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)

РОЗРОБЛЕННЯ ЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ ТА КЛАСИФІКАЦІЇ ОБРАЗІВ КУЛІНАРНИХ СТРАВ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-17-2
Темчур К.О.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Творошенко І.С.
(посада, прізвище, ініціали)

Допускається до захисту

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

2021 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)
«____»_____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Темчур Карині Олексіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення застосунку для розпізнавання та класифікації образів кулінарних страв

затверджена наказом університету від 20 травня 2021 року № 663СТ

2. Термін подання студентом роботи до екзаменаційної комісії 24 травня 2021 р.

3. Вихідні дані до роботи Науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, відкрита програмна бібліотека для машинного навчання TensorFlow, відкритий набір даних кулінарних страв food101 для навчання нейронної мережі, відкрита модель згорткової нейронної мережі Inceptionv3, JavaScript-бібліотека React для створення інтерфейсу користувача

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз існуючих застосунків для розпізнавання та класифікації образів кулінарних страв в Україні та за кордоном.
2. Розроблення підходу для розпізнавання та класифікації образів кулінарних страв.
3. Розроблення застосунку для розпізнавання та класифікації образів кулінарних страв.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми, мета роботи, постановка задачі, етапи виконання роботи, результати тестування, перспективи подальшої роботи

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

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	12.04.2021	
2	Аналіз завдання, підбір літератури	13.04.21-15.04.21	
3	Аналіз літератури з досліджуваної проблеми	16.04.21-17.04.21	
4	Аналіз підходів до розпізнавання та класифікації образів на зображенні	18.04.21-28.04.21	
5	Розробка методу розпізнавання та класифікації образів кулінарних страв	29.04.21-14.05.21	
6	Програмна реалізація	15.05.21-23.05.21	
7	Оформлення пояснювальної записки	24.05.21-26.05.21	
8	Перевірка на плагіат	27.05.21	
9	Рецензування	28.05.21	
10	Підготовка презентації та доповіді	29.05.21-30.05.21	
11	Занесення роботи в електронний архів	31.05.21	
12	Попередній захист кваліфікаційної роботи	31.05.21	

Дата видачі завдання 12 квітня 2021 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Творошенко І.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 81 с., 1 табл., 40 рис., 1 дод., 44 джерела.

КОМП'ЮТЕРНИЙ ЗІР, РОЗПІЗНАВАННЯ ТА КЛАСИФІКАЦІЯ ОБРАЗІВ, МАШИННЕ НАВЧАННЯ, ГЛИБОКЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ.

Об'єктом роботи є процес розроблення застосунку для розпізнавання та класифікації кулінарних страв.

Метою роботи є розроблення програмного застосунку із застосуванням методів глибокого машинного навчання, спрямованого на розпізнавання та класифікацію кулінарних страв.

Проведено дослідження класичних методів розпізнавання та класифікації образів та методів на основі нейронних мереж. Досліджено можливості бібліотеки глибокого навчання TensorFlow, особливості роботи CNN. Розроблено власний підхід до вирішення задачі. У результаті роботи отримано програмний застосунок для розпізнавання та класифікації кулінарних страв.

Результати роботи апробовано у вигляді 3 тез доповідей під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ», XV Регіональної студентської науково-технічної конференції «НАУКА – ПЕРШІ КРОКИ», XXVI Міжнародної науково-практичної конференції «Topical issues of practice and science».

COMPUTER VISION, IMAGE RECOGNITION AND CLASSIFICATION, MACHINE LEARING, DEEP LEARNING, NEURAL NETWORKS, CONVOLUTIONAL NEURAL NETWORKS.

The object of the research is the process of developing an application for the recognition and classification of culinary dishes.

The aim of the work is to develop a software application for recognizing and classifying culinary dishes.

Some methods of image recognition and classification were discovered: SURF, Kohonen map, KNN, neural networks-based methods. Discovered the possibilities of the TensorFlow deep learning library, CNN structure. Was developed own approach to solve the problem. As a result of work the software application for recognition and classification of culinary dishes is received.

The results of the work were tested in the form of 3 theses during the International Youth Forum «RADIOELECTRONICS AND YOUTH IN THE XXI CENTURY», XV Regional Student Scientific Conference «Science – FIRST STEPS», XXVI International Scientific Conference «Topical issues of practice and science».

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

КБЖВ – калорії, білки, жири, вуглеводи

CNN – convolutional neural network, згорткова нейронна мережа

RGBD – red, green, blue, depth

KNN – *k*-nearest neighbors

SURF – speeded up robust features

SPM – spatial pyramid matching

SOM – self-organized map, самоорганізаційна карта

WTA – winner-takes-all, переможець забирає все

ReLU – rectified linear unit

SGD – stochastic gradient descent

HSV – hue, saturation, value

ORB – oriented FAST and rotated BRIEF

BRIEF – binary robust independent elementary features

FAST – features from accelerated segment test

AKAZE – accelerated KAZE

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

ВСТУП

Вміння людини за допомогою зору знаходити та розпізнавати образи в житті, на фотографіях або навіть в абстракціях – дивовижне вміння, що формувалося тисячі років, та приголомшує ефективністю та, водночас, складністю процесів, що ховаються за цим вмінням.

Багато років вчені, що вивчають та вдосконалюють роботу обчислювальних систем таких, як комп'ютер, намагаються надати їм можливість перейняти деякі вміння, що притаманні людині. І якщо сам по собі комп'ютер не може «бачити» та розрізнювати об'єкти, то технології комп'ютерного зору вивчають створення штучних систем, що вже можуть вдало імітувати можливості зору людини.

Комп'ютерний зір – це досить молода галузь комп'ютерних наук, що вивчає інтелектуальну обробку та аналіз візуальних даних, таких як зображення та відео.

До типових задач комп'ютерного зору відносять пошук образів на зображенні, розпізнавання образів, їх класифікація, відновлення візуальних матеріалів або їх генерація та інші.

Системи комп'ютерного зору використовуються у багатьох галузях та повсякденному житті людей:

- у медицині;
- у промисловості;
- у військових цілях;
- для біометричної ідентифікації;
- для розпізнавання об'єктів (від людей до предметів);
- у доповненій реальності;
- у «розумних» пристроях;
- у кіно- та ігровій індустрії;
- у галузі ретейлу;

– у сільськогосподарській галузі і так далі.

Однією з найцікавіших сучасних задач, що намагаються вирішити дослідники – розпізнавання та класифікація їжі та кулінарних страв.

З сучасним темпом життя, з розповсюдженням сидячого способу життя, розмаїттю видів їжі з різним рівнем вмісту корисних речовин, постійним поспіхом, хворобами, що пов’язані з травленням та проблемою надлишку жиру в організмі, багато людей все частіше замислюються над тим, щоб контролювати те, що вони вживають. Але більшість систем, що використовуються для контролю за харчуванням, ґрунтуються на застарілому підході, коли всі дані потрібно власноруч шукати та заносити.

Наявність автоматизованої системи моніторингу та контролю харчування має важливе значення не тільки для лікування людей, що мають харчовий розлад та інші хвороби шлунку, а й для людей, що хочуть контролювати свою вагу та вести здоровий образ життя.

Підхід, коли користувач системи за контролем харчування повинен сам шукати та вносити потрібну страву – вважається неефективним через затрати часу. Це робить задачу розпізнавання та класифікації кулінарних страв актуальною та гідною активних досліджень.

Разом з розвитком області штучного інтелекту та машинного навчання, галузь комп’ютерного зору отримала новий «подих» розвитку та можливість значно збільшити свою ефективність та розмаїття сфер використання. Все більше задач, в яких необхідно використати системи комп’ютерного зору, використовують ті, що базуються на використанні нейронних мереж та глибокого навчання.

1 АНАЛІЗ ІСНУЮЧИХ ЗАСТОСУНКІВ ДЛЯ РОЗПІЗНАВАННЯ ТА КЛАСИФІКАЦІЇ ОБРАЗІВ КУЛІНАРНИХ СТРАВ В УКРАЇНІ ТА ЗА КОРДОНОМ

1.1 Сучасний стан розвитку застосунків для розпізнавання та класифікації образів кулінарних страв в Україні та за кордоном

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 не голодуванням та небезпечними тренуваннями, а завдяки правильному навантаженню та контролю за своєю їжею. Саме остання потреба стала причиною напливу мобільних застосунків для підрахунку калорій у Apple Store та Google Play – найпопулярніших платформах поширення мобільних сервісів.

Більшість сучасних інструментальних засобів, метою яких є підрахунок калорій споживаної користувачем їжі, мають загальний для даного виду мобільних застосунків потік робіт (рис. 1.1). Запропонований підхід використовує заздалегідь визначену базу даних страв та продуктів з усередненою калорійністю та має більше переваг, ніж старіші застосунки такого типу.

Перші мобільні програми для підрахунку калорій були й справді більше схожі на звичайний калькулятор: в них користувач повинен був самостійно вводити страву з усіма її характеристиками: калорійністю, КБЖВ (калорії, білки, жири, вуглеводи) та, звісно, вагою. І хоча сучасні застосунки для підрахунку калорій в усьому перевершують своїх попередників, в нинішньому ритмі життя людей і вони поступово почали втрачати свої переваги.

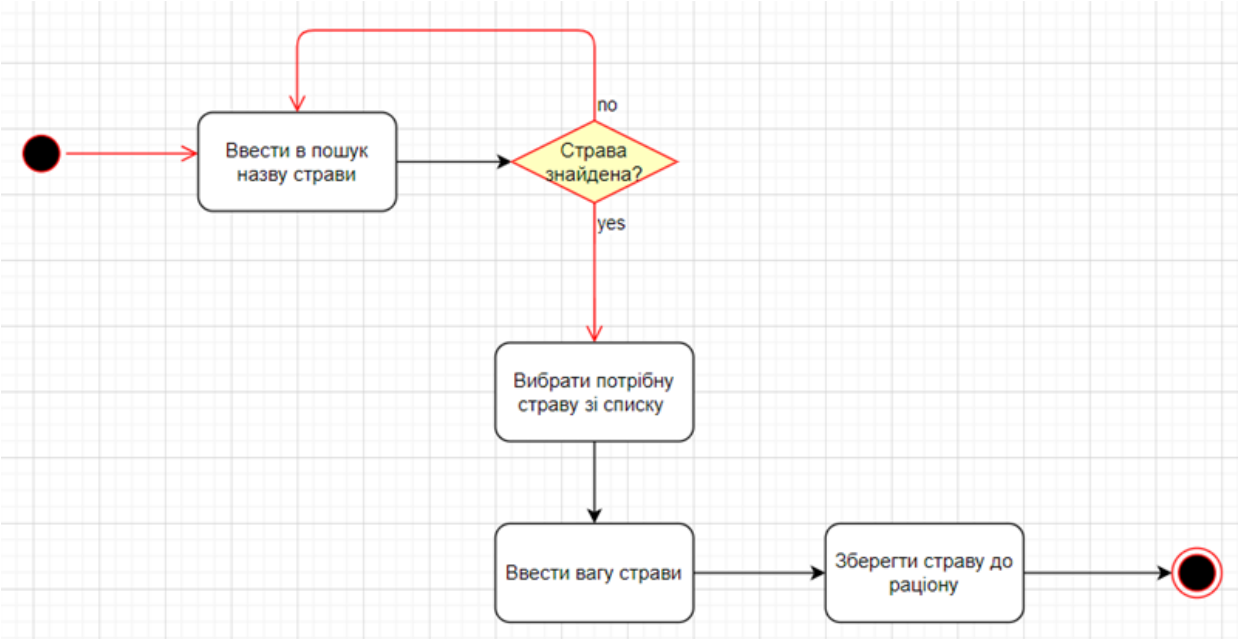

Рисунок 1.1 – Діаграма активності типового мобільного застосунку для підрахунку калорій

Найважливіший ресурс сучасної людини – це час. Найголовніша проблема, з якою стикаються люди, які прагнуть поліпшити своє харчування та почати моніторити свій щоденний раціон, – витрата часу та клопітність внесення страв до сервісів підрахунку калорій.

Саме для розв’язання цієї проблеми IT-розробниками почала вивчатися ідея інтеграції провідних технологій комп’ютерного зору, а саме розпізнавання та класифікації об’єктів (у нашому випадку – кулінарних страв) по зображенню.

Перші роботи, присвячені застосуванню комп’ютерного зору для розпізнавання їжі, почали поступово з’являтися, починаючи з 2010 року. Але через те, що більшість розроблених методів працювали лише в лабораторних умовах, де складові частини страви були добре відділені, кількість категорій їжі була невеликою, а сам підхід не дуже далеко відходив від звичайного традиційного розпізнавання [1]. У даному випадку складно оцінити практичну значущість та цінність зазначених досліджень.

Однак, у 2015 році команда інженерів з Google розробила систему, яка повинну була мінімізувати втручання користувача у звичний процес контролю денного раціону. Розробка отримала назву Im2Calories [1].

Система, розроблена командою, була побудована на використанні декількох методів глибокого машинного навчання, які були пристосовані до роботи на типових мобільних пристроях, що за характеристиками та продуктивністю значно поступаються комп'ютеру та ноутбуку.

Перша перевага, яку мала Im2Calories в порівнянні з іншими розробками, – можливість аналізувати харчові продукти в реальних умовах, коли компоненти страви не є чітко відокремленими один від одного.

Друга перевага – можливість розрахунку не тільки загальної калорійності страви, а ще визначення наявності білків, жирів та вуглеводів, що стало новим викликом в підході до вирішення початкового завдання.

Третя перевага – можливість розпізнавання не просто типу страви, а ще й розміру порції з підрахунком маси продукту. Такий функціонал дозволив вирішити ледь не найголовнішу проблему моніторингу раціону, з якою стикаються всі новачки, – виникнення суттєвих похибок при вимірюванні ваги їжі, яку користувач заносить до раціону. Вся суть підходу, що був запропонований командою інженерів Google, полягала в такому: для проведення дрібнодисперсної класифікації, проводилася локалізація об'єктів на зображенні для «витягування» чітких особливостей.

Метод, що використовувався в Im2Calories для локалізації продуктів на фотографії враховував властивість аморфності, тобто безформності, що характерна багатьом видам їжі. Таким чином, він виділяв харчовий продукт не чіткою рамкою, як це було прийнято для інших об'єктів задач розпізнавання, а прогнозував область, що могла бути притаманною для продукту.

Приклади роботи методу сегментування можна побачити на рисунку 1.2 [1].

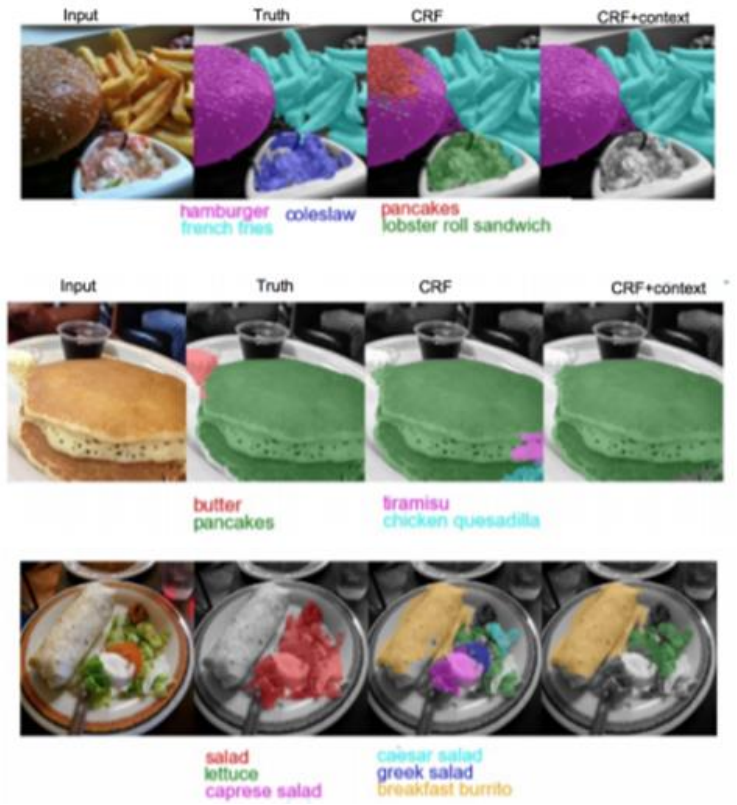

Рисунок 1.2 – Приклад роботи сегментуючого методу Im2Calories

Для того, щоб навчити нейронну мережу правильно сегментувати об’єкти на зображенні, був використаний спеціальний набір даних Food201-MultiLabel, що складався з 12 000 зображень [1]. Спеціальна група оцінювачів мала спочатку вручну сегментувати страви на фотографії. Результати такої сегментації були передані до вхідних даних нейронної мережі, що дозволило їй досить точно сегментувати їжу самостійно.

Щоб вирішити основну задачу – виміряти вагу страви – розробники спочатку розрахували висоту поверхні над тарілкою. Далі згорткова нейронна мережа проводила оцінку глибини кожного пікселю. Мережа CNN, тобто згорткова, була натренована на спеціальному тривимірному наборі даних, який складався з коротких RGBD відеороликів [2]. Загалом набір складається з 150 000 кадрів, де одна половина – оригінальні зображення, а інша – відповідні їм зображення глибини пікселів.

Слід також зазначити, що такий підхід працює однаково ефективно з різними розширеннями фотографії. Тобто немає різниці фото низької або високої якості, для всіх випадків результат буде приблизно однаковим.

Розрахувавши вагу страви, система отримує дані, які необхідні для остаточного розрахунку калорійності їжі та вмісту КБЖВ. Щоб правильно це визначити, потрібно мати спеціальний набір даних про поживні речовини. Im2Calories звертається до національної бази даних поживних речовин – USDA NNDB (це вільна інтегрована система даних, що надає інформацію про поживні речовини, факти про 8 618 основних продуктів харчування та сільськогосподарські дослідження [3]).

Однак, USDA NNDB більшою мірою зосереджена на «сирій» їжі, аніж приготовленій або обробленій якимось чином. Система хоч і надає дані про поживність багатьох видів м’яса, але цю інформацію складно використовувати для розрахунку калорійності тих же м’ясних стейків, бо в залежності від способу, який був використаний для приготування страви, варіюється й вміст калорій.

Таким чином, розробники Im2Calories доки не мають широкої бази даних, яка б могла покрити значну частину існуючих світових видів страв. Обмеження даних, призводить до найбільшої проблеми подібних систем – надто велике значення похибки під час процесу визначення калорійності страви. Розробники зазначають, що наразі під час тестування Im2Calories дуже часто неправильно визначає точну калорійність їжі. Розходження визначеного значення з реальними показниками сягає близько 20%, що є суттєвою похибкою.

Хоча результати роботи методів Im2Calories є проривними для задачі розпізнавання та класифікації кулінарних страв, команда інженерів Google, що займається створенням системи, прагне значно покращити її роботу.

Перш за все, необхідно вирішити такі задачі: максимально знизити значення похибки, покращити розпізнавання страв, що мають розмиті класові

характеристики, додати розпізнавання методів обробки їжі (смаження, варіння, запікання).

Важливим також є завдання модифікувати сегментування та оцінку глибини, зменшити час аналізу та покращити розпізнавання в цілому. Частина цих проблем вже вирішена певною мірою, але попереду команду розробки чекає ще багато випробувань.

Тим не менш, треба зазначити, що рішення, які є на даний момент – це великий крок уперед. Аналізуючи всі викладені факти, а також враховуючи те, що Im2Calories вже запатентована система, слід зазначити, що її повноцінний вихід у якості готового мобільного рішення ще далека перспектива.

Іншою перспективною зарубіжною розробкою в вирішенні проблеми розпізнавання та класифікації кулінарних страв є FoodAI – нейронна мережа від спеціалістів SalesForce.

FoodAI – це досить «молода» розробка, що була представлена в 2019 році. Застосунок розроблено в Сінгапурі, тому його основна орієнтація на продукти харчування, які, зазвичай, вживають саме там [4].

Модель роботи FoodAI побудована на SENet та ResNeXt архітектурах.

SENet – це так звані Squeeze-and-Excitation Networks, набір блоків «стиснення та збудження» для CNN мереж, що значно покращують продуктивність згорткових нейромереж та збільшують точність підрахунків [5].

ResNeXt, в свою чергу, – це також мережева архітектура, яка була створена командою дослідників AI компанії Facebook, вона спеціалізується на побудові рішень класифікації зображення [6].

Система FoodAI також, як і Im2Calories, базується на згорткових нейронних мережах. Вона навчалася на 400 000 зображень, серед яких було виділено 756 категорій класифікації страв, 100 з яких відносилися до національних страв Сінгапуру [4].

На початку розробки даної системи розробники використовували моделі, що вже були попередньо навчені на ImageNet – наборі даних

величезного проекту, створеного для використання в застосунках розпізнавання образів – та налаштовані на класифікацію обраних категорій.

Це дало значно більше продуктивності, ніж підхід навчання моделей з нуля. Далі вже використовувався підхід ResNeXt та SENet архітектура.

Однак, у процесі розробки, команда дослідників виявила вкрай важливу проблему у вигляді дисбалансу класів у наборах даних. У даному випадку слід зазначити різницю між кількістю даних для деяких класів, що призвело до зменшення ефективності та продуктивності на розпізнаванні та класифікації тих представників класів, що мали менше представництво даних.

Тобто система була більш ефективною при розпізнаванні таких продуктів, що відносилися до класів, що мали велику кількість зображень при тренуванні. Навіть, якщо сам по собі продукт не був складним для розпізнавання (як, наприклад, яблуко), система розпізнавала його повільніше та з більшою похибкою за умови, що він належав до класу з невеликою кількістю даних.

Вирішуючи цю проблему, розробники модифікували традиційний підхід та модифікували використання перехресної ентропії так, щоб врівноважити вплив даних з різних класів та змінити фокусування.

Проводилося регулювання, у ході якого оцінювалася легкість класифікації. Якщо вхідний еталон було легко класифікувати, важливість його класу при подальшій класифікації зменшувалася. Якщо ні – система акцентувала увагу нейронної мережі на класі еталону зі складнішим процесом розпізнаванням.

У порівнянні з Im2Calories, FoodAI має менший функціонал та вирішує не всі проблеми процесу моніторингу денного раціону. Так, наприклад, система не підраховує сама вагу страви, як це було в Im2Calories, та також має доволі велику похибку на стравах, що мають розмиті класові характеристики.

Тим не менш, система вже інтегрована як API в мобільний застосунок Healthy 365 та вже намагається калібрувати дані на реальних кейсах. Також перевагою сервісу є його кросплатформеність: FoodAI можна

використовувати у мобільному застосунку, у вебсервісі та на десктопі. Повна архітектура системи показана на рисунку 1.3 [4].

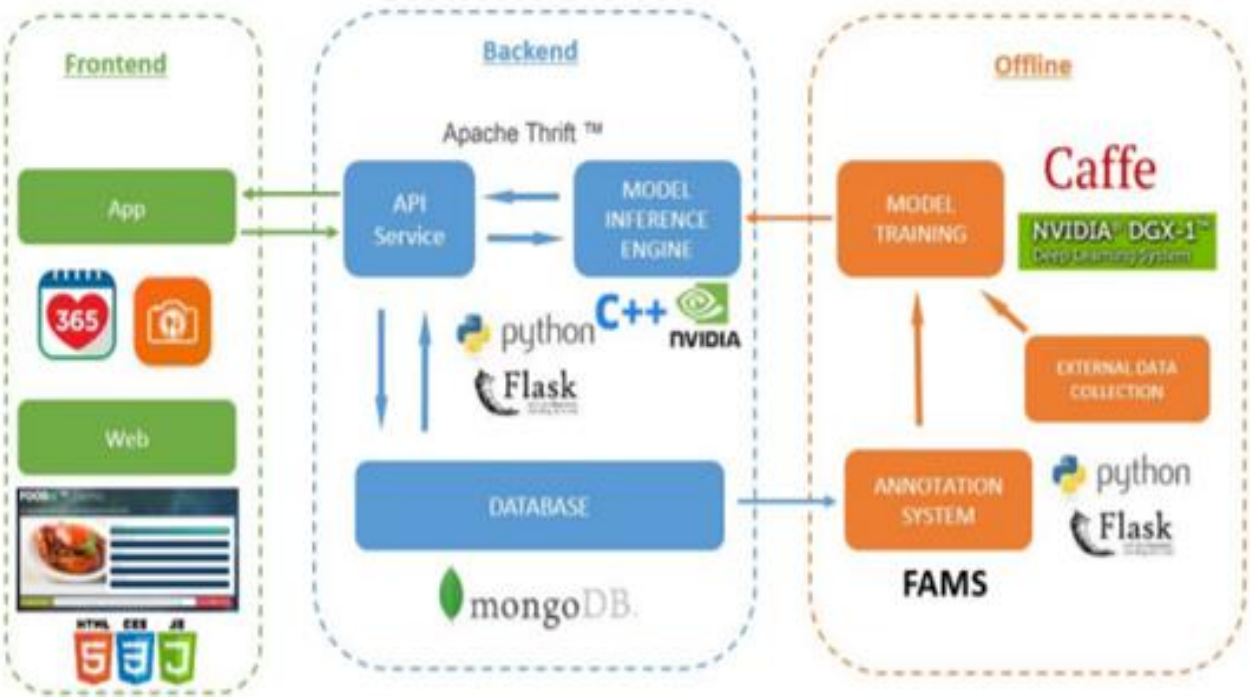

Рисунок 1.3 – Структура системи FoodAI як API сервісу

Крім Im2Calories та FoodAI, існує ще багато закордонний розробок, що продовжують розвивати тему розпізнавання та класифікації кулінарних страв за фотографією. У всіх них також використовуються згорткові нейронні мережі та навчання на наборах даних.

Серед таких систем можна виділити розробку під назвою FoodTracker від канадської команди (рис. 1.4) [7] та сервіс Snap It від массачусетських розробників (рис. 1.5) [7].

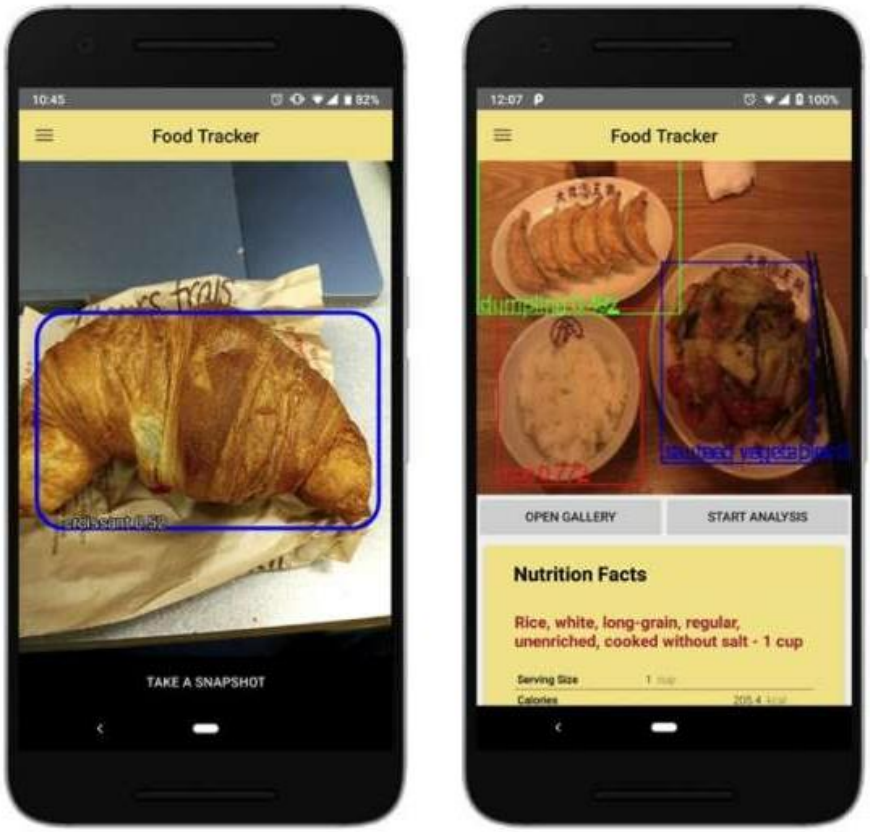

Рисунок 1.4 – Екранні форми застосунку FoodTracker

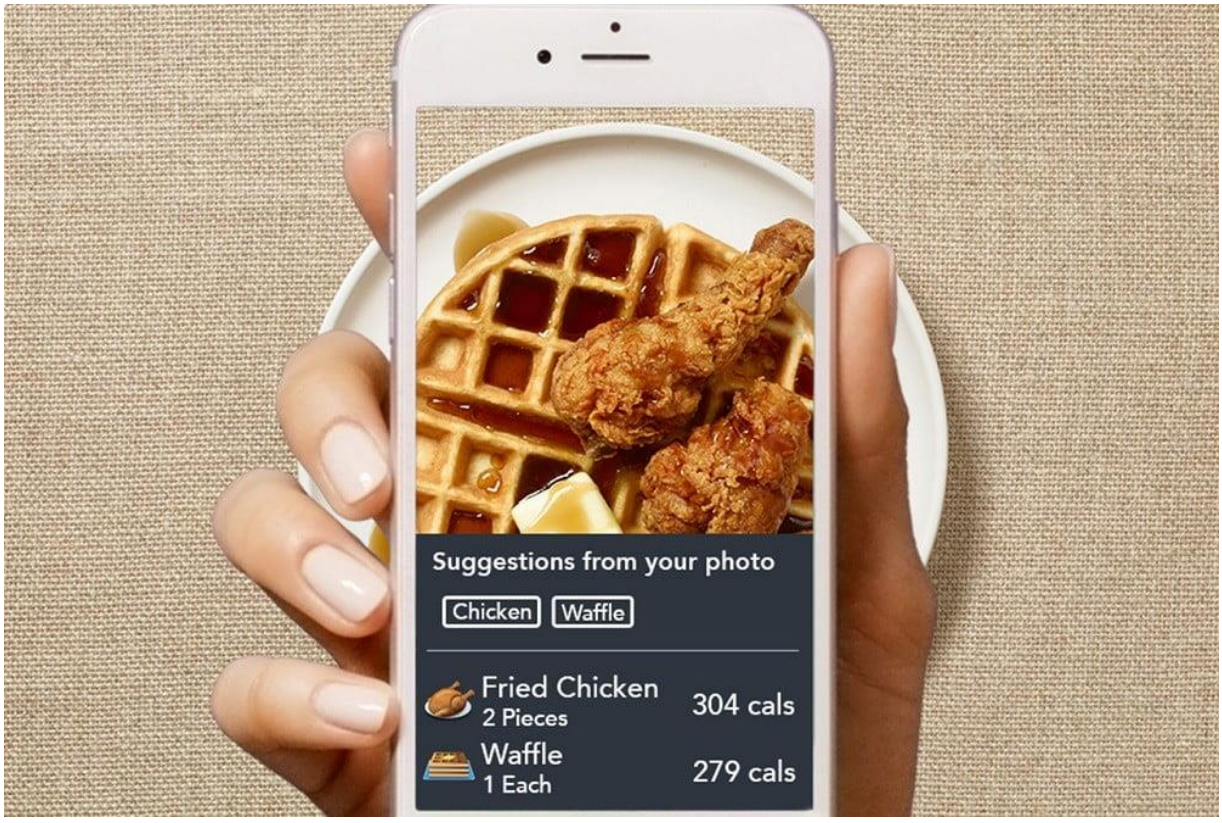

Рисунок 1.5 – Екранна форма застосунку Snap It

Що стосується українських досягнень у сфері рішень задач розпізнавання та класифікації їжі, то поки що можна сказати, що сама сфера комп'ютерного зору ще тільки розвивається в Україні, і увага розробників націлена на такі задачі, як: розпізнавання обличь, розпізнавання відбитків пальців, задачі розпізнавання для промислових, аграрний та суміжних цілей. Поки що в широкому доступі немає інформації щодо активних українських досліджень проблем розпізнавання та класифікації кулінарних страв. Але, аналізуючи темпи розвитку комп'ютерного зору в українській ІТ-сфері, можна стверджувати, що такі дослідження можуть з'явитися досить скоро.

1.2 Аналіз літературних джерел щодо існуючих підходів розпізнавання та класифікації образів кулінарних страв

З початком існування та розвитку комп'ютерного зору як наукової дисципліни, було винайдено велику кількість різноманітних методів, технологій та підходів до задач розпізнавання та класифікації образів. В залежності від об'єкту, який необхідно розпізнати, різні типи методів можуть давати різний відсоток точності та ефективності на одних і тих самих даних.

Як і в інших сферах, в задачах розпізнавання образів немає єдиного універсального методу, що міг би бути однаково ефективним у всіх випадках. Розпізнавання їжі та, наприклад, обличь – це розпізнавання принципово різних об'єктів, з різними особливостями та характеристиками, що потребує особливого підходу в обох випадках. Так само навіть для об'єктів одного класу можуть використовуватися різні методи розпізнавання та класифікації.

Наприклад, для розпізнавання та класифікації фруктів, що є досить великим підкласом їжі, дослідники університету Малайя, у своїй роботі пропонують використовувати метод KNN (*K*-Nearest Neighbors) або метод *k*-найближчих сусідів [8].

У своєму методі дослідники також використали стратегію розпізнавання на основі форми та генерацію примітивів для оцінки параметрів.

На початку над зображенням було проведено адаптивне згладжування для фільтрації гаусівського адитивного шуму. Далі проводиться генерація примітивів для проведення оцінки. Метод також враховує недозрілі фрукти, які на початку дозрівання можуть сильно відрізнятися за формою від стиглого зразка. Точність розпізнавання та класифікації для таких прикладів за даними дослідників складає близько 80%.

Для проведення оцінки та подальшої класифікації знаходять площу та периметр фрукту. Їх значення розраховуються завдяки пікселям зображення. Периметр фрукту розраховують шляхом підрахунку граничних пікселів, а площа оцінюється підрахунком загальної кількості пікселів, що знаходяться всередині знайдених границь. При цьому, метод не розпізнає різницю в розмірах фотографій але пропонує користувачу підігнати їх самостійно, щоб зразки мали приблизно однакові розміри на фото. Тобто, якщо є фото яблука десь далеко на дереві та фото яблука поблизу, фотографії потрібно змінити так, щоб об’єкти були приблизно однаковими за розміром. Тільки після цього система зробить розрахунки площі та периметри, які будуть використовуватися при виконанні класифікації.

Як було зазначено вище, система Fruits Recognition використовує метод KNN або метод *k*-найближчих сусідів. Цей метод широко використовується в задачах класифікації [9].

Для системи розпізнавання фруктів, метод KNN виконує класифікацію фруктів з використанням міри відстані, що є метрикою для евклідової відстані для вимірювання відстані між атрибутами фрукту, що поки невідомий системі, з прикладами фруктів, що вже збережені в ній. Метод вибирає найближчий до невідомого фрукту зразок.

Якщо детальніше розглянути роботу методу, в якому використовується KNN, то дослідники описують його структуру певним чином.

Fruit_Class = knnclassify (sample, training, group, *k*) [8].

На вхід до функції подається вхідний зразок фрукту. Далі визначається відстань між атрибутами вхідного зразка з атрибутами інших фруктів та знаходяться *k*-найближчі приклади.

Невідомий фрукт згодом відноситься функцією до певного класу або групи, з якої походить його *k*-найближчий сусід [10].

В загалом, у своїй роботі малайзійські дослідники пропонують таку послідовність дій [8]:

- вибрати зображення фрукту;
- обрізати область фрукту;
- розрахувати RGB та проаналізувати текстуру;
- видалити шуми, провести морфологічні операції;
- розрахувати геометричні властивості (площу та периметр);
- використати KNN метод;
- отримати вихідний результат.

Запропонований дослідниками з малайського університету підхід дійсно дуже ефективно показує себе в експериментах, видаючи до 90% точності при розпізнаванні та класифікації фруктів [8].

Але його ефективність та точність з більшою кількістю груп їжі значно знизиться через велику міжкласову подібність та внутрікласові варіації.

Ця проблема описується в роботі іспанських дослідників з Барселонського університету [11].

Крім цього існує багато інших методів, що базується на виявлені ключових точок. Серед них BRIEF, ORB, BRISK, FREAK, AKAZE, LATCH, SIFT, SURF [9]. Серед них найбільшу популярність здобув SURF.

SURF або Speeded Up Robust Features – метод, що широко використовується для порівняння зображень, пошуку та класифікації об’єктів на фотографії та інших задач [12].

SURF добре підходить для розпізнавання складних об’єктів з яскраво вираженою текстурою, якими і є кулінарні страви.

У роботах [12, 13] українських дослідників вивчається ефективність методу SURF з використанням апарату нечіткої логіки.

В даному випадку, нечітка логіка застосовується для визначення ступені приналежності об’єкту або унікальної ключової точки до еталонного класу [14].

Використання SURF як методу розпізнавання об’єктів у деяких випадках може призвести до значних затрат часу через велику кількість обраних ключових точок.

Для мобільного застосунку – час відпрацювання системи може стати критичним, але використання підходу нечіткої логіки може допомогти уникнути проблеми збільшення часу обробки [15].

Метод SURF з використанням нечіткої логіки показав гарні результати на розпізнаванні обличь [12], але важко сказати чи покаже він такі ж результати при розпізнаванні їжі, особливо на дуже великій вибірці.

З іншого боку, дослідники зі Стенфордського університету також пропонують використовувати метод SURF, але в поєднанні з його іншою реалізацією під назвою Bag-of-SURF та Spatial Pyramid Matching методом [16]. Схема роботи методу надана на рисунку 1.6 [16].

За допомогою Bag-of-SURF проводиться створення словнику кодових слів, витягуються SURF вектори ознак зображень, будуються дескриптори в 64-мірному просторі и згруповуються у 100, 200 або 300 кластерів за допомогою вже відомого нам методу *k*-середніх.

Метод SPM (Spatial Pyramid Matching) використовується для сегментування зображення на суб-області та побудови гістограми локальних особливостей.

Метод стенфордських дослідників виявився ефективним (86%) на невеликому наборі даних (6 категорій, 53 приклади страв), але на великій вибірці (111 категорій, 2145 зображень) точність методу зменшилася в декілька разів [16].

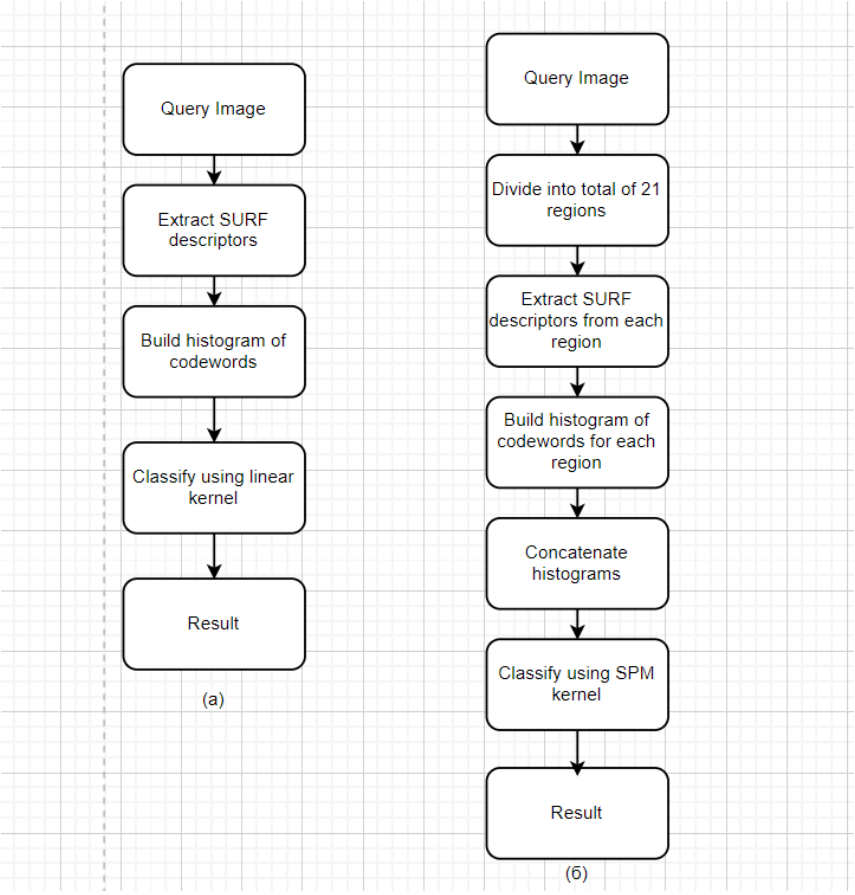

Рисунок 1.6 – Розпізнавання з використанням:
(а) Bag-of-SURF; (б) Spatial Pyramid Matching

В останні роки, дослідники сходяться на думці, що для розпізнавання та класифікації кулінарних страв краще за все підходять методи з використанням нейронних мереж та методів машинного або глибокого навчання [17].

Існує багато різновидів нейронних мереж, деякі з котрих активно використовуються для вирішення задач розпізнавання образів.

У роботі українських дослідників розглядається використання нейронної мережі Кохонена для задачі розпізнавання та класифікації [18]. Особливості топології цієї мережі дозволяють зберігати подібність вхідних зображень, що вкрай важливо при класифікації [18].

Однак більшість дослідників віддають перевагу використанню згорткових нейронних мереж або CNN. CNN успішно використали у своїх дослідженнях розпізнавання та класифікації кулінарних страв японські

дослідники [19], міланські дослідники [20], сінгапурські дослідники, які розробили програмний застосунок FoodAI [4] та навіть дослідники з Google, що розробляють надскладну систему Im2Calories [1].

Хоча використання CNN мереж вимагає значних обчислювальних потужностей та потужну техніку (особливо на вкрай великих вибірках даних), але за висновками різних спеціалістів, для задачі розпізнавання та класифікації кулінарних страв поки що цей підхід є найперспективнішим, що підтверджується результатами практичних досліджень [1, 3, 19, 20].

1.3 Постановка задачі

Актуальність теми зумовлена тим, що з сучасним темпом життя, з розповсюдженням сидячого способу життя, розмаїттю видів їжі з різним рівнем вмісту корисних речовин, хворобами, що пов’язані з травленням та проблемою надлишку жиру в організмі, багато людей все частіше замислюються над тим, щоб контролювати те, що вони вживають.

Але сучасні системи моніторингу надають застарілий функціонал для контролю, при якому користувач власноруч повинен шукати страву або вносити її до системи.

Для того, щоб зробити процес моніторингу за процесом харчування зручнішим та менш затратним за часом, необхідно розробити підхід до автоматичного розпізнавання та класифікації кулінарних страв.

Об’єктом роботи є процес розроблення застосунку для розпізнавання та класифікації кулінарних страв.

Метою роботи є розроблення програмного застосунку із застосуванням методів глибокого машинного навчання, спрямованого на розпізнавання та класифікацію кулінарних страв.

Враховуючи мету роботи необхідно вирішити такі завдання:

– проаналізувати сучасний стан розвитку застосунків для розпізнавання та класифікації образів кулінарних страв в Україні та за кордоном;

– проаналізувати літературні джерела щодо існуючих підходів розпізнавання та класифікації образів кулінарних страв;

– проаналізувати методи розпізнавання та класифікації образів на зображенні;

– розглянути методи глибокого навчання бібліотеки TensorFlow для розпізнавання та класифікації образів;

– дослідити проблему вибору інформаційних ознак на зображенні;

– розробити методику розпізнавання та класифікації образів кулінарних страв;

– провести вибір інструментальних засобів для створення застосунку;

– дати детальний опис етапів розробки застосунку для розпізнавання та класифікації образів кулінарних страв;

– провести тестування розробленого застосунку та провести аналіз результатів;

– виявити перспективи подальшої роботи;

– зробити висновки щодо виконаної роботи.

2 РОЗРОБЛЕННЯ ПІДХОДУ ДЛЯ РОЗПІЗНАВАННЯ ТА КЛАСИФІКАЦІЇ ОБРАЗІВ КУЛІНАРНИХ СТРАВ

2.1 Методи розпізнавання та класифікації образів 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 [21].

Якщо говорити про розпізнавання їжі та кулінарних страв, то, як вже було зазначено до цього, найбільша проблема полягає в високій міжкласовій подібності та широкому різноманітті представників одного класу. Саме тому, багато традиційних підходів до розпізнавання вважаються неефективними для цього типу задач.

Але можна визначити декілька методів, що були успішно застосовані дослідниками питання розпізнавання їжі, та показали себе ефективними для подальшого вивчення проблематики.

Перший метод, що вважається досить класичним для задач комп’ютерного зору, – це Speeded Up Robust Features метод або інакше – SURF.

SURF – один з найефективніших методів комп’ютерного зору, що частково бере за основу метод SIFT (Scale-Invariant Feature Transform), і використовується як локальний детектор дескрипторів ознак [22].

Дескриптор – це ідентифікатор ключової точки, що виділяє її серед інших. Це деякий набір даних, що описує сутність, об’єкт або іншу одиницю, але не є самою сутністю [23]. В реальному житті під опис дескриптору підходять, наприклад, документи, що позначаються особу – вони надають опис людини але не є нею.

Зазвичай на зображенні дескриптор описує інформацію про ключову точку та її оточення. Особливістю дескрипторів SURF є властивість інваріантності, що була присутня й в методі SIFT.

Маленькі області зазнають менше впливу від деформації (зміна розміру зображення, поворот, інші трансформації). Щоб бути ефективним, дескриптор повинен мати властивість інваріантності та берегти опис ключової точки однаковим при будь-якому розмірі, повороті та іншої зміни. Як саме будуються дескриптори в SURF та як працює сам алгоритм розглянемо далі.

Для розуміння роботи методу необхідно ввести поняття матриці Гессе та Гессіану.

Матриця Гессе – це квадратна матриця, що визначається формулою, наведеною далі

$$H\big(f(x,y)\big) = \begin{vmatrix} \frac{\delta^2 f}{\delta x^2} & \frac{\delta^2 f}{\delta x \delta y} \\ \frac{\delta^2 f}{\delta x \delta y} & \frac{\delta^2 f}{\delta y^2} \end{vmatrix}, \tag{2.1}$$

де H – матриця Гессе;

$f(x, y)$ – функція зміни градієнту яскравості [22].

Гессіан – це визначник (детермінант) матриці Гессе, використовується при формуванні дескриптора та працює з яскравістю градієнту. Гессіан визначається за формулою, наведеною далі

$$det\ (H)\ = \frac{\partial^2 f}{\partial x^2} \cdot \frac{\partial^2 f}{\partial y^2} - \frac{(\partial^2 f)^2}{(\partial x \partial y)^2}. \tag{2.2}$$

Саме за допомогою матриці Гессе та Гессіану SURF визначає ключові точки. Особливі місця, такі як кути, границі особливості текстури та ін., визначаються через Гессіан, який досягає свого екстремуму в місцях, що мають максимальні зміни градієнту яскравості.

Для розрахування яскравості використовують інтегральне представлення зображення. Воно представляє собою матрицю, розмір якої відповідає розміру зображення. Елементи матриці при цьому обчислюються за спеціальною формулою

$$II(x, y) = \sum_{i=0, j=0}^{i \leq x, j \leq y} I(i, j), \quad (2.3)$$

де $I(i, j)$ – яскравість пікселю.

Для того, щоб матриця Гессе мала інваріантність не тільки відносно обертання зображення, а й відносно масштабу у багатократних розмірах, у методі SURF використовуються фільтри різних масштабів. Формула обчислення Гессіану в SURF з урахуванням фільтрів різного масштабу наведена далі

$$det\,(H_{approx}) = DD_{xx}DD_{yy} - (0.9 \cdot D_{xy})^2, \quad (2.4)$$

де DD_{xx}, DD_{yy}, D_{xy} – згортки по фільтрах;

0,9 – коефіцієнт, що теоретично обґрунтований та використовується для коригування розрахунків.

У методі SURF задається порогове значення Гессіана. Шукаючи ключові точки, метод проходить по пікселям і шукає максимум визначника матриці Гессе. Якщо значення пікселю вище, ніж заданий поріг, тоді знайдений піксель розглядається у якості кандидату до ключової точки.

Після того, як ключова точка була виділена, метод формує на її основі дескриптори. Дескриптор в SURF – це набір із 64 чисел (інколи 128) для кожної ключової точки. Кожне число – різниця градієнту навколо ключової точки (сама вона має максимальне значення). Інваріантність дескриптора відносно повороту розраховується, як випадкове відхилення величини градієнту від середнього значення в радіусі ключової точки та відносно

напрямку градієнту. Область обчислення дескрипторів визначається розміром матриці Гессе.

Зміна рівня освітлення також на впливає на пошук ключових точок через інваріантність відносно зміщенню яскравості. Метод SURF ефективно знаходить як темні, так і світлі особливості на зображенні.

Загалом весь процес пошуку ключових точок на зображенні та формування дескрипторів на їх основі приведено на рисунку 2.1.

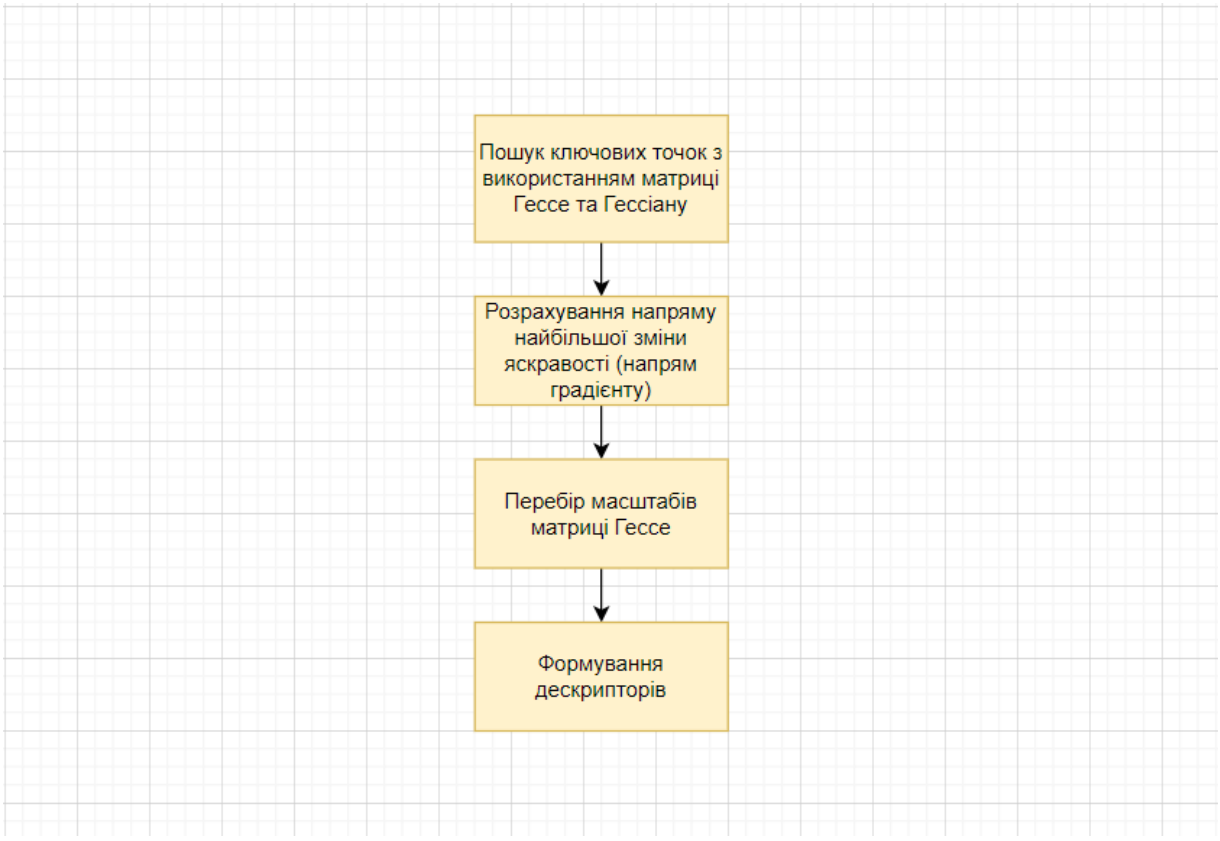

Рисунок 2.1 – Фази пошуку ключових точок та формування дескрипторів методу SURF

Більш детальні кроки побудови дескриптору приведені на рисунку 2.2.

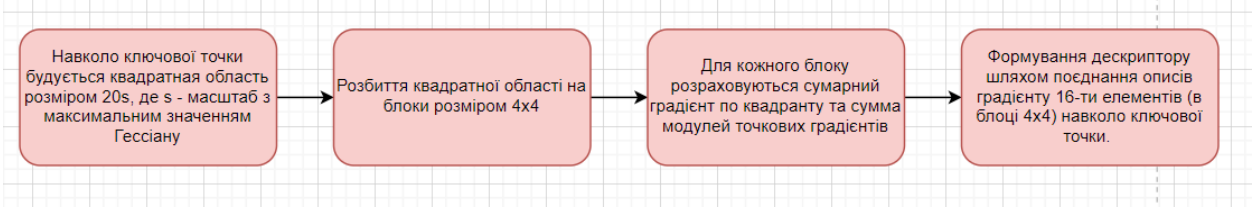

Рисунок 2.2 – Кроки побудови дескриптору SURF

SURF є дуже ефективним та швидким (на відміну від методу SIFT) методом пошуку ключових ознак на зображенні. Він добре підходить для пошуку об’єктів на зображенні, але сам по собі з об’єктом він не працює. Метод аналізує зображення в цілому і таким чином шукає особливості. SURF добре розпізнає складні об’єкти зі складною текстурою, але неефективно розпізнає прості об’єкти з однорідною, простою текстурою [9]. Методу складно знаходити ключові точки у таких об’єктів.

Але властивість інваріантності у різних випадках, ефективність роботи зі складними об’єктами та швидкість методу є важливими перевагами SURF як методу розпізнавання.

Останні роки серед багатьох методів, що використовуються для розпізнавання та класифікації зображень, сильно виділяються методи, що використовують можливості машинного навчання та нейронних мереж [24].

Нейронна мережа – це математична модель, представлена у програмному вигляді, що була заснована на принципах роботи реальних біологічних нейронів мозку. Особливість нейронних мереж полягає в можливості «навчатися» різним задачам, шляхом аналізу різних прикладів.

Використання нейронних мереж для розпізнавання зображень – досить ефективний та популярний метод.

Етапи створення й роботи нейронної мережі для розпізнавання виглядають наступним чином (рис. 2.3).

Існує багато видів нейронних мереж, що відрізняються і топологією, і характером навчання та зв’язків. Для вирішення задачі розпізнавання розглянемо 2 види нейронних мереж, що активно використовуються в задачах комп’ютерного зору.

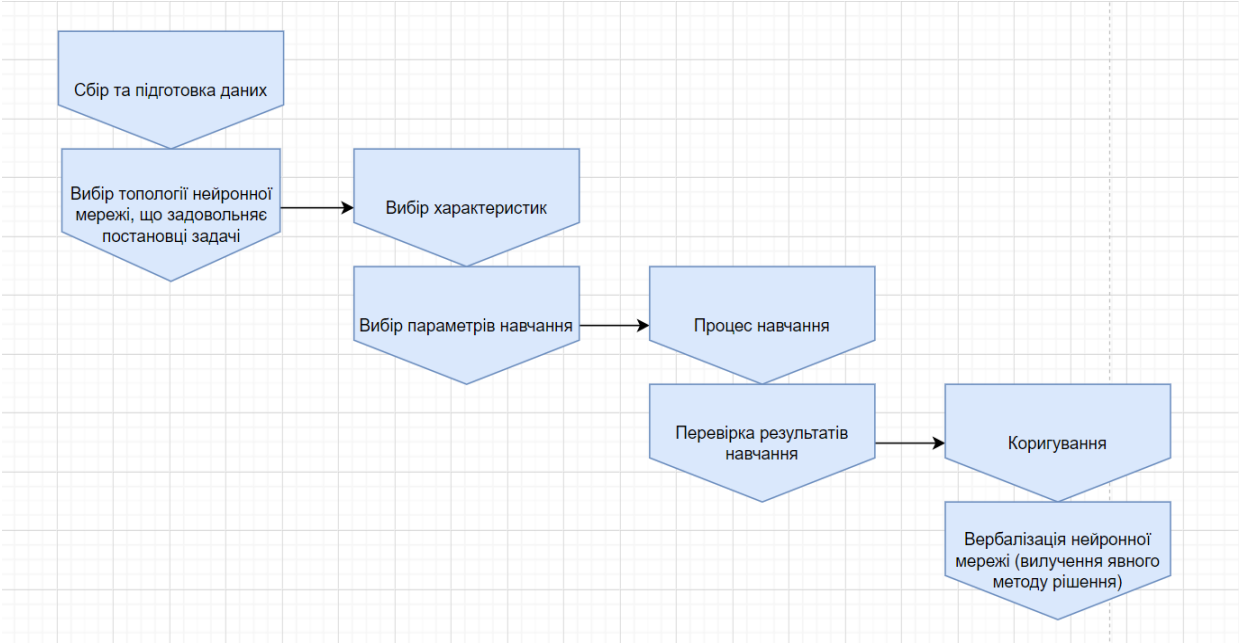

Рисунок 2.3 – Етапи створення та роботи нейронної мережі

Перший вид такої нейронної мережі – мережа Кохонена.

Нейронні мережі Кохонена – це сукупність нейронних мереж, в основі яких лежить структура шару Кохонена [25]. Найвідомішими реалізаціями мереж Кохонена є класична (або базова) версія та Самоорганізаційна Карта Кохонена. Для задачі розпізнавання та класифікації зображень найчастіше використовується друга версію мережі Кохонена [26].

Самоорганізаційна Карта Кохонена, SOM або Self-Organized Map – це спеціальна конкурентоспроможна нейронна мережа з некерованим процесом навчання. У картах Кохонена кількість вхідних елементів співпадає з розміром вхідних векторів, а кожен вихідний елемент відповідає певному кластеру [27].

Наприклад, якщо об’єкт аналізується за трьома ознаками, то мережа буде мати ти входи, що відповідають заданим ознаками. Вхідним вектором буде об’єкт чиї ознаки аналізуються.

Гарною практикою є задання меншої кількість вихідних елементів, ніж вхідних. Це дозволяє отримувати спрощений опис об’єктів, що в подальшому прискорює та полегшує роботу з ними.

Структура мережі при цьому доволі проста: кожен вхідний елемент з'єднується з кожним вихідним. Зв'язки між входом та виходом мають свою вагу, що коригується в процесі навчання.

Для того, щоб класифікувати вектор та віднести до певної групи або кластеру, мережа повинна визначити який кластер є найближчим до вхідного вектору. Критерієм близькості в мережі Кохонена слугує критерій мінімальності квадрату евклідової відстані.

Евклідова відстань – це відстань між двома точками в евклідовому просторі.

В нашому випадку вхідний вектор буде точкою в *n*-вимірному просторі. Потрібно знайти відстань між цією точкою та центрами існуючих кластерів.

Кластер, відстань до якого буде мінімальним, буде об'явлений переможцем. Саме тому метод, за яким обробляються вихідні сигнали в мережі Кохонена, називається методом за правилом WTA або «winner-takes-all» [25].

При цьому оскільки вихідний нейрон з'єднаний з кожним вхідним, то маємо *n* зв'язків та *n* координат точки центру кластеру. Тобто координатами центру кластеру є значення ваги усіх зв'язків, що йдуть до вихідного нейрону.

Таким чином, критерій близькості в мережі Кохонена, квадрат евклідової відстані, обчислюється за наступною формулою

$$d_{pk} = \sum_{i=0}^{n}(x_{pi} - x_{ki})^2, \tag{2.5}$$

де d_{pk} – квадрат відстані між точкою *P* та кластером *K*;

x_{pi} – координати точки *P*;

x_{ki} – координати кластеру *K*.

Але більш цікавим є процес навчання мережі Кохонена. Як вже було зазначено вище, коли на вхід до мережі подається вхідний вектор, за допомогою евклідової відстані визначається кластер, що переміг. Далі для

кластеру-переможця проводиться коригування вагових коефіцієнтів. Коригування проводиться за спеціальною формулою

$$\omega_{ij}(t + 1) = \omega_{ij}(t) + \eta(t)(x_i - \omega_{ij}(t)), \quad (2.6)$$

де $\omega_{ij}(t + 1)$ – вага на ітерації t+1;

$\omega_{ij}(t)$ – вага на попередньому кроці;

η – норма навчання;

x_i – координати вхідного вектору.

Мережа Кохонена навчається без учителя. Це означає, що йде не порівняння вихідного сигналу з еталонним, а налаштуванні ваги зв'язків під максимальне співпадіння з вхідними даними.

Крім того, норма навчання не є постійним значенням та змінюється впродовж усього процесу навчання та залежить від кроку.

Найпоширенішим підходом при навчанні самоорганізаційної мережі Кохонена є коригування ваги певної кількості найближчих сусідів кластеру, що переміг.

Скількох саме сусідів торкнуться зміни визначає величина радіусу, що також змінюється в процесі навчання, поступово зменшуючись на кінцевих етапах навчання.

Схематично процес зміни радіусу та сусідів, чия вага буде скоригованою, можна уявити так, як показано на рисунку 2.4.

Червоним кольором позначено сигнал-переможець, зеленим – радіус.

Загалом, процес роботи мережі Кохонена показано на рисунку 2.5.

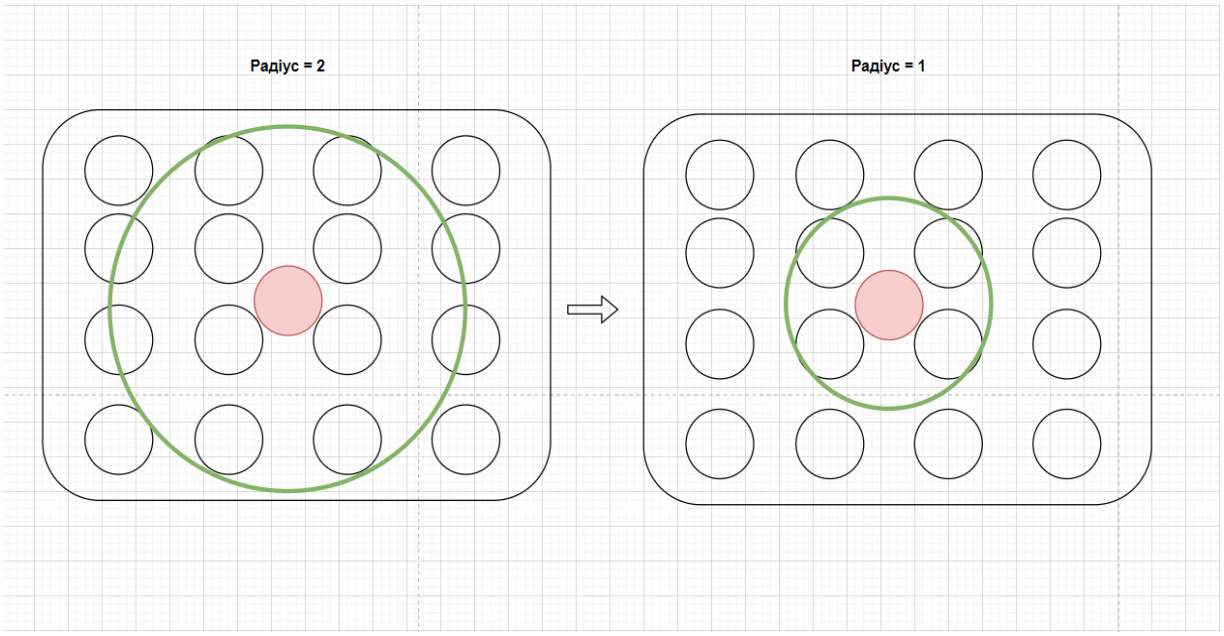

Рисунок 2.4 – Зменшення радіусу та сусідів, чия вага буде скоригована у процесі навчання мережі Кохонена

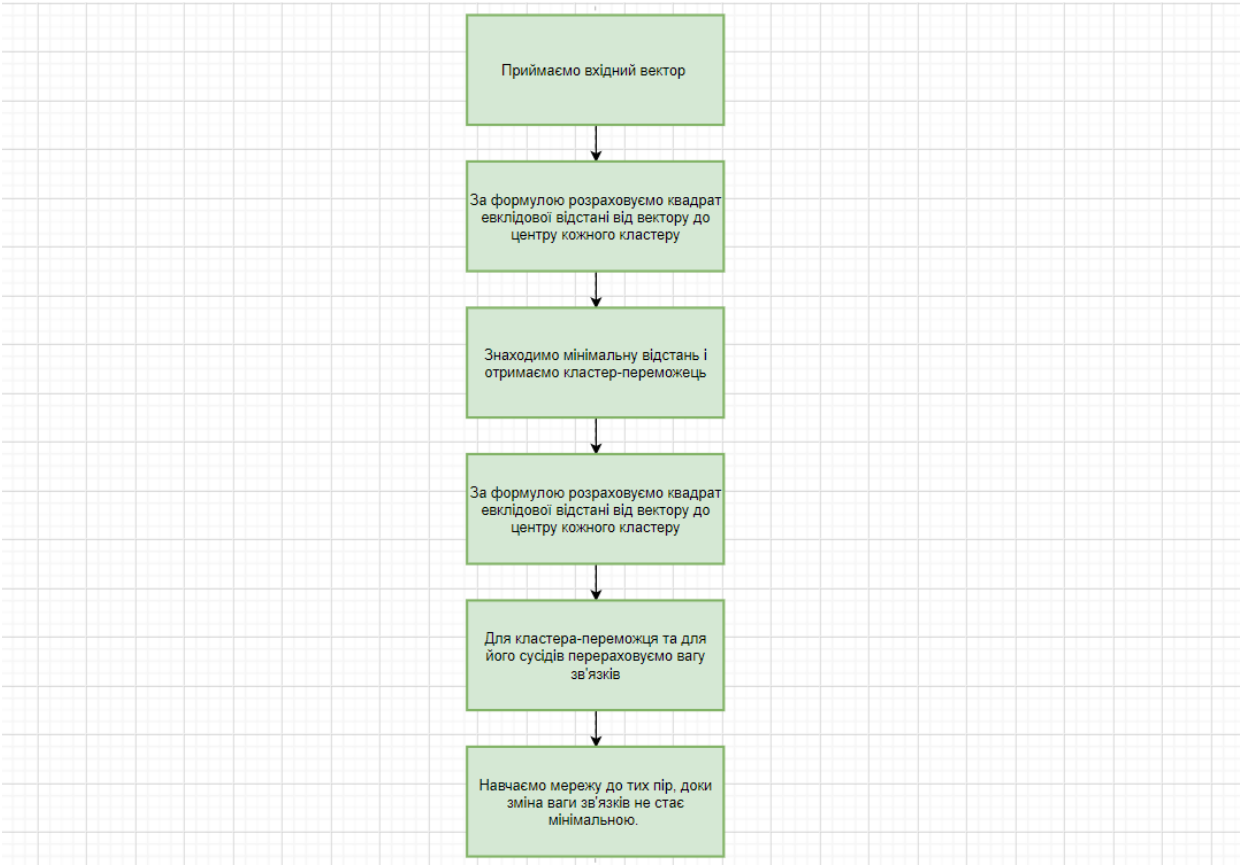

Рисунок 2.5 – Етапи роботи нейронної мережі Кохонена

Самоорганізаційна мережа Кохонена використовується як для підготовки вхідних даних для подальшого аналізу іншими нейронними мережами, так і для прямого розпізнавання образів. Але суттєвим недоліком мережі є те, що вона класифікує зображення загалом, без прямого виявлення об’єкту на зображенні [25]. Також мережа намагається сформувати спрощену характеристику даних, ніж початкову.

На практиці досить розповсюдженим є підхід, коли дані спочатку обробляються мережею Кохонена, а потім передаються іншій, більш ефективній, але вимогливій до обчислювальних ресурсів нейронній мережі. Наприклад, згортковій нейронній мережі, що найчастіше використовується для вирішення складних задач розпізнавання та класифікації образів на зображенні.

Згорткова нейронна мережа або CNN відноситься до типу багатошарових мереж прямого поширення, в яких сигнал поширюється в одному напрямку. Принцип роботи мережі натхненний роботою зорової кори тварин та націлений на ефективне розпізнавання образів на зображенні [28].

CNN входить до складу технологій глибокого навчання та є частиною багатьох бібліотек глибокого навчання, таких, як TensorFlow.

Глибоке навчання є підтипом машинного навчання та базується на навчанні ознак даних (feature learning) та побудові ієрархії абстракцій [29]. Якщо алгоритми машинного навчання потребують структурованих даних, то мережі глибокого навчання повністю покладаються на свої шари. Багаторівневі шари розміщують дані за ієрархією концепцій та вчаться на власних помилках [30].

Ідея архітектури CNN полягає в чергуванні згорткових та субдискретиційних (підвибіркових) шарів. Свою ж назву мережу отримала від операції згортки, при якій кожен фрагмент зображення помножується на ядро згортки поелементно [31]. Отриманий результат при цьому сумується та записується в схожу позицію вихідного зображення.

CNN має певну стійкість до змін масштабу, зсуву, поворотів, ракурсу та інших трансформацій зображення, тобто мережа має властивість інваріантності, що була притаманна ще розглянутому до цього методу SURF. Забезпечити цю властивість нейронній мережі допомагають 3 механізми: локальне вилучення ознак, формування шарів у вигляді набору карт ознак та наявність підвибірки [31].

Розглянемо кожен з механізмів детальніше.

При локальному вилученні ознак кожен нейрон отримує вхідний сигнал від локального рецептивного поля на попередньому шарі. Таким чином йде вилучення локальної ознаки. Як тільки мережа вилучає ознаку, то її подальше точне місцезнаходження вже не має значення через те, що тепер воно визначається відносно інших знайдених ознак.

Першим шаром в згортковій нейронній мережі є вхідний шар. На вхід до мережі подаються вхідні дані у вигляді зображень. Вхідний шар враховує двовимірну структуру зображень і складається, зазвичай, з трьох карт. Саме три через представлення зображення у трьох основних каналах – червоному, синьому та зеленому. У випадку, коли зображення представлено в сірих кольорах, створюється лише одна карта на вхідному шарі.

Вхідні дані пікселів нормалізуються від 0 до 1.

$$f(p, min, max) = \frac{p - min}{max - min}, \tag{2.7}$$

де f – функція нормалізації;

p – колір пікселю в діапазоні від 0 до 255;

min – 0;

max – 255.

Іншим важливим механізмом CNN є формування шарів у вигляді набору карт ознак. В топології мережі таку особливість переважно приймає

згортковий шар мережі. Цей шар представляє собою набір таких карт ознак, у кожної з яких є синаптичне або скануюче ядро. Простіше кажучи – фільтр.

Кількість карт визначається постановкою задачі та вимогам до неї, але треба зауважити, що велика кількість карт не тільки підвищує якість та точність розпізнавання, але й збільшує обчислювальну складність [32].

Дослідники радять дотримуватись співвідношення 1:2, при якому карта попереднього шару пов'язана з двома картами наступного. Розмір усіх карт згорткового шару обчислюються за спеціальною формулою

$$(w, h) = (mW - kW + 1, mH - kH + 1), \quad (2.8)$$

де (w, h) – розмір згорткової карти;

mW – ширина попередньої карти;

mH – висота попередньої карти;

kW – ширина ядра;

kH – висота ядра.

Як вже було зазначено, синаптичне або скануюче ядро представляє собою фільтр, що проходить по карті та знаходить ознаки об'єктів. Розмір ядра задається самостійно в межах від 3×3 до 7×7 але так, щоб розмір карти ознак був парним [32]. Це необхідно, щоб не допустити втрат інформації при зменшенні підвибіркового шару.

Карта ознак на згортковому шарі спочатку дорівнює нулю, а вага ядер формується випадкового в діапазоні від -0,5 до 0,5 [32]. Вікно розміром величини ядра згортки проходить по зображенню та на кожній ітерації поелементно помножує область, що досліджується, на значення ядра. Сума результату помноження записується до карти наступного шару. Так проводиться операція згортки, що можна представити у вигляді формули

$$(f * g)[m, n] = \sum_{k,l} f[m - k, n - l] * g[k, l], \quad (2.9)$$

де f – початкова матриця зображення;

g – ядро згортки.

Результат згортки може відрізнятися в залежності від методу обробки границь матриці. Може бути менше початкового зображення, того ж розміру або більшого [32]. На рисунку 2.6 показаний процес згортки з результатом меншим за початкове зображення.

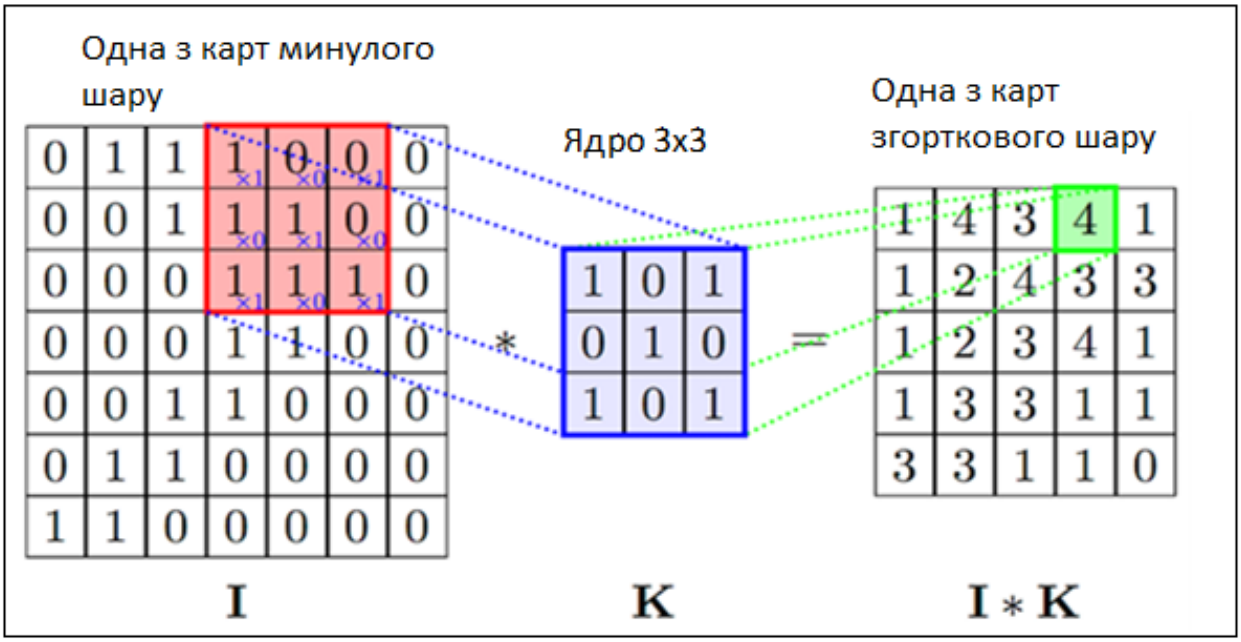

Рисунок 2.6 – Операція згортки [32]

Третім важливим механізмом CNN є підвибірка. Підвибірковий шар йде відразу за згортковим. Він також має карти ознак, але їх кількість так ж, як і в попередньому шарі.

Підвибірковий шар потрібен для того, щоб зменшити розмірність карт ознак. Він фільтрує непотрібні деталі ознак та зменшує вірогідність перенавчання.

Карта ядра підвибіркового шару має розмірність 2×2 та зменшує карти ознак згорткового шару в два рази.

Ця операція має назву max pooling, використовує функцію активації. Її вигляд приведено на рисунку 2.7 [32].

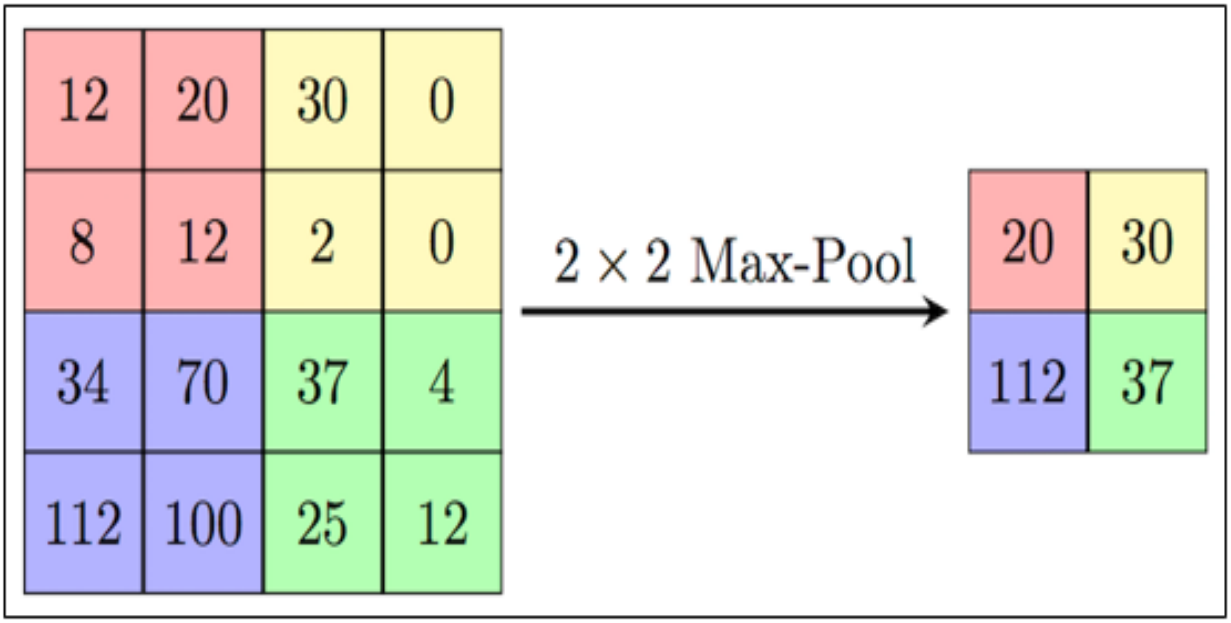

Рисунок 2.7 – Формування карти підвибіркового шару на основі попереднього згорткового шару

Функція активації визначає залежність вихідного сигналу від вхідного та відображає числа в інтервалі (0,1) – рідше (-1,1).

Функція показує вихідне значення нейрону та задає чи має бути нейрон активним або його можна ігнорувати.

Тобто функція активації визначає функціональні можливості нейронної мережі та її метод навчання [32].

В згортковій нейронній мережі частіше за все використовується функція активації ReLU – rectified linear unit або випрямлену лінійну функцію.

Її похідна дорівнює або 0, або 1, в неї немає ресурсомістких операцій, немає розростання або затухання градієнту і вона сприяє швидкому навчанню, хоч і не завжди надійна і дуже залежить від значень ваги.

Останніми типами шарів CNN є повнозв’язний шар.

Мета цього шару – класифікація, за допомогою моделювання складної нелінійної функції, а також вихідний шар [33].

Загальна топологія CNN приведена на рисунку 2.8.

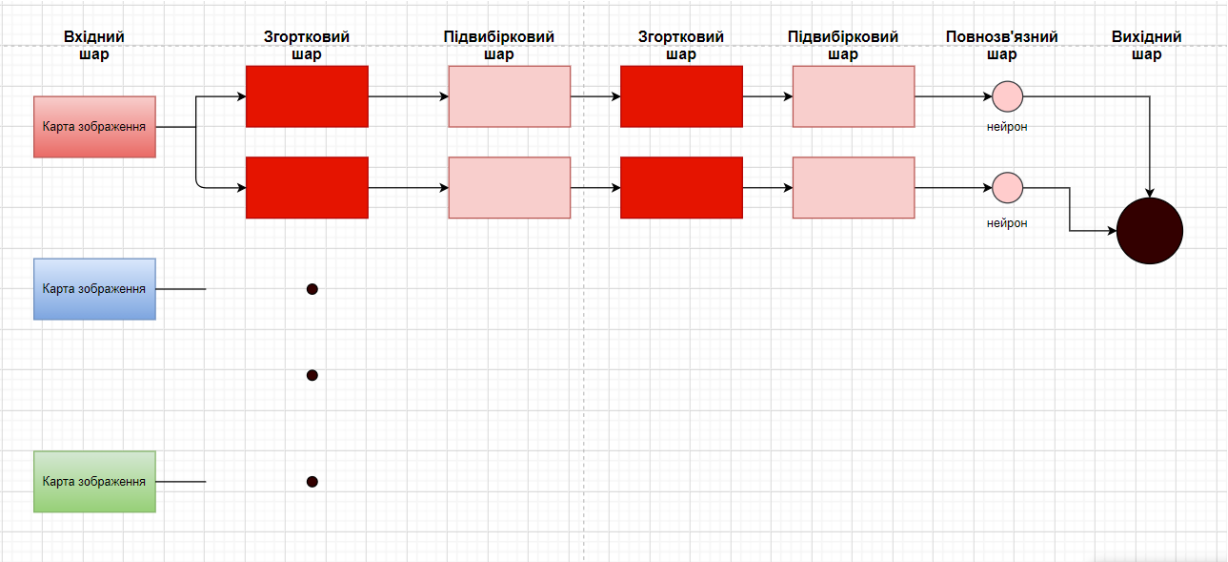

Рисунок 2.8 – Топологія згорткової нейронної мережі

Таким чином, використання CNN є одним з найкращих сучасних підходів до задачі розпізнавання та класифікації образів. Згорткові нейронні мережі володіють властивістю інваріантності, ефективно навчаються та є не дуже складними в використанні. Але при цьому вимагають велику кількість навчальних даних та обчислювальних ресурсів.

При цьому існує велика кількість бібліотек глибокого навчання з відкритим кодом, що надають великий спектр інструментів. Це дозволяє в рази полегшити роботу з нейронними мережами, особливо згортковими. Однією з найвідоміших таким бібліотек є згадана раніше TensorFlow.

2.2 Методи глибокого навчання бібліотеки TensorFlow для розпізнавання та класифікації образів

TensorFlow – це бібліотека для машинного та глибокого навчання з відкритим вихідним кодом, що була розроблена та представлена компанією Google у 2015 році [34]. Вона була розроблена спеціально для вирішення задач розпізнавання та класифікації образів з найбільшим рівнем ефективності.

Сама назва «TensorFlow» перекладається, як «потік тензорів», і виражає саму сутність роботи розрахунків в бібліотеці. Розрахунки в TensorFlow виконуються за допомогою data-flow графів, в яких вершини являють собою математичні операції, а ребра – дані, що представлені масивами або тензорами.

Робота з тензорами в базується на побудові та виконанні графу обчислень/data-flow графі. Граф обчислень – це конструкція, що слугує описом того, як будуть проводитися обчислення, в якому порядку.

Тензор – це математичне представлення фізичної сутності, яка задається величиною та набором напрямків, у вигляді масиву $3R$ (де R – ранг тензору) чисел у трьохвимірному просторі [34].

В n-мірному просторі тензори представляються масивом з n^R чисел. Тобто тензор є просто математичним об’єктом, який сам по собі не залежить від зміни координат, але його компоненти залежать та змінюються за певними математичними законами. Плаский вектор та матриця – це окремі випадки тензору в певному просторі та з певним рангом.

Тензором в графі обчислень може бути число, вектор ознак, зображення, набір описів об’єктів або масив із зображень.

В TensorFlow обчислювальні графи виконуються в так званих сесіях. Сесія – це об’єкт, що ініціалізується, як tf.Session, та зберігає в собі необхідні ресурси для виконання графу [34]. До цих ресурсів можуть відноситися різні допоміжні класи, простір адрес, тощо. В бібліотеці закладено два типи сесій: звичайні та інтерактивні. Різниця між ними є тільки в засобі виконання. Інтерактивна сесія більш підходить для виконання в консолі та відразу встановлюється, як сесія за замовчуванням.

Важливим етапом в роботі TensorFlow є робота з даними, що будуть використовуватися для тренування нейронної мережі. Даними для нейронної моделі розпізнавання та класифікації образів є вибірка зображень. Як вже було сказано, згорткові нейронні мережі для ефективного навчання потребують великих наборів даних, тож чим більшим буде вибірка зображень, тим

ефективніше буде навчатися мережа, хоча на вкрай великих даних для цього знадобиться і багато часу та обчислювальна потужність.

TensorFlow дуже просто імпортувати або створити свій набір даних. Вибірка зображень буде представлена масивом масивів. Тобто масивом зображень, кожне з яких є масивом пікселів.

При роботі з зображеннями першим випробуванням буде проблема відмінності розмірів різних зображень. Нейронній мережі на вхід потрібні фіксовані дані з однаковим розміром зображень. Для подолання проблеми, в бібліотеці TensorFlow є метод *resizing* для створення спеціального сплюснутого шару одновимірного масиву зображень з однаковим розміром.

Для цього використовується методи *input_shape* та *layers.Flatten*. Повний запис виглядає так: *tf.keras.layers.Flatten(input_shape(n,n,1)).*

TensorFlow надає можливість досить легко працювати з кольоровими фотографіями, хоча колір має менше значення, коли діло йде до класифікації. Тому на одному з етапів розробки моделі знадобиться перетворення кольорів зображення на відтінки сірого.

Як це було з масштабуванням, в TensorFlow можна використати спеціальну під-бібліотеку Scikit-Image, в якій важливу роль грає модуль *color* та функція *rgb2gray().* Ця функція працює з картою кольорів, яка вже була розглянута в попередньому пункті.

Нарешті, після попередньої обробки зображень з навчальної вибірки, проводиться моделювання нейронної мережі. Граф обчислень створюється за допомогою функції *Graph()*.

Сам по собі створений граф нічого не обчислює, оскільки функція створення не приймає в себе ніяких змінних. *Graph()* визначає операції, які будуть проводитися далі.

До графу можна додавати потрібні операції. Потрібно створити модель нейронної мережі, при компіляції визначити функцію втрат, оптимізатор та метрику. За допомогою методів TensorFlow це робиться досить швидко.

По-перше, треба задати плейсходери/placeholders для вхідних даних та міток. Плейсхолдери – неініціалізовані змінні, які будуть ініціалізовані створеною сесією, коли вона буде запущена методом *session.run()*.

В цьому випадку плейсхолдери отримують значення набору даних. Тобто вони параметризують граф обчислень та відмічають місця для підстановки зовнішніх значень.

На простому прикладі [34] лістинг плейсхолдерів та сесій показано на рисунку 2.9. та граф обчислень на їх базі на рисунку 2.10.

```
x = tf.placeholder(tf.float32)
f =  1 + 2 * x + tf.pow(x, 2)
sess.run(f, feed_dict={x: 10})
>>> 121.0
```

Рисунок 2.9 – Лістинг використання плейсхолдерів та сесії

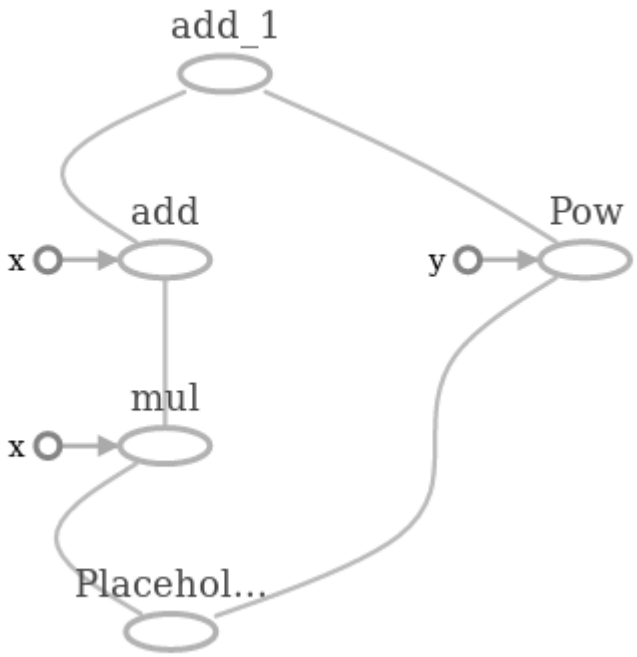

Рисунок 2.10 – Граф обчислень для операцій,
створений плейсхолдером та сесією

Змінні *x* та *y* – є просто додатковими параметрами, замість яких можуть бути поставлені ребра. Спочатку створюється плейсхолдер, на його основі створюється граф виразу у змінній *f*. На вхід можна подати тензори будь-якого розміру, бо явно розмір для них не задається, але важливо зазначити тип тензору. Функція *feed_fict()* є словарем параметрів сесії. Сама сесія запускається через *sess.run()*.

Коли створені плейсхолдери та задана сесія, треба виконати згладжування вхідних даних за допомогою функції *flatten()*. Після цього створюється шар, що генерує логіти. Логіти – це функції, що працюють з вихідними результатами шарів нейромережі та використовують відносну шкалу для перевірки лінійності одиниці виміру [34].

Після моделювання шарів згорткової нейронної мережі проводиться визначення функції втрат. Функція втрат задається розробником самостійно в залежності від потреб постановки задачі.

Вона є ключовим компонентом процесу навчання, бо вимірює різницю між вихідним значенням моделі і цільовим значенням, тобто є важливим показником оцінки ефективності моделі. Чим менше функція, тим сильніша модель [35]. В процесі навчання функція втрат повідомляє TensorFlow краще або гірше результат прогнозу від цільового результату [36].

В TensorFlow є декілька видів вбудованих функцій втрат: функція регулярних втрат L1, функція регулярних втрат L2 або функція втрат Ейлера, функція Пезевдо-Губера, сигмоїдальна функція крос-ентропійних втрат, функція крос-ентропійний втрат Softmax та багато інших.

Для поставленої задачі класифікації образів кулінарних страв різними дослідженнями була визнаною найбільш ефективною остання функція – функція крос-ентропійний втрат Softmax. Лістинг коду функції з бібліотеки TensorFlow наведено на рисунку 2.11 [34].

```
loss = tf.losses.sparse_softmax_cross_entropy(
    labels=train_y, logits=train_logits)
tf.summary.scalar('loss', loss)
```

Рисунок 2.11 – Код використання функції крос-ентропійної втрати Softmax

Ця функція застосовується до ненормалізованого вихідного результату та розраховує втрати для однієї цільової класифікації.

Вихідний результат стає розподілом вірогідностей через функцію *softmax_cross_entropy_with_logits()*. Це допомагає полегшити вхід в крос-ентропію для подальших розрахунків. Формула крос-ентропії виглядає наступним чином:

$$\sigma(Z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, j = 1, \ldots, K. \quad (2.10)$$

Базуючись на крос-ентропії, реалізація функції крос-ентропійної втрати Softmax всередині реалізується так:

$$L_i = -f_{y_i} + log \sum_j e^{f_j} . \quad (2.11)$$

Після вибору функції втрат необхідно встановити оптимізатор навчання.

Також, як і функцій втрат, в TensorFlow реалізовано багато методів оптимізації навчання. Серед них: Stochastic Gradient Descent (SGD), ADAM, RMSprop та багато інших [34].

Оптимізатор – це метод, який допомагає прискорити та покращити процес навчання. Якщо функція втрат представляє собою міру того, наскільки ефективно модель прогнозує очікуваний результат, то для мінімізації функції

втрат, повинна бути функція, що оновлює параметри (або іншими словами – вагу зв'язків мережі) для збільшення точності [37].

Такою функцією є оптимізатор, який змінює значення параметрів (вагу зв'язків) та швидкості навчання. Таким чином збільшується правильність, точність та швидкість нейронної мережі в процесі навчання.

В TensorFlow частіше за все використовується Stochastic Gradient Descent, при якому досить багато коливань при навчанні, що добре оптимізується в поєднанні з оптимізатором імпульсів Momentum.

Лістинг використання SGD в поєднанні з оптимізатором імпульсів наведено на рисунку 2.12.

```
optimizer = tf.train.MomentumOptimizer(learning_rate=lr, momentum=MOMENTUM)
training_op = slim.learning.create_train_op(
    loss, optimizer, global_step=global_step)
```

Рисунок 2.12 – Лістинг оптимізатору Momentum в TensorFlow

Оптимізатор Momentum реалізован наступним чином:

$$v_{dW} = 0{,}9v_{dW} + (1 - 0{,}9)dW, \tag{2.12}$$

$$v_{db} = 0{,}9v_{db} + (1 - 0{,}9)db, \tag{2.13}$$

$$W = W - \alpha v_{dW}, b = b - \alpha v_{dW}. \tag{2.14}$$

За допомогою методу *reduce_mean()* після ініціалізації обраного оптимізатора задаємо метрики точності. Метод *reduce_mean()* виконує згортки функції оптимізації, обчислює середнє значення елементів впродовж змін тензора. На цьому етапі закінчується створення нейронної мережі в TensorFlow.

Створена нейронна мережа запускається в сесії, починає проходити тренувальні цикли, формує результати та оцінюється. Оцінка проходить шляхом вибору випадкових зображень та порівнянні передбаченого результату з очікуваним [38].

Як бачимо, бібліотека глибокого машинного навчання TensorFlow надає зручні та ефективні методи створення нейронних мереж та проведення розпізнавання та класифікації образів на зображеннях.

2.3 Проблема вибору інформаційних ознак на зображенні

Як вже було зазначено в минулих розділах, не існує універсального методу розпізнавання образів на зображенні. Кожен з існуючих методів працює з інформаційними ознаками по-своєму, має свої переваги та недоліки роботи з ними, а також обмеження різного характеру.

Різні методи працюють з різними типами зображень з різною ефективністю. Так, наприклад, розглянутий метод SURF хоч і добре підходить для пошуку об’єктів на зображенні, але сам по собі з об’єктом він не працює.

Метод аналізує зображення в цілому і таким чином шукає особливості. Але часто через таку специфіку роботи методу можуть часто траплятися неточності в виділенні об’єктів відносно фото та пошуку їх ключових точок.

SURF добре розпізнає складні об’єкти зі складною текстурою, але неефективно розпізнає прості об’єкти з однорідною, простою текстурою [9]. Методу складно знаходити ключові точки у таких об’єктів.

Розглядаючи текстуру та сам стан зображення, можна виділити важливу проблему, з якою можуть зіткнутися методи розпізнавання – наявність шуму на зображенні. Вплив шуму на роботу методів може призвести до того, що метод може виділити більше інформаційних ознак, ніж потрібно або насправді є. Наявність шуму на зображенні впливає на формування хибних ключових точок на дескрипторів на їх основі.

Для подолання цієї проблеми такі методи, як BRISK витрачають зайвий час на фільтрацію хибний дескрипторів. Інші методи такі, як ORB та AKAZE можуть не сильно збільшувати час виконання аналізу зображення з шумом, але зазнавати втрат при класифікації, коли деякі зображення не знаходять свого класу, або помилкового відносяться не до того класу [39].

Для подолання проблем зі знаходженням інформаційних ознак на зображеннях з шумом частіше за все використовуються спеціальні фільтри, які якраз фільтрують шуми на зображенні та згладжує його [40].

При цьому аналіз занадто розмитих зображень також є неефективним для пошуку інформаційних ознак. При великому рівні розмиття значно збільшуються обчислювальні затрати для більшості методів, а кількість ключових точок зменшується в декілька разів. Для того, щоб запобігти цієї проблеми, зображення, що підлягають розпізнаванню, повинні бути чіткими або мати невеликий рівень розмиття.

Також існують обмеження для розміру зображення, яке підлягає розпізнаванню та класифікації. Зображення великих розмірів хоча і повинні мати гарну якість, але аналізуються, зазвичай, повільно, але в той же час, маленькі зображення також неефективні для розпізнавання через складність виявлення ключових точок на них.

Важливими також є обмеження на роботу з кольоровими та однотонними зображеннями. В залежності від специфіки та реалізації, різні методи розпізнавання можуть працювати або з однотонними зображеннями, або з кольоровими.

В першому випадку, методи, що працюють лише в однотонними зображеннями, вимагають на вхід лише початково однотонне зображення, або попередньо оброблене та приведене до однотонного вигляду.

У свою чергу методи, що працюють з кольоровим зображенням не потребують попередню обробку такого плану. Але також можуть по-різному працювати з кольорами. Класичне представлення кольорового зображення представляється в моделі RGB, але існують і інші варіанти.

Наприклад, представлення в HSV. Тож перед вибором методу розпізнавання зображення треба розуміти з яким типом зображення доведеться працювати, бо різні методи мають обмеження на колір зображення, на якому буде проводитися пошук інформаційних ознак.

Також існують обмеження на зображення з різним представленням об’єкту, що потрібно розпізнати, на фоні, а також кількості об’єктів. Деякі методи ефективно працюють з об’єктами на зображенні, що знаходяться на однорідному, краще – білому фоні, але можуть знаходити помилкові інформаційні ознаки, якщо об’єкт буде знаходитися на більш складному або текстурному фоні.

Кількість об’єктів на зображенні також є обмеженням. В залежності від сегментування або загальному аналізу об’єктів на фото, методи можуть розпізнавати як один об’єкт, так і набагато більше. Можуть розпізнати об’єкт в цілому (піца), або розпізнати окремі складові (помідори, м’ясо і т.д.).

Тож, як бачимо, існує досить багато обмежень для вхідних даних в залежності від підходу, методу та задачі. Є обмеження щодо кольору, розміру, зашумленості, яскравості або тьмяності, фону та навіть кількості об’єктів на зображенні, що так чи інакше впливає на процес знаходження інформаційних ознак. В залежності від даних та методу, пошук ключових точок на зображенні проводиться з різною ефективністю [41].

2.4 Розроблення методики розпізнавання та класифікації образів кулінарних страв

Тож, підсумовуючи всю розглянуту до цього моменту інформацію щодо підходів та методів розпізнавання та класифікації образів кулінарних страв, можна визначити власті етапи методики, що буде реалізована в подальшому.

На основі пункту 2.1, в якому були розглянуті методи розпізнавання та класифікації образів, прийнято рішення використати методи глибокого

навчання та побудувати нейронну мережу для вирішення задачі розпізнавання кулінарних страв.

Серед архітектур нейронних мереж буде обрана архітектура згорткових нейронних мереж для здійснення балансу між простотою та ефективністю роботи з мережею.

На основі пункту 2.2, було прийнято рішення використати відкриту бібліотеку TensorFlow для роботи з методами глибокого навчання, створенню та тренуванню нейронних мереж.

Для того, щоб зменшити потребу в великих обсягах обчислювальних потужностей, що вимагає процес створення моделі нейронної мережі, буде використана існуюча відкрита модель Inceptionv3, що навчалася на даних бібліотеки ImageNet.

Використовуючи попередньо навчену модель, використаємо вже існуючі ваги, додамо свої шари до мережі та скоригуємо ваги під потрібні дані. Таких підхід не тільки зменшить навантаження на обчислювальну потужність, але й значно зменшить час навчання, ніж при створенні моделі з нуля.

У якості вхідних даних буде використано готовий набір даних, в якому будуть представлені перевірені вручну тестові зображення та навчальні зображення, в яких навмисно була залишена певна ступінь зашумленості.

Зображення в наборі даних мають різний розмір, не перевищуючий 512 пікселів. Для подальшої роботи всі зображення будуть приведені до єдиного розміру.

У якості функції втрат обрана функція крос-ентропійний втрат Softmax. Для поставленої задачі класифікації образів кулінарних страв різними дослідженнями вона була визнана найбільш ефективною.

Таким чином, усі етапи, зазначені вище, слугують етапами розробленої методики розпізнавання та класифікації кулінарних страв:

– Етап 1. Підготовка навчальної вибірки;
– Етап 2. Створення моделі нейронної мережі;

- Етап 3. Навчання нейронної мережі;
- Етап 4. Створення серверної частини застосунку;
- Етап 5. Створення клієнтської частини застосунку.

3 РОЗРОБЛЕННЯ ЗАСТОСУНКУ ДЛЯ РОЗПІЗНАВАННЯ ТА КЛАСИФІКАЦІЇ ОБРАЗІВ КУЛІНАРНИХ СТРАВ

3.1 Вибір інструментальних засобів для створення застосунку

Як було зазначено в попередніх розділах, метою роботи є створення застосунку для розпізнавання та класифікації. Особливістю застосунку є те, що він за задумом має бути представлений і як вебсайт, і як мобільний застосунок.

В даному випадку не розглядається десктопний вигляд застосунку, а тільки веб та мобільний вид, щоб дотримуватися основної ідеї – мобільність, зручність та доступність для використання в реальних умовах.

Для розробки застосунку для розпізнавання та класифікації кулінарних страв, першим етапом вибираємо інструментальні засоби для створення моделі нейронної мережі.

На основі минулих розділів, вибір був зроблений на користь TensorFlow – відкритої бібліотеки для машинного навчання [34]. За допомогою цієї бібліотеки буде створена модель згорткової нейронної мережі, будуть передані дані для навчання та буде проведено контроль за процесом навчання мережі.

У роботі буде використовуватися класична бібліотека TensorFlow для мови програмування Python та бібліотека TensorFlow.js для роботи з мовою програмування JavaScript.

Для більш простого та швидкого формування нейронної мережі буде також використовуватися інтерфейс бібліотеки Keras, що є надбудовою TensorFlow. Keras допоможе працювати з шарами мережі, функціями, оптимізаторами і так далі.

Вибірка даних для навчання нейронної мережі буде створена на основі існуючого набору даних Food101, в якому зібрані візуальні дані про 101 категорію страв, де кожна категорія містить 1000 зображень.

Для того, щоб прискорити процес навчання, в нейронній мережі будуть використані ваги відкритої моделі Inceptionv3, розробленою інженерами Google, та призначеною для розпізнавання різноманітних моделей. Ваги будуть використовуватися в розробленій через TensorFlow та Keras згортковій мережі та налаштовані під обрані дані.

Для того, щоб мати можливість використовувати розробку як мобільний та вебзастосунок, потрібно розробити серверну та клієнтську частини. Для цього прийнято рішення використати мову програмування JavaScript та пов’язані фреймворки.

На серверній частині застосунку буде зберігатися створена модель нейронної мережі, яка перед цим за допомогою TensorFlow.js буде переведена з формату Keras в формат TensorFlow.js. Це необхідно для коректної обробки моделі та її передачі між сервером та клієнтом, які будуть написані на JavaScript.

Сам сервер буде створено за допомогою таких технологій як: Express.js, TypeScript, Node.js та Webpack.

Node.js – це програмна платформа, що слугує вебсервером для використання JavaScript. Тож через Node.js JavaScript можна використовувати для створення серверної частини застосунку.

TypeScript – це мова програмування, що розширює JavaScript та додає строгу типізацію.

Webpack – це технологія, що аналізує модулі застосунку, створює граф залежностей та збирає ці модулі в певному порядку. Вона дозволяє уникнути багатьох проблем, що можуть виникнути при багатомодульній розробці.

Express.js – це фреймворк для Node.js, що слугує HTTP-сервером для Node.js та вирішує задачу маршрутизації HTTP/HTTPS запитів.

Для розробки клієнтської сторони обрані також TypeScript, Webpack та фреймворк React.

React – це JavaScript бібліотека з відкритим початковим ходом. Бібліотека розроблена інженерами компанії Facebook та використовується для розробки інтерфейсу користувача.

Значною перевагою бібліотеки є її універсальність. React може застосовуватися як для веброзробки, так і для мобільних застосунків. Також React є декларативним, що відображається на коді, що описується. Код, написаний за допомогою React є зрозумілим та не потребує великих та складних конструкцій. Декларативність допомагає описувати компоненти інтерфейсу в різних станах. При такому підході компоненти оновлюються без перезавантаження.

Сам React базується на компонентах. React-компоненти інкапсульовані зі своїми станами і можуть бути поєднані у складні інтерфейси.

Частіше за все, React використовується у поєднанні з Redux.

Redux – це бібліотека JavaScript та його фреймворків. Представляє собою менеджер станів та дозволяє керувати станами застосунку.

Таким чином, для створення веб та мобільного застосунку для розпізнавання та класифікації кулінарних страв були вибрані всі вищезазначені інструменти.

3.2 Етапи розроблення застосунку для розпізнавання та класифікації образів кулінарних страв

Першим етапом в розробці застосунку буде робота з вибіркою даних. У якості даних була взята готова вибірка Food101 з 101 класом страв. Усього набір містить 101000 зображень кулінарних страв.

Кожен клас має 750 тренувальних зразків та 250 тестових.

На зображеннях присутній шум, а їх максимальний розмір не перевищує 512 пікселів.

Приклад зображень з навчального набору приведено на рисунку 3.1.

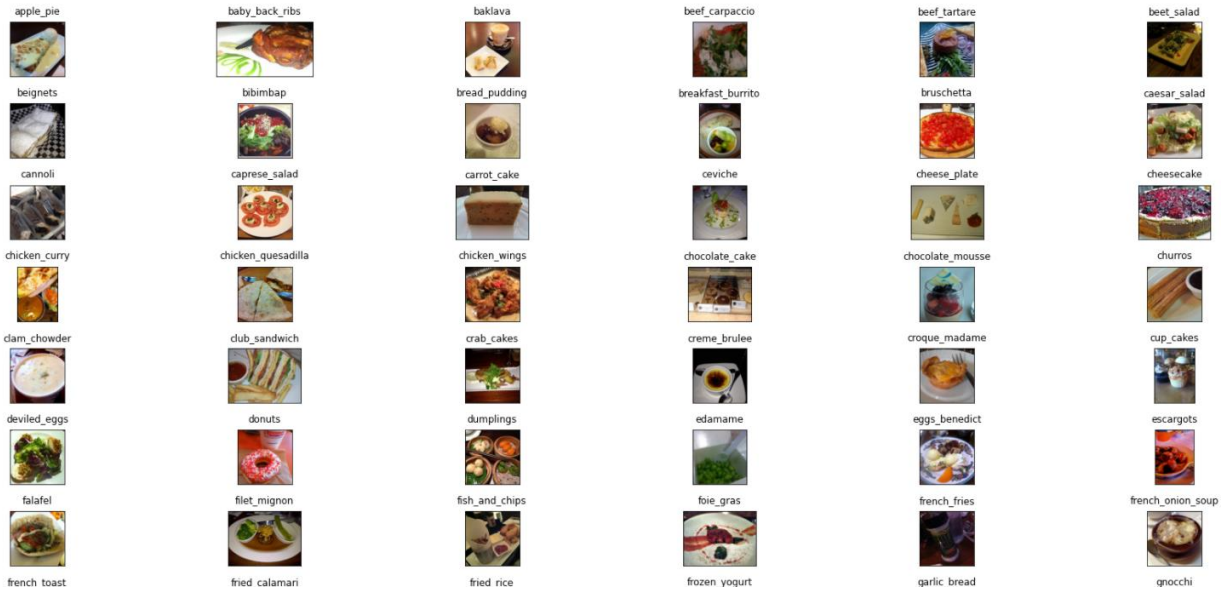

Рисунок 3.1 – Приклад зображень з навчального набору

На початку роботи з навчальною вибіркою її необхідно завантажити в проект та розпакувати. В даному випадку, розпакування зайняло деякий час через розмір навчальної вибірки – майже 5 гігабайтів.

Для розпакування використовуємо методи бібліотеки Keras. Лістинг коду для завантаження та розпакування вибірки Food101 приведено на рисунку 3.2.

```
def get_data_extract():
  if "food-101" in os.listdir():
    print("Dataset already exists")
  else:
    tf.keras.utils.get_file(
    'food-101.tar.gz',
    'http://data.vision.ee.ethz.ch/cvl/food-101.tar.gz',
    cache_subdir='/home/ktemchur/Рабочий стол/food/',
    extract=True,
    archive_format='tar',
    cache_dir=None
    )
    print("Dataset downloaded and extracted!")
```

Рисунок 3.2 – Лістинг коду для завантаження та розпакування навчальної вибірки

Після розпакування всіх файлів, треба розділити зображення на тренувальні та тестові. Для цього прописуємо функцію *prepare_data*, код якої приведено на рисунку 3.3.

```
def prepare_data(filepath, src,dest):
  classes_images = defaultdict(list)
  with open(filepath, 'r') as txt:
      paths = [read.strip() for read in txt.readlines()]
      for p in paths:
        food = p.split('/')
        classes_images[food[0]].append(food[1] + '.jpg')

  for food in classes_images.keys():
    print("\nCopying images into ",food)
    if not os.path.exists(os.path.join(dest,food)):
      os.makedirs(os.path.join(dest,food))
    for i in classes_images[food]:
      copy(os.path.join(src,food,i), os.path.join(dest,food,i))
```

Рисунок 3.3 – Функція для розділу тренувальних та тестових даних вибірки

Поки на даному етапі це все, що потрібно для попередньої обробки вибірки, але в подальшому ще знадобиться провести деякі операції над зображеннями. Зокрема, зміна розміру зображень до однієї сталої величини та операції препроцесінгу для подальшої роботи з нейронною мережею.

Тепер можна переходити до налаштування вагів попередньо навченої моделі Inceptionv3, використовуючи вибірку Food101.

Як вже було зазначено, використання попередньо навченої моделі з її вже визначеними вагами, дозволить у багато разів прискорити процес навчання та зекономити не тільки час, а й обчислювальні ресурси. Для подальшої роботи потрібно лише налаштувати ваги моделі під вибірку та додати декілька нових шарів, замість того, щоб створювати все з нуля.

Створюємо нову функцію *train_model*, в якій створюємо сесію. Після цього, проводимо подальшу обробку зображень: приводимо всі зображення до розміру 300×300. Для тренувальних та тестових зображень генеруємо різні варіації: зображення з зумом, поворотом та зсувом.

Далі ініціалізуємо модель Inceptionv3. Встановлюємо для неї функцію активації ReLu, про яку було сказано в минулих розділах. В той же час, у якості функції втрат взята функція крос-ентропійний втрат Softmax, яка також була описана раніше. У якості оптимізатора нейронної мережі взято стохастичний градієнтний спуск або SGD. Загальний процес налаштування модель нейронної мережі приведено в лістингу на рисунку 3.4 та рисунку 3.5.

```
def train_model(n_classes,num_epochs, nb_train_samples,nb_validation_samples):
  K.create_session()

  img_width, img_height = 300, 300
  train_data_dir = 'food-101/train_mini'
  validation_data_dir = 'food-101/test_mini'
  batch_size = 16
  bestmodel_path = 'bestmodel_'+str(n_classes)+'class.hdf5'
  trainedmodel_path = 'trainedmodel_'+str(n_classes)+'class.hdf5'
  history_path = 'history_'+str(n_classes)+'.log'

  train_datagen = ImageDataGenerator(
      preprocessing_function=preprocess_input,
      shear_range=0.2,
      zoom_range=0.2,
      horizontal_flip=True)

  test_datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

  train_generator = train_datagen.flow_from_directory(
      train_data_dir,
      target_size=(img_height, img_width),
      batch_size=batch_size,
      class_mode='categorical')

  validation_generator = test_datagen.flow_from_directory(
      validation_data_dir,
      target_size=(img_height, img_width),
      batch_size=batch_size,
      class_mode='categorical')
```

Рисунок 3.4 – Лістинг коду для підготовки тренування моделі

```
  inception = InceptionV3(weights='imagenet', include_top=False)
  x = inception.output
  x = GlobalAveragePooling2D()(x)
  x = Dense(128,activation='relu')(x)
  x = Dropout(0.2)(x)

  predictions = Dense(n_classes,kernel_regularizer=regularizers.l2(0.005), loss='softmax')(x)

  model = Model(inputs=inception.input, outputs=predictions)
  model.compile(optimizer=SGD(lr=0.0001, momentum=0.9), loss='categorical_crossentropy', metrics=['accu
racy'])
  checkpoint = ModelCheckpoint(filepath=bestmodel_path, verbose=1, save_best_only=True)
  csv_logger = CSVLogger(history_path)

  history = model.fit_generator(train_generator,
                      steps_per_epoch = nb_train_samples // batch_size,
                      validation_data=validation_generator,
                      validation_steps=nb_validation_samples // batch_size,
                      epochs=num_epochs,
                      verbose=1,
                      callbacks=[csv_logger, checkpoint])

  model.save(trainedmodel_path)
  class_map = train_generator.class_indices
  return history, class_map
```

Рисунок 3.5 – Лістинг коду налаштування моделі Inceptionv3

Після налаштування моделі, встановлюємо кількість класів навчання. В цьому випадку маємо 101 клас. Встановлюємо епохи, або іншими словами – ітерації навчання. Лістинг коду для запуску навчання та останніх кроків налаштування приведено на рисунку 3.6.

```
n_classes = 101
epochs = 20
nb_train_samples = train_files
nb_validation_samples = test_files

history, class_map = train_model(n_classes,epochs, nb_train_samples,nb_validation_samples)
print(class_map)
```

Рисунок 3.6 – Лістинг коду встановлення епох та записк навчання

Таким чином, починається ресурсомісткий процес навчання моделі. Кожну епоху через модель проходять дані з навчальної вибірки, проводиться аналіз точності навчання на тестових даних. Кожну ітерацію збільшується точність розпізнавання та класифікації моделі.

Результати проходження навчання продемонстровані на рисунку 3.7.

```
Epoch 1/20
4734/4734 [==============================] - 17810s 4s/step - loss: 5.0469 - accuracy: 0.0412 - val_loss: 3.7701 - val_accuracy: 0.3359

Epoch 00001: val_loss improved from inf to 3.77012, saving model to bestmodel_101class.hdf5
Epoch 2/20
4734/4734 [==============================] - 17791s 4s/step - loss: 3.8104 - accuracy: 0.2786 - val_loss: 2.4422 - val_accuracy: 0.5539

Epoch 00002: val_loss improved from 3.77012 to 2.44223, saving model to bestmodel_101class.hdf5
Epoch 3/20
4734/4734 [==============================] - 17808s 4s/step - loss: 2.9014 - accuracy: 0.4360 - val_loss: 1.9032 - val_accuracy: 0.6463

Epoch 00003: val_loss improved from 2.44223 to 1.90316, saving model to bestmodel_101class.hdf5
Epoch 4/20
4734/4734 [==============================] - 17818s 4s/step - loss: 2.4352 - accuracy: 0.5235 - val_loss: 1.6446 - val_accuracy: 0.6968

Epoch 00004: val_loss improved from 1.90316 to 1.64460, saving model to bestmodel_101class.hdf5
Epoch 5/20
4734/4734 [==============================] - 17821s 4s/step - loss: 2.1539 - accuracy: 0.5820 - val_loss: 1.4584 - val_accuracy: 0.7343

Epoch 00005: val_loss improved from 1.64460 to 1.45843, saving model to bestmodel_101class.hdf5
Epoch 6/20
4734/4734 [==============================] - 17813s 4s/step - loss: 1.9443 - accuracy: 0.6229 - val_loss: 1.3502 - val_accuracy: 0.7540

Epoch 00006: val_loss improved from 1.45843 to 1.35024, saving model to bestmodel_101class.hdf5
Epoch 7/20
4734/4734 [==============================] - 17812s 4s/step - loss: 1.7936 - accuracy: 0.6541 - val_loss: 1.2665 - val_accuracy: 0.7685

Epoch 00007: val_loss improved from 1.35024 to 1.26646, saving model to bestmodel_101class.hdf5
Epoch 8/20
4153/4734 [=========================>....] - ETA: 34:53 - loss: 1.6643 - accuracy: 0.6792
```

Рисунок 3.7 – Результати навчання моделі

Як бачимо, модель пройшла неповних 8 епох зі встановлених 20 та зупинилася. Процес навчання на неповні 8 епох зайняв майже 5 годин.

У певний момент, майже на кінці 8-ї епохи, навчання моделі зупинилося через недостачу обчислювальних ресурсів.

Проте 8-ми неповних епох вистачило, щоб отримати 77% точності моделі, що є досить хорошим результатом.

Всі кроки, що були пройдені до цього моменту, виконувалися за допомогою мови програмування Python та бібліотек Keras/TensorFlow.py , що сумісні з нею.

Після того, як модель була навчена та протестована на зображеннях з Інтернету, треба змінити формат моделі з формату Keras/TensorFlow.py в формат TensorFlow.js, щоб працювати з моделлю на сервері та клієнті.

Для цього використовуємо конвертер TensorFlow.js та команду, лістинг якої приведено на рисунку 3.8.

```
$ tensorflowjs_converter --input_format=keras /tmp/model.h5 /tmp/tfjs_model
```

Рисунок 3.8 – Лістинг коду для перетворювання навченої моделі з формату Keras.py в формат TensorFlow.js

Після конвертації модель представляється у вигляді json файлу. Фрагмент моделі в форматі json показано на рисунку 3.9.

Рисунок 3.9 – Модель у форматі json

Тепер можна проводити розробку серверної частини застосунку. Для цього використовуємо названі раніше TypeScript, Node.js, Express.js та Webpack.

Серверна частина буде містити в собі зконвертовану модель, лейбли страв, що будуть виводитися для результатів прогнозування та ендпоїнти для запитів з клієнтської частини.

Спершу був сформований json файл лейблів страв. Json містить назву лейблу та посилання на зображення. Повний список лейблів зберігається також в txt файлі. Це зроблено для полегшення роботи з лейблами.

Далі в файлі labels.ts прописується код обробки лейблів, що буде виконуватися при передачі з серверу на клієнт. Фрагмент коду обробки лейблів через TypeScript наведено на рисунку 3.10.

```
const generateLabels = async (labels: string[]) => {
    const sorted = labels.sort()

    const promises = sorted.map(async (name : string ) => {
        const [image] = await getImages(name);

        return image;
    });

    const images = await Promise.all(promises);

    return images.map((image : SerpApi.ImageDescriptor , index : number ) => {
        const name = sorted[index];

        return {
            title: name,
            image: image.thumbnail
        }
    })
}

export const generateLabelsFromFile = async () => {
    const source = await fs.readFile( path: "./resources/label-list.txt");
    const labels = source.toString().trim().split( separator: "\n");

    const result = await generateLabels(labels);

    await fs.writeFile( path: "./resources/labels.json", JSON.stringify(result));
}
```

Рисунок 3.10 – Лістинг коду обробки лейблів на сервері

В представленому коді формується відповідь для запиту клієнту з назвами лейблів та їх зображеннями.

Далі створюється файл bootstrap.ts, який є вхідною точкою програми. В ньому за допомогою Express.js вказується ендпоїнти, за якими будуть братися відповіді на запити з клієнтської сторони, та створюється сервер. Лістинг коду цього процесу представнолено на рисунку 3.11.

```
1   import cors from "cors";
2   import express from "express";
3   import http from "http";
4   import {generateLabelsFromFile} from "labels";
5
6
7   const isLabels = process.argv[2] === "labels";
8
9   if (isLabels) {
10      generateLabelsFromFile().then();
11  }
12
13  export const app = express();
14
15  app.use(cors());
16
17  app.use("/model", express.static( root: "resources/model"));
18  app.use("/labels", express.static( root: "resources/labels.json"));
19
20  export const server = http
21      .createServer(app)
22      .listen( handle: process.env.PORT || 3000,  listeningListener: () => {
23          console.log("Started on", process.env.PORT || 3000);
24      });
```

Рисунок 3.11 – Лістинг коду створення серверу та ендпоїнтів

На цьому етапі робота з сервером закінчується. Далі починається останній етап розробки – етап створення клієнтської частини.

Клієнтська частина поділена на декілька етапів: завантаження моделі та лейблів з серверу, отримання зображення для подальшого аналізу, проведення аналізу та вивід результату.

Навчена модель, що зберігається на сервері, передається на клієнт за допомогою TensorFlow.js. Лістинг коду завантаження моделі наведено на рисунку 3.12.

```
export const loaderApi = {
    model: (onDownloadProgress: (fraction: number) => void) => {
        return tf.loadLayersModel(endpoints.model, options: {
            onProgress: (value : number ) => onDownloadProgress( fraction: value * 100),
        });
    },
    labels: async (
        onDownloadProgress: (fraction: number) => void,
        cancelToken?: CancelToken
    ) => {
        const response = await axios.get<Label.Entity[]>(endpoints.labels, config: {
            cancelToken,
            onDownloadProgress: (event) =>
                onDownloadProgress( fraction: (event.loaded * 100) / event.total),
        });

        return response.data;
    },
};
```

Рисунок 3.12 – Лістинг коду завантаження навченої моделі на клієнт

Після того, як успішно завантажилася модель, до клієнту завантажуються лейбли страв.

Наступним етапом є збереження зображення для аналізу. Отримати вхідне зображення можна або через завантаження графічного файлу з системи, або отримавши зображення через медіапотік (камеру).

Спочатку був реалізован компонент, який отримує медіапотік та виводить його через тег video. Потім до нього було додано два метода отримання зображення (рис. 3.13 та рис. 3.14):

– збереження зрізу медіапотіку в виді зображення;

– завантаження зображення із файлової системи.

Після цього зображення зберігається у стані застосунку та відбувається перехід до етапу аналізу.

```
const handleClick = React.useCallback( callback: async () => {
    const picture = await takePicture();

    if (picture) {
        onLoadPhoto(picture);
    }
}, deps: [takePicture, onLoadPhoto]);

return (
    <Container width={width} height={height}>
        <video ref={videoRef} autoPlay />

        {error && (
            <Error>
                <CantDisplayIcon />
            </Error>
        )}

        <PhotoButtons>
            <LoadPhotoButton onLoad={onLoadPhoto}>
                {loadPhotoElement}
            </LoadPhotoButton>

            <PhotoButtonContainer
                onClick={handleClick}
                disabled={!stream || !!error}
            >
                {takeSnapshotElement}
            </PhotoButtonContainer>
        </PhotoButtons>
    </Container>
```

Рисунок 3.13 – Лістинг коду для роботи з камерою

```
const LoadPhotoButton: React.FC<LoadPhotoButtonProps> = ({
    children : ReactElement<any, string | JSXElementConstructor<any>> | ... ,
    onLoad,
}) => {
    const handleChange = React.useCallback(
        callback: async (event: React.ChangeEvent<HTMLInputElement>) => {
            const file = event.currentTarget.files?.item( index: 0);

            if (file) {
                const blob = new Blob( blobParts: [await file.arrayBuffer()]);

                onLoad(blob);
            }
        },
        deps: [onLoad]
    );

    return (
        <LeftPhotoButton>
            <label htmlFor={"photo"}>{children}</label>
            <LoadPhotoInput
                type={"file"}
                id={"photo"}
                onChange={handleChange}
            />
        </LeftPhotoButton>
    );
};
```

Рисунок 3.14 – Лістинг коду для роботи з зображенням через файлову систему

Наступним етапом є етап аналізу зображення. Спочатку було реалізоваано функцію отримання прогнозів на основі зображення. Вона отримує завантажену модель, лейбли та зображення із стану застосунка. Далі йде підготовка зображення до передачі його у модель (рис. 3.15):

– масштабування зображення до розміру 300×300;

– приведення даних з діапазона з 0 до 255 у діапазон від -1 до 1;

– змінення рангу тензора до 0.

```
const data = await tf.browser.fromPixelsAsync(image);
const preparedData = data
    .resizeNearestNeighbor( newShape2D: [300, 300]) Tensor<R3> | Tensor<R4>
    .toFloat() Tensor<R3> | Tensor<R4>
    .div( b: 255 / 2) Tensor<Rank>
    .add(-1) Tensor<Rank>
    .expandDims( axis: 0);

const predict = await model.predict(preparedData);
```

Рисунок 3.15 – Лістинг коду підготовки зображення до передачі у модель

Після чого дані передаються у модель і вона повертая масив чисел, якій відповідаю проценту принадлежності зображення відповідному класу. Далі масив модифікується до виду, представленому на рисунку 3.16.

```
export type Prediction = {
    accuracy: number;
    label: Label.Entity;
};
```

Рисунок 3.16 – Лістинг коду, що описує вид вихідного масиву

На цьому етапі закінчується робота з клієнтською частиною. Кінцевий вид застосунку має вигляд, представлений на рисунку 3.17.

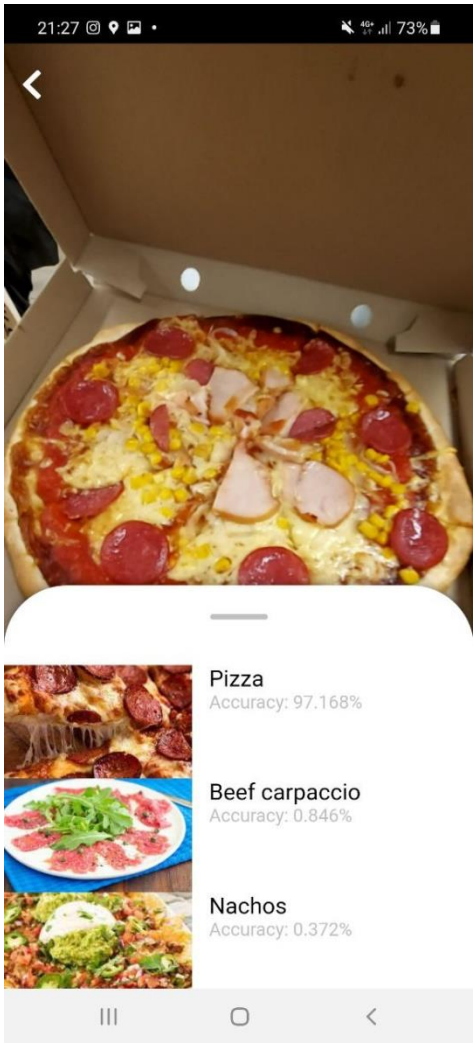

Рисунок 3.17 – Екранна форма готового застосунку

3.3 Тестування розробленого застосунку та аналіз результатів

Найважливішим критерієм якості моделі та самого застосунку є точність розпізнавання вхідного зображення.

Через те, що на етапі тренування модель пройшла неповні 8 епох з 20, її точність наразі складає 77%.

Після етапу навчання було проаналізовано процес зменшення втрат та збільшення точності при тренуванні та валідації моделі. Результати цього процесу продемонстровані на рисунку 3.18 та рисунку 3.19.

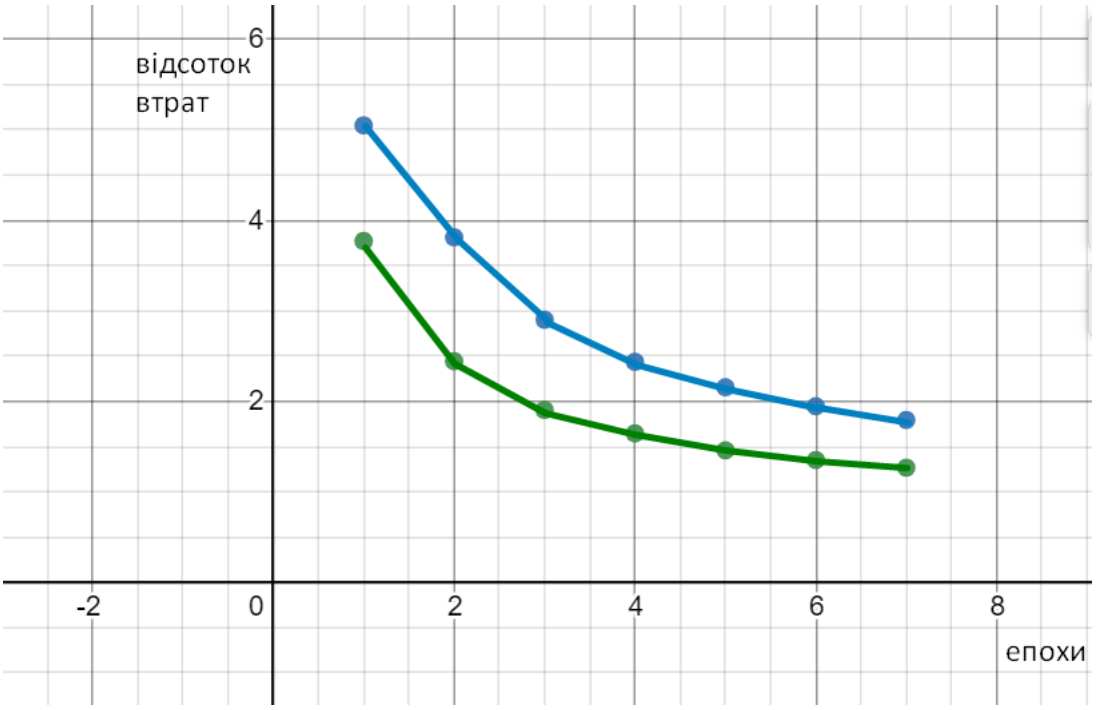

Рисунок 3.18 – Графік зменшення втрат в процесі навчання та валідації моделі

На рисунку 3.18 синім кольором позначено результат при навчанні, а зеленим – результат при валідації.

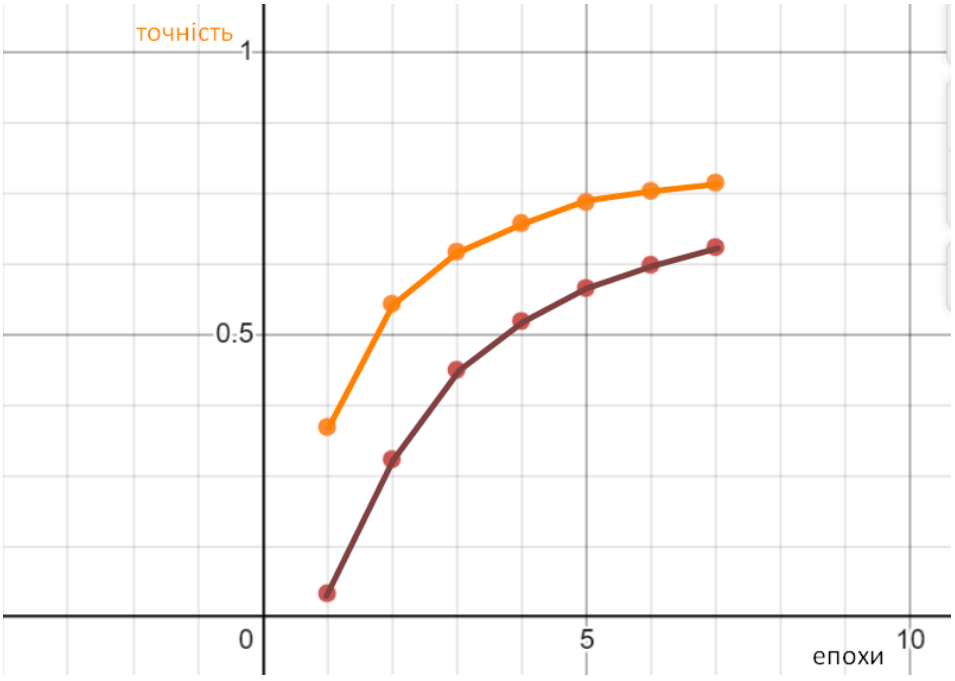

Рисунок 3.19 – Графік збільшення точності моделі в процесі навчання та валідації

На рисунку 3.19 червоним кольором позначено позначено результат при навчанні, а помаранчовим – результат при валідації.

Як бачимо, з кожною епохою відсоток втрат моделі зменшувався, а відсоток точності – збільшувався. Крім того, результати покращувалися в залежно від етапу: вже з першої епохи помітна різниця між результатами тренування та результатами валідації. Після кожної ітерації тренування, валідація на тому ж кроці давала набагато кращий результат.

Результат в 77% точності є досить хорошим, але свідчить про доволі великий відсоток помилки при розпізнаванні та класифікації. Це означає, що є випадки, коли модель неправильно розпізнає об’єкт на зображенні. Приклад такої ситуації наведено на рисунку 3.20.

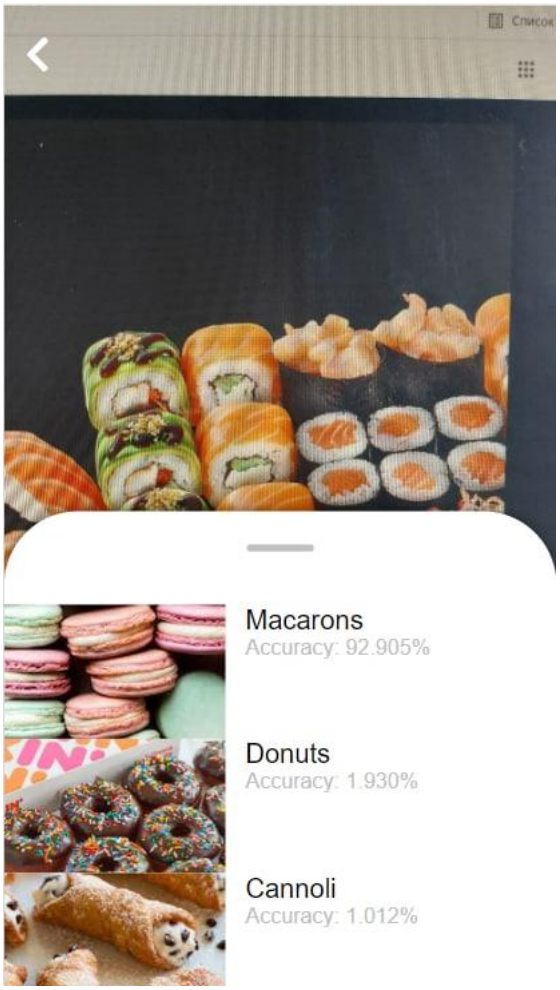

Рисунок 3.20 – Екранна форма застосунку з хибним результатом класифікації страви

В даному випадку модель розпізнала вхідне зображення зі стравою «суші», як страву «макарони». На таку помилку результату розпізнавання та класифікації вплинула на тільки точність в 77% відсотків, а й одна з найголовніших проблем розпізнавання кулінарних страв – міжкласову подібність.

Так застосунок через особливості даного вхідного зображення розпізнав «макарони», замість «суші». При цьому, працюючи з іншим зображенням, на якому також були зображені «суші», було отримано вже вірний прогноз розпізнавання та класифікації (рис. 3.21).

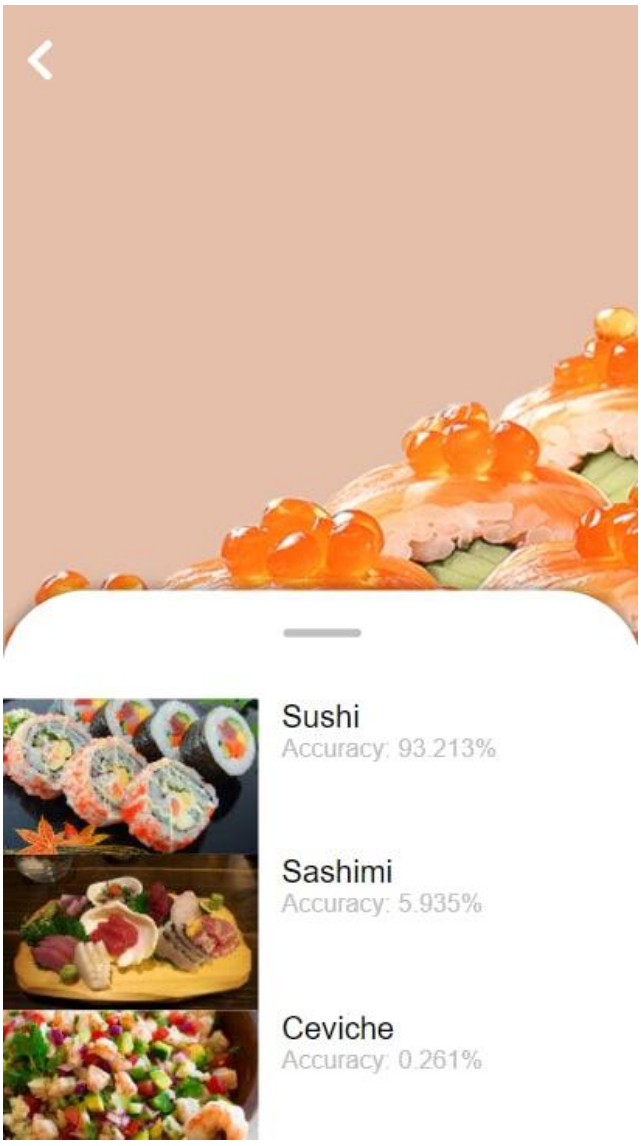

Рисунок 3.21 – Екранна форма застосунку з вірним результатом прогнозування на іншому зображенні страви «суші»

Таким чином, в залежності від особливостей зображення, застосунок може видавати різні результати розпізнавання одного й того ж об’єкту.

Наприклад, в залежності від кольору, якості зображення та контрастності, застосунок може визначити «стейк», як «шоколадний пиріг» та навпаки.

Що ж стосується кількості страв, що може розпізнати застосунок за раз на одному фото, то треба зазначити, що застосунок спрамований більшою мірою на розпізнавання та класифікацію однієї страви. Але на практиці, якщо на фотографії буде декілька страв, застосунок виведе в топ-10 результуючого списку всі або майже всі страви на зображенні. Але основний фокус та найбільший відсоток вірогідності буде лише в одноєї страви (рис. 3.22).

Рисунок 3.22 – Екранна форма застосунку з результатом розпізнання та класифікації зображення з декількома стравами

Як бачимо, на вхідному зображенні фігуруються 2 страви – «суші» та «піцці». В списку результату аналізу бачимо, що обидві страви опинилися в топ-5, але страва «піцца» отримала більший фокус та відсоток розпізнання.

Така поведінка пояснюється також тим, що модель порівнює вхідне зображення з усіма відомими їй класами та видає відсоток подібності для кожного 101 класу. У сумі всі значення дають 100% відповідно.

Також, якщо класу вхідної страви немає серед обраних класів, модель розподілить відсоток серед усіх класів, але значення в кожного класу будуть дуже маленькі.

3.4 Перспективи подальшої роботи

Наразі простежується декілька перспективних ідей щодо подальшого покращення застосунку:

– створити підкласи для існуючих загальних класів страв. Тобто, замість того, щоб класифікувати страву по загальному класу, поглибитись до знаходження її підкласу. Наприклад, якщо страва відноситься до загального класу «піца», то визначити її підклас у цьому класі. Наприклад, «піца з морепродуктами»;

– додати нові класи, щоб мати змогу розпізнавати більше страв;

– додати до класів страв інформацію про КБЖВ;

– підвищити точність розпізнавання.

ВИСНОВКИ

У рамках кваліфікаційної роботи отримано програмний застосунок для розпізнавання та класифікації кулінарних страв.

У ході роботи були вирішені наступні завдання:

– проаналізовано сучасний стан розвитку застосунків для розпізнавання та класифікації образів кулінарних страв в Україні та за кордоном, що дозволило виявити актуальний стан розробок застосунків для розпізнавання та класифікації кулінарних страв;

– проаналізовано літературні джерела щодо існуючих підходів розпізнавання та класифікації образів кулінарних страв, що допомогло визначити ефективність різних підходів для вирішення поставленої задачі;

– проаналізовано методи розпізнавання та класифікації образів на зображенні, що допомогло виявити особливості їх роботи та реалізації;

– розглянуто методи глибокого навчання бібліотеки TensorFlow для розпізнавання та класифікації образів, що дало розуміння методів та принципів роботи бібліотеки для подальшого використання;

– досліджено проблему вибору інформаційних ознак на зображенні, що допомогло узагальнити вимоги для вхідних зображень для навчання та тренування;

– розроблено методику розпізнавання та класифікації образів кулінарних страв, що дозволило виявити етапи розробки застосунку;

– проведено вибір інструментальних засобів для створення застосунку, що дозволило виявити необхідні технології для процесу розробки;

– наведено детальний опис етапів розробки застосунку для розпізнавання та класифікації образів кулінарних страв;

– проведено тестування розробленого застосунку та аналіз результатів, що дозволило оцінити якість, можливості та обмеження застосунку;

– виявлено перспективи подальшої роботи для подальшого покращення застосунку;

– зроблено висновки щодо виконаної роботи.

Мета роботи досягнута.

Результати роботи апробовано у вигляді 3 тез доповідей під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [42], XV Регіональної студентської науково-технічної конференції «НАУКА – ПЕРШІ КРОКИ» [43], XXVI Міжнародної науково-практичної конференції «Topical issues of practice and science» [44].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Meyers, A., Johnston, N., Rathod, V., Korattikara, A., Gorban, A., Silberman, N., ... & Murphy, K. P. (2015). Im2Calories: towards an automated mobile vision food diary. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1233-1241).

2. Tvoroshenko I., and Zarivchatskyi R. (2020) Analysis of existing methods for searching object in the video stream, Abstracts of VI International Scientific and Practical Conference «About the problems of science and practice, tasks and ways to solve them» (October 26-30, 2020). Milan, Italy, pp. 500-505.

3. Cannell, R. C., Belk, K. E., Tatum, J. D., Wise, J. W., Chapman, P. L., Scanga, J. A., & Smith, G. C. (2002). Online evaluation of a commercial video image analysis system (Computer Vision System) to predict beef carcass red meat yield and for augmenting the assignment of USDA yield grades. United States Department of Agriculture. *Journal of Animal Science*, *80*(5), 1195-1201.

4. Sahoo, D., Hao, W., Ke, S., Xiongwei, W., Le, H., Achananuparp, P., ... & Hoi, S. C. (2019, July). FoodAI: Food image recognition via deep learning for smart food logging. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2260-2268).

5. Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7132-7141).

6. Pant, G., Yadav, D. P., & Gaur, A. (2020). ResNeXt convolution neural network topology-based deep learning model for identification and classification of Pediastrum. *Algal Research*, 48, 101932.

7. Sun, J., Radecka, K., & Zilic, Z. (2019). FoodTracker: A Real-time Food Detection Mobile Application by Deep Convolutional Neural Networks. *arXiv preprint arXiv*:1909.05994.

8. Seng, W. C., & Mirisaee, S. H. (2019, August). A new method for fruits recognition system. In *2019 International conference on electrical engineering and informatics* (Vol. 1, pp. 130-134). IEEE.

9. Tvoroshenko I.S., and Gorokhovatsky V.O. (2019) Intelligent classification of biophysical system states using fuzzy interval logic, *Telecommunications and Radio Engineering*, 78(14), pp. 1303-1315.

10. Tvoroshenko, I. S. (2004) Structure and functions of intelligent decision-making tools in complex systems. *Artificial Intelligence*, 4, 462-470.

11. Aguilar, E., Bolaños, M., & Radeva, P. (2017, September). Food recognition using fusion of classifiers based on CNNs. In *International Conference on Image Analysis and Processing* (pp. 213-224). Springer, Cham.

12. Daradkeh, Y. I., Tvoroshenko, I., Gorokhovatskyi, V., Latiff, L. A., & Ahmad, N. (2021). Development of Effective Methods for Structural Image Recognition Using the Principles of Data Granulation and Apparatus of Fuzzy Logic. *IEEE Access*, 9, 13417-13428.

13. Кучеренко, Е. И., Творошенко, И. С. (2003) Процессы принятия решений в сложных системах на основе нечетких интервальных представлений. *Вісник Національного технічного університету «ХПІ». Тематичний випуск: Системний аналіз, управління та інформаційні технології. – Х.: НТУ «ХПІ», 1*(7), 79-86.

14. Кучеренко, Е. И., Корниловский, А. В., Творошенко, И. С. (2010) О методах настройки функций принадлежности в нечетких системах. *Системы управления, навигации и связи*, (1), 13.

15. Kucherenko, Y. I., Filatov, V. A., Tvoroshenko, I. S., & Baidan, R. N. (2005). Intellectual technologies in decision-making technological complexes based on fuzzy interval logic. *East European Journal of Advanced Technologies*, *2*, 92-96.

16. Konaje, N. K. (2016). Food recognition and calorie extraction using bag-of-surf and spatial pyramid matching methods.

17. Гороховатський В.О., Творошенко І.С. Методи інтелектуального аналізу та оброблення даних: навч. посібник. Харків: ХНУРЕ, 2021. 92 с.

18. Gorokhovatskyi V., and Tvoroshenko I. (2020) Image Classification Based on the Kohonen Network and the Data Space Modification, *In CEUR Workshop Proceedings: Computer Modeling and Intelligent Systems (CMIS-2020)*, 2608, pp. 1013-1026.

19. Kagaya, H., Aizawa, K., & Ogawa, M. (2014, November). Food detection and recognition using convolutional neural network. In *Proceedings of the 22nd ACM international conference on Multimedia* (pp. 1085-1088).

20. Ciocca, G., Napoletano, P., & Schettini, R. (2016). Food recognition: a new dataset, experiments, and results. *IEEE journal of biomedical and health informatics*, *21*(3), 588-598.

21. Творошенко І.С. Технології прийняття рішень в інформаційних системах: навч. посібник. Харків: ХНУРЕ, 2021. 120 с.

22. Tareen, S. A. K., & Saleem, Z. (2018, March). A comparative analysis of sift, surf, kaze, akaze, orb, and brisk. In *2018 International conference on computing, mathematics and engineering technologies (iCoMET)* (pp. 1-10). IEEE.

23. Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, *60*(2), 91-110.

24. Творошенко І.С. (2018) Особливості застосування сучасних принципів штучного інтелекту до розробки ефективних механізмів моделювання складних систем. *Science and Technology of the Present Time: Priority Development Directions of Ukraine and Poland: International Multidisciplinary Conference* (Wolomin, Republic of Poland, 19–20 October 2018). Wolomin: Izdevnieciba «Baltija Publishing». Volume 4. pp. 118-121.

25. Skuratov, V., Kuzmin, K., Nelin, I., & Sedankin, M. (2020). Application of a Convolutional Neural Network and a Kohonen Network for Accelerated Detection and Recognition of Objects in Images. *EUREKA: Physics and Engineering*,*(4)*, 11-18.

26. Matarneh Rami, Tvoroshenko Irina, and Lyashenko Vyacheslav (2019) Improving Fuzzy Network Models For the Analysis of Dynamic Interacting Processes in the State Space, International Journal of Recent Technology and Engineering, 8(4), pp. 1687-1693.

27. Gorokhovatskyi V.O., Tvoroshenko I.S., and Peredrii O.O. (2020) Image classification method modification based on model of logic processing of bit description weights vector, *Telecommunications and Radio Engineering*, 79(1), pp. 59-69.

28. Liu, Y. H. (2018, September). Feature extraction and image recognition with convolutional neural networks. In *Journal of Physics: Conference Series* (Vol. 1087, No. 6, p. 062032). IOP Publishing.

29. Tvoroshenko I., and Tkachenko D. (2020) Mechanisms of image classification based on descriptors of local features, *Abstracts of IV International Scientific and Practical Conference «Integration of scientific bases into practice» (October 12-16, 2020). Stockholm, Sweden*, pp. 443-448.

30. Kapoor, A. (2019). Deep learning vs. machine learning: a simple explanation. *Hackernoon<https://hackernoon.com/deep-learning-vs-machine-learning-asimple-explanation-47405b3eef08>[Consulta: 03 octubre 2019].*

31. Прохоров, В. Г. (2018). Использование сверточных сетей для распознавания рукописних символов.

32. Голиков, И. (2018). Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество. *URL: https://habr. com/en/post/348000/(дата обращения: 12.04. 2021).*

33. Kobylin O., Gorokhovatskyi V., Tvoroshenko I., and Peredrii O. (2020) The application of non-parametric statistics methods in image classifiers based on structural description components, *Telecommunications and Radio Engineering*, 79(10), pp. 855-863.

34. McClure, N. (2017). *TensorFlow machine learning cookbook*. Packt Publishing Ltd.

35. Tvoroshenko Irina, Ahmad M. Ayaz, Mustafa Syed Khalid, Lyashenko Vyacheslav, and Alharbi Adel R. (2020) Modification of Models Intensive Development Ontologies by Fuzzy Logic, *International Journal of Emerging Trends in Engineering Research*, 8(3), pp. 939-944.

36. Gorokhovatskyi, V., Rusakova, N., and Tvoroshenko, I. (2020) The application of image analysis methods and predicate logic in applied problems of magnetic monitoring, *Telecommunications and Radio Engineering*, 79(20), pp. 1801-1811.

37. Творошенко, И. С. (2010). Анализ процессов принятия решений в интеллектуальных системах. *Системи обробки інформації*, (2), 248-253.

38. Кучеренко, Є. І., Творошенко, І. С. (2011) Оперативне оцінювання простору станів складних розподілених об’єктів з використанням нечіткої інтервальної логіки. *Искусственный интеллект*. 2011. № 3. С. 382-387.

39. Andersson, O., & Reyna Marquez, S. (2016). A comparison of object detection algorithms using unmanipulated testing images: Comparing SIFT, KAZE, AKAZE and ORB.

40. Кобилін О.А., Творошенко І.С. Методи цифрової обробки зображень: навч. посібник. Харків: ХНУРЕ, 2021. 124 с.

41. Daradkeh Y.I., and Tvoroshenko I. (2020) Technologies for Making Reliable Decisions on a Variety of Effective Factors using Fuzzy Logic, *International Journal of Advanced Computer Science and Applications*, 11(5), pp. 43-50.

42. Темчур К.О. Аналіз сучасного стану розробки програмних застосунків для розпізнавання та класифікації кулінарних страв. *Радіоелектроніка та молодь у XXI столітті*: тези доповідей 25-го Міжнародного молодіжного форуму (Харків, 20–21 квітня 2021 р.). Харків: ХНУРЕ, 2021. Т. 7, 10. С. 20-21.

43. Темчур К.О. Аналіз сучасного стану розробки програмних застосунків для розпізнавання та класифікації кулінарних страв. *XV Регіональна студентська науково-технічна конференція «Наука – перші кроки»*: тези доповідей: в 4 т. Т. 2. Маріуполь: ПДТУ, 2021. С. 45-46.

44. Tvoroshenko I., and Temchur K. (2021) Features of software application development for food recognition using deep machine learning methods, *Abstracts of XXVI International Scientific and Practical Conference «Topical issues of practice and science» (May 18-21, 2021). London, Great Britain*, pp. 680-684.