – 49 p. – ISBN 9789241549028.

38) How to Treat Overfitting in Convolutional Neural Networks [Електронний ресурс] – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2020/09/overfitting-in-cnn-show-to-treat-overfitting-in-convolutional-neural-networks/.

39) Implementation of Training Convolutional Neural Networks [Електронний ресурс] : електронний архів наукових статей: arXiv.org / Tianyi Liu, Shuangsang Fang, Yuehui Zhao, Peng Wang, Jun Zhang // Computer Vision and Pattern Recognition. – Junne 2015. – 10 p. – Режим доступу до ресурсу: https://arxiv.org/ftp/arxiv/papers/1506/1506.01195.pdf.

40) International food standards [Електронний ресурс] : електронний ресурс міжнародного стандарту їжі : Codex Alimentarius / WHO and FAO. – Режим доступу до ресурсу: http://www.fao.org/fao-who-codexalimentarius/home/en.

41) Jamshed Memon. Handwritten Optical Character Recognition (OCR) / Jamshed Memon, Maira Sami, Rizwan Ahmed Khan // IEEE Access. – Volume 8. – July 2020. – pp. 142642 – 142668. – doi:10.1109/ACCESS.2020.3012542.

42) Jay A. Kreibich. Using SQLite: Small. Fast. Reliable. Choose Any Three / Jay A. Kreibich // Computers / [publisher O'Reilly Media]. – August 2010. — 530 p.

43) Kanan Christopher. Color-to-grayscale: Does the Method Matter in Image Recognition? / Christopher Kanan, W Garrison Cottrell // Public Library of Science One. – Volume 7. – № 1 (e29740) – 7 p. – USA, January 2012. – doi: 10.1371/journal.pone.0029740.

44) Karanje Uma B. Maximally Stable Extremal Region Approach for Accurate Text Detection in Natural Scene Images / Uma B. Karanje, Rahul Dagade, Sankirti Shiravale // International Journal of Scientific Development and Research. – Volume 1. – № 11. – August 2018. – pp. 161-168. – ISSN: 2455-2631.

45) Karanje Uma. Survey on Text Detection, Segmentation and Recognition from a Natural Scene Images / Uma Karanje, Rahul Dagade // International Journal of Computer Applications. – Volume 108. – № 13. – December 2014. — pp. 39-43. – doi:10.5120/18974-0472.

46) Kin Ng. Tuned Inception V3 for Recognizing States of Cooking Ingredients [Електронний ресурс] : електронний архів наукових статей: arXiv.org / Kin Ng // Computer Vision and Pattern Recognition. – May 2019. – 6 p. – Режим доступу до ресурсу: https://arxiv.org/pdf/1905.03715.pdf.

47) Lam L. Thinning methodologies – a comprehensive survey / Lam L., Lee S. W., Suen C. Y. // IEEE Transactions on Pattern Analysis and Machine Intelligence. – Volume 14. – № 9. — September 1992. – pp. 869–885. – doi:10.1109/34.161346.

48) LeCun Y. Deep learning / LeCun Y., Bengio Y., Hinton G. / [publisher Nature Publishing Group] // Nature. – Volume 521. – May 2015. – pp. 436–444. – doi:10.1038/nature14539.

49) Liu K. Identification of fork points on the skeletons of handwritten Chinese characters / Liu K., Huang Y. S., Suen C. Y. // IEEE Trans. Pattern Anal. Mach.

Intell. – Volume 21. – № 10. – October 1999. – pp. 1095–1100. – doi:10.1109/34.799914.

50) Luiz G. Hafemann. Learning features for offline handwritten signature verification using deep convolutional neural networks / Luiz G. Hafemann, R. Sabourin, Luiz Oliveira // Pattern Recognition. – Volume 70. – October 2017. – pp. 163-176. – doi: 10.1016/j.patcog.2017.05.012.

51) M. Wang. Factorized convolutional neural networks / M. Wang, B. Liu, H. Foroosh // IEEE International Conference on Computer Vision Workshops (ICCVW). – pp. 545–554. – ISBN 978-1-5386-1034-3.

52) Matas J. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions / Matas J., Chum O., Urban M., Pajdla T. // Image and Vision Computing. – Volume 22. – № 10. – September 2001. – pp. 761-767. – doi: 10.1016/j.imavis.2004.02.006.

53) Max-Pooling Dropout for Regularization of Convolutional Neural Networks. [Електронний ресурс] : електронний архів наукових статей: arXiv.org / Haibing Wu, Xiaodong Gu // Machine Learning. – December 2015. – Режим доступу до ресурсу: https://arxiv.org/ftp/arxiv/papers/1512/1512.01400.pdf.

54) Mobile Phone Apps for Food Allergies or Intolerances in App Stores: Systematic Search and Quality Assessment Using the Mobile App Rating Scale (MARS) / Mandracchia Floriana, Llauradó Elisabet, Tarro Lucia, Valls Rosa Maria [та ін.] // JMIR mHealth and uHealth. – Volume 8. – № 9. – September 2020. – doi:10.2196/18339.

55) MobileNet and MobileNetV2 [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/api/applications/mobilenet/.

56) MobileNetV2: Inverted Residuals and Linear Bottlenecks / Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov // IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). – Salt Lake City, June 2018. – 14 p. – ISBN:978-1-5386-6420-9.

57) Nissim Silanikove. The Interrelationships between Lactose Intolerance and the Modern Dairy Industry / Nissim Silanikove, Gabriel Leitner, Uzi Merin // Global

Perspectives in Evolutional and Historical Backgrounds – Volume 7. – № 9. – September 2015. – pp. 7312–7331. – doi:10.3390/nu7095340

58) Okun O. Robust Text Detection from Binarized Document Images / O. Okun, Y. Yan, M. Pietikainen // IEEE International Conference on Pattern Recognition. – Quebec City, Canada, August 2002. – ISBN 0-7695-1695-X.

59) Open Food Facts – World [Електронний ресурс] – Режим доступу до ресурсу: https://world.openfoodfacts.org/.

60) Pediatric Age Palm Oil Consumption / Lorenza Di Genova, Laura Cerquiglini, Laura Penta, Anna Biscarini [та ін.] // International journal of environmental research and public health. – April 2018. – Volume 15. – № 4. –pp. 651-661. – doi: 10.3390/ijerph15040651.

61) Perez Luis. The Effectiveness of Data Augmentation in Image Classification using Deep Learning [Електронний ресурс] : електронний архів наукових статей: arXiv.org / Perez Luis, Wang Jason // Computer Vision and Pattern Recognition. – December 2017. – Режим доступу до ресурсу: https://arxiv.org/abs/1712.04621.

62) Rahman Md Atiqur. Optimizing intersection-over-union in deep neural networks for image segmentation / Md Atiqur Rahman, Yang Wang // International Symposium on Visual Computing. – December 2016. – 11 p. – doi:10.1007/978-3-319-50835-1_22.

63) Recognition of Handwritten Chinese Characters Based on Concept Learning / Liang Xu, Yuxi Wang, Xiuxi Li, Ming Pan // IEEE Access. – Volume 7. – July 2019. – pp. 102039 – 102053. – doi: 10.1109/ACCESS.2019.2930799.

64) Safabakhsh R. Document skew detection using minimum-area bounding rectangle / R. Safabakhsh, S. Khadivi // Proceedings EEEI International Conference on Information Technology: Coding and Computing (Cat. No.PR00540). – March 2000. – pp. 253-258 – doi:10.1109/ITCC.2000.844226.

65) Safety evaluation of certain food additives / [publisher World Health Organization] // Seventy-first meeting of the Joint FAO/WHO Expert Committee on

Food Additives (JECFA) (Meeting 69th). – Italy, 2006. – 634 p. – ISBN 9789241660600.

66) Save data in a local database using Room [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/training/data-storage/room.

67) Ship Detection from Ocean SAR Image Based on Local Contrast Variance Weighted Information Entropy / Huo Weibo, Huang Yulin, Pei Jifang, Zhang Qia [та ін.] // Automatic Target Recognition of High Resolution SAR/ISAR Images – Volume 18. – № 4. – March 2018. – doi: 10.3390/s18041196.

68) Takahashi Hiroki. Region graph based text extraction from outdoor images / Takahashi Hiroki, Nakajima Masayuki // Third International Conference on Information Technology and Applications (ICITA'05). – Volume 1. – July 2005. – ISBN:0-7695-2316-1.

69) Text detection and recognition using enhanced MSER detection and a novel OCR technique / Rabiul Islam, Chayan Mondal, Md. Kawsar Azam, Abu Syed Md. [та ін.] // 5th International Conference on Informatics, Electronics and Vision (ICIEV). – May 2016. – doi:10.1109/ICIEV.2016.7760054.

70) The MVVM Pattern [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/previous-versions/msp-n-p/hh848246(v=pandp.10)?redirectedfrom=MSDN.

71) The role of mobile health technologies in allergy care / Matricardi PM, Dramburg S, Alvarez-Perea AD, Antolín-Amérigo DS [та ін.] // An EAACI position paper. Edinburgh Research Explorer. – February 2020. – 32 p. – doi: 10.1111/all.13953.

72) Toxicological evaluation of certain food additives and contaminants / [published World Health Organization] // The forty-first meeting of the Joint FAO/WHO Expert Committee on Food Additives (JEFCA). – Geneva, February 1993. – 301 p. – ISBN 9241660325.

73) Understanding Parameter Sharing (or weights replication) Within Convolutional Neural Networks [Електронний ресурс] – Режим доступу до ресурсу:

https://towardsdatascience.com/understanding-parameter-sharing-or-weights-replication-within-convolutional-neural-networks-cc26db7b645a.

74) Using Adaboost to Detect and Segment Characters from Natural Scenes [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/241758387_Using_Adaboost_to_Detect_and_Segment_Characters_from_Natural_Scenes.

75) Venugopal Anita. Multilayer Perceptron Approach for Character Classification / Anita Venugopal, Mukesh Madanan, Akhil Venugopal // European Journal of Molecular & Clinical Medicine. – Volume 7. – № 11. – 2020. – pp. 1283-1289. – ISSN 2515-8260.

76) ViewModel Overview [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/topic/libraries/architecture/viewmodel.

77) Wagdy Marian. Document Image Binarization Based on Retinex Theory / Marian Wagdy, Ibrahima Faye, Dayang Rohaya Awang Rambli // Electronic Letters on Computer Vision and Image Analysis. – Volume 14. – № 1. – June 2015. – doi:10.5565/rev/elcvia.648.

78) What is Serverless Architecture? [Електронний ресурс] – Режим доступу до ресурсу: https://www.twilio.com/docs/glossary/what-is-serverless-architecture.

79) Xiaobing Wang. Natural scene text detection in multi-channel connected component segmentation / Xiaobing Wang, Yonghong Song, Yuanlin Zhang // 12th International Conference on Document Analysis and Recognition. – August 2013. – ISBN:978-0-7695-4999-6

80) Yao Li. Scene text detection via stroke width / Yao Li, Huchuan Lu // 21st International Conference on Pattern Recognition (ICPR2012). – November 2012. – ISBN:978-4-9906441-0-9.

81) Yi C. Scene text recognition in mobile applications by character descriptor and structure configuration / C. Yi, Y. Tian // IEEE Transactions on Image Processing. – Volume 23. – № 7. – July 2014. – doi: 10.1109/TIP.2014.2317980.

82) Yi-Feng Pan. A Hybrid Approach to Detect and Localize Texts in Natural Scene Images / Yi-Feng Pan, Xinwen Hou, Cheng-Lin Liu // IEEE Transactions on Image Processing. – Volume 20. – № 3. – March 2011. – doi: 10.1109/TIP.2010.2070803.

83) Информационная записка. Пищевые аллергии. ИНФОСАН [Електронний ресурс] / Всемирная организация здравоохранения. – №3. – 2006. – Режим доступу до ресурсу: https://www.who.int/foodsafety/fs_management/No_03_allergy_June06_ru.pdf.

84) Общий стандарт кодекса на маркировку фасованных пищевых продуктов [Електронний ресурс] : електронний ресурс міжнародного стандарту їжі : Codex Alimentarius. – Режим доступу до ресурсу: https://docplayer.com/51766305-Obshchiy-standart-kodeksa-na-markirovku-fasovannyh-pishchevyh-produktov-codex-stan.html.

85) Плахотін В.Я. Теоретичні основи технології харчових виробництв / Плахотін В.Я., Тюрікова І.С., Хомич Г.П. // К.: Центр навчальної літератури. – 2006. – 640 с.

86) Принципы оценки безопасности пищевых добавок и контаминантов в продуктах питания // М.: Изд-во «Медицина» по материалам ВОЗ. – Женева. 1991.

87) Про безпечність та якість харчових продуктів. Закон України від 23.12.1997 р. № 771/97-ВР. – Дата оновлення: 31.05.2011 [Електронний ресурс] – Режим доступу до ресурсу: http://search.ligazakon.ua/l_doc2.nsf/linkl/ed_2011_02_03/Z970771.html.

88) Про затвердження переліку харчових добавок, дозволених для використання у харчових продуктах. Закон України від від 4 січня 1999 р. – Дата оновлення 17.02.2000 [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/12-99-%D0%BF#Text.

89) Про захист прав споживачів. Закон України від 1991 р., № ЗО, ст.379. – Дата оновлення: 27.02.2020 [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/1023-12.

90) Про інформацію для споживачів щодо харчових продуктів. Закон України від 2019 р. [Електронний ресурс] – Режим доступу: https://zakon.rada.gov.ua/laws/show/2639-19#n510.

91) Столярчук П. Г. Метод ідентифікації харчових добавок (підсолоджувачів) з метою виявлення фальсифікації продукції / П. Г. Столярчук // Вісник Нац. техн. ун-ту "ХПІ": зб. наук. пр. Темат. вип.: Нові рішення в сучасних технологіях. Харків: НТУ "ХПІ". – Харків, 2010. – № 46. – С. 3-7.

92) Федорова, А. А. Распознавание английского текста сверточной нейронной сетью / А. А. Федорова // Молодой ученый. – № 14 (118). – 2016. — с. 97-102. [Електронний ресурс] – Режим доступу до ресурсу: https://moluch.ru/archive/118/32732/.

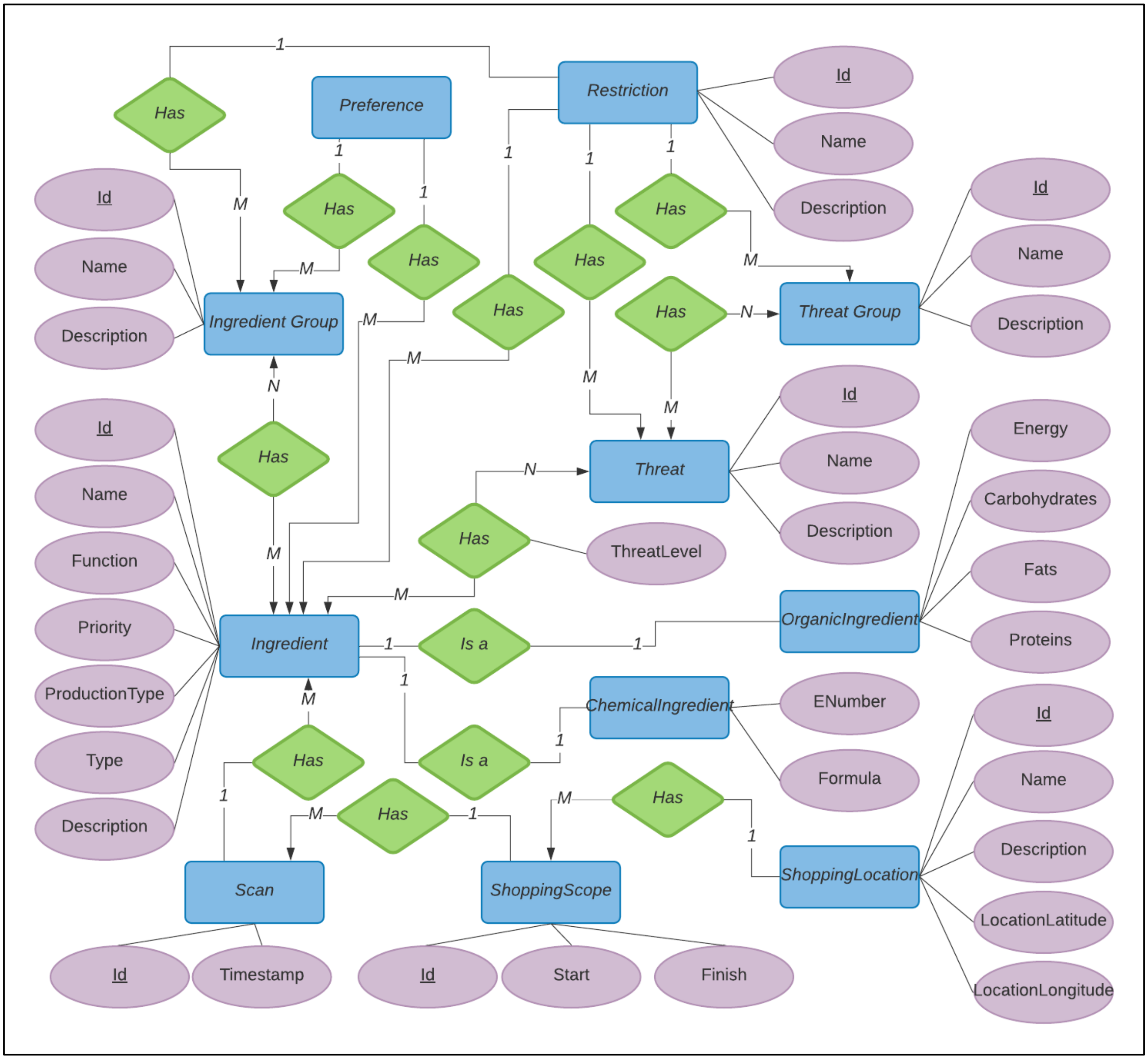
93) Харчова безпека — що потрібно знати [Електронний ресурс] – Режим доступу до ресурсу: https://phc.org.ua/news/kharchova-bezpekascho-potribno-znati.

94) Чи шкідлива пальмова олія [Електронний ресурс] – Режим доступу до ресурсу: https://moz.gov.ua/article/health/chi-shkidliva-palmova-olija.

95) Що треба знати про норми споживання молочних продуктів і проблеми із засвоєнням лактози [Електронний ресурс] – Режим доступу до ресурсу: https://moz.gov.ua/article/health/scho-treba-znati-pro-normi-spozhivannja-molochnih-produktiv-i-problemi-iz-zasvoennjam-laktozi.

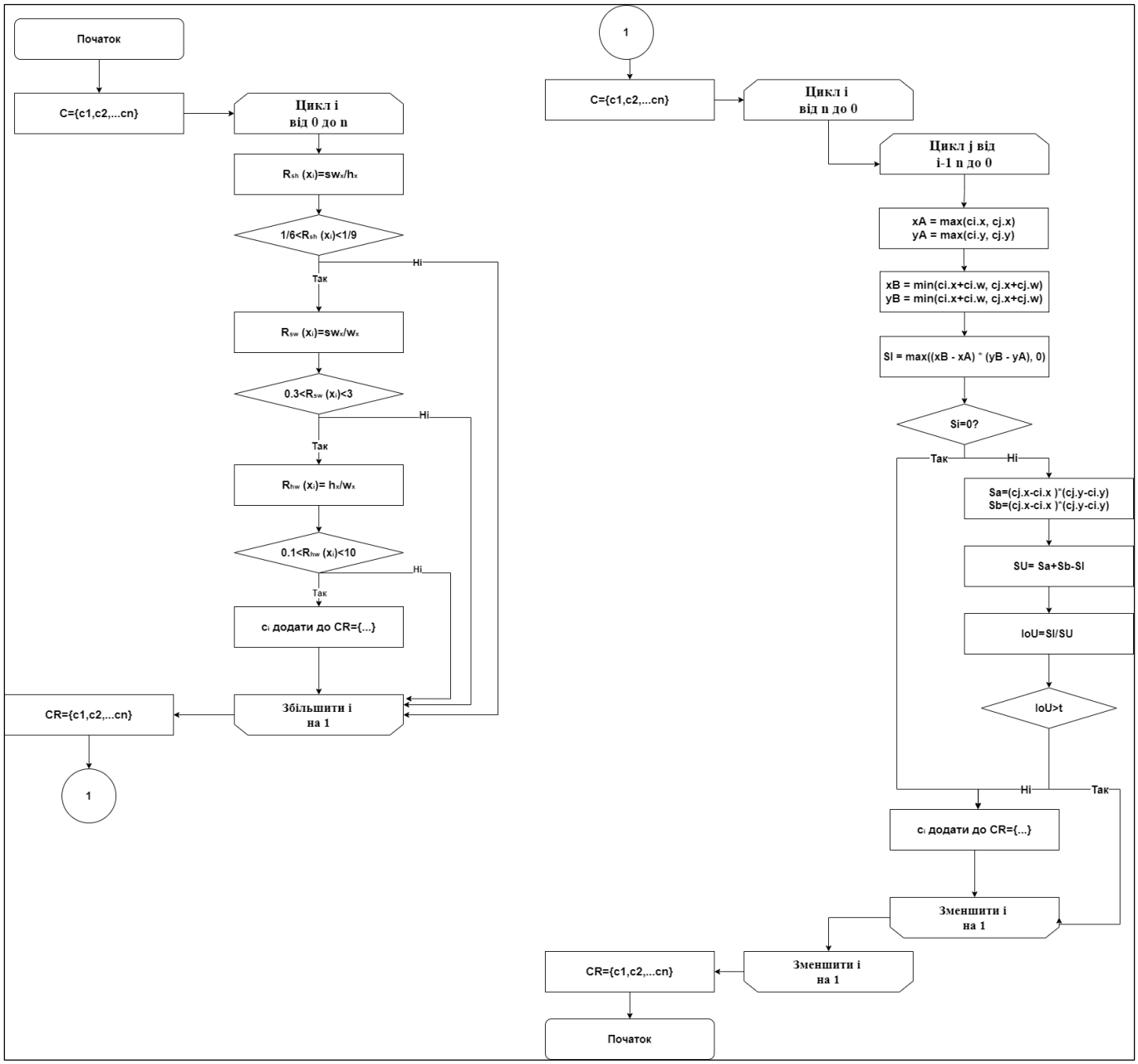
ДОДАТОК А

Інфологічна діаграма бази даних



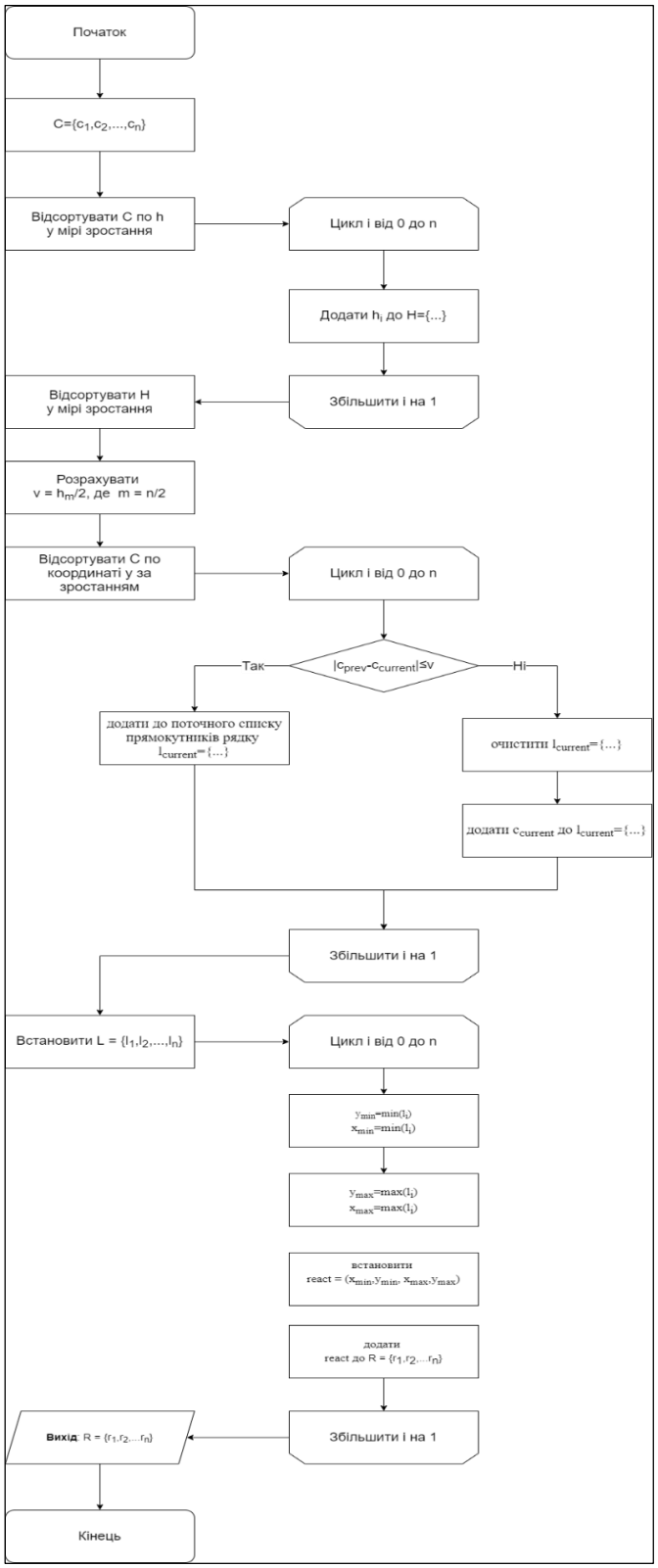
ДОДАТОК Б

Схема структурна роботи системи



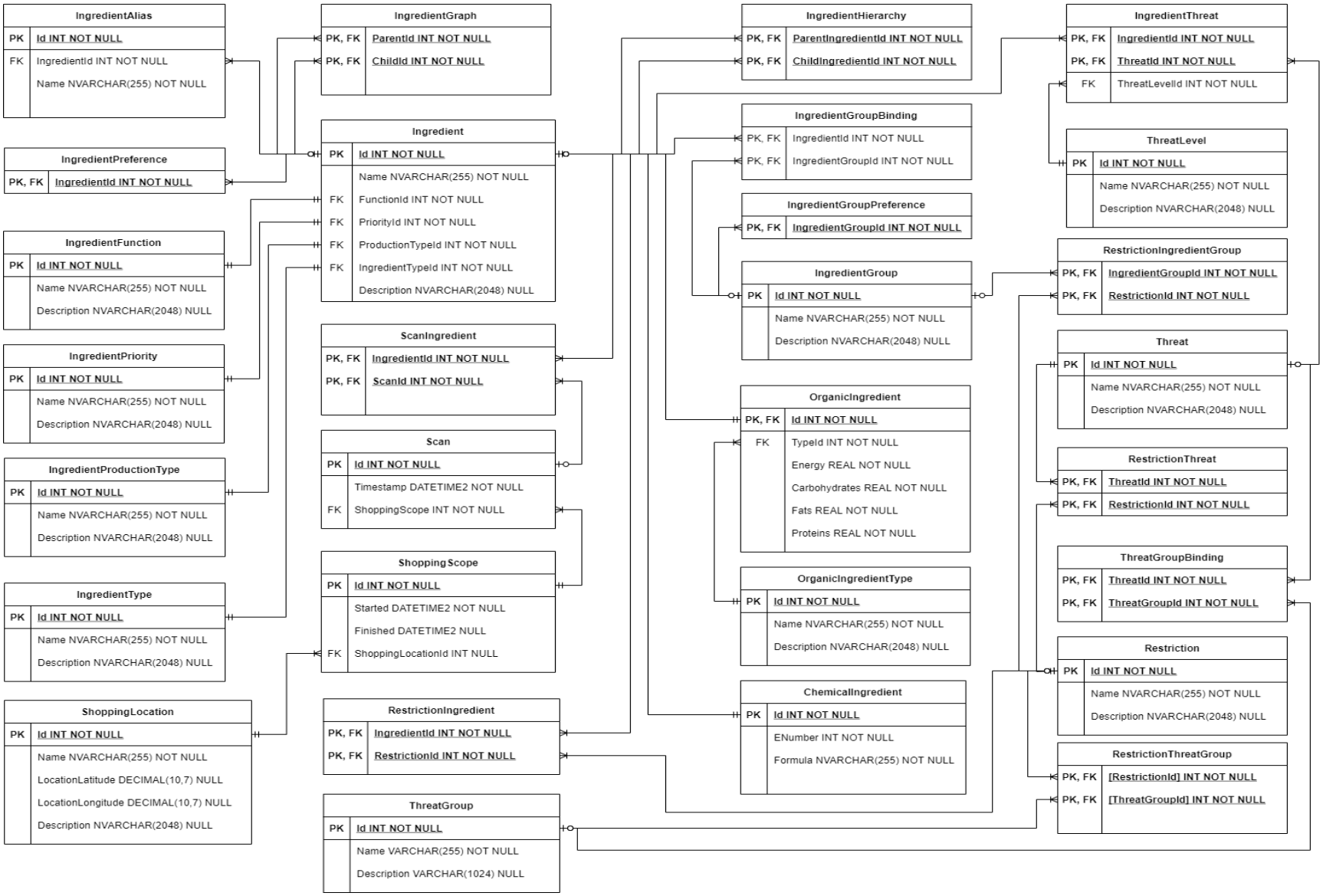
ДОДАТОК В

Алгоритм групування рядків

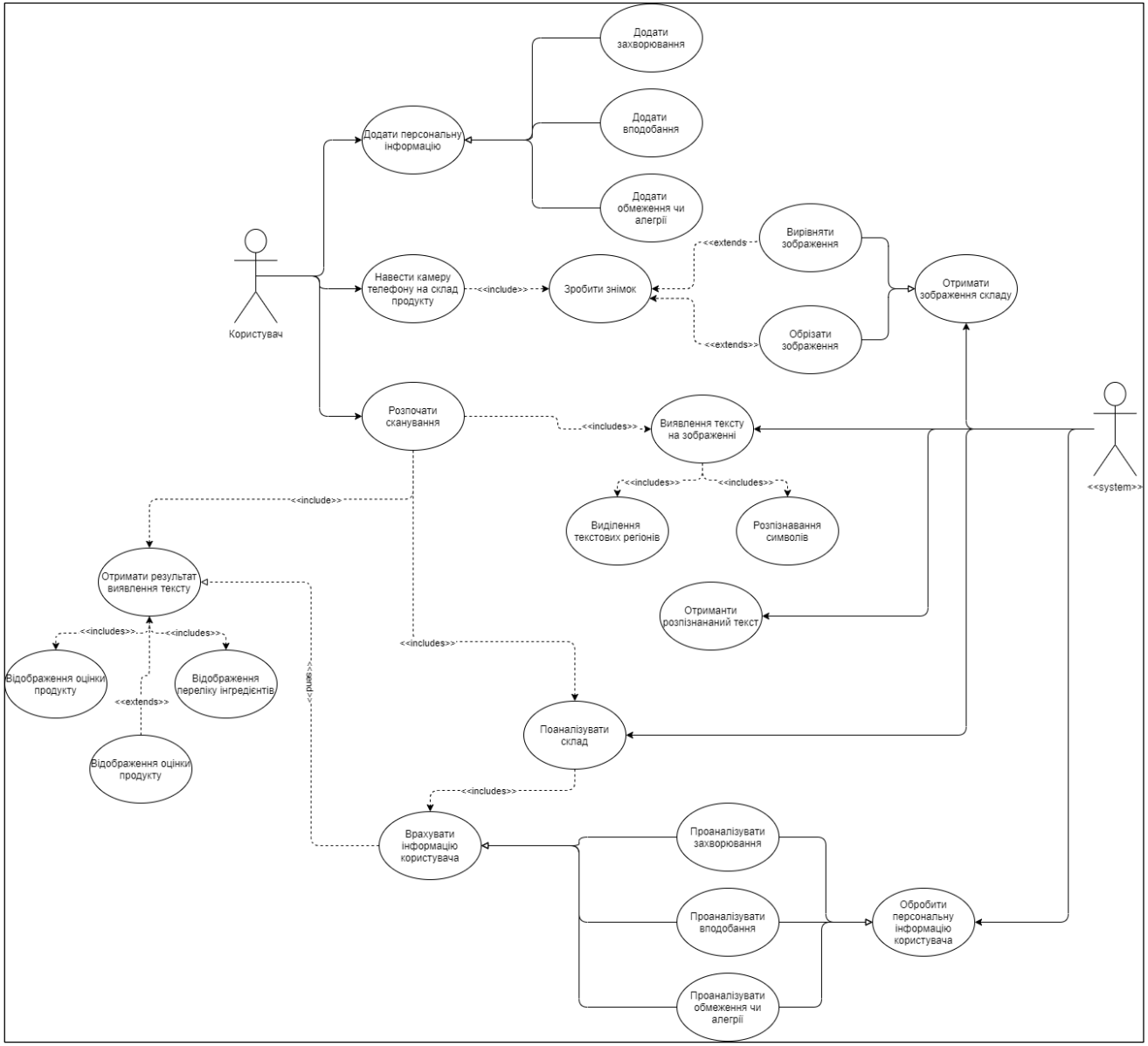


ДОДАТОК Б
Графічні матеріали

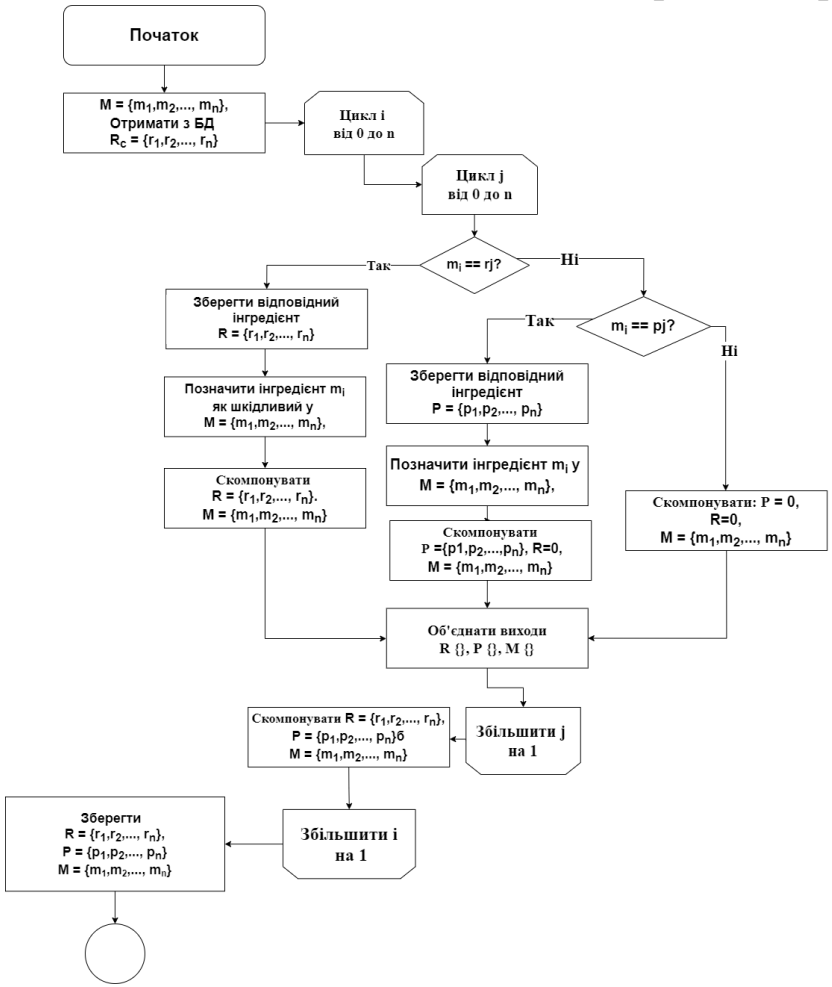
Схема структурна бази даних

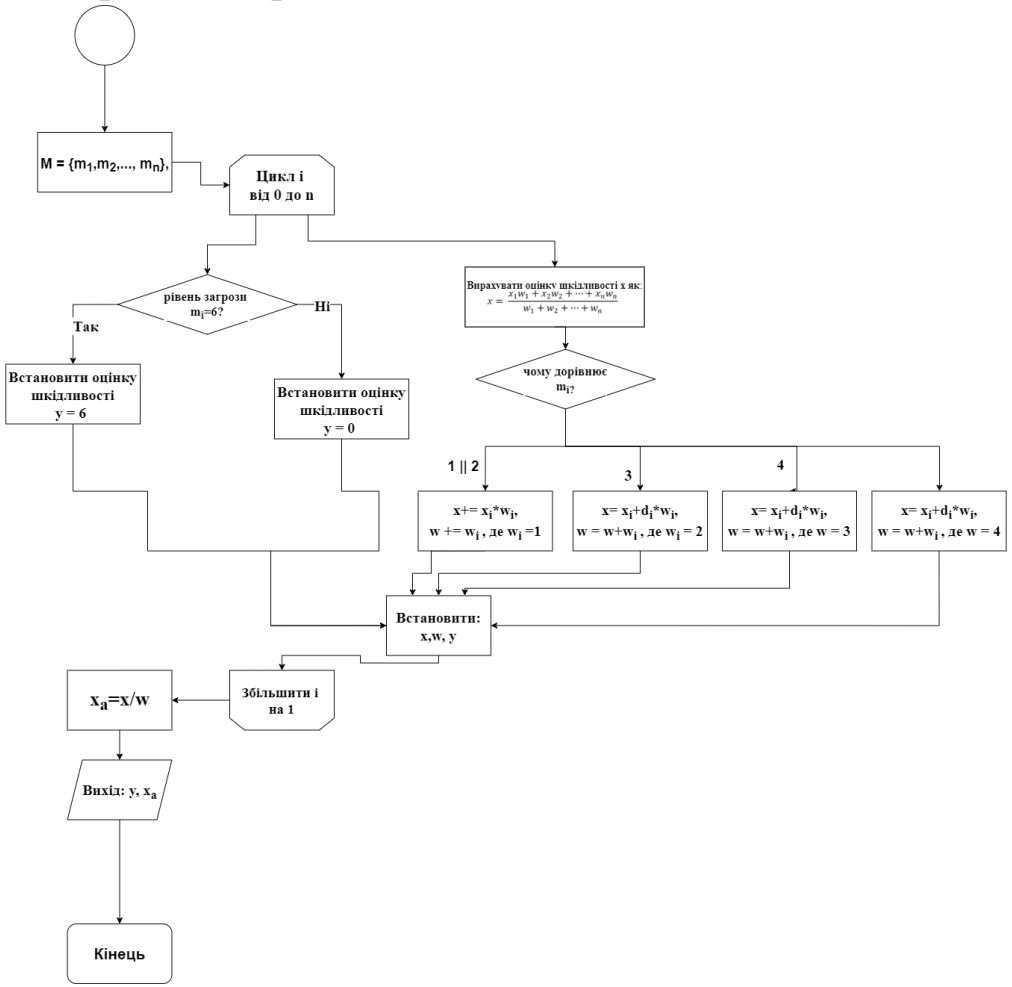


Діаграма використання системи



Алгоритм підтримки прийняття рішень





UNICHECK

Имя пользователя:
Лісовиченко Олег Іванович

ID проверки:
1009305683

Дата проверки:
23.11.2021 07:57:45 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
23.11.2021 07:58:25 EET

ID пользователя:
76913

Название файла: **IT-з03мп_Козак**

Количество страниц: **69** Количество слов: **14814** Количество символов: **112975** Размер файла: **117.88 KB** ID файла: **1009331285**

1.57%
Совпадения

Наибольшее совпадение: **0.23%** с источником из Библиотеки (ID файла: **1009313664**)

0.21% Источники из Интернета	6	Страница 71
1.57% Источники из Библиотеки	136	Страница 71

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

0% Исключений

Нет исключенных источников

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Замененные символы	64

ДОДАТОК Ж

Лістинг коду бібліотеки

Інтерфейс Classifier:

```
package com.example.myapplicationfordiploma.lib;

import org.tensorflow.lite.DataType;
import org.tensorflow.lite.support.model.Model;
import org.tensorflow.lite.support.tensorbuffer.TensorBuffer;

import java.util.HashMap;
import java.util.Map;

public interface Classifier {

    Outputs process(TensorBuffer inputFeature0);

    void close();

    class Outputs {
        private TensorBuffer outputFeature0;

        private Outputs(Model model) {
            this.outputFeature0 =
TensorBuffer.createFixedSize(model.getOutputTensorShape(0), DataType.FLOAT32);
        }

        public TensorBuffer getOutputFeature0AsTensorBuffer() {
            return outputFeature0;
        }

        private Map<Integer, Object> getBuffer() {
            Map<Integer, Object> outputs = new HashMap<>();
            outputs.put(0, outputFeature0.getBuffer());
            return outputs;
        }
    }
}
```

Клас ImageUtils:

```
package com.example.myapplicationfordiploma.lib;

import android.graphics.Bitmap;

import org.opencv.android.OpenCVLoader;
import org.opencv.android.Utils;
import org.opencv.core.Mat;
import org.opencv.core.MatOfPoint;
import org.opencv.core.MatOfRect;
import org.opencv.core.Rect;
import org.opencv.features2d.MSER;
import org.opencv.imgproc.Imgproc;
```

```
import java.util.ArrayList;
import java.util.List;

public class ImageUtils {
    public static Mat binarize(Bitmap imageMatrix) {
        Mat mat = new Mat();
        Mat binarizedMatrix = new Mat();

        Utils.bitmapToMat(imageMatrix, mat);

        Imgproc.cvtColor(mat, binarizedMatrix, Imgproc.COLOR_BGR2GRAY);

        return binarizedMatrix;
    }

    public static List<Rect> detectMserRegions(Bitmap imageMatrix) {
        return detectMserRegions(imageMatrix, new MserOptions.Builder().build());
    }

    public static List<Rect> detectMserRegions(Bitmap imageMatrix, MserOptions options) {
        OpenCVLoader.initDebug();

        MSER mser = MSER.create(options.delta, options.min_area, options.max_area,
options.max_variation, options.min_diversity);

        Mat binarizedMatrix = binarize(imageMatrix);

        List<MatOfPoint> msers = new ArrayList<>();
        MatOfRect bboxes = new MatOfRect();

        mser.detectRegions(binarizedMatrix, msers, bboxes);

        List<Rect> contrastRegions = bboxes.toList();

        return contrastRegions;
    }

    public static class MserOptions {
        private final int delta;
        private final int min_area;
        private final int max_area;
        private final double max_variation;
        private final double min_diversity;

        public MserOptions(int delta, int min_area, int max_area, double max_variation,
double min_diversity) {
            this.delta = delta;
            this.min_area = min_area;
            this.max_area = max_area;
            this.max_variation = max_variation;
            this.min_diversity = min_diversity;
        }

        public static class Builder {
            private static final int DEFAULT_DELTA = 10;
```

```
            private static final int DEFAULT_MIN_AREA = 100;
            private static final int DEFAULT_MAX_AREA = 800;
            private static final double DEFAULT_MAX_VARIATION = 0.25;
            private static final double DEFAULT_MIN_DIVERSITY = 3;

            private int delta;
            private int min_area;
            private int max_area;
            private double max_variation;
            private double min_diversity;

            public Builder() {
                delta = DEFAULT_DELTA;
                min_area = DEFAULT_MIN_AREA;
                max_area = DEFAULT_MAX_AREA;
                max_variation = DEFAULT_MAX_VARIATION;
                min_diversity = DEFAULT_MIN_DIVERSITY;
            }

            public Builder setDelta(int delta) {
                this.delta = delta;
                return this;
            }

            public Builder setMinArea(int min_area) {
                this.min_area = min_area;
                return this;
            }

            public Builder setMaxArea(int max_area) {
                this.max_area = max_area;
                return this;
            }

            public Builder setMaxVariation(double max_variation) {
                this.max_variation = max_variation;
                return this;
            }

            public Builder setMinDiversity(double min_diversity) {
                this.min_diversity = min_diversity;
                return this;
            }

            public MserOptions build() {
                return new MserOptions(delta, min_area, max_area, max_variation,
min_diversity);
            }
        }
    }
}
```

Клас NetRunner:

```
package com.example.myapplicationfordiploma.lib;
```

```
import android.graphics.Bitmap;
import android.graphics.Color;

import org.tensorflow.lite.DataType;
import org.tensorflow.lite.support.tensorbuffer.TensorBuffer;

public class NetRunner {
    public static final int CLASSIFIER_INPUT_SCALAR_SIZE = 1;
    public static final int CLASSIFIER_INPUT_TENSOR_WIDTH = 32;
    public static final int CLASSIFIER_INPUT_TENSOR_HEIGHT = 32;

    public static final int[] CLASSIFIER_SHAPE = new int[]{CLASSIFIER_INPUT_SCALAR_SIZE,
CLASSIFIER_INPUT_TENSOR_WIDTH, CLASSIFIER_INPUT_TENSOR_HEIGHT,
CLASSIFIER_INPUT_SCALAR_SIZE};

    private static final int[] DEFAULT_PIXEL_BUFFER = new
int[CLASSIFIER_INPUT_TENSOR_WIDTH * CLASSIFIER_INPUT_TENSOR_WIDTH];
    private static final float[] DEFAULT_INPUT_BUFFER = new
float[CLASSIFIER_INPUT_TENSOR_HEIGHT * CLASSIFIER_INPUT_TENSOR_HEIGHT];

    private Classifier classifier;
    private char[] classes;

    public NetRunner(Classifier classifier, char[] classes) {
        this.classifier = classifier;
        this.classes = classes;
    }

    public void setClassifier(Classifier classifier, char[] classes) {
        this.classifier = classifier;
        this.classes = classes;
    }

    public char classifyChar(Bitmap charImage) {
        return classifyChar(charImage, DEFAULT_PIXEL_BUFFER, DEFAULT_INPUT_BUFFER);
    }

    public char classifyChar(Bitmap charImage, int[] pixelBuffer, float[] inputBuffer) {
        prepareContentArray(charImage, pixelBuffer, inputBuffer);

        TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(CLASSIFIER_SHAPE,
DataType.FLOAT32);
        inputFeature0.loadArray(inputBuffer);

        Classifier.Outputs outputs = classifier.process(inputFeature0);
        TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();

        float[] result = outputFeature0.getFloatArray();

        int maxAt = 0;
        for (int i = 0; i < result.length; i++) {
            maxAt = result[i] > result[maxAt] ? i : maxAt;
        }

        char recognizedChar = classes[maxAt];
```

```
        return recognizedChar;
    }

    public static void prepareContentArray(Bitmap charImage, int[] pixelBuffer, float[]
inputBuffer) {
        Bitmap scaledCharImage;
        if (charImage.getWidth() > CLASSIFIER_INPUT_TENSOR_WIDTH || charImage.getHeight()
> CLASSIFIER_INPUT_TENSOR_HEIGHT) {
            scaledCharImage = resizeBitmap(charImage, CLASSIFIER_INPUT_TENSOR_WIDTH,
CLASSIFIER_INPUT_TENSOR_HEIGHT);
        } else {
            scaledCharImage = charImage;
        }

        fillPreparedGrayscaleContentArrayWithPadding(scaledCharImage, pixelBuffer,
inputBuffer);
    }

    public static Bitmap resizeBitmap(Bitmap image, int maxWidth, int maxHeight) {
        int width = image.getWidth();
        int height = image.getHeight();

        float ratioBitmap = (float) width / (float) height;
        float ratioMax = (float) maxWidth / (float) maxHeight;

        int finalWidth = maxWidth;
        int finalHeight = maxHeight;

        if (ratioMax > ratioBitmap) {
            finalWidth = (int) ((float) maxHeight * ratioBitmap);
        } else {
            finalHeight = (int) ((float) maxWidth / ratioBitmap);
        }

        image = Bitmap.createScaledBitmap(image, finalWidth, finalHeight, true);
        return image;
    }

    public static void fillPreparedGrayscaleContentArrayWithPadding(Bitmap sourceData,
int[] pixelBuffer, float[] inputBuffer) {

        int contentWidth = sourceData.getWidth();
        int contentHeight = sourceData.getHeight();

        sourceData.getPixels(pixelBuffer, 0, CLASSIFIER_INPUT_TENSOR_WIDTH, 0, 0,
contentWidth, contentHeight);

        int contentIndex = ((CLASSIFIER_INPUT_TENSOR_HEIGHT - contentHeight) / 2) *
CLASSIFIER_INPUT_TENSOR_WIDTH + (CLASSIFIER_INPUT_TENSOR_WIDTH - contentWidth) / 2;

        for (int i = 0; i < contentIndex; i++) {
            inputBuffer[i] = 0;
        }

        for (int i = 0; i < contentHeight; i++) {
            for (int j = 0; j < contentWidth; j++) {
```

```
                int pixelValue = pixelBuffer[i * CLASSIFIER_INPUT_TENSOR_HEIGHT + j];

                int red = Color.red(pixelValue);
                int blue = Color.blue(pixelValue);
                int green = Color.green(pixelValue);

                float grayscale = ((float) (red + blue + green) / 3) / 255;

                inputBuffer[contentIndex++] = grayscale;
            }

            if (i != contentHeight - 1) {
                for (int n = 0; n < CLASSIFIER_INPUT_TENSOR_WIDTH - contentWidth; n++) {
                    inputBuffer[contentIndex + n] = 0;
                }

                contentIndex += (CLASSIFIER_INPUT_TENSOR_WIDTH - contentWidth);
            }
        }

        for (int i = contentIndex; i < inputBuffer.length; i++) {
            inputBuffer[i] = 0;
        }
    }
}
```

Клас RegionService:

```
package com.example.myapplicationfordiploma.lib;

import android.os.Build;

import androidx.annotation.RequiresApi;

import org.opencv.core.Rect;

import java.util.ArrayList;
import java.util.Collection;
import java.util.Comparator;
import java.util.HashSet;
import java.util.List;
import java.util.Set;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

public class RegionService {
    @RequiresApi(api = Build.VERSION_CODES.N)
    public static List<Rect> removeDuplicates(List<Rect> regions, float iouThreshold) {
        List<Rect> regionsWithoutDirectDuplicates =
                regions.stream()
                        .distinct()
                        .sorted(Comparator.comparingInt((Rect r) ->
r.x).thenComparingInt((Rect r) -> r.y))
                        .collect(Collectors.toList());

        Set<Integer> idsToSkip = getIdsToSkip(regionsWithoutDirectDuplicates,
```

```
iouThreshold);

        List<Rect> uniqueRegions =
                IntStream.range(0, regionsWithoutDirectDuplicates.size())
                        .filter(x -> !idsToSkip.contains(x))
                        .mapToObj(regionsWithoutDirectDuplicates::get)
                        .collect(Collectors.toList());

        return uniqueRegions;
    }

    private static Set<Integer> getIdsToSkip(List<Rect> regionsWithoutDirectDuplicates,
float iouThreshold) {
        Set<Integer> idsToSkip = new HashSet<>();

        for (int i = regionsWithoutDirectDuplicates.size() - 1; i >= 0; i--) {
            for (int j = i - 1; j >= 0; j--) {
                Rect rectangleA = regionsWithoutDirectDuplicates.get(i);
                Rect rectangleB = regionsWithoutDirectDuplicates.get(j);

                int intersectionRectangleArea = getIntersectionRectangleArea(rectangleA,
rectangleB);

                if (intersectionRectangleArea != 0) {
                    int areaOfRectangleA = (int) rectangleA.area();

                    if (intersectionRectangleArea == areaOfRectangleA || (float)
intersectionRectangleArea / areaOfRectangleA > iouThreshold) {
                        idsToSkip.add(i);

                        break;
                    } else {
                        int areaOfRectangleB = (int) rectangleB.area();

                        if (intersectionRectangleArea == areaOfRectangleB || (float)
intersectionRectangleArea / areaOfRectangleB > iouThreshold) {
                            idsToSkip.add(j);
                        }
                    }
                }
            }
        }

        return idsToSkip;
    }

    private static int getIntersectionRectangleArea(Rect rectangleA, Rect rectangleB) {
        int xA = Math.max(rectangleA.x, rectangleB.x);
        int yA = Math.max(rectangleA.y, rectangleB.y);
        int xB = Math.min(rectangleA.x + rectangleA.width, rectangleB.x +
rectangleB.width);
        int yB = Math.min(rectangleA.y + rectangleA.height, rectangleB.y +
rectangleB.height);

        return Math.max((xB - xA) * (yB - yA), 0);
    }
```

```
    @RequiresApi(api = Build.VERSION_CODES.N)
    public static List<TextLineRegion> formTextGroups(List<Rect> regions) {
        int halfMedianHeight = getHalfMedianOfAllHeightsOf(regions);
        int halfMedianWidth = getQuarterMedianOfAllWidthsOf(regions);

        ArrayList<ArrayList<Rect>> linesOfBboxes = groupLinesOfBboxes(regions,
halfMedianHeight);

        List<TextLineRegion> textRegions = new ArrayList<>();

        int lineSequenceNumber = 0;
        int wordSequenceNumber = 0;

        for (ArrayList<Rect> lineOfBboxes : linesOfBboxes) {
            ArrayList<ArrayList<Rect>> wordGroups = groupWordsInLineOfBboxes(lineOfBboxes,
halfMedianWidth);

            ArrayList<TextWordRegion> words = new ArrayList<>();

            for (ArrayList<Rect> wordGroup : wordGroups) {
                words.add(new TextWordRegion(wordGroup, wordSequenceNumber++));
            }

            textRegions.add(new TextLineRegion(words, lineSequenceNumber++));
        }

        return textRegions;
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    private static int getHalfMedianOfAllHeightsOf(List<Rect> regions) {
        int[] heights = regions.stream().mapToInt(x -> x.height).sorted().toArray();

        int halfMedianHeight = heights[heights.length / 2] / 2;

        return halfMedianHeight;
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    private static int getQuarterMedianOfAllWidthsOf(List<Rect> regions) {
        int[] widths = regions.stream().mapToInt(x -> x.width).sorted().toArray();

        int quarterMedianWidth = widths[widths.length / 2] / 4;

        return quarterMedianWidth;
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    private static ArrayList<ArrayList<Rect>> groupLinesOfBboxes(List<Rect> bboxes, int
halfMedianHeight) {
        List<Rect> ySortedBboxes = bboxes.stream().sorted(Comparator.comparingInt(b ->
b.y)).collect(Collectors.toList());

        ArrayList<ArrayList<Rect>> groups = new ArrayList<>();
        Rect prevBbox = null;
```

```
        ArrayList<Rect> currentGroup = new ArrayList<>();

        for (Rect currentBbox : ySortedBboxes) {
            if (prevBbox == null || Math.abs(currentBbox.y - prevBbox.y) <=
halfMedianHeight) {
                currentGroup.add(currentBbox);
            } else {
                groups.add(currentGroup);
                currentGroup = new ArrayList<>();
                currentGroup.add(currentBbox);
            }
            prevBbox = currentBbox;
        }

        if (currentGroup.size() > 0) {
            groups.add(currentGroup);
        }

        return groups;
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    private static ArrayList<ArrayList<Rect>> groupWordsInLineOfBboxes(List<Rect> bboxes,
int halfMedianWidth) {
        List<Rect> xSortedBboxes = bboxes.stream().sorted(Comparator.comparingInt(b ->
b.x)).collect(Collectors.toList());

        ArrayList<ArrayList<Rect>> groups = new ArrayList<>();
        Rect prevBbox = null;
        ArrayList<Rect> currentGroup = new ArrayList<>();

        for (Rect currentBbox : xSortedBboxes) {
            if (prevBbox != null && Math.abs(currentBbox.x - (prevBbox.x +
prevBbox.width)) > halfMedianWidth) {
                groups.add(currentGroup);
                currentGroup = new ArrayList<>();
            }
            currentGroup.add(currentBbox);
            prevBbox = currentBbox;
        }

        if (currentGroup.size() > 0) {
            groups.add(currentGroup);
        }

        return groups;
    }

    public static class TextLineRegion extends ArrayList<TextWordRegion> {
        public final int sequenceNumber;

        public TextLineRegion(Collection<? extends TextWordRegion> textWordRegions, int
sequenceNumber) {
            super(textWordRegions);
            this.sequenceNumber = sequenceNumber;
        }
```

```
    }

    public static class TextWordRegion extends ArrayList<Rect> {
        public final int sequenceNumber;

        public TextWordRegion(Collection<? extends Rect> rects, int sequenceNumber) {
            super(rects);
            this.sequenceNumber = sequenceNumber;
        }
    }
}
```

Клас RegionService:

```
package com.example.myapplicationfordiploma.lib;

import android.graphics.Bitmap;
import android.os.Build;
import android.text.TextUtils;

import androidx.annotation.RequiresApi;

import org.opencv.core.Rect;

import java.util.Comparator;
import java.util.List;
import java.util.concurrent.ConcurrentMap;
import java.util.function.Supplier;
import java.util.stream.Collectors;

public class TextExtractor {
    private static final float DEFAULT_IOU_THRESHOLD = 0.8f;

    @RequiresApi(api = Build.VERSION_CODES.N)
    public static String getText(Bitmap image) {
        return getText(image, DEFAULT_IOU_THRESHOLD);
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    public static String getText(Bitmap sourceImage, float iouThreshold) {
        List<Rect> regions = ImageUtils.detectMserRegions(sourceImage);

        List<Rect> regionsWithoutDuplicates = RegionService.removeDuplicates(regions,
iouThreshold);

        List<RegionService.TextLineRegion> textLineRegions =
RegionService.formTextGroups(regionsWithoutDuplicates);

        Supplier<int[]> pixelBufferFactory = createPixelBufferFactory();
        Supplier<float[]> inputBufferFactory = createInputBufferFactory();
        Supplier<NetRunner> netRunnerFactory = createNetRunnerFactory();

        ConcurrentMap<Integer, String> linesOfText =
                textLineRegions
                        .parallelStream()
                        .collect(
```

```
                                Collectors.toConcurrentMap(
                                        x -> x.sequenceNumber,
                                        x -> classifyWord(x, sourceImage,
pixelBufferFactory, inputBufferFactory, netRunnerFactory)
                                )
                        );

        List<String> textWords =
                linesOfText.entrySet().stream()
                        .sorted(Comparator.comparingInt(x -> x.getKey()))
                        .map(x -> x.getValue())
                        .collect(Collectors.toList());

        String text = TextUtils.join(System.lineSeparator(), textWords);

        return text;
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    public static String classifyWord(RegionService.TextLineRegion textLineRegion, Bitmap
image, Supplier<int[]> pixelBufferFactory, Supplier<float[]> inputBufferFactory,
Supplier<NetRunner> netRunnerFactory) {
        NetRunner netRunner = netRunnerFactory.get();

        int[] pixelBuffer = pixelBufferFactory.get();
        float[] inputBuffer = inputBufferFactory.get();

        StringBuilder stringBuilder = new StringBuilder();

        for (RegionService.TextWordRegion wordRegions : textLineRegion) {
            for (Rect charRegion : wordRegions) {
                Bitmap charImage = Bitmap.createBitmap(image, charRegion.x, charRegion.y,
charRegion.width, charRegion.height);

                char recognitionResult = netRunner.classifyChar(charImage, pixelBuffer,
inputBuffer);

                stringBuilder.append(recognitionResult);
            }

            stringBuilder.append(' ');
        }

        return stringBuilder.toString();
    }

    private static Supplier<int[]> createPixelBufferFactory(){
        ThreadLocal<int[]> threadLocalBuffer = new ThreadLocal<>();
        Supplier<int[]> bufferFactory = () -> {
            int[] buffer = threadLocalBuffer.get();

            if (buffer == null) {
                buffer = new int[NetRunner.CLASSIFIER_INPUT_TENSOR_WIDTH *
NetRunner.CLASSIFIER_INPUT_TENSOR_HEIGHT];
                threadLocalBuffer.set(buffer);
            }
```

```
            return buffer;
        };

        return bufferFactory;
    }

    private static Supplier<float[]> createInputBufferFactory() {
        ThreadLocal<float[]> threadLocalInputBuffer = new ThreadLocal<>();
        Supplier<float[]> inputBufferFactory = () -> {
            float[] buffer = threadLocalInputBuffer.get();

            if (buffer == null) {
                buffer = new float[NetRunner.CLASSIFIER_INPUT_TENSOR_WIDTH *
NetRunner.CLASSIFIER_INPUT_TENSOR_HEIGHT];
                threadLocalInputBuffer.set(buffer);
            }

            return buffer;
        };

        return inputBufferFactory;
    }

    private static Supplier<NetRunner> createNetRunnerFactory() {
        ThreadLocal<NetRunner> threadLocalNetRunner = new ThreadLocal<>();
        Supplier<NetRunner> netRunnerFactory = () -> {
            NetRunner netRunner = threadLocalNetRunner.get();

            if (netRunner == null) {
                netRunner = new NetRunner(new DefaultModel(), DefaultModel.CLASSES);
                threadLocalNetRunner.set(netRunner);
            }

            return netRunner;
        };

        return netRunnerFactory;
    }
}
```

Лістинг коду генератору навчальних даних

main.py

```
from PIL import Image, ImageDraw
import numpy as np
import random
import os
from fonts import load_fonts
from ImagePermutator import ImagePermutator

imgHeight = 32
imgWidth = 32
```

```
img_padding_width = 1

min_font_size = 20
max_font_size = 50

image_vertical_shrink_min = 0.75
image_vertical_shrink_max = 1.25
image_horizontal_shrink_min = 0.75
image_horizontal_shrink_max = 1.25

image_rotation_min = -5
image_rotation_max = 5

image_blur_min = 0
image_blur_max = 1

salt_and_pepper_amount = 0.075

chars = "абвгґдеєжзиіїйклмнопрстуфхцчшщьюя"
ignore_chars = "!\"#$%&\'()*+,-./123456789:;<=>?@[]\\^_{}~"

destination_folder = ""

amount_of_images_per_font_char_combination = 40
amount_of_images_per_font_char_to_ignore_combination = 3

def getBackgroundAndForeground():
    foreground = random.randint(0, 255)
    delta = random.randint(50, 255)

    if foreground <= 50:
        background = min(foreground + delta, 255)
    elif foreground >= 205:
        background = max(foreground - delta, 0)
    else:
        if bool(random.getrandbits(1)):
            background = min(foreground + delta, 255)
        else:
            background = max(foreground - delta, 0)

    return (background, background, background), (foreground, foreground, foreground)


def get_content_top_offset(image_array, background):
    height = image_array.shape[0]
    width = image_array.shape[1]

    for i in range(0, height):
        for j in range(0, width):
            if tuple(image_array[i][j]) != background:
                return i

    raise Exception("content on image was not found.")


def trim_char_image(image, background):
```

```
    image_array = np.asarray(image)

    content_height = get_content_top_offset(image_array, background)

    trimmed_image = image_array[content_height:]

    pil_image = Image.fromarray(trimmed_image)

    return pil_image


def add_padding(pil_img, xy, new_size, color):
    new_width, new_height = new_size
    result = Image.new(pil_img.mode, (new_width, new_height), color)
    result.paste(pil_img, xy)
    return result


def produce_image(char, font_builder):
    foreground, background = getBackgroundAndForeground()

    font_size = get_random_biased_to_max(min_font_size, max_font_size)
    font = font_builder.get_font(font_size)

    width, height = font.getsize(char)
    char_image = Image.new("RGB", (width, height), background)
    ImageDraw.Draw(char_image).text((0, 0), char, font=font, fill=foreground)

    image_trimmed = trim_char_image(char_image, background)

    image = add_border(image_trimmed, img_padding_width, background)

    permutator = ImagePermutator(image_rotation_min, image_rotation_max,
image_vertical_shrink_min, image_vertical_shrink_max, image_horizontal_shrink_min,
image_horizontal_shrink_max, image_blur_min, image_blur_max, salt_and_pepper_amount,
background)

    image = permutator.do_permutations(image)

    image_that_fits = resize_image_to_fit(image, imgWidth, imgHeight) if image.width >
imgWidth or image.height > imgHeight else image

    image_padded_to_size = pad_image_to_size(image_that_fits)

    return image_padded_to_size


def pad_image_to_size(image):
    real_width, real_height = image.size

    position_x = int((imgWidth - real_width) / 2)
    position_y = int((imgHeight - real_height) / 2)

    image_padded_to_size = add_padding(image, (position_x, position_y), (imgWidth,
imgHeight), (0, 0, 0))
```

```
    return image_padded_to_size


def resize_image_to_fit(image, max_width, max_height):
    if max_height > 0 and max_width > 0:
        ratio_original = image.width / image.height
        ratio_max = max_width / max_height

        final_width = max_width
        final_height = max_height

        if ratio_max > ratio_original:
            final_width = int(max_height * ratio_original)
        else:
            final_height = int(max_width / ratio_original)

        image = image.resize((final_width, final_height), Image.ANTIALIAS)

        return image
    else:
        return image


def add_border(pil_img, trim_width, color):
    width, height = pil_img.size
    image_with_border = add_padding(pil_img, (trim_width, trim_width), (width + 2 *
trim_width, height + 2 * trim_width), color)
    return image_with_border


def validate_destination_folder(path):
    if os.path.exists(path) and os.path.isdir(path):
        if os.listdir(path):
            raise Exception("Directory not is empty")
    else:
        raise Exception("Given directory doesn't exist")


def get_random_biased_to_max(min, max):
    return int((1 - abs(random.random() - random.random())) * (1 + max - min) + min)


def main():
    validate_destination_folder(destination_folder)

    font_builders = load_fonts(chars)

    image_index_by_chars = {}

    for char in chars:
        image_index_by_chars[char] = 0

        char_folder = rf"{destination_folder}\{char}"
        os.makedirs(char_folder)

        for font_builder in font_builders:
```

```
            for c in [char.lower(), char.upper()]:
                print(f"drawing '{c}' using font {font_builder.font_file}")
                for i in range(amount_of_images_per_font_char_combination):
                    image_index = image_index_by_chars[char]

                    text_on_image = c + random.choice(chars) if random.choice([True,
False, False, False, False, False, False]) else c

                    image = produce_image(text_on_image, font_builder)

                    image.save(rf"{char_folder}\{image_index:06d}.jpg", quality=95)

                    image_index_by_chars[char] = image_index + 1

    char_folder = rf"{destination_folder}\_"
    os.makedirs(char_folder)
    ignore_chars_image_index = 0

    for c in ignore_chars:
        for font_builder in font_builders:
            print(f"drawing '{c}' using font {font_builder.font_file}")
            for i in range(amount_of_images_per_font_char_to_ignore_combination):
                image = produce_image(c, font_builder)

                image.save(rf"{char_folder}\{ignore_chars_image_index:06d}.jpg",
quality=95)

                ignore_chars_image_index += 1

main()
```

ImagePermutator.py

```
from PIL import Image, ImageFilter, ImageOps
import random
import numpy as np


class ImagePermutator:
    def __init__(self, image_rotation_min, image_rotation_max, image_vertical_shrink_min,
image_vertical_shrink_max, image_horizontal_shrink_min, image_horizontal_shrink_max,
image_blur_min, image_blur_max, salt_and_pepper_amount, background):
        self.salt_and_pepper_amount = salt_and_pepper_amount
        self.image_blur_max = image_blur_max
        self.image_blur_min = image_blur_min
        self.image_horizontal_shrink_max = image_horizontal_shrink_max
        self.image_horizontal_shrink_min = image_horizontal_shrink_min
        self.background = background
        self.image_vertical_shrink_max = image_vertical_shrink_max
        self.image_vertical_shrink_min = image_vertical_shrink_min
        self.image_rotation_max = image_rotation_max
        self.image_rotation_min = image_rotation_min
        self.permutations = [self.do_rotation_permutation, self.do_shrink_permutation,
self.do_blur_permutation, self.add_salt_and_pepper]

    def do_permutations(self, image):
        random.shuffle(self.permutations)
```

```
        for permutation in self.permutations:
            image = permutation(image)

        return image

    def do_rotation_permutation(self, image):
        if random.choice([True, False]):
            rotation_angle = random.randint(self.image_rotation_min,
self.image_rotation_max)
            image = image.rotate(rotation_angle, expand=True, fillcolor=self.background)

        return image

    def do_shrink_permutation(self, image):
        if random.choice([True, False]):
            current_width, current_height = image.size

            image_vertical_shrink = random.uniform(self.image_vertical_shrink_min,
self.image_vertical_shrink_max)
            image_horizontal_shrink = random.uniform(self.image_horizontal_shrink_min,
self.image_horizontal_shrink_max)

            image = image.resize((int(current_width * image_vertical_shrink),
int(current_height * image_horizontal_shrink)), Image.ANTIALIAS)

        return image

    def do_blur_permutation(self, image):
        if random.choice([True, False, False, False]):
            image_blur = random.uniform(self.image_blur_min, self.image_blur_max)
            image = image.filter(ImageFilter.GaussianBlur(image_blur))

        return image

    def add_salt_and_pepper(self, image):
        if random.choice([True, False]):
            output = np.copy(np.array(image))
            grain_amount = random.uniform(0, self.salt_and_pepper_amount)

            # add salt
            nb_salt = np.ceil(grain_amount * output.size * 0.5)
            coords = [np.random.randint(0, i - 1, int(nb_salt)) for i in output.shape]
            output[coords] = 255

            # add pepper
            nb_pepper = np.ceil(grain_amount * output.size * 0.5)
            coords = [np.random.randint(0, i - 1, int(nb_pepper)) for i in output.shape]
            output[coords] = 0

            noizy_image = np.dot(output[...,:3], [0.299, 0.587, 0.114])

            int_noizy_image = noizy_image.astype(np.uint8)

            stacked_img = np.stack((int_noizy_image,) * 3, axis=-1)
```

```
            return Image.fromarray(stacked_img)

        return image
```

Лістинг коду створення та навчання моделі

main.py

```
import os
from PIL import Image
import numpy as np
from sklearn.model_selection import train_test_split
import keras
import tensorflow as tf
from datetime import datetime
from ThresholdCallback import ThresholdCallback

version = "v3"

dataset_path = ''
models_path = ''

checkpoint_to_save_lite_model_from = ""
model_load_name = ''
lite_model_path = ""

model_training_stop_override_file_path = "modelTrainingStopOverride.txt"

chars = "абвгґдеєжзиіїйклмнопрстуфхцчшщьюя_"
input_shape = (32, 32, 1)
num_classes = 34

train_model = False
convert_model_to_lite = False
convert_model_to_lite_from_checkpoint = False
validate_checkpoints = False


def read_dataset():
    x = []
    y = []

    for letter in sorted(o for o in os.listdir(dataset_path) if
os.path.isdir(os.path.join(dataset_path, o))):
        print(f"Loading letter {letter}")
        letter_dir = os.path.join(dataset_path, letter)

        for image_file in [o for o in os.listdir(letter_dir) if
os.path.isfile(os.path.join(letter_dir, o))]:
            x.append(read_image(os.path.join(letter_dir, image_file)))
            y.append(chars.index(letter))

    return train_test_split(np.array(x), np.array(y))
```

```
def read_image(path):
    im = np.array(Image.open(path).convert('L')).astype("float32") / 255
    return im


def load_dataset():
    x_train, x_test, y_train, y_test = read_dataset()

    x_train = np.expand_dims(x_train, -1)
    x_test = np.expand_dims(x_test, -1)

    y_train = keras.utils.np_utils.to_categorical(y_train, num_classes)
    y_test = keras.utils.np_utils.to_categorical(y_test, num_classes)

    return x_train, x_test, y_train, y_test


def fit_model():
    x_train, x_test, y_train, y_test = load_dataset()

    model = tf.keras.applications.MobileNetV2(
        input_shape=input_shape,
        alpha=1.0,
        include_top=True,
        weights=None,
        classes=num_classes,
        classifier_activation="softmax"
    )

    batch_size = 256
    epochs = 1000000

    model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

    manual_override_callback =
ThresholdCallback(override_file_path=model_training_stop_override_file_path)

    checkpoint_files_folder = os.path.join(models_path, f"model-
{datetime.today().strftime('%b-%d-%YT%H-%M-%S')}-checkpoints")
    os.makedirs(checkpoint_files_folder)

    checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
        filepath=os.path.join(checkpoint_files_folder, '{epoch:02d}.hdf5'),
        save_weights_only=True,
        monitor='val_accuracy',
        mode='max',
        save_freq='epoch')

    model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
validation_split=0.1, callbacks=[manual_override_callback, checkpoint_callback])

    score = model.evaluate(x_test, y_test, verbose=0)
    print("Test loss:", score[0])
    print("Test accuracy:", score[1])

    model_save_folder = f"model-{datetime.today().strftime('%b-%d-%YT%H-%M-%S')}"
```

```
    model.save(os.path.join(models_path, model_save_folder))
    print(f"saved model '{model_save_folder}'")


def save_full_model_as_lite():
    model = keras.models.load_model(os.path.join(models_path, model_load_name))

    save_model_as_lite(model)


def save_model_as_lite_from_checkpoint():
    checkpoints_path = os.path.join(models_path, f'{model_load_name}-checkpoints')

    model = tf.keras.applications.MobileNetV2(
        input_shape=input_shape,
        alpha=1.0,
        include_top=True,
        weights=os.path.join(checkpoints_path, checkpoint_to_save_lite_model_from),
        classes=num_classes,
        classifier_activation="softmax"
    )

    model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

    save_model_as_lite(model)


def save_model_as_lite(model):
    converter = tf.lite.TFLiteConverter.from_keras_model(model)
    tflite_model = converter.convert()
    open(lite_model_path, "wb").write(tflite_model)

def do_checkpoint_validation():
    checkpoints_path = os.path.join(models_path, f'{model_load_name}-checkpoints')
    checkpoints_files = [o for o in os.listdir(checkpoints_path) if
os.path.isfile(os.path.join(checkpoints_path, o))]

    x_train, x_test, y_train, y_test = load_dataset()

    for checkpoints_file in checkpoints_files:
        model = tf.keras.applications.MobileNetV2(
            input_shape=input_shape,
            alpha=1.0,
            include_top=True,
            weights=os.path.join(checkpoints_path, checkpoints_file),
            classes=num_classes,
            classifier_activation="softmax"
        )

        model.compile(loss="categorical_crossentropy", optimizer="adam",
metrics=["accuracy"])

        score = model.evaluate(x_test, y_test, verbose=0)
        print(f"Checkpoint {checkpoints_file} -> Test loss: {score[0]}, Test accuracy:
{score[1]}")
```

```
def main():
    if train_model:
        fit_model()
    elif convert_model_to_lite:
        save_full_model_as_lite()
    elif convert_model_to_lite_from_checkpoint:
        save_model_as_lite_from_checkpoint()
    elif validate_checkpoints:
        do_checkpoint_validation()

main()
```