Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА IOS-ЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ ВІТАМІННОГО СКЛАДУ ФРУКТІВ ЗА ЇХ ЗОБРАЖЕННЯМ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-2
Черноусова М. С.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір С.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Черноусовій Марії Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка iOS-застосунку для розпізнавання вітамінного складу фруктів за їх зображенням

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 22 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека Core ML, бібліотека Core Data, бібліотека AVFoundation, бібліотека Foundation, бібліотека UIKit, мова програмування Swift, середовище розробки Xcode.

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналіз існуючих програмних застосунків для класифікації фруктів.
2. Аналіз існуючих технічних засобів для створення iOS-застосунків для розпізнавання та класифікації фруктів.
3. Створення методології розпізнавання фруктів на зображеннях.
4. Розробка та тренування моделі для розпізнавання фруктів.
5. Розробка iOS-застосунку для розпізнавання фруктів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми розпізнавання фруктів, постановка задачі, сучасні методи класифікації, тренувальні зображення, тестові зображення, результати тренування моделі, скріншоти роботи застосунку.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-15.04.23	
3	Аналіз літератури з досліджуваної проблеми	16.04.23-19.04.23	
4	Аналіз технічних засобів	20.04.23-22.04.23	
5	Розробка та тренування моделі	23.04.23-29.04.23	
6	Програмна реалізація	30.04.23-12.05.23	
7	Оформлення пояснювальної записки	13.05.23-24.05.23	
8	Перевірка на плагіат	25.05.23	
9	Рецензування	26.05.23	
10	Підготовка презентації та доповіді	27.05.23-30.05.23	
11	Занесення роботи в електронний архів	31.05.23	
12	Попередній захист кваліфікаційної роботи	31.05.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Машталір С.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 67 с., 1 табл., 29 рис., 59 джерел.

КОМП'ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ТА КЛАСИФІКАЦІЯ ОБРАЗІВ, ГЛИБОКЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, MPSCNN, CORE ML, CORE DATA, IOS.

Об'єктом роботи є зображення фруктів, які необхідно класифікувати для визначення їх вітамінного складу.

Метою роботи є розробка iOS застосунку із застосуванням згорткової нейронної мережі, спрямованого на розпізнання фруктів та їх вітамінного складу.

Проведено дослідження класичних методів розпізнавання та класифікації об'єктів на основі нейронних мереж. Досліджено можливості нейронних моделей MPSCNN, особливості роботи бібліотеки Core ML та бібліотеки Core Data. Проведено дослідження методів на основі згорткових нейронних мереж та моделі MPSCNN.

У результаті роботи отримано iOS застосунок для розпізнавання та класифікації фруктів та визначення їх вітамінного складу.

COMPUTER VISION, PICTURE RECOGNITION AND CLASSIFICATION, DEEP LEARNING, NEURAL NETWORKS, CONVERSIONAL NEURAL NETWORKS, MPSCNN, CORE ML, CORE DATA, IOS.

The object of the research is the image of fruits, which must be classified in order to determine their vitamin composition.

The aim of the research is to develop an iOS application using a convolutional neural network aimed at recognizing fruits and their vitamin composition.

Classical methods of object recognition and classification based on neural networks have been studied. The capabilities of the MPSCNN neural models, the features of the Core ML library and the Core Data library have been studied. A study of methods based on convolutional neural networks and the MPSCNN model was conducted.

As a result of the work, an iOS application for recognizing and classifying fruits and determining their vitamin composition was obtained.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 6
Вступ 7
1 Аналіз проблеми розпізнавання візуальних об’єктів 8
1.1 Проблематика виявлення фруктів на зображеннях 8
1.2 Аналіз існуючих методів розпізнавання фруктів 11
1.3 Види нейронних мереж для роботи з зображеннями. 14
1.3.1 Перцептрон Розенблатта 15
1.3.2 Генеративна нейронна мережа 16
1.3.3 Згорткова нейронна мережа 18
1.4 Засоби для роботи з нейронними мережами 21
1.5 Постановка задачі 24
2 Особливості розпізнавання візуальних об’єктів в iOS застосунках 25
2.1 Принципи побудови мобільних застосунків з використанням нейронних мереж 25
2.2 Методологія розпізнавання фруктів на зображенні 26
2.3 Архітектура MPSCNN 29
2.4 Використання бібліотеки CoreML для навчання 32
2.5 Збереження даних у застосунку за допомогою CoreData 35
3 Проєктування програмного продукту 38
3.1 Обґрунтування вибору середовища програмної реалізації 38
3.2 Програмна реалізація застосунку 41
3.2.1 Розробка класів та тренування моделі. 41
3.2.2 Використання API для визначення характеристик класифікованого об’єкту 45
3.2.3 Створення скануючого вікна для отримання зображення 50
3.2.4 Збереження даних у застосунку 52
3.3 Тестування розробленого застосунку та аналіз результатів. 56
3.4 Перспективи подальшої роботи 59
Висновки 60
Перелік джерел посилання 61

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

BIC – Border-Interior Classification (класифікація внутрішньої частини)

CCV – Color Coherence Vector (вектор колірної когерентності)

BPNN – Back Propagation Neural Network

ISADH – Improved Sum and Difference Histogram (вдосконалена функція текстури гістограми суми та різниці)

ШНМ – штучна нейронна мережа

ПР – Перцептрон Розенблатта

PPR – метод регресійного пошуку проекції

GAN – Generative Adversarial Network (генеративні змагальні мережі)

CNN – Convolutional Neural Network (згорткова нейронна мережа)

RoI – Return on Investment (регіони інтересу)

RPN – Regional Propositions Network (мережа регіональних пропозицій)

ПЗ – програмне забезпечення

CPU – Central Processing Unit

GPU – Graphics Processing Unit

ШІ – штучний інтелект

NLP – Natural Language Processing

iOS – мобільна операційна система, розроблена Apple Inc

MPSCNN – Metal Performance Shaders Convolutional Neural Network

API – Application Programming Interface

ВСТУП

Розпізнавання тексту, мови і музики, розпізнавання і обробка зображень, визначення об’єктів на фото і відео, розпізнавання образів і осіб, або навіть створення текстів, зображень і відео – все це, і багато іншого роблять можливим нейронні мережі.

Нейронні мережі – один видів машинного навчання, є системою з’єднаних і взаємодіючих між собою простих процесорів (штучних нейронів).

Сьогодні нейронні мережі використовують як альтернативу всім існуючим алгоритмам для розв’язання завдань розпізнавання образів, кількісного та якісного прогнозування, керування складними об’єктами і процесами. Розпізнавання образів має широке застосування і використовується при створенні усіх комп’ютерних систем, на які покладаються інтелектуальні функції, тобто функції, пов’язані із прийняттям рішень замість людини: медична діагностика, криміналістична експертиза, пошук інформації та інтелектуальний аналіз даних тощо.

У наші дні можна спостерігати справжній вибух розповсюдження ідей здорового способу життя та правильного харчування. Все більше людей хочуть мати здорове тіло, досягнувши бажаного результату завдяки контролю за своєю їжею. Саме остання потреба стала причиною напливу мобільних застосунків для стеження за тим, щоб раціон був збалансованим і здоровим. Але велика кількість систем, що використовуються для контролю за харчуванням, ґрунтуються на застарілому підході, коли всі дані потрібно власноруч шукати та заносити. Такий підхід вважається неефективним через затрати часу та клопітність внесення страв до сервісів підрахунку калорій і визначення складу. Це робить задачу розпізнавання та класифікації об’єктів, а в даному випадку саме фруктів, актуальною та гідною активних досліджень.

1 АНАЛІЗ ПРОБЛЕМИ РОЗПІЗНАВАННЯ ВІЗУАЛЬНИХ ОБ'ЄКТІВ

1.1 Проблематика виявлення фруктів на зображеннях

Зображення є важливим джерелом даних та інформації в нашому житті. Використання методів обробки зображень має видатні наслідки для аналізу даних. Класифікація фруктів є одним із основних програмних продуктів, який можна використовувати в супермаркетах для автоматичного визначення видів фруктів, куплених клієнтами. Навчання на місці є основною передумовою для такого типу роботи, що, як правило, спричинено тим, що користувачі мають низькі експертні знання або зовсім не мають їх. Розробка системи розпізнавання та класифікації фруктів є складним процесом через такі проблеми:

- колірні особливості;
- особливості форми;
- особливості текстури.

Колірні особливості постають найголовнішою проблемою в розробці систем розпізнавання об'єктів [1], в тому числі систем розпізнавання фруктів. Один і той самий вид фрукту може мати різних колір в залежності від його сорту або ступеня стиглості (рис. 1.1).

Рисунок 1.1 – Колірні особливості манго різного ступеня стиглості

І навпаки деякі різні види фруктів можуть мати однакові кольори і відрізнятись лише за розміром, що не можна визначити за зображенням (рис. 1.2).

Рисунок 1.2 – Порівняння розмірів апельсину та мандарину

Багато фруктів мають округлу форму, що ускладнює їх ідентифікацію для системи (рис. 1.3).

Рисунок 1.3 – Відміна особливостей форми яблука та кавуна

Також завдяки формі, подібній до інших об’єктів, є великий ризик помилки. Приклад такого зображення наведено на рисунку 1.4.

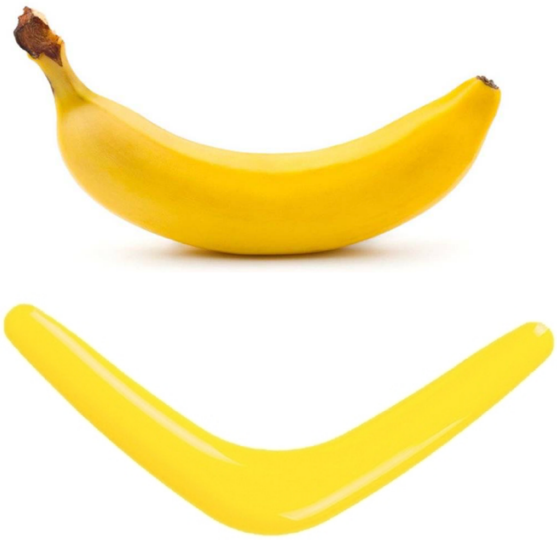

Рисунок 1.4 – Схожість форми банана та бумерангу

Також важливою проблемою є особливості текстури. Багато сучасних моделей опрацьовують зображення фруктів за їх ідеальним виглядом. Тому при ідентифікації можна легко стикнутися з проблемою, коли об’єкт не буде розпізнано через його зовнішні пошкодження, нецілісність або наявність предметів, які заважають його чіткому визначенню (рис. 1.5).

Рисунок 1.5 – Яблука з різними видами пошкоджень

1.2 Аналіз існуючих методів розпізнавання фруктів

Про класифікацію зображень для розпізнавання фруктів було проведено ряд досліджень. Veggie-Vision [2] була першою спробою розробити систему розпізнавання продуктів для використання в супермаркетах. Система могла аналізувати колір, текстуру та щільність і таким чином могла отримати більше інформації. Щільність розраховували шляхом ділення ваги на площу плоду. Повідомлена точність становила приблизно 95%, коли об'єднали особливості кольору та текстури. А. Роша, К. Хауагге, Дж. Вайнер і Д. Сіоме [3] розробили підхід, який може поєднувати багато ознак і класифікаторів. Автори підійшли до проблеми багатокласової класифікації як до проблеми бінарної класифікації таким чином, що можна зібрати разом різноманітні ознаки та підходи до класифікаторів, адаптовані до частин проблеми. Вони досягли точності класифікації до 99% для деяких фруктів, але вони об'єднали три ознаки, а саме: класифікацію межі та внутрішньої частини (BIC), вектор колірної когерентності (CCV) і функції unser, і визначили два найкращі варіанта для досягнення результата. Їхній метод дав погані результати для деяких видів фруктів, таких як яблуко Фуджі. С. Аріважаган, Р.Н. Шебія, С.С. Нідх'янандхан і Л. Ганесан [4] комбінували ознаки кольору та текстури для класифікації продукції. Вони використали класифікатор мінімальної відстані та досягли 86% точності свого набору даних із 15 різними видами продукції. Ф.А. Фаріа, Х.А. дос Сантуш, А. Роша та Р.С. Торрес [5] представили структуру для об'єднання класифікаторів для автоматичного розпізнавання продуктів у супермаркетах. Вони поєднали деякі класифікатори, навчені для конкретних класів, щоб підвищити рівень розпізнавання. М.Т. Чоудхурі, М.С. Алам, М.А. Хасан і М.І. Хан [6] розпізнали 10 різних овочів за допомогою кольорової гістограми та статистичних характеристик текстури. Вони отримали точність класифікації до 96,55%, використовуючи нейронну мережу як класифікатор. А. Данті, М. Мадгі та Б.С. Анамі [7] класифікували 10 видів листових овочів за допомогою

класифікатора BPNN з показником успішності 96,40%. Щоб сформувати вектор ознак, вони спочатку обрізали та змінили розмір зображення, а потім витягли середнє значення та діапазон відтінку та каналу насиченості зображення HSV. М. Суреша, К.С. Сандіп Кумар і Г. Шива Кумар [8] отримали 95% точності класифікації набору даних з восьми типів різних овочів, використовуючи параметри текстури в колірному просторі RGB. Вони використовували сегментацію вододілу, щоб виділити цікаву область як класифікатор попередньої обробки та дерева рішень для навчання та класифікації.

С.Р. Дубей [9], С.Р. Дубей і А.С. Джалал [10, 11] запропонували структуру для розпізнавання та класифікації зображень 15 різних типів продукції. Цей підхід передбачає сегментування зображення для виділення цікавої області, а потім обчислення характеристик із цієї сегментованої області, яка далі використовується під час навчання та класифікації багатокласовою опорною векторною машиною. Крім того, вони запропонували вдосконалену функцію текстури гістограми суми та різниці (ISADH) для такого роду задач. ISADH перевершив інші функції кольору та текстури зображення. А. Арефі, А.М. Мотлаг, К. Моллазаде та Р.Ф. Теймурлу [12] розробили алгоритм сегментації, щоб керувати рукою робота збиранням стиглих помідорів за допомогою техніки обробки зображень. Вони підготували систему машинного зору для отримання зображень з помідорів. Алгоритм працює у два етапи. На першому етапі фон віднімається з колірного простору RGB, а потім стиглий помідор виділяється за допомогою комбінації колірних просторів RGB, HSI та YIQ. На другому – стиглий помідор локалізовано за допомогою морфологічних ознак зображення. Вони досягли точності до 96,36% для 110 зображень помідорів.

Виявлення фруктів значною мірою впливає на ефективність збирання роботом, оскільки це неструктуроване середовище зі змінними умовами освітлення. Д.М. Буланон, Т.Ф. Беркс і В. Альханатіс [13] збільшили частину, яку займають фрукти на зображеннях, використовуючи коефіцієнт

кольоровості червоного кольору, і застосували метод визначення кола для класифікації окремих фруктів. Щоб покращити видимість плодів, вони отримали кілька видів під різними кутами огляду для частини крони дерева. Видимість фруктів покращилася з 50% до приблизно 90% завдяки отриманню кількох переглядів. А. Хайдар, Х. Донг і Н. Маврідіс [14] представили метод автоматичної класифікації даних на основі розпізнавання образів і комп'ютерного зору. Використовуючи вилучене зображення відповідним чином створеної бази 15 різних візуальних характеристик, вони спробували кілька методів класифікації. Ефективність методів коливалася від 89% до 99%. Р. Хіменес, А.К. Джайн, Р. Серес і Дж.Л. Понс [15] розробили метод, який може ідентифікувати сферичні плоди в природному середовищі, в якому присутні складні ситуації: оклюзії, тіні, яскраві області та плоди, що перекриваються. Дані про дальність і затухання сприймаються лазерним датчиком далекоміру, а після етапів розпізнавання отримують тривимірне положення фрукта з радіусом і коефіцієнтом відбиття.

Якість овочів часто вимірюють за кольором, формою, масою, твердістю, розміром і відсутністю пошкоджень, що також може служити основою для класифікації плодів. А.К.Л. Ліно, Дж. Санчес та І.М.Д. Фаббро [16] класифікували лимони та помідори на основі їх розміру та кольору. Ю. Лю, Б. Чень і Дж. Цяо [17] розробили систему для розпізнавання персика в природному середовищі. Система спочатку отримує зображення області червоного персика, а потім використовується відповідне розширення для розпізнавання всієї області. Потенційна центральна точка кола обчислюється перетином бісектриси перпендикуляра лінії на контурі. Центральна точка та радіус відповідного персикового кола отримують шляхом розрахунку статистичних параметрів потенційних центральних точок. Х.Н. Патель, Р.К. Джайн і М.В. Джоші [18] розробили метод виявлення фруктів з використанням покращеного алгоритму, який може обчислювати кілька ознак. Алгоритм може призначити різну вагу для різних характеристик, таких як колір, інтенсивність, край і орієнтація вхідного зображення. Приблизне

розташування фруктів на зображенні представлено вагами різних елементів. Алгоритм може виявити до 90% різних зображень фруктів, зроблених з різних місць на дереві.

Науковці почали досліджувати, як мобільні пристрої можна зручно використовувати для класифікації їжі. М.Х. Рахман, М.Р. Пікерінг, Д. Керр, К.Дж. Буші та Е.Дж. Делп [19] представили концепцію інтеграції камери мобільних телефонів для захоплення зображень споживаної їжі. Ці зображення можна обробляти автоматично для ідентифікації харчових продуктів, присутніх на зображенні. Вони створили особливості текстури із зображень їжі та продемонстрували, що ця функція забезпечує більшу точність системи оцінки дієти на основі мобільного застосунку.

1.3 Види нейронних мереж для роботи з зображеннями.

Нейронна мережа (також відома як штучна нейронна мережа) – це адаптивна система, яка навчається за допомогою взаємопов’язаних вузлів або нейронів у багатошаровій структурі, подібній до людського мозку [20, 21]. Нейронна мережа може навчатися на даних, тому її можна навчити розпізнавати шаблони, класифікувати дані та передбачати майбутні події.

Нейронні мережі розкладають вхідні дані на абстрактному рівні. Її можна навчити розпізнавати шаблони в мові чи зображеннях, наприклад, так само, як і людський мозок. Її поведінка залежить від способу з’єднання окремих елементів і міцності або ваги цих з’єднань. Ці ваги автоматично регулюються під час навчання відповідно до заданих правил навчання, поки ШНМ правильно не виконає бажане завдання.

Нейронні мережі використовуються для таких цілей [22]:

- сегментація зображень та відео;
- виявлення об’єктів на зображеннях;
- розпізнавання символів та голосу;

– класифікація за заданими параметрами;

– тренування роботизованих систем;

– прогнозування та аналіз даних (зокрема фінансові прогнози щодо цін на акції, валюти, опціонів, ф’ючерсів, рейтингів банкрутства та облігацій).

В даній роботі розглянемо детальніше нейронні мережі, які підходять для виявлення образів на зображеннях.

1.3.1 Перцептрон Розенблатта

Перцептрон Розенблатта – це лінійний алгоритм машинного навчання, який використовується для навчання під наглядом для різних двійкових класифікаторів [23]. Цей алгоритм дозволяє нейронам вивчати елементи та обробляти їх один за іншим під час підготовки.

Персептрон намагається розділити вхідні дані через лінійну межу рішення. Подібна процедура виконується іншим алгоритмом навчання під наглядом, відомим як класифікатори опорних векторів.

Персептрон працює, приймаючи набір скалярних вхідних характеристик разом із постійним терміном «зміщення» та призначаючи ваги цим входам. Береться лінійна комбінація ваг і вхідних даних. Ця лінійна комбінація потім подається через функцію активації, яка визначає, до якого стану (або класу) належить набір вхідних даних [23]. На рисунку 1.6 представлено схему роботи Перцептрона Розенблатта.

Для аналізу роботи Перцептона можна використовувати метод регресійного пошуку проекції (PPR). Двійковий вихід ПР можна розглядати як змінну відповіді в аналізі PPR. Вхідні змінні використовуються як предиктори, а PPR використовують для ідентифікації проекцій вхідних змінних, які є найбільш релевантними для прогнозування результату Перцептона.

Таким чином, PPR може допомогти визначити, які вхідні змінні є найбільш важливими для завдання класифікації. Він також може виявити

будь-які складні, нелінійні зв’язки між вхідними змінними та виходом, які можуть бути неочевидними лише за результатами двійкової класифікації Перцептона.

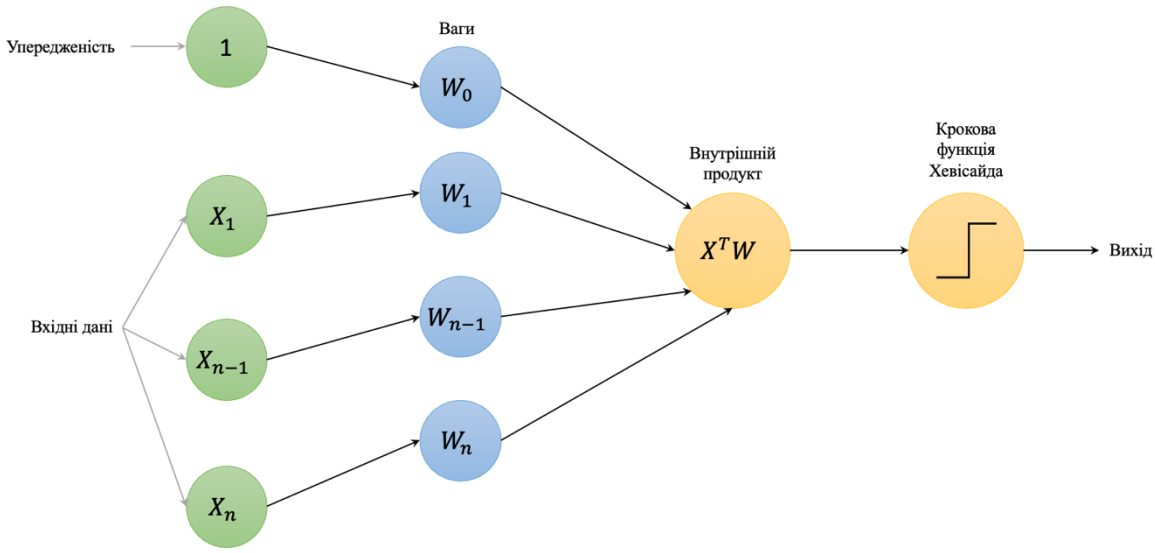

Рисунок 1.6 – Схема роботи Перцептрона Розенблатта

1.3.2 Генеративна нейронна мережа

Генеративні змагальні мережі (GAN) – це потужний клас нейронних мереж, які використовуються для неконтрольованого навчання. GAN в основному складаються із системи двох конкуруючих моделей нейронних мереж, які конкурують одна з одною та здатні аналізувати, фіксувати та копіювати варіації в наборі даних [24].

У GAN є «Генератор» і «Дискримінатор». «Генератор» створює підроблені зразки даних (зображення, аудіо тощо) і намагається обманути «Дискримінатор». «Дискримінатор», з іншого боку, намагається відрізнити справжні зразки від підроблених. «Генератор» і «Дискримінатор» є нейронними мережами, і обидва вони конкурують один з одним на етапі навчання. Ці кроки повторюються кілька разів, і після кожного повторення

«Генератор» і «Дискримінатор» стають все кращими і кращими [25]. Роботу можна наочно представити схемою, наведеною на рисунку 1.7.

Рисунок 1.7 – Схема роботи GAN

Тут генеративна модель фіксує розподіл даних і навчена таким чином, щоб намагатися максимізувати ймовірність помилки «Дискримінатора». «Дискримінатор», з іншого боку, заснований на моделі, яка оцінює ймовірність того, що отриманий зразок отримано з навчальних даних, а не з генератора. GAN сформульовано як мінімаксну гру, де «Дискримінатор» намагається мінімізувати свою винагороду $V(D, G)$, а «Генератор» намагається мінімізувати винагороду «Дискримінатора». Це можна описати формулою:

$$\min_G \max_D V(D, G), \tag{1.1}$$

$$V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_z(z)}[\log(1 - D(G(z)))], \tag{1.2}$$

де G – «Генератор»;

D – «Дискримінатор»;

$Pdata(x)$ – розподіл реальних даних;

$P(z)$ – розподіл «Генератора»;

x – зразок з $Pdata(x)$;

z – зразок з $P(z)$;

$D(x)$ – мережа «Дискримінатора»;

$G(z)$ – мережа «Генератора».

«Дискримінатор» навчається, поки «Генератор» не працює. На цьому етапі мережа передається лише в прямому напрямку, а зворотне не відбувається. «Дискримінатор» навчається на реальних даних за n епох і перевіряє, чи зможе він правильно передбачити їх як реальні. Крім того, на цьому етапі «Дискримінатор» також навчається на підроблених даних, згенерованих «Генератором», і перевіряє, чи зможе він правильно передбачити їх як підроблені.

1.3.3 Згорткова нейронна мережа

Згорткова нейронна мережа (CNN) – це нейронна мережа глибокого навчання, розроблена для обробки структурованих масивів даних, таких як зображення. Згорткові нейронні мережі широко використовуються в комп'ютерному зорі та стали найсучаснішими для багатьох візуальних застосувань, таких як класифікація зображень, а також досягли успіху в обробці природної мови для класифікації тексту.

CNN дуже добре вловлюють шаблони у вхідному зображенні, такі як лінії, градієнти, кола або навіть очі та обличчя. Саме ця властивість робить згорткові нейронні мережі такими потужними для комп'ютерного зору. На відміну від інших алгоритмів комп'ютерного зору, згорткові нейронні мережі можуть працювати безпосередньо з необробленим зображенням і не потребують попередньої обробки.

Архітектура CNN – це багатошарова нейронна мережа прямого зв'язку, створена шляхом послідовного накладання багатьох прихованих шарів один на одного. Саме цей послідовний дизайн дозволяє згортковим нейронним

мережам вивчати ієрархічні особливості [26, 27]. Приклад накладання фільтрів на шари згортки наведено на рисунку 1.8.

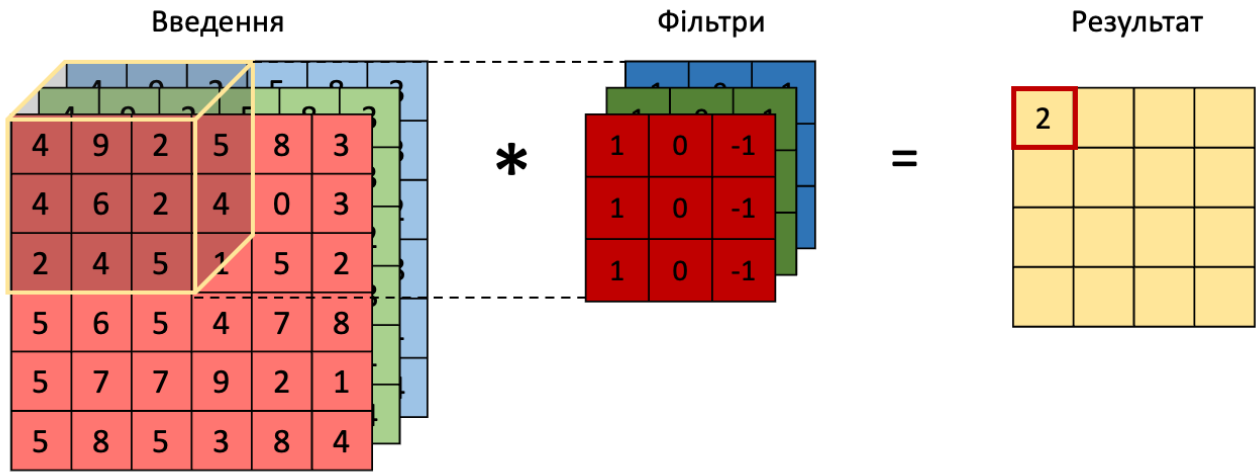

Рисунок 1.8 – Накладання фільтрів на шари згортки

Приховані шари зазвичай являють собою згорткові шари, за якими йдуть шари активації, за деякими з них йдуть шари об’єднання.

На рисунку 1.9 зображено схему роботи архітектури CNN.

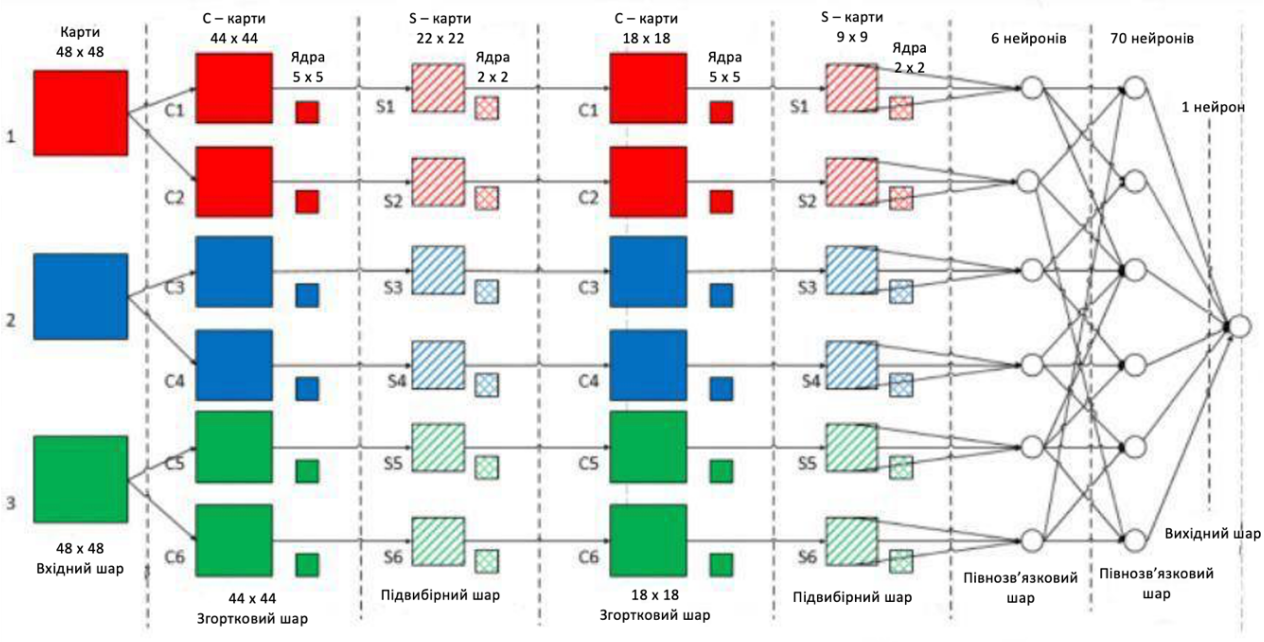

Рисунок 1.9 – Схема роботи CNN

Існує багато типів CNN, таких як [28]:

– R-CNN була однією з перших моделей виявлення об’єктів на основі глибокого навчання, і вона складається з багатоетапного конвеєра, який спочатку пропонує регіони інтересу (RoI) за допомогою вибіркового пошуку, а потім класифікує ці RoI за допомогою CNN;

– Fast R-CNN був удосконаленням у порівнянні з R-CNN, який використовував один CNN для класифікації RoI замість запуску CNN окремо для кожного RoI. Він також представив рівень об’єднання інтересів (RoI Pooling) для вилучення вектора ознак фіксованого розміру з RoI довільного розміру;

– Faster R-CNN удосконалено Fast R-CNN, замінивши вибірковий пошук мережею регіональних пропозицій (RPN), яка генерує регіональні пропозиції безпосередньо з карт об’єктів. Це усуває потребу в окремому етапі пропозиції регіону та робить модель швидшою та точнішою;

– HOG – це традиційний метод виділення ознак на основі комп’ютерного зору, який виявляє об’єкти шляхом обчислення гістограм орієнтацій градієнтів у фрагментах зображення. Це був один із найперших методів виявлення об’єктів до того, як глибоке навчання стало популярним;

– R-FCN є вдосконаленням у порівнянні з Faster R-CNN, яке замінює рівень RoI Pooling на шар об’єднання RoI, чутливий до позиції (PSRoI), який враховує просторове розташування RoI на карті функцій. Це підвищує точність і скорочує час обчислень;

– SSD – це одноетапна модель виявлення об’єктів, яка генерує пропозиції об’єктів і прогнози класів безпосередньо з карт функцій без необхідності окремого етапу пропозиції регіону. Він забезпечує виявлення об’єктів у реальному часі та є швидшим, ніж двоступеневі моделі, такі як Faster R-CNN;

– SPP-net – це архітектура CNN, яка додає шар просторової піраміди поверх карти згорткових функцій. Це дозволяє мережі обробляти вхідні дані

різних розмірів і пропорцій, а також використовується для виявлення об’єктів і класифікації зображень;

– RetinaNet – це одноетапна модель виявлення об’єктів, яка використовує функцію фокусних втрат для вирішення проблеми дисбалансу класів, яка виникає в задачах виявлення об’єктів, де існує багато фонових зразків порівняно із зразками об’єктів. Це покращує здатність моделі виявляти об’єкти в різних масштабах і підвищує точність;

– YOLO – це одноетапна модель виявлення об’єктів, яка передбачає обмежувальні прямокутники та ймовірності класів безпосередньо з усього зображення за один прохід мережі [29]. Він швидкий і ефективний, але може пожертвувати точністю порівняно з двоступеневими моделями, такими як Faster R-CNN;

– MPSCNN – це модель виявлення об’єктів на основі глибокого навчання, яка використовує кілька згорткових шарів із різними розмірами ядра для захоплення характеристик об’єктів у різних масштабах. Він використовує механізм багатопрогнозування для поєднання прогнозів із різних масштабів для підвищення точності [30].

1.4 Засоби для роботи з нейронними мережами

Програмне забезпечення (ПЗ) для штучних нейронних мереж використовується для моделювання, дослідження, розробки та застосування штучних нейронних мереж та концепцій програмного забезпечення.

Інструменти для штучної нейронної мережі використовують концепції, адаптовані з біологічних нейронних мереж, штучного інтелекту та машинного навчання для моделювання, вивчення та вдосконалення штучної нейронної мережі.

ПЗ штучної нейронної мережі використовується для представлення, вивчення, створення та застосування принципів програмного забезпечення, адаптованих із біологічних нейронних мереж до штучних нейронних мереж.

ПЗ для штучних нейронних мереж призначене для реалістичних застосувань штучних нейронних мереж, у першу чергу з аналізом даних і прогнозуванням. Зазвичай ці симулятори аналізу даних мають певні можливості попередньої обробки та використовують відносно просту, налаштовану статичну нейронну мережу.

Бібліотеки для роботи з нейронними мережами в першу чергу поділяються за мовами написання. Так на мові Python відомі такі бібліотеки, як:

- TensorFlow;
- Pytorch;
- Scikit.

Бібліотека TensorFlow вважається однією з найкращих бібліотек Python для програм глибокого навчання. Вона має гнучку архітектуру та структуру, що дозволяє працювати як CPU так з GPU. Часто використовується для реалізації навчання з підкріпленням у моделях глибокого навчання, і ви можете безпосередньо візуалізувати моделі машинного навчання [31].

Pytorch – бібліотека з відкритим вихідним кодом, створена дослідницькою групою ШІ Facebook у 2016 році. Pytorch дозволяє виконувати багато завдань і особливо корисна для програм глибокого навчання, таких як NLP і комп’ютерне бачення. Це також гнучка бібліотека, яка здатна працювати на спрощених процесорах або центральних і графічних процесорах [32].

Бібліотека Scikit – реалізує багаторівневі персептрони, автоматичні кодери та рекурентні нейронні мережі зі стабільним інтерфейсом, готовим до майбутнього, як обгортку для потужних існуючих бібліотек, з планами для блоків, які сумісні з Scikit-learn для більшої зручності користувачів. Імпортувавши пакет sknn, що надається цією бібліотекою, можна легко навчити глибокі нейронні мережі регресорами і класифікаторами [33].

Найбільш популярними бібліотеками на мові C/C++ є:

– Microsoft Cognitive Toolkit (CNTK);

– Fast Artificial Neural Network (FANN);

– Пакет лінійної алгебри Armadillo.

Microsoft Cognitive Toolkit (CNTK) – це уніфікований інструментарій глибокого навчання, який описує нейронні мережі як серію обчислювальних кроків за допомогою орієнтованого графа. Він реалізує навчання зі стохастичним градієнтним спуском (SGD) із автоматичним розрізненням і розпаралелюванням на кількох графічних процесорах і серверах. CNTK дозволяє користувачам легко реалізовувати та комбінувати такі популярні типи моделей, як DNN, CNN і RNN [34].

Fast Artificial Neural Network (FANN) – це бібліотека нейронної мережі з відкритим кодом, написана мовою C. Бібліотека реалізує багаторівневі штучні нейронні мережі на C з підтримкою як повнозв’язаних, так і розріджених мереж. Вона проста у використанні, універсальна, добре задокументована і швидка. Функції включають навчання зворотному поширенню, навчання розвитку топології, кросплатформенність і можливість використання як чисел з плаваючою, так і з фіксованою комою [35].

Пакет лінійної алгебри Armadillo має функції, подібні до Matlab. Бібліотека відома своєю здатністю швидко перекладати дослідницький код для різних галузей, включаючи розпізнавання образів, комп’ютерне бачення, обробку сигналів, біоінформатику, статистику та економетрику [36].

Прикладом бібліотеки на інших мовах програмування може бути Core ML – це платформа Apple для інтеграції моделей машинного навчання у програму. Core ML забезпечує уніфіковане представлення для всіх моделей. Програма використовує API Core ML і дані користувача, щоб робити прогнози та точно налаштовувати моделі на пристрої. Запуск моделі виключно на пристрої користувача усуває будь-яку потребу в мережевому з’єднанні, що допомагає зберегти конфіденційність даних користувача та оперативність програми [37].

1.5 Постановка задачі

Таким чином, розпізнавання об’єктів на зображеннях, а саме фруктів та їх вітамінного складу, з використанням нейромережевих засобів є актуальним завданням у різноманітних галузях, включаючи маркетинг, сільське господарство та наукові дослідження.

Об’єктом роботи є зображення фруктів, які необхідно класифікувати для визначення їх вітамінного складу.

Метою роботи є розробка iOS застосунку із застосуванням згорткової нейронної мережі, спрямованого на розпізнання фруктів та їх вітамінного складу. Вхідні дані включають в себе зображення з різними класами об’єктів – різноманітні класи фруктів.

Для досягнення мети необхідно вирішити такі завдання:

– обрати нейронну мережу для класифікації фруктів;

– розробити методику розпізнавання фруктів;

– провести вибір інструментальних засобів для реалізації методу розпізнавання фруктів;

– навести детальний опис етапів програмної реалізації методу розпізнавання фруктів;

– провести тестування розробленого застосунку та проаналізувати результати;

– виявити перспективи подальшої роботи;

– зробити висновки щодо виконаної роботи.

2 ОСОБЛИВОСТІ РОЗПІЗНАВАННЯ ВІЗУАЛЬНИХ ОБ'ЄКТІВ В IOS ЗАСТОСУНКАХ

2.1 Принципи побудови мобільних застосунків з використанням нейронних мереж

Сьогодні мобільні пристрої використовуються повсюдно – більше трьох мільярдів людей у всьому світі активно встановлюють і використовують програми, згідно з інформацією провідного світового статистичного порталу Statista [38]. В даний час впровадження нейронних мереж вже дозволяє розширити функціональність застосунків і підвищити якість їх роботи.

Створення мобільних додатків за допомогою нейронних мереж включає в себе кілька принципів, які є критично важливими для успіху. Ці принципи включають:

- ефективний дизайн моделі;
- попередня обробка даних;
- навчання на пристрої;
- оптимізація батареї;
- конфіденційність і безпека.

Мобільні пристрої мають обмежену обчислювальну потужність і пам'ять, тому важливо розробляти моделі нейронних мереж, які є ефективними та можуть безперебійно працювати на мобільних пристроях. Цього можна досягти за допомогою таких методів, як стиснення моделі, скорочення та квантування.

Мобільні пристрої часто мають обмежену пропускну здатність і ємність пам'яті, тому важливо попередньо обробити дані на пристрої, щоб зменшити обсяг даних, які потрібно передати та зберегти. Це може включати такі методи, як стиснення даних, вилучення функцій і збільшення даних.

Мобільні пристрої не завжди можуть мати доступ до надійного підключення до Інтернету, тому важливо створювати моделі нейронних мереж, які можуть навчатися та адаптуватися на пристрої. Це може включати такі методи, як навчання з перенесенням і навчання з підкріпленням.

Обчислення нейронної мережі можуть потребувати ресурсів, тому важливо оптимізувати моделі нейронної мережі для споживання батареї. Це може включати такі методи, як оптимізація архітектури моделі, пакетна нормалізація та рання зупинка.

Мобільні пристрої часто містять конфіденційні дані, тому важливо створювати моделі нейронних мереж, які захищають конфіденційність користувачів і підтримують безпеку даних. Це може включати такі методи, як інтегроване навчання, диференціальна конфіденційність і шифрування моделі.

Дотримуючись цих принципів, розробники можуть створювати моделі нейронних мереж, які є ефективними, масштабованими та безпечними та можуть безперебійно працювати на мобільних пристроях. Ці моделі можна використовувати для широкого спектру програм, включаючи розпізнавання зображень і мови, обробку природної мови та прогнозну аналітику.

2.2 Методологія розпізнавання фруктів на зображенні

Робота з зображенням, а особливо розпізнавання та класифікація об’єкту на ньому – досить складна задача для комп’ютеру. Якщо для людини виділити та миттєво розпізнати, що саме зображено для фотографії – це справа, яка не потребує великих зусиль, то для комп’ютера потрібен цілий ряд складних методів, кожен з яких може дати різну ефективність на різних даних.

Класифікація фруктів із зображень за допомогою згорткових нейронних мереж (CNN) може складатися з таких етапів:

Крок 1. Збір даних. Потрібно зібрати великий набір даних зображень фруктів із відповідними мітками класів. Набір даних має бути різноманітним і охоплювати діапазон типів, розмірів і кольорів фруктів.

Крок 2. Попередня обробка даних. Потрібно попередньо обробити зображення, змінивши їх розмір до фіксованого розміру, нормалізувавши значення пікселів і розділивши набір даних на набори для навчання, перевірки та тестування.

Крок 3. Архітектура моделі. Потрібно обрати архітектуру CNN для класифікації фруктів. Архітектура має складатися з кількох згорткових шарів з операціями об'єднання, активації та нормалізації, за якими слідують один або більше пов'язаних шарів із регуляризацією вилучення. Рівень виводу повинен мати функцію активації softmax, яка створює ймовірності класу для кожного типу фруктів.

Крок 4. Навчання моделі. Потрібно навчити модель CNN на попередньо обробленому навчальному наборі даних, оптимізувавши параметри моделі, щоб мінімізувати перехресну втрату ентропії між прогнозованими та фактичними мітками [39, 40]. Процес навчання передбачає подачу в модель пакетів зображень і відповідних міток і коригування ваг моделі на основі різниці між прогнозованими та фактичними мітками за допомогою зворотного поширення та градієнтного спуску.

Крок 5. Оцінка моделі. Потрібно оцінити продуктивність навченої моделі на затриманих перевірочних і тестових наборах даних.

Крок 6. Розгортання моделі. Коли навчена модель досягне задовільної продуктивності, розгорніть її для класифікації нових зображень фруктів. Це передбачає подачу вхідного зображення через навчену модель і використання ймовірностей вихідного класу моделі для визначення найімовірнішого типу плоду.

Розглянемо, як працює цей метод з математичної точки зору.

Нехай X – набір вхідних зображень із пов'язаними мітками класів Y. Мета – вивчити функцію $f(X) \rightarrow Y$, яка може точно класифікувати фрукти на основі їхніх візуальних особливостей.

Припускаючи модель CNN, функція може бути представлена як:

$$Y = f(X) = Softmax(W * X + b), \quad (2.1)$$

де X – 4-вимірний тензор, що представляє вхідне зображення з розмірами висоти, ширини та глибини;

W – набір вагових матриць;

b – набір векторів зміщення;

$Softmax$ – функція активації, яка перетворює вихідну логіку в клас ймовірності.

Ваги та зміщення моделі вивчаються під час процесу навчання за допомогою зворотного поширення та градієнтного спуску.

Модель можна навчити, мінімізуючи перехресну втрату ентропії між прогнозованою та фактичною мітками:

$$\text{Втрата} = \frac{-1}{m} * \sum Y * \log(f(X)) + (1 - Y) * \log(1 - f(X)), \quad (2.2)$$

де m – кількість прикладів навчання;

Y – однократно закодований вектор, що представляє фактичну мітку класу;

$f(X)$ – прогнозований вектор ймовірності класу.

Функція втрат вимірює різницю між прогнозованими й фактичними ймовірностями класу та керує процесом оптимізації для оновлення параметрів моделі в напрямку мінімізації втрат.

Коли модель навчена, її можна використовувати для класифікації нових зображень фруктів, пропускаючи їх через навчену модель і використовуючи ймовірності вихідного класу для визначення найімовірнішого типу фруктів.

2.3 Архітектура MPSCNN

Metal Performance Shaders Convolutional Neural Network (MPSCNN) – це архітектура, розроблена Apple для прискорення обчислень згорткової нейронної мережі на пристроях iOS. Вона дозволяє напряму працювати нейронній мережі з фреймворком Metal, який забезпечує обробку зображень в програмних продуктах під операційну систему iOS.

Використання MPSCNN може значно покращити продуктивність згорткових нейронних мереж на пристроях iOS, що призведе до швидших і ефективніших обчислень. Це особливо корисно для додатків, які вимагають обробки в реальному часі, наприклад доповненої реальності та розпізнавання зображень [30].

Мережа складається з трьох рівнів:

- злитий згортково-пакетний шар нормалізації;
- шар об’єднання;
- повністю зв’язаний шар.

Після того, як код завершить прохід вперед, він обчислює свої втрати, які є оцінкою, яка вказує, як прогнозовані значення відхиляються від міток. Код використовує градієнти, які генерує втрата, як основу для зворотного проходу, де він застосовує три шари у зворотному порядку.

Зворотні проходи генерують значення градієнта, які оптимізатор використовує для оновлення таких параметрів:

- ваги згортки, які ініціалізуються випадковими значеннями, і зміщення;
- нормалізація бета (зсув) і гамма (масштаб);
- повністю зв’язані ваги.

Поступові зміни оптимізатора вагових коефіцієнтів, зміщення, бета-версії та гами підвищують ефективність мережі у розпізнаванні пікселів із кожною ітерацією.

На наступному зображенні (рис. 2.1) показано зв’язки між шарами.

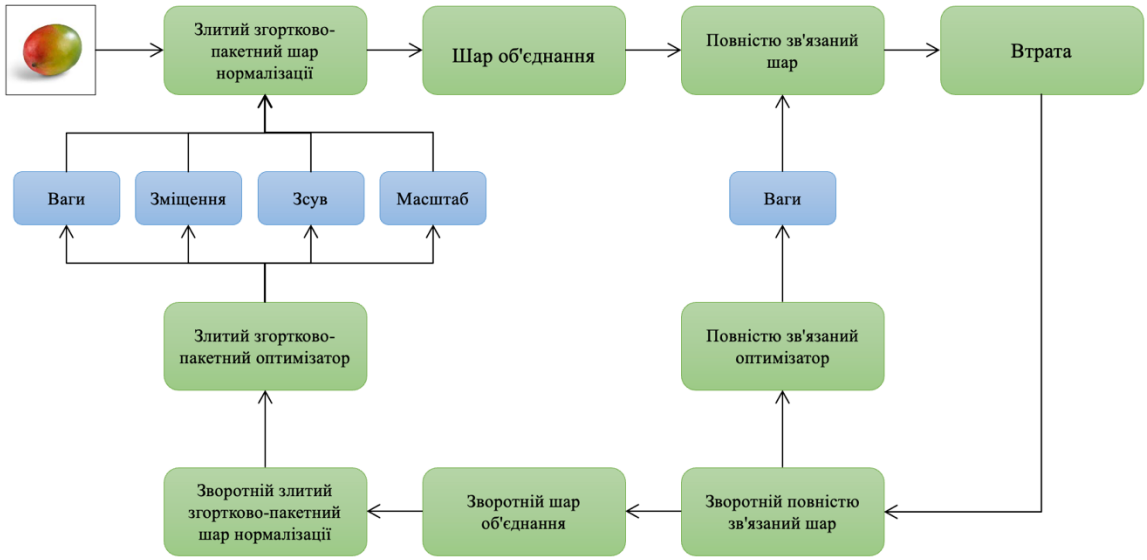

Рисунок 2.1 – Схема роботи архітектури MPSCNN

Згортка передбачає взяття вхідного тензора, згортання його за допомогою набору фільтрів і створення вихідного тензора. Математично цю операцію можна представити так:

$$y[i,j,k] = sum(f[m,n,k] * x[i+m, j+n, k] + b[k]), \quad (2.3)$$

де y – вихідний тензор;

f – тензор фільтра;

x – вхідний тензор;

b – тензор зміщення;

i, j, k, m і n – індекси.

MPSCNN надає набір попередньо створених шарів згортки, оптимізованих для виконання на GPU. Ці шари реалізують операцію згортання з використанням набору методів паралельної обробки, включаючи мозаїку, групування та паралельне скорочення.

Можна представити операцію паралелізованої згортки в MPSCNN наступним чином:

$$y[i,j,k] = sum(w[k][l] * x[i+m, l, j+n] + b[k]), \quad (2.4)$$

де w – набір ваг для k-ої вихідної карти ознак;

l – індекс вхідної карти ознак;

m і n – індекси вікна згортки.

Щоб виявити об'єкт MPSCNN бере класифікатор об'єкта і оцінює його в різних місцях зображення (рис. 2.2).

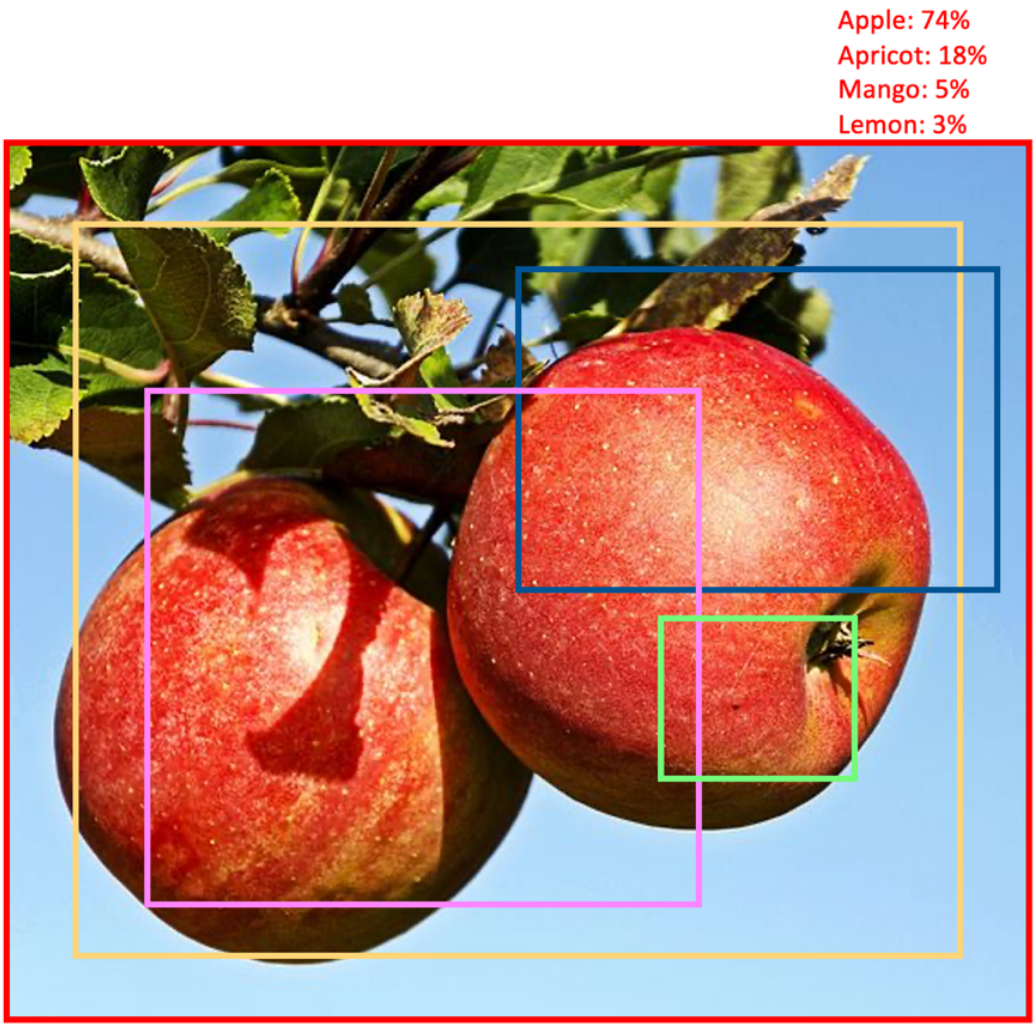

Рисунок 2.2 – Накладення рамок класифакатора MPSCNN на зображення

Далі вона відкидає усі нульові значення та видає результат зі значенням класифікації більше нулю. Як фінальний результат можна обрати максимальну вірогідність класу або усі існуючи варіанти класифікації.

2.4 Використання бібліотеки CoreML для навчання

CoreML – це платформа машинного навчання, розроблена Apple для інтеграції моделей машинного навчання в програми iOS, macOS і watchOS. Він надає набір інструментів і API, які дозволяють розробникам легко створювати та розгортати моделі машинного навчання на пристроях Apple [37].

Основні переваги використанняпе CoreML для програм машинного навчання:

– висока продуктивність: CoreML оптимізовано для високопродуктивного результату на пристроях Apple, що дозволяє плавно та швидко запускати моделі машинного навчання на iOS, macOS і watchOS;

– навчання на пристрої: CoreML підтримує навчання на пристрої, дозволяючи моделям машинного навчання адаптуватися та вдосконалюватися з часом без необхідності навчання на стороні сервера;

– проста інтеграція: CoreML легко інтегрується з Xcode, середовищем розробки Apple, і надає набір API, які спрощують інтеграцію моделей машинного навчання в програми iOS, macOS і watchOS;

– безпека та конфіденційність: CoreML надає набір функцій безпеки та конфіденційності, таких як шифрування моделі та диференціальна конфіденційність, які дозволяють розробникам створювати програми машинного навчання, які захищають дані користувача та підтримують безпеку.

Ця бібліотека працює за принципом if-then-else (рис. 2.3).

Рисунок 2.3 – Алгоритм роботи бібліотеки CoreML

Нейронна мережа може аналізувати піксельні дані та самостійно виявляти корисні функції для отримання правильних відповідей. Вона вивчає це під час навчального процесу з навчальних зображень і міток, які надав користувач. Потім вона використовує ці функції, щоб зробити остаточні прогнози.

Типові ознаки, які використовуються класифікаторами зображень, включають краї, кольорові плями, абстрактні форми та зв'язки між ними [41 – 47]. На практиці не має значення, які функції обрала модель, якщо вони дозволяють моделі робити хороші прогнози.

Однією з причин, чому глибоке навчання настільки популярно, є те, що навчання моделі самому знаходити цікаві характеристики зображення працює набагато краще, ніж будь-які правила «if-then-else», які люди придумали вручну в минулому. Незважаючи на це, глибоке навчання все одно отримує переваги від будь-яких підказок, які користувач може надати щодо структури навчальних даних.

Процес навчання з вчителем виглядає так:

Крок 1. Потрібно ініціалізувати нейронну мережу з невеликими випадковими числами. Ці числа відомі як ваги або вивчені параметри моделі. Навчання – це процес зміни цих ваг із випадкових чисел на щось значуще.

Крок 2. Нейронна мережа робить прогнози для всіх навчальних прикладів. Для класифікатора зображень навчальними прикладами є зображення.

Крок 3. Порівняння передбачення з очікуваними відповідями. Коли навчається класифікатор, очікувані відповіді, які зазвичай називають «цілями», є мітками класів для навчальних зображень. Цілі використовуються для обчислення «помилки» або втрати, міри того, наскільки далекі прогнози від очікуваних відповідей. Втрата є багатовимірною функцією, яка має мінімальне значення для певної конфігурації ваг, і метою навчання є визначення найкращих можливих значень для ваг, які дають втрату дуже

близько до цього мінімуму. На нетренованій моделі, де ваги є випадковими, втрата висока. Чим менше це значення втрат, тим більше навчилася модель.

Крок 4. Щоб покращити вагові коефіцієнти та зменшити похибку, потрібно обчислити градієнт функції втрат. Цей градієнт показує, наскільки кожна вага вплинула на втрату.

$$\nabla L(W) = \frac{\partial L}{\partial y} * \frac{\partial y}{\partial W}, \tag{2.5}$$

де L – функція втрат для моделі Core ML;

W – вектор ваг моделі;

y є результатом моделі;

$\partial L/\partial y$ та $\partial y/\partial W$ представляють часткові похідні функції втрат відносно результату моделі та ваг відповідно.

Використовуючи цей градієнт, можна виправити ваги, щоб наступного разу втрати були трохи меншими. Цей крок корекції називається градієнтним спуском і повторюється багато разів під час тренування. Це трохи підштовхує знання моделі в правильному напрямку, щоб наступного разу вона зробила дещо правильніші прогнози для навчальних зображень. Для великих наборів даних використання всіх навчальних даних для обчислення градієнта займає багато часу.

Крок 5. Потрібно повернутись до Кроку 2, щоб повторити цей процес сотні разів для всіх зображень у навчальному наборі. З кожним кроком навчання модель стає трохи кращою: засвоєні параметри змінюються з випадкових чисел на більш підходящі значення. З часом модель вчиться робити все кращі прогнози.

Схема процесу навчання з вчителем представлена на рисунку 2.4.

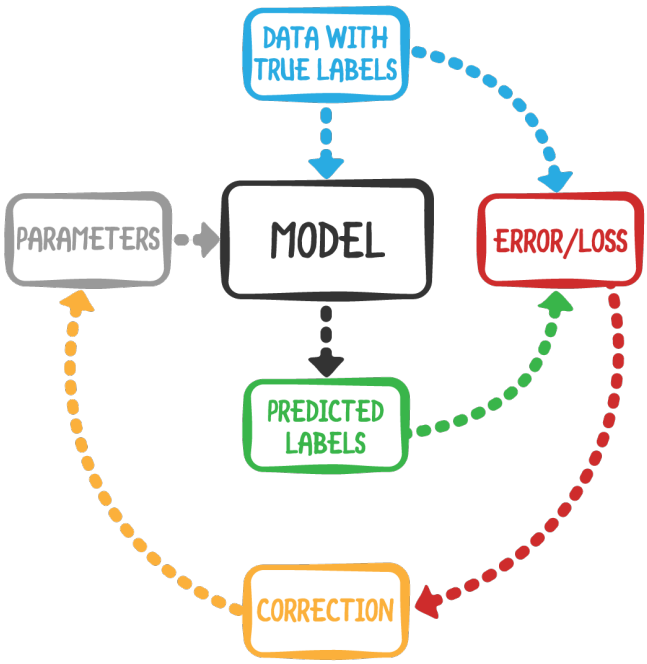

Рисунок 2.4 – Схема процесу навчання з вчителем

2.5 Збереження даних у застосунку за допомогою CoreData

Core Data – це структура, надана Apple для керування збереженням даних у програмах macOS, iOS і watchOS. Це високорівневий об'єктно-орієнтований інтерфейс до основної бази даних SQLite і надає ряд потужних функцій для керування складними об'єктними графами та зв'язками.

Core Data дозволяє розробникам визначати схему для своїх даних за допомогою редактора графічної моделі даних, який генерує необхідний код для створення об'єктів у базі даних і керування ними [48]. Фреймворк надає низку інструментів для отримання, сортування, фільтрації та кешування даних, а також керування зв'язками між об'єктами в базі даних.

Core Data надає колекцію попередньо визначених модулів, які ми можемо використовувати для створення, читання, оновлення та видалення об'єктів на рівні збереження.

Взаємозв'язок класів Core Data схематично зображено на рисунку 2.5.

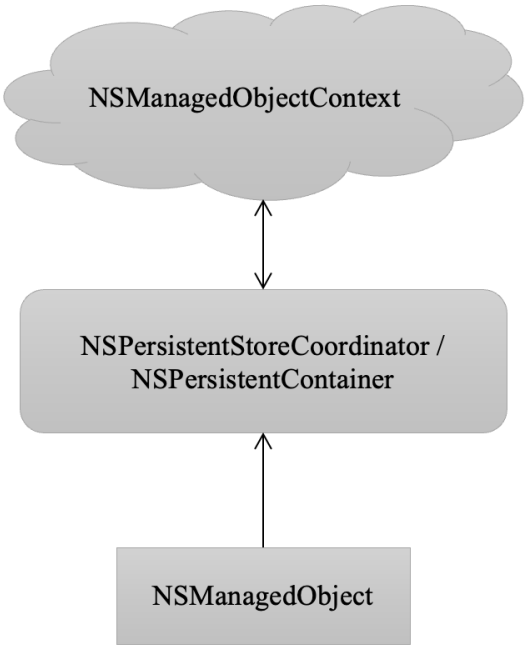

Рисунок 2.5 – Взаємозв’язок класів Core Data

Всі класи Core Data є підкласами класу NSManagedObject, що дозволяє їм працювати з Core Data. Це стандартні класи Swift із кількома додатковими анотаціями. @NSManaged – це спеціальна інструкція, яка дозволяє Core Data працювати з цими властивостями особливим чином. Це не обгортка властивостей, хоча виглядає так само. Завдяки атрибуту Core Data може відстежувати зміни властивостей об’єкта і, отже, знати, що потрібно зберегти.

Для створення сховища в програмі використовуються класи NSPersistentStoreCoordinator або NSPersistentContainer.

NSPersistentStoreCoordinator діє як міст між основним стеком даних і сховищем (базою даних). Коли контекст об’єкта робить запит на отримання всіх елементів у базі даних, постійний координатор завантажує їх зі сховища SQLite і передає назад до контексту об’єкта.

NSPersistentContainer – це клас абстракції вищого рівня, який інкапсулює найважливіші компоненти основного стеку даних. Створення основного стека даних за допомогою Persistent Container спрощує взаємодію з основними даними. Він спрощує створення та керування стеком основних даних,

обробляючи створення керованої об’єктної моделі, постійного координатора сховища та контексту керованого об’єкта.

NSManagedObjectContext дозволяє нам отримувати дані з бази даних, а також зберігати їх. Програма зазвичай має один основний контекст, якого може вистачити у багатьох випадках. Цей контекст має пов’язані сутності, тому він може відстежувати зміни та зберігати їх за потреби.

Це також причина, через яку потрібно передати контекст в ініціалізацію NSManagedObject. Таким чином пов’язується об’єкт із контекстом, який ним управлятиме.

Також потрібно повернутися до NSPersistentContainer. Ми можемо викликати для нього функцію BackgroundTask, що дасть замикання з фоновим NSManagedObjectContext як вхідний параметр. Таким чином, ми можемо легко виконати фонову роботу з базою даних, не впливаючи на видиму продуктивність. Можна використовувати вибірку у цьому фоновому контексті, змінювати сутності та зберігаючи їх.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи був розроблений iOS застосунок для розпізнання вітамінного складу фруктів за їх зображенням. Для реалізації було обране інтегроване середовище розробки для мови програмування Swift XCode. Це обумовлено тим, що iOS володіє рядом таких переваг як:

– програми для iPhone мають більший ROI, ніж програми для Android. Факт, який значною мірою збільшує переваги розробки застосунків для iOS;

– iOS застосунки захищають мікропрограму та програмне забезпечення за допомогою суворих заходів безпеки;

– розробка програм для iPhone залишається незавершеною, доки програми не будуть створені відповідно до стандартів високої якості Apple Store, перш ніж вони будуть доступні на ринку. Коли користувач завантажує програму для iPhone, він може бути впевнений у бездоганній продуктивності та неймовірному досвіді. Ця довіра та доброзичливість до спадщини Apple зуміли зібрати велику та лояльну базу споживачів iOS застосунків;

– коли справа доходить до того, скільки часу потрібно для створення програми, застосунки для iOS займають майже на 28% менше часу, ніж програми для Android з такими ж характеристиками.

Xcode – це інтегроване середовище розробки (IDE) від Apple для macOS, яке використовується для розробки програмного забезпечення для macOS, iOS, iPadOS, watchOS і tvOS. Ця IDE доступна безкоштовно в Mac App Store і на веб-сайті Apple Developer. Інтерфейс Xcode можна побачити на рисунку 3.1. Зареєстровані розробники також можуть завантажувати попередні випуски та попередні версії набору через веб-сайт Apple Developer. Xcode містить інструменти командного рядка, які дозволяють розробку в стилі UNIX через програму Terminal у macOS. Їх також можна завантажити та встановити без графічного інтерфейсу користувача.

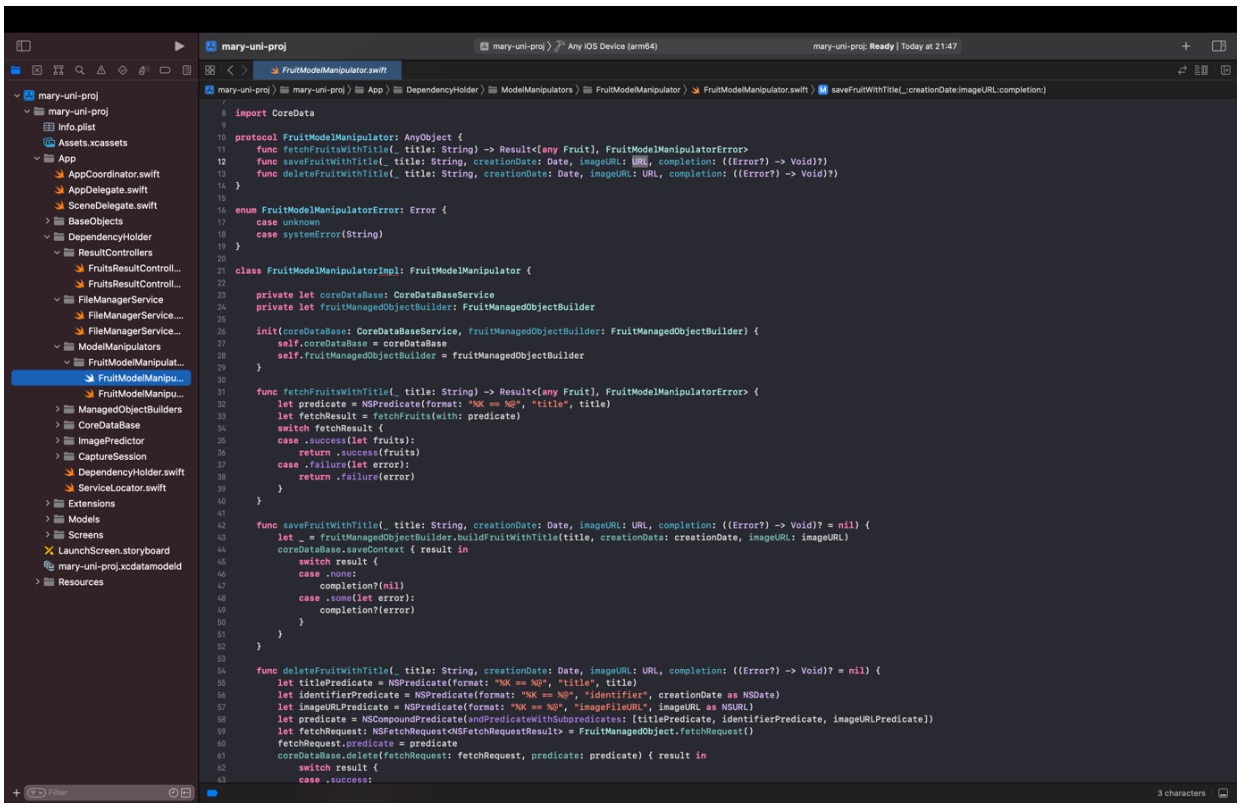

Рисунок 3.1 – Інтерфейс IDE Xcode

Xcode підтримує вихідний код для мов програмування: C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit (Rez) і Swift, з різними моделями програмування, включаючи, але не обмежуючись Cocoa , Carbon і Java.

Xcode також інтегрує вбудовану підтримку керування вихідним кодом за допомогою системи та протоколу контролю версій Git, дозволяючи користувачеві створювати та клонувати сховища Git (які можна розміщувати на сайтах розміщення сховищ вихідного коду, таких як GitHub, Bitbucket і Perforce або самостійно розміщений за допомогою програмного забезпечення з відкритим вихідним кодом, такого як GitLab), а також для фіксації, надсилання та вилучення змін, усе з Xcode, автоматизуючи завдання, які традиційно виконувалися б за допомогою Git з командного рядка.

Як правило, для реалізації iOS застосунків переважна більшість розробників обирають мову програмування Swift.

Swift – це багатопарадигмальна скомпільована мова програмування високого рівня загального призначення, розроблена Apple Inc та спільнотою відкритих кодів. Вперше випущений у 2014 році, Swift був розроблений як заміна попередньої мови програмування Apple Objective-C, оскільки Objective-C майже не змінювався з початку 1980-х років і не мав сучасних функцій мови програмування.

Apple планувала, що Swift підтримує багато основних концепцій, пов’язаних з Objective-C, зокрема динамічну диспетчеризацію, поширене пізнє зв’язування, розширюване програмування та подібні функції, але «безпечнішим» способом, що полегшує виявлення програмних помилок. Swift має функції, які виправляють деякі поширені помилки програмування, як-от розіменування нульового вказівника, і забезпечує синтаксичний цукор, щоб уникнути піраміди приреченості. Swift підтримує концепцію розширюваності протоколу, систему розширюваності, яка може бути застосована до типів, структур і класів, яку Apple просуває як реальну зміну парадигм програмування, яку вони називають «протокольно-орієнтованим програмуванням».

Для використання нейронних мереж в iOS застосунках використовується бібліотека Core ML.

Core ML застосовує алгоритм машинного навчання до набору навчальних даних для створення моделі. Вона використовуєте модель, щоб робити прогнози на основі нових вхідних даних. Моделі можуть виконувати різноманітні завдання, які було б важко або непрактично написати в коді. Наприклад, можна навчити модель класифікувати фотографії або виявляти певні об’єкти на фотографії безпосередньо за її пікселями. Інтерфейс Core ML преставлено на рисунку 3.2.

Створивши модель, вона інтегрується у застосунок і розгортається на пристрої користувача. iOS застосунки використовують API Core ML і дані користувача, щоб робити прогнози та навчати чи налаштовувати модель.

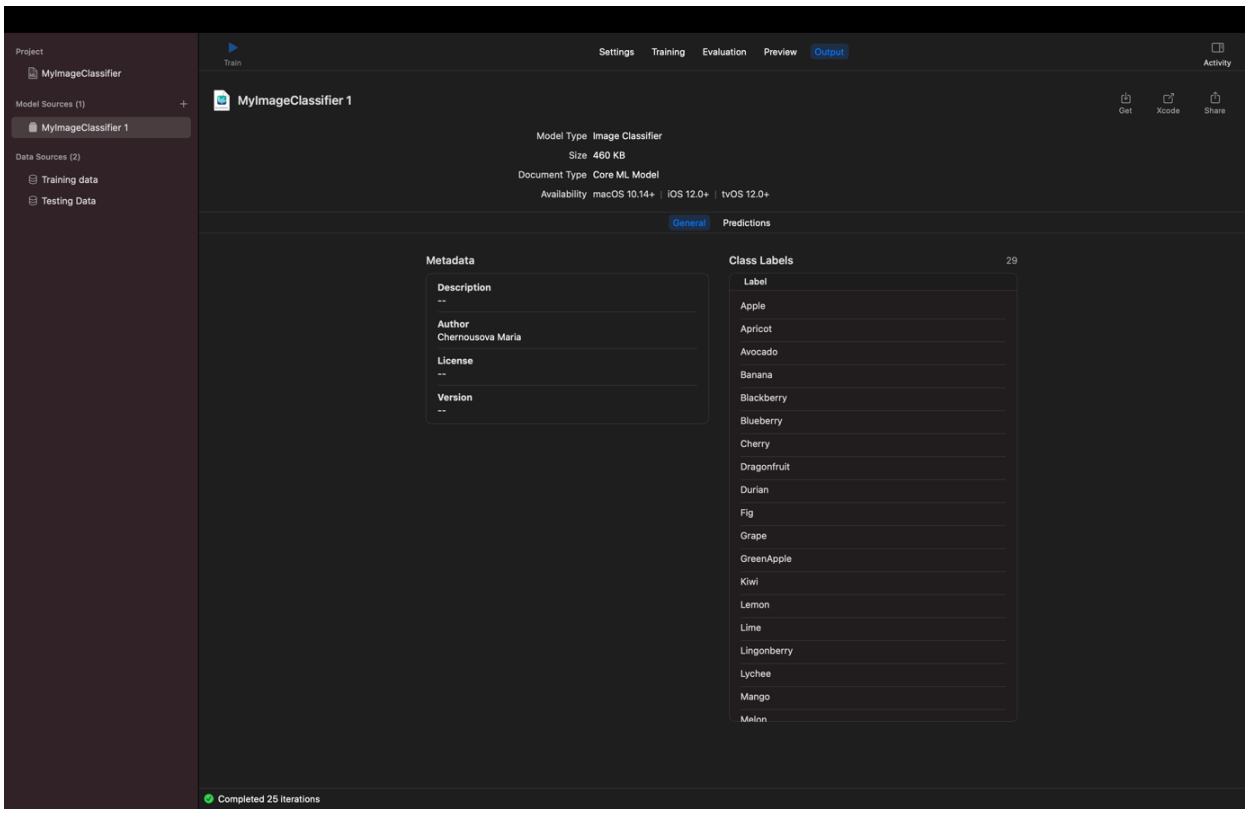

Рисунок 3.2 – Інтерфейс Core ML

Також важливою є бібліотека, що дозволяє працювати з фотографіями. AVFoundation – це мультимедійна структура з API у Objective-C і Swift, яка надає високорівневі сервіси для роботи з аудіовізуальними медіа на основі часу в операційних системах Apple: iOS, macOS, tvOS і watchOS.

3.2 Програмна реалізація застосунку

3.2.1 Розробка класів та тренування моделі.

Першим етапом програмної реалізації є підготовка вибірки даних. Підготовка тестових та навчальних зображень з нуля є дуже складним процесом, що займає дуже велику кількість часу, але наразі в інтернеті немає готових моделей, які б охоплювали більше 10 класів фруктів. Саме тому було прийнято рішення створювати власну вибірку даних. Приклад структури вибірки зображень представлено на рисунку 3.3.

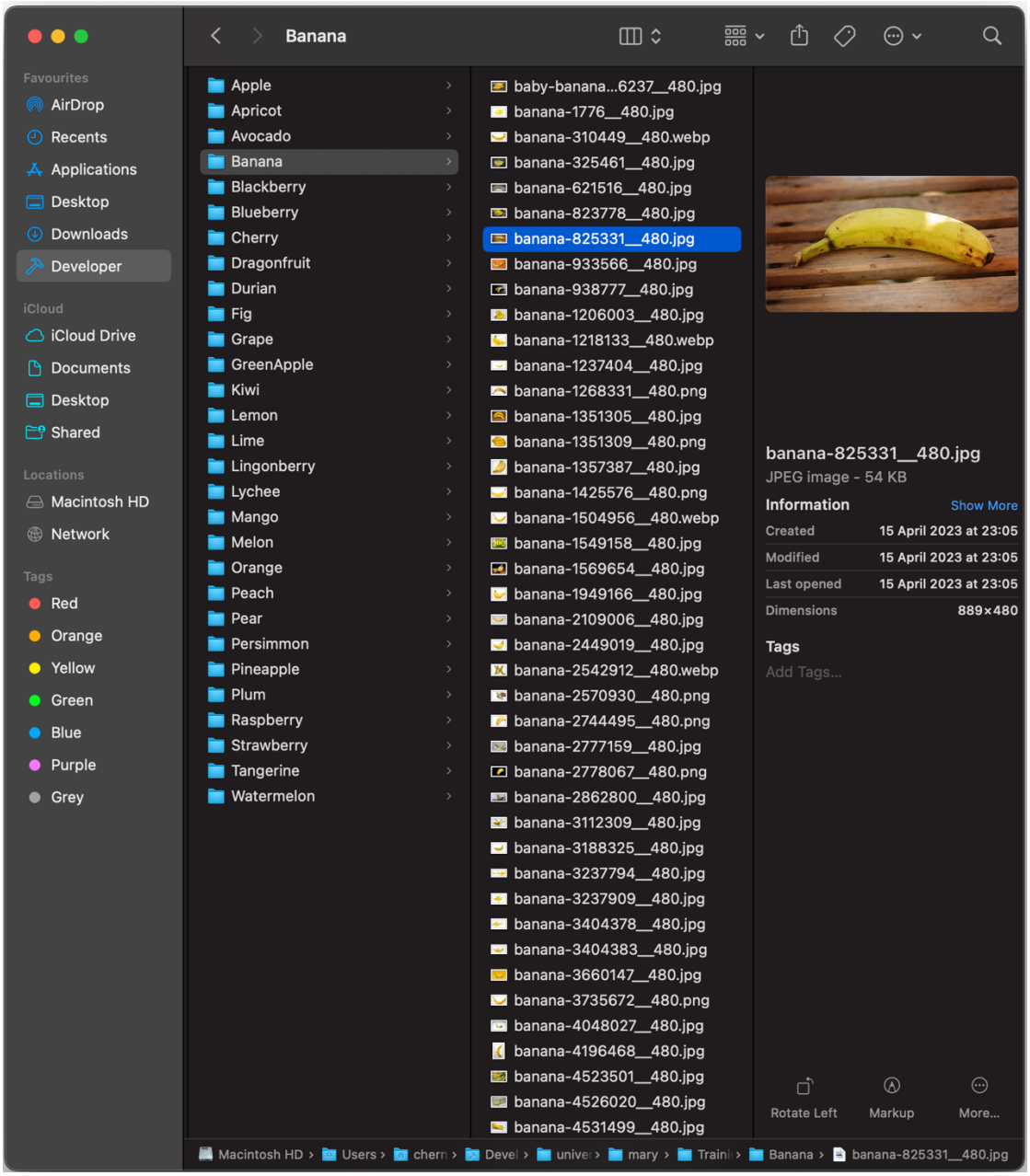

Рисунок 3.3 – Структура класів вибірки

В процесі аналізу були виявлені класи зображень, що відповідають тим різновидам фруктів, які можна зустріти у нас в магазинах. Усього кінцева вибірка містить 29 класи фруктів. Загальна кількість усіх зображень – 3450.

Зображення у вибірці мають різні характеристики. Вони відрізняються за ступінню зашумленості, освітленості та затемненості, а також розмиття (рис. 3.4). Такий вибір зображень зумовлений тим, що у житті зображення фруктів можуть деформуватися від умов освітленості, наявності упакування або різних пошкоджень.

Рисунок 3.4 – Приклад зображень з навчальної вибірки

Наступним важливим етапом є створення моделі за допомогою бібліотеки CoreML. Для цього потрібно обрати шаблон для класифікації зображень серед усіх запропонованих. Вибір шаблону для навчання моделі представлено на рисунку 3.5.

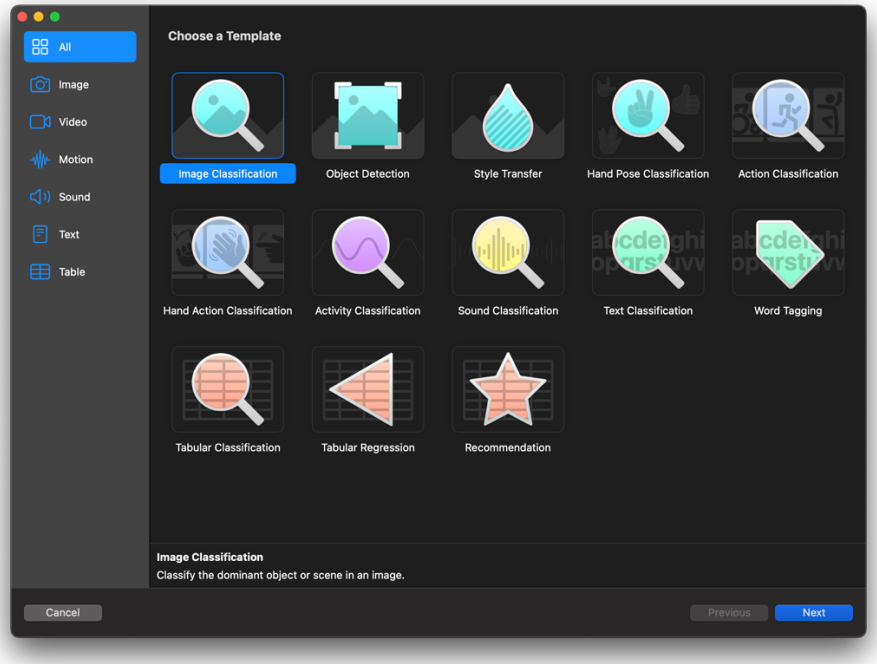

Рисунок 3.5 – Шаблони CoreML для навчання моделей

Далі потрібно додати створену вибірку даних до моделі та розпочати процес тренування. На цьому етапі робота бібліотеки полягає у тому щоб витягнути об’єкти з вибірки та обробити їх. Далі розпочинається сам процес тренування. В інтерфейсі з’являється графік, що відображає точність навчання, точність перевірки для кожної ітерації. Графік тренування моделі показано на рисунку 3.6.

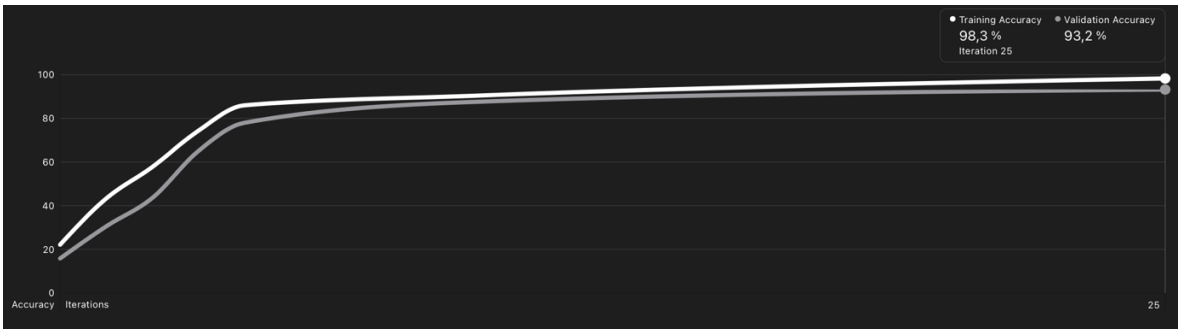

Рисунок 3.6 – Графік тренування моделі CoreML

Коли навчання завершиться, формується таблиця показників із переліком точності та запам’ятовування для усіх 29 класів (рис. 3.7).

Training data
Apr 28, 2023 at 9:41 PM
29 classes with 3450 items

Filter class

Class	Count	Precision	Recall	F1 Score
Watermelon	104	98%	100%	0,99
Tangerine	188	91%	96%	0,94
Strawberry	110	100%	100%	1,0
Raspberry	159	99%	100%	1,0
Plum	128	98%	100%	0,99
Pineapple	99	100%	99%	0,99
Persimmon	99	100%	99%	0,99
Pear	117	100%	99%	1,0
Peach	109	94%	99%	0,96
Orange	175	96%	91%	0,94
Melon	101	100%	98%	0,99
Mango	107	96%	100%	0,98
Lychee	91	100%	100%	1,0
Lingonberry	96	100%	100%	1,0
Lime	92	100%	99%	0,99
Lemon	116	97%	96%	0,96
Kiwi	99	100%	99%	0,99
GreenApple	159	97%	99%	0,98
Grape	135	100%	100%	1,0
Fig	127	100%	100%	1,0
Durian	50	100%	100%	1,0
Dragonfruit	57	100%	100%	1,0
Cherry	113	100%	100%	1,0
Blueberry	113	100%	100%	1,0
Blackberry	102	100%	99%	1,0
Banana	114	100%	100%	1,0
Avocado	107	100%	100%	1,0
Apricot	109	98%	93%	0,95
Apple	97	97%	90%	0,93

Рисунок 3.7 – Результати точності та запам’ятовування для усіх класів

Також можна побачити графік з кількістю об’єктів для кожного класу. Графік наведено на рисунку 3.8. Таким чином можна зробити аналіз стосовно навченості самої моделі. Після цього етап навчання моделі вважається завершеним.

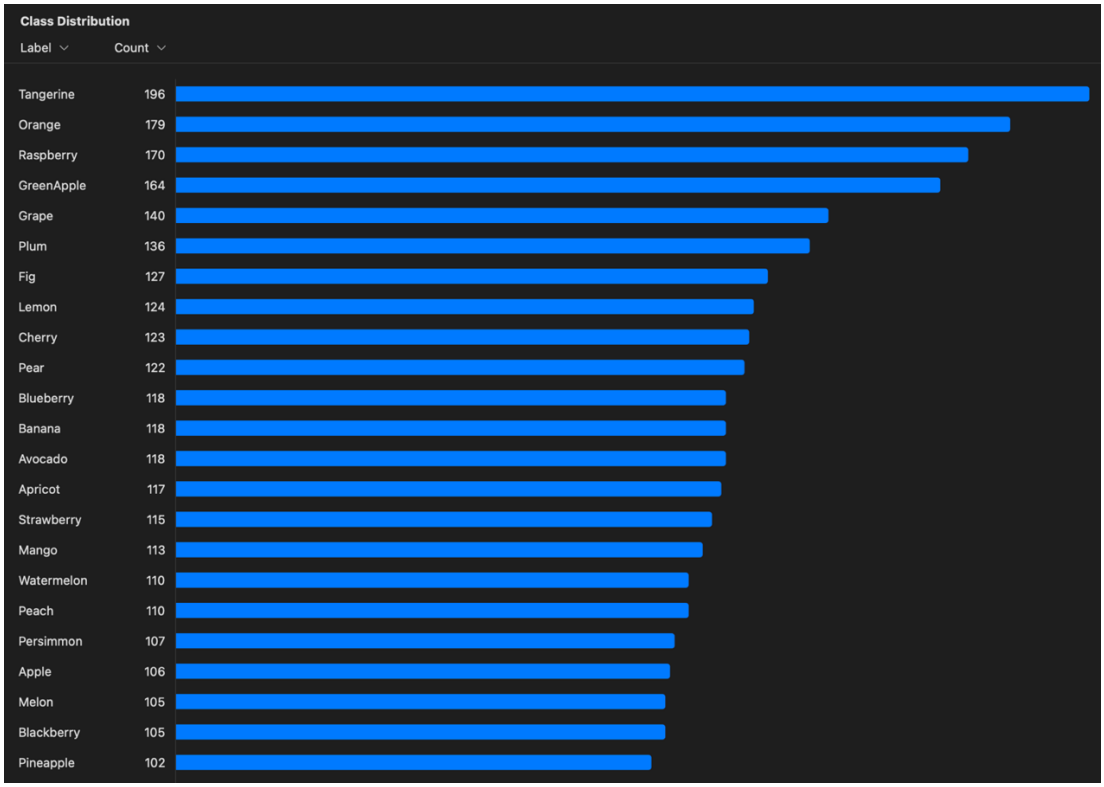

Рисунок 3.8 – Графік з кількістю об’єктів для кожного класу

3.2.2 Використання API для визначення характеристик класифікованого об’єкту

Одна з головних цілей створення даного застосунку – це показати користувачу вітамінний склад фрукту, який він хоче визначити. Для цього до застосунку підключається API, в якому зберігаються дані для кожного класу фрукту. У роботі використовується API з відкритим доступом Fruityvice.

Наразі API складається з двох функцій: отримання даних для певного фрукта або всіх фруктів і функції додавання власних даних. Приклад того, як виглядає тіло відповіді, можна побачити в Лістингу 3.1. Щоб отримати

відображені дані, потрібно виконати HTTP-виклик GET на ресурсі /api/fruit/{ID} або /api/fruit/{name} IP-адреси Fruityvice. Щоб додати дані, потрібно зробити виклик HTTP PUT на ресурсі /api/fruit з даними фрукта у форматі JSON у тілі запиту.

Лістинг 3.1 Тіло відповіді виклику HTTP:

```
{
  "name": "Apple",
  "id": 6,
  "family": "Rosaceae",
  "order": "Rosales",
  "genus": "Malus",
  "nutritions": {
    "calories": 52,
    "fat": 0.4,
    "sugar": 10.3,
    "carbohydrates": 11.4,
    "protein": 0.3
  }
```

Для виконання HTTP-запиту API в iOS застосунку є спеціальний метод *URLSession*. З самого початку потрібно створити завдання даних, що є екземпляром класу *URLSessionDataTask.* Завдання завжди прив'язане до екземпляра *URLSession*. Для простоти виконання потрібно запитати у класу *URLSession* спільний об'єкт сеансу, синглтон, через його властивість спільного класу. Потім екземпляр *URLSession* створює завдання даних, викликавши метод *dataTask(with:completionHandler:)*. Цей метод повертає екземпляр *URLSessionDataTask* і приймає два аргументи, об'єкт URL-адреси та обробник завершення. Обробник завершення (закриття) виконується, коли завдання даних завершується, успішно чи неуспішно. Обробник завершення

приймає три аргументи: необов'язковий об'єкт *Data*, необов'язковий об'єкт *URLResponse* та необов'язковий об'єкт *Error*.

Лістинг 3.2 Приклад використання *URLSession:*

```
import UIKit
let url = URL(string: "https://bit.ly/2LMtByx")!
let task = URLSession.shared.dataTask(with: url) { data, response, error in
}
```

dataTasks використовують обробники завершення, і вони завжди повертають ті самі типи інформації: дані, відповідь і помилку.

Якщо *dataTask* повертає помилку, то потрібно знати про це раніше, ніж у користувача виникнуть проблеми із використанням застосунку. Це означає, що потрібно скерувати код так, щоб витончено обробляти помилки. Це також означає, що не потрібно буде намагатися прочитати дані та щось з ними зробити, оскільки під час повернення даних є помилка.

Нижче показано приклад обробки помилки.

Лістинг 3.3 Обробка помилок:

```
if let error = error {
  print("Error accessing fruityvice.com: /(error)")
  return
}
```

Далі йде обробка відповіді. Відповідь можна перетворити на *httpResponse.* Таким чином можна переглянути коди стану та прийняти деякі рішення на основі коду. Наприклад, якщо код статусу 404, означає, що сторінку не знайдено.

Наведений нижче код використовує запобіжник, щоб перевірити наявність двох речей – відповідь має відповідати типу *HTTPURLResponse* та

має код стану у проміжку 200...299. Якщо обидва існують, це дозволяє коду продовжувати до наступного оператора після захисного речення. Якщо будь-який із операторів не виконується, йде вихід з функції. Це типовий випадок використання захисної пропозиції.

Лістинг 3.4 Обробка відповіді за допомогою запобіжника:

```
guard let httpResponse = response as? HTTPURLResponse,
        (200...299).contains(httpResponse.statusCode) else {
   print("Error with the response, unexpected status code: \(response)")
   return
  }
```

Для отримання відповіді для конкретного фрукта потрібно зробити обмеження на запит. Об’єкт *URLRequest* дозволяє налаштувати HTTP-запит, який виконує URL-сеанс. Можна встановити метод HTTP за допомогою властивості *httpMethod*. У цьому випадку це не обов’язково, оскільки за замовчуванням завжди стоїть GET.

Лістинг 3.5 Налаштування HTTP-запиту:

```
request.httpMethod = "GET"
```

Також потрібно визначити поля заголовка HTTP запиту. Для цього потрібно використовувати властивість *allHTTPHeaderFields,* словник типу *[String:String]*.

Лістинг 3.6 Приклад встановлення запиту для полуниці:

```
request.allHTTPHeaderFields = [
   " name ": " Strawberry "
]
```

Останнім кроком є декодування отриманого результату. Для цього використовується клас *JSONDecoder*. Він вимагає наявності структури яка буде мати тип Codable. J*SONDecoder* з'єднує результат, який було отримано з HTTP запиту з структурою Fruit.

Лістинг 3.7 Декодування результату запиту:

```
if let data = data,
    let result = try? JSONDecoder().decode(Fruit.self, from: data) {
    completionHandler(results ?? nil)
}
```

Таким чином після сканування фрукта користувач має можливість побачити вікно з результатом, де буде розміщатись зроблена їм фотографія, назва фрукту, визначеного нейронною мережею та опис. Опис складається з параметрів, які були надані в HTTP запиті та відповідають вітамінному складу визначеного фрукта.

Результат запиту api/fruit/apple в застосунку показано на рисунку 3.9.

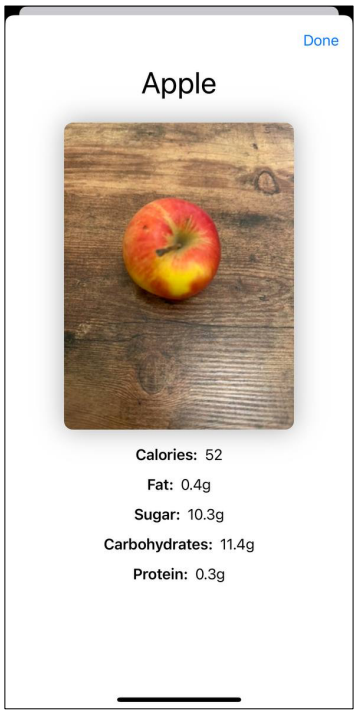

Рисунок 3.9 – Екран з результатом запиту

3.2.3 Створення скануючого вікна для отримання зображення

Для створення скануючого вікна використовується AVFoundation. AVFoundation поєднує кілька основних технологічних напрямків, які разом охоплюють широкий спектр завдань для перевірки, відтворення, захоплення та обробки аудіовізуальних медіа на платформах Apple.

Щоб виконати захоплення в реальному часі, створюються екземпляр сеансу захоплення та додаються відповідні входи та виходи. Наступний фрагмент коду ілюструє, як налаштовувся пристрій захоплення для запису відео та створення фото.

Лістинг 3.8 Створення сеансу захоплення з входами та виходами:

```
        let deviceInput: AVCaptureDeviceInput
        do {
           deviceInput = try AVCaptureDeviceInput(device: captureDevice)
    } catch {
       throw CaptureSessionServiceError.undefinedDeviceInput
    }

let photoOutput = AVCapturePhotoOutput()
    guard captureSession.canAddOutput(photoOutput) else {
       throw CaptureSessionServiceError.cannotAddPhotoOutput
    }
    captureSession.addOutput(photoOutput)
```

Викликається метод *startRunning()*, щоб розпочати потік даних від входів до виходів, і викликається метод *stopRunning()*, щоб зупинити потік.

Лістинг 3.9 Приклад методу *startRunning():*

```
    func startSession() {
        processQueue.async { [weak self] in
           self?.captureSession?.startRunning()
        }}
```

Лістинг 3.10 Приклад методу *stopRunning():*

```
func stopSession() {
    processQueue.async { [weak self] in
        self?.captureSession?.stopRunning()
        self?.captureSession = nil
        self?.photoOutput = nil
        self?.metadata.send(nil)
    }}
```

Властивість *sessionPreset* використовується, щоб налаштувати рівень якості, бітрейт та інші параметри виведення. Більшість поширених конфігурацій захоплення доступні через попередні налаштування сеансу; однак деякі спеціалізовані параметри (наприклад, висока частота кадрів) вимагають безпосереднього встановлення формату захоплення на екземплярі *AVCaptureDevice*.

Лістинг 3.11 Приклад налаштування сеансу:

```
AVCaptureDevice.default(for: .video)
func takePhoto(imageDataHandler: @escaping (CaptureSessionImageData)
-> Void) {
        self.imageDataHandler = imageDataHandler
        let settings = AVCapturePhotoSettings(format: [AVVideoCodecKey:
AVVideoCodecType.jpeg])
        photoOutput?.capturePhoto(with: settings, delegate: self)
    }
    func photoOutput(_ output: AVCapturePhotoOutput,
didFinishProcessingPhoto photo: AVCapturePhoto, error: Error?) {
        guard let imageData = photo.fileDataRepresentation() else { return }
        imageDataHandler?(imageData) }
```

Скануюче вікно застосунку представлено на рисунку 3.10.

Рисунок 3.10 – Скануюче вікно застосунку

3.2.4 Збереження даних у застосунку

У застосунку розроблено сервісний шар, який відповідає за збереження даних у застосунку за допомогою Core Data.

В першу чергу було створено модель, яка визначає параметри, що повинні зберігатися, та їх типи. Модель FruitManagedObject показано на рисунку 3.11.

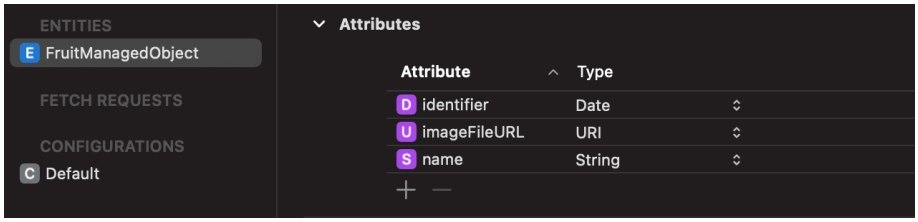

Рисунок 3.11 – Модель FruitManagedObject

В даному застосунку Core Data відповідає за такі дії:

- збереження даних;
- видалення даних;
- завантаження вже збережених даних.

Щоб не було перенавантаження системи, збереження даних виконується при кожному закритті застосунка користувачем. Система порівнює вже збережені дані та ті, які відображались користувачу перед закриттям застосунку. Якщо дані відрізняються, то зберігається остання версія.

Лістинг 3.12 Збереження даних у Core Data:

```
func saveContext () {
    let context = persistentContainer.viewContext
    if context.hasChanges {
        do {
            try context.save()
        } catch {
            let nserror = error as NSError
            fatalError("Unresolved error \(nserror), \(nserror.userInfo)")
        }
    }
}
```

Для видалення даних потрібно вказати *NSPredicate*, по якому система визначить, які саме дані потрібно видалити. Після того як потрібний об’єкт визначено, створюється спеціальний запит *NSBatchDeleteRequest*. По цьому запиту система опрацьовує видалення і повертає відповідь чи було видалення успішним.

Лістинг 3.13 Видалення даних з Core Data:

```
func delete(fetchRequest: NSFetchRequest<NSFetchRequestResult>,
```

```
                predicate: NSPredicate,
completionHandler: @escaping (Result<Int, CoreDataBaseError>)    ->    Void)
fetchRequest.predicate = predicate
let batchDeleteRequest = NSBatchDeleteRequest(fetchRequest: fetchRequest
batchDeleteRequest.resultType = .resultTypeCount
            do {
                let deletedCount = try mainContext.execute(batchDeleteRequest) as!
NSBatchDeleteResult
                completionHandler(.success(deletedCount.result as! Int))
            } catch let error as NSError {
                completionHandler(.failure(.systemError(error)))
            }
```

Екран з можливістю видалення представлено на рисунку 3.12.

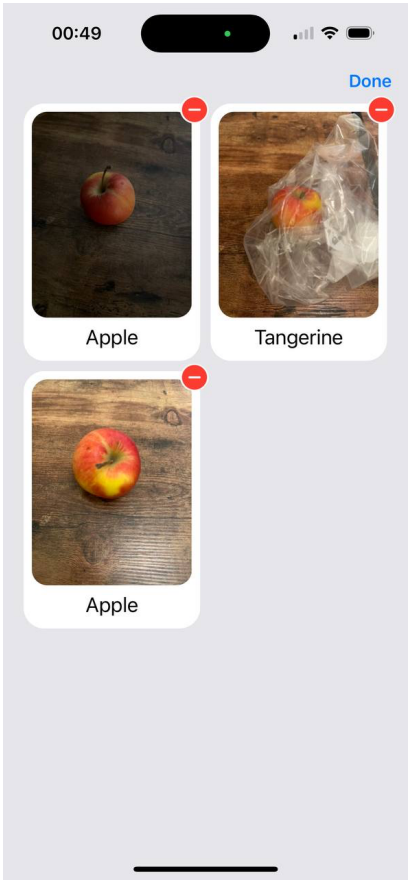

Рисунок 3.12 – Екран застосунку з можливістю видалення

Один з найважливіших етапів – завантаження збережених даних та показ їх користувачу. За це відповідає метод *fetchRequest*. Один з головних нюансів завантаження даних це те, що воно повинно відбуватися асинхронно для того щоб не було блокування інтерфейсу користувача.

Лістинг 3.14 Завантаження даних з Core Data:

```
func fetch<T: NSFetchRequestResult>(fetchRequest: NSFetchRequest<T>,
 completionHandler: @escaping (Result<[T], CoreDataBaseError>) -> Void) {
        let asynchronousFetchRequest = NSAsynchronousFetchRequest(fetchRequest: fetchRequest) { fetchRequest in

            if let finalResult = fetchRequest.finalResult {

                completionHandler(.success(finalResult))
            } else {

                completionHandler(.failure(.unknown))
            }
        }

        do {

            try mainContext.execute(asynchronousFetchRequest)
        } catch let error as NSError {

            completionHandler(.failure(.systemError(error)))
        }
    }
```

3.3 Тестування розробленого застосунку та аналіз результатів

Після того як тренування завершено розпочинається процес тестування. Для цього потрібно створити вибірку даних для тих самих класів, але кожен клас має містити інші зображення об’єктів. Бібліотека CoreML запускає процес розпізнання та класифікації кожного об’єкту. Після цього відбувається процес порівняння вихідного класу та отриманого результату.

Для тестування було створено вибірку з 29 класів та 2043 зображень. Усі з цих зображень максимально наближені до тих, що можна зустріти у реальному житті (рис. 3.13).

Рисунок 3.13 – Приклад зображень для тестування

Під час тестування моделі було визначено, що точність визначення результату становить 91%. Це дуже високий показник. Статистику результатів тестування показано на рисунку 3.14.

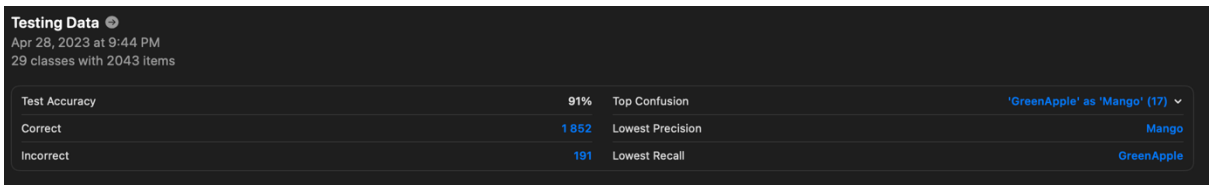

Рисунок 3.14 – Результати тестування моделі

Крім того, існує також відмінність у якості розпізнавання зображень зблизу та у віддаленні. Якщо на вхід до моделі подаються зображення фруктів зблизька, то модель у більшості випадків повертає правильний результат розпізнавання. Але по мірі віддаленості точність стає гіршою. Також, звісно, точність зменшувалася, якщо фрукт був прикритий опакуванням, або мав погану освітленість. Результати перевірки представлено на рисунку 3.15.

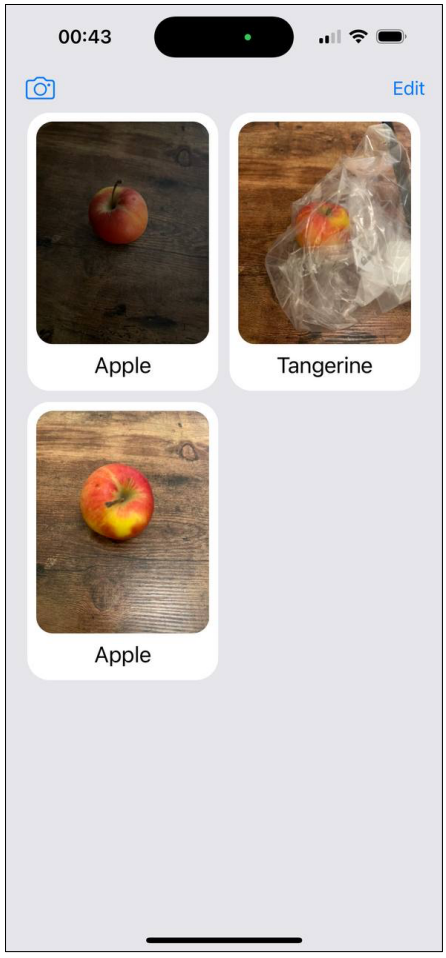

Рисунок 3.15 – Результат детектування яблука при різних умовах

Результати тестування для усіх класів моделі представлено в таблиці 3.1.

Таблиця 3.1 – Результати розпізнавання моделі

Назва класу	**Кількість протестованих об'єктів**	**Точність**	**Показник F1**
Apple	83	82%	0,86
Apricot	61	77%	0,8
Avocado	62	98%	0,9
Banana	63	100%	1,0
Blackberry	64	98%	0,99
Blueberry	82	94%	0,97
Cherry	78	88%	0,94
Dragonfruit	38	100%	0,96
Durian	33	97%	0,98
Fig	88	100%	0,95
Grape	110	100%	1,0
GreenApple	116	74%	0,83
Kiwi	85	98%	0,97
Lemon	79	97%	0,85
Lime	59	78%	0,85
Lingonberry	70	97%	0,93
Lychee	73	89%	0,94
Mango	57	95%	0,64
Melon	65	92%	0,9
Orange	87	71%	0,77
Peach	54	83%	0,87
Pear	54	94%	0,93
Persimmon	62	97%	0,98
Pineapple	58	100%	1,0
Plum	77	84%	0,9
Raspberry	78	96%	0,96
Strawberry	52	100%	0,97
Tangerine	102	75%	0,84
Watermelon	53	100%	0,99

Як видно з результатів, наведених в таблиці вище, було отримано 2 різних оцінки тестування – точність та показник F1. Точність – це доля правильно класифікованих об’єктів зі всіх об’єктів класу. Найгірші результати згідно з точністю було отримано для класів «Абрикос», «Зелене яблуко», «Лайм» та «Апельсин». Точність визначення для цих класів менше 80%. Показник F1 – це спосіб поєднання точності та запам’ятовування моделі, і він визначається як середнє гармонійне точності та запам’ятовування моделі, де результат 100% дорівнює 1,0. Він використовується для оцінки бінарних систем класифікації, які класифікують об’єкти на «позитивні» та «негативні». Найгірші значення згідно з показником F1 було отримано для класів «Манго» та «Апельсин». Для них значення показнику F1 менше 0,8.

3.4 Перспективи подальшої роботи

Наразі простежуються декілька перспективних ідей для подальшої роботи над проблемою:

– покращити вибірку даних фруктів, урізноманітніти зображення фруктів з різними ступінями віддаленості, деформації, освітлення та ракурсами зйомки [49 – 53];

– розширити кількість класів вибірки, щоб наблизити до актуальної кількості видів фруктів;

– створити власне API, яке буде відповідати усім заданим класам та містити більше важливої інформації про кожен фрукт;

– сфокусувати роботу на розпізнавання фруктів у реальних умовах та наблизити роботу до можливості практичного використання [54 – 58];

– розширити функціонал застосунка з можливістю розпізнання фруктів, овочей, готових страв.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований iOS застосунок для розпізнавання вітамінного складу фруктів за їх зображенням.

Для досягнення мети роботи була обрана нейронна мережа для класифікації фруктів, а саме архітектура MPSCNN, розроблена Apple для прискорення обчислень згорткової нейронної мережі на пристроях iOS.

Для програм машинного навчання використана платформа CoreML, що надає набір інструментів і API, які дозволяють створювати та розгортати моделі машинного навчання на пристроях Apple.

Збереження та керування даними у застосунку було проведено за допомогою CoreData -високорівневого об’єктно-орієнтований інтерфейсу до основної бази даних SQLite, що надає ряд потужних функцій для керування складними об’єктними графами та зв’язками.

Застосунок створено з використанням сучасних технологій, а саме: середи для розробки програмного забезпечення XCode, мови програмування Swift, бібліотеки AVFoundation, бібліотеки Foundation, бібліотеки UIKit.

Виконана робота має практичне застосування – створення застосунку для розпізнавання вітамінного складу фруктів за їх зображенням, що дозволяє у зручний спосіб контролювати своє харчування та підтримувати здоровий спосіб життя.

Робота має перспективи подальшої розробки в направленні розширення переліку об’єктів для розпізнавання та класифікації (овочі, готові страви), що дасть можливість створити універсальний сучасний застосунок для контролю за корисним збалансованим харчуванням, що є одним з кроків вирішення проблеми здорового харчування, яка є досить актуальною в сучасному суспільстві.

Результати роботи PPR апробовано у вигляді тез доповідей під час Міжнародного молодіжного форуму «Радіоелектроніка і молодь у XXI столітті» [59].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Zaidi, Q., & Bostic, M. (2008). Color strategies for object identification. *Vision research*, *48*(26), 2673-2681.

2. Bolle, R. M., Connell, J. H., Haas, N., Mohan, R., & Taubin, G. (1996, December). Veggievision: A produce recognition system. In Proceedings Third IEEE Workshop on Applications of Computer Vision. WACV'96 (pp. 244-251). IEEE.

3. Rocha, A., Hauagge, D. C., Wainer, J., & Goldenstein, S. (2010). Automatic fruit and vegetable classification from images. Computers and Electronics in Agriculture, 70(1), 96-104.

4. Arivazhagan, S., Shebiah, R. N., Nidhyanandhan, S. S., & Ganesan, L. (2010). Fruit recognition using color and texture features. Journal of Emerging Trends in Computing and Information Sciences, 1(2), 90-94.

5. Faria, F. A., dos Santos, J. A., Rocha, A., & Torres, R. D. S. (2012, August). Automatic classifier fusion for produce recognition. In 2012 25th SIBGRAPI Conference on Graphics, Patterns and Images (pp. 252-259). IEEE.

6. Chowdhury, M. T., Alam, M. S., Hasan, M. A., & Khan, M. I. (2013). Vegetables detection from the glossary shop for the blind. IOSR Journal of Electrical and Electronics Engineering, 8(3), 43-53.

7. Danti, A., Madgi, M., & Anami, B. S. (2012). Mean and range color features based identification of common Indian leafy vegetables. International Journal of Signal Processing, Image Processing and Pattern Recognition, 5(3), 151-160.

8. Suresha, M., Kumar, K. S., & Kumar, G. S. (2012). Texture features and decision trees based vegetables classification. IJCA Proceedings on National Conferecne on Advanced Computing and Communications 2012, 1, 21-26.

9. Dubey, S. R. (2012). Automatic recognition of fruits and vegetables and detection of fruit diseases. Master's theses, GLA University Mathura, India.

10. Dubey, S. R., & Jalal, A. S. (2012). Robust approach for fruit and vegetable classification. Procedia Engineering, 38, 3449-3453.

11. Dubey, S. R., & Jalal, A. S. (2013). Species and variety detection of fruits and vegetables from images. International Journal of Applied Pattern Recognition, 1(1), 108-126.

12. Arefi, A., Motlagh, A. M., Mollazade, K., & Teimourlou, R. F. (2011). Recognition and localization of ripen tomato based on machine vision. Australian Journal of Crop Science, 5(10), 1144-1149.

13. Bulanon, D. M., Burks, T. F., & Alchanatis, V. (2008, January). Improving fruit detection for robotic fruit harvesting. In International Symposium on Application of Precision Agriculture for Fruits and Vegetables 824 (pp. 329-336).

14. Haidar, A., Dong, H., & Mavridis, N. (2012, October). Image-based date fruit classification. In 2012 IV International Congress on Ultra Modern Telecommunications and Control Systems (pp. 357-363). IEEE.

15. Jiménez, A. R., Jain, A. K., Ceres, R., & Pons, J. L. (1999). Automatic fruit recognition: a survey and new results using range/attenuation images. Pattern recognition, 32(10), 1719-1736.

16. Lino, A. C. L., Sanches, J., & Fabbro, I. M. D. (2008). Image processing techniques for lemons and tomatoes classification. Bragantia, 67, 785-789.

17. Liu, Y., Chen, B., & Qiao, J. (2011). Development of a machine vision algorithm for recognition of peach fruit in a natural scene. Transactions of the ASABE, 54(2), 695-702.

18. Patel, H. N., Jain, R. K., & Joshi, M. V. (2011). Fruit detection using improved multiple features based algorithm. International journal of computer applications, 13(2), 1-5.

19. Rahmana, M. H., Pickering, M. R., Kerr, D., Boushey, C. J., & Delp, E. J. (2012, July). A new texture feature for improved food recognition accuracy in a mobile phone based dietary assessment system. In 2012 IEEE International Conference on Multimedia and Expo Workshops (pp. 418-423). IEEE.

20. Mashtalir, S., & Mikhnova, O. (2017). Detecting significant changes in image sequences. Multimedia Forensics and Security: Foundations, Innovations, and Applications, 161-191.

21. Bilonoh, B., Bodyanskiy, Y., Kolchygin, B., & Mashtalir, S. (2022). Tunable Activation Functions for Deep Neural Networks. In Lecture Notes in Computational Intelligence and Decision Making: 2021 International Scientific Conference" Intellectual Systems of Decision-making and Problems of Computational Intelligence", Proceedings (pp. 624-633). Springer International Publishing.

22. Artificial Neural Networks.What they are & why they matter. URL: https://www.sas.com/en_us/insights/analytics/neural-networks.html#importance (дата звернення 12.04.2023).

23. Professor's perceptron paved the way for AI – 60 years too soon. URL: https://news.cornell.edu/stories/2019/09/professors-perceptron-paved-way-ai-60-years-too-soon (дата звернення 20.04.2023).

24. Guide to Generative Adversarial Networks (GANs) in 2023. URL: https://viso.ai/deep-learning/generative-adversarial-networks-gan/ (дата звернення 24.04.2023).

25. Overview of GAN Structure. URL: https://developers.google.com/machine-learning/gan/gan_structure (дата звернення 25.04.2023).

26. A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (дата звернення 24.04.2023).

27. Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, Al-Shamma, O., Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. Journal of big Data, 8(1), 1-74.

28. R-CNN, Fast R-CNN, Faster R-CNN, YOLO – Object Detection Algorithms. URL: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e (дата звернення 24.04.2023).

29. Yan, B., Fan, P., Lei, X., Liu, Z., & Yang, F. (2021). A real-time apple targets detection method for picking robot based on improved YOLOv5. Remote Sensing, 13(9), 1619.

30. Metal Performance Shaders. URL: https://developer.apple.com/documentation/metalperformanceshaders (дата звернення 26.04.2023).

31. TensorFlow. URL: https://www.tensorflow.org (дата звернення 26.04.2023).

32. PYTORCH. URL: https://pytorch.org (дата звернення 26.04.2023).

33. Scikit-learn. URL: https://scikit-learn.org/stable/ (дата звернення 26.04.2023).

34. The Microsoft Cognitive Toolkit. URL: https://learn.microsoft.com/en-us/cognitive-toolkit/ (дата звернення 26.04.2023).

35. Fast Artificial Neural Network Library. URL: http://leenissen.dk/fann/wp/ (дата звернення 26.04.2023).

36. Armadillo. URL: https://arma.sourceforge.net (дата звернення 26.04.2023).

37. Core ML. URL: https://developer.apple.com/documentation/coreml (дата звернення 30.04.2023).

38. Statista Dossier about mobile app usage. URL: https://www.statista.com/study/11559/mobile-app-usage-statista-dossier/ (дата звернення 01.05.2023).

39. Mashtalir, S. V., Stolbovyi, M. I., & Yakovlev, S. V. (2019). Clustering video sequences by the method of harmonic k-means. Cybernetics and Systems Analysis, 55, 200-206.

40. Mashtalir, S., & Mashtalir, V. (2020). Spatio-temporal video segmentation. Advances in Spatio-Temporal Segmentation of Visual Data, 161-210.

41. Bodyanskiy, Y. V., Tyshchenko, O. K., & Mashtalir, S. V. (2019). Fuzzy clustering high-dimensional data using information weighting. In Artificial Intelligence and Soft Computing: 18th International Conference, ICAISC 2019, Zakopane, Poland, June 16–20, 2019, Proceedings, Part I 18(pp. 385-395). Springer International Publishing.

42. Bodyanskiy, Y., Shafronenko, A., & Mashtalir, S. (2020). Online robust fuzzy clustering of data with omissions using similarity measure of special type. In Lecture Notes in Computational Intelligence and Decision Making: Proceedings of the XV International Scientific Conference “Intellectual Systems of Decision Making and Problems of Computational Intelligence”(ISDMCI’2019), Ukraine, May 21–25, 2019 15 (pp. 637-646). Springer International Publishing.

43. Mashtalir, S. V., Stolbovoi, M. I., & Yakovlev, S. V. (2019). Hybrid Approach to Clustering Various Lengths Video. Journal of Automation and Information Sciences, 51(3).

44. Mashtalir, S., Mashtalir, V., & Stolbovyi, M. (2017). Video shot boundary detection via sequential clustering. International Journal “Information Theories and Applications, 24(1), 50-59.

45. Mashtalir, S., Mikhnova, O., & Stolbovyi, M. (2019). Multidimensional sequence clustering with adaptive iterative dynamic time warping. International Journal of Computing, 18(1), 53-59.

46. Mashtalir, S., Mikhnova, O., & Stolbovyi, M. (2018, August). Sequence matching for content-based video retrieval. In 2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP) (pp. 549-553). IEEE. doi: 10.1109/DSMP.2018.8478597.

47. Hu, Z., Mashtalir, S. V., Tyshchenko, O. K., & Stolbovyi, M. I. (2018). Clustering matrix sequences based on the iterative dynamic time deformation procedure. International Journal of Intelligent Systems and Applications, 10(7), 66-73. doi: 10.5815/ijisa.2018.07.0.

48. Core Data. URL: https://developer.apple.com/documentation/coredata/ (дата звернення 03.05.2023).

49. Mashtalir, S., Mashtalir, V., & Stolbovyi, M. (2018, August). Representative based clustering of long multivariate sequences with different lengths. In 2018 IEEE second international conference on Data Stream Mining & Processing (DSMP) (pp. 545-548). IEEE. doi: 10.1109/DSMP.2018.8478493.

50. Hu, Z., Mashtalir, S. V., Tyshchenko, O. K., & Stolbovyi, M. I. (2017). Video shots' matching via various length of multidimensional time sequences. International Journal of Intelligent Systems and Applications, 9(11), 10. doi: 10.5815/ijisa.2017.11.02.

51. Bilonoh, B., & Mashtalir, S. (2020, August). Parallel multi-head dot product attention for video summarization. In 2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP) (pp. 158-162). IEEE. doi: 10.1109/DSMP47368.2020.9204059.

52. Ye, B., Shafronenko, A., & Mashtalir, S. (2019). CORRUPTED DATA ON-LINE ROBUST FUZZY CLUSTERING BY SPECIAL TYPE SIMILARITY MEASURE. ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ ПРИЙНЯТТЯ РІШЕНЬ ТА ПРОБЛЕМИ ОБЧИСЛЮВАЛЬНОГО ІНТЕЛЕКТУ, 17.

53. Bodyanskiy, Y., Grimm, P., Mashtalir, S., & Vinarski, V. (2010). Fast training of neural networks for image compression. In Advances in Data Mining. Applications and Theoretical Aspects: 10th Industrial Conference, ICDM 2010, Berlin, Germany, July 12-14, 2010. Proceedings 10 (pp. 165-173). Springer Berlin Heidelberg.

54. Бодянський, Є. В., & Машталір, С. В. (2012). Виявлення змін у потоці відеоданих на основі аналізу багатовимірних часових рядів. Доповіді НАН України.

55. Mashtalir, S., Mikhnova, O., & Stolbovyi, M. (2019). Multidimensional sequence clustering with adaptive iterative dynamic time warping. International Journal of Computing, 18(1), 53-59.

56. Kinoshenko, D., Mashtalir, S., Shlyakhov, V., & Stolbovyi, M. (2019). Video shots retrieval with use of pivot points. In Advances in Computer Science for Engineering and Education 13 (pp. 102-111). Springer International Publishing.

57. Bilonoh, B., & Mashtalir, S. (2020, August). Parallel multi-head dot product attention for video summarization. In 2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP) (pp. 158-162). IEEE.

58. Kinoshenko, D., Kobylin, O., Mashtalir, S., & Stolbovyi, M. (2019, March). Metric video retrieval speedup by irrelevant data elimination. In Eleventh International Conference on Machine Vision (ICMV 2018) (Vol. 11041, pp. 176-183). SPIE.

59. Черноусова М. С. (2023) Оцінка методу регресійного пошуку проекції, 27-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. Матеріалів форуму. Т. 6, Ч. II. Харків: ХНУРЕ. 2023. (167 – 168).