Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

ІНФОРМАЦІЙНА СИСТЕМА РОЗПІЗНАВАННЯ ТЕКСТУ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

Виконала: студентка 4 курсу, групи 1СП-20б
спеціальності 123 — Комп'ютерна інженерія
Освітня програма — Системне програмування
_____ Поташна К.Я
Керівник: к.т.н., доц.каф. ОТ
_____ Городецька О.С
«11» 06 2024 р.
Рецензент: к.т.н., доц. каф. ПЗ
_____ Войтко В.В
«12» 06 2024 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
«14» 06 2024 р

ВНТУ 2024

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність — 123 Комп'ютерна інженерія
Освітньо-професійна програма — Системне програмування

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

д.т.н., проф. О.Д. Азаров

«06» травня 2024 р.

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ ДИПЛОМНУ РОБОТУ

студентці Поташній Каріні Ярославівні

1 Тема роботи «Інформаційна система розпізнання тексту на основі нейронної мережі» керівник роботи Городецька Оксана Степанівна к.т.н., доцент, затверджено наказом вищого навчального закладу «11» березня 2024 року №80.

2 Строк подання студентом роботи 14.06.2024

3 Вихідні дані до роботи: реалізувати додаток для розпізнавання тексту з фотографій та його озвучення, який буде спрощувати доступ до інформації для людей з обмеженими можливостями.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз технологій, вибір та обгрунтування підходів для реалізації програми, розробка клієнтського додатку, розробка серверної частини на тестування.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): структурна схема взаємодії користувача з додатком та головний екран інформаційної системи.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-3	Городецька Оксана Степанівна к.т.н., доцент		

7 Дата видачі завдання 12.03.2024

8 Календарний план виконання БР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів БР	Строк виконання	Підпис
1	Постановка задачі	14.03.24	
2	Огляд існуючих рішень	18.03.24	
3	Виконання першого теоретичного розділу	25.03.24	
4	Виконання другого розділу	03.04.24	
5	Виконання третього програмного розділу	28.04.24	
6	Оформлення пояснювальної записки та ілюстративного матеріалу	31.04.24	
7	Перевірка якості виконання бакалаврської дипломної роботи та усунення недоліків	03.05.24	
8	Підписи супроводжувальних документів у керівника, опонента, нормоконтролера	28.05.24	

Студент _____ Поташна Каріна Ярославівна

Керівник _____ к.т.н., доц. Городецька Оксана Степанівна

АНОТАЦІЯ

УДК 004.932.72:004.852

Поташна К.Я. Інформаційна система розпізнавання тексту на основі нейронної мережі. Бакалаврська дипломна робота за спеціальності 123 — Комп'ютерна інженерія, освітня програма — Системне програмування. Вінниця: ВНТУ, 2024. 91 с.

На укр. Мові. Бібліогр.: рис.: 5; табл. 1.

У бакалаврській роботі реалізована інформаційна система, яка дозволяє користувачам з обмеженими можливостями легко отримувати інформацію з фотографій, що сприятиме їхній самостійності та інтеграції в суспільство. Розвиток інформаційних технологій і штучного інтелекту відкриває нові можливості для підтримки людей з обмеженими можливостями. Доступ до інформації є важливою складовою для повноцінного функціонування кожної людини. Проте для людей з порушеннями зору отримання інформації з фотографій є складним завданням. У зв'язку з цим, актуальним стає питання розробки інформаційних систем, здатних розпізнавати та озвучувати текст. Метою даної роботи є створення додатку, який забезпечить доступність інформації для незрячих та людей з порушеннями зору за допомогою використання нейронних мереж.

Інформаційна сиситема для розпізнання тексту розроблена мовою Python, мікрофреймовком Flask та відкритим програмним забезпеченням для швидкої розробки мультимедійних додатків Kivy.

Ключові слова: інформаційна система, розпізнавання тексту, нейронні мережі, доступність, обмежені можливості.

ABSTRACT

UDC 004.932.72:004.852

Potashna K.Ya. Text recognition information system based on a neural network. Bachelor thesis on specialty 123 — Computer engineering, educational program — System programming. Vinnytsia: VNTU, 2024. 91 p.

In Ukrainian languages Bibliography: fig.: 5; table 1.

The bachelor thesis implements an information system that allows users with disabilities to easily obtain information from photos, which will contribute to their independence and integration into society. The development of information technologies and artificial intelligence opens up new opportunities for supporting people with disabilities. Access to information is an important component for the full functioning of every person. However, for visually impaired people, obtaining information from photographs is a difficult task. In this regard, the issue of developing information systems capable of recognizing and voicing text becomes relevant. The purpose of this work is to create an application that will ensure the availability of information for blind and visually impaired people using neural networks.

The text recognition information system is developed using the Python language, the Flask microframework, and Kivy, an open source software for the rapid development of multimedia applications.

Keywords: information system, text recognition, neural networks, accessibility, limited capabilities.

ЗМІСТ

ВСТУП

У сучасному світі доступ до інформації є **актуальною** складовою для повноцінного функціонування кожної людини. Однак, для людей з обмеженими можливостями, зокрема для незрячих та людей з порушеннями зору, отримання інформації з фотографій може бути викликом. У зв'язку з цим, актуальним стає питання розробки інформаційних систем, що можуть допомогти у вирішенні цього завдання. Ці системи не лише вирішують конкретне технологічне завдання, а й розширюють доступні можливості для широкого кола користувачів, роблячи інформацію більш доступною для людей з обмеженими можливостями. Вони відкривають нові способи для вивчення, дослідження, розуміння різних задач та полегшують життя.

Мета даної роботи полягає у створенні інформаційної системи для людей з обмеженими можливостями із розширеними функціональними можливостями, а саме розпізнавання тексту та його озвучення шляхом використання нейронної мережі.

По-перше, розвиток інформаційних технологій і, зокрема, технологій штучного інтелекту, відкриває нові можливості для підтримки людей з обмеженими можливостями. Нейронні мережі, що використовуються для розпізнавання тексту, дозволяють значно покращити точність і швидкість обробки інформації. Це робить можливим створення систем, які можуть ефективно розпізнавати текст на зображеннях і перетворювати його на озвучений текст. Така технологія є особливо корисною для незрячих та слабозорих людей, надаючи їм можливість отримувати інформацію з друкованих джерел, вивісок, меню, документів та інших носіїв, які раніше були для них недоступними.

По-друге, створення такої системи має велике соціальне значення. Люди з обмеженими можливостями часто стикаються з проблемами в повсякденному житті, які для інших можуть здаватися незначними. Інформаційна система, здатна розпізнавати та озвучувати текст, може значно підвищити рівень їх незалежності та самостійності. Це може сприяти їх інтеграції в суспільство, наданню їм можливості активно брати участь у соціальному житті, навчанні, професійній діяльності та інших аспектах життя.

По-третє, така система може знайти застосування в різних сферах, включаючи освіту, медицину, обслуговування та багато інших. В освітньому середовищі вона може бути використана для створення доступних навчальних матеріалів, що є важливим для забезпечення рівних можливостей у навчанні. У медицині вона може допомогти в наданні медичних інструкцій та інформації пацієнтам з порушеннями зору. В обслуговуванні вона може бути використана для покращення якості обслуговування клієнтів з обмеженими можливостями, надаючи їм доступ до інформації про продукти та послуги.

Технологічний прогрес також робить такі системи все більш доступними. Розвиток апаратного забезпечення та зниження вартості обчислювальних потужностей дозволяють створювати ефективні та недорогі рішення, що можуть бути впроваджені у повсякденне життя. Це робить можливим створення портативних пристроїв або мобільних додатків, які можуть використовуватися людьми з обмеженими можливостями в будь-який час і в будь-якому місці.

Отже, створення інформаційної системи для людей з обмеженими можливостями, здатної розпізнавати та озвучувати текст, є актуальним завданням, що має як соціальне, так і технологічне значення. Така система може значно покращити якість життя людей з порушеннями зору, надаючи їм більше незалежності та можливостей для самореалізації. Вона сприятиме їх інтеграції в суспільство, наданню рівних можливостей у навчанні, професійній діяльності та

інших аспектах життя. Завдяки розвитку сучасних технологій, створення таких систем стає все більш реальним та доступним, що робить це завдання ще більш важливим і актуальним.

Об'єкт дослідження - процес розпізнавання тексту на основі нейронної мережі.

Предмет дослідження - методи розпізнавання тексту з фотографій та його озвучення на клієнтському пристрої.

Задачі дослідження:

— оглянути існуючі рішення розпізнавання символів;

— проаналізувати теоретичні аспекти і сучасний стан предметної області;

— проаналізувати та обгрунтувати вибір технології;

— розробити інформаційну систему розпізнавання тексту на основі нейронної мережі;

— протестувати інформаційну систему розпізнавання.

Апробація результатів бакалаврської роботи: зроблено доповідь на LIII науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії [1].

Матеріали роботи доповідались та **опубліковувались**.

Порівняльний аналіз бібліотек для обробки зображень та комп’ютерного зору/ К. Я. Поташна, О. С. Городецька // Матеріали LIII науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії. Вінниця 2024 р. 3 с. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20949/17355

1 ТЕОРЕТИЧНІ АСПЕКТИ І СУЧАСНИЙ СТАН ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Машинне навчання. Основні поняття і використання.

Машинне навчання (ML) — це галузь дослідження штучного інтелекту, яка займається розробкою та дослідженням статистичних алгоритмів, які можуть вивчати дані та узагальнювати їх у невидимі дані, а отже, виконувати завдання без чітких інструкцій [1]. Останнім часом штучні нейронні мережі змогли перевершити багато попередніх підходів у продуктивності [2,3].

Підходи машинного навчання застосовувалися в багатьох галузях, включаючи обробку природної мови, комп'ютерний зір, розпізнавання мови, фільтрацію електронної пошти, сільське господарство та медицину [4,5]. ML відомий у своєму застосуванні в бізнес-проблемах під назвою прогнозна аналітика. Хоча не все машинне навчання базується на статистиці, обчислювальна статистика є важливим джерелом методів у цій галузі.

Математичні основи ML забезпечуються методами математичної оптимізації (математичного програмування). Інтелектуальний аналіз даних є спорідненою (паралельною) галуззю дослідження, яка зосереджена на дослідницькому аналізі даних (EDA) за допомогою неконтрольованого навчання [7,8].

З теоретичної точки зору, ймовірно, приблизно правильне (PAC) навчання забезпечує основу для опису машинного навчання.

1.1.1 Взаємозв'язок з іншими науковими напрямами

Як науковий напрям, машинне навчання виникло в результаті пошуків штучного інтелекту (ШІ). На початку ШІ як академічної дисципліни деякі дослідники були зацікавлені в тому, щоб машини навчалися на даних. Вони намагалися підійти до проблеми різними символьними методами, а також тим,

що тоді називали «нейронними мережами»; це були в основному персептрони та інші моделі, які пізніше виявились перевинаходами узагальнених лінійних моделей статистики [23]. Також використовували ймовірнісні міркування, особливо в автоматизованій медичній діагностиці [24].

Однак дедалі більший акцент на логічному, заснованому на знаннях підході спричинив розрив між ШІ та машинним навчанням. Імовірнісні системи страждали від теоретичних і практичних проблем збору та представлення даних [24]. До 1980 року експертні системи стали домінувати над штучним інтелектом, і статистика була в опіці [25]. Робота над символічним навчанням/навчанням, заснованим на знаннях, продовжувалася в рамках ШІ, що призвело до індуктивного логічного програмування (ILP), але більш статистична лінія досліджень тепер була поза межами власне ШІ, у сфері розпізнавання образів та пошуку інформації [24]. Штучний інтелект та інформатика припинили дослідження нейронних мереж приблизно в той же час. Цю лінію також було продовжено за межами області AI/CS, як «коннекціонізм», дослідниками з інших дисциплін, включаючи Хопфілда, Румельхарта та Хінтона. Їхній головний успіх прийшов у середині 1980-х років із повторним винаходом зворотного поширення [24].

Машинне навчання (ML), реорганізоване та визнане окремою сферою, почало процвітати в 1990-х роках. Сфера змінила свою мету з досягнення штучного інтелекту на вирішення вирішуваних проблем практичного характеру. Він змістив фокус від символічних підходів, успадкованих від ШІ, до методів і моделей, запозичених зі статистики, нечіткої логіки та теорії ймовірностей [25].

1.2 Нейронні мережі

Нейронні мережі— це клас математичних моделей і алгоритмів машинного навчання, які імітують роботу біологічних нейронних мереж. Вони складаються

з великої кількості взаємопов’язаних обчислювальних вузлів або нейронів, що організовані в шари. Сучасні нейронні мережі використовуються в багатьох сферах, включаючи розпізнавання образів, обробку природної мови, прогнозування та багато інших. У даному тексті ми розглянемо основні різновиди нейронних мереж, їхні сильні та слабкі сторони.

Персептрон є однією з найпростіших форм нейронних мереж, запропонованою Френком Розенблаттом у 1958 році. Він складається з одного шару нейронів, кожен з яких з’єднаний з усіма вхідними сигналами. Персептрон використовує вагові коефіцієнти для обчислення лінійної комбінації вхідних сигналів і застосовує порогову функцію активації для прийняття рішення про вихід. Персептрон може вирішувати тільки лінійно роздільні задачі, що є його основною слабкістю. Відтак, для більш складних задач потрібні більш складні моделі, такі як багатошаровий персептрон.

Багатошаровий персептрон (MLP) розширює концепцію персептрона, додаючи один або більше прихованих шарів між вхідним і вихідним шарами. Це дозволяє моделі навчатися нелінійним залежностям між вхідними даними і виходом. Кожен шар складається з нейронів, що використовують нелінійні функції активації, такі як сигмоїдальна або ReLU (Rectified Linear Unit). Багатошаровий персептрон має здатність моделювати складні функції та розпізнавати складні патерни, що робить його універсальним інструментом у машинному навчанні. Проте, навчання MLP може бути повільним і потребувати великої кількості обчислювальних ресурсів, особливо для глибоких мереж з багатьма шарами.

Згорткова нейронна мережа (CNN) спеціально розроблена для обробки даних із сітчастою топологією, таких як зображення. Основна ідея CNN полягає у використанні згорткових шарів, які застосовують фільтри для автоматичного виділення важливих ознак з вхідних даних. Ці фільтри дозволяють мережі бути

менш чутливою до зміщень і деформацій у зображеннях. Згорткові мережі стали стандартом для задач розпізнавання образів, класифікації зображень та детектування об'єктів. Однак, навчання CNN потребує великої кількості даних і обчислювальних ресурсів, що може бути обмеженням для деяких застосувань.

Рекурентні нейронні мережі (RNN) використовуються для обробки послідовних даних, таких як текст або часові ряди. Відмінною рисою RNN є зворотні зв’язки, що дозволяють передавати інформацію через часові кроки. Це робить їх придатними для задач, де важлива послідовність або контекст. Проте, традиційні RNN страждають від проблеми зникання градієнтів, що ускладнює їх навчання на довгих послідовностях. Щоб подолати цю проблему, були розроблені вдосконалені архітектури, такі як довготривала короткочасна пам'ять (LSTM).

Довготривала короткочасна пам'ять (LSTM) є спеціальним видом RNN, що вирішує проблему зникання градієнтів, використовуючи комірки пам'яті та механізми гейтування. LSTM має три основні гейти: вхідний, вихідний та забуття, які контролюють потік інформації через комірки пам'яті. Це дозволяє LSTM запам'ятовувати важливу інформацію протягом довгих періодів часу і забувати нерелевантні дані. LSTM ефективні для задач обробки природної мови, таких як машинний переклад, розпізнавання мови і аналіз настроїв. Проте, складність їх архітектури призводить до більшого обсягу обчислень і часу навчання порівняно з традиційними RNN.

Автоенкодери є типом нейронних мереж, що використовуються для навчання ефективного представлення даних, зазвичай з метою зменшення розмірності або виявлення ознак. Автоенкодер складається з двох частин: енкодера, що перетворює вхідні дані у компактне латентне представлення, і декодера, що відновлює вихідні дані з цього представлення. Основна ідея автоенкодера полягає у навчанні стиснення даних із мінімальною втратою

інформації. Автоенкодери знаходять застосування у задачах обробки зображень, видалення шуму та генерації даних. Проте, як і інші нейронні мережі, автоенкодери можуть страждати від перенавчання і вимагати значної кількості даних для ефективного навчання.

Трансформери є сучасною архітектурою нейронних мереж, що стала популярною завдяки своїй ефективності у задачах обробки природної мови. Основна інновація трансформерів полягає у використанні механізму самоуваги, що дозволяє моделі динамічно зважувати значимість різних частин вхідної послідовності при обчисленні виходу. Це дозволяє трансформерам обробляти послідовності паралельно, що значно прискорює навчання порівняно з RNN та LSTM. Трансформери виявилися дуже ефективними для задач перекладу, генерації тексту, відповіді на запити та інших завдань обробки тексту. Проте, вони є обчислювально інтенсивними і потребують великих обсягів даних для навчання, що може бути обмеженням для деяких застосувань.

Кожен з описаних типів нейронних мереж має свої сильні та слабкі сторони. Персептрон є простим і ефективним для лінійно роздільних задач, але його можливості обмежені. Багатошаровий персептрон здатен моделювати складні нелінійні залежності, проте його навчання може бути ресурсомістким. Згорткові нейронні мережі є дуже ефективними для задач обробки зображень, але потребують великих обчислювальних потужностей. Рекурентні нейронні мережі та LSTM підходять для послідовних даних, але можуть бути складними для навчання. Автоенкодери корисні для зменшення розмірності та виявлення ознак, але можуть страждати від перенавчання. Трансформери демонструють видатні результати в обробці природної мови, проте їх навчання потребує значних обчислювальних ресурсів.

Незважаючи на свої слабкі сторони, кожен тип нейронної мережі має певні переваги, що робить його корисним для конкретних задач. Успішне застосування

нейронних мереж часто вимагає глибокого розуміння їх властивостей та можливостей, а також значного досвіду у налаштуванні та навчанні моделей.

1.3 Види нейронних мереж

В рамках розробки буде використовуватися бібліотека EasyOCR, яка в свою чергу використовує згорткові нейронні мережі. Згорткові нейронні мережі (Convolutional Neural Networks, CNN) є однією з найпотужніших і найпоширеніших архітектур у галузі комп'ютерного зору. Вони знайшли своє застосування у багатьох задачах, включаючи розпізнавання образів, класифікацію зображень, детекцію об'єктів, сегментацію зображень та інші. CNN використовуються у багатьох сучасних додатках, таких як системи розпізнавання облич, автономні автомобілі, медичні діагностичні системи, а також у багатьох інших сферах.

Основна концепція згорткових нейронних мереж полягає у використанні згорткових шарів, які застосовують фільтри для автоматичного виділення важливих ознак з вхідних даних. Кожен згортковий шар складається з набору фільтрів, які ковзають по вхідному зображенню і виконують операцію згортки, виділяючи різні характеристики, такі як границі, текстури та інші важливі деталі. Результати цієї операції формують карти ознак, які передаються далі через мережу для подальшої обробки.

Різні типи рівнів, які разом вивчають ієрархічні представлення вхідних даних наведенні на рисунку 1.1

В CNN для вилучення ознак із зображень використовується операція згортки. Це особливість використовує спільні параметри шарів, де для обробки різних частин вхідного зображення використовуються однакові вагові коефіцієнти. Таким чином в нейронній мережі зменшується кількість параметрів для навчання.

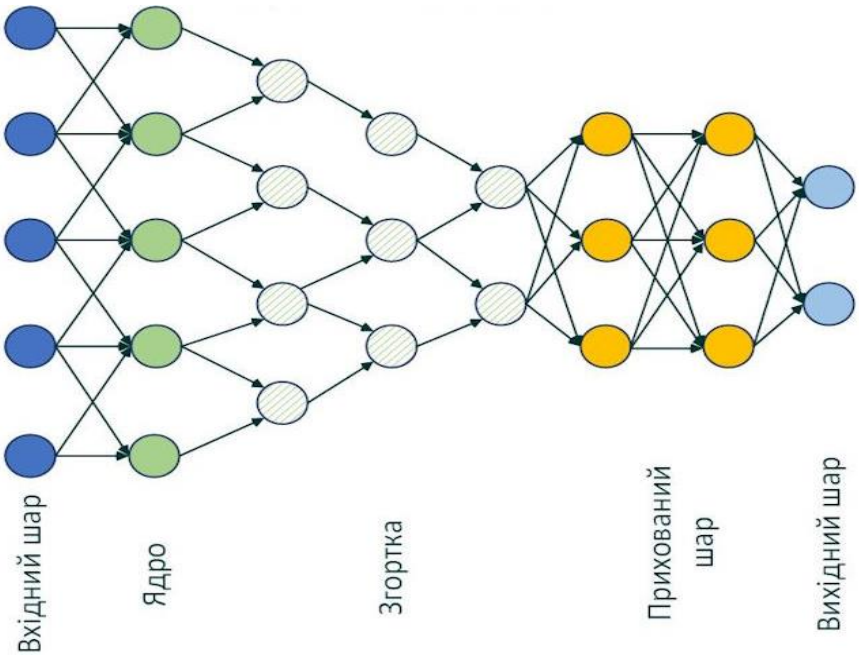

Рисунок 1.1 — Згорткова нейронна мережа

Згорткові шари складаються з набору фільтрів, які можна розглядати як просто двовимірні матриці чисел. Операція згортки — це поелементне множення частини вхідного зображення на ядро фільтра, щоб отримати нову одну точку даних, як показано на рисунку 1.2

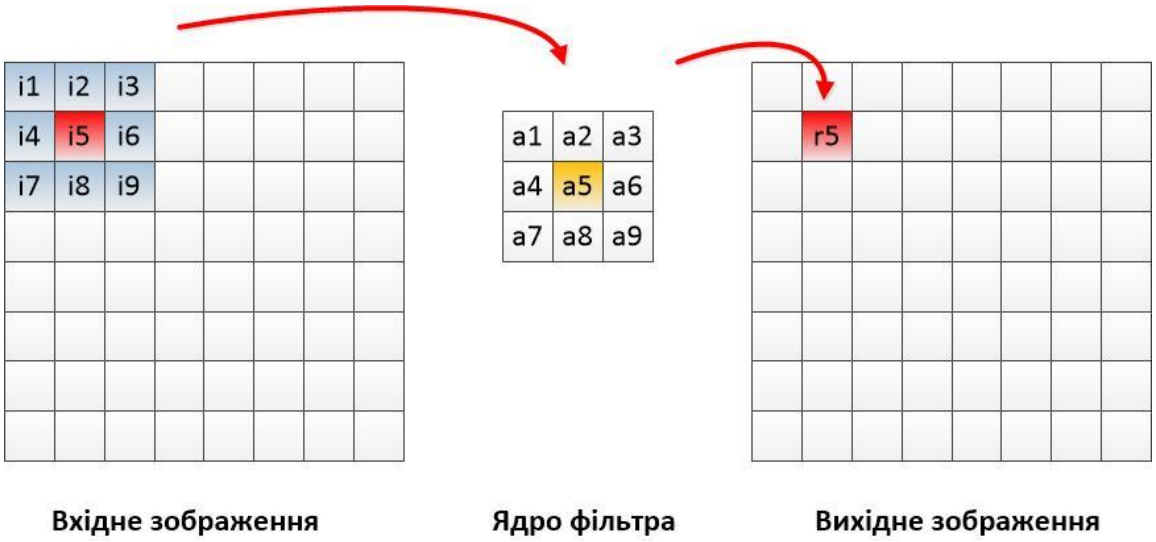

Рисунок 1.2 — Приклад операції згортки

Ми можемо створити вихідне зображення, застосувавши до вхідного зображення фільтр шляхом згортання. Це складається з таких етапів:

— накладання фільтра поверх зображення у конкретному місці;

— виконання поелементного множення між значеннями фільтра та відповідними значеннями на зображенні;

— поелементне додавання результатів множення;

— повторення всіх етапів для кожного пікселя.

Однією з ключових переваг CNN є їх здатність автоматично навчатися виділяти найбільш релевантні ознаки з даних. Це досягається за допомогою процесу навчання, під час якого мережа налаштовує вагові коефіцієнти фільтрів на основі помилок, що виникають при порівнянні прогнозів з реальними мітками даних. Такий підхід дозволяє CNN бути дуже ефективними у задачах, де важливо враховувати просторову структуру вхідних даних.

Згорткові нейронні мережі зазвичай складаються з кількох основних типів шарів: згорткових шарів, шарів підвибірки (pooling layers) та повнозв'язних шарів (fully connected layers). Згорткові шари виконують операцію згортки, яка вже була описана вище. Шари підвибірки зменшують розмірність карт ознак, зменшуючи обчислювальні витрати і зберігаючи найважливіші характеристики. Повнозв'язні шари використовуються для остаточного прогнозування класу або регресії на основі виділених ознак.

Однією з головних особливостей CNN є їх здатність до трансляційної інваріантності, що означає, що вони можуть розпізнавати об'єкти незалежно від їхнього положення у зображенні. Це досягається завдяки тому, що фільтри в згорткових шарах застосовуються до всього зображення з однаковими вагами, що дозволяє мережі виявляти однакові ознаки в різних частинах зображення. Крім того, CNN можуть бути стійкими до змін масштабу і обертання об'єктів, що робить їх дуже потужним інструментом для розпізнавання образів.

Згорткові нейронні мережі знайшли широке застосування у багатьох галузях. Наприклад, в медицині CNN використовуються для автоматичної діагностики захворювань за медичними зображеннями, таких як рентгенівські знімки або МРТ. У сфері автономних транспортних засобів CNN використовуються для розпізнавання дорожніх знаків, пішоходів та інших об'єктів на дорозі. У сфері безпеки CNN застосовуються для розпізнавання облич і відбитків пальців. Вони також використовуються у системах відеоспостереження для автоматичного виявлення підозрілих дій або об'єктів.

1.3.1 Особливості навчання згорткових нейронних мереж

Навчання згорткових нейронних мереж зазвичай виконується за допомогою великих обсягів даних і потужних обчислювальних ресурсів. Для цього використовуються графічні процесори (GPU), які здатні значно прискорити обчислення завдяки своїй паралельній архітектурі. Процес навчання включає кілька етапів, таких як пряма передача (forward pass), зворотне поширення помилки (backpropagation) і оновлення ваг (weight update) на основі градієнтного спуску (gradient descent). Ці етапи повторюються багато разів, доки мережа не досягне прийнятного рівня точності.

Однією з основних проблем при навчанні CNN є перенавчання (overfitting), коли мережа добре працює на навчальних даних, але погано узагальнює нові, невідомі дані. Для боротьби з перенавчанням використовуються різні методи регуляризації, такі як відкидання (dropout), згладжування ваг (weight decay) і рання зупинка (early stopping). Крім того, збільшення кількості навчальних даних за допомогою аугментації даних (data augmentation) також може допомогти зменшити ризик перенавчання.

Згорткові нейронні мережі також використовуються у поєднанні з іншими типами нейронних мереж для створення гібридних моделей, які можуть

вирішувати більш складні задачі. Наприклад, вони можуть бути поєднані з рекурентними нейронними мережами (RNN) для обробки послідовних даних або з автоенкодерами для безконтрольного навчання ознак. Такий підхід дозволяє використовувати переваги різних архітектур і створювати більш потужні та універсальні моделі.

Важливо також зазначити, що розробка і застосування згорткових нейронних мереж потребує не лише знань у галузі машинного навчання, але й глибокого розуміння специфіки задачі, для якої вони застосовуються. Це включає вибір відповідної архітектури, налаштування гіперпараметрів, підготовку даних і оцінку результатів. Правильний вибір і налаштування всіх цих компонентів є критичним для досягнення високої точності і ефективності моделі.

Завдяки своїй потужності і універсальності, згорткові нейронні мережі продовжують залишатися однією з найпопулярніших і найпотужніших архітектур у галузі комп'ютерного зору. Їх розвиток і вдосконалення відкривають нові можливості для вирішення складних задач, що раніше були недоступні для автоматичного оброблення. Вони знаходять застосування у все більшій кількості сфер, змінюючи наші уявлення про те, що можуть досягти сучасні технології.

У розробці будь-якої системи на основі згорткових нейронних мереж важливо також враховувати етичні та соціальні аспекти. Це включає забезпечення приватності даних, уникнення дискримінації і забезпечення прозорості алгоритмів. Використання CNN повинно бути відповідальним і орієнтованим на досягнення позитивного впливу на суспільство.

Завдяки своїй здатності автоматично виділяти і аналізувати складні ознаки зображень, згорткові нейронні мережі є потужним інструментом для розпізнавання і обробки візуальної інформації. Вони дозволяють створювати системи, які можуть ефективно вирішувати широкий спектр задач, від автоматичної класифікації зображень до складних систем відеоспостереження і

автономного керування транспортними засобами. Технології на основі CNN продовжують розвиватися і вдосконалюватися, відкриваючи нові можливості для автоматизації і підвищення ефективності у різних галузях.

1.3.2 Рекурентні нейронні мережі

Також EasyOCR використовує рекурентні нейронні мережі (Recurrent Neural Networks, RNN), що є важливим компонентом сучасних систем для розпізнавання тексту. Рекурентні нейронні мережі (RNN) отримують інформацію не тільки від попередніх шарів, а й від своїх вузлів. Вони навчаються запам’ятовувати також послідовність вхідних даних. Ці мережі відрізняються від інших типів нейронних мереж своєю здатністю обробляти послідовності даних завдяки зворотним зв'язкам, що дозволяють їм запам'ятовувати інформацію з попередніх кроків. Ця особливість робить RNN надзвичайно корисними для задач, де важлива послідовність вхідних даних, таких як обробка природної мови, машинний переклад, розпізнавання мови та багато інших.

Основною особливістю RNN є наявність зворотного зв'язку, що дозволяє передавати інформацію з одного часу до іншого. Це дозволяє мережі запам'ятовувати контекст інформації, що є важливим для задач, де порядок вхідних даних має значення. Наприклад, у випадку розпізнавання тексту, кожен символ впливає на розуміння наступних символів, тому здатність запам'ятовувати попередні символи є критичною.

Одним із найпростіших варіантів RNN є стандартна рекурентна нейронна мережа, де кожен нейрон пов'язаний із самим собою через часовий проміжок. Однак, такі мережі мають обмеження, пов'язані з проблемою зникання градієнтів, що ускладнює навчання довгих послідовностей. Щоб подолати цю проблему, були розроблені більш складні архітектури, такі як довготривала короткочасна пам'ять (Long Short-Term Memory, LSTM) та градієнтно нормалізовані рекурентні

нейронні мережі (Gated Recurrent Unit, GRU). Схема рекурентної нейронної мережі зображена на рисунку 1.3

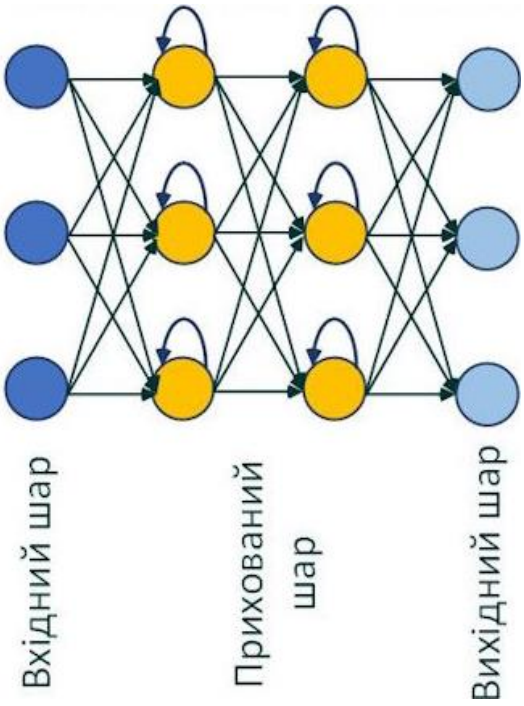

Рисунок 1.3 — Схема рекурентної нейронної мережі

LSTM є однією з найпоширеніших і найефективніших архітектур RNN. Основна інновація LSTM полягає у використанні комірок пам'яті та механізмів гейтування, що дозволяють контролювати потік інформації через мережу. LSTM складається з трьох основних компонентів: вхідного гейту, гейту забуття та вихідного гейту. Вхідний гейт визначає, яка нова інформація буде додана до комірки пам'яті. Гейт забуття контролює, яка частина інформації з комірки пам'яті буде збережена, а яка - видалена. Вихідний гейт визначає, яка інформація з комірки пам'яті буде використана для обчислення виходу мережі. Ці механізми дозволяють LSTM ефективно запам'ятовувати та забувати інформацію, що робить її дуже потужною для обробки довгих послідовностей даних.

GRU є спрощеною версією LSTM, яка також використовує гейти для контролю потоку інформації, але має менше параметрів і простішу архітектуру.

GRU має два основні гейти: оновлення та гейт забуття. Це спрощує обчислення та зменшує обсяг необхідних ресурсів для навчання мережі, зберігаючи при цьому здатність ефективно обробляти послідовні дані. GRU часто використовується у задачах, де важливо знайти компроміс між точністю та обчислювальною ефективністю.

Однією з ключових переваг RNN є їх здатність обробляти дані змінної довжини. Це особливо корисно для задач обробки природної мови, де довжина речень або документів може сильно варіюватися. Завдяки зворотному зв'язку, RNN можуть динамічно адаптуватися до різної довжини вхідних послідовностей, що робить їх дуже гнучкими і універсальними.

Рекурентні нейронні мережі знаходять широке застосування у багатьох галузях. У сфері обробки природної мови RNN використовуються для задач машинного перекладу, де важливо зберігати контекст перекладу на довгих послідовностях. RNN також використовуються для розпізнавання мови, де вони допомагають перетворювати звукові сигнали у текст. У сфері генерації тексту RNN можуть створювати зв'язні та граматично правильні тексти на основі заданих вхідних даних. Вони також застосовуються у задачах аналізу настроїв, де допомагають визначати емоційний тон тексту.

1.3.3 Особливості навчання рекурентних нейронних мереж

Навчання RNN включає кілька етапів, таких як пряма передача (forward pass), зворотне поширення помилки (backpropagation through time, BPTT) і оновлення ваг (weight update) на основі градієнтного спуску. Процес навчання потребує великої кількості даних та обчислювальних ресурсів, особливо для складних архітектур, таких як LSTM та GRU. Використання графічних процесорів (GPU) є стандартною практикою для прискорення обчислень та підвищення ефективності навчання.

Одна з основних проблем при навчанні RNN – це проблема зникання та вибуху градієнтів, яка виникає через багаторазове множення малих або великих чисел під час зворотного поширення помилки. Це може призводити до дуже малих або дуже великих значень градієнтів, що ускладнює або навіть робить неможливим оновлення ваг. Для вирішення цієї проблеми використовуються спеціальні техніки, такі як нормалізація градієнтів, використання LSTM та GRU, а також застосування різних методів регуляризації, таких як відкидання (dropout).

Важливо також зазначити, що розробка та застосування RNN потребує глибокого розуміння специфіки задачі, для якої вони застосовуються. Це включає вибір відповідної архітектури, налаштування гіперпараметрів, підготовку даних та оцінку результатів. Правильний вибір та налаштування всіх цих компонентів є критичним для досягнення високої точності та ефективності моделі.

Рекурентні нейронні мережі також можуть поєднуватися з іншими типами нейронних мереж для створення гібридних моделей, які можуть вирішувати більш складні задачі. Наприклад, RNN можуть бути поєднані зі згортковими нейронними мережами (CNN) для обробки послідовних даних, що містять візуальні ознаки, або з трансформерами для підвищення ефективності обробки природної мови. Такий підхід дозволяє використовувати переваги різних архітектур і створювати більш потужні та універсальні моделі.

Завдяки своїй здатності ефективно обробляти послідовні дані, рекурентні нейронні мережі продовжують залишатися одним із найпотужніших інструментів у галузі машинного навчання. Їх розвиток і вдосконалення відкривають нові можливості для вирішення складних задач, що раніше були недоступні для автоматичного оброблення. Вони знаходять застосування у все більшій кількості сфер, змінюючи наші уявлення про те, що можуть досягти сучасні технології.

У розробці будь-якої системи на основі рекурентних нейронних мереж важливо також враховувати етичні та соціальні аспекти. Це включає забезпечення приватності даних, уникнення дискримінації і забезпечення прозорості алгоритмів. Використання RNN повинно бути відповідальним і орієнтованим на досягнення позитивного впливу на суспільство.

Завдяки своїй здатності запам'ятовувати та аналізувати послідовні дані, рекурентні нейронні мережі є потужним інструментом для обробки інформації, що має послідовний характер. Вони дозволяють створювати системи, які можуть ефективно вирішувати широкий спектр задач, від машинного перекладу до розпізнавання мови та генерації тексту. Технології на основі RNN продовжують розвиватися і вдосконалюватися, відкриваючи нові можливості для автоматизації і підвищення ефективності у різних галузях.

1.4 Розпізнавання образів

Розпізнавання образів — розділ кібернетики, що розвиває теоретичні основи та методи класифікації та ідентифікації предметів, явищ, процесів, сигналів, ситуацій тощо об'єктів, які характеризуються кінцевим набором деяких властивостей і ознак. Такі завдання вирішуються досить часто, наприклад, під час переходу чи проїзду вулиці за сигналами світлофора. Розпізнавання кольору лампи світлофора і знання правил дорожнього руху дозволяє прийняти правильне рішення про те, можна чи не можна переходити вулицю в даний момент.

У процесі біологічної еволюції багато тварин за допомогою зорового та слухового апарату вирішили завдання розпізнавання образів досить добре. Створення штучних систем розпізнавання образів залишається складною теоретичною та технічною проблемою. Необхідність у такому розпізнаванні виникає в різних областях — від військової справи і систем безпеки до оцифрування різноманітних аналогових сигналів.

Приклади завдань розпізнавання образів:

— розпізнавання літер;

— розпізнавання штрих-кодів;

— розпізнавання автомобільних номерів;

— розпізнавання осіб;

— розпізнавання мови;

— розпізнавання зображень.

Для оптичного розпізнавання образів можна застосувати метод перебору виду об'єкта під різними кутами, масштабами, зсувами тощо. буд. Для літер потрібно перебирати шрифт, властивості шрифту тощо.

Другий підхід — знайти контур об'єкта та дослідити його властивості (зв'язність, наявність кутів тощо).

Ще один підхід — використовувати штучні нейронні мережі. Цей метод вимагає або великої кількості прикладів завдання розпізнавання (з правильними відповідями), або спеціальної структури нейронної мережі, що враховує специфіку цього завдання.

Ф. Розенблатт вводячи поняття про модель мозку, завдання якої полягає в тому, щоб показати, як у деякій фізичній системі, структура та функціональні властивості якої відомі, можуть виникати психологічні явища — описав найпростіші експерименти з розрізнення. Дані експерименти цілком відносяться до методів розпізнавання образів, але відрізняються тим, що алгоритм рішення не детермінований.

Найпростіший експеримент, на основі якого можна отримати психологічно значущу інформацію про деяку систему, зводиться до того, що моделі пред'являються два різні стимули і потрібно, щоб вона реагувала на них по-різному. Метою такого експерименту може бути дослідження можливості їхнього спонтанного розрізнення системою за відсутності втручання з боку

експериментатора, або, навпаки, вивчення примусового розрізнення, при якому експериментатор прагне навчити систему проводити необхідну класифікацію.

У досвіді з навчанням перцептрон зазвичай пред'являється деяка послідовність образів, в яку входять представники кожного з класів, що підлягають розрізненню. Відповідно до деякого правила модифікації пам'яті правильний вибір реакції підкріплюється. Потім перцептрону пред'являється контрольний стимул визначається ймовірність отримання правильної реакції для стимулів даного класу. Залежно від того, збігається або не збігається обраний контрольний стимул з одним із образів, які використовувалися в навчальній послідовності, отримують різні результати:

— якщо контрольний стимул не збігається з жодним із навчальних стимулів, то експеримент пов'язаний не тільки з чистим розрізненням, але включає і елементи узагальнення;

— якщо контрольний стимул збуджує деякий набір сенсорних елементів, зовсім відмінних від тих елементів, які активізувалися при дії раніше пред'явлених стимулів того ж класу, експеримент є дослідженням чистого узагальнення.

Перцептрони не мають здатність до чистого узагальнення, але вони цілком задовільно функціонують в експериментах з розрізнення, особливо якщо контрольний стимул досить близько збігається з одним із образів, щодо яких перцептрон вже накопичив певний досвід.

1.5 Оптичне розпізнавання символів

Оптичне розпізнавання символів або оптичний зчитувач символів (OCR) — це електронне або механічне перетворення зображень надрукованого, рукописного або друкованого тексту в машинно-кодований текст, чи то зі сканованого документа, фотографії документа, фотографії сцени (наприклад,

текст на вивісках і рекламних щитах у альбомній фотографії) або з тексту субтитрів, накладеного на зображення (наприклад: з телевізійної трансляції) [1].

Широко використовується як форма введення даних із друкованих паперових записів даних — будь то паспортні документи, рахунки-фактури, банківські виписки, комп'ютеризовані квитанції, візитні картки, пошта, друковані дані чи будь-яка відповідна документація — це поширений метод оцифрування друкованих текстів, щоб їх можна електронно редагувати, шукати, зберігати більш компактно, відображати в Інтернеті та використовувати в машинних процесах, таких як когнітивні обчислення, машинний переклад, (витягнуте) перетворення тексту в мовлення, ключові дані та аналіз тексту. OCR — це галузь досліджень у сфері розпізнавання образів, штучного інтелекту та комп'ютерного зору.

Ранні версії потрібно було навчати із зображеннями кожного символу та працювати над одним шрифтом за раз. Удосконалені системи, здатні виробляти високу точність для більшості шрифтів, зараз є звичайними, а також підтримують різні вхідні формати файлів зображень [2]. Деякі системи здатні відтворювати відформатований вихід, який дуже наближений до оригінальної сторінки, включаючи зображення, стовпці та інші нетекстові компоненти.

У 1974 році Рей Курцвейл заснував компанію Kurzweil Computer Products, Inc. і продовжив розробку багатошрифтового OCR, який міг розпізнавати текст, надрукований практично будь-яким шрифтом. (Курцвейлу часто приписують винахід багатошрифтового OCR, але його використовували компанії, зокрема CompuScan, наприкінці 1960-х і 1970-х років [3,6].) Курцвейл використав цю технологію, щоб створити машину для читання для сліпих людей. нехай комп'ютер читає їм текст вголос. Пристрій включав планшетний сканер типу CCD і синтезатор тексту в мову. 13 січня 1976 року готовий продукт був представлений під час широко розповсюдженої прес-конференції під

керівництвом Курцвейла та лідерів Національної федерації сліпих. У 1978 році компанія Kurzweil Computer Products почала продавати комерційну версію оптичного персонажа. комп'ютерна програма розпізнавання. LexisNexis був одним із перших клієнтів і купив програму для завантаження юридичних паперів і новин до своїх онлайн-баз даних, що зароджувалися. Через два роки Курцвейл продав свою компанію Xerox, яка згодом відокремила її як Scansoft, яка об’єдналася з Nuance Communications.

У 2000-х OCR став доступним онлайн як послуга (WebOCR), у середовищі хмарних обчислень і в мобільних додатках, як-от переклад іншомовних знаків у реальному часі на смартфоні. З появою смартфонів і смарт-окулярів оптичне розпізнавання символів можна використовувати в додатках мобільних пристроїв, підключених до Інтернету, які витягують текст, знятий за допомогою камери пристрою. Ці пристрої, які не мають вбудованої функції оптичного розпізнавання символів, зазвичай використовують OCR API для вилучення тексту з файлу зображення, захопленого пристроєм [7,8]. OCR API повертає витягнутий текст разом із інформацією про розташування виявленого тексту на вихідному зображенні назад до програми пристрою для подальшої обробки (наприклад, перетворення тексту в мовлення) або відображення.

Для більшості поширених систем письма доступні різні комерційні системи оптичного розпізнавання символів із відкритим кодом, зокрема латиниця, кирилиця, арабська, іврит, індійська, бенгальська (бенгальська), деванагарі, тамільська, китайська, японська та корейська.

Механізми оптичного розпізнавання символів (OCR) були розроблені в програмних додатках, що спеціалізуються на різних темах, таких як квитанції, рахунки-фактури, чеки та юридичні платіжні документи.

Програмне забезпечення можна використовувати для:

— введення даних для ділових документів, (чеки, паспорти, рахунки-фактури, банківські виписки та квитанції);

— автоматичне розпізнавання номерних знаків;

— розпізнавання паспортів та отримання інформації в аеропортах;

— автоматичне вилучення ключової інформації зі страхових документів;

— розпізнавання дорожніх знаків [9];

— вилучення інформації візитної картки в список контактів [10];

— створення текстових версій друкованих документів, напр. сканування книги для проекту Гутенберг;

— створення електронних зображень друкованих документів доступними для пошуку, напр. Книги Google;

— перетворення рукописного тексту в режимі реального часу для керування комп’ютером (обчислення пером);

— знищення або перевірка надійності систем захисту від ботів CAPTCHA, хоча вони спеціально розроблені для запобігання OCR [11,12,13];

— допоміжні технології для сліпих і людей із вадами зору;

— написання інструкцій для транспортних засобів шляхом ідентифікації зображень САПР у базі даних, які відповідають конструкції транспортного засобу, оскільки він змінюється в реальному часі;

— перетворення відсканованих документів на PDF-файли доступними для пошуку;

Методи оптичного та інтелектуального розпізнавання тексту:

— оптичне розпізнавання символів;

— оптичне розпізнавання слів;

— інтелектуальне розпізнавання символів;

— інтелектуальне розпізнавання слів.

Оптичне розпізнавання символів (OCR) – націлюється на машинописний текст. Розпізнавання тексту відбувається по одному гліфу або символу за раз. Застосовують для перетворення різних друкованиї документів у цифровий формат, але для рукописного, пошкодженого чи із складним шрифтом текстом ефективніть розпізнавання зменшується.

Оптичне розпізнавання слів (OWR) – націлюється на машинописний текст, одне слово за раз, тобто розпізнає текст як цілі слова, а не як окремі символи. Застосовують для мов, які використовують пробіл як роздільник слів, але для тексту де слова не розділяються пробілами точність розпізнавання знижується.

Інтелектуальне розпізнавання символів (ICR) – також націлено на рукописний шрифт або курсивний текст по одному гліфу або символу за раз, зазвичай залучають машинне навчання. Застосовують для перетворення рукописних документів, але може зменшитись точність розпізнавання через різні стилі письма.

Інтелектуальне розпізнавання слів (IWR) – також націлено на рукописний шрифт або курсивний текст, одне слово за раз. Це особливо корисно для мов, де гліфи не розділені пробілами, але якщо слова сильно зливаються, то ефективність розпізнавання тексту знижується.

OCR, як правило, є автономним процесом, який аналізує статичний документ. Існують хмарні служби, які надають онлайн-службу OCR API. Аналіз руху рукописного тексту можна використовувати як вхідні дані для розпізнавання рукописного тексту [14]. Замість того, щоб просто використовувати форми гліфів і слів, ця техніка здатна зафіксувати рух, наприклад порядок, у якому намальовано сегменти, напрямок і схему опускання та підняття пера. Ця додаткова інформація може зробити процес точнішим. Ця технологія також відома як «онлайн-розпізнавання символів», «динамічне

розпізнавання символів», «розпізнавання символів у реальному часі» та «інтелектуальне розпізнавання символів».

1.5.1 Розпізнавання тексту

Існує два основних типи основного алгоритму OCR, який може створювати ранжований список символів-кандидатів [23];

Зіставлення матриці передбачає порівняння зображення із збереженим гліфом на основі піксель за пікселем; це також відомо як зіставлення шаблонів, розпізнавання шаблонів або кореляція зображень. Це залежить від того, що вхідний гліф правильно ізольований від решти зображення, а збережений гліф має подібний шрифт і той самий масштаб. Ця техніка найкраще працює з машинописним текстом і погано працює, коли зустрічаються нові шрифти. Це техніка раннього фізичного оптичного розпізнавання символів на основі фотоелементів, реалізована досить безпосередньо.

Екстракція функцій розкладає гліфи на «об’єкти», як-от лінії, замкнуті цикли, напрямок ліній і перетини ліній. Функції вилучення зменшують розмірність представлення та роблять процес розпізнавання обчислювально ефективним. Ці функції порівнюються з абстрактним векторним представленням символу, яке може зводитися до одного або кількох прототипів гліфів. Загальні методи виявлення ознак у комп’ютерному зорі застосовні до цього типу OCR, який зазвичай можна побачити в «інтелектуальному» розпізнаванні рукописного тексту та більшості сучасних програм OCR [24]. Класифікатори найближчих сусідів, такі як алгоритм k-найближчих сусідів, використовуються для порівняння характеристик зображення зі збереженими функціями гліфа та вибору найближчої відповідності [25].

Такі програми, як Cuneiform і Tesseract, використовують двопрохідний підхід до розпізнавання символів. Другий прохід відомий як адаптивне

розпізнавання. У ньому використовуються форми літер, розпізнані з високою впевненістю під час першого проходу, щоб краще розпізнавати решту літер під час другого проходу. Це корисно для незвичайних шрифтів або сканованих зображень низької якості, де шрифт спотворений (наприклад, розмитий або вицвілий) [22].

Станом на грудень 2016 року сучасне програмне забезпечення OCR включає Google Docs OCR, ABBYY FineReader і Transym [26]. Інші, такі як OCRopus і Tesseract, використовують нейронні мережі, які навчені розпізнавати цілі рядки тексту, а не зосереджуватися на окремих символах.

Техніка, відома як ітераційне OCR, автоматично обрізає документ на розділи на основі макета сторінки. Розпізнавання символів виконується для окремих розділів із використанням порогових значень рівня достовірності змінних символів, щоб максимально підвищити точність OCR на рівні сторінки. На цей метод був виданий патент Патентного відомства США [27].

Результат OCR можна зберегти у стандартизованому форматі ALTO, спеціальному XML-схемі, який підтримується Бібліотекою Конгресу США. Інші поширені формати включають hOCR і PAGE XML.

1.5.2 Подальша обробка розпізнавання тексту

Точність оптичного розпізнавання символів можна підвищити, якщо вихідні дані обмежені лексиконом – списком слів, які можуть зустрічатися в документі [15]. Це можуть бути, наприклад, усі слова англійської мови або більш технічний лексикон для певної галузі. Ця техніка може бути проблематичною, якщо документ містить слова, яких немає в лексиконі, наприклад власні іменники. Tesseract використовує свій словник, щоб впливати на крок сегментації символів для підвищення точності [22].

Вихідним потоком може бути звичайний текстовий потік або файл символів, але більш складні системи оптичного розпізнавання символів можуть зберегти оригінальний макет сторінки та створити, наприклад, анотований PDF-файл, який містить як оригінальне зображення сторінки, так і текстове представлення з можливістю пошуку.

Аналіз ближніх сусідів може використовувати частоти спільного зустрічання для виправлення помилок, зауважуючи, що певні слова часто зустрічаються разом [28]. Наприклад, "Вашингтон, округ Колумбія" як правило, набагато більш поширений в англійській мові, ніж "Washington DOC".

Знання граматики мови, яка сканується, також може допомогти визначити, чи слово ймовірно є дієсловом чи іменником, наприклад, що забезпечує більшу точність.

Алгоритм відстані Левенштейна також використовувався в пост-обробці OCR для подальшої оптимізації результатів OCR API [29].

1.6 Огляд існуючих рішень

Розпізнавання тексту з фотографій є актуальною задачею у сучасних додатках та системах, які працюють з великим обсягом інформації. Існує кілька популярних рішень, які забезпечують високу точність розпізнавання тексту зображень.

Один із популярних програмних рішень - це Microsoft Cognitive Services, який надає API для розпізнавання тексту на фотографіях. Його основною перевагою є висока точність розпізнавання та широкі можливості інтеграції з іншими сервісами Microsoft. Проте варто враховувати, що використання Microsoft Cognitive Services може бути вартим для великих обсягів даних.

Ще одним програмним рішенням є Amazon Textract, який є частиною сервісів AWS та призначений для розпізнавання тексту на фотографіях та

документах. Він відомий своєю високою швидкістю роботи, підтримкою різних мов та форматів. Це робить його дуже привабливим для використання у великих масштабах, де потрібна висока продуктивність і швидкість обробки.

Крім цього, варто звернути увагу на Google Cloud Vision API, який надає можливість розпізнавання тексту на фотографіях та інших зображеннях. Він відомий своєю високою точністю та різноманітністю функцій, таких як визначення об'єктів та сцен на зображенні. Завдяки потужним алгоритмам машинного навчання, Google Cloud Vision API може надавати точні результати та підтримувати різні мови, що робить його одним з найпопулярніших рішень на ринку.

Ще однією цікавою альтернативою є ABBYY FineReader, який спеціалізується на оптичному розпізнаванні символів (OCR). Він відомий своєю високою точністю та можливістю роботи з різними типами документів та мовами. ABBYY FineReader має тривалу історію розвитку та є одним з найбільш надійних інструментів для OCR, що робить його особливо корисним для корпоративного використання.

Нарешті, одним з варіантів є ABBYY Cloud OCR SDK, який також спеціалізується на оптичному розпізнаванні символів та надає можливість розпізнавання тексту з фотографій. Він відомий своєю високою точністю та можливістю роботи з різними мовами. Використання хмарних технологій дозволяє зменшити вимоги до апаратного забезпечення користувача, що робить це рішення доступним для широкого кола користувачів.

Крім наведених рішень, існують й інші, менш відомі, але не менш ефективні технології для розпізнавання тексту. Наприклад, Tesseract є одним з найпопулярніших безкоштовних інструментів для OCR. Цей інструмент, спочатку розроблений HP і пізніше підтримуваний Google, є відкритим і доступним для використання у різних проектах. Tesseract підтримує велику

кількість мов і є дуже гнучким, що робить його привабливим для розробників з обмеженим бюджетом.

Kraken OCR є ще одним відкритим інструментом для оптичного розпізнавання тексту, який фокусується на високій точності та підтримці різних мов, включаючи історичні і рідкісні. Цей інструмент розроблений з урахуванням потреб академічних досліджень і зручний для роботи з рукописними текстами та документами з архівів.

Програма EasyOCR також є одним з рішень, що заслужили визнання завдяки своїй простоті у використанні та високій точності. Вона підтримує понад 80 мов і є відкритим інструментом, що дозволяє розробникам легко інтегрувати функції OCR у свої додатки. EasyOCR використовує сучасні архітектури нейронних мереж, що забезпечує високу точність розпізнавання тексту навіть на складних зображеннях.

Додатково, у сфері мобільних додатків варто згадати такі інструменти, як Adobe Scan та CamScanner. Adobe Scan є додатком, що дозволяє користувачам сканувати документи та розпізнавати текст за допомогою технологій OCR. Цей додаток інтегрований з іншими сервісами Adobe, що робить його зручним для роботи з документами. CamScanner також є популярним мобільним додатком для сканування документів і розпізнавання тексту, відомим своєю простотою використання та високою якістю обробки зображень.

Ще одним важливим аспектом, який слід враховувати при виборі рішення для OCR, є підтримка спеціалізованих функцій, таких як розпізнавання рукописного тексту або математичних формул. Для таких задач існують спеціалізовані інструменти, наприклад, Mathpix, який дозволяє розпізнавати та конвертувати математичні формули у текстовий формат. Цей інструмент є дуже корисним для науковців та студентів, які працюють з науковою літературою.

Також важливо відзначити, що розвиток технологій штучного інтелекту та машинного навчання відкриває нові можливості для покращення якості та точності розпізнавання тексту. Сучасні нейронні мережі та алгоритми глибокого навчання дозволяють значно підвищити точність розпізнавання навіть на складних і низькоякісних зображеннях. Наприклад, використання глибоких згорткових нейронних мереж (CNN) для попередньої обробки зображень дозволяє виділяти важливі ознаки і покращувати якість розпізнавання тексту.

Таким чином, на ринку існує широкий спектр рішень для розпізнавання тексту з фотографій, кожне з яких має свої переваги і недоліки. Вибір конкретного інструменту залежить від специфіки задачі, вимог до точності, швидкості обробки, підтримки різних мов та інших факторів. Незалежно від обраного рішення, використання технологій OCR дозволяє значно підвищити ефективність обробки інформації і зробити її більш доступною для користувачів

1.7 Постановка задачі

Дана бакалаврська дипломна робота присвячена розробці додатку для розпізнавання тексту з фотографій з метою покращення доступності до інформації для людей з обмеженими можливостями, зокрема для незрячих та людей з порушеннями зору. Цей додаток є дуже важливим, оскільки дозволяє користувачам з обмеженими можливостями легко отримувати інформацію з фотографій, що може бути важливою у їхньому повсякденному житті та сприяти їхній більшій самостійності та інтеграції в суспільство.

Додаток буде мати наступний функціонал:

— зйомка фотографій;

— розпізнавання тексту;

— озвучення тексту.

Під зйомкою фотографій мається на увазі те, що користувач може зробити фотографію об'єкта, на якому знаходиться текст, за допомогою камери пристрою.

Розпізнавання тексту — це коли додаток автоматично розпізнає текст на фотографії та перетворює його в текстовий формат.

Озвучення тексту — це коли розпізнаний текст може озвучуватися синтезатором мови для зручності користувача.

Цей функціонал дозволить користувачам легко отримувати доступ до тексту на фотографіях і зробить їхнє повсякденне життя більш комфортним та самостійним.

2 АНАЛІЗ ТА ОБГРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ

2.1 Вибір архітектури і життєвий цикл системи

Модель водоспаду - це розподіл процесу розробки на послідовні фази, де кожна фаза передбачає певний набір завдань та ґрунтується на результатах попередньої фази. Цей підхід типовий для певних галузей інженерного проектування. У виправленні програмного забезпечення він зазвичай використовується як менш ітеративний та більш жорсткий метод, оскільки процес рухається в одному напрямку (зверху вниз), через послідовні фази, які включають в себе концепцію, ініціацію, аналіз, дизайн, розробку, тестування, впровадження та підтримку.

Модель водоспаду мала своє походження в промисловості та будівництві, де структуроване фізичне оточення призводило до високих витрат на зміни в проектуванні на ранніх стадіях розробки. Коли цей підхід був застосований до розробки програмного забезпечення, він був єдиним підходом для творчої роботи, що базується на знаннях.

Перша детальна презентація, яка описує використання таких етапів у розробці програмного забезпечення, була проведена Феліксом Торресом та Гербертом Д. Бенінгтоном на симпозіумі з передових методів програмування для цифрових комп'ютерів у 1956 році.

Хоча термін "водоспад" не був використаний в цій статті, перша офіційна детальна діаграма процесу, пізніше відома як "модель водоспаду", зазвичай приписується статті Вінстона В. Ройса 1970 року. У 1985 році Міністерство оборони Сполучених Штатів відобразило цей підхід у своєму стандарті DOD-STD-2167A для роботи з підрядниками щодо розробки програмного забезпечення, вимагаючи виконання шести етапів:

— аналіз вимог до програмного забезпечення;

— попередній проект;

— детальний дизайн;

— кодування та модульне тестування;

— інтеграція;

— тестування.

Вимоги до системи та програмного забезпечення перераховані в документі з вимогами до продукту.

Під час аналізу формуються моделі, схеми та бізнес-правила.

Під час дизайну створюється архітектура програмного забезпечення.

Кодування включає в себе розробку, перевірку та інтеграцію програмного забезпечення.

Тестування полягає в систематичному виявленні та виправленні дефектів.

Операції включають встановлення, міграцію, підтримку та технічне обслуговування повних систем.

Отже, модель водоспаду наголошує на тому, що перехід до наступної фази повинен відбуватися лише після того, як попередню фазу перевірено і переглянуто.

Існують різні модифіковані версії моделі водоспаду, включаючи остаточну модель Ройса, які можуть включати незначні або значні зміни в цьому процесі. Ці варіації можуть включати повернення до попередньої фази після виявлення недоліків або повний повтор фази проектування, якщо наступні фази виявилися недостатньо успішними.

Важливим аргументом на користь моделі водоспаду є те, що вона підкреслює значення документації, такої як вимоги та проектні документи, а також вихідний код. У менш структурованих та докладно задокументованих методологіях втрачена інформація, якщо члени команди покидають проект до його завершення. Модель водоспаду забезпечує можливість легкої

ознайомленості з проектом для нових членів команди або для випадків зміни команди, оскільки всі необхідні документи доступні.

Модель водоспаду забезпечує систематизований підхід, оскільки вона лінійно проходить через окремі, легкі для розуміння і пояснення фази, надаючи чіткі пункти в розробці. Саме ця структура робить її зрозумілою. Модель водоспаду також надає чіткі точки відсічі в процесі розробки, що полегшує відслідковування прогресу. Через це, вона часто використовується як вихідна точка для розуміння моделей розробки в багатьох підручниках і курсах з програмної інженерії.

2.1.1 Керування вимогами та модифіковані моделі розробки

Клієнти можуть не мати точно сформульованих вимог до продукту до того, як побачать його в дії, і це може викликати зміни у вимогах. Це може призвести до необхідності редизайну, перерозробки та повторного тестування, що призводить до додаткових витрат.

Розробники можуть не передбачити майбутніх труднощів під час створення нового програмного продукту або функцій. У таких випадках, було б краще переглянути проект, а не настоювати на виконанні проекту, який не враховує виявлені обмеження, вимоги або проблеми.

Деякі організації намагаються вирішити відсутність конкретних вимог від клієнтів шляхом залучення системних аналітиків до вивчення існуючих ручних систем і аналізу їх функціоналу. Проте в практиці може бути важко дотримуватися жорсткого розділення між системним аналізом і програмуванням, оскільки впровадження складної системи майже завжди призводить до виявлення проблем та крайових випадків, які системний аналітик не передбачив.

У відповідь на ці проблеми були запропоновані модифіковані моделі водоспаду, такі як "модель Сашимі", "модель з фазами, що перекриваються",

"модель з підпроектами" та "модель з зменшенням ризику". Модифіковані моделі водоспаду відповідають на критику "чистої" моделі водоспаду, шляхом впровадження змін у процес розробки програмного забезпечення для зменшення витрат і полегшення управління змінами.

2.1.2 Модель клієнт-сервер: принципи та використання

Модель клієнт-сервер, також відома як мережева архітектура клієнт-сервер, — це розподілена структура додатків, яка розподіляє завдання або навантаження між постачальниками ресурсів або послуг, які називаються серверами, і запитувачами послуг, які називаються клієнтами [1]. Часто клієнти та сервери спілкуються через комп’ютерну мережу на окремому обладнанні, але і клієнт, і сервер можуть знаходитися в одній системі. Хост сервера запускає одну або кілька серверних програм, які спільно використовують свої ресурси з клієнтами. Клієнт зазвичай не ділиться жодними своїми ресурсами, але запитує вміст або послугу на сервері. Тому клієнти ініціюють сеанси зв’язку з серверами, які очікують вхідних запитів. Прикладами комп’ютерних програм, які використовують модель клієнт-сервер, є електронна пошта, мережевий друк і Всесвітня павутина.

Характеристика «клієнт-сервер» описує взаємозв'язок програм, що співпрацюють у додатку. Серверний компонент надає функцію або послугу одному або багатьом клієнтам, які ініціюють запити на такі служби. Сервери класифікуються за послугами, які вони надають. Наприклад, веб-сервер обслуговує веб-сторінки, а файловий сервер — комп’ютерні файли. Спільним ресурсом може бути будь-яке програмне забезпечення та електронні компоненти серверного комп’ютера, від програм і даних до процесорів і пристроїв зберігання. Спільне використання ресурсів сервера є послугою.

Те, чи є комп’ютер клієнтом, сервером або обома, визначається характером програми, яка потребує функцій обслуговування. Наприклад, один комп’ютер може одночасно запускати веб-сервер і програмне забезпечення файлового сервера, щоб надавати різні дані клієнтам, які роблять різні типи запитів. Клієнтське програмне забезпечення також може спілкуватися з серверним програмним забезпеченням на тому самому комп’ютері [2]. Зв’язок між серверами, наприклад для синхронізації даних, іноді називають міжсерверним або міжсерверним зв’язком.

Загалом послуга є абстракцією ресурсів комп’ютера, і клієнту не потрібно турбуватися про те, як працює сервер під час виконання запиту та доставки відповіді. Клієнт має лише зрозуміти відповідь на основі загальновідомого протоколу програми, тобто зміст і форматування даних для запитуваної послуги.

Клієнти та сервери обмінюються повідомленнями за схемою запит-відповідь. Клієнт надсилає запит, а сервер повертає відповідь. Цей обмін повідомленнями є прикладом міжпроцесної комунікації. Щоб спілкуватися, комп’ютери повинні мати спільну мову, і вони повинні дотримуватися правил, щоб і клієнт, і сервер знали, чого очікувати. Мова та правила спілкування визначаються протоколом спілкування. Усі протоколи працюють на прикладному рівні. Протокол прикладного рівня визначає основні шаблони діалогу. Щоб ще більше формалізувати обмін даними, сервер може реалізувати інтерфейс прикладного програмування (API) [3]. API — це рівень абстракції для доступу до служби. Обмежуючи спілкування певним форматом вмісту, це полегшує аналіз. Абстрагуючи доступ, він полегшує міжплатформний обмін даними [4].

Сервер може отримувати запити від багатьох різних клієнтів за короткий проміжок часу. Комп’ютер може виконувати лише обмежену кількість завдань у будь-який момент і покладається на систему планування, щоб визначити

пріоритетність вхідних запитів від клієнтів для їх виконання. Щоб запобігти зловживанням і збільшити доступність, програмне забезпечення сервера може обмежити доступність для клієнтів. Атаки на відмову в обслуговуванні призначені для використання зобов'язань сервера обробляти запити, перевантажуючи його надмірною швидкістю запитів. Якщо конфіденційна інформація має передаватися між клієнтом і сервером, слід застосовувати шифрування.

«Серверне програмне забезпечення» стосується комп'ютерної програми, наприклад веб-сервера, яка працює на апаратному забезпеченні віддаленого сервера, доступного з локального комп'ютера, смартфона або іншого пристрою користувача. Операції можуть виконуватися на стороні сервера, оскільки вони потребують доступу до інформації чи функцій, недоступних на клієнті, або тому, що виконання таких операцій на стороні клієнта буде повільним, ненадійним або небезпечним.

Клієнтські та серверні програми можуть бути загальнодоступними, такими як безкоштовні або комерційні веб-сервери та веб-браузери, які спілкуються один з одним за допомогою стандартизованих протоколів. Або програмісти можуть написати власний сервер, клієнт і протокол зв'язку, які можна використовувати лише один з одним.

Операції на стороні сервера включають як ті, які виконуються у відповідь на запити клієнта, так і не орієнтовані на клієнта операції, такі як завдання обслуговування [6,7].

2.2 Проектування функціоналу системи

Необхідно ретельно дослідити варіанти використання, що забезпечить глибоке розуміння призначення та можливої функціональної діяльності інформаційної системи в майбутньому.

Найпростішою формою візуалізації взаємодії між користувачем та системою є діаграма використання, яка дозволяє ілюструвати взаємодію користувача з різними сценаріями використання системи. Ці діаграми можуть ідентифікувати різні типи користувачів системи та різні моделі використання. Вони зазвичай поєднуються з іншими видами діаграм для отримання повної картини.

Хоча кожну можливість можна детально розглядати, використання діаграм допомагає надати загальний огляд системи на вищому рівні. Ці діаграми спрощують комунікацію зі зацікавленими сторонами і допомагають їм краще зрозуміти, як система буде розроблятися. Якщо говорити відомими словами, "план використання — це суть вашої системи."

Завдяки їх простоті, плани використання можуть стати ефективним інструментом комунікації для зацікавлених сторін. Ці зображення намагаються відтворити реальний світ та допомагають зацікавленим сторонам зрозуміти, як система буде розроблятися. Дослідження показало, що використання діаграм передає наміри системи зацікавленим сторонам більш зрозуміло, ніж діаграми класів.

Основною метою використання діаграм є відображення динамічних аспектів системи. Для більш повного функціонального і технічного опису системи можуть використовуватися інші схеми та документи. Вони надають спрощене графічне представлення того, як система має функціонувати:

— межа системи представляє собою прямокутник з ім'ям та еліпсом (прецедент), але може опускатися в випадках відсутності корисної інформації;

— актор — стилізована роль особи, яка відображає набір користувачів, які взаємодіють з системою або іншими сутностями (актори не пов'язані один з одним);

— прецедент —еліпс з підписом, що вказує на системну операцію, яка виконується інтерфейсом користувача та призводить до певних результатів. Назва може бути описом того, "що" в системі виконується, а не "як".Прецеденти представляють різні сценарії використання системи.

На рисунку 2.1 зображено діаграму варіантів використання, яка описує можливі дії користувача в системі.

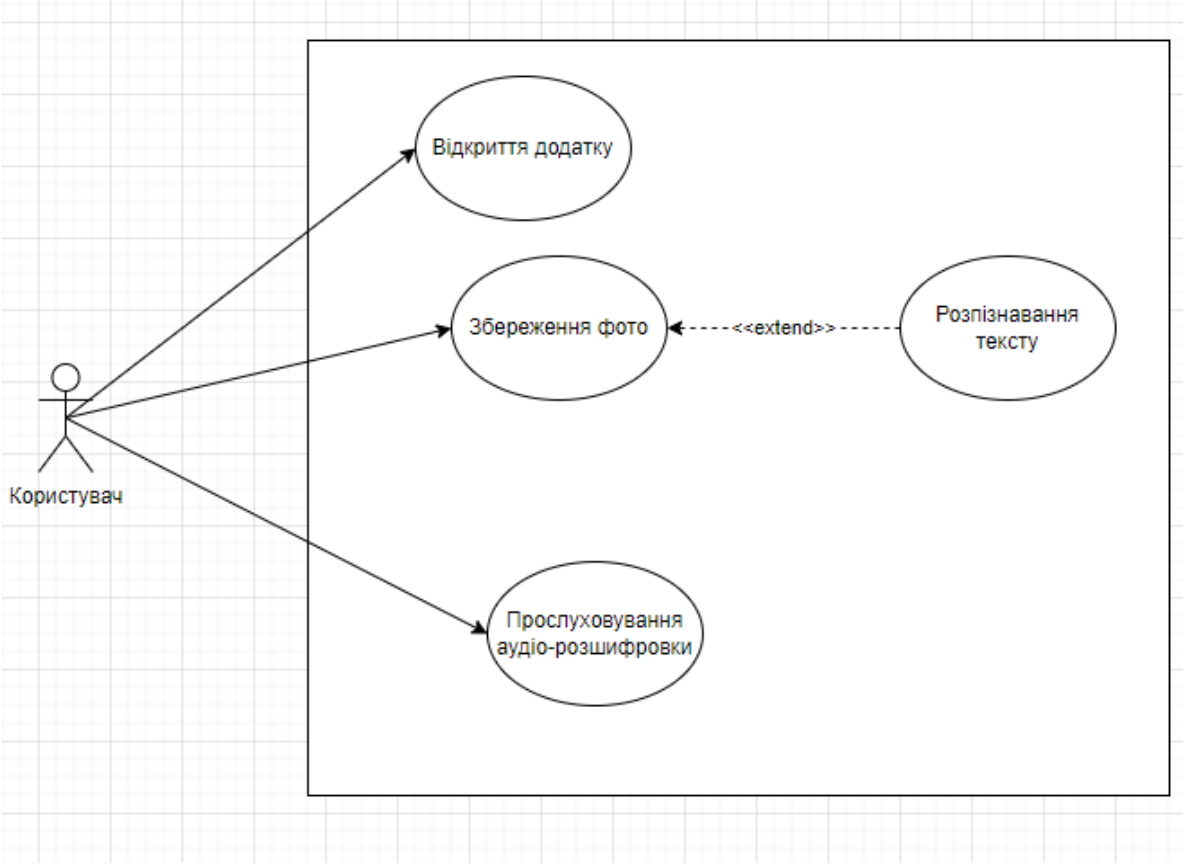

Рисунок 2.1 — Діаграма варіантів використання

2.3 Проектування внутрішньої будови

Під час розробки програмного забезпечення, зазвичай, зображення внутрішньої структури системи подається у формі діаграми груп та курсів, яка демонструє склад груп та модулів проекту та їх взаємодію.

У сфері програмної інженерії, діаграма груп курсів, згідно з уніфікованою мовою моделювання (UML), є статичною структурною діаграмою, що надає огляд структури системи, включаючи системні класи, їх атрибути, операції (або методи) та зв'язки між об'єктами.

Діаграми класів виступають як ключові будівельні блоки об'єктно-орієнтованого моделювання. Вони використовуються для загального концептуального моделювання структури програми, а також для більш детального моделювання з метою перетворення моделі в програмний код. Також ці діаграми можуть служити інструментом моделювання даних. Кожен клас на діаграмі класів представляє основні елементи програми та класи, які будуть реалізовані.

На графічному зображенні класу зазвичай можна виділити три окремі частини:

— верхній блок, який містить назву класу, зазвичай надруковану жирним шрифтом та вирівняну за центром (перша літера назви має бути великою);

— середній блок, де розташовані атрибути класу, які вирівнюються ліворуч, а перша літера їх назви повинна бути мала;

— нижній блок, який містить операції, які можуть бути виконані класом, операції також вирівнюються ліворуч, і перша літера їх назви має бути мала.

При проектуванні системи, визначення багатьох класів та їх групування в схему класів допомагають зрозуміти статичні відносини між ними. Для більш детального моделювання, категорія концептуального проектування зазвичай поділяється на декілька підкатегорій.

Асоціація описує взаємозв'язок між елементами моделі. Якщо зміна одного елемента може призвести до зміни іншого, то між ними існує асоціація. Це відношення є одностороннім і представлене пунктирною лінією зі стрілкою, що вказує на напрямок від відправлювача до отримувача.

Для подальшого опису системи, ці діаграми груп можуть бути доповнені діаграмами станів або становими автоматами UML.

Асоціація включає в себе різні типи: двосторонню, односторонню, агрегаційну (включаючи комбіновану агрегацію) та рефлексивну. Серед них найпоширеніші двосторонні та односторонні асоціації.

Наприклад, взаємозв'язок між класом "польот" і класом "літак" можна описати як двосторонню асоціацію. Ця асоціація відображає статичні відносини, що існують між об'єктами обох класів.

Агрегація представляє собою варіант відношення "має", і вона є більш специфічною за асоціацію. Агрегація виражає зв'язок, в якому один клас містить частину іншого класу. Наприклад, як показано на діаграмі, клас "професор" має асоціацію з класом "викладання". У якості типу асоціації, агрегація може мати назву та модифікації, подібні до асоціації. Але важливо відзначити, що агрегація обмежена бінарними відносинами, тобто це завжди асоціація між двома класами.

Особливістю агрегації є те, що класи, що містяться в агрегації, не мають сильної залежності від життєвого циклу контейнера. Навіть після знищення контейнера, елементи, що містяться в ньому, все ще існують.

Графічно агрегацію в UML зображують у вигляді порожнистого ромба, що з'єднується однією лінією з класом-господарем. Агрегація може розглядатися як семантично вищий рівень об'єкта, що взаємодіє з іншими об'єктами, хоча фізично складається з кількох менших об'єктів.

Наприклад, розглянемо взаємозв'язок між "бібліотекою" та "студентами". У цьому випадку, "студенти" можуть існувати незалежно від "бібліотеки", і відношення між ними може бути описане як агрегація, оскільки "студенти" є частиною "бібліотеки". Таке відношення можна позначити як сукупність.

Узагальнення визначає відношення "є спеціалізацією" між класами. Воно показує, що один клас (підтип) є більш конкретним представленням іншого класу

(супертипу). Наприклад, "люди" є підтипом "ссавців", і "ссавці" є супертипом "тварин". Графічно відношення узагальнення в UML позначаються порожнім трикутником, який з'єднується зі супертипом.

Відношення узагальнення часто отримують назву спадковістю або відносинами "є". У контексті цих відносин, суперклас (також відомий як базовий клас) можна називати "батьківським класом", "суперкласом", "базовим класом" або "базовим типом". Спеціалізовані класи, що успадковують від суперкласу, можуть називатися "дочірніми" підкласами, "похідними класами", "похідними типами", "успадкованими класами" або "успадкованими типами". Важливо зазначити, що ці терміни вживаються у контексті програмування та моделювання, і вони не мають жодного спільного з біологічними відносинами між батьком і дитиною.

Звідси ми можемо висловити відношення "А є типом В" такими прикладами, як "дуб є одним із видів дерев", "автомобіль є підкласом транспорту".

В моделюванні UML відношення реалізації вказують на те, що один елемент моделі (клієнт) втілює або реалізує функціональність, яка була задана іншим елементом моделі (постачальником). Це відношення графічно позначається порожнім трикутником, який з'єднаний інтерфейсом або деревом реалізаторів. Крім того, на діаграмі компонентів використовується графічна позначка "м'ячна розетка" для позначення реалізації. Важливо відзначити, що відношення реалізації зазвичай зображуються лише на діаграмах класів або компонентів, оскільки вони вказують на зв'язок між класами, інтерфейсами, компонентами і пакетами.

Залежність є слабкішою формою взаємодії, яка вказує на те, що один клас залежить від іншого і може використовувати його в певний момент часу. Відмінністю від асоціації є те, що залежність не передбачає наявність атрибутів

одного класу, які були б екземплярами іншого класу. Відношення реалізації і асоціації графічно представлені як лінії, що з'єднують класи, з додатковими символами на кінцях ліній для вказівки додаткових деталей відношення.

Класи сутності в моделюванні представляють інформацію, яку система обробляє, а також поведінку, пов'язану з цією інформацією. Графічне представлення класів сутностей зазвичай має форму кола з короткою лінією, прикріпленою до нижньої частини кола, або їх можна намалювати як звичайні класи зі стереотипом "сутність" над назвою класу.

На рисунку 2.2 зображено діаграму класів, яка відображає внутрішню будову клієнського додатку.

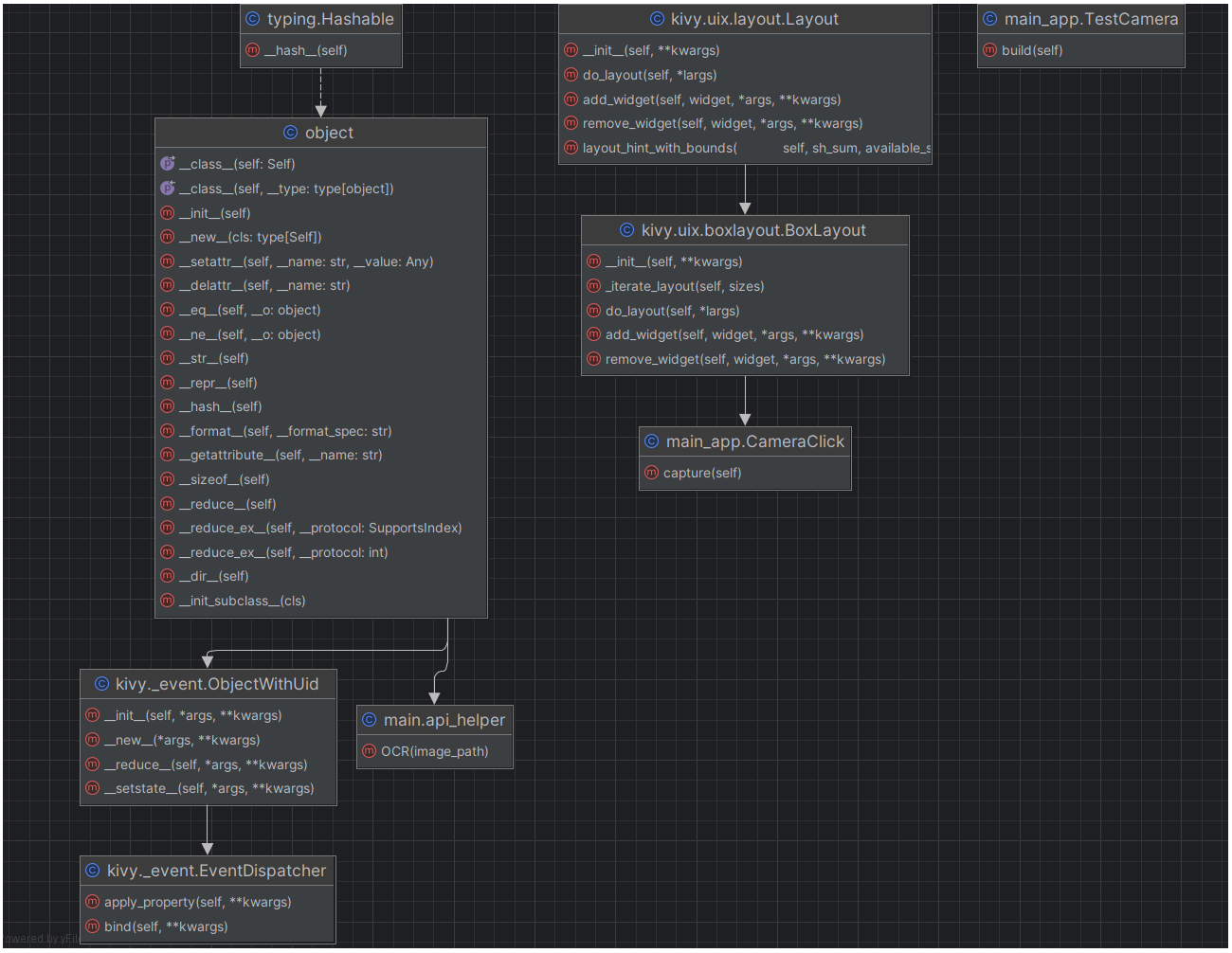

Рисунок 2.2 — Діаграма класів клієнтської частини

На рисунку 2.3 представлено діаграму класів серверної частини.

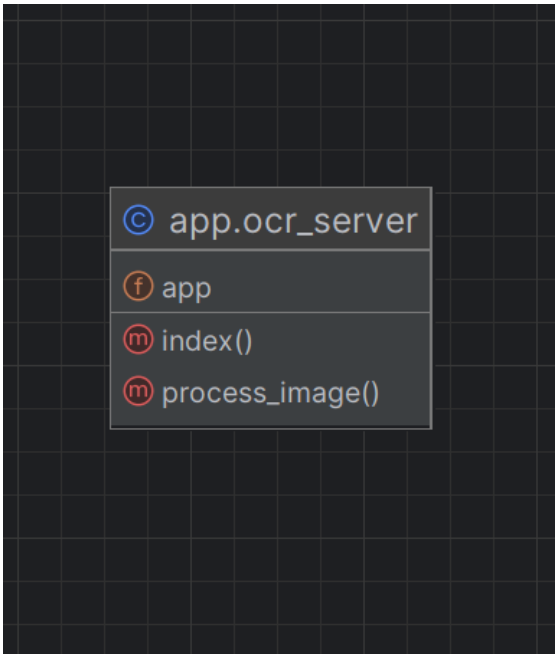

Рисунок 2.3 — Діаграма класів серверної частини

EasyOCR використовує складну архітектуру нейронних мереж, яка включає кілька типів шарів для забезпечення високої точності розпізнавання тексту. Основні компоненти архітектури включають згорткові нейронні мережі (CNN) для екстракції ознак і рекурентні нейронні мережі (RNN) для обробки послідовностей символів.

2.3.1 Архітектура CNN і RNN в EasyOCR

Згорткові нейронні мережі в EasyOCR використовуються для виділення важливих ознак з вхідних зображень. Типова структура CNN в EasyOCR може включати такі шари:

— згорткові шари;

— шари підвибірки;

— активуючі шари;

— шари нормалізації.

Згорткові шари (Convolutional Layers) — кілька згорткових шарів з різними кількістю фільтрів (наприклад, 32, 64, 128 фільтрів) для виділення різних рівнів ознак.

Шари підвибірки (Pooling Layers) — макс-пулінг або середнє пулінг для зменшення розмірності карт ознак і збереження найбільш важливих ознак.

Активуючі шари (Activation Layers) — функції активації, такі як ReLU (Rectified Linear Unit), для введення нелінійності в модель. Для того щоб допомогти моделі навчатися складнішим взаємозв’язкам між вхідними та вихідними даними.

Шари нормалізації (Normalization Layers) — допомагають стабілізувати та прискорити процес навчання нейронних мереж. Наприклад, шар нормалізації пакетів (Batch Normalization) нормалізує вихідні дані перед надходженням їх до наступного шару, що робить навчання більш стабільним та швидким.

Рекурентні нейронні мережі в EasyOCR використовуються для обробки послідовностей символів після виділення ознак згортковими шарами. Типова структура RNN в EasyOCR може включати:

— LSTM або GRU шари;

— шари нормалізації;

— повнозв'язні шари.

LSTM або GRU шари (LSTM/GRU Layers) — Один або кілька шарів LSTM (Long Short-Term Memory) або GRU (Gated Recurrent Unit) допомагають зберігати попередні дані та враховувати їх при обробці нових. Вони зберігають попередній контекст для кращого розуміння даних.

Шари нормалізації (Normalization Layers) — допомагають забезпечити, що дані ,які потрапляють у нейронну мережу не будуть занадто великими чи малими. Якщо дані мають різні масштаби, то це ускладнює навчання мережі.

Повнозв'язні шари (Fully Connected Layers) — Один або кілька шарів для остаточного перетворення ознак у передбачувані символи. Це останній етап після того, як дані пройшли через різні шари і потребують остаточної обробки перед виведенням кінцевих даних. Повнозв’язні шари групують всі дані, які вони отримали від попередніх шарів

2.4 Вибір технології для реалізації клієнтської та серверної частини

Python є мовою програмування високого рівня, яка зосереджується на читабельності коду за рахунок використання значних відступів. Ця мова підтримує кілька парадигм програмування, включаючи структурне, об'єктно-орієнтоване та функціональне програмування. Особливістю Python є його багата стандартна бібліотека, що забезпечує широкий спектр функціональності "з коробки".

Розробку Python розпочав Гвідо ван Россум наприкінці 1980-х років як наступника мови програмування ABC. Перша версія Python 0.9.0 була випущена у 1991 році. Особливістю Python 2.0, випущеного у 2000 році, були нові можливості, такі як розширений синтаксис для керування списками та система збору сміття. Остання версія Python 2.7.18 була офіційно припинена у 2020 році.

Python 3.0, представлений у 2008 році, став основною гілкою мови. Він вніс значні зміни, що зробили його не повністю сумісним із попередньою версією, але це було необхідно для подальшого розвитку мови.

Гвідо ван Россум, засновник Python, був її провідним розробником до 2018 року. Він відіграв ключову роль у розвитку мови та її спільноти. В січні 2019 року активні розробники ядра Python обрали раду директорів, до якої входили Бретт Кеннон, Нік Коглан, Баррі Варшав, Керол Віллінг та сам Гвідо ван Россум. Однак у 2020 році ван Россум вирішив не брати участь у виборах до ради директорів.

Python 2.0, який вніс значні нововведення у мову, особливо в області управління пам'яттю та підтримки Unicode, зіграв важливу роль у її розвитку і популяризації. Python продовжує залишатися однією з найпопулярніших мов програмування, завдяки своїй універсальності, широкій підтримці та активній спільноті.

Python 3.0, представлений у 2008 році, вніс значні зміни у мову, що зробило його не повністю сумісним з Python 2. Ці зміни були спрямовані на поліпшення мови та усунення деяких недоліків попередніх версій. Щоб полегшити перехід з Python 2 на Python 3, була розроблена утиліта 2to3, яка автоматизує процес конвертації коду.

Планування припинення підтримки Python 2.7 було відтерміновано до 2020 року через значну кількість існуючого коду на Python 2, який було б складно перенести на новішу версію. Після завершення підтримки, Python 2.7 більше не отримуватиме оновлень безпеки або інших виправлень, і спільнота зосередилась на підтримці Python 3.6 і новіших версій.

Python є мовою з багатопарадигмовим підходом. Він повністю підтримує об'єктно-орієнтоване та структурне програмування та пропонує можливості функціонального та аспектно-орієнтованого програмування. Це включає метапрограмування та метаоб'єктний підхід, а також динамічний збір сміття і динамічне розділення імен.

Python також активно підтримує функціональне програмування, включаючи функції фільтрації, відображення, скорочення, ітератори. Стандартна бібліотека містить модулі, такі як itertools і functools, які реалізують функціональні інструменти, запозичені від мов Haskell та Standard ML.

Концепція "Дзену Python" (PEP 20) узагальнює філософію та основні принципи Python.

Це включає такі підходи та вислови:

— краса краща за некрасивість;

— явність краща за неявність;

— простота краща за складність;

— складність краща за заплутаність;

— читабельність має значення;

— особливі випадки не настільки особливі, щоб порушувати правила;

— практичність важливіша за чистоту;

— помилки не повинні проходити непоміченими;

— якщо неясно, чи є помилка — це помилка.

Ці принципи відображають філософію Python як мови, орієнтованої на читабельність, простоту та ясність в кодуванні.

Python, як мова програмування, відзначається своєю розширюваністю та модульністю, дозволяючи легко інтегрувати різні програмовані інтерфейси та розширювати функціональність існуючих програм. Гвідо ван Россум, творець Python, заклав основи дизайну мови, орієнтуючись на читабельність, зрозумілу синтаксичну структуру та простоту вибору методів кодування.

Python має декілька особливостей:

а) модульний підхід;

б) фундаментальні принципи дизайну;

в) філософія Python;

г) оптимізація та продуктивність;

д) розвага у програмуванні;

е) спільнота Python (Pythonists);

ж) синтаксис Python;

з) основні оператори Python:

— оператор присвоєння =;

— умовний оператор if, разом з else та elif;

— оператор циклу for для ітерацій;

— оператор циклу while для виконання коду, поки умова є істинною.

Модульний підхід — Python підтримує модульність, дозволяючи розробникам легко додавати або розширювати функціональність.

Фундаментальні принципи дизайну — Python зосереджений на читабельності коду, що дозволяє розробникам легше розуміти та підтримувати програми.

Філософія Python — відрізняється від філософії Perl, Python слідує принципу "Повинен бути один очевидний спосіб зробити це."

Оптимізація та продуктивність — розробники Python уникають передчасної оптимізації, зосереджуючись на читабельності та підтримці коду. Водночас, для оптимізації швидкості можуть використовуватися інші інтерпретатори, такі як PyPy, або розширення, написані на C.

Розвага у програмуванні — важливою ціллю Python є забезпечення задоволення від процесу програмування, відображено в його назві, натхненній британською комедійною групою "Монті Пайтон".

Спільнота Python (Pythonists) — Python сприяє створенню активної та знаної спільноти користувачів.

Синтаксис Python — відмінною особливістю Python є використання відступів для визначення блоків коду, що робить код більш читабельним і структурованим.3

Python відрізняється своєю універсальністю та багатопарадигмовістю, підтримуючи різноманітні стилі програмування і надаючи потужні інструменти для розв'язання широкого спектру задач.

Python надає набір потужних операторів, які роблять мову гнучкою та виразною. Нижче наведений докладний опис найбільш важливих операторів.

Оператор "try" — використовується для перехоплення та обробки винятків у коді. Він дозволяє виконати блок коду та обробити виняток, якщо він виникає, забезпечуючи більш безпечне виконання програми.

Оператор "raise" — дозволяє генерувати винятки вручну, вказуючи на помилки або неправильні стани в програмі.

Оператор "class" — визначає новий клас, створюючи шаблон для об'єктів з однаковою структурою і поведінкою.

Оператор "def" — використовується для визначення функцій та методів, дозволяючи групувати код, який виконує певну задачу.

Оператор "with" — введений у Python 2.5, використовується для обгортання виконання блоку коду з методами, що визначені в контекстному менеджері, який дозволяє автоматично керувати ресурсами.

Оператор "break" — використовується для виходу з циклу передчасно.

Оператор "continue" — переходить до наступної ітерації циклу, пропускаючи решту коду в поточній ітерації.

Оператор "del" — видаляє об'єкт, звільняючи пам'ять і знищуючи посилання на цей об'єкт.

Оператор "pass" — використовується як заповнювач для порожніх блоків коду та не виконує жодної дії.

Оператор "assert" — дозволяє перевірити певну умову і генерувати виняток, якщо умова не виконується.

Оператор "yield" — використовується у функціях-генераторах для повернення значення, зберігаючи стан функції для наступних викликів.

Оператор "return" — використовується для повернення значення з функції.

Оператори імпорту — дозволяють включити модулі та їх компоненти у програму для використання їх функцій та об'єктів.

Оператор присвоєння ("=") — використовується для прив'язування значення до змінної.

Python відомий своєю динамічною типізацією, що дозволяє змінним бути пов'язаними з різними типами об'єктів у різний час. Це сприяє гнучкості та зменшує необхідність жорсткого визначення типів даних, як у статично типізованих мовах.

Python не підтримує оптимізацію хвостових викликів або першокласні розширення, і, за словами Гвідо ван Россума, ніколи не буде. Однак, розширивши генератор Python, у версії 2.5 надається краща підтримка функцій, подібних до корутина. До 2,5 генераторів є ледачими ітераторами, інформація передається від генераторів в одному напрямку. З Python 2.5 ви можете передавати інформацію назад до функції генератора, а з Python 3.3 ви можете передавати інформацію через кілька рівнів стека.

Python має унікальні особливості та синтаксис, що відрізняють його від інших мов програмування, таких як C та Java. Нижче наведений докладний опис деяких з цих особливостей.

Оператори додавання, віднімання та множення виконуються так само, як у більшості мов. Однак, для ділення Python використовує два різні оператори: цілочисельне ділення // та ділення з плаваючою точкою /. Оператор ** використовується для піднесення до степеня.

Інфіксний оператор @ — доданий у Python 3.5, використовується переважно у бібліотеках для множення матриць, наприклад, в NumPy.

Оператор присвоєння "Walrus" (:=) введений у Python 3.8, дозволяє присвоювати значення змінним в межах виразу, полегшуючи написання деяких конструкцій коду.

Оператори порівняння Python використовує == для порівняння значень та is для перевірки ідентичності об'єктів. Також дозволяє ланцюгове порівняння, наприклад, a <= b <= c.

Логічні оператори у Python використовуються ключові слова "and", "or" та "not", замість символів &&, ||, та !, характерних для C та Java.

Спискові та генераторні вирази Python підтримує генерацію списків та інших колекцій за допомогою компактного синтаксису.

Лямбда-вирази використовуються для створення анонімних функцій, але обмежені одним виразом.

Умовні вирази у Python вони використовують синтаксис x if c else y, відрізняючись порядком операндів від умовних виразів інших мов.

Списки та кортежі Python розрізняє між змінними списками ([1, 2, 3]) та незмінними кортежами ((1, 2, 3)). Оператор + використовується для об'єднання кортежів.

Розпакування послідовності Python дозволяє розпакувати послідовності в кілька змінних за допомогою структури, подібної до літералу кортежу.

Оператори форматування рядків Python підтримує % для форматування рядків, метод format() та "f-рядки" у Python 3.6, які дозволяють вставляти вирази безпосередньо в рядок.

Ці та інші особливості Python роблять його виразним і гнучким інструментом для програмування.

Python пропонує різноманітні можливості для роботи з рядками, включаючи конкатенацію, різні типи літералів рядків та індексацію.

Особливості обробки рядків та виразів у Python

— конкатенація рядків;

— одинарні та подвійні лапки;

— багаторядкові рядки;

— сирі рядкові літерали;

— індексація та зрізи;

— відмінності між виразами та виразами.

Конкатенація рядків використовується для з'єднання рядків. Застосовується оператор +. Наприклад, "спам" + "яйце" дасть "спамяйце". Це включає і обробку числових значень у рядках, наприклад, "2" + "2" рівнозначно "22".

Одинарні та подвійні лапки. Рядки можуть бути обмежені як одинарними ('), так і подвійними (") лапками. В обох випадках вони працюють однаково, дозволяючи використовувати зворотну косу риску \ для вставки спеціальних символів.

Багаторядкові рядки (String Literals) обмежені потрійними одинарними (''') або подвійними (""") лапками, можуть охоплювати кілька рядків і часто використовуються для документування коду.

Сирі (Raw) рядкові літерали позначаються префіксом r і не обробляють послідовності зворотнього слеша, корисні для використання у регулярних виразах та шляхах файлової системи.

Індексація та зрізи Python підтримує у рядках та списках. Індексація рахується з нуля, а негативні індекси відраховуються від кінця. Зрізи використовують синтаксис [початок:кінець:крок], де початок та кінець визначають діапазон, а крок - інтервал між елементами у зрізі.

Важливою особливістю Python є його чітке розмежування між виразами (expressions) та виразами (statements), що відрізняється від підходів у таких мовах, як Common Lisp, Scheme або Ruby. Це може призвести до певного дублювання функцій, але також робить мову більш структурованою і передбачуваною.

Ці особливості роблять Python гнучкою мовою для роботи з рядками та обробки даних, дозволяючи легко виконувати складні операції з текстом.

Python є динамічною, об'єктно-орієнтованою мовою програмування з різними унікальними особливостями та підходами до типізації, класів та методів.

Ось огляд деяких ключових аспектів Python:

— методи об’єктів;

— типізація і принцип качки;

— класи та об’єкти;

— старі та нові стилі класів;

— статична типізація;

— реалізація CPython;

— розробка Python.

У Python методи об'єкта є функціями, приєднаними до класів. Методи включають явний параметр self для доступу до екземплярів класу. Синтаксис instance.method(argument) є скороченням для Class.method(instance, argument).

Типізація і принцип качки. Python використовує динамічну типізацію та принцип качки (якщо це виглядає як качка і крякає як качка, це, ймовірно, качка). Типи об'єктів визначаються під час виконання, і імена змінних не пов'язані з жодним конкретним типом.

Класи та об'єкти Python дозволяє програмістам створювати свої власні класи. Класи використовуються для створення нових об'єктів, які є екземплярами цих класів. Класи в Python можуть бути метакласами, що дозволяє виконувати метапрограмування.

Python 2 існували старі та нові стилі класів. У Python 3 всі класи є "новими стилями" і неявно успадковують від object.

Починаючи з Python 3.5, синтаксис мови дозволяє вказувати статичні типи, але вони не перевіряються в реалізації CPython за замовчуванням. Існує експериментальний інструмент mypy для статичної перевірки типів.

Реалізація CPython. Офіційна реалізація Python, CPython, інтерпретує мову на платформі C і використовує стандарт C89 з деякими елементами C99. CPython компілює Python-код у байт-код для виконання на віртуальній машині.

Розробка Python ведеться через Програми Покращення Python (PEP), які використовуються для внесення нових функцій та узгодження розробок. Стиль кодування в Python описаний у PEP 8.

Ці особливості роблять Python гнучкою та могутньою мовою, яка підтримує як об'єктно-орієнтоване, так і функціональне програмування, а також дозволяє ефективне використання для широкого спектру застосувань.

PyCharm — це інтегроване середовище розробки (IDE) для мов програмування Python, що надає широкі можливості для зручного написання, відлагодження та тестування коду. Розроблений компанією JetBrains, PyCharm володіє високою функціональністю та продуктивністю, що робить його популярним серед програмістів, що працюють з Python.

Однією з ключових особливостей PyCharm є його розширена система підказок, що допомагає розробникам швидко знаходити помилки та оптимізувати код. Вбудований редактор коду має велику кількість функцій, таких як автодоповнення, підсвічування синтаксису та автоматичне вирівнювання коду, що полегшує написання програм.

Крім того, PyCharm включає в себе потужний візуальний відлагоджувач, що дозволяє крок за кроком відстежувати виконання програми та аналізувати її стан у реальному часі. Це робить відлагодження програм більш ефективним та продуктивним.

PyCharm підтримує використання віртуальних середовищ (virtual environments) та керування залежностями проекту за допомогою інструмента pipenv. Це дозволяє створювати ізольовані середовища для кожного проекту та легко керувати його залежностями.

Узагальнюючи, PyCharm — це потужний інструмент для розробки програм на Python, що надає широкі можливості для швидкого та ефективного написання коду, відлагодження та тестування програм.

Flask — це легкий та елегантний мікрофреймворк для веб-розробки на мові програмування Python. Flask дозволяє швидко створювати веб-додатки та веб-сервіси з мінімальними зусиллями завдяки своїй простоті та гнучкості.

Однією з основних особливостей Flask є його мінімалістичний підхід до розробки. Він має маленький кодову базу та мінімальну кількість залежностей, що робить його ідеальним вибором для швидкого створення простих веб-додатків або прототипів.

Flask також надає широкі можливості для розширення за допомогою різноманітних розширень (extensions), що дозволяють розширювати функціонал фреймворку в залежності від потреб проекту. Наприклад, розширення Flask-SQLAlchemy дозволяє зручно працювати з базами даних SQLAlchemy, а Flask-WTF - з формами WTForms.

Ще однією важливою особливістю Flask є його вбудована підтримка веб-шаблонізатора Jinja2, що дозволяє створювати динамічний контент для веб-сторінок. Jinja2 надає можливості для використання умов, циклів та інших конструкцій для генерації HTML-коду з даними сервера.

Фреймворк також має вбудовану підтримку для роботи з HTTP-запитами та відповідями, що дозволяє зручно обробляти різноманітні запити від клієнтів та генерувати відповіді.

Узагальнюючи, Flask — це простий у використанні та потужний фреймворк для швидкої розробки веб-додатків на Python, який надає широкі можливості для створення різноманітних веб-проектів.

Kivy — це відкрите програмне забезпечення для швидкої розробки мультимедійних додатків, основаних на Python. Основною особливістю Kivy є те, що він дозволяє створювати крос-платформенні додатки, які можуть працювати на різних операційних системах, таких як Windows, macOS, Linux, Android та iOS.

Один з ключових принципів Kivy — це використання мульті-тач вводу, що дозволяє створювати сенсорно-сумісні додатки для пристроїв з сенсорними екранами. Це робить фреймворк ідеальним для розробки мультимедійних додатків для планшетів, смартфонів та інших пристроїв.

Kivy також має велику кількість вбудованих інструментів та модулів, що полегшують розробку різних типів додатків. Наприклад, він має модуль для роботи з графікою, що дозволяє створювати анімації та інші візуальні ефекти, а також модуль для роботи з звуком, що дозволяє додавати аудіо ефекти та музику до додатків.

Фреймворк також надає можливості для роботи зі стандартними елементами інтерфейсу користувача, такими як кнопки, текстові поля та списки, а також для створення власних віджетів за допомогою мови програмування Python.

Однією з переваг Kivy є його відкритість та активна спільнота розробників, що постійно вносять у фреймворк покращення та нові функції.

Узагальнюючи, Kivy — це потужний та універсальний фреймворк для розробки мультимедійних додатків на Python, який дозволяє створювати крос-платформенні додатки для різних операційних систем та пристроїв.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ТЕКСТУ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

3.1 Розробка графічного інтерфейсу

Графічний інтерфейс користувача, або GUI, є особливим типом інтерфейсу, який дозволяє користувачам взаємодіяти з електронними пристроями за допомогою графічних елементів, таких як кнопки, піктограми та аудіоіндикатори, замість введення текстових команд через клавіатуру. GUI використовується в багатьох сучасних пристроях і програмах, і він важливий для полегшення використання і поліпшення ефективності.

Основні характеристики GUI включають:

— графічні елементи;

— безпосередня взаємодія;

— розширені можливості;

— широке використання;

— адаптивність до потреб користувача;

— візуальна композиція.

GUI використовує графічні елементи, такі як кнопки, вікна, меню, полотно та інші, які користувач може взаємодіяти безпосередньо за допомогою миші або сенсорного екрана.

Безпосередня взаємодія — користувачі можуть взаємодіяти з графічними елементами без введення текстових команд. Вони можуть натискати кнопки, перетягувати об'єкти та виконувати інші дії, використовуючи лише мишу або сенсорний екран.

Розширені можливості — GUI дозволяє створювати складні інтерфейси з різними графічними елементами, що полегшує взаємодію користувача з програмою або пристроєм.

Широке використання — GUI використовується не лише на комп'ютерах, але й на різних інших пристроях, включаючи смартфони, планшети, ігрові консолі, побутові та промислові пристрої, автомобільні системи та інше.

Адаптивність до потреб користувача — дизайн GUI орієнтований на користувача і спрямований на полегшення використання для користувачів, поліпшення ефективності та дотримання принципів юзабіліті.

Розробка візуальної композиції та тимчасової поведінки GUI є важливою частиною програмування прикладних програм, що спрямовані на взаємодію людини з комп'ютером.

Усі ці характеристики роблять GUI важливим інструментом для створення дружніх до користувача інтерфейсів для різних програм і пристроїв. Розглянемо переваги та недоліки графічного інтерфейсу користувача.

Переваги GUI:

— інтуєтивність;

— зручність;

— візуалізація;

— мультимедіа.

GUI використовує візуальні елементи, такі як іконки, кнопки та меню, що робить його доступнішим та легшим для розуміння користувачами з різним рівнем технічних знань. Візуальне представлення допомагає користувачам швидко зрозуміти доступні функції та опції.

Графічний інтерфейс користувача забезпечує комфортну взаємодію з комп'ютером, дозволяючи виконувати дії простим клацанням миші, перетягуванням об'єктів та використанням клавіатури для швидкого введення даних.

Візуалізація — GUI надає можливість відображення інформації через графічні елементи, що сприяє ефективнішому сприйняттю та аналізу даних.

Графіки, діаграми, таблиці та інші візуальні елементи допомагають швидше і краще зрозуміти складну інформацію.

Мультимедіа — GUI підтримує інтеграцію мультимедійних елементів, таких як зображення, відео та звук, що дозволяє створювати більш привабливі та інтерактивні інтерфейси, підвищуючи загальне враження користувачів.

Інтерфейс користувача складається з одного вікна (рис. 3.1).

Рисунок 3.1 — Інтерфейс клієнтського додатку

3.2 Розробка логіки клієнтського додатку

При відкритті клієнтського додатку користувач отримує доступ до головного вікна програми.

На головному вікні розташоване зображення з камери і кнопка розпізнавання. Елемент камери влаштований таким чином, щоб відразу виводити зображення, без додаткових налаштувань реалізація наведена на лістингу 3.1.

Лістинг 3.1 — Реалізація отримання зображення

```
Builder.load_string('''
<CameraClick>:
    orientation: 'vertical'
    Camera:
        id: camera
        resolution: (640, 480)
        play: True
    Button:
        text: 'Розпізнавання'
        size_hint_y: None
        height: '48dp'
        on_press: root.capture()
''')
```

При натисканні на кнопку розпізнавання система зберігає зображення з камери і викликає метод комунікації з API, після чого, відтворює звукову доріжку, яку створив попередній метод. Реалізація наведена на лістингу 3.2.

Лістинг 3.2 — Реалізація комунікації з API та відтворення звукової доріжки

```
class CameraClick(BoxLayout):
```

```
def capture(self):
    camera = self.ids['camera']
    timestr = time.strftime("%Y%m%d_%H%M%S")
    name = "IMG_{}.jpg".format(timestr)
    camera.export_to_png(name)
    print(name)
    main.OCR(str(name))
    sound = SoundLoader.load('test.wav')
    if sound:
        sound.play()
    print("Captured")
```

Метод взаємодії з API перетворю отримане зображення в масив байтів, потім в рядок, а потім в JSON, який в свою чергу прикріпляється до http-запиту, який відправляється на сервер, як наведено на лістингу 3.3.

Лістинг 3.3 — Обробка отриманого зображення

```
url = 'https://joreikarr.pythonanywhere.com/process_image'
  with open(image_path, 'rb') as file:
    image_data = file.read()
  image_base64 = base64.b64encode(image_data).decode('utf-8')
  data = {'image': image_base64}
  response = requests.post(url, json=data)
  result = response.json()
  print(result['result'])
```

Після отримання результату метод формує і зберігає звукову доріжку з отриманого тексту, як наведено на лістингу 3.4.

Лістинг 3.4 — Збереження звукової дорожки

```
    tts = TTS(device="cpu")
    with open("test.wav", mode="wb") as file:
        _, output_text = tts.tts(result['result'], Voices.Dmytro.value, Stress.Dictionary.value, file)
    print("Accented text:", output_text)
    ipd.Audio(filename="test.wav")
```

Всі файли, які зберігаються під час роботи програми є взаємозамінними і не зберігаються у великій кількості в пам’яті пристрою.

3.3 Розробка логіки серверної частини

Серверна частина розроблена з використанням фреймворку Flask та OCR бібліотеки easyocr.

Серверна частина має вітальну сторінку, яка призначена виключно для мануальної перевірки роботи серверу.

```
@app.route('/')
def index():
    return 'hello'
```

Основний функціонал полягає в отриманні запиту за посиланням /process_image, в ході обробки якого система ініціалізує бібліотеку розпізнавання тексту, отримує від неї відповідь і формує відповідь на запит, з врахуванням механізму обробки помилок. Реалізацію передбачення помилок наведено в лістингу 3.5.

Лістинг 3.5 — Реалізація розпізнавання тексту та обробки помилок

```
    with contextlib.redirect_stdout(None):
```

```
try:
    reader = easyocr.Reader(['uk','ru','en'])
    data = request.get_json()
    image_data = base64.b64decode(data['image'])
    image = Image.open(io.BytesIO(image_data))
    result = reader.readtext(image, detail=0, paragraph=True)
    fnl_res = ''.join(result)
    return jsonify({'result': fnl_res})

except Exception as ex:
    return jsonify({'result': traceback.format_exc()})
```

3.4 Тестування системи

Тестування програмного забезпечення є критично важливим етапом у процесі розробки будь-якого програмного продукту. Цей процес включає в себе виконання програми або додатку з метою знаходження помилок (багів) та переконання у правильності їх функціонування у різних умовах. Тестування є невід'ємною частиною розробки, оскільки дозволяє гарантувати якість та надійність програмного продукту, підвищує довіру користувачів, знижує ризики використання та сприяє ефективному впровадженню програми.

Важливість тестування:

— забезпечення якості та надійності;

— удосконалення користувацького досвіду;

— зниження вартості розробки;

— забезпечення відповідності стандартам та вимогам.

Забезпечення якості та надійності — забезпечити, що програма працює без помилок та відповідає всім заявленим вимогам та специфікаціям. Це особливо

важливо у випадку систем критичного застосування, де помилки можуть призвести до значних фінансових втрат або навіть загрози життю.

Удосконалення користувацького досвіду — помилки у програмному забезпеченні можуть спричинити погіршення користувацького досвіду, втрату даних або навіть злом системи. Тестування дозволяє виявити та усунути ці проблеми до того, як програма потрапить до кінцевого користувача.

Зниження вартості розробки — виявлення та виправлення помилок на ранніх етапах розробки є значно дешевшим, ніж їх усунення після запуску продукту. Затримка з виявленням помилок може призвести до додаткових витрат на їх виправлення та потенційно до втрати репутації.

Забезпечення відповідності стандартам та вимогам — тестування допомагає переконатися, що програмне забезпечення відповідає всім встановленим стандартам якості та безпеки, а також специфічним вимогам замовника або ринку.

Типи тестування

Тестування може бути різноманітним, включаючи:

— функціональне тестування;

— тестування навантаження;

— тестування безпеки;

— автоматизоване тестування.

Функціональне тестування: перевіряє, чи програма виконує всі свої заявлені функції. Це включає перевірку окремих модулів програми, таких як алгоритми аналізу URL-адрес та вмісту веб-сторінок, а також інтеграцію цих модулів у загальну систему.

Тестування навантаження: оцінює, як програма працює під значним навантаженням. Метою є визначення меж продуктивності системи та її здатності обробляти великий обсяг запитів одночасно. Це включає створення тестових

сценаріїв, які імітують поведінку великої кількості користувачів, що одночасно використовують програму.

Тестування безпеки: виявляє вразливості в програмі, які можуть бути використані для несанкціонованого доступу або витоку даних. Це включає проведення різних видів тестів, таких як аналіз коду на наявність вразливостей, тестування на проникнення, перевірка на відповідність стандартам безпеки та інші.

Автоматизоване тестування: використовує спеціальне програмне забезпечення для автоматичного виконання тестяів. Це дозволяє прискорити процес тестування та зменшити ймовірність людських помилок. Автоматизовані тести можуть бути налаштовані для виконання, що забезпечує своєчасне виявлення помилок. Такі тести можуть охоплювати різні аспекти програми, включаючи функціональне тестування, тестування навантаження та тестування безпеки. Завдяки автоматизації можна швидко перевіряти великий обсяг різноманітних тестів, що підвищує загальну ефективність та якість процесу тестування.

Тестування у розробці програми для виявлення фішингових атак

У контексті розробленої програми для виявлення фішингових атак, тестування включає перевірку алгоритмів аналізу URL-адрес та вмісту веб-сторінок, а також забезпечення правильної роботи графічного інтерфейсу користувача. Особлива увага приділяється тестуванню безпеки, щоб переконатися, що програма не має вразливостей, які можуть бути використані зловмисниками.

Таким чином, тестування є ключовим елементом процесу розробки, який дозволяє не тільки виявити та усунути помилки, але й забезпечити високу якість та безпеку програмного продукту.

В таблиці 3.1 представлено тест-кейси, проведені в ході тестування системи.

Таблиця 3.1— Тестування системи

№	Назва	Очікуваний результат	Отриманий результат
1	Відкриття додатку	Додаток відкривається, камера починає транслювати зображення на головному вікні.	Додаток відкривається, камера починає транслювати зображення на головному вікні.
2	Натискання кнопки «Розпізнавання»	Зображення камери завмирає, спостерігається нормальна затримка до 5-7 секунд, після чого починається відтворення звукової доріжки, в якій промовляється розпізнаний текст.	Зображення камери завмирає, спостерігається нормальна затримка до 5-7 секунд, після чого починається відтворення звукової доріжки, в якій промовляється розпізнаний текст.
3	Закінчення розпізнавання	Закінчується відтворення звукової доріжки, поновлюється відображення зображення з камери, система готова до нового розпізнавання	Закінчується відтворення звукової доріжки, поновлюється відображення зображення з камери, система готова до нового розпізнавання

Виходячи з отриманих результатів тестування, система не має функціональних неточностей і виконує усі закладені в неї функції.

ВИСНОВКИ

У рамках даної бакалаврської дипломної роботи було розроблено програмний додаток для розпізнавання тексту з фотографій та його озвучення на клієнтському пристрої. Додаток був створений з метою полегшення доступу до інформації для людей з обмеженими можливостями, зокрема для незрячих та людей з порушеннями зору.

У розділі 1 були розглянуті теоретичні аспекти та сучасний стан предметної області, зокрема машинне навчання, розпізнавання образів, оптичне розпізнавання символів, огляд існуючих рішень та постановка задачі.

У розділі 2 було проведено проектування програмного продукту, включаючи архітектуру і життєвий цикл, проектування функціоналу додатку та внутрішньої будови.

У розділі 3 була описана розробка програмного продукту, включаючи вибір інструментарію розробки, розробку графічного інтерфейсу, логіку клієнтського та серверного додатків, а також тестування системи.

Отже, розроблений додаток дозволяє незрячим та людям з порушеннями зору отримувати інформацію з фотографій швидко та зручно, що сприяє їхній більшій самостійності та інтеграції в суспільство. Результати роботи можуть бути використані для подальшого вдосконалення програмних продуктів у галузі доступності до інформації для людей з обмеженими можливостями.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Порівняльний аналіз бібліотек для обробки зображень та комп’ютерного зору/ К. Я. Поташна, О. С. Городецька // Матеріали LIII науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії. Вінниця 2024 р. 3 с. [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2024/paper/view/20949/17355

2. OnDemand, HPE Haven. "OCR Document". Archived from the original on April 15, 2016.

3. Підручник з Python [Електронний ресурс] — Режим доступу: https://docs.python.org/uk/3/tutorial/

4. OnDemand, HPE Haven. "undefined". Archived from the original on April 19, 2016.

5. Schantz, Herbert F. (1982). The history of OCR, optical character recognition. [Manchester Center, Vt.]: Recognition Technologies Users Association. ISBN 9780943072012.

6. Dhavale, Sunita Vikrant (2017). Advanced Image-Based Spam Detection and Filtering Techniques. Hershey, PA: IGI Global. p. 91. ISBN 9781683180142.

7. d'Albe, E. E. F. (July 1, 1914). "On a Type-Reading Optophone". Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences. 90 (619): 373–375. Bibcode:1914RSPSA..90..373D. doi:10.1098/rspa.1914.0061.

8. "The History of OCR". Data Processing Magazine. 12: 46. 1970.

9. "Extracting text from images using OCR on Android". June 27, 2015. Archived from the original on March 15, 2016.

10. "[Tutorial] OCR on Google Glass". October 23, 2014. Archived from the original on March 5, 2016.

11. Zeng, Qing-An (2015). Wireless Communications, Networking and Applications: Proceedings of WCNA 2014. Springer. ISBN 978-81-322-2580-5.

12. "[javascript] Using OCR and Entity Extraction for LinkedIn Company Lookup". July 22, 2014. Archived from the original on April 17, 2016.

13. "How To Crack Captchas". andrewt.net. June 28, 2006. Retrieved June 16, 2013.

14. "Breaking a Visual CAPTCHA". Cs.sfu.ca. December 10, 2002. Retrieved June 16, 2013.

15. Resig, John (January 23, 2009). "John Resig – OCR and Neural Nets in JavaScript". Ejohn.org. Retrieved June 16, 2013.

16. Tappert, C. C.; Suen, C. Y.; Wakahara, T. (1990). "The state of the art in online handwriting recognition". IEEE Transactions on Pattern Analysis and Machine Intelligence. 12 (8): 787. doi:10.1109/34.57669. S2CID 42920826.

17. "Optical Character Recognition (OCR) – How it works". Nicomsoft.com. Retrieved June 16, 2013.

18. Sezgin, Mehmet; Sankur, Bulent (2004). "Survey over image thresholding techniques and quantitative performance evaluation" (PDF). Journal of Electronic Imaging. 13 (1): 146. Bibcode:2004JEI....13..146S. doi:10.1117/1.1631315. Archived from the original (PDF) on October 16, 2015. Retrieved May 2, 2015.

19. Gupta, Maya R.; Jacobson, Nathaniel P.; Garcia, Eric K. (2007). "OCR binarisation and image pre-processing for searching historical documents" (PDF). Pattern Recognition. 40 (2): 389. Bibcode:2007PatRe..40..389G. doi:10.1016/j.patcog.2006.04.043. Archived from the original (PDF) on October 16, 2015. Retrieved May 2, 2015.

20. Trier, Oeivind Due; Jain, Anil K. (1995). "Goal-directed evaluation of binarisation methods" (PDF). IEEE Transactions on Pattern Analysis and Machine

Intelligence. 17 (12): 1191–1201. doi:10.1109/34.476511. Archived (PDF) from the original on October 16, 2015. Retrieved May 2, 2015.

21. Milyaev, Sergey; Barinova, Olga; Novikova, Tatiana; Kohli, Pushmeet; Lempitsky, Victor (2013). "Image Binarization for End-to-End Text Understanding in Natural Images". 2013 12th International Conference on Document Analysis and Recognition (PDF). pp. 128–132. doi:10.1109/ICDAR.2013.33. ISBN 978-0-7695-4999-6. S2CID 8947361. Archived (PDF) from the original on November 13, 2017. Retrieved May 2, 2015.

22. Pati, P.B.; Ramakrishnan, A.G. (May 29, 1987). "Word Level Multi-script Identification". Pattern Recognition Letters. 29 (9): 1218–1229. Bibcode:2008PaReL..29.1218P. doi:10.1016/j.patrec.2008.01.027.

23. "Basic OCR in OpenCV | Damiles". Blog.damiles.com. November 20, 2008. Retrieved June 16, 2013.

24. Smith, Ray (2007). "An Overview of the Tesseract OCR Engine" (PDF). Archived from the original (PDF) on September 28, 2010. Retrieved May 23, 2013.

25. "OCR Introduction". Dataid.com. Retrieved June 16, 2013.

26. "How OCR Software Works". OCRWizard. Archived from the original on August 16, 2009. Retrieved June 16, 2013.

27. "The basic pattern recognition and classification with openCV | Damiles". Blog.damiles.com. November 14, 2008. Retrieved June 16, 2013.

28. Assefi, Mehdi (December 2016). "OCR as a Service: An Experimental Evaluation of Google Docs OCR, Tesseract, ABBYY FineReader, and Transym". ResearchGate.

29. "How the Best OCR Technology Captures 99.91% of Data". www.bisok.com. Retrieved May 27, 2021.

30. Woodford, Chris (January 30, 2012). "How does OCR document scanning work?". Explain that Stuff. Retrieved June 16, 2013.

31. "How to optimize results from the OCR API when extracting text from an image? - Haven OnDemand Developer Community". Archived from the original on March 22, 2016.

32. "What is the point of an online interactive OCR text editor? - Fenno-Ugrica". February 21, 2014.

33. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 123 «Комп’ютерна інженерія» (освітня програма «Комп’ютерна інженерія») [Електронний ресурс] / уклад. О. Д. Азаров, С. В. Богомолов, С. І. Швець. — Вінниця : ВНТУ, 2023. — 105 с.

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н. О. Д. Азаров

« 06 » травня 2024 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання дипломної роботи

«Інформаційна система розпізнання тексту на основі нейронної мережі»

08-54.БДР.012.00.000 ПЗ

Науковий керівник: к.т.н., доц. каф. ОТ

_____Городецька О.С.

«_____» _____2024 р.

Студентка групи 1СП-20б

_____ Поташна К.Я.

ВНТУ 2024

1 Підстава для виконання бакалаврської дипломної роботи

1.1 Покращення та полегшення отримання інформації для людей з обмеженими можливостями стає де далі актуальнішою темою, ніж це було пару років тому. Оскільки доступ до інформації з фотографій є важким завданням для людей незрячих або з частковою втратою зору важливим є створення інформаційних систем, які допомагаю вирішити ряд питань. Саме за допомогою нейронних мереж та штучного інтелекту створення таки систем стало можливим. Сучасні технології дозволяють розпізнавати та озвучувати текст, що допомагають людям у взаємодії з суспільством, навчанням та іншими необхідними їм завданнями.

2 Мета і призначення БДР

2.1 Мета роботи – реалізувати додаток для розпізнавання тексту з фотографій та його озвучення, щоб спрощувати отримання необхідної інформації.

2.2 Призначення розробки – забезпечити доступність інформації для людей з обмеженими можливостями, зокрема для незрячих та людей з порушеннями зору. За допомогою цього додатку користувачі зможуть самостійно отримувати інформацію з різних джерел, таких як документи, вивіски, меню, книги та інші текстові матеріали. Це сприятиме їхній більшій незалежності у повсякденному житті.

3 Вихідні дані для виконання БДР

3.1 Аналіз існуючих програмних рішень;

3.2 Проведення детального аналізу вихідних даних ;

3.3 Особливості машинного навчання згорткових нейронних мереж (CNN)

та рекурентних нейронних мереж (RNN);

3.4 Продуктивність та тестування інформаційної системи.

4 Вимоги до виконання БДР

Головною та найнеобхіднішою вимогою до виконання роботи є забезпечення зручного та зрозумілого графічного інтерфейсу клієнтської частини додатку, створення додатку для розпізнавання тексту на основі нейронної мережі з високою точністю розпізнавання, підтримку різних типів зображень, швидку обробку зображень, підтримку функції озвучення розпізнаного тексту та оптимізацію для роботи на різних платформах і пристроях.

5 Етапи БДР та очікувані результати в таблиці А.1

Таблиця А.1 — Етапи БДР

№ з/п	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз індивідуального завдання. Вступ	13.03.24	17.03.24	Вступ
2	Написання технічного завдання	19.03.24	21.03.24	Технічне завдання
3	Огляд існуючих програмних рішень та аналіз літературних джерел	23.03.24	29.03.24	Розділ 1, тези доповідей
4	Вибір технології для клієнтської та серверної частини	02.04.24	09.04.24	Розділ 2
5	Розробка клієнтської та серверної частини	11.04.24	26.04.24	Розділ 3
6	Оформлення пояснювальної записки та презентаці	29.04.24	10.05.24	ПЗ, графічний матеріал, презентація

6 Матеріали, що подаються до захисту БДР

До захисту подаються: пояснювальна записка БДР, ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відгук наукового керівника,

анотації до БДР українською та іноземною мовами, довідка про відповідність оформлення БДР діючим вимогам.

7 Порядок контролю виконання та захисту БДР

Виконання етапів графічної та розрахункової документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлення та порядок виконання БДР

При оформлюванні БДР використовуються:

— Методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія». Кафедра обчислювальної техніки ВНТУ 2022;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— документами на які посилаються у вище вказаних.

ДОДАТОК Б

Схема взаємодії користувача з додатком

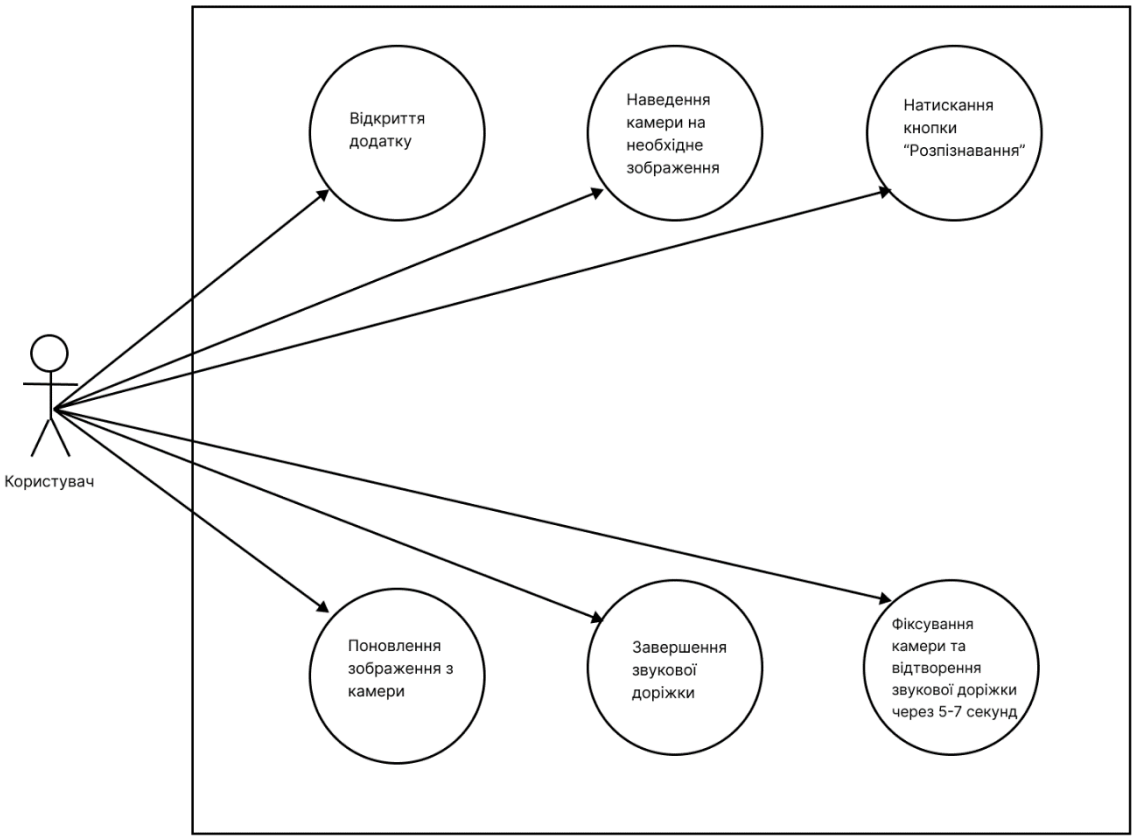

Рисунок Б.1 — Блок-схема взаємодії користувача з додатком

ДОДАТОК В

Схема обробки запиту на розпізнавання тексту

Рисунок В.1 — Блок-схема опрацювання запиту на обробку тексту

ДОДАТОК Г

Схема взаємодії клієнтської та серверної частини

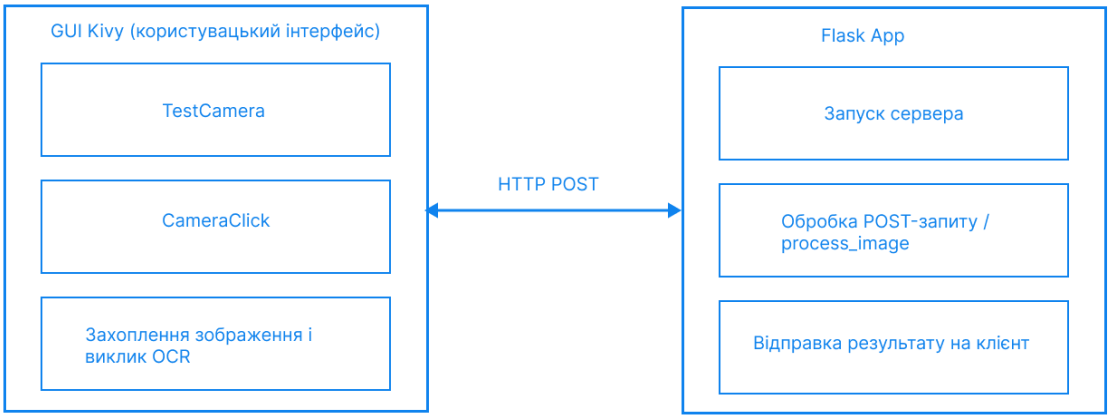

Рисунок Г.1 — Схема взаємозв’язку клієнтської та серверної частини

ДОДАТОК Д

Лістинг програмного коду

```
import base64
from ukrainian_tts.tts import TTS, Voices, Stress
import IPython.display as ipd
import requests
import json

def OCR(image_path):
    url = 'https://joreikarr.pythonanywhere.com/process_image'

    with open(image_path, 'rb') as file:
        image_data = file.read()
    image_base64 = base64.b64encode(image_data).decode('utf-8')
    data = {'image': image_base64}
    response = requests.post(url, json=data)
    result = response.json()
    print(result['result'])
    tts = TTS(device="cpu")
    with open("test.wav", mode="wb") as file:
        _, output_text = tts.tts(result['result'], Voices.Dmytro.value,
Stress.Dictionary.value, file)
    print("Accented text:", output_text)
    ipd.Audio(filename="test.wav")
import os
```

```
from kivy.app import App
from kivy.lang import Builder
from kivy.uix.boxlayout import BoxLayout
from kivy.core.audio import SoundLoader
import time

import main
Builder.load_string('''
<CameraClick>:
    orientation: 'vertical'
    Camera:
        id: camera
        resolution: (640, 480)
        play: True
    Button:
        text: 'Розпізнавання'
        size_hint_y: None
        height: '48dp'
        on_press: root.capture()
''')

class CameraClick(BoxLayout):
    def capture(self):
        camera = self.ids['camera']
        timestr = time.strftime("%Y%m%d_%H%M%S")
        name = "IMG_{}.jpg".format(timestr)
        camera.export_to_png(name)
```

```
        print(name)
        main.OCR(str(name))
        sound = SoundLoader.load('test.wav')
        if sound:
            sound.play()
        print("Captured")
class TestCamera(App):
    def build(self):
        return CameraClick()
TestCamera().run()
from flask import Flask, request, jsonify
from PIL import Image
import io
import easyocr
import contextlib
import traceback
import base64
app = Flask(__name__)

@app.route('/')
def index():
    return 'hello'

@app.route('/process_image', methods=['POST'])
def process_image():
    with contextlib.redirect_stdout(None):
        try:
```

```
            reader = easyocr.Reader(['uk','ru','en'])
            data = request.get_json()
            image_data = base64.b64decode(data['image'])
            image = Image.open(io.BytesIO(image_data))
            result = reader.readtext(image, detail=0, paragraph=True)
            fnl_res = ''.join(result)
            return jsonify({'result': fnl_res})
        except Exception as ex:
            return jsonify({'result': traceback.format_exc()})
if __name__ == '__main__':
    app.run(debug=True)
```

ДОДАТОК Е

ПРОТОКОЛ
ПЕРЕВІРКИ БАКАЛАВРСЬКОЇ ДИПЛОМНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна система розпізнавання тексту на основі нейронної мережі

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 81% Схожість 19%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Поташна К.Я
(підпис) (прізвище, ініціали)

Керівник роботи _____ Городецька О.С
(підпис) (прізвище, ініціали)