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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

АДАПТОВАНІ НЕЙРОМЕРЕЖЕВІ ТЕХНОЛОГІЇ ДЛЯ ЛЮДЕЙ З ОБМЕЖЕНИМИ МОЖЛИВОСТЯМИ

Виконала: магістрант групи 1КІ-24м
спеціальності 123 — Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
Поташна К.Я.
(прізвище та ініціали)
Керівник: к.т.н., доц., доцент кафедри ОТ Колесник І.С.
(прізвище та ініціали)
«15» 12 2025р.
Опонент
к.т.н., доц., завідувач кафедри МБІС: Карпінець В.В.
(прізвище та ініціали)
«15» 12 2025 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«18» 12 2025 р.

ВНТУ 2025

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень Магістр

Галузь знань — 12 «Інформаційні технології»

Спеціальність — 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О. Д.
25.09.2025 року

З А В Д А Н Н Я
НА МАГІСТЕСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студентці Поташній Карині Ярославівні

1 Тема роботи: «Адаптовані нейромережеві технології для людей з обмеженими можливостями», керівник роботи Колесник І.С. к.т.н., доцент кафедри ОТ, затверджено наказом вищого навчального закладу від «24» вересня 2025 року № 313.

2 Строк подання студентом роботи 04.12.2025 р.

3 Вихідні дані до роботи: актуальність, аналіз та обґрунтування засобів реалізації адаптованої нейромережевої технології для людей з обмеженими можливостями.

4 Зміст пояснювальної записки: вступ, огляд предметної області, обґрунтування вибору технологій для виконання роботи, розробка клієнтської та серверної частини, тестування, висновок, перелік джерел посилання, додатки.

5 Перелік графічного матеріалу — схема взаємодії користувача з додатком, схема обробки запиту на розпізнавання тексту

6 Консультанти розділів роботи представлені у таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1 – 4	к.т.н., доц. каф. Колесник І.С.		
5	к.т.н., доц., ЕПВМ Ратушняк О.Г.		
Нормо Контроль	асист. каф. ОТ Швець С.І.		

Дата видачі завдання 25.09.2025 р.

Календарний план виконання МКР наведений у таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапу	Термінвиконання		Очікувані результати
		початок	кінець	
1	Аналіз поставленого задачі	26.09.25	30.09.25	Огляд джерел, висновки із взаємодії викладачів та студентів
	Огляд існуючих програмних рішень та аналіз літературних джерел	06.10.25	15.10.25	Розділ 1
4	Вибір технології для клієнтської та серверної частини	16.10.25	31.10.25	Розділ 2
5	Розробка клієнтської та серверної частини	01.11.25	17.11.25	Розділ 3
6	Тестування розробки	18.11.25	20.11.25	Розділ 4
7	Оформлення пояснювальної записки та презентації	21.11.25	01.11.25	ПЗ, графічний матеріал, презентація

Студент _____ Поташна К,Я,

Керівник роботи _____ Колесник.І.С.

АНОТАЦІЯ

УДК 004.932.72:004.852

Поташна К.Я. Адаптовані нейромережеві технології для людей з обмеженими можливостями. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна інженерія, освітня програма — комп’ютерна інженерія. Вінниця: ВНТУ, 2025. 99 с.

На укр. Мові. Бібліогр. 25 назв, рис. 7, табл. 9, ліст.1 .

У магістерській кваліфікаційній роботі розробленю інформаційну систему, яка дозволяє користувачам з обмеженими можливостями легко отримувати інформацію з фотографій, що сприятиме їхній самостійності та інтеграції в суспільство. 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

Potashna K.Ya. Adapted neural network technologies for people with disabilities Master's Qualification Thesis in Specialty 123 – Computer Engineering, Educational Program "Computer Engineering". Vinnytsia: VNTU, 2025. 99 p.

Language: Ukrainian. References: 25 sources, 7 figures, 9 tables, 1 listings.

In the master's qualification work, I developed an information system that allows users with disabilities to easily obtain information from photographs, which will contribute to their independence and integration into society. For people with visual impairments, 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 accessibility of information for the blind and visually impaired using neural networks.

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

Keywords: information system, text recognition, neural networks, accessibility, disabilities.

ЗМІСТ

ВСТУП

Актуальність. Реалії існування сучасного світу вирізняються тим, що володіння відповідною інформацією є суттєво важливим елементом для комфортного існування та функціонування будь-якого індивідуума суспільних відносин. Проте, доступ до інформації, відображеної певним чином, наприклад, фотографій може виявитись проблемою для осіб з певними вадами здоров'я, особливо для незрячих та слабозорих. Через існування цієї проблеми надзвичайно важливим повстає питання створення інформаційних систем, призначених для допомогти у вирішенні цього складного та актуального питання.Такі системи не тільки вирішують конкретну технологічну проблему, але й розширюють існуючі можливості для ширшого кола користувачів, роблячи інформацію доступнішою для людей з інвалідністю. Вони відкривають нові шляхи для навчання, дослідження, розуміння різних завдань та полегшення життя.

Метою цієї роботи є створення інформаційної системи з розширеною функціональністю для людей з інвалідністю, тобто розпізнавання тексту та його озвучування за допомогою нейронних мереж.

Це пов'язано з тим, що, постійний і стімкий розвиток інформаційних технологій, зокрема, розвиток технологій штучного інтелекту відкриває безліч нових можливостей для підтримки комфортнішого життя людей з інвалідністю. Зокрема, нейронні мережі, призначені для розпізнавання тексту, можуть значно підвищити точність і швидкість обробки інформації. Це сприяє створенню систем, здатних ефективно і точно ідентифікувати текст на зображеннях та перетворювати його в розмовний формат. Подібні технології стають особливо корисними для людей із вадами зору, допомагаючи їм отримувати доступ до інформації з друкованих матеріалів, вивісок назв закладів , меню, різноманітних документів та інших джерел інформації, які раніше були для них недосяжними.

Ше однією причиною розробки подібних систем має важливе соціальне значення. У людей з інвалідністю постійно виникають труднощі у

повсякденному житті, які можуть здаватися незначними для інших. Інформаційні системи, які призначені для розпізнавання та подальшого озвучування тексту, допомагають досить суттєво підвищити рівень їх комфортнішого проживання та адаптації до самостійного життя. Цей фактор сприяє їх інтеграції у навколишній світ, дозволяє брати активнішу часть у соціальних сферах життя суспільства, освіті, подальшій професійній діяльності чи інших сферах життя.

Ще один аспект створення, подібних систем, це те, що вони можуть знайти застосування у найрізноманітніших галузях, наразі, освіті, медицині, сфері надання послуг та багатьох інших. Що стосується освітнього середовища, то їх можна використовувати при створенні відкритих для доступу навчальних матеріалів, це важливо при забезпеченні однакових можливостей отримання знань. Що стосується медичної сфери, то у ній подібні системи можуть знадобитися при наданні медичних інструкцій чи інформації пацієнтам із порушеннями зору. Щодо сфери надання послуг, то у ній може використовуватися для підващення якості роботи з клієнтами, які мають інвалідність, для забезпечення їм доступу до інформації, яка стосується певної продукції чи послуги.

Подібні системи стають доступнішими завдяки стрімкому розвитку технологічного прогресу. Досягнення в сфері створення апаратного забезпечення, зниження вартості на використання обчислювальної потужності сприяють створенню ефективніших та дешевших рішень, які можна впроваджувати в повсякденному житті. Це дозволить створювати портативні пристрої або мобільні додатки, якими люди з інвалідністю можуть користуватися будь-коли та будь-де.

Таким чином, створення інформаційних систем для контингенту з обмеженими можливостями, для розпізнавання та подальшого озвучуваня тексту, дійсно, є актуальною задачею із соціальними та технологічними наслідками. Завдяки розвитку новітніх технологій створення подібних систем стає реалістичнішим та доступнішим, що робить цей напрямок все важливішим та важливішим.

Об'єктом дослідження є процес розпізнавання символів тексту на основі використання нейронної мережі.

Предметом дослідження є методи розпізнавання символів тексту із зображень та подальшим його озвученням на клієнтському пристрої.

Завданнями дослідження є:

— аналіз існуючих рішень для розпізнавання символів;

— аналіз теоретичних аспектів та сучасного стану предметної області;

— аналіз та обґрунтування вибору технології;

— створення інформаційної системи розпізнавання тексту на основі нейронних мереж;

— тестування можливостей функціонування інформаційної системи.

Наукова новизна: покращено метод ідентифікації тексту, що відрізняється від відомих методів використанням адаптованої нейромережевої технології для людей з обмеженими можливостями

Апробація результатів магістерської кваліфікаційної роботи: Опубліковані тези у матеріалах МНП Інтернет конференції студентів, аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи (МН-2026)»

2 ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1. Машинне навчання

Машинне навчання (ML) являється галуззю досліджень штучного інтелекту, яка займається розробкою та вивченням статистичних алгоритмів, які мають змогу навчатися на даних та узагальнювати їх за певними ознаками, це дозволяє виконувати завдання без постійних інструкцій користувача [1]. Нещодавно штучні нейронні мережі змогли перевершити багато попередніх методів за продуктивністю [2,3].

Методи машинного навчання знаходять застосування в багатьох сферах, зокрема, в обробці природної мови, комп'ютерному зорі, розпізнаванні мовлення, аналізі електронної пошти, сільському господарстві та медицині. Особливої популярності вони набули при реалізації бізнес-задач, орієнтованих на прогнозну аналітику. Хоча не всі методи машинного навчання мають статистичну природу, обчислювальна статистика залишається ключовим джерелом підходів у цій галузі. Математична база машинного навчання ґрунтується на методах математичної оптимізації. Пов’язана з ним сфера — інтелектуальний аналіз даних — акцентує увагу на дослідницькому аналізі даних, особливо через методи навчання без учителя. Теоретичною основою для опису машинного навчання є модель ймовірнісно коректного навчання (PAC). Машинне навчання як наукова дисципліна виникло з прагнення до розвитку штучного інтелекту (ШІ). На ранніх етапах розвитку цієї галузі дослідники прагнули навчити машини аналізувати інформацію, використовуючи різноманітні моделі. Виникли нейронні мережі, які були перцептонами або подібними моделями й фактично слугували адаптацією лінійних статистичних моделей. Ймовірнісне моделювання також застосовувалося, наприклад, у системах автоматизованого медичного огляду. Однак фокус уваги ШІ тоді змістився на логічний підхід, заснований на знаннях, що створило певний розрив між ШІ та машинним навчанням. Ймовірнісним системам викликали довіру менше через труднощі в збиранні та інтерпретації даних. До 1980-х років в області ШІ домінували експертні системи, тоді як

статистичні методи досягли свого піку. Водночас, тривалі дослідження логічного навчання, призвели до створення індуктивного логічного програмування (ILP). Однак, найбільш перспективні напрями вже переходили до сфер розпізнавання образів і пошуку інформації. Дослідження нейронних мереж у рамках ШІ тимчасово припинилися, але вони продовжували розвиватися дослідниками інших дисциплін під назвою "коннекціонізм". Серед найвідоміших дослідників цього напряму – Хоупфілд, Румельхарт і Хінтон, які досягли значних результатів у середині 1980-х років завдяки вдосконаленню методу зворотного поширення помилки. До 1990-х років машинне навчання виділилось як окрема дисципліна з чітким акцентом на вирішення прикладних проблем замість створення загальної системи штучного інтелекту. Замість символічних методів, успадкованих від ШІ, галузь почала активно використовувати підходи зі статистики, теорії ймовірностей та нечіткої логіки. Це стало основою для сучасного процвітання машинного навчання.

1.2 Нейронні мережі

Нейронні мережі — це математичні моделі та алгоритми машинного навчання, які імітують функціонування біологічних нейронних мереж. Вони формуються.

Велика кількість взаємопов'язаних обчислювальних вузлів або нейронів організована в шари. Сучасні нейронні мережі використовуються в багатьох галузях, включаючи розпізнавання образів, обробку природної мови, прогнозування та багато інших. У цій статті ми розглянемо основні типи нейронних мереж, їхні сильні та слабкі сторони.

Персептрон — одна з найпростіших форм нейронних мереж, запропонована Френком Розенблаттом у 1958 році. Він складається з одного шару асоціативних нейронів, кожен з яких підключений до всіх входів. Персептрон використовує вагові коефіцієнти для обчислення лінійної комбінації вхідних сигналів та застосовує порогову функцію активаії для визначення вихідного сигналу. Персептрон може вирішувати лише лінійно роздільні задачі,

що є його головною слабкістю. Таким чином, для складніших задач потрібні складніші моделі, такі як багатошарові персептрони.

Багатошаровий персептрон (MLP) розширює концепцію персептрона, додаючи один або кілька прихованих шарів між вхідним та вихідним шарами. Це дозволяє моделі вивчати нелінійні залежності між вхідними та вихідними даними. Кожен шар містить нейрони, які використовують нелінійні функції руху, такі як сигмоїдальна або ReLU (випрямлена лінійна одиниця). Багатошаровий персептрон здатний моделювати складні дії та розпізнавати складні закономірності, що робить його універсальним інструментом машинного навчання. Однак навчання MLP може бути повільним та обчислювально ресурсоємним, особливо для глибоких мереж з кількома шарами.

Згорткові нейронні мережі (CNN) спеціально розроблені для обробки даних у сітчастій топології, таких як зображення. Основна ідея CNN полягає у використанні згорткових шарів, які виконують фільтри для автоматичного вилучення важливих ознак з вхідних даних. Ці фільтри дозволяють мережі менш чутливою до спотворень та деформацій на зображеннях. Згорткові мережі стали стандартом для таких завдань, як розпізнавання образів, класифікація зображень та розпізнавання об'єктів. Однак навчання CNN вимагає багато даних та обчислювальних ресурсів, що може бути обмеженням для деяких застосувань.

Рекурентні нейронні мережі (RNN) використовуються для обробки послідовних даних, таких як текст або часові ряди. Особливістю RNN є зворотні зв'язки, які дозволяють інформації проходити через часові кроки. Це робить їх ідеальними для завдань, де важлива послідовність або контекст. Однак традиційні RNN страждають від проблеми втрати оновлень, 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) — це особливий тип RNN, який вирішує проблему втрати оновлень за допомогою комірок пам'яті та механізмів вентилів. LSTM має три основні вентилі: вхід, вихід та забуття, які

контролюють потік інформації через комірки пам'яті. Це дозволяє LSTM запам'ятовувати важливу інформацію протягом тривалого часу та забувати нерелевантні дані. LSTM ефективні для завдань обробки природної мови, таких як машинний переклад, розпізнавання мовлення та аналіз настроїв. Однак їхня складна архітектура призводить до більших обчислювальних витрат та часу навчання порівняно з традиційними RNN. Автоенкодер — це тип нейронної мережі, що використовується для навчання ефективного представлення даних, зазвичай з метою зменшення розмірності або виявлення ознак. Автоенкодер складається з двох частин: кодера, який перетворює вхідні дані на скінченне латентне представлення, та декодера, який реконструює вихідні дані з цього представлення. Основна ідея автоенкодера полягає в тому, щоб навчитися стискати дані з мінімальною втратою інформації. Автоенкодер знаходять застосування в обробці зображень, видаленні шуму та генерації даних. Однак, як і інші нейронні мережі, автоенкодер може страждати від перевантаження зв'язків та вимагати значної кількості даних для ефективного навчання.

Трансформери це сучасна архітектура нейронної мережі, популярна завдяки своїй ефективності в завданнях обробки природної мови. Головним нововведенням трансформаторів є використання механізму самоконтролю, який дозволяє моделі динамічно зважувати важливість різних частин вхідної послідовності під час обчислення вихідного сигналу. Це дозволяє трансформаторам послідовностей працювати паралельно, що значно пришвидшує навчання порівняно з RNN та LSTM. Трансформери виявилися дуже ефективними для перекладу, генерації тексту, відповідей на запитання та інших завдань обробки тексту. Однак вони є обчислювально ресурсоємними та вимагають багато даних для навчання, що може бути обмеженням для деяких застосувань.

Кожен з описаних типів нейронних мереж має свої сильні та слабкі сторони. Персептрон простий та ефективний для задач лінійного розділення шляхів, але його можливості обмежені. Персептрон — одна з найпростіших форм нейронних мереж, запропонована Френком Розенблаттом у 1958 році. Він

складається з одного шару асоціативних нейронів, кожен з яких підключений до всіх входів. Персептрон використовує вагові коефіцієнти для обчислення лінійної комбінації вхідних сигналів та застосовує порогову функцію активаії для визначення вихідного сигналу. Персептрон може вирішувати лише лінійно роздільні задачі, що є його головною слабкістю. Таким чином, для складніших задач потрібні складніші моделі, такі як багатошарові персептрони.

Багатошаровий персептрон (MLP) розширює концепцію персептрона, додаючи один або кілька прихованих шарів між вхідним та вихідним шарами. Це дозволяє моделі вивчати нелінійні залежності між вхідними та вихідними даними. Кожен шар містить нейрони, які використовують нелінійні функції руху, такі як сигмоїдальна або ReLU (випрямлена лінійна одиниця). Багатошаровий персептрон здатний моделювати складні дії та розпізнавати складні закономірності, що робить його універсальним інструментом машинного навчання. Однак навчання MLP може бути повільним та обчислювально ресурсоємним, особливо для глибоких мереж з кількома шарами.

Згорткові нейронні мережі (CNN) спеціально розроблені для обробки даних у сітчастій топології, таких як зображення. Основна ідея CNN полягає у використанні згорткових шарів, які виконують фільтри для автоматичного вилучення важливих ознак з вхідних даних. Ці фільтри дозволяють мережі менш чутливою до спотворень та деформацій на зображеннях. Згорткові мережі стали стандартом для таких завдань, як розпізнавання образів, класифікація зображень та розпізнавання об'єктів. Однак навчання CNN вимагає багато даних та обчислювальних ресурсів, що може бути обмеженням для деяких застосувань.

Рекурентні нейронні мережі (RNN) застосовуються для роботи з послідовними даними, такими як текст чи часові ряди. Головна особливість RNN — наявність зворотних зв'язків, які забезпечують перенесення інформації через часові етапи. Це робить їх ідеальними для завдань, де важливі послідовність або контекст. Проте традиційні RNN мають обмеження, пов'язані з втратою оновлень, що ускладнює їх навчання на довгих послідовностях. Для вирішення цієї проблеми були створені вдосконалені архітектури, зокрема структури типу

LSTM (довготривала короткочасна пам'ять). Довготривала короткочасна пам'ять (LSTM) є різновидом RNN, яка вирішує проблему втрати оновлень шляхом використання спеціальних комірок пам'яті та механізмів вентилів. Основними компонентами LSTM є три вентилі — вхідний, вихідний та вентиль забуття, що регулюють потік інформації через комірки пам'яті. Це дозволяє LSTM зберігати важливу інформацію протягом тривалого часу та відкидати нерелевантні дані. Така архітектура ефективно застосовується у задачах обробки природної мови, наприклад, машинному перекладі, розпізнаванні мовлення та аналізі настроїв. Водночас складність LSTM спричиняє більші обчислювальні витрати та збільшує час навчання порівняно з базовими RNN. Автоенкодер є видом нейронної мережі, що використовується для створення компактного представлення даних, найчастіше задля зменшення розмірності або виділення ключових ознак. Він складається з двох основних частин: кодера, який стискає вхідні дані в компактне латентне представлення, та декодера, що відновлює вихідні дані на основі цього представлення. Основна мета автоенкодера – навчитися стискати дані з мінімальними втратами інформації. Автоенкодери знаходять своє застосування у різних завданнях, таких як обробка зображень, видалення шуму або генерування нових даних. Проте, як і більшість інших нейронних мереж, вони схильні до перенавчання та потребують великих обсягів даних для ефективного функціонування. Трансформери представляють сучасну архітектуру нейромереж, що відзначається високою ефективністю у завданнях обробки природної мови. Їхнім ключовим нововведенням є механізм самоконтролю (self-attention), який дає змогу моделі оцінювати вплив різних частин вхідної послідовності на результат. Завдяки цьому трансформери можуть обробляти послідовності паралельно, що значно пришвидшує процес навчання порівняно з RNN та LSTM. Вони виявилися надзвичайно потужними у перекладі текстів, створенні тексту, відповіді на запити та багатьох інших задачах мовного аналізу. Попри свої переваги, трансформери потребують значних обчислювальних ресурсів і великих обсягів даних для навчання, що може стати обмеженням у певних випадках.

Кожен із перелічених типів нейронних мереж має свої переваги та недоліки. Персептрон є простим у реалізації та ефективним для задач лінійного розділення, але його сферу застосування обмежено. Багатошаровий персептрон здатний моделювати складні нелінійні залежності, проте його навчання потребує значних ресурсів. Згорткові нейронні мережі відмінно підходять для обробки зображень, хоча і потребують великої обчислювальної потужності. Рекурентні нейронні мережі та LSTM ідеальні для роботи з послідовними даними, однак їх навчання може бути досить складним. Автоенкодери добре справляються зі зменшенням розмірності та виявленням характерних ознак, але вразливі до перенавчання. Трансформери демонструють високий рівень ефективності в задачах обробки природної мови, водночас їх тренування потребує значних обчислювальних ресурсів. Попри наявність недоліків, кожен тип нейронної мережі має унікальні переваги, які роблять її незамінною для певних завдань. Успішна робота з нейронними мережами вимагає глибокого розуміння їх функціональних особливостей, а також набуття суттєвого досвіду у процесах налаштування та навчання моделей.

1.3.Види нейронних мереж

Кожен із перелічених типів нейронних мереж має свої переваги та недоліки. Персептрон є простим у реалізації та ефективним для задач лінійного розділення, але його сферу застосування обмежено. Багатошаровий персептрон здатний моделювати складні нелінійні залежності, проте його навчання потребує значних ресурсів. Згорткові нейронні мережі відмінно підходять для обробки зображень, хоча і потребують великої обчислювальної потужності. Рекурентні нейронні мережі та LSTM ідеальні для роботи з послідовними даними, однак їх навчання може бути досить складним. Автоенкодери добре справляються зі зменшенням розмірності та виявленням характерних ознак, але вразливі до перенавчання. Трансформери демонструють високий рівень ефективності в задачах обробки природної мови, водночас їх тренування потребує значних обчислювальних ресурсів. Попри наявність недоліків, кожен тип нейронної

мережі має унікальні переваги, які роблять її незамінною для певних завдань. Успішна робота з нейронними мережами вимагає глибокого розуміння їх функціональних особливостей, а також набуття суттєвого досвіду у процесах налаштування та навчання моделей.

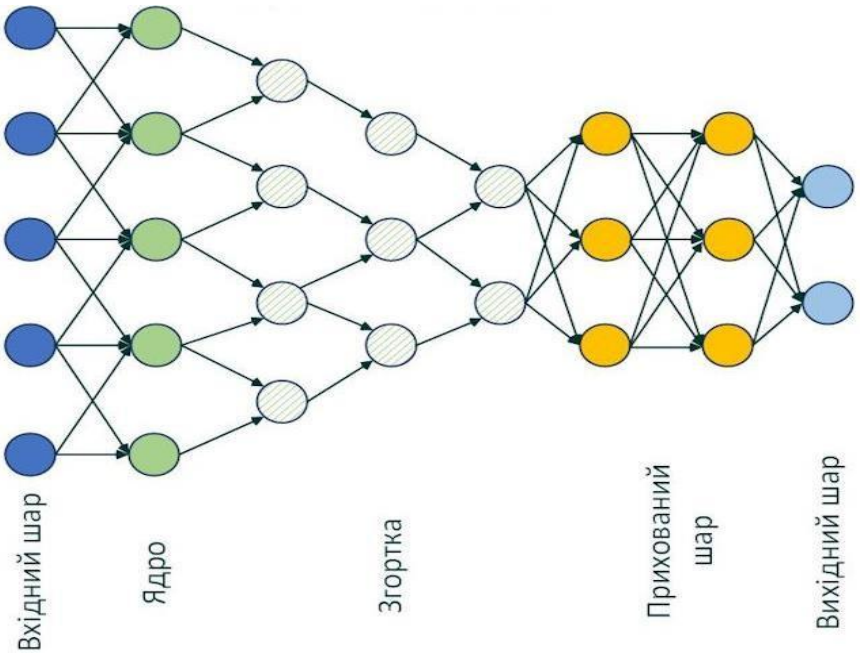

Рисунок 1.1 — Згорткова нейронна мережа

Згорткові шари включають набір фільтрів, які можна уявити як двовимірні матриці чисел. Операція згортки полягає у множенні фрагмента вхідного зображення на елементи ядра фільтра і, як показано на рисунку 1.2, формуванні нової точки даних.

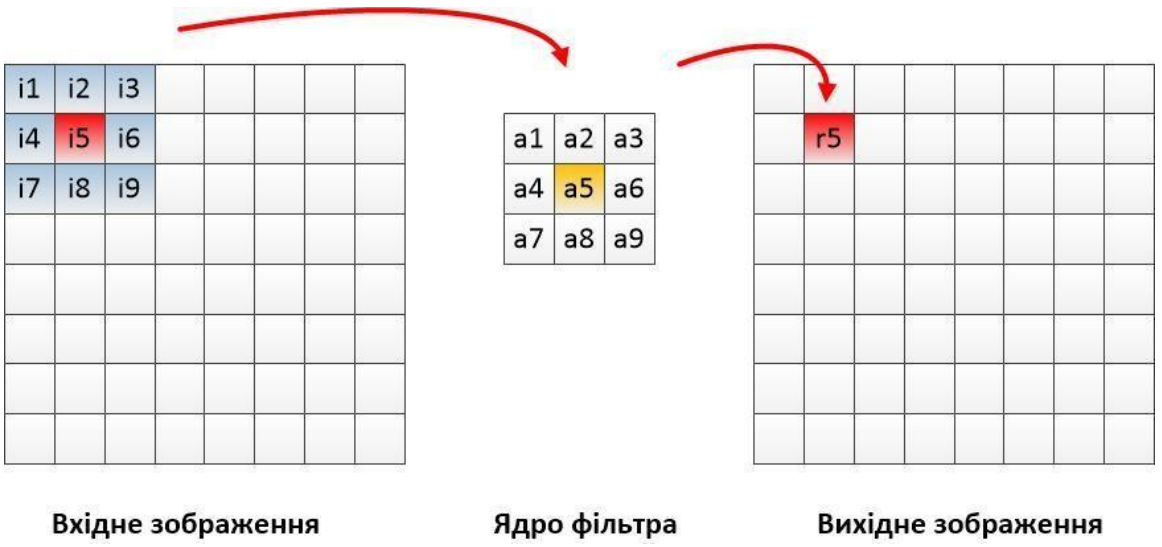

Рисунок 1.2 — Приклад операції згортки

Ми можемо створити вихідне зображення, застосувавши згортковий фільтр до вхідного зображення. Цей процес включає кілька кроків: застосування фільтра до певної області зображення, виконання множення значень фільтра на відповідні значення пікселів зображення, підсумовування отриманих результатів для кожного елемента та повторення цих операцій для кожного пікселя. Однією з ключових переваг згорткових нейронних мереж (CNN) є їх здатність автоматично навчатися виділяти найбільш значущі ознаки з даних. Це стає можливим завдяки процесу навчання, під час якого мережа налаштовує ваги фільтрів, враховуючи похибки, що виникають під час порівняння прогнозів із фактичними значеннями міток. Такий механізм дозволяє CNN ефективно працювати із завданнями, де просторові особливості даних відіграють важливу роль. Згорткові нейронні мережі зазвичай складаються з трьох основних типів шарів: згорткових шарів, шарів об'єднання та повністю зв'язаних шарів. Згорткові шари відповідають за виконання операції згортки, описаної раніше. Шари субдискретизації або об'єднання виконують зменшення розміру карт ознак, що одночасно скорочує обчислювальні витрати та зберігає ключову інформацію. Повністю зв'язані шари використовуються для формування остаточних прогнозів, базуючись на витягнутих ознаках. Важливою характеристикою CNN є їх семантична інваріантність — здатність розпізнавати

об'єкти незалежно від їхнього розташування на зображенні. Це досягається шляхом застосування однакових ваг фільтра у згорткових шарах до всього зображення, що допомагає виявляти аналогічні ознаки на різних його ділянках. Окрім цього, CNN залишаються стійкими до змін масштабу чи обертання об'єктів, що робить їх потужним інструментом у сфері розпізнавання образів. Згорткові нейронні мережі знаходять застосування у багатьох галузях. У медицині вони широко використовуються для автоматичної діагностики захворювань на основі аналізу медичних зображень, таких як рентген чи МРТ. В автономних автомобілях CNN застосовуються для розпізнавання дорожніх знаків, пішоходів та інших елементів інфраструктури. У сфері безпеки вони використовуються для ідентифікації облич і відбитків пальців, а також у системах відеоспостереження для автоматичного виявлення підозрілих дій або об'єктів.

1.3.1 Характеристики навчання згорткових нейронних мереж

Навчання згорткових нейронних мереж проводиться з використанням значних обсягів даних та потужних обчислювальних ресурсів. Для прискорення обчислювальних процесів зазвичай залучають графічні процесори (GPU), які завдяки паралельному підходу до обробки інформації значно підвищують ефективність. Процес побудови моделі охоплює кілька ключових етапів: пряма передача (forward pass), зворотне поширення помилки (backpropagation) та оновлення ваг через метод градієнтного спуску. Ці процедури повторюються багаторазово, поки модель не досягне бажаного рівня точності. Серед головних викликів у навчанні згорткових нейронних мереж виділяють перенавчання, коли модель демонструє високу ефективність на навчальних даних, проте погано працює з новими, невідомими прикладами. Для протидії цьому використовують різні методи регуляризації, наприклад, згладжування вагів (weight decay), відкидання (dropout) чи ранню зупинку (early stopping). Окрім цього, збагачення та розширення навчальних даних також здатне суттєво знизити ризик перенавчання. Згорткові нейронні мережі можна інтегрувати з іншими

архітектурами для вирішення комплексніших завдань. Наприклад, їх поєднують із рекурентними нейронними мережами (RNN) для роботи з послідовностями або автокодерами для безконтрольного навчання ознак. Комбіновані архітектури підвищують гнучкість та функціональні можливості моделей. Розробка моделей на основі згорткових нейронних мереж вимагає не лише технічних знань, а й чіткого розуміння специфіки завдання. До ключових аспектів належать вибір найбільш придатної архітектури, точне коригування гіперпараметрів, якісна підготовка навчальних даних та об'єктивна оцінка результатів. Усі ці складові мають вирішальне значення для забезпечення високої точності й ефективності моделі. У сфері комп'ютерного зору згорткові нейронні мережі залишаються провідною технологією завдяки своїй потужності та універсальності. Їхній розвиток відкриває нові можливості для вирішення задач, що донедавна були поза досяжністю автоматизованих систем. Від автоматизованої класифікації зображень до систем відеомоніторингу та управління автономними транспортними засобами — використання цих моделей змінює уявлення про сучасні технології. Водночас особливого значення набувають етичні та соціальні аспекти впровадження систем, побудованих на згорткових нейронних мережах. Це включає захист конфіденційності даних, уникнення дискримінації та забезпечення прозорості алгоритмічних рішень. Застосування технологій має бути не лише продуктивним, а й спрямованим на позитивний вплив у суспільному контексті. Здатність згорткових нейронних мереж автоматично виділяти й аналізувати складні ознаки із візуальних даних робить їх ключовим інструментом для розпізнавання та обробки зображень. У міру розвитку цієї технології з'являється все більше можливостей для її застосування у промисловості, сприяючи автоматизації та підвищенню ефективності у різноманітних галузях.1

1.3.2 Рекурентні нейронні мережі

EasyOCR використовує рекурентні нейронні мережі (RNN), які є ключовим елементом сучасних технологій розпізнавання тексту. Ці мережі здатні

отримувати інформацію як із попередніх шарів, так і з власних вузлів, що дозволяє їм навчатися запам'ятовувати та відтворювати послідовність вхідних даних. Особливістю RNN є їхня здатність обробляти дані в послідовному форматі завдяки петлям зворотного зв’язку. Ці петлі дозволяють мережам накопичувати інформацію з попередніх кроків і використовувати її в наступних, що є надзвичайно корисним для роботи з задачами, де важлива часово-послідовна структура даних, такими як природна обробка мови, машинний переклад, розпізнавання мовлення та інші подібні випадки. Основною характеристикою RNN є саме петлі зворотного зв’язку, які дають змогу передавати інформацію між різними моментами обробки даних у часі. Це дозволяє мережі враховувати контекст, що стає критичним у задачах, які залежать від порядку введення. Наприклад, під час розпізнавання тексту здатність пам’ятати попередні символи впливає на правильність розуміння наступних, оскільки кожен символ визначає сприйняття наступного. Найпростішим прикладом RNN є стандартна рекурентна нейронна мережа, де кожен нейрон пов’язаний із самим собою протягом певного часового інтервалу. Однак такі моделі мають недоліки, зокрема втрату градієнта під час навчання довгих послідовностей. Для вирішення цієї проблеми були створені більш складні архітектури, як-от довготривала короткочасна пам’ять (LSTM) і рекурентні нейронні мережі з керованими осередками станів (GRU). Загальна структура роботи RNN зазвичай ілюструється на відповідних схемах. 1.3.

LSTM є однією з найпопулярніших і дієвих архітектур серед рекурентних нейронних мереж (RNN). Її головна особливість полягає у використанні комірок пам’яті та механізмів гейтування, які регулюють потік інформації через мережу. Архітектура LSTM включає три ключові компоненти: вхідний гейт, гейт забуття та вихідний гейт. Вхідний гейт відповідає за додавання нової інформації до комірки пам’яті, гейт забуття управляє процесом збереження чи видалення інформації з комірки пам’яті, а вихідний гейт визначає, яка частина інформації з комірки пам’яті буде використана для формування вихідних даних.

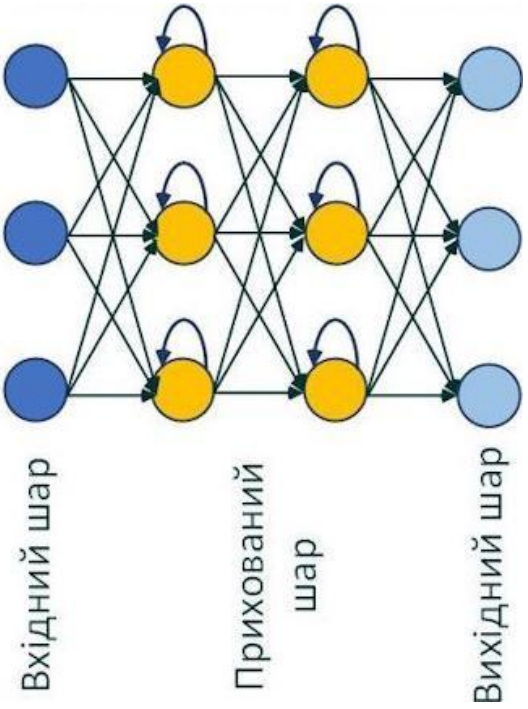

Рисунок 1.3 — Вигляд рекурентної нейронної мережі

Завдяки цим механізмам LSTM ефективно зберігає та забуває інформацію, що робить її потужним інструментом для роботи з довгими послідовностями даних. GRU є спрощеним варіантом LSTM, який також використовує механізми гейтування, проте має менш складну архітектуру та потребує меншої кількості параметрів. Його структура базується на двох основних гейтах: гейт оновлення та гейт забуття. Така спрощена модель оптимізує обчислення та потребує менше ресурсів для навчання мережі, одночасно зберігаючи здатність працювати з послідовними даними ефективно. GRU часто застосовується там, де важливо знайти баланс між точністю розв'язання задачі та обчислювальною продуктивністю. Однією з головних переваг архітектури рекурентних нейронних мереж є здатність працювати із даними змінної довжини. Це особливо актуально для задач обробки природної мови, де довжина текстів — речень або документів — може відчутно варіюватися. Завдяки системі зворотного зв’язку RNN можуть адаптуватися до різної довжини вхідної послідовності, що робить їх надзвичайно універсальними та гнучкими. Рекурентні нейронні мережі знаходять широке застосування у різних сферах. У галузі обробки природної мови вони

використовуються для машинного перекладу, де важливим є збереження контексту перекладу в довгих текстах. Також RNN застосовуються в розпізнаванні мовлення для трансформації аудіосигналів у текст. У генерації тексту вони здатні створювати зв’язні та граматично правильні тексти на основі заданих даних. Крім того, їх використовують у задачах аналізу настроїв для визначення емоційного забарвлення тексту.

1.4 Властивості навчання рекурентної нейронної мережі

Навчання рекурентних нейронних мереж (RNN) включає кілька ключових етапів, таких як пряма передача (forward pass), зворотнє поширення помилки (backpropagation through time, BPTT), і коригування вагів за допомогою градієнтного спуску. Цей процес вимагає значних обчислювальних ресурсів і великого обсягу даних, особливо для складних архітектур на зразок LSTM або GRU. Для пришвидшення обчислень і підвищення ефективності навчання широко застосовуються графічні процесори (GPU). Важливим викликом у навчанні RNN є проблема зникнення та вибухів градієнтів, яка виникає через повторне множення малих або великих чисел під час зворотного поширення помилки. Це може призводити до зміщення величин градієнтів — вони стають надто малими або надмірно великими, що ускладнює оновлення параметрів моделі. Для вирішення цієї проблеми застосовуються методи, як-от нормалізація градієнтів, використання архітектур LSTM і GRU чи регуляризація через відкидання (dropout). Крім того, розробка RNN потребує ретельного врахування задачі, для якої модель створюється, включаючи вибір оптимальної архітектури, налаштування гіперпараметрів, підготовку даних та оцінювання результатів. Від цих факторів залежить кінцева точність та ефективність роботи системи. Рекурентні нейронні мережі можна комбінувати з іншими архітектурами для створення потужних гібридних моделей. Наприклад, їх можна інтегрувати зі згортковими нейронними мережами (CNN) для аналізу послідовних даних із візуальними характеристиками або з трансформерами для підвищення ефективності обробки природної мови. Такий підхід дозволяє створювати більш

універсальні моделі, здатні вирішувати складні завдання. Серед головних переваг RNN — здатність ефективно працювати із серіями даних, що робить їх одним із найкращих інструментів у сучасному машинному навчанні. Їхній розвиток відкриває нові можливості для автоматизації процесів і вирішення завдань, які раніше вважалися непридатними для машинної обробки. RNN знаходять застосування у все більшій кількості сфер, суттєво розширюючи потенціал сучасних технологій. Під час розробки систем на основі RNN важливо враховувати соціальні й етичні аспекти, включаючи захист конфіденційності даних, запобігання дискримінації та забезпечення прозорості алгоритмів. Використання таких технологій має бути відповідальним і спрямованим на створення позитивного впливу на суспільство. Завдяки можливості запам'ятовувати та аналізувати послідовну інформацію, рекурентні нейронні мережі є одним із найпотужніших інструментів для роботи з динамічними даними. Вони дозволяють розробляти системи для виконання завдань машинного перекладу, розпізнавання мовлення, генерації тексту та інших задач. Постійне вдосконалення технологій на основі RNN відкриває нові перспективи для їх застосування в різних галузях, сприяючи автоматизації і підвищенню ефективності.

1.5 Розпізнавання образів

Розпізнавання образів є розділом кібернетики, що спрямований на розробку теоретичних основ та методів класифікації й ідентифікації об'єктів, явищ, процесів, симптомів, станів та інших елементів, які характеризуються обмеженим набором чітко визначених властивостей і характеристик. Такого роду завдання часто трапляються в повсякденному житті, наприклад, під час переходу дороги на світлофорі або руху автомобілем. Визначення кольору сигналу світлофора та знання правил дорожнього руху дають змогу прийняти правильне рішення щодо моменту переходу дороги. У процесі біологічної еволюції багато видів тварин успішно вирішили проблему розпізнавання образів завдяки своїм зоровим і слуховим системам. Водночас створення штучних систем

розпізнавання образів залишається однією з найбільших теоретичних і технічних викликів. Потреба в таких системах існує в найрізноманітніших сферах – від військової справи та систем безпеки до оцифрування аналогових сигналів. До завдань розпізнавання образів можуть належати:

— ідентифікація літер;

—ідентифікація штрих-кодів;

— ідентифікація номерних знаків;

— ідентифікація облич;

— ідентифікація мовлення;

— ідентифікація зображень.

Для оптичного розпізнавання зображень використовуються різні підходи, які враховують тип об'єкта, його положення під різними кутами, масштаб, зсув тощо. Якщо мова йде про літери, необхідно визначити шрифт, його параметри та властивості. Другий підхід базується на аналізі форми об'єкта з подальшою перевіркою його характеристик, таких як з'єднання або наявність кутів. Іншим методом є застосування штучних нейронних мереж. Для цього підходу потрібен або великий обсяг навчальних прикладів з визначеними правильними відповідями, або спеціальна структура нейронної мережі, яка враховує особливості завдання. Френк Розенблатт описав експерименти, що спрощують моделювання мозкової діяльності, демонструючи, як психологічні явища можуть виникати у фізичних системах із чітко заданою структурою і функціональними властивостями. Ці експерименти, хоч і пов’язані з розпізнаванням зображень, не мають строго детермінованого алгоритму розв’язання. У найпростішому експерименті випробувачеві демонструють два різні стимули та вимагають від нього неоднакової реакції на кожен із них. Мета такого експерименту може полягати у вивченні здатності системи спонтанно диференціювати стимули без прямого втручання експериментатора або у дослідженні примусової диференціації. Це останнє орієнтоване на навчання системи виконувати певну класифікацію за підказкою експериментатора. У ході навчання персептрона йому послідовно подають зображення, які відповідають кожному класу для

розпізнавання. Пам’ять моделі вдосконалюється завдяки правилу підкріплення правильних відповідей. Після цього систему тестують за допомогою контрольного стимулу, визначаючи ймовірність правильної реакції на стимули цього класу. Результати тесту залежать від того, чи збігається контрольний стимул із будь-яким із зображень, запропонованих у процесі навчання: — якщо контрольний стимул не співпадає з жодним із навчальних зразків, тест включає не тільки елемент дискримінації, але й узагальнення; — якщо контрольний стимул активує набір сенсорних елементів, повністю відмінних від тих, що діяли під час навчання, дослідження є прикладом чистого узагальнення. Персептрони не володіють здатністю до чистого узагальнення, однак вони демонструють високі результати в задачах дискримінації. Особливо ефективні вони в ситуаціях, коли контрольний стимул точно відповідає одному з образів, які система обробляла раніше.

1.6 Ідентифікація оптичних властивостей

Оптичне розпізнавання символів (OCR) – це технологія, що використовується для обробки сканованих документів, зображень текстових матеріалів, сцен із текстом (наприклад, написи на вивісках або дошках), чи навіть для аналізу тексту, накладеного на зображення (як, наприклад, субтитри на екранах телевізійних трансляцій). Вона поширено застосовується для введення та обробки даних з різних джерел, таких як паспорти, рахунки-фактури, банківські виписки, квитанції, візитні картки, пошта та будь-які інші текстові документи. Основними її завданнями є електронне редагування, пошук тексту, компактне зберігання даних, їх перегляд онлайн або машинна обробка для складніших завдань: аналізу тексту, когнітивних обчислень, перетворення тексту в мовлення чи переклад на інші мови. OCR є ключовою технологією в розробці алгоритмів розпізнавання образів, штучного інтелекту та комп’ютерного зору. Перші системи OCR вимагали підготовки — кожний шрифт мав бути визначений та "вивчений" системою окремо. Зараз же сучасне програмне забезпечення OCR здатне обробляти багато шрифтів одночасно, підтримуючи різні формати

зображень і досягаючи високої точності. Деякі системи навіть можуть зберігати оригінальну структуру документа, включаючи зображення, колонки та інші графічні елементи. Історично важливу роль у розвитку OCR відіграв Рей Курцвейл. У 1974 році він заснував Kurzweil Computer Products Inc. та створив багатоформатну OCR-систему, яка могла розпізнавати друкований текст майже будь-яким шрифтом. Ця технологія стала основою для пристрою читання для осіб із порушеннями зору: комп'ютер перетворював друкований текст у голос за допомогою планшетного сканера CCD та синтезатора мовлення. У 1976 році цей пристрій був офіційно презентований під час масштабної прес-конференції. Уже в 1978 році компанія почала продаж програмного забезпечення OCR. Першим великим клієнтом стала компанія LexisNexis, яка впровадила цю технологію для завантаження юридичних текстів у свої бази даних. Пізніше Курцвейл продав свою компанію корпорації Xerox, яка в подальшому перетворилася на Scansoft і злилася з Nuance Communications. У 2000-х роках OCR як послуга (так звана WebOCR) знайшла застосування в середовищах хмарних обчислень і почала активно інтегруватися у мобільні додатки. Один із прикладів — це переклад тексту іноземними мовами в реальному часі через камеру смартфона. До того ж із появою таких гаджетів як смартфони і розумні окуляри OCR почав використовуватися на мобільних пристроях, підключених до Інтернету. Такі пристрої найчастіше використовують API для розпізнавання тексту, що дозволяє аналізувати відзняті зображення та передавати текст разом із його розташуванням на оригінальному зображенні прямо на пристрій користувача для подальшого застосування чи обробки, наприклад, перетворюючи текст у мовлення. Сьогодні доступні як комерційні системи OCR, так і рішення з відкритим кодом для роботи з найрізноманітнішими системами письма: латиницю, кирилицю, арабську, іврит, індійські та азійські алфавіти — бенгальський, деванагарі, тамільський, китайський, японський і корейський. Технології OCR знаходять широкий спектр застосувань у повсякденному житті й бізнесі.

Програмне забезпечення може застосовуватись у різноманітних сценаріях:

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

— розпізнавання паспортних документів і доступ до інформації щодо аеропортів;

— автоматичне вилучення ключової інформації з документів страхування;

— виявлення дорожніх знаків; — перетворення візитних карток у контактні дані;

— створення текстових версій друкованих документів, наприклад, сканування книг для проєктів на кшталт Gutenberg;

— створення електронних версій друкованих документів з функцією пошуку, як у Google Books;

— перетворення рукописного тексту в реальному часі для управління комп'ютером або обчислень «від руки»;

— тестування систем CAPTCHA, навіть тих, що розраховані на запобігання роботам типу OCR;

— розробки допоміжних технологій для людей з вадами зору;

— написання технічних інструкцій до транспортних засобів через аналіз зображень з баз даних CAD у реальному часі;

— перетворення сканованих документів у PDF-файли з можливістю пошуку.

Оптичне та машинне навчання займаються такими задачами, як визначення оптичних властивостей:

— оптичне розпізнавання тексту;

— інтелектуальне розпізнавання символів або слів.

Серед технологій розпізнавання тексту можна виділити приведені нижче. Оптичне розпізнавання символів (OCR), дозволяє розпізнавати машинописний текст, аналізуючи та обробляючи його за символами. OCR використовується для цифровізації друкованих документів, але точність може знижуватись при

обробці рукописного тексту чи текстів, надрукованих особливими шрифтами або з ушкодженнями.

Оптичне розпізнавання слів (OWR) спрямоване на розпізнавання тексту по словам, а не по окремих символах. Найкращих результатів досягають для мов із пробільним поділом між словами, але ефективність знижується в суцільних текстах без проміжків. Інтелектуальне розпізнавання символів (ICR). У своєму аналізі використовує техніки машинного навчання для розпізнання рукописного чи декоративного тексту за окремими символами. Проте, успішність може залежати від особливостей почерку та стилю написання.

Інтелектуальне розпізнавання слів (IWR) використовується для роботи з рукописними текстами та графікою для аналізу цілих слів. Воно особливо корисно для мов, де слова не розділяються пробілами. Ефективність може падати на текстах із надмірним злиттям слів.

Також існують хмарні сервіси із підтримкою онлайн-OCR API. Для обробки рукописного тексту можлива інтеграція рухових даних про написання (наприклад, напрямки ліній пера, порядок і тривалість його рухів), що значно полегшує процес розпізнавання. Така техніка належить до методів динамічного розпізнавання тексту та використовується в режимах реального часу. Методи OCR поділяються на дві основні категорії.

Одна з них — кореляційні матриці, вони дозволяють порівнювати зображення із збереженим гліфом на основі пікселів. Цей процес також відомий як зіставлення зі зразком, розпізнавання образів або порівняння зображень. Ефективність цього методу залежить від того, чи правильно ізольований вхідний гліф та чи відповідає збережений гліф за шрифтом і масштабом. Метод добре працює з друкованим текстом, але має труднощі зі шрифтами, не представленими у базі. Це був перший підхід до автоматизованого оптичного розпізнавання тексту, заснований на фотоелементах.

Друга категорія — метод вилучення ознак, він розбиває гліфи на складові елементи, такі як лінії, замкнуті контури, напрями чи вузли перехресть. Такий

підхід значно знижує рівень складності даних та оптимізує обчислювальні ресурси. Отримані ознаки аналізуються у вигляді абстрактного векторного представлення символу, що зводиться до одного чи декількох прототипних гліфів. Цей підхід широко використовується в додатках для «розумного» розпізнавання рукописного тексту та є основою сучасних моделей OCR. Для класифікації використовуються такі методи, як алгоритми k-найближчих сусідів, які співвідносять ознаки зображення із заздалегідь збереженими зразками та обирають найкращий збіг. Програми, наприклад, Cuneiform і Tesseract, застосовують двопрохідний метод розпізнавання символів. Другий етап — адаптивне розпізнавання — орієнтований на підвищення точності ідентифікації завдяки аналізу символів, розпізнаних на першому етапі. Це особливо корисно для обробки незвичних шрифтів або низькоякісних сканованих матеріалів із спотворенням текстури. Сучасні OCR-інструменти, такі як ABBYY FineReader, Google Docs OCR і Transym, активно застосовуються у різних галузях. Наприклад, інструменти на основі нейронних мереж, такі як нові версії Tesseract, можуть обробляти текст цілими рядками замість окремих символів.

Інший перспективний підхід — ітеративне OCR-розпізнавання. У цьому методі система динамічно розбиває документ на сегменти відповідно до макета сторінки. Для кожного сегмента оцінюється рівень достовірності символів, що дозволяє покращити загальну точність і забезпечити коректне розпізнавання документів у цілому. Патентне відомство США офіційно зафіксувало цей метод. Отриманий результат OCR можна експортувати у спеціалізовані формати, такі як ALTO XML, підтримуваний Бібліотекою Конгресу США, або стандартні формати hOCR і PAGE XML. Подальша обробка даних дозволяє збільшити точність за допомогою використання спеціального лексикону слів, які, ймовірно, присутні у тексті (наприклад, технічного словника певної галузі). Однак, це може викликати складнощі за наявності незвичних слів чи власних назв. Tesseract активно використовує лексикон для покращення сегментації символів і підвищення якості результатів. Інші сучасні системи OCR підтримують високоінформативне представлення результатів, наприклад створення PDF-

документів із можливістю пошуку за текстом і збереженням вихідного макета сторінки. Для виправлення помилок аналіз ближніх сусідів враховує частоти спільної появи слів у тексті. Наприклад, конструкція типу "Вашингтон, округ Колумбія" зустрічається значно частіше за помилковий варіант "Вашингтон ДОК".

2 ОБГРУНТУВАННЯ ВИБОРУ ТЕХНОЛОГІЙ ДЛЯ ВИКОНАННЯ РОБОТИ

2.1 Огляд наявних рішень

Розпізнавання тексту з фотографій є важливим завданням у сучасних інформаційних системах і додатках, що обробляють значні обсяги даних. Сьогодні існує низка поширених рішень, які забезпечують високу якість та точність розпізнавання тексту з зображень.

Одним із таких рішень є Microsoft Cognitive Services, що пропонує API для вилучення тексту з фотографій. Серед його ключових переваг — висока точність результатів і зручна інтеграція з іншими сервісами екосистеми Microsoft. Водночас слід зважати на те, що використання цього сервісу може бути фінансово затратним при роботі з великими обсягами даних.

Microsoft Cognitive Services, Amazon Textract та Google Cloud Vision API є ідеальними варіантами для компаній, які шукають інтеграційні можливості з іншими популярними інфраструктурами чи потребують швидкості й надійності. ABBYY FineReader та ABBYY Cloud OCR SDK залишаються перевіреними рішеннями для корпоративного сектору завдяки високій точності та адаптивності до різних типів документів. Для бюджетних чи академічних потреб відкриті інструменти, такі як Tesseract, Kraken OCR чи EasyOCR, можуть стати чудовою альтернативою, особливо коли мова йде про підтримку рідкісних мов або роботи з історичними текстами чи рукописами. Мобільні додатки, такі як Adobe Scan і CamScanner, надають зручність використання при скануванні документів у дорозі. У свою чергу, спеціалізовані інструменти на кшталт Mathpix пропонують розширені функції, наприклад, розпізнавання математичних формул, що є корисними для студентів чи дослідників. Прогрес у нейронних мережах і алгоритмах машинного навчання також відкриває нові горизонти для підвищення точності та адаптивності OCR-технологій. Завдяки сучасним підходам до обробки зображень стало можливим досягати ефективних результатів навіть за умови роботи з низькоякісними матеріалами. Зрештою,

остаточний вибір залежить від конкретних вимог проекту. Розуміння особливостей кожного інструмента допоможе підібрати оптимальне рішення для задач у будь-якій сфері — від бізнесу до освіти чи науки.

Одним із цікавих варіантів є ABBYY FineReader, спеціалізований на оптичному розпізнаванні символів (OCR). Ця програма відзначається високою точністю і здатністю працювати з різними типами документів та мовами. Завдяки довгій історії розвитку, ABBYY FineReader є одним із найнадійніших інструментів OCR, що особливо цінується у корпоративному середовищі. Ще одним цікавим рішенням є ABBYY Cloud OCR SDK, який також фокусується на розпізнаванні тексту з використанням фотографій. Висока точність і підтримка широкого спектра мов роблять цей інструмент універсальним. Використання хмарних технологій дає змогу знизити вимоги до ресурсів апаратного забезпечення користувача, що забезпечує доступність цього продукту для різних категорій користувачів. Окрім зазначених рішень, існують і менш відомі, проте не менш ефективні альтернативи для розпізнавання тексту. Наприклад, Tesseract — один із найпопулярніших безкоштовних інструментів OCR. Ця програма, спочатку створена HP і пізніше підтримана Google, є відкритою для використання в різних проєктах. Tesseract підтримує велику кількість мов і пропонує високу гнучкість, що є особливо цінним для розробників із обмеженим бюджетом. Kraken OCR — ще один інструмент із відкритим кодом, орієнтований на забезпечення високої точності та підтримку різноманітних мов, зокрема історичних і рідкісних. Цей інструмент створений з урахуванням потреб наукових досліджень і зручний для роботи з рукописними документами та архівними матеріалами. З-поміж інших рішень виділяється EasyOCR, який здобув популярність завдяки своїй простоті у використанні й високій точності. Він підтримує понад 80 мов і також є відкритим продуктом, що дозволяє легко інтегрувати функції OCR у різні додатки. Ця програма використовує сучасні нейронні мережі, що забезпечує відмінні результати навіть на зображеннях зі складними умовами.

Крім того, у галузі мобільних застосунків варто звернути увагу на такі інструменти, як Adobe Scan **і** CamScanner. Adobe Scan — це додаток, який надає можливість сканувати документи та здійснювати розпізнавання тексту з використанням технологій OCR. Його тісна інтеграція з іншими сервісами Adobe робить цей інструмент зручним і ефективним для роботи з різними типами документів. CamScanner, у свою чергу, також є широко розповсюдженим мобільним застосунком для сканування та розпізнавання тексту, який вирізняється простотою використання та якісною обробкою зображень.

Важливим критерієм під час вибору OCR-рішення є наявність підтримки додаткових функцій, зокрема розпізнавання рукописного тексту або математичних формул. Для виконання таких завдань застосовуються спеціалізовані інструменти, серед яких варто виділити Mathpix**.** Цей сервіс дає змогу розпізнавати математичні вирази та конвертувати їх у текстовий формат, що є особливо корисним для студентів і науковців, які працюють із навчальними та науковими матеріалами.

Окремо слід зазначити, що розвиток технологій штучного інтелекту та машинного навчання суттєво розширює можливості систем розпізнавання тексту. Сучасні нейронні мережі та методи глибокого навчання забезпечують підвищення точності навіть при роботі зі складними або низькоякісними зображеннями. Зокрема, використання згорткових нейронних мереж (CNN) на етапі попередньої обробки зображень дозволяє виокремлювати ключові ознаки та покращувати результати розпізнавання.

Отже, на сьогоднішній день існує широкий вибір рішень для розпізнавання тексту з фотографій, кожне з яких має свої сильні та слабкі сторони. Вибір оптимального інструменту визначається характером поставлених завдань, вимогами до точності й швидкості обробки, підтримкою різних мов та іншими чинниками. Незалежно від обраного підходу, застосування OCR-технологій суттєво підвищує ефективність роботи з інформацією та робить її більш доступною для користувачів.

2.2 Опис проблеми

Метою даної роботи є створення застосунку для розпізнавання тексту з зображень, спрямованого на підвищення доступності інформації для людей з інвалідністю, зокрема для незрячих і слабозорих користувачів. Розробка такого застосунку має важливе соціальне значення, оскільки надає можливість швидко та зручно отримувати текстову інформацію з візуальних матеріалів, що є актуальним у повсякденному житті та сприяє підвищенню рівня самостійності й інтеграції цих людей в суспільство.

Запланований застосунок передбачає реалізацію таких основних функцій:
— зйомка зображень за допомогою камери пристрою;
— автоматичне розпізнавання тексту на отриманих зображеннях;
— озвучення розпізнаного тексту з використанням технологій синтезу мовлення.

Функція фотографування дозволяє користувачеві зафіксувати об’єкт, що містить текстову інформацію. Механізм розпізнавання забезпечує перетворення тексту із зображення у цифровий текстовий формат, а озвучення надає можливість прослуховувати отриманий результат. Сукупність цих можливостей значно спрощує доступ до тексту з фотографій, робить повсякденне користування інформацією зручнішим та сприяє більш комфортному й незалежному способу життя користувачів.

2.3 Вибір моделей архітектури

2.3.1 Типи моделей архітектур

2.3.1.1 Каскадна модель

У 1960-х роках Вінстон Ройс описав підхід до розробки програмного забезпечення, відомий як **Waterfall Model**, або каскадна модель. Вона передбачає поділ процесу створення програмного забезпечення на послідовні етапи, кожен з яких охоплює визначений набір завдань і спирається на результати попередньої фази. Такий підхід тривалий час застосовувався в окремих галузях проєктної інженерії та характеризується лінійним, односпрямованим рухом процесу — від початкової концепції до етапу супроводу.

У межах каскадної моделі розробка програмного забезпечення проходить через послідовність фаз, серед яких визначення концепції, ініціація проєкту, аналіз, проєктування, розробка, тестування, впровадження та подальша підтримка. Через відсутність гнучких ітерацій цей підхід вважається жорстким, однак він забезпечує чітку структуру та передбачуваність процесу.

Спочатку каскадна модель була орієнтована на використання в промисловості та будівництві, де зміни в проєкті на ранніх етапах потребували значних витрат. Проте на той час вона фактично залишалася єдиним можливим підходом для організації процесу розробки програмного забезпечення. Перший детальний опис застосування фаз у програмній інженерії був запропонований Феліксом Торресом і Гербертом Д. Шмідтом.

Хоча термін «каскадна модель» не використовувався безпосередньо під час її створення, формальні принципи цього підходу часто пов’язують зі статтею Вінстона В. Ройса, опублікованою у 1970 році. У 1985 році Міністерство оборони США закріпило цей підхід у стандарті DOD-STD-2167A**,** який передбачав обов’язкове проходження шести основних фаз розробки програмного забезпечення: аналіз вимог, попереднє проєктування, детальне проєктування, кодування та модульне тестування, інтеграцію і системне тестування.

Вимоги до системи та програмного забезпечення фіксуються у відповідній документації. На етапі аналізу формуються моделі, діаграми та бізнес-правила, а під час проєктування створюється архітектура програмного забезпечення. Фаза кодування охоплює розробку, перевірку та інтеграцію програмних компонентів. Діагностика спрямована на виявлення та усунення помилок, тоді як експлуатація включає встановлення, супровід і обслуговування завершених систем.

Ключовим принципом каскадної моделі є перехід до наступного етапу лише після завершення, перевірки та оцінювання попереднього. Водночас існують модифіковані версії цього підходу, зокрема пізніші варіації моделі Ройса, які допускають повернення до попередніх фаз або повторне виконання окремих етапів у разі виявлення недоліків.

Однією з головних переваг каскадної моделі є акцент на детальній документації, зокрема вимогах, проєктних матеріалах і вихідному коді. Це зменшує ризик втрати інформації у разі зміни складу команди та полегшує залучення нових учасників до проєкту. Завдяки чітко визначеним фазам і контрольним точкам каскадна модель забезпечує систематичний та зрозумілий процес розробки, що спрощує відстеження прогресу. Саме тому вона часто використовується як базова модель для вивчення принципів програмної інженерії в навчальних курсах і підручниках.

2.3.1.2 Модель Беллмана

«Модель Беллмана» насправді є математичним методом для оптимізації серії рішень. У розпізнаванні образів вона була і залишається ключовим інструментом для вирішення проблем вирівнювання, порівняння спотворених послідовностей та розробки ефективних функцій втрат в архітектурах нейронних мереж, де потрібно знайти найкращий спосіб зіставлення між даними та виходом.

Робота Річарда Беллмана безпосередньо пов'язана з розпізнаванням шрифтів (і в ширшому сенсі оптичним розпізнаванням символів або OCR) через використання динамічного програмування та його принципу оптимальності.

Спочатку відбувається оптимізація на основі шаблонів (зіставлення шаблонів). Ранні та класичні системи оптичного розпізнавання символів (OCR) використовували підхід, заснований на порівнянні зображення символу з набором еталонних шаблонів. Проблема полягала в тому, щоб знайти найкраще відповідне зображення, враховуючи можливі спотворення, незначні зміни, зміни масштабу тощо.

Динамічне програмування (DP) дозволяє ефективно вирішувати завдання пошуку оптимального шляху або найкращого відповідного зображення, розбиваючи його на послідовність менших та більш керованих підзадач.

У контексті розпізнавання, DP використовується для «вирівнювання» або зіставлення контурів чи рис ідентифікованого символу з рисами шаблону. Він шукає оптимальну послідовність локальних змін (наприклад, деформації або

зміщення), яка мінімізує загальну помилку або «штраф» між вхідним зображенням та шаблоном.

2.3.2 Принцип оптимальності Беллмана

Основний принцип Беллмана стверджує, що «оптимальне рішення багатоетапного процесу має властивість, що незалежно від попередніх рішень, решта рішень повинні утворювати оптимальне рішення відносно умов, отриманих в результаті попередніх рішень».

Коли ми розпізнаємо форми літер (або літер у тексті), тобто, якщо ми розпізнаємо текст як послідовність літер, вибір найкращої поточної літери залежить не лише від її власного шаблону найкращого збігу, але й від того, як цей вибір впливає на ймовірність кращого розпізнавання наступних літер у рядку або слові (враховуючи контекст і когерентність).

Динамічне викривлення часу (DTW) — це класичне застосування динамічного програмування, яке можна використовувати для порівняння вироджених послідовностей, наприклад, при порівнянні текстової розмітки зі стандартною.

2.3.3 Сучасний підхід

У сучасних системах розпізнавання шрифтів домінують глибокі нейронні мережі (ГНМ). Хоча явне «ручне» програмування на основі ДП використовується рідше, основні ідеї Беллмана залишаються актуальними;

Навчання з підкріпленням (НП), яке активно використовує рівняння Беллмана для визначення оптимальної стратегії прийняття рішень, може бути застосоване до складних систем розпізнавання, де агент приймає серію рішень щодо класифікації сегментів зображення або корекції процесу розпізнавання.

Таким чином, робота Беллмана надає базовий математичний інструмент (динамічне програмування) для вирішення задач оптимізації, які є основою класичних та деяких сучасних методів розпізнавання зображень.

2.4 Керування вимогами в модифікованих моделях розробки

На практиці замовники не завжди мають чітко сформульовані вимоги до програмного продукту до моменту, коли можуть побачити його роботу. Це часто призводить до змін у вимогах уже в процесі розробки, що, своєю чергою, може вимагати повторного проєктування, доопрацювання та додаткового тестування, збільшуючи витрати часу й ресурсів.

Щоб уникнути потенційних проблем під час створення нового продукту або функціоналу, розробникам доцільно переглядати початкові рішення та адаптувати проєкт відповідно до виявлених обмежень і потреб. У таких ситуаціях доопрацювання концепції є ефективнішим підходом, ніж реалізація рішення, яке не відповідає реальним вимогам або не усуває наявні проблеми.

Деякі організації намагаються мінімізувати ризики, пов’язані з нечіткими вимогами, залучаючи системних аналітиків для дослідження наявних ручних або автоматизованих систем і аналізу їх функціональних можливостей. Однак на практиці чітке розмежування між етапами системного аналізу та програмування є складним, оскільки впровадження складних програмних рішень майже завжди виявляє нові проблеми та граничні випадки, які не були враховані на початкових етапах.

З огляду на ці виклики були запропоновані модифіковані варіанти каскадної моделі, зокрема «модель сашимі», «модель із перекриттям фаз», «модель підпроєктів» та «модель зниження ризиків». Такі удосконалені підходи спрямовані на подолання обмежень класичної каскадної моделі шляхом підвищення гнучкості процесу розробки, зменшення витрат і спрощення управління змінами впродовж життєвого циклу програмного забезпечення.

2.5 Модель клієнт-сервер: принципи та застосування

Модель клієнт–сервер, або мережева архітектура клієнт–сервер, є розподіленою платформою для побудови застосунків, у межах якої обчислювальні завдання та робочі навантаження розподіляються між

постачальниками ресурсів чи послуг — серверами, та їх споживачами — клієнтами [1]. Зазвичай клієнти й сервери взаємодіють через комп'ютерну мережу, розміщуючись на різних фізичних пристроях, однак можливий і варіант їх роботи в межах однієї системи. Серверний хост запускає одну або декілька серверних програм, які надають доступ до його ресурсів, тоді як клієнтські програми звертаються до сервера із запитами, не надаючи власні ресурси у спільне користування.

Клієнти ініціюють сеанси взаємодії з серверами, очікуючи відповіді на надіслані запити. До прикладів програмних систем, побудованих за моделлю клієнт–сервер, належать електронна пошта, веб-застосунки та сервіси Всесвітньої мережі. Сам термін «клієнт–сервер» використовується для опису взаємозв'язку між програмними компонентами, що спільно функціонують у межах одного застосунку. Серверна частина надає певні функції або сервіси одному чи кільком клієнтам, які звертаються до неї із запитами.

Сервери класифікуються відповідно до типу послуг, які вони забезпечують. Наприклад, веб-сервери відповідають за обслуговування веб-сторінок, а файлові сервери — за зберігання та надання доступу до файлів. Спільними ресурсами можуть бути програмні засоби, дані, обчислювальні потужності, а також пристрої зберігання інформації. Роль комп'ютера як клієнта або сервера визначається програмним забезпеченням, яке на ньому виконується. Один пристрій може одночасно виконувати функції кількох серверів, а також бути клієнтом інших сервісів. Крім того, клієнтське та серверне програмне забезпечення можуть взаємодіяти в межах одного комп'ютера [2]. Взаємодію між серверами, наприклад для обміну або синхронізації даних, називають міжсерверною комунікацією.

Клієнту не потрібно знати внутрішні механізми роботи сервера — достатньо розуміти формат і зміст відповіді відповідно до визначеного протоколу. Взаємодія відбувається за схемою «запит–відповідь», що є прикладом міжпроцесної комунікації. Для коректної роботи клієнт і сервер мають використовувати спільні правила обміну даними, які визначаються протоколами

прикладного рівня. З метою формалізації взаємодії сервер може надавати інтерфейс прикладного програмування (API), який виступає рівнем абстракції доступу до сервісів, спрощує обробку даних і забезпечує сумісність між різними платформами [3,4].

Сервер здатний обробляти запити від великої кількості клієнтів за короткий проміжок часу, проте його ресурси є обмеженими, тому використовується механізм планування для розподілу обчислювальних задач. З метою підвищення доступності та захисту від зловживань серверне програмне забезпечення може обмежувати кількість запитів. Зокрема, атаки типу «відмова в обслуговуванні» спрямовані на перевантаження сервера надмірною кількістю запитів. У разі передавання конфіденційної інформації між клієнтом і сервером застосовуються механізми шифрування.

Серверне програмне забезпечення є програмою, що виконується на віддаленому серверному обладнанні, доступ до якого здійснюється з персональних комп'ютерів, мобільних пристроїв або інших клієнтських систем. Частина операцій виконується саме на стороні сервера, оскільки вони потребують доступу до централізованих даних, спеціалізованих ресурсів або мають підвищені вимоги до безпеки й продуктивності.

Клієнтські та серверні застосунки можуть бути як загальнодоступними, наприклад веб-браузери та веб-сервери, що взаємодіють за стандартизованими протоколами, так і спеціалізованими рішеннями з власними протоколами обміну даними. Серверні операції охоплюють не лише обробку клієнтських запитів, а й фонові процеси, зокрема завдання з адміністрування та технічного обслуговування системи [6,7].

2.6 Проектування системи

Випадки використання необхідно детально аналізувати з метою глибшого розуміння призначення інформаційної системи та можливих напрямів її подальшого функціонування. Одним із найпростіших і водночас наочних

способів відображення взаємодії користувачів із системою є діаграма варіантів використання, яка демонструє сценарії роботи системи з точки зору користувача.

Такі діаграми дозволяють виокремити різні категорії користувачів та відповідні моделі їхньої взаємодії із системою. Для формування повнішого уявлення про архітектуру та поведінку системи їх часто застосовують у поєднанні з іншими видами діаграм. Хоча кожен окремий сценарій може бути детально описаний, саме діаграма варіантів використання забезпечує узагальнений огляд функціональних можливостей системи.

Використання цих діаграм значно полегшує комунікацію між усіма зацікавленими сторонами та сприяє кращому розумінню логіки й напрямів розвитку системи. Недарма вважається, що саме варіанти використання становлять основу будь-якої інформаційної системи.

Завдяки простоті та наочності діаграми варіантів використання є ефективним засобом представлення вимог і очікувань для зацікавлених сторін. Вони відображають реальні сценарії взаємодії та допомагають користувачам і розробникам чіткіше уявити, як система працюватиме на практиці. Дослідження підтверджують, що такі діаграми доносять мету й функціональність системи зрозуміліше, ніж діаграми класів, особливо на ранніх етапах проєктування.

На рисунку 2.1 показано діаграму використання, що описує можливі дії користувача в системі.

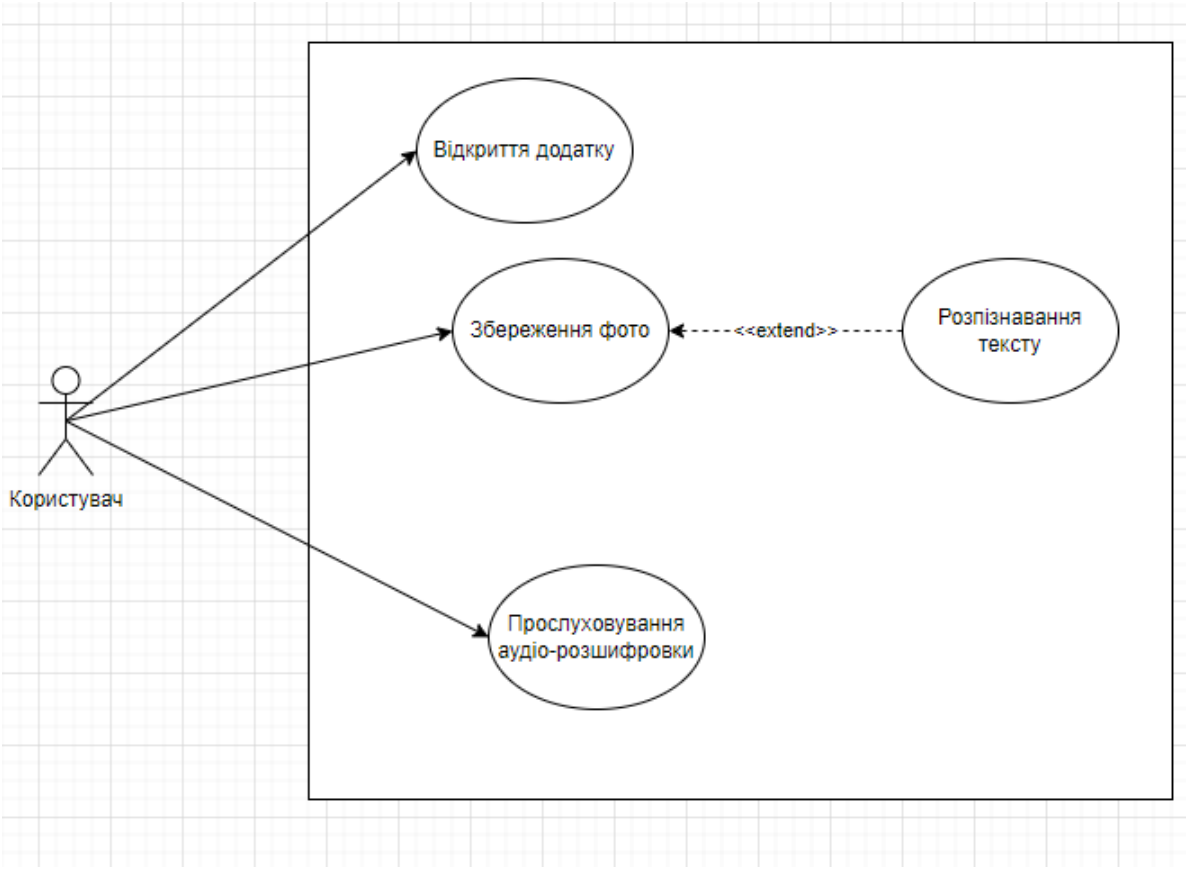

Рисунок 2.1 — Діаграма використання

Основною метою використання діаграм є демонстрація динамічних аспектів системи. Інші діаграми та документація можуть бути використані для надання більш повного функціонального та технічного опису системи. Вони забезпечують спрощене графічне представлення того, як система повинна працювати, для цього оперують такими поняттями:

— межа — це прямокутник (випадок використання) з порядком назви та еліпса, але може бути опущена за відсутності корисної інформації;

— актор — стилізована роль особи (акторів, не пов'язаних один з одним), яка представляє набір користувачів, що взаємодіють із системою або іншими сутностями;

— випадок використання — позначений еліпс, що вказує на системну операцію, що виконується інтерфейсом користувача та призводить до заданого результату. Назва може описувати «що» робить система, а не «як». Випадки використання представляють різні сценарії використання системи.

2.7 Дизайн проекту

У процесі розробки програмного забезпечення внутрішню структуру системи зазвичай подають у вигляді структурної діаграми, яка відображає складові проєкту, зокрема групи та модулі, а також взаємозв'язки між ними. У програмній інженерії відповідно до стандартів Уніфікованої мови моделювання (UML) для цього використовується діаграма класів — статична структурна діаграма, що надає узагальнене уявлення про будову системи, її компоненти, їхню поведінку, механізми роботи та зв'язки між об'єктами.

Діаграми класів є базовими елементами об'єктно-орієнтованого моделювання. Вони застосовуються як на етапі концептуального проєктування для загального опису структури програмного забезпечення, так і на детальніших етапах — для подальшого перетворення моделей у програмний код. Окрім цього, діаграми класів можуть використовуватися як інструмент моделювання даних, оскільки кожен клас відображає ключові сутності та елементи, що підлягають реалізації.

Стандартна діаграма класів складається з трьох основних частин:

— верхній блок, який містить назву класу, зазвичай виділену жирним шрифтом і розташовану по центру, при цьому перша літера назви пишеться з великої літери;

— середній блок, у якому наведено атрибути класу, вирівняні ліворуч, із назвами, що починаються з малої літери;

— нижній блок, який описує операції або методи класу, також вирівняні ліворуч і позначені назвами з малої літери.

Під час проєктування системи визначення компонентів та їх групування на діаграмах допомагає зрозуміти статичні зв'язки між елементами. Для більш детального опису системи концептуальне проєктування часто поділяють на кілька підкатегорій, використовуючи різні типи UML-діаграм.

Асоціація в UML описує зв’язок між елементами моделі. Якщо зміна одного елемента може впливати на інший, між ними існує асоціативний зв’язок. Такий зв’язок може бути односпрямованим або двонаправленим і зазвичай зображається лінією зі стрілкою, що вказує напрям взаємодії. Для розширеного опису поведінки системи ці діаграми можуть доповнюватися діаграмами станів або машинами станів UML.

Асоціації поділяються на кілька типів, серед яких найпоширенішими є односпрямовані та двонаправлені зв’язки, а також агрегація, композиція та рефлексивна асоціація. Наприклад, зв’язок між класами Flight і Airplane може бути поданий як бінарна асоціація, що відображає статичну залежність між об’єктами цих класів.

Агрегація є спеціалізованою формою асоціації типу «має» та описує відношення, за якого один клас містить інший як свою частину. Важливою особливістю агрегації є те, що об’єкти-частини не мають жорсткої залежності від життєвого циклу контейнера і можуть існувати незалежно від нього. Графічно в UML агрегація позначається порожнистим ромбом, з’єднаним із класом-власником. Хоча агрегат складається з кількох окремих об’єктів, на семантичному рівні він розглядається як єдина логічна сутність.

Узагальнення визначає відношення «є різновидом» між класами та вказує на те, що один клас є спеціалізацією іншого. У такому зв’язку підклас наслідує властивості суперкласу. Наприклад, «дуб» є різновидом «дерева», а «автомобіль» — підкласом «транспортного засобу». У UML узагальнення зображається у вигляді лінії з порожнім трикутником, спрямованим до суперкласу. Такі зв’язки також називають відношеннями успадкування або «є».

Зв’язок реалізації в UML вказує на те, що один елемент моделі реалізує функціональність, визначену іншим елементом, зазвичай інтерфейсом. Графічно він позначається порожнім трикутником і використовується переважно на діаграмах класів та компонентів. Для наочного подання реалізації на діаграмах компонентів також застосовується нотація «кульового гнізда».

Залежність є найслабшою формою зв’язку та означає, що один клас тимчасово використовує інший під час виконання певних операцій. Такий зв’язок вказує на потенційний вплив змін одного класу на інший, але не передбачає постійної взаємодії.

На рисунку 2.2 наведено діаграму класів, яка відображає внутрішню структуру клієнтської частини програмного забезпечення.

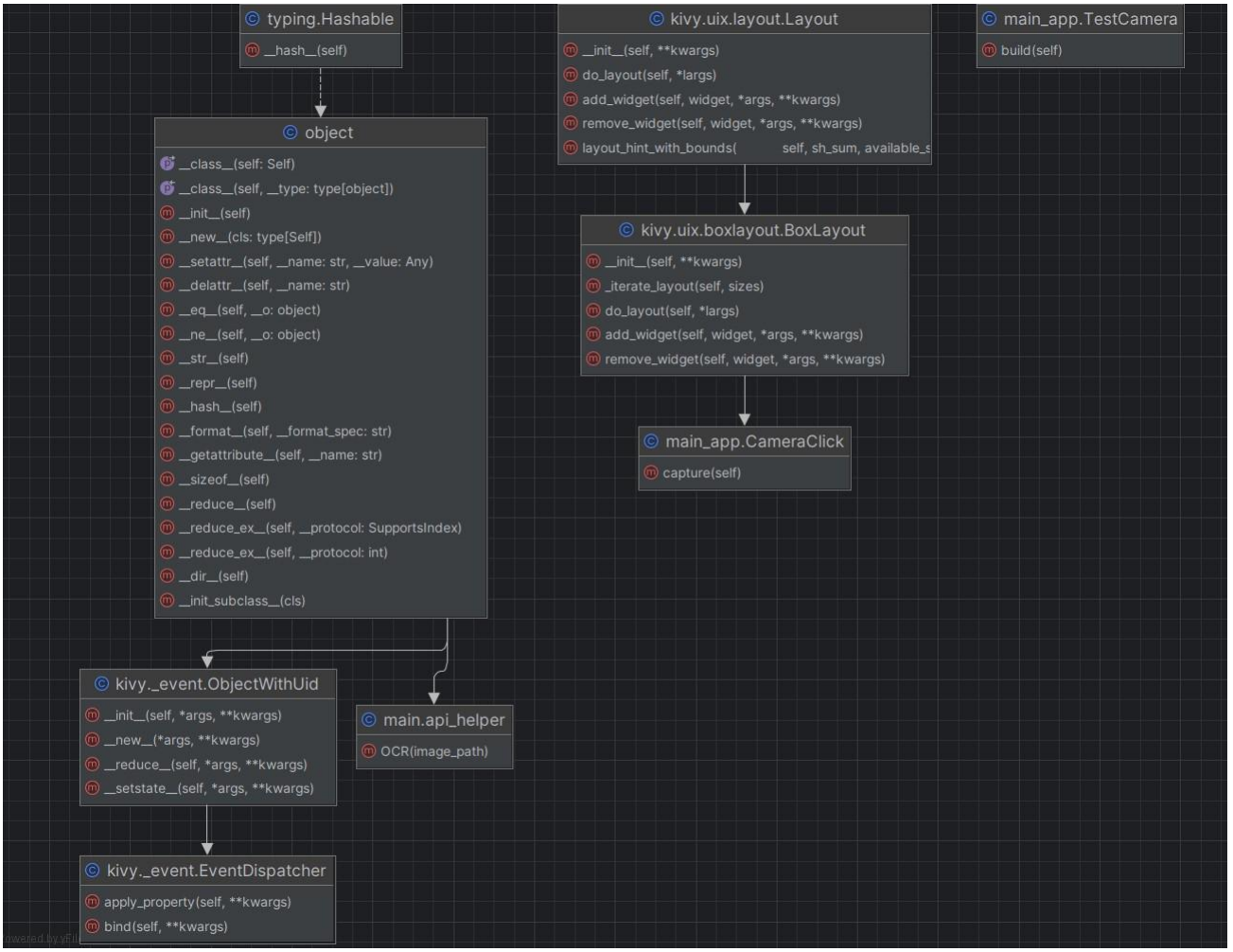

Рисунок 2.2 — Діаграма зв’язків між класами клієнтської частини

Відмінність від асоціації не означає, що залежність одного класу функцій є екземплярами іншого класу. Зв'язки реалізації та взаємодії представлені графічно як лінії, що з'єднують частини, з додатковими символами на кінцях ліній для позначення подальших деталей зв'язку.

Частини тіла в моделюванні представляють інформацію, яку система обробляє, та поведінку, пов'язану з цією інформацією. Графічні зображення частин тіла зазвичай мають форму кола з короткою лінією, прикріпленою до нижньої частини кола, або їх можна намалювати як звичайні частини зі стереотипним «тілом» над назвою частини.

На рисунку 2.3 показано діаграму зв’язків між класами серверної частини.

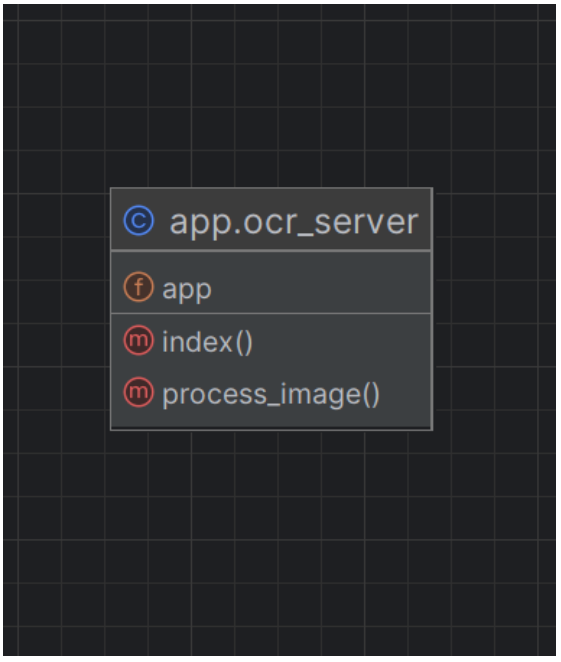

Рисунок 2.3 — Діаграма класів на стороні сервера

EasyOCR застосовує складну нейронну архітектуру, що поєднує декілька типів шарів з метою досягнення високої точності розпізнавання тексту. Ключовими складовими цієї архітектури є згорткові нейронні мережі (CNN), які використовуються для ефективного виділення ознак із зображень, та рекурентні нейронні мережі (RNN), призначені для аналізу та обробки послідовностей символів.

2.8 Архітектура CNN та RNN в EZOCR

Архітектура EasyOCR містить такі основні типи шарів:

— згорткові шари;

— шари підвибірки;

— шари активації;

— шари нормалізації.

Згорткові шари представлені кількома рівнями з різною кількістю фільтрів (наприклад, 32, 64, 128), що дозволяє вилучати ознаки різної складності та абстракції з вхідних зображень.

Шари підвибірки реалізуються у вигляді максимального або середнього об’єднання та застосовуються для зменшення розмірності карт ознак при збереженні найбільш значущої інформації. Це сприяє зниженню обчислювальної складності та підвищенню стійкості моделі до шуму.

Шари активації, зокрема функція ReLU (Rectified Linear Unit), вводять нелінійність у модель, що дає змогу нейронній мережі навчатися складнішим залежностям між вхідними та вихідними даними.

Шари нормалізації використовуються для стабілізації та пришвидшення процесу навчання нейронної мережі. Наприклад, пакетна нормалізація (Batch Normalization) нормалізує вихідні значення шару перед передаванням їх до наступного рівня, що покращує збіжність моделі та підвищує загальну ефективність навчання.

Після етапу вилучення ознак за допомогою згорткових шарів у EasyOCR застосовуються рекурентні нейронні мережі (RNN), призначені для обробки послідовностей символів. Вони дають змогу враховувати контекст та взаємозв’язки між символами, що є критично важливим для точного розпізнавання тексту.

До складу рекурентної частини нейронної мережі EasyOCR входять такі основні шари:

— шари LSTM або GRU;

— шари нормалізації;

— повністю зв’язані шари.

Шари LSTM (Long Short-Term Memory) або GRU (Gated Recurrent Unit) можуть бути реалізовані в одному або кількох рівнях і призначені для збереження інформації про попередні стани під час обробки послідовних даних. Завдяки цьому модель здатна враховувати контекст і залежності між символами, що суттєво підвищує якість розпізнавання тексту.

Шари нормалізації забезпечують приведення вхідних даних до стабільного діапазону значень, що запобігає появі надто великих або надто малих величин. Це спрощує процес навчання нейронної мережі та сприяє підвищенню її стійкості й швидкості збіжності.

Повністю зв’язані шари використовуються на завершальному етапі обробки та відповідають за перетворення вилучених ознак у кінцеві передбачувані сигнали. Вони узагальнюють інформацію з усіх попередніх шарів і формують остаточний результат роботи моделі.

2.9 Вибір технології для реалізації клієнтської та серверної частин

Python — це мова програмування високого рівня, орієнтована на читабельність і простоту коду завдяки використанню логічних відступів та зрозумілого синтаксису. Вона підтримує кілька парадигм програмування, зокрема структурне, об’єктно-орієнтоване та функціональне програмування. Однією з ключових переваг Python є багата стандартна бібліотека, яка надає широкий набір інструментів для розробки без потреби використання сторонніх модулів.

Мову Python було розроблено наприкінці 1980-х років Гвідо ван Россумом як альтернативу мові програмування ABC. Перша публічна версія — Python 0.9.0 — з’явилася у 1991 році. У 2000 році було випущено Python 2.0, який приніс низку важливих удосконалень, зокрема розширені можливості роботи зі списками та механізм автоматичного керування пам’яттю. Останньою версією гілки Python 2 стала Python 2.7.18, підтримку якої офіційно завершено у 2020 році.

У 2008 році було представлено Python 3.0, який став основною та активно підтримуваною версією мови. Вона містила значні зміни, що порушили зворотну сумісність із Python 2, однак ці кроки були необхідні для подальшого розвитку, зокрема покращення роботи з Unicode та усунення архітектурних обмежень попередніх версій. Для спрощення міграції коду з Python 2 на Python 3 була створена спеціальна утиліта 2to3**,** яка автоматизує процес адаптації програм.

Гвідо ван Россум залишався головним ідейним лідером і провідним розробником Python до 2018 року, зробивши значний внесок у розвиток мови та формування її спільноти. У 2019 році управління розвитком мови було передано Раді директорів, до складу якої увійшли провідні розробники Python. У 2020 році ван Россум вирішив не продовжувати участь у роботі ради.

Сьогодні Python залишається однією з найпопулярніших мов програмування у світі завдяки своїй універсальності, широкій екосистемі бібліотек, активній спільноті та підтримці сучасних технологій, зокрема веброзробки, аналізу даних і штучного інтелекту. Після завершення підтримки Python 2 спільнота повністю зосередилася на розвитку Python 3.6 та новіших версій, які регулярно отримують оновлення, виправлення безпеки та функціональні покращення.

Python є багатопарадигмальною мовою програмування, що повністю підтримує об’єктно-орієнтований і структурний підходи, а також надає засоби для функціонального й аспектно-орієнтованого програмування. Мова забезпечує можливості метапрограмування та використання метаоб’єктних механізмів, підтримує динамічне керування пам’яттю, автоматичне збирання сміття та динамічне розв’язання імен.

Python активно використовується і в функціональному програмуванні, пропонуючи такі інструменти, як фільтрація, відображення (map), згортання (reduce) та ітеративні обчислення. Стандартна бібліотека містить модулі itertools і functools, які реалізують функціональні підходи, запозичені з мов Haskell і Standard ML.

Філософію та базові принципи мови узагальнено в документі «Zen of Python» (PEP 20), який визначає ключові ідеї, зокрема простоту, читабельність і однозначність коду.

У процесі розробки програмного забезпечення внутрішню структуру системи зазвичай подають у вигляді діаграм або блок-схем, що відображають організацію модулів і груп компонентів, а також взаємодію між ними. У програмній інженерії відповідно до стандартів Уніфікованої мови моделювання (UML) для цього застосовується діаграма класів — статична структурна діаграма, яка надає узагальнений огляд системи, включаючи її елементи, атрибути, операції та зв’язки між об’єктами.

Діаграми класів є базовими структурними елементами об’єктно-орієнтованого моделювання. Вони застосовуються як для загального концептуального опису структури програмного забезпечення, так і для детальнішого моделювання з метою подальшого перетворення моделі в програмний код. Окрім цього, діаграми класів можуть виконувати роль інструментів моделювання даних, оскільки кожен клас на діаграмі представляє ключові сутності системи та класи, що підлягають реалізації.

Стандартну діаграму класів зазвичай поділяють на три основні частини:

— верхній блок, який містить назву класу, зазвичай виділену жирним шрифтом і розташовану по центру, при цьому перша літера назви пишеться з великої літери;

— середній блок, у якому наведено атрибути класу, вирівняні по лівому краю, з назвами, що починаються з малої літери;

— нижній блок, який містить операції (методи), що може виконувати клас, також вирівняні по лівому краю, з назвами, що починаються з малої літери.

Під час проєктування системи визначення окремих елементів та їх групування на діаграмах допомагає краще зрозуміти статичні зв’язки між компонентами. Для більш глибокого опису системи концептуальне проєктування часто поділяють на кілька підкатегорій і доповнюють іншими видами UML-діаграм.

Асоціація в UML описує зв’язок між елементами моделі. Якщо зміна одного класу може впливати на інший, між ними встановлюється асоціативний зв’язок. Такий зв’язок може бути односпрямованим або двонаправленим і зазвичай зображається лінією зі стрілкою, що вказує напрям взаємодії.

Для розширеного опису поведінки системи діаграми класів можуть доповнюватися діаграмами станів або машинами станів UML. Асоціації мають кілька різновидів, зокрема односпрямовані та двонаправлені, а також спеціалізовані форми, такі як агрегація та композиція. Найпоширенішими на практиці є односпрямовані й двонаправлені асоціації.

Наприклад, зв’язок між класами «Рейс» і «Літак» може бути поданий як двонаправлена асоціація, що відображає статичний взаємозв’язок між об’єктами цих двох класів.

Спорідненість — це один із варіантів зв'язку «має» і є більш специфічним, ніж асоціація. Агрегація є різновидом зв’язку типу «має» та виступає більш спеціалізованою формою асоціації. Вона описує відношення, за якого один клас включає інший як свою складову частину. Наприклад, на діаграмі клас «Професор» пов’язаний із класом «Викладання», що відображає наявність агрегованого зв’язку між цими елементами моделі.Однак важливо зазначити, що агрегація обмежується двосторонніми зв'язками, тобто завжди взаємодією між двома класами. Унікальність збірки полягає в тому, що частини в збірці не мають сильної залежності від життєвого циклу контейнера. Навіть після знищення контейнера його елементи все ще існують.

Одна з унікальних особливостей збірки полягає в тому, що компоненти в збірці не мають сильної залежності від життєвого циклу контейнера. Навіть після знищення контейнера його елементи все ще існують.

Графічно збірка в UML представлена як порожнистий ромб, з'єднаний з класом-господарем лінією. Хоча збірка фізично складається з кількох невеликих об'єктів, її можна розглядати як семантично вищий рівень сутності, яка взаємодіє з іншими об'єктами.

Розглянемо, наприклад, взаємозв’язок між класами «Бібліотекар» та «Студент». У цьому випадку об’єкти класу «Студент» можуть існувати незалежно від об’єктів класу «Бібліотекар», що свідчить про відсутність жорсткої залежності між їхніми життєвими циклами. Такий зв’язок доцільно описувати як агрегацію, оскільки студенти можуть бути пов’язані з бібліотекарями, але не є невід’ємною частиною їхнього існування. Подібний тип зв’язку інколи називають колективним.

Узагальнення в UML описує зв’язок типу «є різновидом» між класами. Воно вказує на те, що один клас (підтип) є більш конкретною формою іншого класу (супертипу). Наприклад, клас «Людина» є підтипом класу «Ссавець», а «Ссавець», у свою чергу, є підтипом класу «Тварина». Графічно зв’язок узагальнення в UML позначається лінією з порожнім трикутником, спрямованим у бік суперкласу.

Відношення узагальнення також відоме як успадкування або зв’язок типу «є». У цьому контексті суперклас може називатися базовим, батьківським класом або базовим типом. Класи, що наслідують його властивості, називають підкласами, похідними або успадкованими класами. Важливо підкреслити, що ці терміни використовуються виключно в межах програмування та моделювання і не мають відношення до біологічних зв’язків.

Таким чином, відношення типу «А є типом В» можна проілюструвати прикладами:
«Дуб є типом дерева», «Автомобіль є підкласом транспортного засобу».

У UML зв’язок реалізації вказує на те, що один елемент моделі (зазвичай клас) реалізує функціональність, визначену іншим елементом, найчастіше інтерфейсом. Графічно цей зв’язок зображається пунктирною лінією з порожнім трикутником, спрямованим до інтерфейсу або абстрактного елемента. На діаграмах компонентів для відображення реалізації також може застосовуватися нотація «кульового гнізда».

Окремо варто зазначити, що філософія проєктування програмного забезпечення в Python базується на принципах Zen of Python, серед яких ключовими є такі положення:

— краса краща за потворність;

— явність краща за неявність;

— простота краща за складність;

— складність краща за заплутаність;

— читабельність має вирішальне значення.

Ці принципи безпосередньо впливають на підходи до моделювання та проєктування програмних систем.

Вони не є винятковими, оскільки порушують певні правила:

— функціональність важливіша за якість;

— шкідники не повинні залишатися непоміченими;

— якщо незрозуміло, чи є помилка, це помилка.

Ці принципи відображають філософію Python як мови, яка зосереджена на читабельності, простоті та зрозумілості кодування.

Python як мова програмування відомий своєю широтою та модульністю, що дозволяє йому легко поєднувати різні інтерфейси програмування та розширювати функціональність існуючих програм. Винахідник Python, Гвідо ван Россум, заклав основу для розвитку мови, зосередившись на читабельності, чіткій структурі синтаксу та простоті вибору методів кодування.

Python має кілька характеристик наведених нижче:

— основні принципи проектування;

— оптимізація та продуктивність;

— спільнота Python (пітоністи);

— основні оператори Python;

Python надає широкий набір базових керувальних конструкцій, серед яких оператор присвоєння =, умовний оператор if з можливістю використання гілок else та elif, цикл for, що застосовується для ітерації по послідовностях, а також цикл while, який виконує блок коду доти, доки умова залишається істинною.

Мова підтримує модульний підхід, що дає змогу легко розширювати або модифікувати функціональність програм шляхом підключення окремих модулів. Одним з основних принципів проєктування Python є орієнтація на читабельність коду, завдяки чому програми легше розуміти, супроводжувати та розвивати.

Філософія Python суттєво відрізняється від підходу, притаманного, наприклад, Perl. Вона ґрунтується на ідеї, що для розв'язання задачі має існувати «один очевидний спосіб». При цьому розробники Python роблять акцент не на передчасній оптимізації, а на зрозумілості та підтримуваності коду. За потреби підвищення продуктивності можуть використовуватися альтернативні інтерпретатори, такі як PyPy, або розширення, написані мовою C.

Однією з цілей створення Python було зробити програмування захопливим і приємним процесом, що відображено навіть у назві мови, натхненній творчістю комедійної групи Monty Python. Завдяки цьому навколо мови сформувалася активна та професійна спільнота користувачів, відома як Pythonists.

Унікальною рисою синтаксису Python є використання відступів для визначення блоків коду, що сприяє його структурованості та підвищує читабельність. Загалом Python вирізняється універсальністю та багатопарадигмальністю, поєднуючи різні стилі програмування та надаючи потужні інструменти для розв'язання широкого кола завдань.

Python також містить велику кількість операторів, які роблять мову гнучкою та виразною. Зокрема, оператор try використовується для перехоплення й обробки винятків, забезпечуючи безпечне виконання програми, а оператор raise дозволяє явно ініціювати винятки. Оператори class та def застосовуються для визначення класів і функцій відповідно, що сприяє структуризації програмного коду.

Оператор with, запроваджений у Python 2.5, спрощує керування ресурсами за допомогою контекстних менеджерів. Оператори break і continue дозволяють керувати виконанням циклів, тоді як del використовується для видалення об'єктів і звільнення пам'яті. Оператор pass слугує заповнювачем порожніх блоків коду, assert — для перевірки умов під час виконання програми, yield —

для реалізації генераторів, а return — для повернення значень із функцій. Оператори імпорту дають змогу підключати зовнішні модулі та використовувати їхню функціональність.

Python характеризується динамічною типізацією, що дозволяє змінним у різні моменти часу посилатися на об’єкти різних типів. Це підвищує гнучкість мови та зменшує потребу в жорсткому визначенні типів, характерному для статично типізованих мов.

Мова не підтримує оптимізацію хвостової рекурсії, однак із версії Python 2.5 генератори отримали розширені можливості обміну даними, а починаючи з Python 3.3 — підтримку передавання інформації через кілька рівнів викликів.

Python має низку синтаксичних особливостей, що відрізняють його від таких мов, як C або Java. Зокрема, для ділення використовуються два оператори: / для дійсного ділення та // для цілочисельного. Оператор ** призначений для піднесення до степеня. Починаючи з Python 3.5, було додано інфіксний оператор @, який переважно використовується для множення матриць у наукових бібліотеках.

У Python 3.8 з’явився оператор присвоєння :=, відомий як «моржовий оператор», який дозволяє виконувати присвоєння безпосередньо у виразах. Мова також підтримує ланцюгові порівняння, умовні вирази виду x if condition else y, спискові включення, генератори, лямбда-вирази та розпакування послідовностей.

Python чітко розрізняє змінні списки та незмінні кортежі, надає різні засоби форматування рядків (оператор %, метод format() та f-рядки), а також широкий набір інструментів для маніпулювання рядковими даними, зокрема конкатенацію, індексацію та різні методи обробки тексту.

Завдяки цим можливостям Python залишається потужним, гнучким і зручним інструментом для розробки програмного забезпечення різного призначення.

Особливості обробки рядків і виразів у Python охоплюють низку важливих механізмів, серед яких зіставлення рядків, використання одинарних і подвійних

лапок, багаторядкові та необроблені рядкові літерали, індексація і зрізи, а також чітке розмежування між інструкціями та виразами.

Конкатенація рядків у Python виконується за допомогою оператора +. Наприклад, вираз "spam" + "egg" утворює рядок "spamegg". Аналогічно, при роботі з числовими значеннями, представленими у вигляді рядків, "2" + "2" дає результат "22", а не арифметичну суму.

Рядкові літерали можуть бути визначені як в одинарних ('), так і в подвійних (") лапках — обидва варіанти є рівнозначними. Це забезпечує зручність при використанні спеціальних символів та екранування за допомогою зворотної скісної риски (\).

Багаторядкові рядки задаються у потрійних одинарних (''') або подвійних (""") лапках. Вони можуть охоплювати кілька рядків тексту та часто застосовуються для створення документаційних рядків (docstrings).

Необроблені (сирі) рядки мають префікс r і трактують символи буквально, без інтерпретації escape-послідовностей. Це особливо корисно при роботі з регулярними виразами або шляхами файлової системи.

Python підтримує індексацію та зрізи рядків і списків. Індексація починається з нуля, при цьому від’ємні індекси дозволяють звертатися до елементів із кінця послідовності. Зрізи мають формат [start:end:step], де start і end визначають межі вибірки, а step — крок між елементами.

Важливою особливістю Python є чітке розділення між інструкціями та виразами, що відрізняє його від мов на кшталт Common Lisp, Scheme або Ruby. Хоча такий підхід може призводити до певного дублювання конструкцій, він робить мову більш структурованою, зрозумілою та передбачуваною.

Завдяки цим характеристикам Python є гнучкою та потужною мовою для роботи з текстовими даними, що дозволяє ефективно виконувати як прості, так і складні операції обробки рядків. Загалом Python — це динамічна об’єктно-орієнтована мова програмування, яка поєднує зручний синтаксис, багаті можливості та розвинену екосистему класів і методів.

Ось огляд деяких основних аспектів Python:

— методи речей;

— письмо та принцип «качки»;

— частини та інструменти;

— старий та новий стиль класної кімнати;

— статична типізація;

— реалізація на CPython;

— розробка на Python.

У Python методи об’єктів є функціями, пов’язаними з класами. Вони отримують явне посилання на екземпляр класу, зазвичай через параметр «self», що дозволяє методам отримувати доступ до атрибутів і стану об’єкта. Запис «instance.method(argument)» є скороченою формою виклику «Class.method(instance, argument)».

Динамічна типізація та принцип «качки»

Python використовує динамічну типізацію та дотримується принципу «duck typing» («якщо щось виглядає як качка і крякає як качка, то це, ймовірно, качка»). Тип об’єкта визначається під час виконання програми, а змінні не прив’язані до конкретних типів даних, що забезпечує гнучкість і спрощує розробку.

Python дозволяє створювати власні класи, на основі яких формуються об’єкти — екземпляри цих класів. Класи також можуть бути метакласами, що відкриває можливості для метапрограмування. У Python 2 існували класи старого та нового стилю, однак у Python 3 усі класи є класами нового стилю та неявно успадковуються від базового класу «object».

Починаючи з Python 3.5, мова підтримує анотації типів, які дозволяють описувати статичні типи змінних і функцій. Хоча стандартна реалізація CPython не перевіряє ці типи під час виконання, для статичного аналізу використовується інструмент mypy.

Офіційною реалізацією мови є CPython, яка написана мовою C та компілює Python-код у байт-код для виконання у віртуальній машині. Розвиток Python

регулюється документами PEP (Python Enhancement Proposal), які описують нові можливості та зміни мови. Основні правила стилю кодування визначені у PEP 8.

Завдяки цим характеристикам Python є гнучкою та потужною мовою програмування, що підтримує як об’єктно-орієнтований, так і функціональний підходи, і може ефективно застосовуватися у широкому спектрі завдань.

PyCharm — це інтегроване середовище розробки (IDE) для мови Python, створене компанією JetBrains. Воно надає потужні інструменти для написання, налагодження та тестування програм, що робить його одним із найпопулярніших IDE серед Python-розробників.

Однією з ключових переваг PyCharm є розвинена система підказок і аналізу коду, яка допомагає виявляти помилки та оптимізувати програму. Редактор коду підтримує автодоповнення, підсвічування синтаксису та автоматичне форматування, що значно спрощує розробку.

PyCharm також містить візуальний налагоджувач, який дозволяє покроково відстежувати виконання програми та аналізувати її стан у режимі реального часу. Крім того, середовище підтримує роботу з віртуальними середовищами та керування залежностями проєкту за допомогою таких інструментів, як «pipenv».

«Flask» — це легкий та гнучкий мікрофреймворк для веб-розробки на Python. Він орієнтований на мінімалізм і дозволяє швидко створювати веб-додатки та REST-сервіси без зайвих накладних витрат.

Flask має невелику кодову базу та мінімальну кількість залежностей, що робить його ідеальним для прототипування та невеликих проєктів.
Функціональність фреймворку легко розширюється за допомогою додаткових модулів, таких як Flask-SQLAlchemy для роботи з базами даних або Flask-WTF для обробки веб-форм.

Вбудований шаблонізатор Jinja2 дозволяє створювати динамічний HTML-контент, використовуючи умовні конструкції, цикли та інші логічні елементи. Flask також забезпечує зручну роботу з HTTP-запитами та відповідями, що спрощує реалізацію серверної логіки.

Kivy — це фреймворк з відкритим кодом для розробки мультимедійних та кросплатформних додатків мовою Python. Він дозволяє створювати програми, що працюють на Windows, macOS, Linux, Android та iOS.

Однією з ключових особливостей Kivy є підтримка мультисенсорного введення, що робить його придатним для розробки застосунків для сенсорних пристроїв, таких як смартфони та планшети. Фреймворк містить графічні та аудіомодулі, які дозволяють створювати анімації, візуальні ефекти та працювати зі звуком.

Kivy надає набір стандартних елементів інтерфейсу користувача, таких як кнопки, текстові поля та списки, а також дозволяє створювати власні віджети. Завдяки відкритому коду та активній спільноті фреймворк постійно розвивається та вдосконалюється.

Отже, Kivy є потужним інструментом для створення кросплатформних мультимедійних застосунків на Python, що поєднує гнучкість, продуктивність і зручність розробки.

3 РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ РОЗПІЗНАВАННЯ ТЕКСТУ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

3.1 Розробка графічного інтерфейсу

Графічний інтерфейс користувача (GUI) — це тип інтерфейсу, який забезпечує взаємодію користувача з електронними пристроями через візуальні елементи, зокрема кнопки, іконки, меню та підказки, без необхідності вводити текстові команди з клавіатури. GUI широко застосовується в сучасних програмах і пристроях та відіграє ключову роль у підвищенні зручності використання, інтуїтивності та загальної ефективності роботи користувача.

Ключові характеристики GUI:

— графічні елементи;

— пряме підключення;

— розширені можливості;

— широке використання;

— адаптація до потреб споживачів;

— зображення.

GUI базується на використанні візуальних елементів, таких як кнопки, вікна, списки, полотна та інші компоненти, з якими користувач може безпосередньо взаємодіяти за допомогою миші або сенсорного екрана.

Пряма взаємодія полягає в тому, що користувачам не потрібно вводити текстові команди — вони можуть натискати кнопки, перетягувати об’єкти та виконувати інші дії інтуїтивно, використовуючи графічні елементи керування.

Розширені можливості графічного інтерфейсу дозволяють створювати складні та функціональні інтерфейси з різноманітними графічними компонентами, що значно спрощує та покращує взаємодію користувача з програмою або пристроєм.

Широкий спектр використання — графічні інтерфейси користувача використовуються не лише в комп’ютерах, а й у різних інших пристроях, таких як

смартфони, планшети, ігрові консолі, побутова та промислова техніка, автомобільні системи тощо.

Зручний для користувача — дизайн графічного інтерфейсу користувача орієнтований на користувача та спрямований на спрощення використання, підвищення ефективності та дотримання принципів зручності використання.

Розробка графічної інтеграції та часової поведінки графічного інтерфейсу користувача є важливою частиною програм, спрямованих на взаємодію людини з комп’ютером.

Ось переформульований та стилістично відредагований варіант тексту:

Усі перелічені можливості роблять графічний інтерфейс користувача важливим засобом створення зручних і зрозумілих інтерфейсів для різноманітних програмних продуктів і пристроїв. Розглянемо основні переваги та недоліки графічного інтерфейсу користувача.

Переваги графічного інтерфейсу користувача:

— психологічна привабливість;

— зручність використання;

— наочність графічних елементів;

— підтримка мультимедіа.

Графічний інтерфейс користувача ґрунтується на використанні візуальних компонентів, таких як значки, кнопки та меню, що робить його більш доступним і зрозумілим для користувачів із різним рівнем технічної підготовки. Візуальне подання інформації дозволяє швидко орієнтуватися у функціях та можливостях системи.

GUI забезпечує інтуїтивно зрозумілу взаємодію з комп’ютером, даючи змогу виконувати основні дії за допомогою клацань миші, перетягування об’єктів або введення даних із клавіатури.

Візуалізація є однією з ключових переваг графічного інтерфейсу, оскільки інформація подається у вигляді графіків, діаграм, таблиць та інших наочних елементів. Це сприяє швидшому сприйняттю, аналізу та розумінню складних даних.

Мультимедійна підтримка дозволяє інтегрувати зображення, відео та аудіо, створюючи більш інтерактивні та привабливі інтерфейси, що значно покращує користувацький досвід.

Інтерфейс користувача розробленого застосунку складається з одного основного вікна (рисунок 3.1).

Рисунок 3.1 — Інтерфейс клієнтського додатку

3.2 Розробка логіки клієнтської програми

Після запуску клієнтського застосунку користувач отримує доступ до головного вікна програми. Головне вікно містить зображення з камери пристрою та кнопку для запуску процесу розпізнавання. Компонент камери налаштований таким чином, що одразу відображає поточне зображення без необхідності

виконання додаткових налаштувань. Реалізацію цього інтерфейсу наведено в лістингу 3.1.

Лістинг 3.1 — Реалізація інтерфейсу з камерою

```
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()
  ''')
```

Після натискання кнопки розпізнавання система зберігає поточне зображення з камери та викликає метод взаємодії з API. Після обробки отриманих даних автоматично відтворюється аудіодоріжка, згенерована на основі розпізнаного тексту. Реалізація цього механізму представлена в лістингу 3.2.

Лістинг 3.2 — Реалізація виклику API та відтворення аудіо

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

Продовження лістингу 3.2

```
= "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 перетворює отримане зображення у байтовий масив, після чого кодує його у формат Base64 та формує JSON-об’єкт. Даний об’єкт передається на сервер за допомогою HTTP-запиту, як показано в 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 відповіді від сервера застосунок створює аудіодоріжку на основі розпізнаного тексту та зберігає її у файл. Реалізація цього етапу наведена в лістингу 3.4.

Лістинг 3.4 — Генерація та збереження аудіодоріжки

```
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")
```

Усі файли, створені під час роботи застосунку, зберігаються тимчасово та є енергозалежними, тобто не накопичуються постійно в пам’яті пристрою. Це дозволяє зменшити використання дискового простору та підвищити ефективність роботи програми.

4 ТЕСТУВАННЯ РОЗРОБКИ

4.1 Тестування програмного забезпечення

Тестування програмного забезпечення є одним із ключових етапів процесу розробки будь-якого програмного продукту. Воно передбачає запуск і перевірку програми з метою виявлення помилок, забезпечення коректної роботи та стабільності функціонування за різних умов експлуатації. Тестування є невід'ємною складовою життєвого циклу програмного забезпечення, оскільки дозволяє підвищити якість і надійність продукту, зміцнити довіру користувачів, зменшити ризики під час використання та забезпечити ефективне впровадження застосунку.

Причини важливості тестування

— забезпечення якості та надійності програмного забезпечення;

— покращення взаємодії з користувачем;

— зменшення витрат на розробку та супровід;

— відповідність стандартам і технічним вимогам.

Забезпечення якості та надійності полягає у перевірці безпомилкової роботи програмного продукту та його відповідності заявленим вимогам і специфікаціям. Це особливо важливо для критично важливих систем, у яких помилки можуть спричинити значні фінансові втрати або створити загрозу безпеці та життю людей.

Покращення взаємодії з користувачем досягається завдяки своєчасному виявленню дефектів, які можуть негативно впливати на зручність використання, призводити до втрати даних або збоїв у роботі системи. Тестування дозволяє усунути такі проблеми ще до випуску продукту.

Зниження витрат на розробку пояснюється тим, що виправлення помилок на ранніх етапах створення програмного забезпечення є значно дешевшим, ніж після його впровадження. Несвоєчасне виявлення дефектів може призвести до додаткових витрат і погіршення репутації розробника.

Відповідність стандартам та вимогам забезпечується шляхом перевірки програмного забезпечення на відповідність нормам якості, безпеки та очікуванням замовника або ринку.

4.2 Види тестування

Тестування програмного забезпечення може здійснюватися в різних формах, зокрема:

— функціональне тестування;

— навантажувальне тестування;

— тестування безпеки;

— автоматизоване тестування.

Функціональне тестування спрямоване на перевірку правильності виконання всіх заявлених функцій програми. Воно охоплює тестування окремих модулів, таких як алгоритми аналізу URL-адрес або обробки вмісту веб-сторінок, а також перевірку їх інтеграції в єдину систему.

Навантажувальне тестування дозволяє оцінити роботу програми за умов підвищеного навантаження. Його метою є визначення граничних можливостей системи та здатності обробляти велику кількість запитів одночасно шляхом моделювання роботи багатьох користувачів.

Тестування безпеки спрямоване на виявлення вразливостей, які можуть бути використані для несанкціонованого доступу або витоку даних. Воно включає аналіз програмного коду, тестування на проникнення та перевірку відповідності вимогам інформаційної безпеки.

Автоматизоване тестування використовує спеціалізовані інструменти для автоматичного виконання тестів, що дозволяє значно прискорити процес перевірки та зменшити ймовірність людської помилки. Такі тести можуть бути заплановані для регулярного запуску та охоплювати різні аспекти роботи програми, включаючи функціональність, продуктивність і безпеку.

4.3 Тестування під час розробки програми для виявлення фішингу

У межах розробки програми для виявлення фішингових атак тестування включає перевірку коректності роботи алгоритмів аналізу URL-адрес і вмісту веб-сторінок, а також тестування графічного інтерфейсу користувача. Особлива увага приділяється аспектам безпеки, щоб гарантувати відсутність вразливостей, які можуть бути використані зловмисниками.

Отже, тестування є ключовим елементом процесу розробки програмного забезпечення, оскільки воно спрямоване не лише на виявлення та усунення помилок, а й на забезпечення високого рівня якості, надійності та безпеки програмного продукту.

У таблиці 4.1 представлені тестові випадки, проведені під час дослідження системи.

Таблиця 4.1 — Результати проведення проведених тестів

№	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.	Зображеннякамери завмирає, постерігається нормальна затримка до 5-7 секунд, після чого починається ідтворення звукової доріжки, в якій промовляється розпізнаний текст.

Продовження таблиці 4.1 — Результати проведення проведених тестів

№	Назва	Очікуваний результат	Отриманий результат
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

Аналіз результатів тестування показав відсутність функціональних неточностей у роботі системи та підтвердив коректне виконання всіх вбудованих функцій.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нової адаптовані нейромережеві технології для людей з обмеженими можливостями. Особливістю розробки є удосконалений метод розпізнавання тексту з фотографій та його озвучення на клієнтському пристрої на основі нейронної мережі з допомогою комп'ютерного зору.

Аналогом розробки може бути SuperNova Access Suite – 1450 $ (≈ 60000 грн).

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 5.1.

Таблиця 5.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження табл. 5.1

Ринкові переваги					
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною .	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	3	4
Наявність аналогів на ринку	3	3	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	3	4	3
Ринок збуту	4	3	3
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	3	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	3	3
Супровідна документація	3	3	4
Сума	41	40	42
Середньоарифметична сума балів	(41+40+42) / 3 = 41		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Таблиця 5.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 - 1 0	Низький
1 1 - 2 0	Нижче середнього
2 1 - 3 0	Середній
3 1 - 4 0	Вище середнього
4 1 - 4 8	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, оскільки розробка інформаційної системи для людей з обмеженими можливостями, здатної розпізнавати та озвучувати текст 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. Також високий рівень комерційного потенціалу розроблюваного нового програмного продукту досягається за рахунок того, що використовуються удосконалено метод розпізнавання тексту з фотографій та його озвучення на клієнтському пристрої на основі нейронної мережі, які дозволяють підвищити точність такого розпізнавання у порівнянні з аналогами.

5.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

5.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t \,, \tag{5.1}$$

де М — місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ — число робочих днів за місяць, 22 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.4.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	52000	2363,64	32	75636,364
Програміст	48000	2181,82	32	69818,182
Всього				145454,55

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

5.2.2 Додаткова заробітна плата розробників, які брати участь в розробці обладнання/програмного продукту.

Додаткову заробітну плату прийнято розраховувати як 12 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 12\ \% / 100\ \% \qquad (5.2)$$

$$З_д = (145454,55 \cdot 12\ \% / 100\ \%) = 17454,55 \ \text{(грн.)}$$

4.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату (ЄСВ) складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (5.3)$$

$Н_з$ = (145454,55 + 17454,55) · 22 % / 100 % = 35840,00 (грн.)

5.2.4. Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

5.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \quad (5.4)$$

де Ц — балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років;

$t_{вик}$ — термін використання під час розробки, місяців.

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 32300 грн., термін його корисного використання згідно податкового законодавства — 2 роки, а термін його фактичного використання – 1,45 міс.

$$А_{обл} = \frac{32300}{2} \times \frac{1,45}{12} = 1957,576 \text{ грн.}$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 4.5. Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $В_{нем.ак.}$ = 8000 грн. Вартість спеціалізованих

ліцензійних нематеріальних активів (Python, Kivy, Flask) є безкоштовною.

Таблиця 4.5 — Амортизаційні відрахування на матеріальні та нематеріальні ресурси для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія (COBRA Advanced, MSI PRO MP2412, XFX AMD Radeon RX 590 8Gb GME Black Wolf)	32300	2	1,45	1957,576
Офісне обладнання (меблі)	25000	4	1,45	757,576
Приміщення	1150000	20	1,45	6969,697
Всього				9684,85

5.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (5.5)$$

де В — вартість 1 кВт-години електроенергії для малих непобутових споживачів, електроустановки яких приєднані до електричних мереж АТ «Вінницяобленерго», 2 класу підприємства (напруга до 27,5 кВт/год) з ПДВ в

2025 році за даними Енера-Вінниця, В = (9677,45/1000)*1,2 = 11,613 грн./кВт [https://vin.enera.ua/el/tariff];

П — встановлена потужність обладнання, кВт. П = 0,4 кВт;

Ф — фактична кількість годин роботи обладнання, годин;

$К_п$ — коефіцієнт використання потужності, $К_п$ = 0,9.

$$В_е = 0,9 \cdot 0,4 \cdot 8 \cdot 32 \cdot 11,613 = 1070,2541 \text{ (грн.)}$$

5.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.6)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

$$І_в = 145454,55 * 75\% / 100\% = 109090,9 \text{ (грн.)}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$H_{нзв} = (З_о + З_р) \cdot \frac{H_{нзв}}{100\%}, \quad (5.7)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$Н_{нзв} = 145454,55 * 130\ \% / 100\ \% = 189091 \text{ (грн.)}$$

5.2.9 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$В_{заг}$ = 145454,55+17454,55+35840,00+9684,85+8000+1070,25+109090,9+
+189091 = 515686,01 грн.

5.2.11 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \quad (5.8)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7;

впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$$ЗВ = 515686,01 / 0,5 = 1031372 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

— вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

— зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

— кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов'язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об'єднати, враховуючи визначені ситуації з відповідними умовами.

5.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.9)$$

де $\pm\Delta Ц_о$ — зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N — кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_о$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_о = Ц_б \pm \Delta Ц_о$;

Цб – вартість програмного продукту у році до впровадження результатів розробки;

ΔN — збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

р — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2025 році ϑ =18%.

Припустимо, що при прогнозованій ціні 12800 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 1000 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 10000 шт., протягом другого року – на 7000 шт., протягом третього року на 5000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П_1 = (0*1000 + (12800 + 1000)*10000)* 0,8333* 0,21) * (1 - 0,18) = 18367999,265 \text{ грн.}$$

$$\Delta П_2 = (0*1000 + (12800 + 1000)*(10000+7000)* 0,8333* 0,21) * (1 - 0,18) = 33665098,653 \text{ грн.}$$

$$\Delta П_3 = (0*1000 + (12800 + 1000)*(10000+7000+5000)* 0,8333* 0,21) * (1 - 0,18) = 43566598,257 \text{ грн.}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 95599696,18 грн.

5.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.10)$$

де — $\Delta П_i$ збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т — період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}05 \ldots 0{,}15$;

t — період часу (в роках).

Збільшення прибутку ми отримаємо, починаючи з першого року:

$$ПП = (18367999{,}265/(1+0{,}1)^1)+(33665098{,}653/(1+0{,}1)^2)+(43566598{,}257/(1+0{,}1)^3) = 16698181{,}15 + 27822395{,}58 + 32732230{,}1 = 77252806{,}83 \text{ грн.}$$

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (5.11)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$=2...5, але може бути і більшим;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 1031372 = 2062744{,}05 \text{ грн.}$$

Тоді абсолютний економічний ефект $Е_{абс}$ або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{абс} = ПП – PV, \quad (5.12)$$

$Е_{абс} = 77252806,83 – 2062744,05 = 75190062,78$ грн.

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.13)$$

T — життєвий цикл наукової розробки, роки.

$Е_в = \sqrt[3]{(1 + 75190062,78/2062744,05} - 1 = 2,346$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.14}$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2024 році в Україні d = (0,09…0,14);

f — показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0,14 + 0,05 = 0,19.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{5.15}$$

$T_{ок} = 1 / 2,346 = 0,43$ р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 0,43 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 1031372 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 0,43 роки.

ВИСНОВОК

У ході виконання магістерської кваліфікаційної роботи було розроблено програмний додаток для розпізнавання тексту з фотографій та його подальшого озвучення на клієнтському пристрої. Основною метою створення додатка є полегшення доступу до інформації для людей з інвалідністю, зокрема для незрячих та слабозорих користувачів.

У першому розділі розглянуто теоретичні аспекти та сучасний стан досліджуваної теми, зокрема питання машинного навчання, розпізнавання образів, оптичного розпізнавання символів, проведено аналіз існуючих рішень та сформульовано основну проблему дослідження.

У другому розділі здійснено проєктування програмного продукту, включаючи розробку архітектури та життєвого циклу системи, визначення функціональних можливостей додатка та опис його внутрішньої структури.

У третьому розділі описано процес практичної реалізації програмного продукту, зокрема обґрунтовано вибір інструментів розробки, розроблено графічний інтерфейс користувача, реалізовано логіку клієнтської та серверної частин, а у четвертому — проведено системне тестування.

Таким чином, розроблений додаток забезпечує можливість швидкого та зручного отримання текстової інформації з зображень шляхом її аудіовідтворення, що сприяє підвищенню рівня незалежності та соціальної інтеграції незрячих і слабозорих людей. Отримані результати можуть бути використані для подальшого розвитку та вдосконалення програмних засобів у сфері доступності інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

2. Основи програмування мовою Python : навчальний посібник / Селіверстов Р. Г., Мельничин А. В. – Львів : ЛНУ імені Івана Франка, 2020. – 190 с.

3. Фесенко М.А., Кисіль Т.М., Чичкарьов Є.А., Звенігородський О.С.. «Штучні нейронні мережі». - 2023. https://duikt.edu.ua/ua/lib/1/category/2658/view/1676

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

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

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

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

8. 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.

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

10. 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.

11. 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.

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

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

14. 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.

15. 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.

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

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

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

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

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

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

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

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

24. "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.

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

ДОДАТОК А

Технічне завдання
Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ проф., д.т.н. Азаров О. Д.

«25» вересня 2025р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

«Адаптовані нейромережеві технології для людей з обмеженими можливостями»

Науковий керівник:
доцент к.т.н.
Колесник І.С.
Студент групи:
1КІ-24м Поташна К.Я.

ВНТУ 2025

1 Підстава для виконання магістерської кваліфікаційної роботи.

Підставою для виконання магістерської кваліфікаційної роботи є необхідність розробки адаптованих нейромережевих технологій для людей з обмеженими можливостями, що забезпечать їм розпізнавання тексту та його озвучування за допомогою нейронних мереж.

2 Мета і призначення МКР

Метою даної магістерської кваліфікаційної роботи є створення адаптованих нейромережевих технологій для людей з обмеженими можливостями.

3 Вихідні дані для виконання МКР

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 МКР

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 МКР та очікувані результати

Етапи роботи та очікувані результати приведено в таблиці А.1.

Таблиця А.1 — Етапи МКР

№ з/п	Назва етапу	Термінвиконання		Очікувані результати
		початок	кінець	
1	Аналіз поставленого задачі	26.09.25	30.09.25	Огляд джерел, висновки із взаємодії викладачів та студентів
	Огляд існуючих програмних рішень та аналіз літературних джерел	06.10.25	15.10.25	Розділ 1
4	Вибір технології для клієнтської та серверної частини	16.10.25	31.10.25	Розділ 2
5	Розробка клієнтської та серверної частини	01.11.25	17.11.25	Розділ 3
6	Тестування розробки	18.11.25	20.11.25	Розділ 4
7	Оформлення пояснювальної записки та презентації	21.11.25	01.11.25	ПЗ, графічний матеріал, презентація

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

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

8 Вимоги до оформлення та порядок виконання МКР

При оформлюванні МКР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— 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
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Адаптовані нейромережеві технології для людей з обмеженими можливостями

Тип роботи: ____ магістерська кваліфікаційна робота _____

(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ Кафедра обчислювальної техніки. Факультет інформаційних технологій та комп'ютерної інженерії. Група 1КІ-24м

(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) __14___%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Експертна комісія:

Азаров О.Д., завідувач кафедри ОТ _____
(прізвище, ініціали, посада) (підпис)

Мартинюк Т.Б, гарант освітньої програми КІ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ _____Захарченко С.М___
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ к.т.н., доц., доцент кафедри ОТ Колесник І.С.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ _____Поташна К.Я._____
(підпис) (прізвище, ініціали)

ДОДАТОК В

Схема взаємодії користувача з додатком

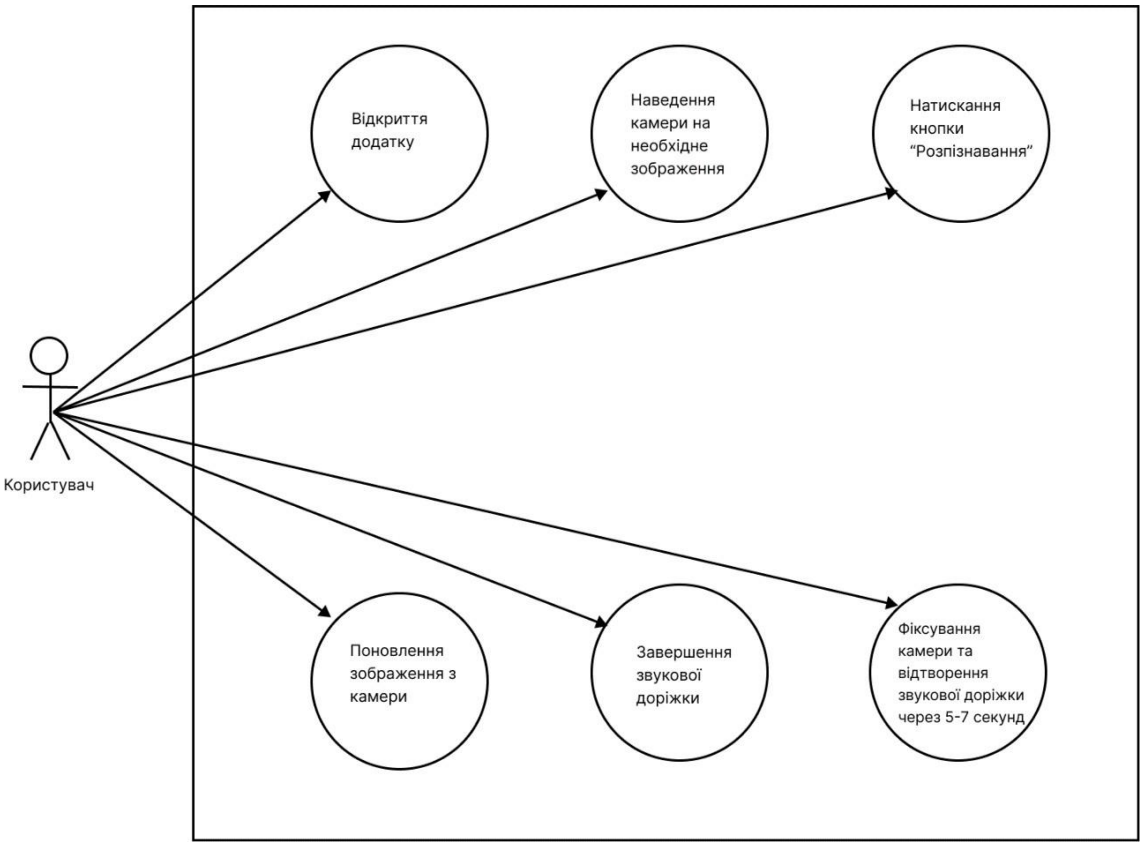

Рисунок В.1 — Блок-схема взаємодії користувача з додатком

ДОДАТОК Г

Схема обробки запиту на розпізнавання тексту

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

ДОДАТОК Д

Код програми

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)
```