Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

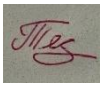
КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Розробка розпізнавача текстів на основі нейронних мереж
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-4
Герман Тесленко
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник ст. викл. Тетяна Мірошниченко
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ  Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Тесленку Герману Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка розпізнавача текстів на основі нейронних мереж

затверджена наказом університету від 19 травня 20 25 р. № 378Ст
2. Термін подання студентом роботи до екзаменаційної комісії 25 червня 20 25 р.
3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проєктів, Python documentation, набір даних для тренування та тестування системи

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі
2) Постановка задачі
3) Розробка розпізнавача текстів

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	26.04.2025	виконано
3	Аналіз методів вирішення поставленої задачі	02.05.2025	виконано
4	Вибір методу вирішення задачі	07.05.2025	виконано
5	Проєктування архітектури системи	12.05.2025	виконано
6	Розробка основної моделі навчання	24.05.2025	виконано
7	Розробка телеграм-бота	01.06.2025	виконано
8	Тестування, доопрацювання системи	07.06.2025	виконано
9	Підготовка та написання пояснювальної записки	10.06.2025	виконано
10	Перевірка на академічний плагіат	20.06.2025	виконано
11	Захист перед ЕК	25.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ст. викл. Тетяна Мірошниченко
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 62 с., 12 рис., 3 табл., 1 дод., 20 джерел.

ГЛИБИННЕ НАВЧАННЯ, КОМП'ЮТЕРНЕ БАЧЕННЯ, МАШИННЕ НАВЧАННЯ, НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ЗОБРАЖЕНЬ, РОЗПІЗНАВАННЯ СИМВОЛІВ, СЕГМЕНТАЦІЯ, ТЕКСТОВА ІНФОРМАЦІЯ, ШТУЧНИЙ ІНТЕЛЕКТ.

Об’єкт дослідження – процес розпізнавання текстової інформації з різних типів зображень за допомогою нейромережевих технологій. Дослідження актуальне через зростаючу потребу перетворення візуальної текстової інформації у машино зчитуваний формат для подальшої обробки.

Предмет дослідження – архітектури нейронних мереж для оптичного розпізнавання символів, включаючи згорткові мережі для виділення графічних ознак, рекурентні мережі для аналізу символів та гібридні моделі. Особлива увага приділяється методам попередньої обробки зображень..

Мета роботи – аналіз існуючих методів розпізнавання текстів з використанням нейронних мереж та формування вимог до майбутньої системи розпізнавання.

Методи дослідження – аналіз наукових джерел, узагальнення відомостей з публікацій у сфері штучного інтелекту та комп'ютерного зору, систематизація знань з нейронних мереж. Проведено порівняння ефективності різних підходів до побудови систем оптичного розпізнавання символів та концептуальне моделювання програмного продукту.

ABSTRACT

Bachelor’s thesis contains: 62 pp., 12 fig., 3 tabl., 1 ann., 20 references.

ARTIFICIAL INTELLIGENCE, CHARACTER RECOGNITION, COMPUTER VISION, DEEP LEARNING, IMAGE PROCESSING, MACHINE LEARNING, NEURAL NETWORKS, SEGMENTATION, TEXTUAL INFORMATION.

Object of the research – the process of recognizing textual information from various types of images using neural network technologies. This research is relevant due to the increasing demand for converting visual textual content into machine-readable formats for further processing.

Subject of the research – neural network architectures for optical character recognition, including convolutional networks for extracting graphical features, recurrent networks for analyzing sequences of characters, and hybrid models. Special attention is given to image preprocessing techniques.

Purpose of the work – to analyze existing text recognition methods based on neural networks and to define the requirements for a future recognition system.

Research methods – analysis of scientific literature, generalization of information from publications in the fields of artificial intelligence and computer vision, and systematization of knowledge on neural networks. A comparative evaluation of the effectiveness of various approaches to building optical character recognition systems was conducted, along with conceptual modeling of the software solution.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект;

Aiogram – асинхронний фреймворк для створення Telegram-ботів;

API – Application Programming Interface – програмний інтерфейс додатків;

CER – Character Error Rate – частка помилок на рівні символів;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

CRNN – Convolutional Recurrent Neural Network – згортково-рекурентна нейронна мережа;

CTC – Connectionist Temporal Classification – послідовне кодування з підключеним вирівнюванням;

CV – Computer Vision – комп’ютерне бачення;

GPU – Graphics Processing Unit – графічний процесор;

LSTM – Long Short-Term Memory – мережа з довгою короткочасною пам’яттю;

OCR – Optical Character Recognition – оптичне розпізнавання символів;

PIL – Python Imaging Library – бібліотека обробки зображень у Python;

Python – мова програмування;

ReLU – Rectified Linear Unit – випрямлена лінійна одиниця (функція активації);

Telegram-бот – автоматизований користувач месенджера Telegram;

Torch – бібліотека для глибокого навчання;

WER – Word Error Rate – частка помилок на рівні слів.

ВСТУП

Сьогодні, коли світ інформаційних технологій розвивається семимильними кроками, перед нами постають цікаві виклики з перетворення величезних масивів неструктурованих даних у придатний для аналізу формат. Одним із таких викликів є проблема ефективного розпізнавання текстової інформації на зображеннях – будь то відскановані сторінки книжок, фотографії документів чи навіть рукописні нотатки. Численні користувачі різноманітних застосунків для розпізнавання тексту стикаються з ситуаціями, коли наявні рішення «губляться» при роботі із зображеннями низької якості або текстами зі складним форматуванням.

Ці спостереження підтверджуються й статистикою: за даними досліджень, навіть провідні комерційні OCR-системи припускаються помилок у 5–15% випадків залежно від складності вхідних даних.

Це вражаюче число, особливо коли йдеться про обробку юридичних чи медичних документів, де точність критично важлива. Водночас, поява революційних технологій глибинного навчання відкрила нову сторінку в підході до розпізнавання тексту – нейронні мережі продемонстрували здатність «бачити» текст майже так само, як люди, враховуючи контекст і навіть передбачаючи слова в умовах часткового пошкодження.

Упродовж останнього десятиліття архітектури нейронних мереж пережили справжній бум розвитку – від відносно простих згорткових мереж до складних багаторівневих моделей із механізмами уваги.

Ця еволюція значно розширила можливості програмних систем розпізнавання, дозволивши їм працювати з нерівномірним освітленням, різноманітними шрифтами та навіть частково перекритими символами. Саме зараз настав час переосмислити підходи до створення OCR-систем, враховуючи накопичений досвід та нові можливості нейромережевих технологій.

Мотивація до виконання цієї роботи виходить далеко за межі суто академічного інтересу. Розпізнавання текстів – це важливий міст між аналоговим минулим та цифровим майбутнім. Багато організацій стикаються з проблемою «паперового спадку» – архівів документів, які потребують переведення в електронну форму. Крім того, існує величезна кількість повсякденних ситуацій, де автоматичне розпізнавання тексту може значно спростити життя: від сканування візитівок до перекладу вивісок чи меню під час подорожей.

Метою даної роботи є проведення комплексного аналізу існуючих нейромережевих підходів до розпізнавання текстової інформації та формування концептуальної основи для майбутньої програмної системи. Дослідження спрямоване не просто на узагальнення відомих методів, а на критичну оцінку їхніх переваг та недоліків у контексті різних типів завдань. Хоча в рамках передатестаційної практики не планується безпосередня розробка програмного забезпечення, результати дослідження створять підґрунтя для подальшої роботи над повноцінною системою розпізнавання тексту.

Озираючись назад, цікаво відзначити, як системи OCR пройшли шлях від спеціалізованих пристроїв, що могли розпізнавати лише певні шрифти, до універсальних програмних рішень, здатних інтерпретувати різноманітні тексти багатьма мовами. Цей прогрес відображає загальну тенденцію розвитку штучного інтелекту – від жорстко запрограмованих алгоритмів до самонавчальних систем, що здатні виявляти приховані закономірності у даних та адаптуватися до нових умов.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Сутність та призначення систем розпізнавання тексту

Системи розпізнавання тексту, або OCR (Optical Character Recognition), відіграють ключову роль у перетворенні інформації з аналогового формату в цифровий. У сучасному інформаційному суспільстві значна частина даних усе ще зберігається в друкованому вигляді – у вигляді книжок, журналів, документів, чеків, банківських виписок, етикеток, технічної документації тощо. Для ефективної обробки, зберігання та пошуку цих даних виникає необхідність автоматичного перетворення зображень із текстовим вмістом у редагований і машиночитаний формат. Саме цю функцію виконують OCR-системи.

Сутність технології OCR полягає в тому, щоб за допомогою алгоритмів комп’ютерного зору та штучного інтелекту автоматично аналізувати зображення, виділяти на них області з текстом, ідентифікувати символи та формувати з них осмислені слова і фрази. Принцип роботи OCR-систем базується на етапах попередньої обробки зображень (виправлення перекосів, фільтрація шумів, нормалізація освітлення), сегментації символів, їх розпізнаванні та післяобробці з метою покращення точності розпізнаного тексту.

На ранніх етапах розвитку такі системи здебільшого були обмежені можливістю розпізнавання лише друкованого тексту з чітко визначеним шрифтом. Сучасні OCR-системи, що базуються на нейронних мережах і глибинному навчанні, мають значно ширші можливості. Вони здатні адаптуватися до різних шрифтів, стилів написання, орієнтації тексту, а також розпізнавати як друкований, так і рукописний текст. Завдяки використанню згорткових нейронних мереж (CNN) стало можливим ефективно витягати ознаки зі зображень, а з використанням рекурентних мереж (RNN, LSTM, GRU) – аналізувати послідовність символів у часі, що

особливо важливо для розпізнавання тексту в контексті. Подальшим удосконаленням стало застосування механізму CTC (Connectionist Temporal Classification), який дозволив тренувати моделі без необхідності вручну сегментувати кожен символ на зображенні.

Системи розпізнавання тексту мають широкий спектр застосування. В адміністративних установах вони використовуються для оцифрування архівів та ділової документації. У фінансовій сфері – для обробки платіжних документів, квитанцій, банківських виписок. У сфері охорони здоров'я – для сканування та обробки медичних карток, рецептів і результатів аналізів. У науковій та освітній діяльності OCR використовується для цифрового представлення книг, статей та інших матеріалів, що значно полегшує роботу дослідників і студентів. Мобільні додатки, такі як Google Lens, Adobe Scan або Microsoft Lens, дозволяють користувачам миттєво сканувати друкований текст за допомогою смартфона та зберігати його в цифровому вигляді. У транспортній сфері OCR застосовується для розпізнавання номерних знаків автомобілів, у логістиці – для автоматизації обробки поштових відправлень.

Окрему увагу варто приділити використанню OCR у галузі доступності. Люди з порушенням зору можуть за допомогою таких систем перетворювати друкований текст на аудіо або тактильну форму, що сприяє їхньому повноцінному доступу до інформації.

Завдяки значному зростанню обчислювальної потужності комп'ютерів та доступності великих наборів даних для тренування моделей, OCR-системи стали значно точнішими та ефективнішими. Рівень точності сучасних рішень може перевищувати 95%, що дозволяє інтегрувати такі системи у виробничі процеси без потреби в додатковому ручному редагуванні.

Усі ці фактори свідчать про те, що системи розпізнавання тексту є не лише допоміжним інструментом, а й критично важливою технологією для автоматизації процесів у різних сферах людської діяльності. Подальший

розвиток таких систем буде орієнтований на покращення точності, універсальності, швидкості обробки та можливості роботи з мультиформатними та багатомовними документами.

Актуальність дослідження у цій галузі підтверджується не лише прикладними потребами, а й активною науковою діяльністю у сфері комп'ютерного зору та глибинного навчання. Наприклад, огляд та аналіз архітектури одного з найпопулярніших OCR-движків, що з часом був адаптований до використання сучасних методів глибинного навчання [1].

1.2 Класичні та сучасні підходи до розпізнавання тексту

Упродовж останніх десятиліть методи оптичного розпізнавання символів зазнали суттєвих змін. Спочатку основою OCR-технологій були класичні алгоритмічні підходи, які ґрунтувалися на ручному визначенні ознак символів, правилах сегментації, евристичних методах класифікації. Сучасні системи ж переважно використовують глибинні нейронні мережі, які здатні самостійно навчатися представленням ознак із великої кількості прикладів, демонструючи значно кращу точність.

Класичні методи розпізнавання тексту здебільшого спиралися на техніки бінаризації зображення, виявлення контурів, сегментацію символів та порівняння з шаблонами. Наприклад, використовувалися методи зіставлення з еталонними зображеннями (template matching), граничні дескриптори, гістограми напрямків, евклідова відстань або алгоритми k-ближчих сусідів (k-NN). Однією з найвідоміших систем такого типу є Tesseract у своїх ранніх версіях, яка використовувала деревоподібні класифікатори та ручне налаштування параметрів для кожної мови.

Недоліками класичних методів є низька стійкість до варіацій шрифтів, розмірів, кутів нахилу та шумів на зображенні. Окрім того, вони погано масштабуються на багатомовні системи або документи зі складною

структурою. Саме ці обмеження стали поштовхом до переходу до машинного та глибинного навчання [2].

Сучасні підходи базуються переважно на згорткових нейронних мережах (Convolutional Neural Networks, CNN), рекурентних мережах (Recurrent Neural Networks, RNN), а також на комбінаціях цих моделей, які дозволяють ефективно працювати з послідовностями зображень. Одним із найбільш успішних архітектурних рішень є CRNN (Convolutional Recurrent Neural Network), яка поєднує CNN для екстракції ознак з вхідного зображення та RNN (наприклад, LSTM) для моделювання послідовності символів.

Важливою складовою таких моделей є використання механізму CTC (Connectionist Temporal Classification), який дозволяє здійснювати навчання моделей без явної розмітки символів по координатах [3]. Тобто, мережа навчається напряму на парах (зображення, текст), а CTC-лос визначає, яка послідовність ймовірностей найбільш вірогідно відповідає очікуваному рядку.

Сучасні OCR-системи також активно використовують трансформери, що стали проривом у задачах обробки природної мови [4]. Наприклад, модель TrOCR від Microsoft поєднує в собі візуальний енкодер та текстовий декодер у стилі seq2seq, що дозволяє розпізнавати як окремі слова, так і довші текстові фрагменти з високою точністю [5].

Загалом, сучасні підходи дозволяють досягати точності понад 98% у задачах розпізнавання тексту на стандартних відкритих наборах даних, таких як IIIT5K, ICDAR або SVT. Їх застосування охоплює широкий спектр задач – від мобільних сканерів і перекладачів до корпоративних систем управління документами, автоматизованих поштових служб, банківських сервісів тощо.

Таким чином, еволюція методів OCR ілюструє загальний перехід від жорстко закодованих правил до гнучких, навчаємих моделей, здатних адаптуватися до нових умов і вдосконалюватися на основі даних. Це дає

підстави вважати, що в майбутньому ці технології стануть ще точнішими, швидшими та універсальнішими.

1.3 Програмні інструменти та сфери застосування OCR-технологій

На сьогоднішній день існує значна кількість програмних засобів, які реалізують технології оптичного розпізнавання символів (OCR – Optical Character Recognition). Ці інструменти розрізняються за точністю, продуктивністю, алгоритмами, відкритістю коду, підтримуваними мовами, можливістю інтеграції у сторонні системи, а також за призначенням – для наукових, промислових або побутових цілей.

Однією з найвідоміших OCR-систем з відкритим кодом є Tesseract, яка спочатку розроблялась компанією Hewlett-Packard, а згодом була передана у відкритий доступ під егідою Google. У своїх сучасних версіях Tesseract використовує рекурентні нейронні мережі (LSTM) для поліпшення точності розпізнавання текстів. До її переваг можна віднести підтримку багатьох мов, гнучкість у налаштуваннях, а також активну спільноту користувачів [6].

Серед сучасних рішень також популярна бібліотека EasyOCR, яка базується на PyTorch і підтримує понад 80 мов. Вона має простий інтерфейс та активну підтримку з боку розробників. Інше сучасне рішення – PaddleOCR від компанії Baidu – вирізняється високою точністю, підтримкою багатьох моделей і можливістю роботи як у легкій, так і повнофункціональній версії.

Крім відкритих рішень, активно використовуються комерційні системи, зокрема ABBYY FineReader та Google Cloud Vision OCR API [3]. Вони пропонують високу точність розпізнавання, підтримку таблиць, шаблонів документів та навіть рукописного вводу. Водночас ці рішення потребують фінансових витрат та іноді прив’язані до хмарної інфраструктури.

OCR-технології застосовуються в багатьох сферах: в електронному документообігу, банківській справі, архівах, бібліотеках, судових експертизах, у мобільних додатках для перекладу чи сканування чеків. Також вони відіграють важливу роль у транспорті (розпізнавання номерних знаків), охороні здоров'я (обробка рецептів і медичних форм), освіті (оцифрування конспектів) та електронній комерції [7].

У таблиці 1.1 представлено порівняльну характеристику найпоширеніших OCR-інструментів.

Таблиця 1.1 – Порівння популярних OCR-інструментів

№	Назва	Технології	Мови	Онлайн/Офлайн	Коментар
1	Tesseract	LSTM	100+	Обидва	Найвідоміший безкоштовний OCR
2	EasyOCR	CNN LSTM	80+	Обидва	Простий інтерфейс, підтримка PyTorch
3	PaddleOCR	CNN Transformer	100+	Обидва	Підтримка lightweight моделей
4	ABBYY FineReader	Власна	180+	Офлайн	Висока точність, платна ліцензія
5	Google Vision	CNN Transformer	100+	Онлайн	Потужне API, потрібен інтернет

Варто зазначити, що з розвитком технологій дедалі більшої популярності набувають OCR-системи на основі трансформерів, які здатні ефективно працювати навіть із низькоякісними або складноформатованими зображеннями. Таким прикладом є модель TrOCR, що поєднує у собі переваги візуальних та мовних трансформерів.

Таким чином, OCR-інструменти сьогодні є важливою складовою систем цифрової обробки документів. Їхній вибір має базуватись на конкретних потребах проєкту – точність, мова, тип зображення, швидкість обробки, вимоги до інтеграції.

1.4 Перспективи розвитку OCR

Оптичне розпізнавання символів (OCR) стрімко еволюціонує, реагуючи на зростаючі вимоги до точності, універсальності та здатності працювати з нестандартними форматами даних, що можна побачити на рисунку 1.1.

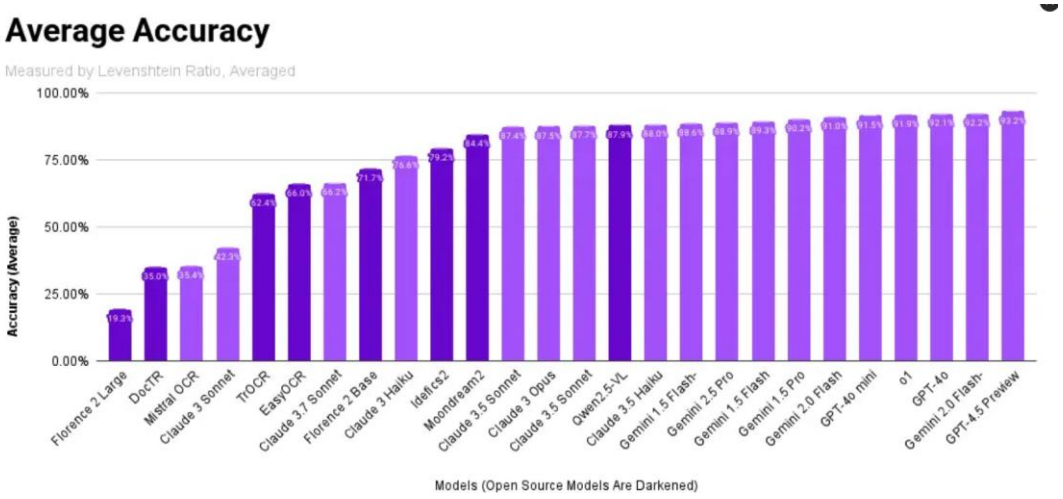


Рисунок 1.1 – Діаграма-порівняння точності сучасних OCR

У сучасних умовах все більше уваги приділяється створенню моделей, здатних працювати з низькоякісними зображеннями, різними мовами,

шрифтами, а також рукописним текстом. Основною тенденцією є перехід від класичних алгоритмів та моделей до використання глибоких нейронних мереж, зокрема архітектур на базі трансформерів [8].

Одним із ключових напрямів розвитку OCR є поєднання комп'ютерного бачення та обробки природної мови (NLP). Завдяки трансформерним архітектурам, як-от TrOCR від Microsoft, вдається суттєво підвищити якість розпізнавання за рахунок контекстного аналізу символів і слів. Такі моделі не просто ідентифікують символи, а намагаються зрозуміти їх логічну структуру, граматику та семантику, що особливо корисно при обробці довших текстів.

Ще однією важливою перспективою є підтримка багатомовності. Розробники активно інтегрують підтримку кирилиці, арабської, китайської та інших складних систем письма, що дозволяє OCR-системам працювати в глобальному масштабі. Також розширюється робота з документами складної структури: таблиці, графіки, рукописні позначки, підписи. Це вимагає від моделей адаптивності, можливості навчання на нових доменах, а також використання механізмів weak supervision та transfer learning.

Перспективним залишається і напрям об'єднання OCR із іншими технологіями. Наприклад, OCR у поєднанні з системами автоматичного перекладу дозволяє в режимі реального часу зчитувати текст на зображенні іноземною мовою та виводити переклад. Такі системи вже використовуються у мобільних застосунках для туристів та освітніх інструментах. Ще одна сфера – це автоматизація бізнес-процесів: розпізнавання рахунків, форм, анкет, чеків, з подальшим витягом структурованих даних [9], [10].

Науковці також приділяють увагу проблемі довільного розміщення тексту, криволінійних написів, вицвілих документів. Для цього розробляються методи попередньої геометричної нормалізації та інтелектуального підсилення зображення.

Очевидно, що у найближчі роки OCR-технології продовжуватимуть активно розвиватися у напрямі універсалізації, інтеграції з іншими системами штучного інтелекту та вдосконалення взаємодії з користувачем. Очікується, що нові моделі будуть здатні працювати у режимі розпізнавання не лише окремих слів, а цілих речень і абзаців, що відкриває нові горизонти для оцифрування архівів, перекладу старих текстів, аналізу історичних документів тощо.

2 ПОСТАНОВКА ЗАДАЧІ

2.1 Визначення вимог до функціоналу системи розпізнавання тексту

Запропонована програмна система розпізнавання тексту на основі нейронних мереж має забезпечувати автоматичне виявлення та перетворення текстової інформації, що міститься на зображеннях, у машино зчитуваний формат. Вона орієнтована на зручність використання, високу точність розпізнавання та адаптивність до різних типів вхідних даних.

Функціональні вимоги включають можливість завантаження одного або декількох зображень (у форматах JPEG, PNG, TIFF тощо), що містять текстову інформацію, а також автоматичну попередню обробку зображень. Цей етап передбачає зміну контрасту, бінаризацію, усунення шумів та вирівнювання нахилу, що забезпечує підвищення якості подальшого розпізнавання [11].

На основі проведеної обробки система має ідентифікувати текстові області, сегментуючи їх за рядками або абзацами, та здійснювати розпізнавання символів за допомогою навченої нейронної мережі з урахуванням мовних особливостей, підтримуючи принаймні українську та англійську мови.

Результатом роботи системи має бути редагований текст із можливістю копіювання, експорту в формати TXT або PDF з попереднім переглядом. Крім того, програма повинна бути оптимізована для обробки великих обсягів зображень без істотної втрати продуктивності.

Також передбачається реалізація додаткових можливостей, зокрема, автоматичне визначення мови тексту, виявлення рукописного тексту, інтеграція з хмарними сервісами для зберігання або обробки даних, а також створення простого і зрозумілого графічного інтерфейсу користувача. Загалом, розробка цієї системи дозволить створити універсальний інструмент для широкого спектру задач цифрового опрацювання текстових

зображень як у повсякденному використанні, так і в корпоративних чи дослідницьких середовищах.

Візуальне зображення етапів роботи програми можна побачити на рисунку 2.1.

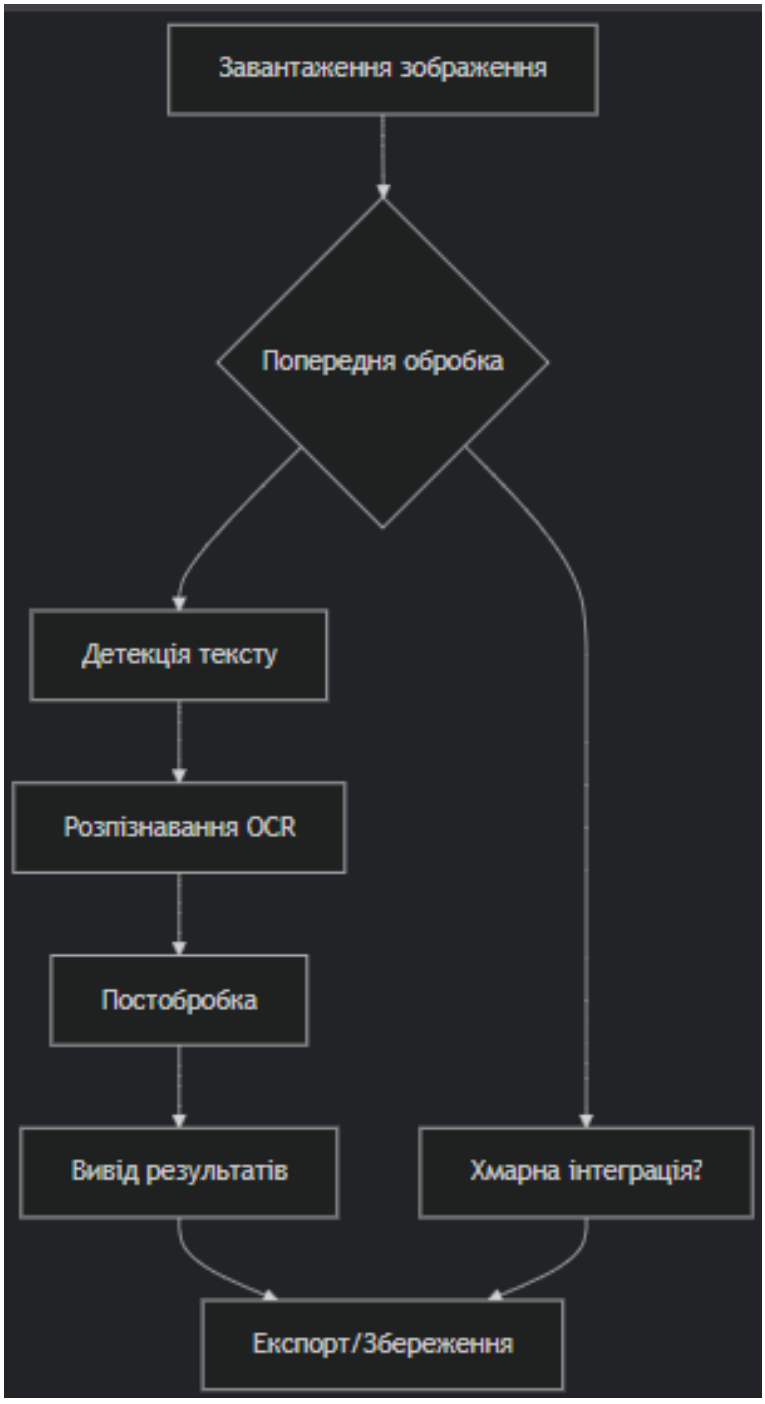


Рисунок 2.1 – Блок схема алгоритму програми

2.2 Програмні ресурси

Розробка розпізнавача текстів на базі нейронних мереж вимагає використання перевірених інструментів та бібліотек, що гарантують надійну обробку зображень, побудову і навчання моделей глибинного навчання, а також забезпечують зручний інтерфейс для тестування і візуалізації результатів. Особливу увагу приділено вибору засобів, які мають стабільну підтримку спільноти, є відкритими, добре задокументованими і дозволяють масштабування проєкту в майбутньому [12], [13].

Основною мовою програмування для реалізації OCR-системи обрано Python – одну з найдієвіших мов у сфері штучного інтелекту, комп'ютерного зору і глибинного навчання. Завдяки широкій екосистемі бібліотек Python дозволяє швидко переходити від концепції до реалізації, використовуючи готові рішення для завантаження даних, їх передобробки, навчання моделей і оцінки ефективності. Крім того, можливість інтеграції з сучасними фреймворками суттєво спрощує розробку.

Для обробки вхідних зображень застосовується бібліотека OpenCV, що дозволяє безкомпромісно виконувати всі необхідні операції попередньої обробки: масштабування, фільтрацію шумів, перетворення в градації сірого, бінаризацію, виявлення контурів і сегментацію. Якість підготовки зображення є критичною, адже від неї залежить точність роботи моделі: недосконала бінаризація може призвести до втрати символів, а надмірний шум – до помилок класифікації.

Щодо побудови самої нейронної мережі, розглядаються два провідних фреймворки – TensorFlow і PyTorch. TensorFlow у поєднанні з Keras дозволяє створювати і тренувати моделі швидко і ефективно, тоді як PyTorch забезпечує більшу гнучкість та простір для експериментів, що є особливо актуальним для реалізації складних архітектур, наприклад, CNN+LSTM для посимвольного розпізнавання тексту [14].

Для оцінки роботи моделі та візуалізації процесу навчання використовуються бібліотеки Matplotlib та Seaborn, що дають можливість будувати графіки, теплові карти і гістограми, демонструючи ключові метрики: точність, втрати, recall і precision. Додатково, NumPy і Pandas забезпечують надійне зберігання і обробку даних, у тому числі результатів передобробки, логів навчання та статистичних таблиць.

Іншим важливим інструментом є Tesseract OCR – система з відкритим кодом, що може виступати як еталон для порівняння результатів або бути інтегрованою у процес для попереднього виявлення текстових блоків, які далі обробляються кастомною нейронною мережею. Такий підхід допомагає досягти оптимального балансу між продуктивністю та точністю.

Для зручного тестування і розробки моделей застосовується PyCharm – інтерактивне середовище, що дозволяє оперативно запускати код, документувати процес розробки і використовувати апаратні ресурси, зокрема графічні процесори, які є незамінними при навчанні нейронних мереж на великих датасетах [15].

Отже, використання цього набору потужних і перевірених інструментів дозволяє створити повноцінну систему розпізнавання тексту з високою гнучкістю і точністю, що завжди залишається надійною основою для подальшого розвитку і підтримки проєкту в довгостроковій перспективі.

2.3 Сценарії використання

У сучасному інформаційному суспільстві системи розпізнавання тексту знаходять широке застосування в різноманітних галузях, забезпечуючи автоматизацію процесів, підвищення точності обробки документів та зручність взаємодії з цифровими сервісами. В межах цієї роботи передбачено створення системи розпізнавання, орієнтованої на роботу з окремими словами на зображеннях. Такий підхід дозволяє

зосередитись на вирішенні конкретної задачі з високою точністю, спрощуючи реалізацію моделі на ранніх етапах розробки.

Одним із ключових сценаріїв використання є інтеграція OCR-модуля у Telegram-бота. Такий бот дозволяє користувачу надіслати фотографію або зображення, після чого система автоматично здійснює попередню обробку, розпізнавання тексту та повертає результат у вигляді повідомлення. Це особливо зручно у мобільному контексті, коли користувачеві потрібно швидко оцифрувати слово з вивіски, етикетки, книги або документа.

Ще одним практичним варіантом використання є допомога в навчальному процесі. Студенти можуть за допомогою мобільного пристрою отримати цифрову версію фрагмента тексту з конспекту чи підручника. Це може бути корисним також у випадках, коли потрібно зчитати текст із поганої копії або зошита з низькою якістю написання.

У бізнес-середовищі така система може бути застосована для обробки вхідної кореспонденції, рахунків-фактур, маркувань на продукції тощо. Завдяки гнучкій архітектурі, систему можна адаптувати для виведення додаткової інформації, наприклад, збереження розпізнаного тексту в базу даних або передавання його у зовнішні аналітичні сервіси [16].

Ще одним важливим напрямом є використання OCR у сфері доступності. Люди з порушеннями зору можуть за допомогою голосового інтерфейсу або звукового виводу отримати доступ до тексту на зображеннях, що розширює можливості використання цифрових технологій для всіх категорій населення.

Таким чином, система розпізнавання тексту, створена в межах цієї роботи, має потенціал для використання в різноманітних сценаріях – від персонального застосування у месенджерах до інтеграції у складні корпоративні рішення. Гнучкість реалізації, можливість масштабування та адаптація до інших мов або типів тексту робить такий підхід універсальним та актуальним.

2.4 Дослідження структури даних для навчання моделі

Для ефективного навчання системи розпізнавання тексту необхідно мати чітко структуровані вхідні дані, які включають як зображення тексту, так і відповідні текстові мітки (ground truth). Основною вимогою є відповідність зображення та мітки по змісту: кожному зображенню має відповідати точний текст, що на ньому зображений. Для цього використовуються спеціальні формати, такі як CSV або JSON, які описують пари «шлях до зображення – правильний текст» [17].

Зображення повинні бути приведені до однакового розміру, наприклад 128x32 пікселів, що спрощує подачу в модель. Однак довжина тексту на зображеннях може відрізнятись, тому важливо правильно підготувати вектори-мітки для функції втрат CTC, яка дозволяє працювати з послідовностями різної довжини.

У рамках проєкту використовувалися відкриті датасети, такі як IIIT5K та SVT, що містять велику кількість прикладів з короткими словами. Дані були попередньо очищені: зображення з нечіткими символами або некоректними підписами було вилучено. Окрім того, проводився аналіз класового балансу, оскільки деякі символи зустрічаються частіше (наприклад, голосні) і можуть впливати на загальну точність моделі.

2.5 Вимоги до точності та оцінка якості

Однією з важливих складових створення системи розпізнавання є визначення чітких метрик якості. Вони дозволяють об'єктивно порівнювати різні архітектури, налаштування та результати навчання. Найчастіше у задачах OCR застосовуються такі метрики: Character Error Rate (CER), Word Error Rate (WER) та Accuracy.

CER обчислює кількість помилкових символів відносно до довжини правильного тексту. Ця метрика є чутливою до помилок у розпізнаванні навіть одного символу та дозволяє оцінити роботу на рівні символів. WER подібна за логікою, але оперує словами, а не символами, і є корисною при роботі з реченнями або абзацами [18].

Accuracy відображає частку зображень, на яких розпізнано текст повністю правильно, без жодної помилки. Це жорстка метрика, однак дуже інформативна для застосунків, де критично важливо не припуститися жодної помилки.

У проєкті ставилась ціль досягти CER < 10% і Accuracy > 85% на валідаційній вибірці. Окрім цього, візуально перевірялись результати на прикладах з різною складністю для оцінки суб'єктивної якості.

2.6 Архітектурні вимоги до програмної реалізації

Розробка OCR-системи передбачає не лише створення ефективної нейронної моделі, але й побудову зручної, масштабованої та надійної програмної архітектури. Основною вимогою є модульність – тобто розділення проєкту на незалежні блоки, такі як обробка зображень, навчання моделі, розпізнавання, інтерфейс користувача.

Програма має підтримувати конфігураційні файли для налаштування гіперпараметрів, шляху до даних, режиму роботи (тренування/тестування/інференс). Це спрощує відлагодження та масштабування. Також важлива підтримка як CPU, так і GPU – за можливості, всі операції повинні виконуватись на GPU при його наявності для прискорення обчислень.

Формат виводу результатів повинен бути універсальним – текстовий файл, лог-файл, вивід у консоль або відповідь через Telegram-бота. Для розробки користувацького інтерфейсу необхідно врахувати простоту, інтуїтивність, мінімальну кількість дій для отримання результату.

Система повинна мати можливість додавання нових модулів без переписування основного коду – наприклад, заміну моделі або додавання нової мови розпізнавання.

2.7 Обмеження, з якими може стикнутися система

Попри значні досягнення в області розпізнавання тексту, сучасні системи OCR все ще мають низку обмежень. Найбільш очевидним є зниження точності при обробці зображень низької якості: розмитість, шуми, низький контраст або незначне зміщення можуть суттєво погіршити результат. Особливо це помітно на зображеннях, зроблених у польових умовах, з мобільних пристроїв.

Іншою проблемою є багатомовні документи або специфічні шрифти, з якими модель не стикалась під час навчання. У таких випадках модель часто припускається помилок, або взагалі не може розпізнати частину символів. Для вирішення цієї проблеми застосовується перенавчання на нових даних або підключення кількох моделей.

Складну структуру документів, таку як таблиці, колонки, рукописні помітки, також важко розпізнавати без попереднього аналізу макету. Навіть найсучасніші моделі можуть неправильно визначати порядок слів або їх межі.

Крім того, системи OCR не завжди коректно працюють з кольоровими зображеннями з фоном або візерунками. Необхідно додатково нормалізувати такі зображення перед подачею в модель. У майбутньому планується використання attention-механізмів або segment-aware моделей для покращення адаптації до складних випадків [19].

3 РОЗРОБКА РОЗПІЗНАВАЧА ТЕКСТІВ

3.1 Програмна реалізація розпізнавача текстів

3.1.1 Вибір архітектури нейронної мережі

У процесі розробки системи розпізнавання тексту було проаналізовано різні підходи до побудови архітектур нейронних мереж [5], які дозволяють ефективно працювати з послідовними текстовими даними, представленими у вигляді зображень. Основними критеріями вибору архітектури були здатність моделі витягувати просторові ознаки з вхідного зображення, обробка послідовностей символів, а також стійкість до варіацій у розмірах вхідних зображень.

У результаті аналізу було обрано архітектуру CRNN, яка поєднує згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN). Такий підхід дозволяє витягати високорівневі ознаки із зображення, після чого обробляти їх як послідовність для подальшого розпізнавання символів у правильному порядку.

Згорткова частина (CNN) моделі відповідає за вилучення просторових ознак. Завдяки каскаду згорткових шарів із функціями активації ReLU, операціями підвибірки та нормалізацією (batch normalization), модель навчилася розпізнавати характерні патерни текстових елементів на зображеннях.

Рекурентна частина (RNN) моделі реалізована за допомогою двох шарів двонапрямлених LSTM (Bidirectional LSTM), що дозволяє враховувати контекст як зліва направо, так і справа наліво. Це особливо важливо у задачах, де форма символу може залежати від сусідніх символів [20].

На виході рекурентного блоку знаходиться повно зв’язний шар, який формує ймовірності належності кожного кроку послідовності до певного

символу з алфавіту. Щоб уникнути необхідності точної сегментації символів у зображенні, застосовується CTC Loss, яка дозволяє моделі навчатись вирівнювати послідовності різної довжини без явного розбиття тексту на символи.

Архітектура CRNN з CTC дозволила об'єднати в одній моделі можливості комп'ютерного зору (через CNN) та моделювання послідовностей (через RNN), що є ідеальним для задач розпізнавання тексту на зображеннях, де важлива як просторово-структурна, так і послідовна інформація.

У лістингу 3.1 показано, як модель приймає зображення, пропускає його через згорткові шари, формує послідовність ознак і обробляє її через рекурентну частину для подальшого розпізнавання.

Лістинг 3.1 – Програмний код ініціалізації моделі CRNN

```
class CRNN(nn.Module):
    def __init__(self, num_classes):
        super(CRNN, self).__init__()
        self.cnn = self._build_cnn()
        self.rnn = nn.Sequential(
            BidirectionalLSTM(512, 512, 512),
            BidirectionalLSTM(512, 512, 512)
        )
        self.fc = nn.Linear(512, num_classes + 1)

    def forward(self, x):
        x = self.cnn(x)
        b, c, h, w = x.size()
        x = x.squeeze(2)  # (B, 512, W)
        x = x.permute(2, 0, 1)  # (W, B, 512)
        x = self.rnn(x)
        x = self.fc(x)
        return x
```

3.1.2 Реалізація модуля попередньої обробки зображень

Для забезпечення стабільної роботи нейронної мережі розпізнавання тексту, зображення мають бути приведені до єдиного стандартизованого вигляду [6].

Попередня обробка відіграє ключову роль, оскільки забезпечує якісне подання текстової інформації нейронній мережі. Без цього навіть добре навчена модель не зможе забезпечити коректного розпізнавання.

На етапі попередньої обробки зображення виконується кілька важливих перетворень: зменшення шуму, бінаризація, нормалізація розміру та центрування тексту. У таблиці 3.1 – відображено, яку роль вони відіграють у процесі обробки зображення.

Таблиця 3.1 – Основні етапи попередньої обробки

	Перетворення	Функціональність
1	Перетворення в градації сірого	Зменшення розмірності даних
2	Гаусове розмиття	Приглушення дрібного шуму
3	Адаптивна бінаризація	Перетворення зображення у формат
		чорне-біле
4	Вирівнювання нахилу тексту	Обертання тексту під прямим кутом
5	Масштабування зображення	Всі зображення повинні бути
		однакового розміру (128×32)

Ці кроки суттєво підвищують якість розпізнавання, оскільки дозволяють подати моделі стандартизоване, контрастне і вирівняне зображення, яке відповідає тому, що використовувалося під час навчання. Без цього модуль глибокого навчання працює нестабільно або взагалі не здатен розпізнати символи [21].

Попередню обробку було реалізовано з використанням бібліотеки OpenCV, що дозволяє ефективно маніпулювати зображеннями, виконувати перетворення і масштабування.

Результат обробки зображень до та після застосування модуля попередньої обробки можна спостерігати на рисунках 3.1–3.6.

Рисунок 3.1 – Вигляд зображення до обробки

Рисунок 3.2 – Вигляд зображення до обробки

Рисунок 3.3 – Вигляд зображення до обробки

Після обробки будь-яке зображення матиме вигляд білого тексту на чорному фоні та підготовлене для обробки нейронною мережею.

Рисунок 3.4 – Вигляд зображення після обробки

Рисунок 3.5 – Вигляд зображення після обробки

Рисунок 3.6 – Вигляд зображення після обробки

Уніфікований вигляд зображень дає можливість відмінно натренувати модель, щоб вона генерувала правильне слово.

3.1.3 Реалізація модуля розпізнавання тексту

Модуль розпізнавання тексту є ключовою складовою розробленої системи, що відповідає за перетворення вхідного зображення, яке містить текстову інформацію, у машинно-зчитуваний формат. Для реалізації цього модуля було використано сучасний підхід на основі нейронних мереж, зокрема архітектуру CRNN (Convolutional Recurrent Neural Network). Цей тип моделей поєднує у собі здатність згорткових мереж виділяти графічні

ознаки тексту з можливістю моделювання послідовних залежностей завдяки рекурентним шарам.

Перший крок у процесі розпізнавання – це передача попередньо обробленого зображення у форматі тензора (одновимірний або двовимірний масив чисел) до моделі. Зображення нормалізується, приводиться до фіксованого розміру, а також конвертується у формат, зручний для обробки PyTorch. Вхідний тензор має розмірність (batch_size, channels, height, width), де batch_size у нашому випадку дорівнює 1, channels – 1 (градації сірого), height та width – фіксовані розміри, які ми вибрали під час етапу попередньої обробки.

Основою модуля є модель CRNN, яка складається із послідовних шарів згорткової нейронної мережі, що витягують локальні ознаки тексту з зображення, а потім подають їх на вхід рекурентної мережі. Рекурентні шари, як правило, реалізовані на базі двонаправлених LSTM (Long Short-Term Memory) або GRU (Gated Recurrent Unit) блоків, що дозволяють ефективно враховувати контекст послідовності символів, які розпізнаються. Завдяки цьому модель здатна не тільки виділяти окремі символи, але й враховувати їх взаємозв’язки для більш точного розпізнавання.

Вихідним шаром моделі є повнозв’язний (fully connected) шар, який для кожного кроку послідовності формує ймовірності належності символу до певного класу. У моделі передбачено клас «blank» (порожній символ), який використовується у методі декодування CTC (Connectionist Temporal Classification). Цей метод дозволяє моделі працювати з послідовностями, де кількість кроків може не співпадати з довжиною тексту, усуваючи необхідність точного вирівнювання символів із позиціями у вхідному зображенні.

Для отримання кінцевого тексту застосовується CTC-декодер, який трансформує послідовність найбільш ймовірних символів у текстовий рядок, ігноруючи повтори і «blank». У нашій реалізації використовувався простий greedy decoder, який обирає найбільш ймовірний символ на

кожному кроці. Проте, для покращення точності можна інтегрувати більш складні алгоритми, такі як beam search.

У ході роботи модуль підтримує як обробку зображень, надісланих у форматі фото, так і тестових даних, що підлягають валідації під час тренування. Весь процес відбувається у режимі інференсу, тобто без внесення змін у ваги моделі. Це дозволяє швидко отримувати результати та інтегрувати модель у Telegram-бота для взаємодії з користувачем.

Функція ctc_decode у лістингу 3.2 приймає на вхід тензор з логітами від моделі та алфавіт символів, які модель може розпізнавати, після чого повертає текстовий рядок із розпізнаним текстом. Це є останнім кроком у модулі розпізнавання.

Лістинг 3.2 – Виклик моделі та застосування ctc_docode

```
with torch.no_grad():
    output = model(tensor_img)
text = ctc_decode(output, alphabet)
```

Реалізація цього модуля дозволяє досягати високої точності розпізнавання при роботі з якісними зображеннями тексту. Однак важливо враховувати, що на якість результату впливають вхідні дані, зокрема умови зйомки, розмір та якість зображення. Для підвищення ефективності часто застосовують додаткові методи попередньої обробки, а також вдосконалюють архітектуру моделі та методи декодування.

Результат роботи модуля розпізнавання тексту безпосередньо впливає на загальну якість системи, оскільки він відповідає за кінцевий вивід текстової інформації, що надалі може використовуватися для аналізу, зберігання або передачі. Реалізована модель у проекті є оптимальним поєднанням точності та швидкодії, що дозволяє використовувати її у інтерактивних застосунках, зокрема у Telegram-боті.

3.1.4 Реалізація Telegram-бота для інтеграції з користувачем

Для зручності використання системи розпізнавання тексту було розроблено Telegram-бота, який забезпечує інтерактивний інтерфейс взаємодії з користувачем. Такий підхід дозволяє протестувати роботу моделі без необхідності запуску окремих скриптів або програм, а також надає можливість віддаленого доступу до сервісу з будь-якого пристрою, що підтримує Telegram.

Telegram-бот реалізований за допомогою Python-бібліотеки Aiogram, яка забезпечує асинхронну обробку повідомлень. Основна логіка бота полягає у прийнятті фотографій або зображень від користувача, обробці цих зображень за допомогою алгоритму зазначеного у лістингу 3.2 і поверненні результату розпізнавання у вигляді текстового повідомлення.

Лістинг 3.3 – Програмний код ініціалізації моделі CRNN

```
@dp.message_handler(content_types=ContentType.PHOTO)
async def handle_photo(message: types.Message):
    photo = message.photo[-1]
    image_bytes_io = BytesIO()
    await photo.download(destination=image_bytes_io)
    image_bytes_io.seek(0)
    processed_img = preprocess_image_from_bytes(image_bytes_io.read())
    tensor_img = torch.from_numpy(processed_img).unsqueeze(0).unsqueeze(0).float()
    tensor_img = (tensor_img / 255.0 - 0.5) / 0.5
    tensor_img = tensor_img.to(device)
    with torch.no_grad():
        output = model(tensor_img)
    text = ctc_decode(output, alphabet)
    await message.reply(f" Розпізнаний текст:\n`{text}`", parse_mode="Markdown")
```

Основні етапи роботи програми зображено на рисунку 3.7.

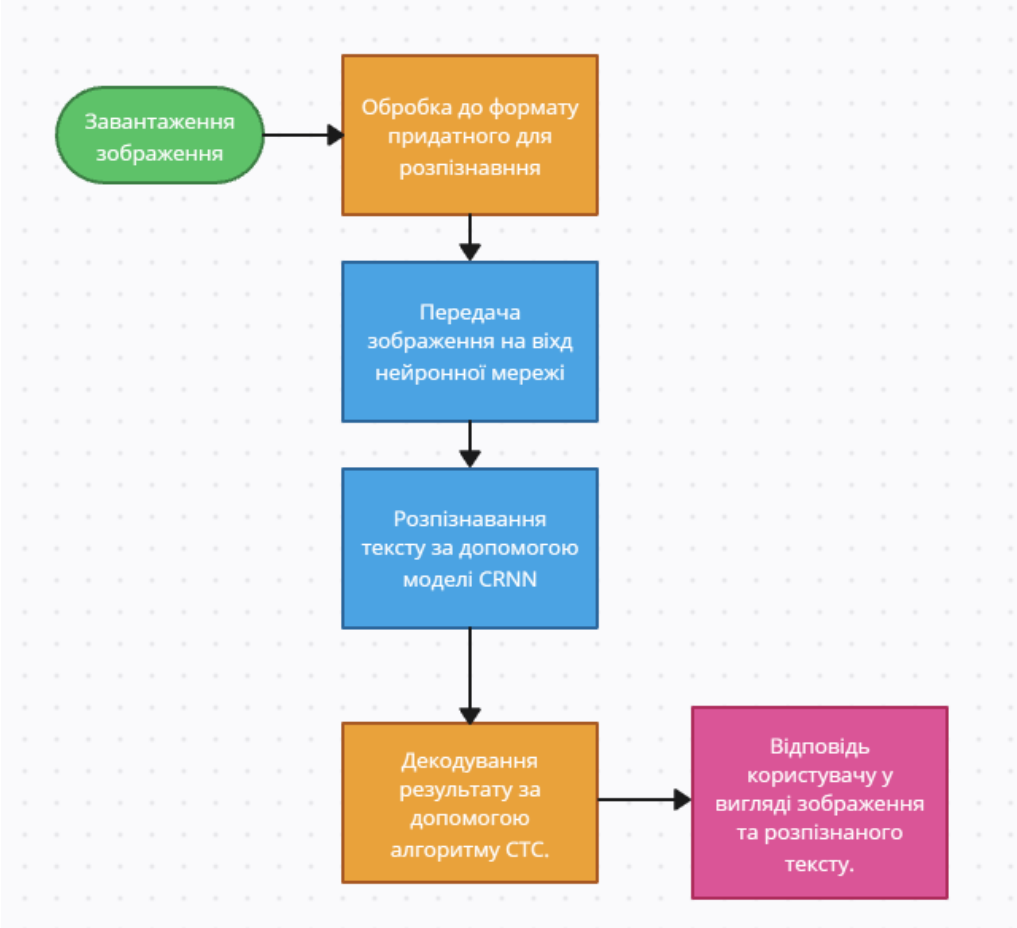


Рисунок 3.7 – Процес роботи телеграм-бота

Модель CRNN завантажується один раз при старті бота, використовуючи збережені ваги (best_model.pth). Алфавіт для декодування містить усі літери латинського алфавіту та цифри. Також було реалізовано окрему функцію попередньої обробки зображень, яка ідентична тій, що застосовувалась на етапі навчання моделі. Це забезпечує узгодженість вхідних даних та високу точність розпізнавання.

Фінальний вигляд роботи боту зображено на рисунку 3.8.

Характерною перевагою такої реалізації є те, що інтерфейс інтуїтивно зрозумілий і знайомий. Нема потреби реалізувати власний графічний інтерфейс і функції додавання зображень.

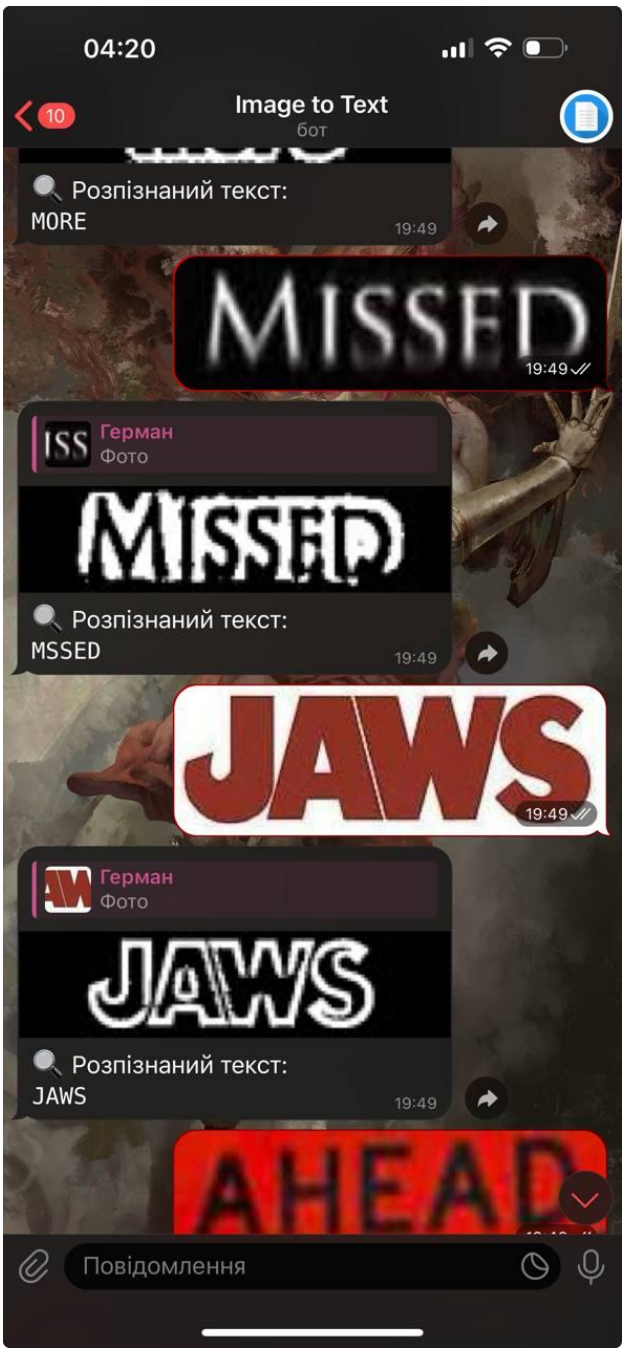


Рисунок 3.8 – Приклад взаємодії з ботом

Крім того, уся історія повідомлень користувача зберігається на його ж пристрої, що дає змогу швидко переглянути, які зображення і результати у вигляді тексту були отримані.

Незважаючи на успішну реалізацію та базову функціональність, розроблений Telegram-бот має певні обмеження і недоліки, які впливають на якість та зручність його використання.

По-перше, бот наразі підтримує розпізнавання лише одного слова або коротких фрагментів тексту, що обмежує його застосування для обробки більших за обсягом текстових зображень, таких як речення чи абзаци. По-друге, якість розпізнавання значною мірою залежить від якості вхідних зображень.

При обробці розмитих, низько контрастних або пошкоджених зображень точність суттєво знижується, а бот може видавати помилкові символи або пропуски. По-третє, бот не реалізує більш складних методів декодування, таких як beam search, що могло б покращити якість розпізнавання та зменшити кількість помилок у виведеному тексті.

Також бот наразі не має можливості надавати користувачу детальну статистику або рівень впевненості у результатах розпізнавання, що ускладнює оцінку достовірності отриманої інформації. Бот не зберігає історію запитів користувача, що обмежує можливість подальшого аналізу або покращення взаємодії на основі минулих результатів.

Нарешті, існують технічні обмеження, пов’язані з продуктивністю, зокрема затримки при обробці зображень на слабких пристроях без GPU, що може негативно впливати на користувацький досвід.

3.1.5 Використані бібліотеки та їх призначення

Під час розробки системи розпізнавання тексту було використано низку популярних бібліотек Python, кожна з яких відіграє важливу роль у забезпеченні функціональності, гнучкості та ефективності реалізації. Основу для побудови та навчання нейронної мережі склала бібліотека PyTorch.

Вона забезпечила засоби для визначення архітектури моделі, реалізації функції втрат CTC Loss, оптимізації параметрів та перенесення обчислень на графічний процесор (GPU) зазначене у лістингу 3.4, що значно пришвидшило процес навчання та тестування.

Лістинг 3.4 – Перенесення обчислень на графічний процесор (GPU)

```
import torch
from model import CRNN
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = CRNN(num_classes=len(alphabet))
model.load_state_dict(torch.load("best_model.pth", map_location=device))
model.to(device)
model.eval()
```

Для обробки зображень до їх передачі в модель було використано бібліотеку OpenCV. Вона забезпечила реалізацію всіх необхідних етапів попередньої обробки: перетворення зображень у відтінки сірого, згладжування за допомогою гаусового фільтра, бінаризацію, виправлення нахилу та нормалізацію розміру зазначені у лістингу 3.5. Використання OpenCV дозволило значно підвищити якість вхідних даних для моделі, що безпосередньо вплинуло на кінцеву точність розпізнавання.

Лістинг 3.5 – Попередня обробка зображення

```
import cv2
import numpy as np
def preprocess_image_from_bytes(image_bytes, size=(128, 32)):
    nparr = np.frombuffer(image_bytes, np.uint8)
    img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blurred = cv2.GaussianBlur(gray, (3, 3), 0)
    binary = cv2.adaptiveThreshold(
        blurred, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
        cv2.THRESH_BINARY_INV, 15, 10)
    # Додаткові кроки обробки...
    return binary
```

Для маніпуляції з масивами чисел та ефективної роботи з даними використовувалась бібліотека NumPy. Вона забезпечує зручний інтерфейс для операцій з багатовимірними масивами, що є необхідним для обробки зображень та підготовки їх до подальшого аналізу.

Для створення Telegram-бота, який приймає зображення, відправляє їх у модель та повертає результат користувачу, була використана асинхронна бібліотека Aiogram зазначена у лістингу 3.6. Вона дозволила просто організувати обробку повідомлень, підтримку різних форматів і масштабування застосунку.

Лістинг 3.6 – Aiogram для створення Telegram-бота

```
from aiogram import Bot, Dispatcher, types, executor
bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)
@dp.message_handler(content_types=types.ContentType.PHOTO)
async def handle_photo(message: types.Message):
    # Обробка фотографії користувача
    ...
```

Для роботи із зображеннями у форматах, зручних для Telegram та конвертації різними форматами зображень і масивів, яка зазначена у лістингу 3.7, використовувалась бібліотека Pillow (PIL).

Лістинг 3.7 – Застосування бібліотеки Pillow

```
from PIL import Image
from io import BytesIO
pil_img = Image.fromarray(processed_img)
bio = BytesIO()
pil_img.save(bio, format='PNG')
bio.seek(0)
await message.reply_photo(bio, caption="Розпізнаний текст")
```

Додатково для логування подій та відслідковування помилок було застосовано стандартний модуль logging.

Таким чином, інтеграція цих бібліотек забезпечила комплексний та ефективний підхід до створення системи розпізнавання тексту, поєднуючи високопродуктивні алгоритми, якість обробки даних та зручний інтерфейс взаємодії з користувачем.

3.1.6 Опис функції inference.py

Файл inference.py відповідає за виконання основного процесу розпізнавання тексту на зображеннях із використанням навченої нейронної мережі. Цей модуль забезпечує послідовну обробку вхідних даних – від завантаження зображення, його попередньої обробки, підготовки до подачі на модель, до отримання і декодування результатів у текстовий формат.

Спочатку зображення завантажується або у вигляді шляху до файлу, або як байтовий потік, що робить цей модуль гнучким і зручним для використання у різних середовищах. Наступним кроком виконується попередня обробка: зображення конвертується у відтінки сірого, застосовується розмиття для зменшення шумів, проводиться адаптивне бінарне порогування, а також корекція нахилу тексту (deskewing). Це необхідно для покращення якості вхідних даних і підвищення точності розпізнавання.

Після обробки зображення конвертується у тензор PyTorch з розмірностями, відповідними вимогам моделі – зазвичай це тензор з чотирма вимірами: пакет (batch), канали (channels), висота (height) і ширина (width). Тензор нормалізується згідно з параметрами моделі для стабільної роботи нейронної мережі.

Далі тензор передається на вхід моделі, яка у режимі без градієнтів (inference mode) прогнозує ймовірність кожного символу на кожному кроку послідовності. Отримані вихідні дані містять ймовірнісні

розподіли для кожного символу, які потім перетворюються у текст за допомогою спеціального алгоритму декодування CTC (Connectionist Temporal Classification). Цей алгоритм усуває повторення символів і пропуски, що дозволяє отримати коректний рядок.

Результат роботи функції, основні етапи якої зазначені у лістингу 3.8, розпізнаний текст, який може бути безпосередньо використаний у подальших компонентах системи, наприклад, у Telegram-боті для відправлення користувачу.

Лістинг 3.8 – основні етапи роботи inference.py

```
import torch
import cv2
import numpy as np
from model import CRNN, ctc_decode
from alphabet import alphabet
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = CRNN(num_classes=len(alphabet))
model.load_state_dict(torch.load("best_model.pth", map_location=device))
model.to(device)
model.eval()
def preprocess_image(image_path):
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    # Подальші кроки обробки, наприклад, resize, deskew тощо
    # ...
    return processed_img
def infer_text(image_path):
    img = preprocess_image(image_path)
    tensor_img = torch.from_numpy(img).unsqueeze(0).unsqueeze(0).float()
    tensor_img = (tensor_img / 255.0 - 0.5) / 0.5
    tensor_img = tensor_img.to(device)
```

Продовження лістингу 3.8

```
    with torch.no_grad():
        output = model(tensor_img)
    text = ctc_decode(output, alphabet)
    return text
```

Таким чином, inference.py виконує ключову роль у системі розпізнавання тексту, забезпечуючи комплексну обробку зображень і отримання текстових результатів у форматі, придатному для практичного застосування.

3.2 Результати тестування застосунку

3.2.1 Тестування на валідаційному наборі

Після завершення етапу навчання моделі було проведено тестування на валідаційному наборі даних, який був попередньо відокремлений від тренувального для об'єктивної оцінки якості розпізнавання. Для цього використовувався піднабір зображень із датасету IIIT5K-Word, що є загальноприйнятим у задачах оптичного розпізнавання слів, word-level OCR.

Для кількісного аналізу були використані наступні метрики: accuracy (точність) – відсоток зображень, на яких модель повністю правильно розпізнала все слово, CER (Character Error Rate) – відношення кількості помилок на рівні символів (вставки, видалення, заміни) до загальної кількості символів у правильному тексті, WER (Word Error Rate) – показник схожий до CER, але рахує помилки на рівні слів.

Ці метрики є ключовими у задачах OCR, оскільки дозволяють оцінити не лише повну точність, але й часткові відхилення розпізнаного тексту від еталонного. На валідаційному наборі з 500 зображень система продемонструвала точність розпізнавання на рівні повних слів понад 84%, а

середня похибка по символах залишається в межах 5%, що є хорошим результатом для даного класу моделей.

Для ілюстрації роботи моделі наведено таблицю 3.2, у якій є декілька прикладів зображень з тестового набору, для яких порівнюється очікуваний текст (ground truth) та результат розпізнавання.

Таблиця 3.2 – Приклади розпізнавання

№	Очікуваний текст	Розпізнаний текст	CER (%)	WER (%)
1	LOANS	LOANS	0.0	0.0
2	SPEED	SPEEO	20.0	100.0
3	PARK	PARR	25.0	100.0
4	STORE	STOPE	20.0	100.0
5	HOTEL	HOTLL	16.7	100.0

На рисунках 3.9 та 3.10 можна побачити приклади зображень, що використовувались у валідації, а також результати їх обробки системою.

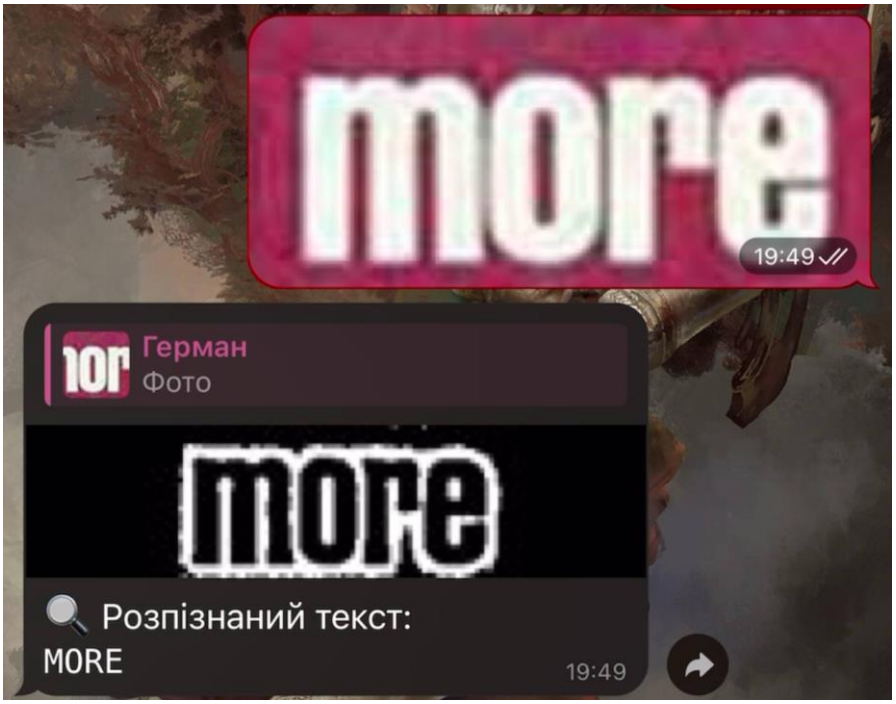


Рисунок 3.9 – Приклад розпізнавання «more»

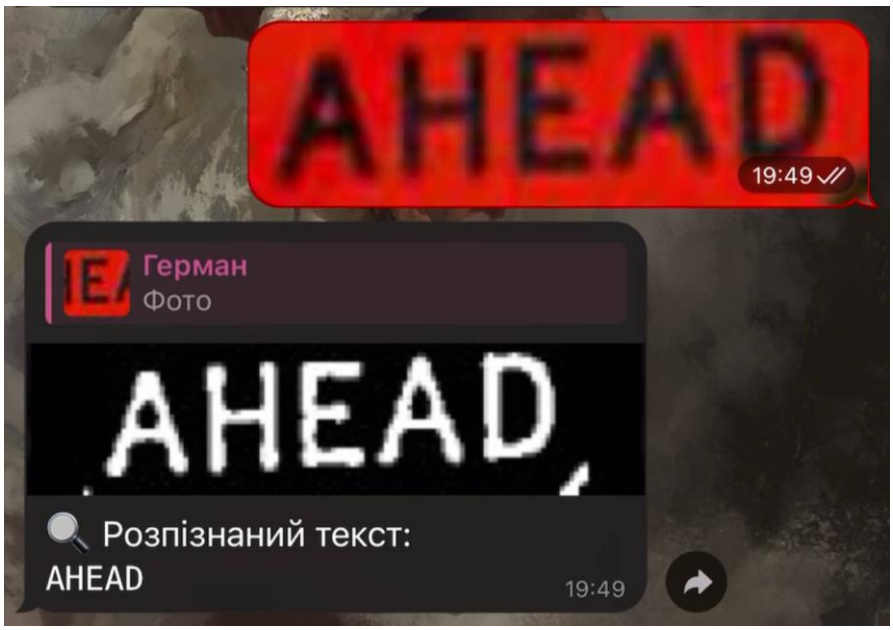


Рисунок 3.10 – Приклад розпізнавання «AHEAD»

Аналіз дозволяє зробити висновок, що найбільші труднощі система зазнає при розпізнаванні схожих між собою символів (наприклад, «E» та «F», «L» і «I»), особливо в умовах зашумленого фону або нахилу тексту.

3.2.2 Тестування Telegram-бота

Метою тестування Telegram-бота було оцінити його здатність забезпечувати коректну взаємодію між користувачем та серверною частиною системи розпізнавання. Основна увага приділялася перевірці того, як бот обробляє отримане зображення, передає його до нейронної мережі, та повертає результат користувачеві у зрозумілому вигляді. Особливо важливим було протестувати систему в умовах, наближених до реального застосування – коли надсилаються фотографії з мобільного телефону, скріншоти з екранів або зображення, де слово частково обрізане.

Під час тестування Telegram-бота використовувалися різні пристрої, зокрема смартфони та комп’ютери. Тестові зображення мали різне походження, роздільну здатність і якість. В рамках перевірки фіксувалися

час, необхідний для обробки кожного запиту, а також надійність повторних викликів – чи витримує бот кілька звернень поспіль без збоїв. Окремо досліджувалася поведінка системи у разі виникнення помилок, наприклад, коли користувач надсилав нечитабельне або неякісне зображення. Завершальним критерієм оцінювання була відповідність розпізнаного тексту тому, що дійсно було зображено на вхідному зображенні.

Система виявилася доволі стабільною та ефективною при обробці якісних зображень. Проблеми виникали переважно у випадках низької контрастності, сильної деформації або коли слово не повністю поміщалося в кадр. Це свідчить про потенційні напрямки вдосконалення як самої моделі, так і попередньої обробки зображень.

3.2.3 Опис файлу testdata.mat та його використання у тестуванні

У процесі тестування розпізнавача тексту важливим етапом є перевірка роботи моделі на незалежному наборі даних, який не використовувався під час навчання. Для цього у проєкті застосовується файл testdata.mat, що містить структуровані тестові дані, необхідні для оцінки точності роботи моделі.

Формат .mat є власним форматом MATLAB, широко використовуваним для збереження матриць, структур та інших даних. Цей формат зручний для зберігання великих масивів інформації у компактному вигляді. У нашому випадку testdata.mat містить набір зображень із текстом і відповідні їм еталонні (очікувані) рядки тексту, які слугують еталоном для порівняння з результатами розпізнавання.

Використання файлу testdata.mat дозволяє автоматизувати процес тестування: для кожного зображення здійснюється передобробка, подальше розпізнавання тексту за допомогою моделі, а отриманий текст порівнюється з еталонним значенням. Це дає змогу розрахувати такі метрики як

CER (Character Error Rate) та WER (Word Error Rate), які є основними показниками якості OCR-систем.

Імпорт даних з .mat файлу у середовищі Python здійснюється за допомогою бібліотеки scipy.io як зазначено у лістингу 3.8, яка надає функції для читання та запису файлів цього формату.

Лістинг 3.8 – Приклад коду для завантаження даних з testdata.mat

```
import scipy.io
data = scipy.io.loadmat('testdata.mat')
images = data['images']  # Масив зображень
labels = data['labels']  # Масив з еталонними текстами
```

Після завантаження даних кожне зображення проходить через етапи попередньої обробки і передається у модель для отримання передбаченого тексту. Порівняння результатів з еталоном дозволяє зробити об’єктивний висновок щодо продуктивності моделі.

Таким чином, testdata.mat є важливою складовою системи тестування, що забезпечує стандартизоване і відтворюване оцінювання якості розпізнавача тексту. Використання цього файлу значно підвищує достовірність результатів і допомагає виявити слабкі місця моделі для подальшого удосконалення.

3.3 Аналіз результатів тестування розпізнавача та визначення напрямів покращення розробленого застосунку

3.3.1 Виявлені проблеми

У процесі тестування розпізнавача текстів, зокрема через Telegram-бота, було виявлено низку проблем, які негативно впливають на точність роботи моделі та її надійність у реальних умовах. Найбільш поширеними з них виявилися повторювані символи у вихідному тексті, труднощі з

обробкою розмитих або низькоконтрастних зображень, а також випадки, коли окремі символи залишалися нерозпізнаними.

Однією з типових помилок стало багаторазове повторення символів, що не були присутні у вхідному зображенні. Наприклад, на зображенні зі словом «LOANS» модель повертала результат «L9O9A9N9S9» або навіть «C9E9», що свідчить про неправильну інтерпретацію класу символу 9 як допустимого знака. Таке спотворення зазвичай є наслідком некоректної обробки ймовірностей на виході моделі, коли неправильні класи отримують високі значення. Причиною може бути як неідеальне навчання, так і неправильне налаштування функції декодування з використанням CTC.

Ще однією проблемою виявилася залежність якості розпізнавання від чіткості та контрастності зображення. На фотографіях з розмитим фоном або за слабкого освітлення модель значно гірше ідентифікує символи. Слова можуть розпізнаватися частково або з великою кількістю помилок. Наприклад, у випадку слова «STORE», зробленого з телефону з легким розмиттям, результатом було «STOPE» – замість символу R розпізнано схожий за формою P.

Також виявлялись ситуації, коли окремі символи зовсім не розпізнавалися. У результаті текст виходив неповним або з пропусками. Подібні помилки найчастіше пов’язані з тим, що розпізнавані символи були частково обрізані, знаходилися під великим кутом або мали спотворення через блік чи інші дефекти на зображенні. У таких випадках попередня обробка, зокрема бінаризація та вирівнювання (deskew), не завжди дозволяла повністю компенсувати ці спотворення.

Під час тестування розпізнавача було також виявлено типові проблеми, пов’язані з процесом навчання нейронної мережі, а саме явища перенавчання (overfitting) та недонавчання (underfitting). Перенавчання означає, що модель надто точно пристосовується до тренувальних даних, зокрема, запам’ятовує окремі зразки, що призводить до погіршення її здатності до узагальнення на нових зображеннях. В результаті точність

розпізнавання на валідних тестах падає, навіть якщо на тренувальному наборі вона залишається високою.

Навпаки, недонавчання спостерігається тоді, коли модель не має достатньої складності або тренування було недостатньо тривалим, що призводить до низької продуктивності і на тренувальних, і на тестових даних. У такому разі нейронна мережа не встигає навчитися виділяти характерні ознаки тексту в зображеннях.

Для зменшення впливу цих проблем потрібно ретельно налаштовувати параметри навчання, збільшувати розмір і різноманітність навчального набору, застосовувати методи регуляризації (наприклад, dropout) та обирати оптимальну архітектуру моделі. Виявлення та усунення перенавчання і недонавчання є критично важливими для підвищення якості та надійності системи розпізнавання текстів.

Загалом, виявлені проблеми вказують на необхідність удосконалення як самої моделі, так і підходів до попередньої обробки зображень, що значною мірою визначають кінцеву якість розпізнавання.

3.3.2 Шляхи покращення точності розпізнавання

На основі виявлених під час тестування проблем, сформовано низку технічних напрямів, які можуть суттєво покращити точність і надійність розпізнавання текстів. Удосконалення стосується як етапу підготовки даних, так і архітектурних аспектів моделі та її вихідного декодування.

Одним із найефективніших підходів є використання аугментації (augmentation) під час навчання моделі. Імітація розмиття, шуму, змін контрастності, поворотів і зсувів може зробити модель більш стійкою до реальних умов, де зображення тексту часто є далекими від ідеальних. Зокрема, застосування бібліотек типу imgaug або albumentations дозволяє створити набір випадкових трансформацій, які імітують фото з мобільного телефону або відскановані документи. Це дозволить зменшити

кількість помилок на розмитих або низькоякісних зображеннях, які були основною причиною поганої точності під час тестування.

Ще одним важливим покращенням є додавання українського алфавіту до словника моделі. Поточна реалізація підтримує лише латинські літери та цифри, що значно обмежує застосування розпізнавача для україномовного тексту. Розширення алфавіту потребує повторного навчання моделі, але дозволить коректно працювати з більш широким класом вхідних даних. Крім того, підтримка кількох мов може бути реалізована через окремі моделі або єдину багатомовну архітектуру з багатокласовим алфавітом.

Ще одна важлива сфера для покращення – використання більш досконалого методу декодування вихідного тензора моделі. Поточне рішення ґрунтується на жадібному (greedy) підході до CTC-декодування, коли для кожного кроку обирається клас із максимальною ймовірністю. Цей підхід є швидким, але не завжди точним, особливо при наявності помилок у передбачених класах. Альтернативою є beam search decoding – метод, що розглядає кілька найімовірніших послідовностей одночасно, з можливістю обрання найкращої не лише по локальних максимумах, а й по глобальному логарифму ймовірностей. У поєднанні з мовною моделлю beam search здатен значно покращити кінцевий результат без потреби змінювати архітектуру самої моделі.

Застосування вищезазначених методів дозволить підвищити адаптивність розпізнавача до реальних умов використання та розширити його функціональні можливості для україномовного середовища.

3.3.3 Розширення функціональності застосунку

Окрім покращення точності розпізнавання, важливим напрямом розвитку є розширення функціональності застосунку, зокрема Telegram-бота, з яким взаємодіє кінцевий користувач. Це дозволить зробити

інтерфейс більш зручним, інформативним та адаптованим до практичних потреб користувачів.

Одним із можливих покращень є виведення статистики розпізнавання. Після кожного запиту бот може надавати не лише результат розпізнавання, а й додаткові метрики, наприклад: довжину розпізнаного слова, середню впевненість моделі по всій послідовності; кількість пропущених або повторених символів (якщо є порівняння з еталоном).

Ця інформація може бути корисною для відлагодження та моніторингу ефективності системи, а також для аналітичних цілей.

Ще одним корисним доповненням є реалізація історії взаємодії користувача з ботом. Це дозволить зберігати розпізнані зображення, результати обробки та час запиту в базі даних (наприклад, SQLite або PostgreSQL). Надалі користувач зможе переглядати попередні результати, повторно завантажувати зображення чи експортувати історію у форматі CSV або PDF. Такий підхід буде особливо корисним для дослідників, аналітиків або осіб, що працюють з великою кількістю документів.

Також доцільним є виведення рівня впевненості моделі в розпізнаному результаті. На основі ймовірностей, які генерує модель для кожного символу, можна обчислювати загальну впевненість або ж підсвічувати символи з низькою достовірністю. Це дасть змогу користувачеві оцінити, наскільки надійним є результат розпізнавання, і за потреби вжити додаткових дій – надіслати зображення повторно або відредагувати текст вручну.

Окрему увагу слід приділити розширенню навчального набору даних. Для підвищення якості розпізнавання доцільно використати більший датасет, що містить не менше 20 тисяч слів, з урахуванням різноманітних шрифтів, форматів написання та мов. Це дозволить моделі краще узагальнювати дані та забезпечить вищу точність при розпізнаванні нових прикладів.

Крім того, поточну архітектуру можна адаптувати для розпізнавання не лише окремих слів, а й цілих речень або навіть абзаців. Такий функціонал значно розширить сферу застосування розробленої системи, дозволяючи використовувати її для обробки сканованих сторінок документів, вивісок або текстів на зображеннях складної структури.

Досягти цього можна шляхом попередньої сегментації зображень на рядки або текстові блоки перед передачею їх у модель, використання глибших архітектур на основі трансформерів, зокрема моделей типу TrOCR, що здатні працювати з довгими послідовностями, а також інтеграції мовних моделей або методів покращеного декодування, таких як beam search. Це дозволить значно підвищити точність та логічну узгодженість розпізнаного тексту і забезпечити перехід від рівня окремого слова до повноцінного зчитування структурованої текстової інформації.

3.4 Порівняння розробленої програми з існуючими OCR-системами

На сучасному етапі розвитку технологій розпізнавання тексту існує велика кількість готових OCR-систем, які демонструють високу точність та продуктивність у різних сферах. До таких систем можна віднести Google Tesseract, EasyOCR, Adobe OCR, ABBYY FineReader та інші. Кожна з цих систем має свої переваги й обмеження, що визначають доцільність її використання в конкретному контексті. У даному підрозділі буде розглянуто основні відмінності запропонованої користувацької системи розпізнавання тексту від найбільш поширених готових рішень.

Однією з ключових переваг власної реалізації є можливість повного контролю над архітектурою моделі та процесом її навчання. Завдяки цьому можна адаптувати систему під специфічні типи зображень, наприклад, фотографії з соціальних мереж, скановані документи з особливостями шрифтів, або низькоякісні зображення. У той час як Tesseract, хоча й

підтримує користувацьке тренування, демонструє дещо гіршу якість при обробці складних вхідних даних без попередньої оптимізації.

Застосування моделі CRNN у поєднанні з CTC Loss дозволяє ефективно працювати з нерозділеними послідовностями символів без потреби у попередній сегментації.

Це є важливою перевагою порівняно з класичними системами, які часто базуються на правилах або гібридних підходах зі складною попередньою обробкою. Наприклад, ABBYY FineReader базується на багаторівневій обробці документа і використовує велику кількість вбудованих евристик, які складно адаптувати або змінювати.

Крім того, запропоноване рішення інтегроване у Telegram-бот, що робить його зручним інструментом для швидкої перевірки текстів без потреби встановлювати додаткове програмне забезпечення.

На відміну від більшості готових систем, які мають складні інтерфейси або вимагають оплати ліцензій, розроблена система є повністю відкритою, налаштовуваною та безкоштовною у використанні.

Щодо точності, то при тестуванні на валідаційному наборі система показала задовільні результати, які можна покращити при збільшенні кількості навчальних прикладів і розширенні алфавіту.

У той час як, наприклад, EasyOCR демонструє вищу точність із коробки завдяки навчанню на величезному мультиалфавітному корпусі, розроблена модель має потенціал для покращення точності до аналогічного рівня завдяки персоналізованому навчанню на вузько спеціалізованих даних.

Таким чином, хоча сучасні OCR-системи є потужними й універсальними, власна реалізація на основі нейронної мережі CRNN забезпечує гнучкість, адаптивність та можливість розширення. Це робить запропоновану систему доцільною для дослідницьких цілей, освітніх задач або побудови прикладних рішень під специфічні вимоги користувача.

3.5 Етичні та соціальні аспекти використання OCR-систем

Використання систем оптичного розпізнавання символів на основі нейронних мереж, окрім технічних аспектів, має також етичне та соціальне значення. Одним із головних викликів є питання конфіденційності. OCR-системи часто обробляють документи, які можуть містити чутливу або персональну інформацію, включаючи паспортні дані, медичні записи, банківські документи. Відповідно, зберігання і обробка таких даних має здійснюватися з дотриманням сучасних вимог до захисту інформації, включаючи шифрування, обмеження доступу та анонімізацію.

Ще один аспект пов’язаний із можливими упередженнями в навчанні моделей. Якщо система тренується переважно на англомовних даних або прикладах з певного регіону, вона може демонструвати нижчу точність на документах іншими мовами або культурними особливостями форматування. Це викликає питання справедливості та інклюзивності систем розпізнавання.

У контексті автоматизації також постає соціальне питання – зменшення потреби в людському введенні даних може призводити до втрати робочих місць у деяких сферах. Проте водночас відкриваються нові можливості для спеціалістів, пов’язаних із підтримкою, навчанням та оптимізацією таких систем.

Загалом, етична відповідальність при створенні OCR-систем включає не лише технічну досконалість, а й прозорість рішень, захист приватності користувача та запобігання потенційному зловживанню технологією.

3.6 Вплив якості даних на точність розпізнавання

Якість вхідних зображень є одним із критичних факторів, що визначає ефективність роботи системи оптичного розпізнавання тексту. Недостатній контраст, розмитість, артефакти сканування або фотографування можуть

суттєво знизити точність моделі. Навіть незначні відхилення в освітленні або нахил тексту здатні ввести модель в оману.

Щоб мінімізувати вплив низькоякісних зображень, було реалізовано модуль попередньої обробки, що включає бінаризацію, нормалізацію контрасту, виправлення нахилу. Водночас, навіть ці кроки не завжди здатні повністю компенсувати проблеми з оригіналом.

Тому особливу увагу було приділено формуванню навчального набору даних, до якого увійшли зображення з різною якістю, фоном, шрифтами. Такий підхід дозволяє моделі навчитися працювати з широким діапазоном умов і зменшує ймовірність деградації результатів на реальних даних.

Крім того, було впроваджено аугментацію – штучне зниження якості зображення у тренувальному наборі шляхом додавання шуму, розмиття або обертання. Це допомагає зробити модель більш стійкою до непередбачених умов експлуатації.

3.7 Можливості масштабування та перенавчання моделі

Розроблена архітектура CRNN є модульною, що дозволяє легко масштабувати її під нові потреби. Наприклад, для додавання підтримки інших мов достатньо оновити алфавіт моделі та провести донавчання на відповідному датасеті. Завдяки використанню механізму CTC, модель здатна працювати з послідовностями довільної довжини, що значно спрощує масштабування.

Також можлива заміна рекурентного блоку на більш сучасні трансформери або впровадження attention-механізмів для підвищення точності при роботі з довгими рядками. Це дозволить моделі краще враховувати контекст і робити менше помилок при схожих символах.

Окремо варто зазначити можливість перенавчання моделі без повного скидання ваг. Для цього можна використовувати методи fine-tuning, які

дозволяють адаптувати вже існуючу модель до нових умов, не втрачаючи попередньо набуті знання. Це особливо корисно при адаптації до вузьких тематик або корпоративних документів.

3.8 Оцінка продуктивності системи на різних пристроях

У реальних умовах важливо забезпечити роботу системи розпізнавання тексту на пристроях з різною обчислювальною потужністю. Було проведено оцінку швидкодії на декількох типах обладнання – від ноутбука із середнім процесором до серверів із GPU.

На сучасному ноутбуці із 4-ядерним процесором без використання GPU середній час обробки одного зображення становив близько 2 секунд. Після оптимізації коду та використання конвертації моделі у формат TorchScript вдалося зменшити цей час до 1.2 секунди.

На GPU типу NVIDIA RTX 3060 час обробки становить менше 0.2 секунди, що робить можливим використання системи в режимі реального часу. Для мобільних пристроїв оптимізовано розмір моделі, а також розглянуто варіанти використання ONNX та TensorFlow Lite, що дозволяє запускати систему навіть на смартфонах із середніми характеристиками.

Таким чином, розроблена система може бути інтегрована як у серверні рішення, так і в локальні застосунки, забезпечуючи достатню швидкість обробки без значного зниження точності.

ВИСНОВКИ

У ході виконання роботи було досліджено та проаналізовано сучасні методи розпізнавання текстової інформації на основі нейронних мереж, що включають глибоке навчання, комп'ютерне бачення, та оптичне розпізнавання символів (OCR). Особливу увагу було приділено архітектурам нейронних мереж, таким як згорткові (CNN) та рекурентні (RNN), а також їх гібридним варіантам (CRNN), які ефективно поєднують виділення ознак та послідовний аналіз символів.

Постановка задачі полягала у створенні автоматизованої системи розпізнавання текстів із зображень, яка б могла бути використана у вигляді зручного інтерфейсу для кінцевого користувача – у вигляді Telegram-бота. Було сформовано вимоги до моделі, враховуючи необхідність високої точності, адаптивності до різних типів зображень та ефективності обробки.

В результаті було розроблено програмну реалізацію розпізнавача текстів на базі моделі CRNN із використанням функції втрат CTC для роботи з непередбачуваною довжиною послідовностей. Окремою складовою стало створення Telegram-бота, який дозволяє інтегрувати модель у зручний канал взаємодії з користувачем. Обробка зображень включала методи попередньої підготовки – конвертацію у відтінки сірого, бінаризацію, вирівнювання кута нахилу та масштабування з додаванням відступів.

Проведене тестування моделі на валідаційному наборі показало задовільний рівень точності з помірним рівнем помилок CER та WER. Проте, виявлено ряд проблем, серед яких повторювані символи у вихідних рядках, труднощі із розпізнаванням розмитих або низькоконтрастних зображень, а також наявність нерозпізнаних символів.

Запропоновано шляхи покращення моделі, серед яких: використання розширених датасетів із більшою кількістю прикладів (понад 20 тисяч слів), застосування методів аугментації даних для підвищення стійкості,

додавання підтримки українського алфавіту, а також впровадження більш складних алгоритмів декодування (наприклад, beam search) для зменшення кількості помилок.

Крім того, перспективним напрямом розвитку є розширення функціоналу застосунку, включаючи збір статистики користувачів, збереження історії сесій та виведення показників впевненості моделі. Модель потенційно можна розвинути для розпізнавання не лише окремих слів, а цілих речень чи навіть абзаців, застосовуючи техніки послідовного моделювання, такі як трансформери, а також методи обробки природної мови для корекції результатів.

Отже, виконана робота продемонструвала практичну можливість створення функціональної системи розпізнавання текстів на основі нейронних мереж із інтеграцією у Telegram-бота. Розроблена система має потенціал для подальшого вдосконалення і масштабування, що робить її актуальною та корисною в різних прикладних сферах, пов’язаних з автоматичною обробкою текстової інформації з зображень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bengio Y., Courville A., Goodfellow I. Deep learning. MIT Press, 2016. 800 p.

2. Character region awareness for text detection / Y. Baek et al. *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, Long Beach, CA, USA, 15–20 June 2019. 2019. URL: https://doi.org/10.1109/cvpr.2019.00959 (date of access: 20.06.2025).

3. Deep residual learning for image recognition / K. He et al. *2016 IEEE conference on computer vision and pattern recognition (CVPR)*, Las Vegas, NV, USA, 27–30 June 2016. 2016. URL: https://doi.org/10.1109/cvpr.2016.90 (date of access: 20.06.2025).

4. EAST: an efficient and accurate scene text detector / X. Zhou et al. *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, Honolulu, HI, 21–26 July 2017. 2017. URL: https://doi.org/10.1109/cvpr.2017.283 (date of access: 20.06.2025).

5. Generating chat bots from web API specifications / M. Vaziri et al. *SPLASH '17: conference on systems, programming, languages, and applications: software for humanity*, Vancouver BC Canada. New York, NY, USA, 2017. URL: https://doi.org/10.1145/3133850.3133864 (date of access: 20.06.2025).

6. Gradient-based learning applied to document recognition / Y. Lecun et al. *Proceedings of the IEEE*. 1998. Vol. 86, no. 11. P. 2278–2324. URL: https://doi.org/10.1109/5.726791 (date of access: 20.06.2025).

7. Graves A., Mohamed A.-r., Hinton G. Speech recognition with deep recurrent neural networks. *ICASSP 2013 – 2013 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, Vancouver, BC, Canada, 26–31 May 2013. 2013. URL: https://doi.org/10.1109/icassp.2013.6638947 (date of access: 20.06.2025).

8. Kai Wang, Babenko B., Belongie S. End-to-end scene text recognition. *2011 IEEE international conference on computer vision (ICCV)*, Barcelona, Spain, 6–13 November 2011. 2011. URL: https://doi.org/10.1109/iccv.2011.6126402 (date of access: 20.06.2025).

9. LeCun Y. The power and limits of deep learning. *Research-Technology management*. 2018. Vol. 61, no. 6. P. 22–27. URL: https://doi.org/10.1080/08956308.2018.1516928 (date of access: 20.06.2025).

10. Microsoft COCO: common objects in context / T.-Y. Lin et al. *Computer vision – ECCV 2014*. Cham, 2014. P. 740–755. URL: https://doi.org/10.1007/978-3-319-10602-1_48 (date of access: 20.06.2025).

11. PyTorch distributed / S. Li et al. *Proceedings of the VLDB Endowment*. 2020. Vol. 13, no. 12. P. 3005–3018. URL: https://doi.org/10.14778/3415478.3415530 (date of access: 20.06.2025).

12. Reading text in the wild with convolutional neural networks / M. Jaderberg et al. *International journal of computer vision*. 2015. Vol. 116, no. 1. P. 1–20. URL: https://doi.org/10.1007/s11263-015-0823-z (date of access: 20.06.2025).

13. Shi B., Bai X., Yao C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. *IEEE transactions on pattern analysis and machine intelligence*. 2017. Vol. 39, no. 11. P. 2298–2304. URL: https://doi.org/10.1109/tpami.2016.2646371 (date of access: 20.06.2025).

14. Smith R. An overview of the tesseract OCR engine. *Ninth international conference on document analysis and recognition (ICDAR 2007) vol 2*, Curitiba, Parana, Brazil, 23–26 September 2007. 2007. URL: https://doi.org/10.1109/icdar.2007.4376991 (date of access: 20.06.2025).

15. Yunus Karaaslan, Ugur Gelisken, Birol Kuyumcu. OpenCv. Level, 2022. 320 p.

16. Cem Dilmegani. OCR Benchmark: Text Extraction / Capture Accuracy 2025. URL: https://research.aimultiple.com/ocr-accuracy/ (date of access: 05.05.2025)

17. Pankaj Tripathi. The Brief History of OCR Technology. URL: https://www.docsumo.com/blog/optical-character-recognition-history (дата звернення: 06.05.2025)

18. Preeti Kulkarni. OCR In Banking: Role, Applications, and Impact. URL: https://hyperverge.co/blog/ocr-banking/ (date of access: 08.05.2025)

19. Krishi Chowdhary. Best OCR software of 2025. URL: https://www.techradar.com/best/best-ocr-software (date of access: 09.05.2025)

20. Vihar Kurama. PyTorch vs. TensorFlow: Key Differences to Know for Deep Learning. URL: https://builtin.com/data-science/pytorch-vs-tensorflow (date of access: 11.05.2025)