МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри

_____ Ігор ШЕЛЕХОВ
(підпис)

11 грудня 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня магістр

зі спеціальності 122 – Комп'ютерних наук,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційна технологія автоматизації ціноутворення та калькуляції у закладах громадського харчування»
здобувача групи ІН.мдн – 21к Соловйова Андрія Володимировича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Андрій СОЛОВЙОВ
(підпис)

Керівник
доцент, к.т.н. Ігор ШЕЛЕХОВ _____
(підпис)

Суми – 2023

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

Затверджую:
В. о. зав.кафедрою _____
_____
"_____" _____ 20___р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 - Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН.мдн – 21к Соловйова Андрія Володимировича

1. Тема роботи *Інформаційна технологія автоматизації ціноутворення та калькуляції у закладах громадського харчування*

затверджую наказом по інституту від " 20 " листопада 2023 року № _1308-VI_

2. Термін здачі здобувачем вищої закінченої роботи до 13 грудня 2023року

3. Вхідні данні до роботи.

Необхідно розробити інформаційну технологію, яка здатна детектувати страви на фото, виконати розробку та навчання моделей глибокого навчання для виявлення та локалізації об'єктів на зображеннях, інтеграцію з месенджером Telegram для зручного доступу користувачів, а також тестування та оптимізацію цієї технології.

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)

1) Аналіз проблеми. Огляд існуючих рішень. Огляд алгоритмів. 2) Постановка задачі й формування завдань дослідження. 3) Вибір методів рішення задачі. 4) Моделювання та проєктування системи. 5) Практична реалізація.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____
_____

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 28 вересня 2023 року

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту (роботи)	Термін виконання проекту (роботи)	Примітка
1.	Аналіз проблеми. Огляд аналогів розроблюваного програмного продукту. Огляд алгоритмів.	*25.09.2023*	*виконано*
2.	Постановка задачі та вибір методів реалізації.	29.09.2023	*виконано*
3.	Проєктування та моделювання програмного продукту.	25.10.2023	*виконано*
4.	Практична реалізація.	15.11.2023	*виконано*
5.	Оформлення пояснювальної записки до дипломної роботи.	25.11.2023	*виконано*

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 58 стор., 31 рис., 3 додатків, 20 джерел.

Обгрунтування актуальності теми роботи — економія часу та ресурсів,оптимізація управління , адаптація до ринкових умов,залучення клієнтів через інновації, спрощення бізнес – процесів.

Об'єкт дослідження — процеси ціноутворення, калькуляції та автоматизації у сфері громадського харчування, процеси детектування страв з використанням підходів глибокого навчання

Предмет дослідження — моделі та методи інформаційної технології ціноутворення, калькуляції та автоматизації у сфері громадського харчування.

Мета роботи — створення та впровадження інформаційної технології, яка здатна детектувати страви на фото. Це включає в себе розробку та навчання моделей глибокого навчання для виявлення та локалізації об'єктів на зображеннях, інтеграцію з месенджером Telegram для зручного доступу користувачів, а також тестування та оптимізацію цієї технології.

Результати — виконано вибір методів вирішення поставленої задачі; на основі мови Python з використанням моделей глибокого навчання, розроблено технологію, що може детектувати їжу зображеннях, автоматизовано процес ціноутворення.

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ, ЕКСТРАКТОР ОЗНАК,
TELEGRAM BOT API, НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ,
ЗАКЛАДИ ГРОМАДСЬКОГО ХАРЧУВАННЯ.

ЗМІСТ

ВСТУП

У сучасному світі, де технології постійно розвиваються, важливо виявляти і впроваджувати нові підходи до управління та обслуговування в галузі громадського харчування. Однією з таких новітніх технологій є автоматизоване ціноутворення та калькуляція, яка використовує засоби машинного навчання для детектування та класифікації їжі. Ця технологія не тільки є актуальною та необхідною для оптимізації бізнес-процесів у закладах громадського обслуговування, але також має значущі переваги як для самого закладу, так і для його клієнтів.

У сучасному ритмі життя, де швидкість та ефективність грають ключову роль, автоматизація ціноутворення та калькуляції стає важливою частиною ефективного управління закладом. За допомогою машинного навчання, система може автоматично розпізнавати страви та розраховувати їх вартість, забезпечуючи точність та швидкість в процесі управління витратами.

Технологія також здатна враховувати фактори, які можуть впливати на ціноутворення, такі як зміна вартості інгредієнтів або сезонність продуктів. Це дозволяє бізнесу бути гнучким та швидко адаптуватися до змін на ринку, забезпечуючи конкурентні переваги.

Новизна цієї технології полягає у здатності системи відновлювати та оптимізувати процес ціноутворення та калькуляції на основі навчання на реальних даних. Це дозволяє закладам громадського харчування не тільки автоматизувати рутинні процеси, а й постійно пристосовувати свою стратегію до змін в ринкових умовах [1].

У результаті впровадження технології автоматизованого ціноутворення та калькуляції за допомогою машинного навчання стає ефективним кроком уперед для громадських закладів харчування. Це сприяє не лише підвищенню прибутковості та оптимізації бізнес-процесів, але й створює комфортне та інноваційне середовище для клієнтів, підвищуючи рівень їхньої задоволеності від обслуговування.

Впровадження штучного інтелекту у різні сфери, включаючи громадське харчування, важливе з погляду покращення ефективності та конкурентоспроможності. Машинне навчання дозволяє системам навчатися на основі даних та вдосконалюватися з часом, що робить їх більш адаптивними та точними. У галузі обслуговування клієнтів, впровадження технології автоматизованого ціноутворення та калькуляції через машинне навчання покращує якість обслуговування та робить його більш індивідуалізованим.

1 ІНФОРМАЦІЙНИЙ ОГЛЯД

1.1 Дослідження предметної області

У сучасному світі галузь громадського харчування переживає значні зміни, спричинені впровадженням інформаційних технологій. Ресторани та кафе все більше використовують програмні продукти для автоматизації рутинних операцій, таких як ціноутворення та калькуляція, з метою оптимізації управління та покращення обслуговування клієнтів. У цьому розділі буде проведено докладний аналіз предметної області з акцентом на існуючі виклики та можливості.

Галузь громадського харчування в умовах сучасного світу стикається з викликами оптимізації процесів управління, впровадження новаторських рішень та поліпшення користувацького досвіду. Інформаційні технології надають унікальні можливості для досягнення цих цілей, зокрема, у впровадженні технології детектування їжі.

Технологія детектування їжі базується на використанні алгоритмів машинного навчання та комп'ютерного зору. Останні розробки в цій області дозволяють розпізнавати не лише види страв, але і їхні складові частини, ступінь прожарювання та подачу. Це відкриває широкі перспективи для покращення взаємодії клієнтів із закладами громадського харчування.

Інформаційні технології в галузі громадського харчування вирішують ряд проблем та пропонують нові можливості. Тенденції, які домінують в цьому сегменті, включають в себе використання хмарних технологій для збереження та обробки даних, інтеграцію систем для покращення взаємодії між різними етапами обслуговування, а також впровадження рішень на базі штучного інтелекту для аналізу великих обсягів інформації [2].

Завдання управління громадським харчуванням стає складнішим через зростання конкуренції, зміни в уподобаннях споживачів та необхідність ефективного використання ресурсів. Дослідження визначає ключові виклики,

такі як нестабільність вартості інгредієнтів, складність управління персоналом та потреба у точних методах ціноутворення.

Переваги для Закладу:

1. Оптимізація витрат. Автоматизоване ціноутворення дозволяє точно визначати вартість приготування кожної страви, що веде до оптимізації витрат та максимізації прибутковості закладу.

2. Швидкість та ефективність. Зменшення часу, витраченого на ручний розрахунок цін та калькуляцію, дозволяє працівникам закладу зосередитися на інших аспектах обслуговування клієнтів.

3. Гнучкість. Система може легко адаптуватися до змін у рецептурах або цінах на інгредієнти, роблячи процес управління більш гнучким.

Переваги для клієнтів:

1. Точність та прозорість. Клієнти отримують точні та прозорі ціни на страви, що сприяє довірчості та задоволенню від обслуговування.

2. Швидкість обслуговування. Замовлення обробляються швидше завдяки автоматизованому ціноутворенню, що полегшує та прискорює обслуговування.

Застосування технології детектування їжі в чат-ботах вносить інновації у галузь громадського харчування. Клієнти отримують візуальне підтвердження свого вибору, що сприяє збільшенню задоволення від вибору та стимулює їх до знову замовлення. З точки зору закладу громадського харчування, це є конкурентною перевагою та можливістю покращити взаємодію з клієнтами.

1.2 Огляд аналогів розроблюваного програмного продукту

В контексті розробки інформаційної технології автоматизації ціноутворення та калькуляції у закладах громадського харчування, аналоги на ринку, які використовують технології детектування їжі, відіграють ключову роль. Нижче подано огляд деяких готових програмних продуктів у цій області:

1. Yummly (https://www.yummly.com/) на рисунку 1.1.

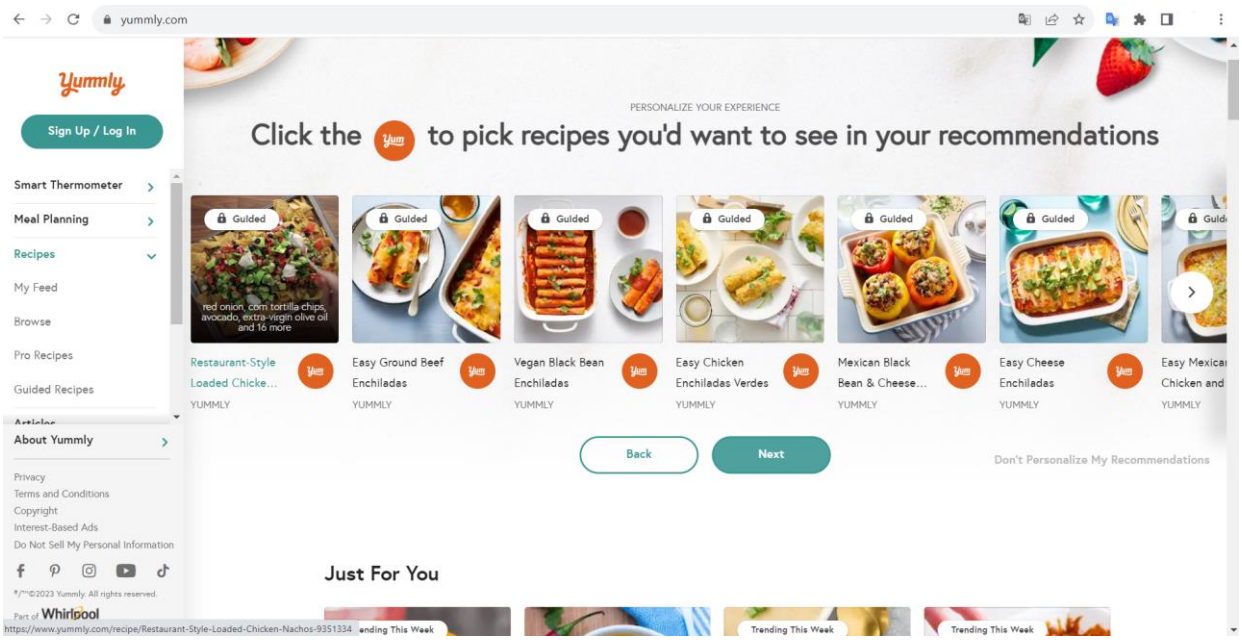

Рисунок 1.1 — Головна сторінка Yummly

Переваги:

– точне розпізнавання страв на основі фотографій;

– велика база рецептів та можливість підбору страв за інгредієнтами.

Недоліки:

– не має функціоналу для автоматизації ціноутворення та калькуляції;

– фокусується на рецептах, а не на взаємодії з закладами громадського харчування.

2. Calorielab (https://www.calorielab.com/)

Переваги:

– детальна інформація про калорійність та харчовий склад страв.

– велика база даних продуктів та страв для вибору.

Недоліки:

– відсутність функціоналу для автоматизації ціноутворення;

– не спрямований на взаємодію з закладами громадського харчування.

3. MyFitnessPal (https://www.myfitnesspal.com/). Головну сторінку зображено на рисунку 1.2.

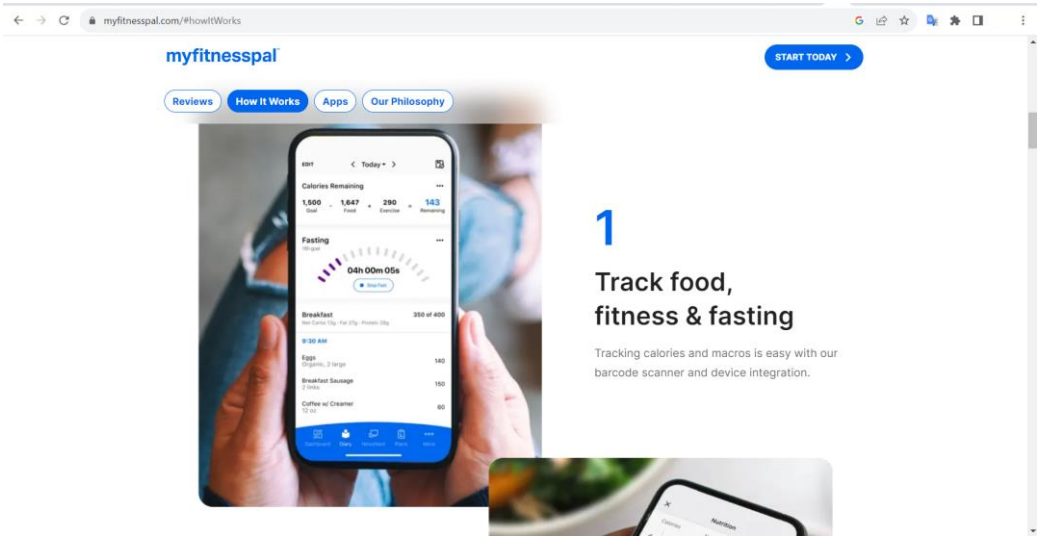

Рисунок 1.2 — Головна сторінка MyFitnessPal

Переваги:

– зручний інтерфейс для введення та відстеження спожитих продуктів;

– можливість взаємодії з іншими користувачами та отримання порад.

Недоліки:

– не використовує технології детектування їжі на фото;

– обмежений функціонал для закладів громадського харчування.

4. SeeFood (https://www.seefoodapp.com/). Головну сторінку зображено на рисунку на рисунку 1.3.

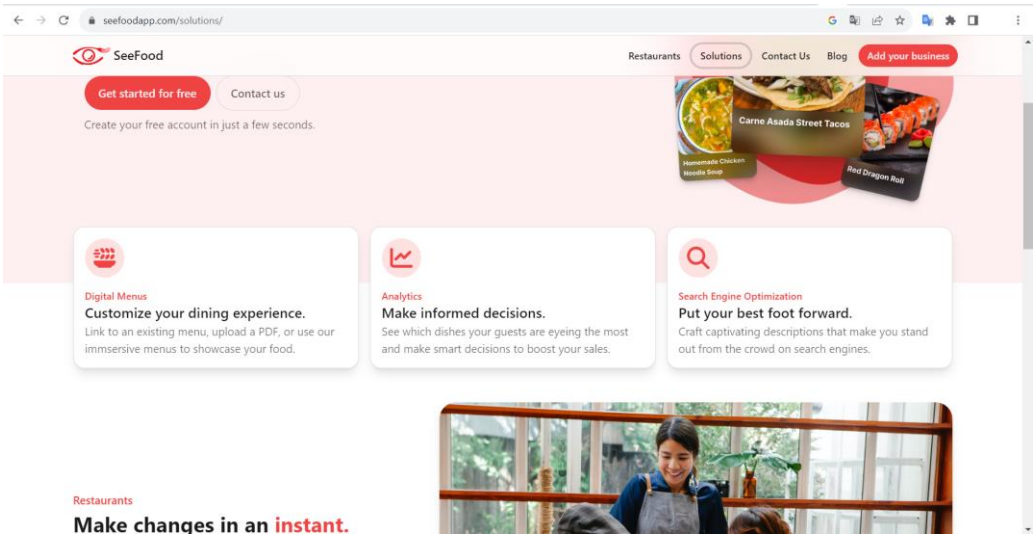

Рисунок 1.3 — Головна сторінка SeeFood

Переваги:

– технологія розпізнавання їжі на зображеннях для визначення калорій та складу;

– спрямований на взаємодію з ресторанами та кафе.

Недоліки:

– обмежений функціонал для автоматизації ціноутворення;

– поки не є популярним у широкому розумінні.

Отже, аналіз готових програмних продуктів, що використовують детектування їжі, показує, що вони в основному спрямовані на здоров'я та харчові пристрасті. Проте, жоден з них не повністю задовольняє потреби у впровадженні технологій ціноутворення та калькуляції у закладах громадського харчування. Розробка програмного продукту, що поєднує в собі обидві ці функціональності, може бути значущим внеском у цю галузь.

1.3 Огляд алгоритмів

Детектування страв на фотографіях та автоматизація ціноутворення у закладах громадського харчування через чат-бот може бути реалізована за допомогою наступного плану:

1. Збір та підготовка даних:

– зібрати великий набір фотографій страв різних типів і категорій. Наприклад можна частково використати датасет Food-101 — це набір даних, який містить зображення 101 виду їжі. Кожна категорія має по 1000 зображень. Також на платформі Kaggle є багато конкурсів та наборів даних, пов'язаних із зображеннями їжі. Можна розглянути доступні набори даних та конкурси за пошуком ключових слів «food», «eat» та інші. Також можна використати інші загальнодоступні джерела для збору інформації;

– розмічати власні фото, які будуть безпосередню використовуватися при експлуатації системи.

2. Навчання моделі детектування:

– використовувати глибокі нейронні мережі для створення моделі

розпізнавання страв на фото (наприклад, з використанням Convolutional Neural Networks — CNN, Recurrent Neural Network — RNN, автоекодерів, трансформерів та інші типи) [4];

– розглянути варіант використовувати попередньо навчених моделей визначення об’єктів на фото для так званого механізму трансферу знань.

3. Розробка механізму взаємодії з чат-ботом:

– розробити інтерфейс та логіку роботи чат-боту з користувачами для взаємодії з системою. Розглянути додатки та фреймворки існуючих рішень для побудови чат-ботів, або розглянути варіант написання власного додатку;

– у рішенні надати можливість завантаження фотографій користувачами.

4. Обробка та розпізнавання:

– після отримання фото від користувача, використовувати розроблену модель для детектування страв на фото;

– використовувати алгоритми пост-обробки результатів розпізнавання для видалення зайвих або дублюючих спрацювань.

5. Автоматизація ціноутворення:

– навчати систему розпізнавати страви та за допомогою програмних засобів створити інтеграцію системи з поточною системою калькуляції чеків, щоб система могла отримувати актуальні ціни та детальний склад страв, який з часом може дещо змінюватися (розрахунок співвідношення калорій і основних нутрієнтів у харчуванні: білків, жирів та вуглеводів)

– розробити алгоритми для автоматичного визначення вартості та енергетичної цінності усіх обраних страв на основі цих даних.

6. Тестування та оптимізація:

– провести тестування системи на реальних даних та користувацьких сценаріях;

– вносити корективи та оптимізовувати систему для покращення точності та швидкодії.

7. Впровадження та підтримка:

– запустити систему в роботу в обмеженому режимі;

– забезпечити підтримку та зворотний зв'язок для користувачів та закладу.

Цей план дозволить створити повноцінний продукт для автоматизації ціноутворення та калькуляції страв у закладах громадського харчування. Взаємодія з чат-ботом зробить процес замовлення та розрахунку зручним та ефективним для користувачів, а автоматизоване детектування страв спростить роботу персоналу закладів.

Необхідно також звертати увагу на заходи забезпечення безпеки та конфіденційності даних користувачів, оскільки в процесі взаємодії з фотографіями їжі можуть збиратися особисті дані. Регулярне оновлення моделей машинного навчання та забезпечення підтримки користувачів також є важливими етапами впровадження та експлуатації системи.

Загалом, цей план дозволить створити продукт, який поєднає в собі елементи машинного навчання, обробки зображень та чат-бот технологій для оптимізації процесів у галузі громадського харчування.

В контексті нашої задачі, детектування — це процес визначення та обмеження областей на фотографіях, які містять зображення страв у закладах громадського харчування. Основна мета полягає в автоматичному виявленні різних страв на зображеннях та наданні інформації про їхнє розташування. Це може включати в себе визначення контурів кожної страви та областей, що їх оточують, для подальшого аналізу та обробки даних, таких як ціноутворення та калькуляція. Детектування дозволяє системі точно визначати положення та кількість страв на зображеннях, щоб полегшити автоматизацію управління гастрономічним закладом.

Детектування страв на фотографіях є завданням у сфері комп'ютерного зору, і для його вирішення використовуються різні методи та техніки. Нижче представлені деякі основні методи, які можна використовувати для

детектування страв:

1. Використання згорткових нейронних мереж (Convolutional Neural Networks — CNN):

– CNN є ефективними для задач обробки зображень, і їх можна використовувати для розпізнавання об'єктів, включаючи страви. Принцип роботи згорткового шару наведено на рисунку 1.4;

– моделі, такі як VGG16 (рис. 1.5), ResNet, або MobileNet, які навчені на великих наборах даних, можуть бути використані або адаптовані для задачі детектування страв.

Згоркова нейронна мережа — це клас глибоких нейронних мереж, які найчастіше застосовуються для аналізу візуальних зображень. CNN — це вдосконалена версія багатошарового персептрона. Це клас глибокої нейронної мережі, натхненний зоровою корою людини [3].

Основна ідея між згортковими операціями полягає у виділенні ознак або, можна сказати, фільтрації. Пізніше мережа спробує порівняти всі можливі ознаки зі вхідного зображення з зображенням класу (клас — це ім’я об’єкта, яке ми хочемо розпізнати, наприклад: автомобіль).

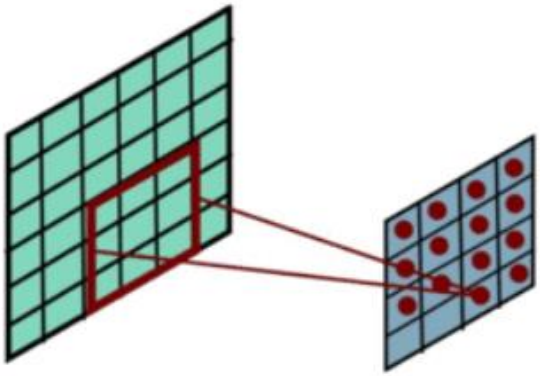

Рисунок 1.4 — Принцип роботи згорткового шару

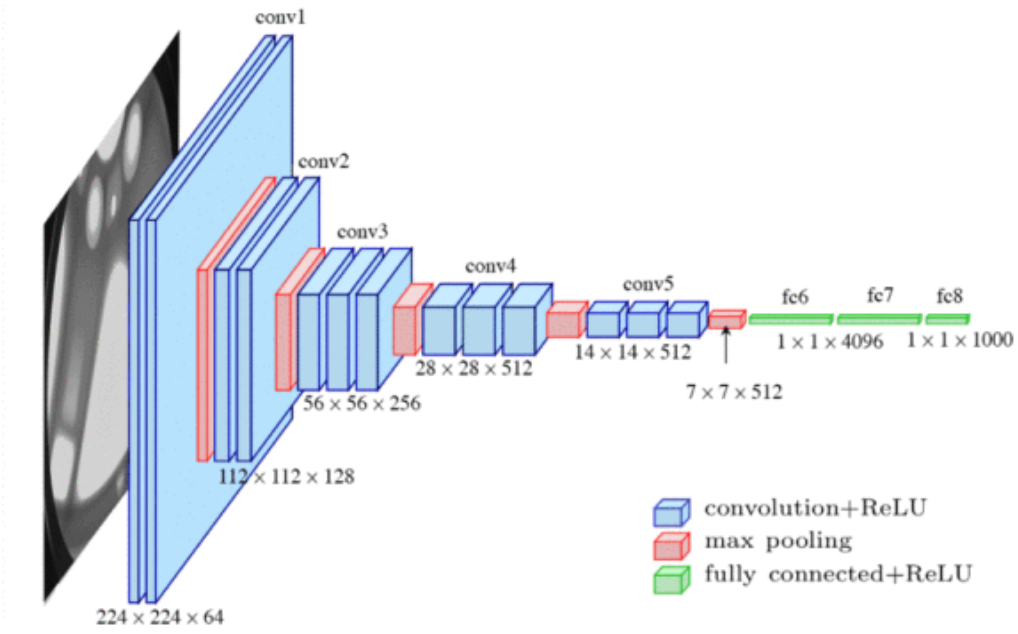

Рисунок 1.5 — Архітектура VGG16

Наприклад, вхідне зображення VGG16 має розмір 224×224 пікселів. На рисунку 1.6 наведено короткий опис архітектури VGG16:

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Рисунок 1.6 — Опис архітектури VGG16

Переваги VGG16: дуже проста архітектура з повністю з'єднаними шарами; легко інтерпретована та налаштовується.

До недоліків можна віднести: має велику кількість параметрів, що може призвести до великого обсягу пам'яті та обчислювальних витрат; швидко насичується навчальними даними.

Основною метою архітектури ResNet (рис. 1.7) є створення згорткової нейронної мережі з багатьма шарами для ефективного навчання.

Проблема глибокої згорткової нейронної мережі полягає в тому, що коли ми збільшуємо глибину мережі, виникає проблема затухаючого градієнта. Коли мережа заглиблюється, її продуктивність стає насиченою або навіть починає знижуватися точність.

ResNet розбиває більш глибоку мережу на трирівневі фрагменти та передає вхідні дані в кожен фрагмент безпосередньо до наступного блоку разом із залишковим виходом блоку за вирахуванням входу в блок, який повторно вводиться [5].

Переваги ResNet: введення блоків з відстеженням залишкового сигналу (residual connections) дозволяє робити навчання глибоких мереж без втрати якості та затухання градієнту; ефективна для навчання глибоких нейронних мереж.

Недоліки ResNet: ResNet може вимагати значного обсягу пам'яті для зберігання активацій і градієнтів під час навчання глибоких мереж. Це може ускладнити навчання на ресурсах з обмеженим обсягом пам'яті. ResNet може бути схильною до перенавчання (overfitting), особливо якщо навчальний набір має обмежену кількість зображень або якщо дані не репрезентативні.

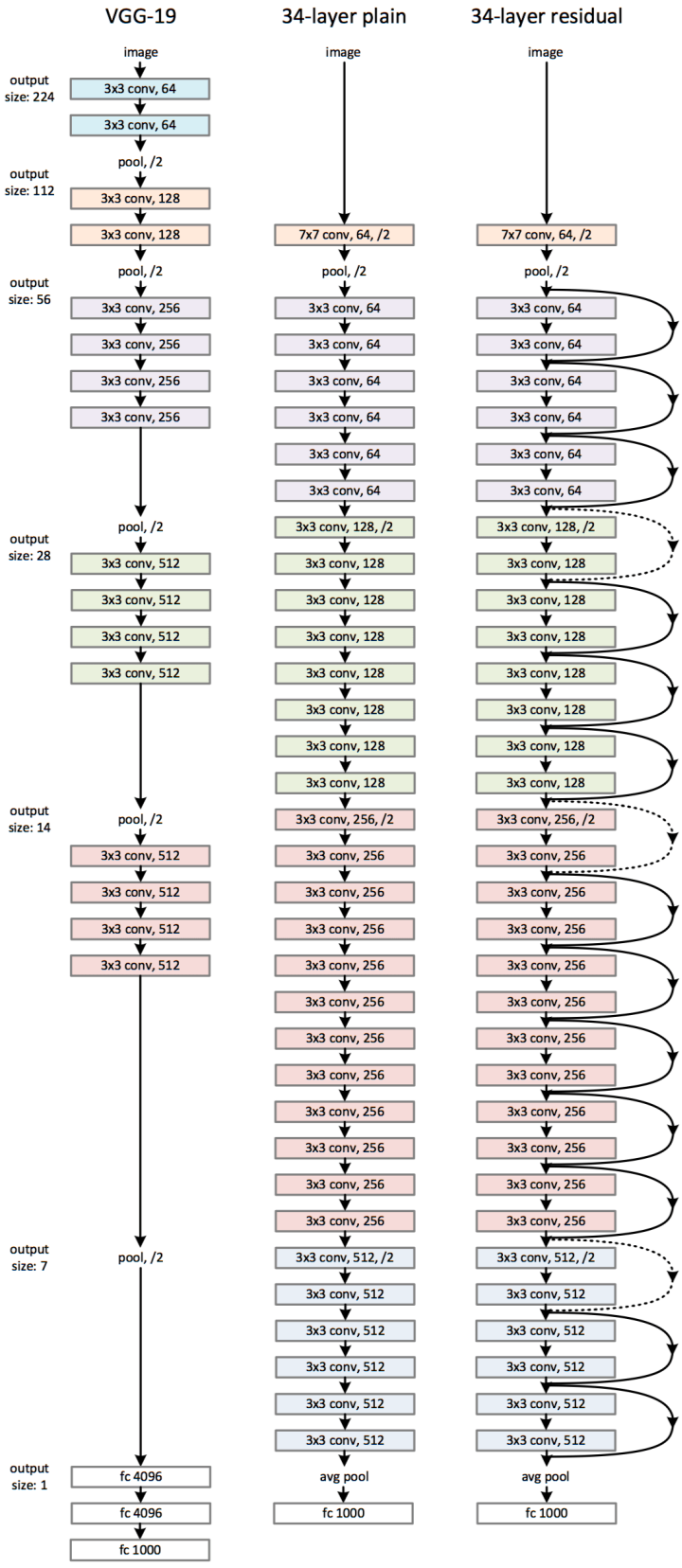

Рисунок 1.7 — Архітектура ResNet

MobileNet — це спрощена архітектура, яка використовує згортки, що розділяються по глибині, для побудови легких глибоких згорткових нейронних мереж і забезпечує ефективну модель для мобільних і вбудованих програм бачення [8]. Структура MobileNet базується на фільтрах, що розділяються по глибині, як показано на рисунку 1.8.

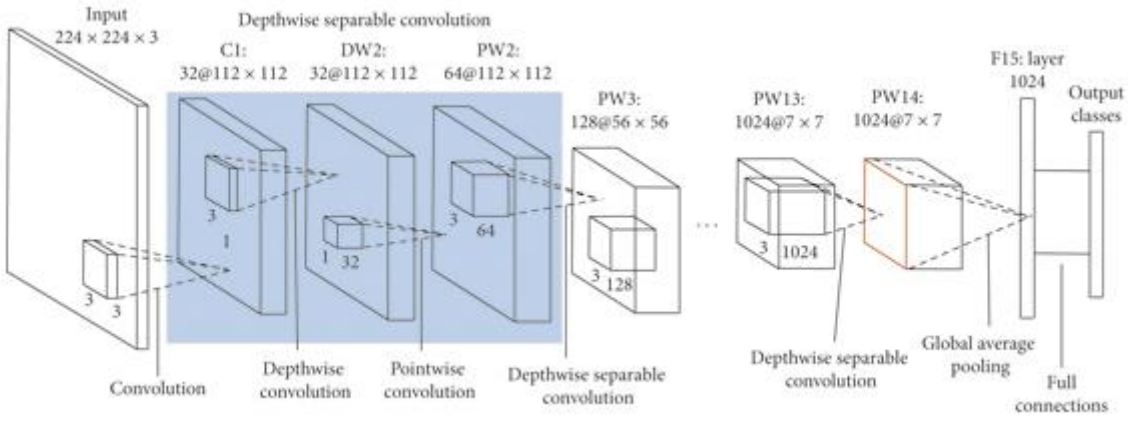

Рисунок 1.8 — Архітектура MobileNet

Переваги ResNet: спеціально призначена для мобільних пристроїв та вимагає менше ресурсів; застосовує глибоку сверточну архітектуру з точковими свертками (depthwise separable convolutions) для зменшення кількості параметрів.

Недоліки ResNet: може бути менш точною в порівнянні з більш потужними моделями, особливо для складних завдань.

Оскільки важливо мати легку та швидку модель, і враховуючи, що наша задача включає в себе обробку зображень страв, MobileNet може бути оптимальним варіантом. Він забезпечить ефективність на мобільних пристроях та в інтерфейсі чат-боту, зберігаючи при цьому достатню точність для детектування страв на фотографіях.

2. Методи регіонів з пропозиціями (Region-based CNN — R-CNN):

– R-CNN та його модифікації, такі як Fast R-CNN та Faster R-CNN,

дозволяють ефективно виявляти області зображення, які містять об'єкти (у цьому випадку, страви);

– ці методи спочатку визначають області з імовірністю містити об'єкт, а потім класифікують та точно розміщують ці об'єкти.

Архітектура R-CNN (Region-Based Convolutional Neural Network) (рис. 1.9) була розроблена в 2014 році Россом Гіршиком та іншими. Основним компонентом R-CNN є виявлення областей (regions) на зображенні та подальший аналіз кожної окремої області для визначення класу об'єкта та точного місця його розташування. Ознаки, отримані з кожної області, подаються в класифікатор та регресор для визначення класу об'єкта і точного розташування цього об'єкта [12]. Класифікатор визначає, якого класу є об'єкт, а регресор прогнозує корекції для координат області, щоб точно визначити положення об'єкта. Класифікація оброблених ознак відбувається за допомогою методу опорних векторів (SVM, Support Vector Machine) а уточнення меж регіонів за допомогою лінійної регресії.

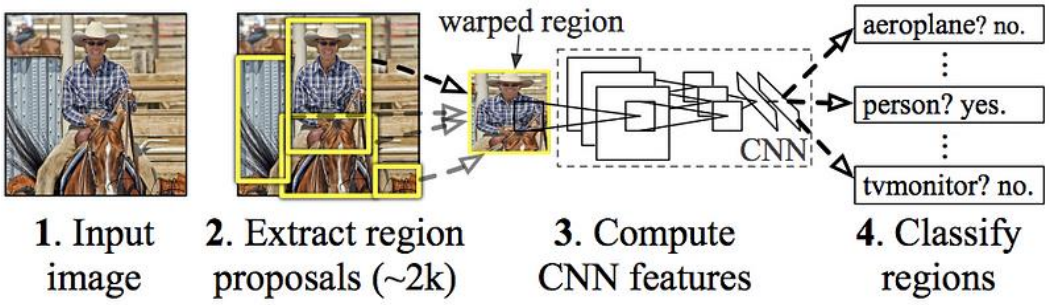

Рисунок 1.9 — Архітектура R-CNN

Недоліки R-CNN: повільність через потребу в обчисленнях для кожного окремого регіону; різноманітні зображення для обробки роблять R-CNN вимогливим до пам'яті; велика кількість параметрів і деталізована архітектура роблять навчання даної моделі досить витратним за ресурсами.

Недоліки R-CNN привели авторів у 2015 році до покращення моделі. Вони назвали її Fast R-CNN. В її основі лежить наступна архітектура

(див рис. 1.10). Зображення подається на вхід нейронної мережі і обробляється вибірковим пошуком. У підсумку маємо карту позначок і регіонів потенційних об'єктів. Координати регіонів потенційних об'єктів перетворюються в координати на карті позначок. Отримана карта з регіонами передається шару опитування RoI (Region of Interest). Тут на кожен регіон накладається сітка. Потім використовується MaxPolling для зменшення розмірності [9]. Так, усі регіони потенційних об’єктів мають однакову фіксовану розмірність. Отримані визнання подаються на вхід повнозв'язного шару (Fully-conectedlayer), який передається двом іншим повнозв'язковим шарам. Перший з функцією активації softmax визначає ймовірність належності класу, другий — межі (розміщення) регіону потенційного об'єкта.

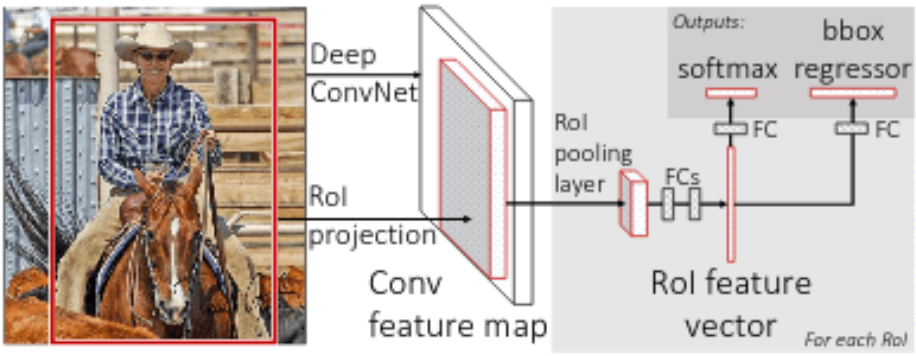

Рисунок 1.10 — Архітектура Fast R-CNN

Причина того, що Fast R-CNN швидший за R-CNN, полягає в тому, що не потрібно щоразу передавати 2000 пропозицій регіонів до згорткової нейронної мережі. Натомість операція згортки виконується лише один раз для зображення, і на основі цього створюється карта функцій.

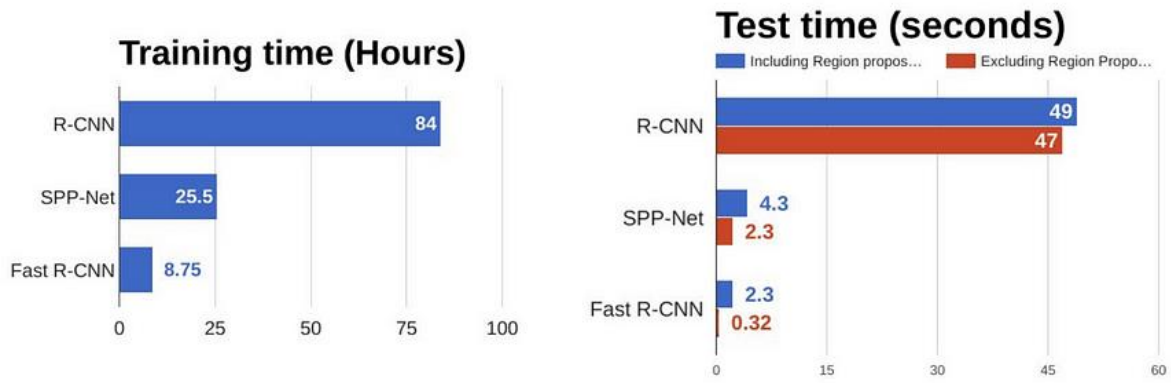

Рисунок 1.11 — Порівняння алгоритмів виявлення об'єктів

З наведених вище діаграм (рис. 1.11) можна зробити висновок, що Fast R-CNN значно швидший у сеансах навчання та тестування порівняно з R-CNN. Якщо дивитись на продуктивність Fast R-CNN під час тестування, включення пропозицій регіону значно сповільнює роботу алгоритму порівняно з невикористанням пропозицій регіону. Таким чином, регіональні пропозиції стають вузькими місцями в алгоритмі Fast R-CNN, що впливає на його продуктивність.

Faster R-CNN включає в себе RPN (Region Proposal Network) для автоматичного генерації пропозицій регіонів. Використовує єдину мережу для визначення пропозицій та обчислення класифікації ознак (рис. 1.12). Швидший та більш ефективний порівняно з попередніми версіями.

Недоліки Faster R-CNN: деяка складність у налаштуванні параметрів, що може вимагати досвіду та експертизи; використання RPN може збільшити вимоги до обчислювальних ресурсів.

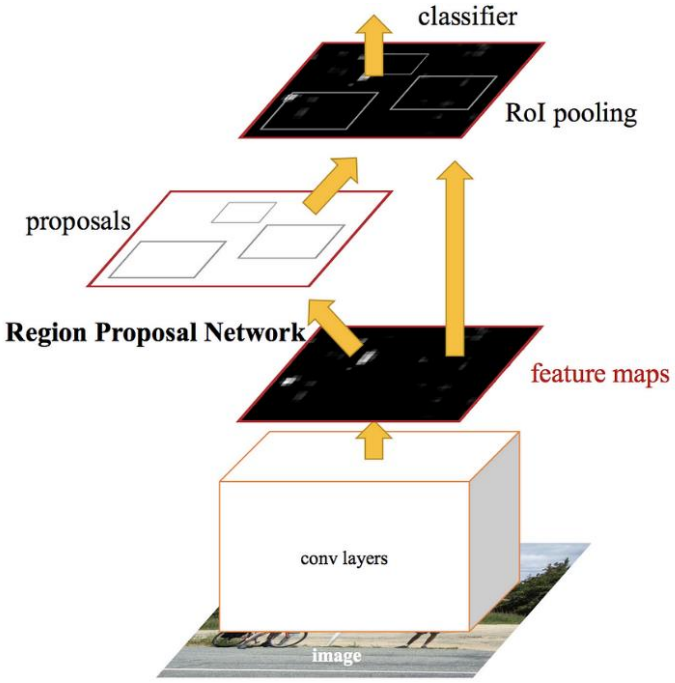

Рисунок 1.12 — Архітектура Faster R-CNN

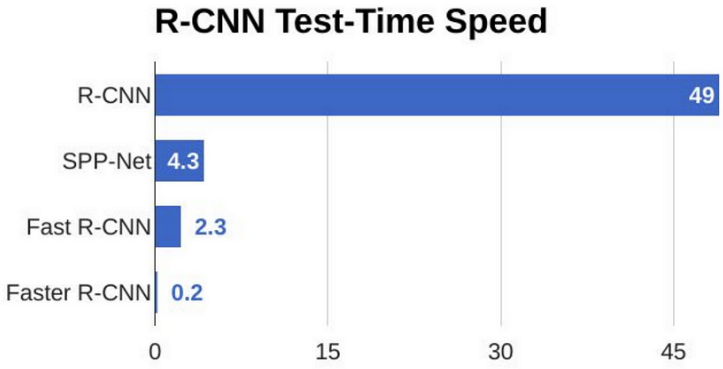

Рисунок 1.13 — Порівняння алгоритмів виявлення об'єктів

З наведеної вище діаграми (рис. 1.13) можна побачити, що Faster R-CNN набагато швидший за своїх попередників. Тому його можна навіть використовувати для виявлення об’єктів у реальному часі.

3. Методи обробки зображень та сегментації:

– методи, які використовують обробку зображень та сегментацію, можуть розділити зображення на окремі частини, ідентифікувати контури страв та визначати їхні межі;

– алгоритми сегментації, такі як U-Net чи Mask R-CNN, можуть

здійснювати точнішу сегментацію страв на зображеннях.

Обидва методи використовують згорточні нейронні мережі (CNN) для виконання сегментації зображення, але вони відрізняються своєю архітектурою та підходом до завдання.

Mask R-CNN (регіональна згорточна нейронна мережа з маскою) — це двоетапна структура, яка базується на моделі виявлення об’єктів Faster R-CNN.Mask R-CNN розширює модель Faster R-CNN, додаючи гілку для прогнозування масок об’єктів паралельно з існуючою гілкою для виявлення об’єктів [10]. Гілка маски використовує повністю згорнуту мережу для створення маски для кожного об’єкта на зображенні, яка потім використовується для отримання остаточної сегментації (рис. 1.14).

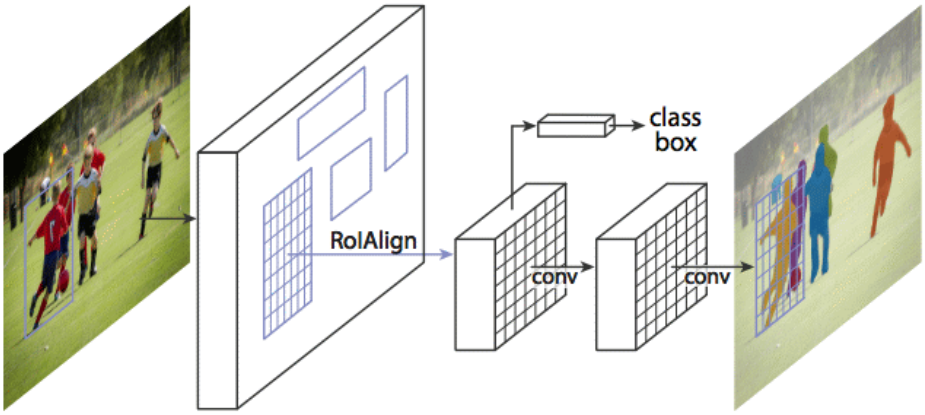

Рисунок 1.14 — Архітектура Mask R-CNN

Mask R-CNN має кілька переваг перед традиційними методами сегментації. По-перше, він може працювати з об’єктами складної форми та кількома екземплярами одного об’єкта. Він також здатний одночасно виявляти та класифікувати об’єкти, що зменшує загальні обчислювальні витрати. Нарешті, його можна навчити наскрізно за допомогою стандартних методів зворотного поширення.

4. Перенавчання (Transfer Learning) [11]:

– використання попередньо навчених моделей на великих наборах

зображень, таких як ImageNet, може значно полегшити процес навчання для детектування страв;

– можливе використання попередньо навчених моделей для виявлення загальних образів на зображеннях.

5. Детектування на основі ключових точок (Keypoint-based Detection):

– визначення ключових точок страв та їхніх характеристик може допомогти в детектуванні та розпізнаванні страв на фотографіях.

Важливо враховувати специфіку завдання та доступність великого обсягу даних для навчання моделей. Кожен метод має свої переваги та недоліки, і вибір залежить від конкретних умов та вимог проєкту.

1.4 Постановка задачі

Функціональні вимоги до чат-боту для детектування страв та визначення їх вартості:

1. Введення даних:

– завантаження фотографій страв: користувач може завантажити фотографії обраних страв зі свого пристрою або використовувати камеру для зйомки нових.

2. Детектування страв:

– автоматичне виявлення об'єктів: система повинна використовувати технологію детектування їжі для точного визначення та локалізації страв на фотографії.

3. Класифікація та ідентифікація страв:

– визначення виду та складу страв: чат-бот повинен класифікувати та ідентифікувати страви, використовуючи розпізнані компоненти.

4. Вартість та ціноутворення:

– автоматичне визначення ціни

5. Відображення результатів:

– показ результатів користувачу: чат-бот повинен надсилати користувачеві інформацію про визначені страви та їх вартість.

6. Інтерактивність та користувацький досвід:

– діалоговий інтерфейс: забезпечення інтуїтивно зрозумілого діалогового інтерфейсу для взаємодії з користувачем під час завантаження та виведення результатів.

– корекція результатів: можливість коригувати результати, якщо система допустила помилку або якщо користувач хоче внести зміни.

2 ВИБІР МЕТОДІВ ВИРІШЕННЯ ЗАДАЧІ

2.1 Вибір засобів програмування

У контексті створення інтегрованої системи для детектування страв та автоматизації ціноутворення в громадських закладах харчування, важливим є вибір засобів програмування та фреймворків.

Python визначається своєю широкою популярністю і зручністю в розробці. У нашому випадку, використання Python для створення чат-бота Telegram відбувається за допомогою фреймворку Telebot. Його можливості роблять його ідеальним інструментом для взаємодії з користувачами, зокрема, приймання фотографій для передачі до моделі машинного навчання [6].

Обрані бібліотеки та фреймворки (аналіз яких буде проведено далі) для машинного навчання та веб-частини системи, Telebot для розробки інтерфейсу Телеграм-боту, утворять збалансований та потужний інструментарій для створення детектора страв та автоматизації ціноутворення в громадських закладах харчування. Цей вибір відзначається не лише технічною ефективністю, але і зручністю в інтеграції з різними компонентами системи, що дозволить створити ефективний та зручний продукт для кінцевого користувача.

2.2 Аналіз та вибір алгоритмів розроблення

Для реалізації системи автоматизації ціноутворення та калькуляції у закладах громадського харчування з використанням технології детектування їжі можна використати ряд спеціалізованих бібліотек та фреймворків. Нижче будуть розглянуті можливі варіанти для кожного етапу розробки [12]:

1. Детектування їжі:

– TensorFlow — це потужний фреймворк машинного навчання. Розроблений компанією Google, TensorFlow є однією з найпопулярніших бібліотек для глибокого навчання та створення нейронних мереж. Він надає велику гнучкість та широкі можливості для вирішення завдань машинного

навчання. Моделі, побудовані на TensorFlow, можуть використовувати попередньо навчені нейронні мережі для детектування об'єктів, включаючи їжу [3];

– Keras — високорівневий інтерфейс для TensorFlow та інших бібліотек глибокого навчання. Він дозволяє швидко створювати та навчати моделі нейронних мереж з мінімальними зусиллями.

2. Класифікація страв:

– MobileNet — легка та швидка архітектура нейронної мережі, спеціально розроблена для мобільних та вбудованих пристроїв. Може бути використана для класифікації страв.

3. Аналіз ціноутворення та калькуляції:

– Pandas — бібліотека для роботи з даними, дуже зручна для обробки та аналізу табличних даних. Можна використовувати для обробки цін на інгредієнти та розрахунку вартості.

– NumPy — бібліотека для наукових обчислень в Python. Забезпечує ефективні операції з числовими масивами, що може бути корисно при математичних обчисленнях для калькуляції вартості [7].

4. Реалізація серверної та клієнтської частин:

– Flask або Django — популярні фреймворки для розробки веб-додатків на Python. Flask — легший і простіший для малих проектів, Django — повнофункціональний фреймворк для великих застосунків.

5. Реалізація чат-боту:

Dialogflow або Microsoft Bot Framework. Dialogflow — сервіс Google для створення чат-ботів з підтримкою мови. Microsoft Bot Framework — фреймворк для розробки чат-ботів, який працює на платформі Microsoft.

6. Інші допоміжні бібліотеки:

OpenCV — бібліотека для комп'ютерного зору, яка може використовуватися для обробки зображень та роботи з відео. Допоможе в реалізації функцій обробки зображень для покращення детектування.

3 ПРОЄКТУВАННЯ ТА МОДЕЛЮВАННЯ

3.1 Структурно-функціональне моделювання

IDEF0 (Integration Definition for Information Modeling) — це методологія моделювання функцій, яка дозволяє створювати візуальні моделі взаємозв'язків та взаємодій між функціями в складних системах [17]. Цей підхід часто використовується для аналізу та проектування систем, а також для вдосконалення їхнього функціонування. Розглянемо, як використовується IDEF0 (рис. 3.1) в розробці та моделюванні системи детектування страв та автоматизації ціноутворення в громадських закладах харчування через чат-бот у Телеграмі.

IDEF0 дозволяє розбити велику систему на менші функціональні блоки та визначити їхні взаємозв'язки. В контексті нашого завдання, можна використовувати IDEF0 для розкриття функцій, таких як детектування страв, ціноутворення, взаємодія з користувачем тощо.

Кожен блок в IDEF0 відображає конкретну функцію системи. Для нашої системи це можуть бути блоки, такі як «Детектування страв», «Ціноутворення»

Рисунок 3.1 — IDEF0

У методології IDEF1 (Integration Definition for Information Modeling) моделювання інформації в системі використовується для подання детального

уявлення про обмін інформацією між функціональними блоками, які були ідентифіковані за допомогою IDEF0. Далі буде розглянуто як можна використовувати IDEF1 (рис. 3.2) для моделювання обміну інформацією в системі детектування страв та автоматизації ціноутворення в громадських закладах харчування через чат-бот у Телеграмі.

1. Визначення інформаційних потоків:

– В IDEF1 інформаційні потоки представлені стрілками, які вказують напрямок обміну інформацією між різними функціональними блоками. Для системи детектування страв це може означати потік фотографій страв від користувача до функції «Детектування страв».

2. Опрацювання інформації:

– Кожен функціональний блок має власні процеси обробки інформації. Наприклад, функція «Детектування страв» може використовувати алгоритми машинного навчання для аналізу фотографій та визначення видів страв.

3. Розподіл інформації:

– IDEF1 дозволяє визначити, як інформація розподіляється між різними частинами системи. Наприклад, результати детектування страв можуть передаватися функції «Ціноутворення» для формування вартості страв.

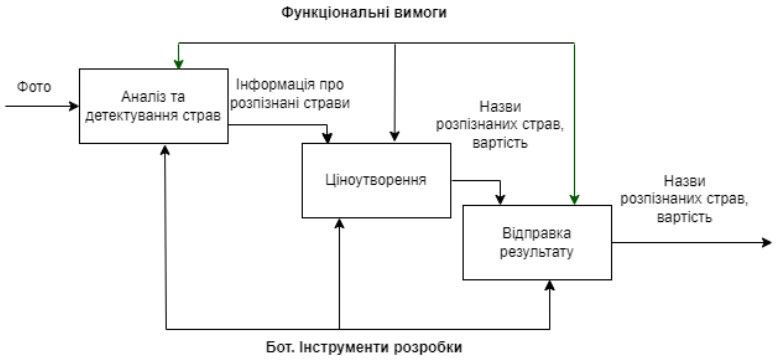

Рисунок 3.2 — IDEF1

Використання IDEF1 дозволяє детально моделювати інформаційні потоки та їх взаємодії в системі, що полегшує аналіз та оптимізацію обміну інформацією.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ

4.1 Реалізації основних компонентів

Нейронні мережі можуть бути навчені за допомогою різних методів, кожен з яких має свої особливості та призначення. Найбільш поширені і ті які підходять для виконання поставленого завдання це такі:

1. Навчання з учителем (Supervised Learning): Це найбільш поширений тип навчання, де модель тренується на даних, які вже містять відповіді (мітки). Задача моделі - навчитися прогнозувати відповіді на основі вхідних даних [13].

2. Навчання без учителя (Unsupervised Learning): У цьому випадку модель навчається на нерозмічених даних, тобто без явних відповідей або міток. Задача моделі - знайти приховані структури або закономірності у даних. Типові завдання включають кластеризацію (групування подібних об'єктів) та зниження розмірності (наприклад, PCA).

3. Слабокероване навчання (Semi-Supervised Learning): Це підхід, що поєднує елементи навчання з учителем і без учителя. Моделі тренуються на частково розмічених даних, де деякі зразки містять мітки, а інші - ні. Це корисно, коли розмітка даних є трудомісткою або дорогою.

4. Навчання з підкріпленням (Reinforcement Learning): У цьому випадку модель (агент) навчається, взаємодіючи з навколишнім середовищем. Агент вчиться виконувати дії таким чином, щоб максимізувати певну винагороду. Навчання з підкріпленням широко застосовується в задачах, де потрібно вчитися стратегіям або політикам поведінки, наприклад, у іграх або робототехніці [14].

5. Перенос знань (Transfer Learning): Цей підхід використовує моделі, які вже були навчені на одній задачі, для вирішення іншої схожої задачі. Трансфер знань дуже ефективний, коли доступно мало даних для нової задачі.

Враховуючи всі вище перераховані методи було вирішено скористатися останнім методом, методом переносу знань, оскільки задача розпізнавання

страв в цілому не нова і інші дослідники та компанії вже навчали подібні моделі для детектування та класифікації страв.

Було вирішено обрати фреймворк для навчання Tensorflow, так як це один із найбільш порширених та популярних фреймворків для машинного навчання. В мережі інтернет було знайдено модель класифікатора страв, яка навчалась на достаньо великому об’ємі даних приблизно 7000 тисяч оригінальних зображень і містила 25 класів (рис. 4.1).

```
kaathi_rolls     690
pizza            830
chapati          210
pakode           240
kulfi            300
chai             180
jalebi           165
momos            250
masala_dosa      230
idli             220
kadai_paneer     200
fried_rice       200
chole_bhature    230
paani_puri       250
samosa           250
butter_naan      250
hamburger        330
dal_makhani      160
pav_bhaji        310
dhokla           240
egg              150
tomato           150
cheese           201
french_fries     295
fom_tam          250
```

Рисунок 4.1 — Класи моделі класифікатора страв

Цей класифікатор був побудований на мережі з архітектурою MobilenetV2, тому було досить просто зробити трансфер вагових коефіцієнтів за назвою шарів нейроної мережі, приклад коду як це можливо зробити.

Приклади розпізнавання данних з попередньо навченої моделі для трансферу знань зображено на рисунках 4.2-4.4.

```
import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2

from keras.models import load_model
# Load or create your source MobileNetV2-based classification model
source_model = load_model('origin_model.h5')

# Create the target MobileNetV2 backbone model
```

```
target_model = MobileNetV2(weights=None, include_top=False)

# Create a dictionary of source model layers
source_layers = {layer.name: layer for layer in source_model.layers}

# Transfer weights from source to target model by layer name
for layer in target_model.layers:
    if layer.name in source_layers and
source_layers[layer.name].get_weights():
        # Check if the layer shapes are compatible
        if source_layers[layer.name].get_weights()[0].shape ==
layer.get_weights()[0].shape:
            layer.set_weights(source_layers[layer.name].get_weights())
```

Рисунок 4.2 — Приклад розпізнавання данних

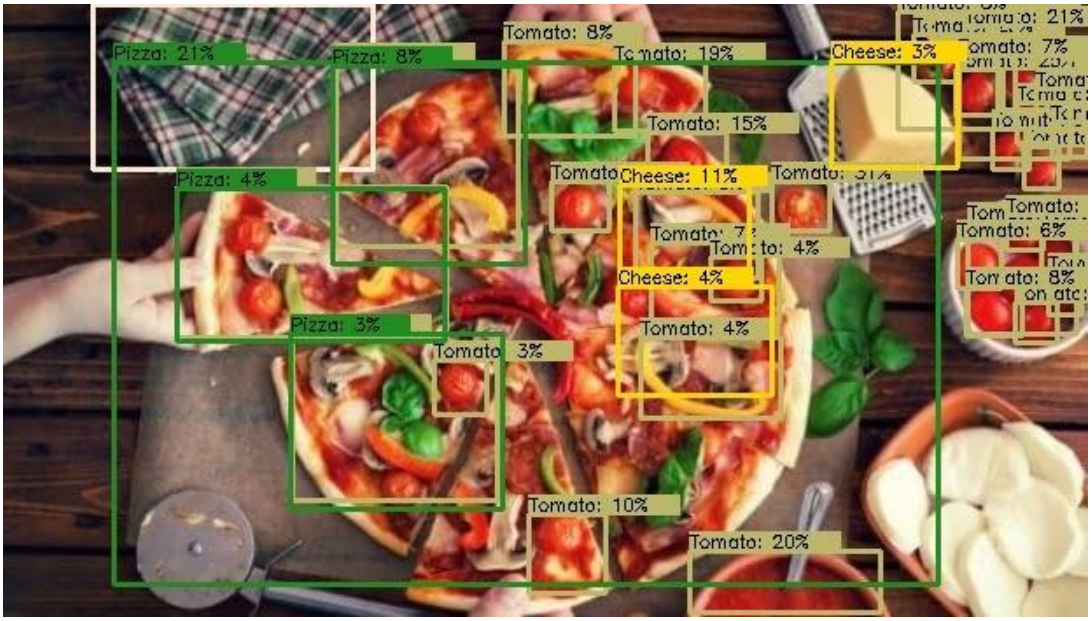

Рисунок 4.3 — Приклад розпізнавання данних

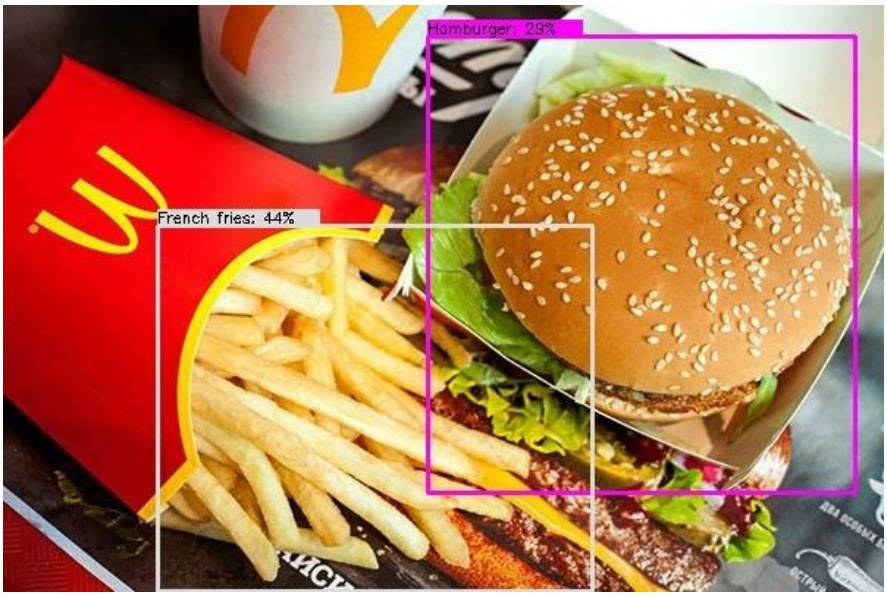

Рисунок 4.4 — Приклад розпізнавання данних

Перш за все необхідно зібрати датасет та перетворити його до формату tfrecord для більш швидкого та зручного навчання [15].

Ось код функції яка це робить.

```
def build_tfrecords(dataset, output_path, images_annotations,
category_index):
    writer = tf.compat.v1.python_io.TFRecordWriter(output_path)
    label_map_dict = label_map_util.get_label_map_dict(LABEL_PATH)

    print("Building TFRecord files for dataset: %s" % dataset)

    for idx, (image, _annotations) in enumerate(images_annotations):
        if idx % 100 == 0:
            print('%d of %d annotations' % (idx,
len(images_annotations)))

        _, tf_example, num_annotations_skipped = create_tf_example(
            image=image,
            annotations_list=_annotations,
            image_dir=IMAGES_DIR,
            category_index=category_index,
            include_masks=False
        )

        writer.write(tf_example.SerializeToString())

    writer.close()
    print("Done!")
```

Приклад навчальних даних представлено на рисунку 4.5.

Рисунок 4.5 — Приклад навчальних даних

Далі необхідно створити модель детектору, добудувати верхівку моделі MobileNetV2, щоб навчити детектор страв.

Для добудови верхівки детектора до моделі MobileNetV2, потрібно додати додаткові шари, які будуть відповідати за виявлення та класифікацію об'єктів у зображеннях. Модель MobileNetV2 зазвичай використовується як ефективний бекбон для екстракції ознак зображення, але для детектування об'єктів потрібні додаткові компоненти [16]. Ось основні спеціалізовані шари, які можуть бути додані для добудови верхівки SSD детектора страв:

1. Шари для виявлення обмежувальних рамок: це можуть бути згорткові шари, які передбачають координати обмежувальних рамок для кожного об'єкта.

2. Шари для класифікації: це шари, які визначають клас кожного об'єкта у передбаченій обмежувальній рамці.

3. Шари для поліпшення точності: наприклад, шари Non-Maximum

Suppression (NMS) для усунення надмірних обмежувальних рамок.

Приклад коду для створення SSD детектору на основі MobileNetV2:

```
def create_ssd_model(num_classes, backbone=MobileNetV2, image_size=(224,
224)):
    # Завантаження бекбону без верхніх шарів
    base_model = backbone(input_shape=(*image_size, 3),
include_top=False)
    base_model.trainable = False

    # Додавання шарів для детектування
    x = base_model.output
    x = Conv2D(4 * (num_classes + 4), kernel_size=3, padding='same')(x)
# 4 якорні рамки на клас

    # Формування виходу
    num_anchors = 4
    num_boxes = num_anchors * (num_classes + 4)
    x = Reshape((image_size[0] // 32, image_size[1] // 32, num_anchors,
num_classes + 4))(x)

    model = Model(inputs=base_model.input, outputs=x)
    return model
```

Також потрібно додати модифіковані функції втрат, щоб вони враховували як похибку класифкації страви так і похибку розташування на зображені. Ось приклад таких функцій втрат:

```
def smooth_l1_loss(y_true, y_pred):
    diff = tf.abs(y_true - y_pred)
    less_than_one = tf.cast(tf.less(diff, 1.0), tf.float32)
    loss = (less_than_one * 0.5 * diff ** 2) + (1 - less_than_one) *
(diff - 0.5)
    return loss

def ssd_loss(num_classes):
    def ssd_custom_loss(y_true, y_pred):
        # Розділення міток і передбачень на класифікацію та локалізацію
        y_true_cls, y_true_loc = y_true[..., :num_classes], y_true[...,
num_classes:]
        y_pred_cls, y_pred_loc = y_pred[..., :num_classes], y_pred[...,
num_classes:]

        # Втрата класифікації (кросс-ентропія)
        cls_loss = tf.keras.losses.categorical_crossentropy(y_true_cls,
y_pred_cls, from_logits=True)

        # Втрата локалізації рамки (Smooth L1)
        loc_loss = smooth_l1_loss(y_true_loc, y_pred_loc)

        # Комбінування втрат
        total_loss = cls_loss + loc_loss
        return total_loss
    return ssd_custom_loss
```

Також для зручності навчання налаштуємо зберігання чекпоінтів, та налаштуємо механізм ранньої зупинки навчання для запобігання переначанню (overfitting).

Код для цих налаштувань:

```
early_stopping = EarlyStopping(monitor='val_loss', patience=5,
verbose=1)

# Налаштування Model Checkpoint
# Це збереже модель після кожної епохи, де спостерігається покращення в
метриці
model_checkpoint = ModelCheckpoint(
    'ssd_model_checkpoint.h5', monitor='val_loss', verbose=1,
save_best_only=True, save_weights_only=False)
```

Далі виконуємо навчання моделі та оцінюємо результати на валідаційній вибірці. Графіки залежності похибок від кількості ітерацій можна побачити на рисунку 4.6

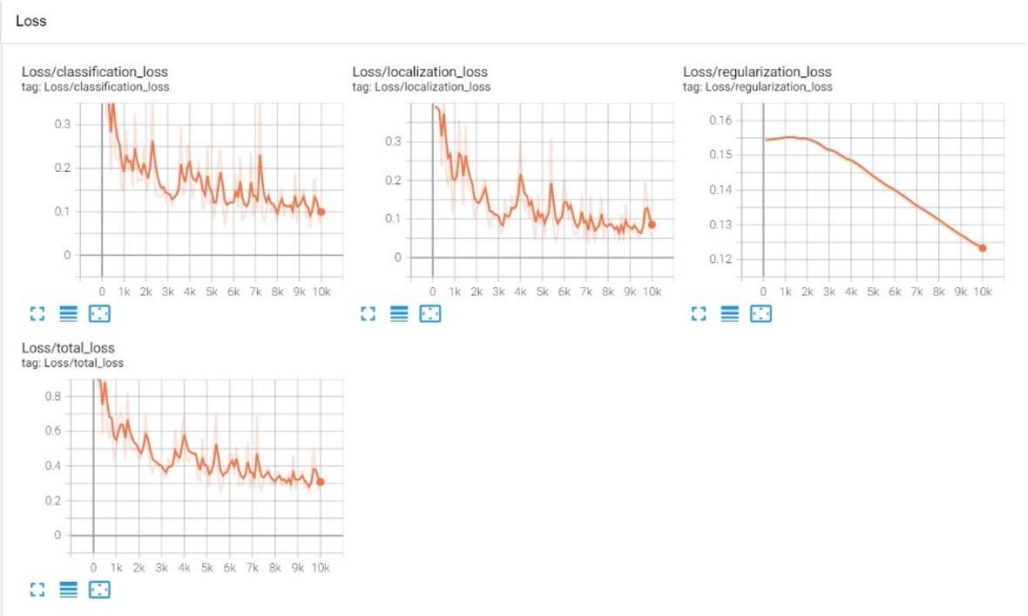

Рисунок 4.6 — Графіки залежності похибок від кількості ітерацій

Оцінимо результати кількісним показником, зробивши прогнози на тестовій вибірці та побудувавши матрицю невідповідностей (confusion matrix) для спрогнозованих результатів. Побудована матриця невідповідностей показана на рисунку 4.7

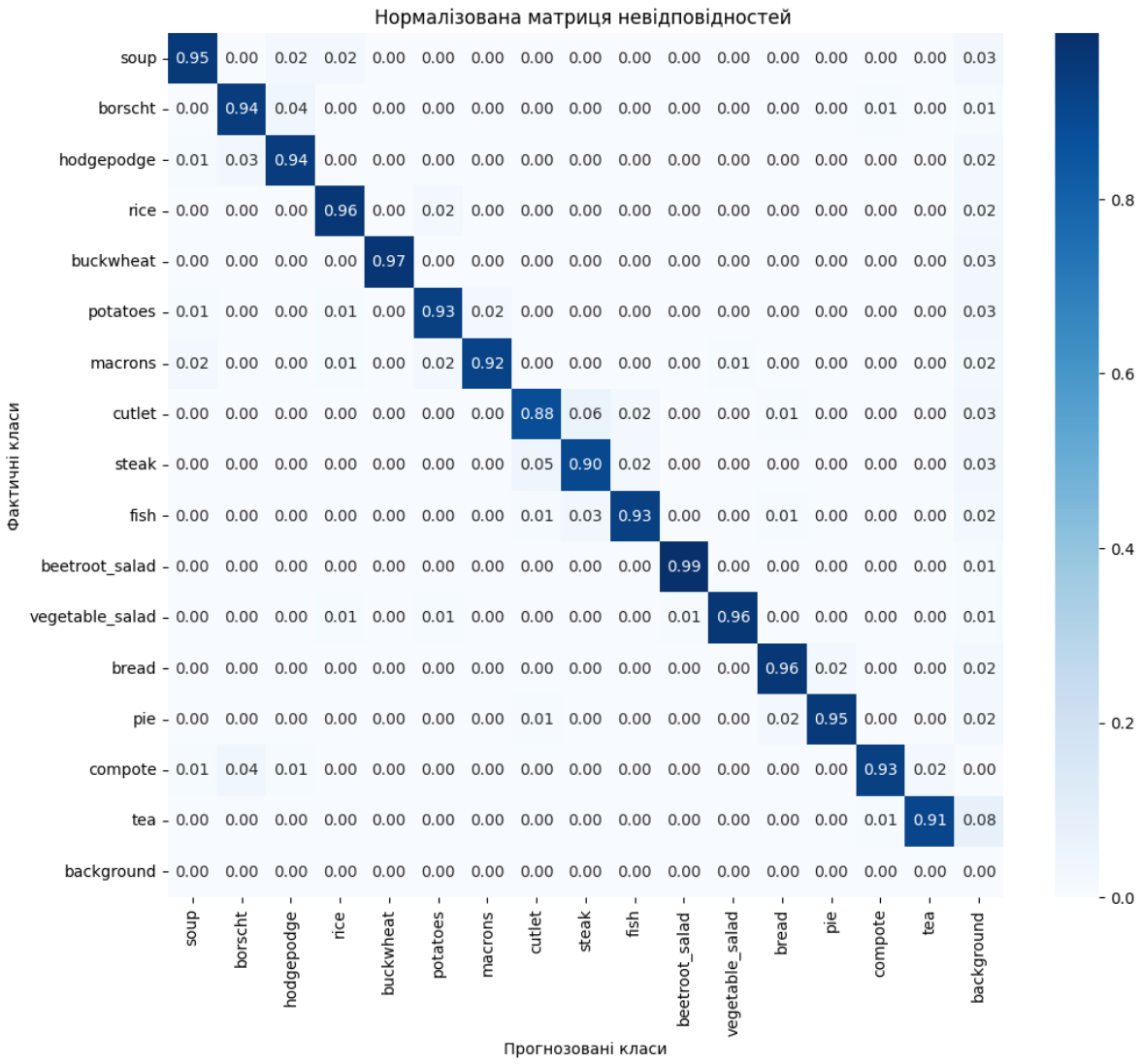

Рисунок 4.7 — Нормалізована матриця невідповідностей для тестового набору даних

Наступним етапом для реалізації задачі є розробка чат-боту в Telegram та отримання токену [18].

Перш за все, потрібно створити бота в Telegram. Це можна зробити через @BotFather (рис. 4.8).

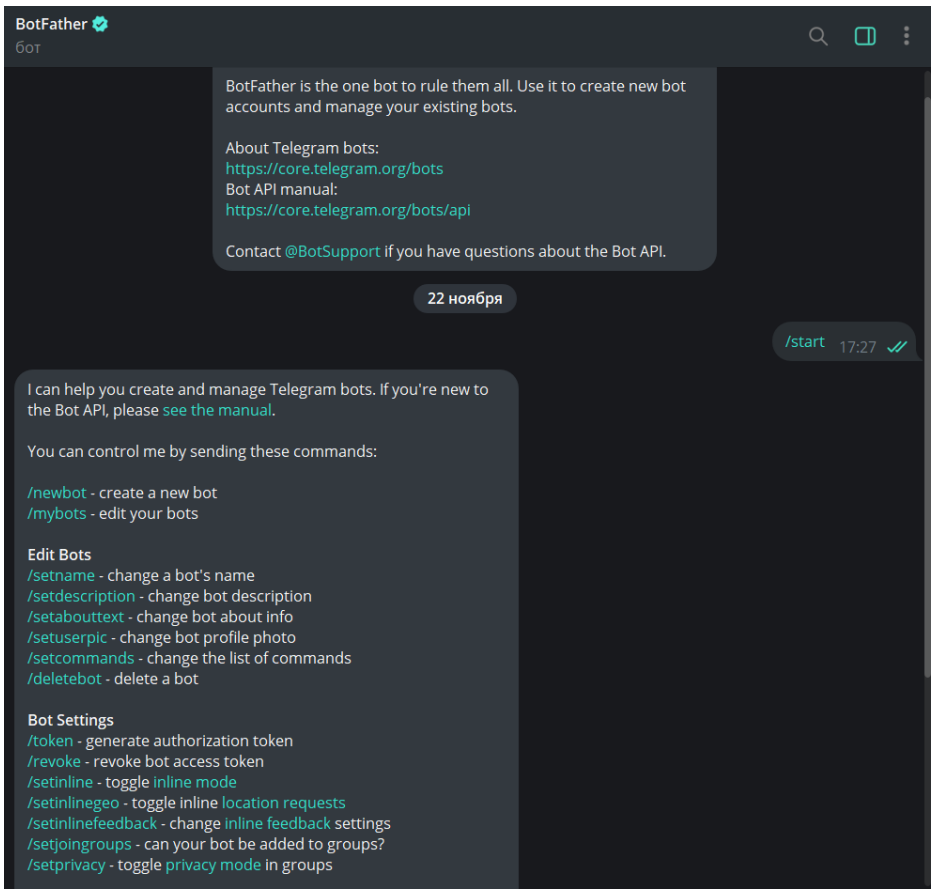

Рисунок 4.8 — Початок реєстрації бота

Для подальшого створення та отримання токена для доступу до API, треба написати команду «/newbot», що дозволить зареєструвати нового бота та вказати ім’я та username бота (рис. 4.9).

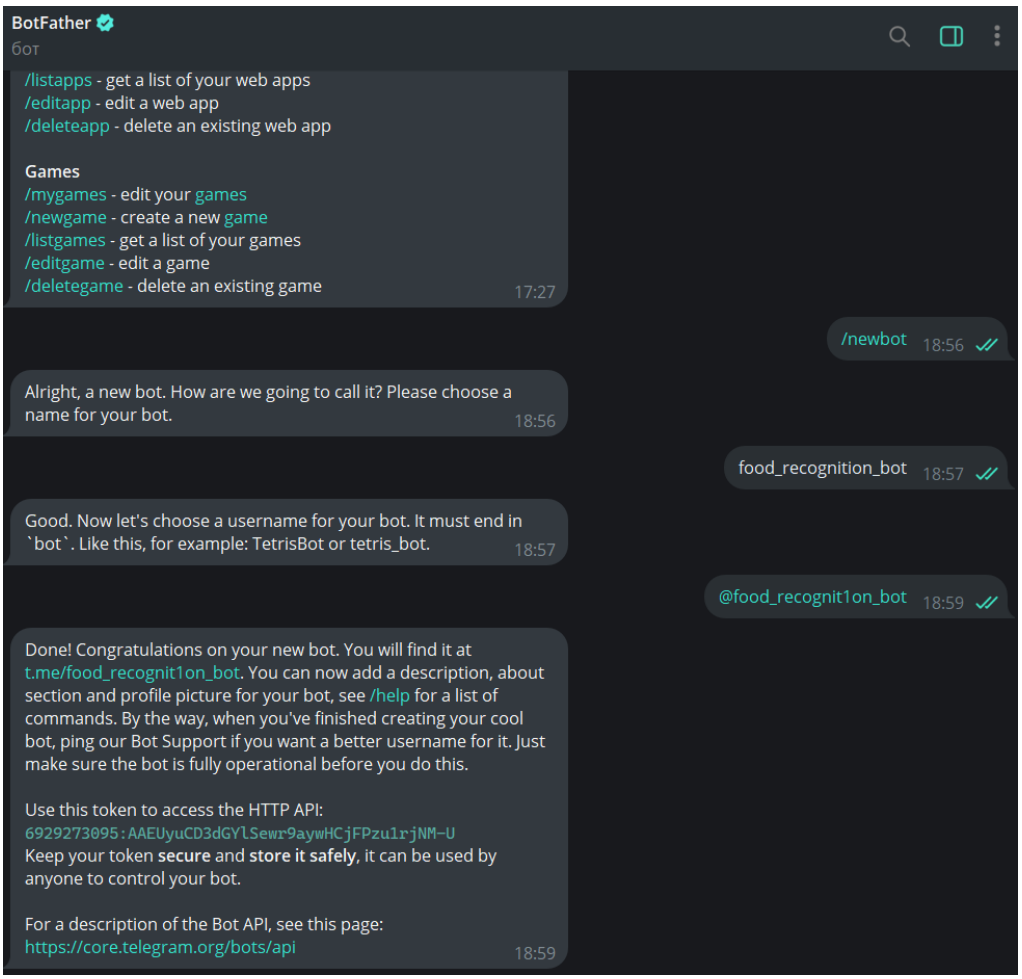

Рисунок 4.9 — Команди для створення бота

Отримавши Telegram Token можна приступити до написання коду.

Опис деяких ключових елементів структури:

– bot/: Ця папка містить всі файли та модулі, пов'язані з ботом.

– main.py: Основний файл для запуску бота.

– handlers/: Модуль для обробки команд та повідомлень.

– utils/: Допоміжні функції та утиліти, такі як розпізнавання їжі.

– config.py: Конфігураційний файл, для зберіганя токену та інших налаштувань.

– requirements.txt: Файл, де перераховані всі бібліотеки та версії, які використовуються в проєкті. Можна створити його за допомогою команди `pip freeze > requirements.txt`.

– main_project_file.py: Головний файл проєкту для імпорту та запуску

бота.

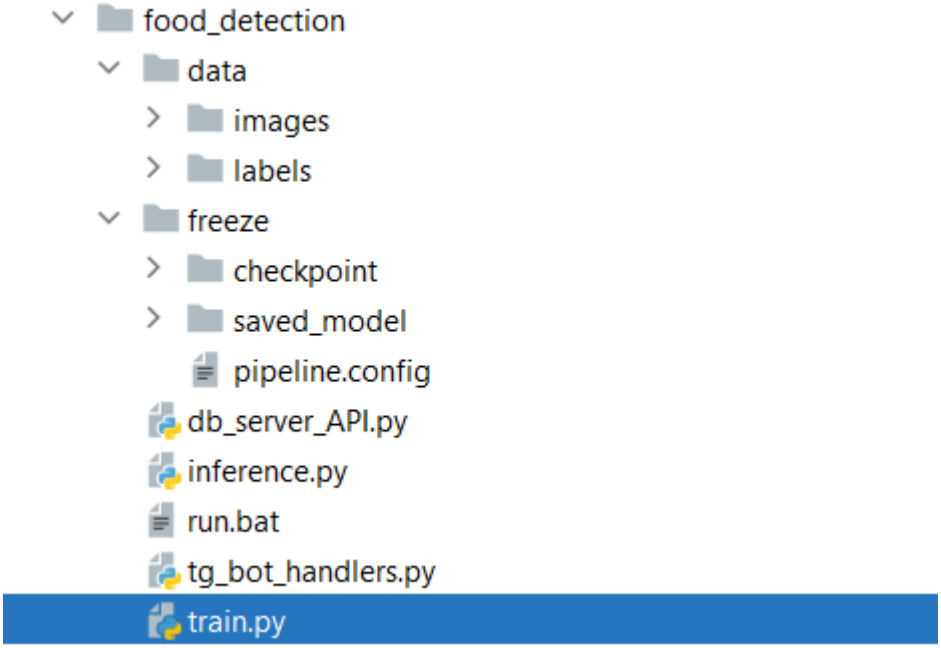

Рисунок 4.10 — Структура проєкту

Для спрощення роботи з API Telegram, встановлено бібліотеку «telebot», та використано pip [19]:

```
pip install pyTelegramBotAPI
```

Pip — це система управління пакетами для мови програмування Python. Вона дозволяє легко встановлювати та управляти бібліотеками та пакетами, які використовуються в проектах на Python. Pip є скороченням від «Pip Installs Packages» або «Pip Installs Python».

Основні команди pip включають [20]:

- `pip install package_name`: встановлення пакету;
- `pip uninstall package_name`: видалення встановленого пакету;
- `pip freeze`: виведення списку всіх встановлених пакетів та їх версій;
- `pip install -r requirements.txt`: встановлення пакетів з файлу requirements.txt.

Далі необхідно створити файл «main.py» та нап+-+9исати код для початку взаємодії з Telegram API:

```
import telebot
# свій токен бота
TOKEN = '6929273095:AAEUyuCD3dGYlSewr9aywHCjFPzu1rjNM-U'
bot = telebot.TeleBot(TOKEN)
# Обробник команди /start
@bot.message_handler(commands=['start'])
def handle_start(message):
    bot.send_message(message.chat.id, "Завантажте будь ласка фото вашого замовлення")

# Запуск бота
if __name__ == "__main__":
    bot.polling(none_stop=True)
```

Після запуску скрипта, бот буде відповідати на команду «/start» (рис. 4.11).

Рисунок 4.11 — Відповідь на команду запуску

Обробник для отримання фотографій від користувачів:

```
# Обробник для фото
@bot.message_handler(content_types=['photo'])
def handle_photo(message):
    # Обробка фото
    file_id = message.photo[-1].file_id
    file_info = bot.get_file(file_id)
    file = bot.download_file(file_info.file_path)

    # Тут можна додати обробку фото, викликати функції детектування і т.д.

    # Відправлення повідомлення про успішне отримання фото
    bot.send_message(message.chat.id, "Фото отримано та оброблено.")
```

Повний код обробника подій та алгоритм функціонування Telegram боту наведено у Додатку В.

4.2 Використання програмного додатку

На етапі тестування необхідно зосередитись на оцінці та перевірці функціональності чат-бота. Тестування є важливою частиною розробки, оскільки воно дозволяє впевнитися в правильності роботи бота та його відповідності визначеним вимогам.

У цьому розділі будуть представлені результати тестування які зображені на рисунках 4.12-4.16. Тестування дозволить виявити можливі недоліки та покращити якість чат-бота перед його впровадженням в роботу.

Спочатку потрібно знайти бота та запустити його (рис. 4.12). Після чого надсилаємо боту коректне зображення зі стравами (рис. 4.13).

Рисунок 4.12 — Початкова сторінка бота

Рисунок 4.13 — Тестування роботи чат-бота на мобільній версії

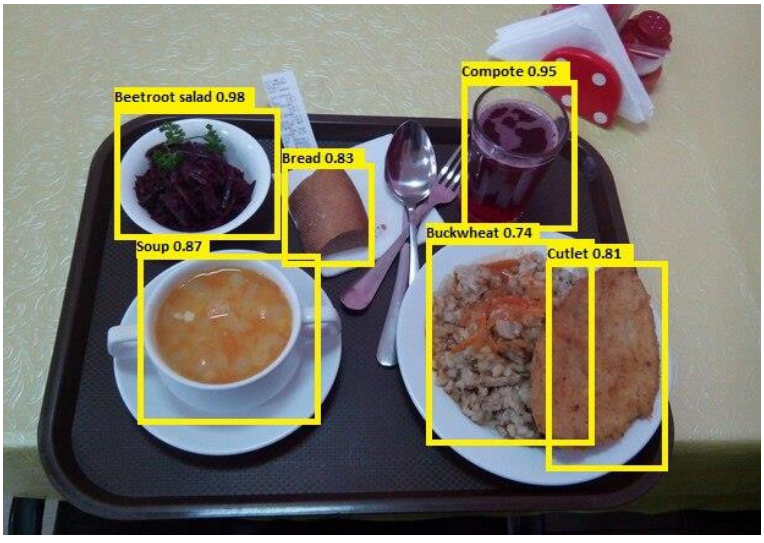

Рисунок 4.14 — Результат детектування страв

У випадку відправки тексту або файлів іншого формату, користувач отримає відповідне повідомлення про некоректні дані (рис. 4.15).

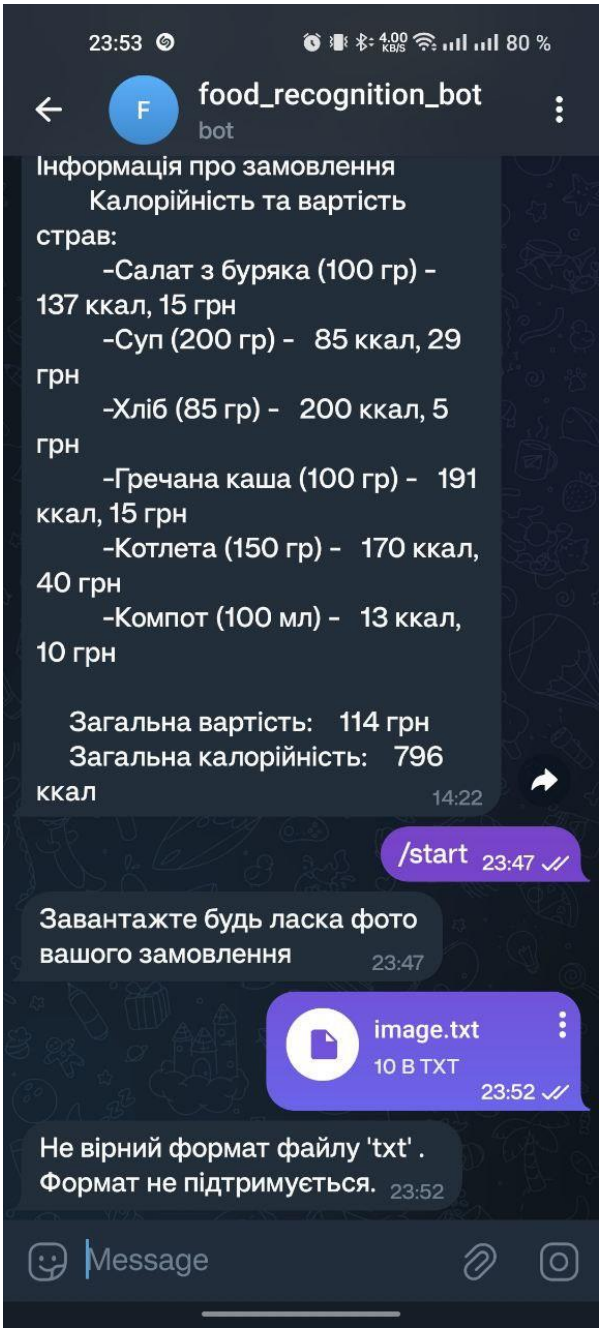

Рисунок 4.15 — Демонстрація попередження користувача

Якщо користувач введе неякісне фото або з зображенням, що не містить страви, бот попередить про неможливість детектування страв (рис. 4.16).

Рисунок 4.16 — Демонстрація попередження користувача

ВИСНОВКИ

Розробка та впровадження інформаційної технології автоматизації ціноутворення та калькуляції у закладах громадського харчування, зокрема використання технології детектування їжі, є актуальним та перспективним напрямком для оптимізації процесів у сфері обслуговування клієнтів. Розглядаючи результати роботи та вивчення предметної області, можна зробити декілька важливих висновків:

– Реалізація інформаційної технології, яка використовує алгоритми машинного навчання для детектування їжі та калькуляції вартості страв, є інноваційним кроком у сфері громадського харчування.

– Інтерактивний чат-бот спрощує взаємодію клієнтів з закладом. Можливість завантаження фотографій страв та отримання швидкої інформації про їх вартість робить процес замовлення та вибору страв більш зручним та привабливим для клієнтів.

– Використання сучасних алгоритмів машинного навчання забезпечує високу точність розпізнавання страв, що робить систему довірчою та надійною в щоденному використанні.

Під час виконання роботи були вирішені наступні задачі:

– визначено об’єкт та предмет дослідження;

– виявлена проблема та актуальність розробки;

– виконано аналіз продуктів-аналогів, у результаті чого були виявлені недоліки та переваги;

– визначені функції та задачі, які повинна виконувати система;

– проведено вибір програмних засобів для реалізації;

– виконано вибір методів вирішення поставленої задачі;

Програмна реалізація чат-боту була виконана за допомогою мови програмування Python, з використанням згорткових нейронних мереж та моделей машинного навчання.

Висновки роботи дають підстави вважати, що впровадження інформаційної технології автоматизації ціноутворення та калькуляції у закладах громадського харчування є важливим етапом для модернізації та оптимізації гастрономічного бізнесу. Технологічні інновації, такі як використання алгоритмів машинного навчання та чат-ботів, дозволяють закладам не лише збільшити ефективність управління, але й підвищити якість обслуговування клієнтів.

СПИСОК ЛІТЕРАТУРИ

1. The role of information technology in the food industry [Електронний ресурс]. — Режим доступу: https://www.researchgate.net/publication/339572993_The_role_of_information_technology_in_the_food_industry – (дата звернення 21.11.2023) — Назва з титулу екрана.

2. Food and Society / Chammy Lai Peng Tai — London: Academic Press, 2020. — 395 с.

3. Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems / Geron Aurelien. — Boston: O'Reilly Media, 2019. — 510 с.

4. Prediction of agricultural waste compost maturity using fast regions with convolutional neural network(R-CNN)/ Sangeetha J. — Tokyo: Proceedings, 2022. — 205 с.

5. A quantum convolutional network and ResNet (50)-based classification architecture for the MNIST / Esraa Hassan — London: Academic Press, 2022. — 195 с.

6. The Python Book: The ultimate guide to coding with Python / Alex Hoskins — London: Richmond House, 2018. — 472 с.

7. Machine learning in architecture / Beyza Topuz — London: Automation, 2023. — 154 с.

8. A-MobileNet: An approach of facial expression recognition / Yahui Nan — London: Alexandria Engineering Journal, 2022. — 224 с.

9. A brief overview of R-CNN, Fast R-CNN and Faster R-CNN [Електронний ресурс]. — Режим доступу: https://medium.com/mlearning-ai/a-brief-overview-of-r-cnn-fast-r-cnn-and-faster-r-cnn-9c6843c9ffc0 – (дата звернення 21.11.2023) — Назва з титулу екрана.

10. Object Detection. Розпізнавай та володарюй. Частина 2 [Електронний ресурс]. — Режим доступу:

https://habr.com/ru/companies/jetinfosystems/articles/498652/ – (дата звернення 21.11.2023) — Назва з титулу екрана.

11. R-CNN vs Fast R-CNN vs Faster R-CNN – A Comparative Guide [Електронний ресурс]. — Режим доступу: http://analyticsindiamag.com/r-cnn-vs-fast-r-cnn-vs-faster-r-cnn-a-comparative-guide/ – (дата звернення 21.11.2023) — Назва з титулу екрана.

12. Tegdet: An extensible Python library for anomaly detection using time evolving graphs/ Simona Bernardi — Boston: O'Reilly Media, 2023. — 354 с.

13. 7 Best Computer Vision Libraries in Python [Електронний ресурс]. — Режим доступу: https://www.projectpro.io/article/computer-vision-libraries/772 – (дата звернення 21.11.2023) — Назва з титулу екрана.

14. Object Detection Algorithms and Libraries [Електронний ресурс]. — Режим доступу: https://neptune.ai/blog/object-detection-algorithms-and-libraries – (дата звернення 21.11.2023) — Назва з титулу екрана.

15. The Python Book: The ultimate guide to coding with Python / Alex Hoskins — London: Richmond House, 2016. — 472 с.

16. Object Detection Algorithms and Libraries [Електронний ресурс]. — Режим доступу: https://neptune.ai/blog/object-detection-algorithms-and-libraries – (дата звернення 21.11.2023) — Назва з титулу екрана.

17. Компоненти синтаксису IDEF0 [Електронний ресурс]. — Режим доступу: https://elib.lntu.edu.ua/sites/default/files/elib_upload/page9.html – (дата звернення 21.11.2023) — Назва з титулу екрана.

18. Use of chat bots in Learning Management Systems / Eugeny Bezverhny — Tokyo: Procedia Computer Science, 2020. — 169 с.

19. Machine learning algorithms for teaching AI chat bots / Evgeny Tebenkov — Tokyo: Procedia Computer Science, 2021. — 191 с.

20. How to Create a Telegram Bot using Python[Електронний ресурс]. — Режим доступу: https://www.freecodecamp.org/news/how-to-create-a-telegram-bot-using-python/ – (дата звернення 21.11.2023) — Назва з титулу екрана.

ДОДАТКИ

Додаток А

Код препроцесінгу даних та конвертації їх в tfrecord формат

```
ROOT_DIR = './data/DATASET'
DATA_DIR = '/input/data'
IMAGES_SUB_DIR = 'images'
IMAGES_DIR = os.path.join(ROOT_DIR, IMAGES_SUB_DIR)
ANNOTATIONS_FILE = os.path.join(DATA_DIR, 'annotations.json')

TRAIN_PROP = .80
SEED = 17

if not os.path.exists(IMAGES_DIR):
    subprocess.run(['mkdir', '-p', IMAGES_DIR]

with open(ANNOTATIONS_FILE) as json_file:
    data = json.load(json_file)

images = data['images']
categories = data['categories']
annotations = data['annotations']

images_annotations = []
for idx, image in enumerate(images):
    image_id = int(image['id'])
    random_number = idx
    file_name = image['file_name']

    # rename files to unique numbers
    new_file_name = '%s.jpg' % str(random_number)
    file_location = '%s/%s' % (DATA_DIR, file_name)
    new_file_location = '%s/%s' % (IMAGES_DIR, new_file_name)
    if os.path.isfile(file_location):
        # print('renamed: %s to %s' % (file_location,
new_file_location))
        shutil.copy(file_location, new_file_location)
    image['file_name'] = new_file_name
    image['folder'] = DATA_DIR

    # get annotations for the image
    _annotations = [a for a in annotations if int(a['image_id']) ==
image_id]

    # something wrong with y coordinates in data
    for a in _annotations:
        (x,y,w,h) = a['bbox']
        a['bbox'][1] = image['height'] - y - h

    images_annotations.append((image, _annotations))

np.random.seed(SEED)

images_annotations_idx = range(0,len(images_annotations))

images_annotations_train_idx = np.random.choice(
    len(images_annotations),
    size=int(len(images_annotations)*TRAIN_PROP),
```

```
    replace=False
)
images_annotations_train = [images_annotations[i] for i in
images_annotations_train_idx]

images_annotations_val_idx = np.random.choice(
    list(set(images_annotations_idx)-set(images_annotations_train_idx)),
    size=int(len(images_annotations_idx)*(1-TRAIN_PROP)/2),
    replace=False
)
images_annotations_val = [images_annotations[i] for i in
images_annotations_val_idx]

images_annotations_test_idx = list(set(images_annotations_idx)-
set(images_annotations_train_idx)-set(images_annotations_val_idx))
images_annotations_test = [images_annotations[i] for i in
images_annotations_test_idx]

print(
'''
# TRAIN IMAGES: %d
# VALIDATION IMAGES: %d
# TEST IMAGES: %d
''' % (len(images_annotations_train), len(images_annotations_val),
len(images_annotations_test))
)

LABEL_PATH = os.path.join(ROOT_DIR, 'labelmap.pbtxt')

if not os.path.exists(LABEL_PATH):
    print('Building label map from examples')

    from object_detection.protos import string_int_label_map_pb2
    from google.protobuf import text_format

    labelmap = string_int_label_map_pb2.StringIntLabelMap()
    for category in categories:
        item = labelmap.item.add()
        # label map id 0 is reserved for the background label
        item.id = int(category['id'])+1
        item.name = category['name']

    with open(LABEL_PATH, 'w') as f:
        f.write(text_format.MessageToString(labelmap))

    print('Label map witten to labelmap.pbtxt')
else:
    print('Reusing existing labelmap.pbtxt')

with open(LABEL_PATH, 'r') as f:
    pprint.pprint(f.readlines())

label_map = label_map_util.load_labelmap(LABEL_PATH)
categories = label_map_util.convert_label_map_to_categories(label_map,
max_num_classes=NCLASSES, use_display_name=True)
category_index = label_map_util.create_category_index(categories)

def build_tfrecords(dataset, output_path, images_annotations,
category_index):
    writer = tf.compat.v1.python_io.TFRecordWriter(output_path)
    label_map_dict = label_map_util.get_label_map_dict(LABEL_PATH)
```

```
    print("Building TFRecord files for dataset: %s" % dataset)

    for idx, (image, _annotations) in enumerate(images_annotations):
        if idx % 100 == 0:
            print('%d of %d annotations' % (idx,
len(images_annotations)))

        _, tf_example, num_annotations_skipped = create_tf_example(
            image=image,
            annotations_list=_annotations,
            image_dir=IMAGES_DIR,
            category_index=category_index,
            include_masks=False
        )

        writer.write(tf_example.SerializeToString())

    writer.close()
    print("Done!")
```

Додаток Б

Код для трансферу вагових коефіцієнтів, створення моделі детектору, налаштування параметрів навчання моделі.

```
import tensorflow as tf
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
from tensorflow.keras.layers import Conv2D, Reshape
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint


import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2

from keras.models import load_model
# Load or create your source MobileNetV2-based classification model
source_model = load_model('origin_model.h5')

# Create the target MobileNetV2 backbone model
target_model = MobileNetV2(weights=None, include_top=False)

# Create a dictionary of source model layers
source_layers = {layer.name: layer for layer in source_model.layers}

# Transfer weights from source to target model by layer name
for layer in target_model.layers:
    if layer.name in source_layers and
source_layers[layer.name].get_weights():
        # Check if the layer shapes are compatible
        if source_layers[layer.name].get_weights()[0].shape ==
layer.get_weights()[0].shape:
            layer.set_weights(source_layers[layer.name].get_weights())


classses = {0:'soup',
            1:'borscht',
            2:'hodgepodge',
            3:'rice',
            4:'buckwheat',
            5:'potatoes',
            6:'macrons',
            7:'cutlet',
            8:'steak',
            9:'fish',
            10:'beetroot_salad',
            11:'vegetable_salad',
            12:'bread',
            13:'pie',
            14:'compote',
            15:'tea'}

def create_ssd_model(num_classes, backbone=MobileNetV2, image_size=(224,
224)):
    # Завантаження бекбону без верхніх шарів
    base_model = backbone(input_shape=(*image_size, 3),
include_top=False)
```

```
    base_model.trainable = False

    # Додавання шарів для детектування
    x = base_model.output
    x = Conv2D(4 * (num_classes + 4), kernel_size=3, padding='same')(x)
# 4 якорні рамки на клас

    # Формування виходу
    num_anchors = 4
    num_boxes = num_anchors * (num_classes + 4)
    x = Reshape((image_size[0] // 32, image_size[1] // 32, num_anchors,
num_classes + 4))(x)

    model = Model(inputs=base_model.input, outputs=x)
    return model


def smooth_l1_loss(y_true, y_pred):

    diff = tf.abs(y_true - y_pred)
    less_than_one = tf.cast(tf.less(diff, 1.0), tf.float32)
    loss = (less_than_one * 0.5 * diff ** 2) + (1 - less_than_one) *
(diff - 0.5)
    return loss

def ssd_loss(num_classes):
    def ssd_custom_loss(y_true, y_pred):
        # Розділення міток і передбачень на класифікацію та локалізацію
        y_true_cls, y_true_loc = y_true[..., :num_classes], y_true[...,
num_classes:]
        y_pred_cls, y_pred_loc = y_pred[..., :num_classes], y_pred[...,
num_classes:]

        # Втрата класифікації (кросс-ентропія)
        cls_loss = tf.keras.losses.categorical_crossentropy(y_true_cls,
y_pred_cls, from_logits=True)

        # Втрата локалізації рамки (Smooth L1)
        loc_loss = smooth_l1_loss(y_true_loc, y_pred_loc)

        # Комбінування втрат
        total_loss = cls_loss + loc_loss
        return total_loss
    return ssd_custom_loss



# Створення моделі
num_classes = 16  # Кількість класів у датасеті
model = create_ssd_model(num_classes)
early_stopping = EarlyStopping(monitor='val_loss', patience=5,
verbose=1)

# Налаштування Model Checkpoint
# Це збереже модель після кожної епохи, де спостерігається покращення в
метриці
model_checkpoint = ModelCheckpoint(
    'ssd_model_checkpoint.h5', monitor='val_loss', verbose=1,
save_best_only=True, save_weights_only=False)

model.compile(optimizer='adam', loss=ssd_loss(num_classes))
```

```
# Тренування моделі з колбеками
history = model.fit(
    train_data,
    validation_data=val_data,
    epochs=100,
    callbacks=[early_stopping, model_checkpoint]
)
```

Додаток В

Реалізація на мові Python створеного телеграм боту

```
from telegram import Update
from telegram.ext import Updater, CommandHandler, MessageHandler, Filters,
CallbackContext

import cv2
import requests


class FoodDetector:
    def __init__(self, model, database_api_url):
        self.model = model
        self.database_api_url = database_api_url

    def detect_food(self, image_path):
        # Завантаження зображення
        image = cv2.imread(image_path)

        # Використання моделі для детекції страв
        detected_foods = self.model.predict(image)

        # Отримання інформації про страви
        food_info = self.get_food_info(detected_foods)

        # Візуалізація результатів на зображенні
        for food in detected_foods:
            x, y, w, h = food['bbox']
            font = cv2.FONT_HERSHEY_SIMPLEX
            org = (x, y)
            fontScale = 1
            color = (255, 0, 0)
            thickness = 2
            image = cv2.putText(image, food.label, org, font,
                                fontScale, color, thickness, cv2.LINE_AA)
            cv2.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 2)

        # Збереження зображення
        cv2.imwrite(image_path, image)

        return food_info

    def get_food_info(self, foods):
        # Збір інформації про кожну страву з API
        food_details = {}
        for food in foods:
            response = requests.get(f"{self.database_api_url}/get_food_info",
params={'food_name': food['name']})
            if response.status_code == 200:
                food_details[food['name']] = response.json()
            else:
                food_details[food['name']] = "No information available"

        return image_path, food_info
```

```
img_recognition = FoodDetector("./best_model.h5",
'"https://127.0.0.1:5555')

def start(update: Update, context: CallbackContext) -> None:
    update.message.reply_text('Завантажте будь ласка фото вашого
замовлення')

def handle_file(update: Update, context: CallbackContext) -> None:
    file = update.message.document
    chat_id = update.message.chat_id

    # Отримання файлу
    new_file = context.bot.get_file(file.file_id)
    file_path = f"{file.file_name}"
    form = file.file_name.split('.')[1]
    if form not in ['jpg', 'png', 'jpeg']:
    # Відправка відповіді користувачу
        update.message.reply_text(f"Не вірний формат файлу '{form}' .
Формат не підтримується.")
    else:
        new_file.download(file_path)
        recognized_file_path, description =
img_recognition.detect_food(file_path)
        update.message.reply_photo(photo=open(recognized_file_path,
'rb'), caption = description)


def handle_photo(update: Update, context: CallbackContext) -> None:
    # Текст, який буде відправлено разом з фото
    photo_file = update.message.photo[-1].get_file()
    file_name = update.user_id + ".jpg"
    photo_file.download(file_name)
    recognized_file_path, description =
img_recognition.detect_food(file_name)
    photo_path = recognized_file_path
    text = description

    # Відправляємо фото з текстом
    update.message.reply_photo(photo=open(photo_path, 'rb'),
caption=text)


def main():
    updater = Updater("6553304049:AAGFzyXXXXXXXXXXXXIBrvDGjPUyMoo832M")

    dispatcher = updater.dispatcher

    dispatcher.add_handler(CommandHandler("start", start))
    dispatcher.add_handler(MessageHandler(Filters.photo, handle_photo))
    dispatcher.add_handler(MessageHandler(Filters.document,
handle_file))


    updater.start_polling()
    updater.idle()

if __name__ == '__main__':
    main()
```