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 мобільного застосунку для конвертації рукописного тексту та математичних формул у цифровий формат з контекстною автокорекцією
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-4
Андрій Носарєв
(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 ас. Микола Черненко
(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 Носарєву Андрію Едуардовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка мобільного застосунку для конвертації рукописного тексту та математичних формул у цифровий формат з контекстною автокорекцією

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 вимоги до функціоналу мобільного додатку для розпізнавання рукописного тексту, наявні методи OCR та NLP, публічні датасети для навчання моделей розпізнавання тексту, моделі архітектур глибокого навчання CNN, BiLSTM та Transformer, інструмент для розробки кросплатформених застосунків Flutter, інструмент для розробки серверної логіки FastAPI та баз даних PostgreSQL.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі дослідження
2) Проектування архітектури застосунку та серверної частини
3) Розробка моделей штучного інтелекту
4) Розробка мобільного застосунку

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної області	23.05.2025	виконано
3	Постановка задачі	24.05.2025	виконано
4	Проектування архітектури застосунку	26.05.2025	виконано
5	Проєктування бази даних	28.05.2025	виконано
6	Створення україномовного датасету	01.06.2025	виконано
7	Розробка моделей штучного інтелекту	10.06.2025	виконано
8	Розробка мобільного застосунку	20.06.2025	виконано
9	Захист перед ЕК	25.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Микола Черненко
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 68 с., 24 рис., 2 дод., 20 джерел.

НЕЙРОННІ МЕРЕЖІ, РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ, CNN, CTC, FASTAPI, FLUTTER, LSTM, OCR, POSTGRESQL, TRANSFORMERS.

Об’єкт дослідження – процес конвертації рукописного тексту та фомул в цифрову форму з урахуванням помилок, що виникають під час автоматичного розпізнавання..

Предмет дослідження – методи оптичного розпізнавання та контекстної автокорекції україномовного рукописного тексту з використанням нейронних мереж і сучасних мовних моделей.

Мета роботи – розробка мобільного застосунку для розпізнавання україномовного рукописного тексту та формул з використанням сучасних моделей глибокого навчання та реалізацією контекстної автокорекції результатів розпізнавання.

Методи дослідження – теоретичні та експериментальні методи штучного інтелекту, зокрема підходи машинного навчання для обробки зображень і природної мови, методи побудови багаторівневих програмних систем, а також принципи розробки інтерфейсів користувача для мобільних платформ.

У результаті роботи було проаналізовано актуальні задачі та виклики розпізнавання україномовного рукописного тексту, досліджено сучасні методи оптичного разпізнавання та корекції тексту. Було розроблено клієнт-серверний із мобільним інтерфейсом, який дозволяє конвертувати україномовний рукописний текст у цифровий формат із можливістю автоматичної контекстної автокорекції.

ABSTRACT

Bachelor’s thesis contains: 68 pp., 24 fig., 2 ann., 20 references.

CNN, CTC, FASTAPI, FLUTTER, HANDWRITTEN TEXT RECOGNITION, LSTM, NEURAL NETWORKS, OCR, POSTGRESQL, TRANSFORMERS.

Object of research – the process of converting handwritten text and formulas into digital form, taking into account errors that occur during automatic recognition.

Subject of research – methods of optical recognition and contextual auto-correction of Ukrainian handwritten text using neural networks and modern language models.

Purpose of the work – development of a mobile application for recognizing Ukrainian handwritten text and formulas using state-of-the-art deep learning models and implementing contextual auto-correction of recognition results.

Research methods – theoretical and experimental methods of artificial intelligence, including machine learning approaches for image and natural language processing, methods for building multi-level software systems, as well as principles of user interface development for mobile platforms.

As a result of the work, current challenges and issues in Ukrainian handwritten text recognition were analyzed, and modern methods of optical recognition and text correction were studied. A client-server system with a mobile interface was developed, enabling the conversion of Ukrainian handwritten text into digital format with automatic contextual auto-correction.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Бенчмаркінг – benchmarking – процес вимірювання ефективності (точності, швидкодії тощо) програмного забезпечення або моделей штучного інтелекту шляхом порівняння з еталонними показниками або іншими системами.

Датасет – dataset – структурований набір даних, що використовується для навчання, валідації або тестування моделей машинного навчання.

Донавчання – fine-tuning – це навчання моделі після її попереднього переднавчання.

Епоха – в контексті штучного інтелекту, один повний прохід усіх навчальних даних через модель під час її тренування.

Переднавчання – це процес навчання моделі «з нуля»: ваги моделі випадково ініціалізуються, після чого розпочинається навчання без попередніх налаштувань.

AI – Artificial Intelligence – штучний інтелект, галузь комп’ютерних наук, що займається створенням систем, здатних до навчання, адаптації та імітації людської поведінки.

Batch processing – батч-обробка – метод автоматизованої обробки групи завдань у вигляді єдиного пакета без участі користувача.

LaTeX – система комп’ютерної верстки, орієнтована на створення наукових і технічних документів з підтримкою математичної нотації.

OCR – Optical Character Recognition – технологія автоматичного розпізнавання тексту на зображеннях або відсканованих документах.

Transformer – архітектура нейронної мережі, що використовує механізм «уваги» для ефективного оброблення послідовностей без рекурентних з'єднань; є основою багатьох сучасних моделей обробки природної мови.

ВСТУП

У сучасному світі, де інформаційні технології стрімко розвиваються, а рівень цифровізації невпинно збільшується, зростає потреба в ефективних засобах для обробки та збереження даних. Одна з важливих задач полягає в автоматизації процесу переходу із використання паперових до використання цифрових документів, тобто перетворення рукописного тексту в цифровий формат. Хоча існує безліч програмних рішень для розпізнавання тексту, більшість з них орієнтовані на друковані тексти та часто не мають підтримки української мови, що залишає завдання розпізнавання україномовного рукописного тексту складним та трудомістким завданням. Особливо важливим є створення рішень, які враховують контекст при розпізнаванні та здатні автоматично коригувати помилки, що неминуче виникають у процесі розпізнавання.

Дана робота присвячена дослідженню процесу розпізнавання рукописних текстів та можливостей їх автокорекції з урахуванням контексту. Основна увага зосереджена на роботі із україномовними рукописними текстами, а також на створенні мобільного додатка їх обробки. У процесі роботи буде використано сучасні методи оптичного розпізнавання символів, алгоритми мовної обробки тексту та підходи до реалізації інтерфейсів користувача у мобільних додатках.

Процес обробки тексту при роботі з будь-яким сервісом розпізнавання тексту зазвичай можна поділити три основні етапи. Спочатку користувач вибирає зображення документа. Потім відбувається розпізнавання тексту на сервері, де зображення перетворюється на цифровий текст за допомогою технології OCR. На завершальному етапі користувач редагує отриманий текст і зберігає результат. Практика показує, що саме третій етап – редагування – часто займає більше часу, аніж усі попередні етапи разом, тому мінімізація часу його виконання є однією з ключових цілей цієї роботи.

Метою даної роботи є розробка мобільного застосунку, здатного ефективно конвертувати рукописний текст та математичні формули у цифровий формат з підтримкою контекстної автокорекції.. Для досягнення поставленої мети планується використання моделей трансформерів, таких як TrOCR та BERT а також традиційних моделей глибокого навчання на основі CNN + LSTM + CTC .

Завданням цієї роботи є розробка як серверної, так і клієнтської частини системи. На серверній стороні буде використовуватись локальна система для розпізнавання рукописного тексту та контекстної автокорекції. Клієнтська частина буде реалізована за допомогою технології Flutter – кросплатформного фреймворку, що забезпечить зручний та інтуїтивно зрозумілий інтерфейс для користувачів Android-пристроїв.

Розроблена система може бути корисна у галузях, які потребують інтенсивної роботи із нестандартизованими рукописними документами, такими як освітня та наукова, а також при індивідуальному використанні.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Опис предметної галузі

1.1.1 Історичний контекст

Перші спроби розв’язання проблеми оптичного розпізнавання символів (OCR) сягають початку XX століття, коли в 1914 році фізик Емануель Гольдберг представив пристрій, здатний розпізнавати друковані символи та перетворювати їх у телеграфний код [1]. Цей апарат, який у 1931 році отримав патент на «Статистичну машину», вважався першим механізмом для автоматичного пошуку та ідентифікації символів, попередником сучасного OCR. У 1950–60-х роках ця технологія стала комерційною: IBM створила пристрій IBM 1287 (1966) [2], а в 1968 році були стандартизовані шрифти OCR-A/B.

Із розвитком технологій OCR з’явилася потреба у спеціалізованих датасетах для навчання та бенчмаркінгу, або ж вимірювання точності моделей. MNIST – один із перших датасетів машинного навчання із рукописними цифрами, узагальнено опублікований влітку 1994 року й широко використаний з 1998 року як стандартний бенчмарк [3].

Найбільш відомий англомовний набір даних IAM Handwriting Database був представлений У. Марті та Г. Бунке під час конференції ICDAR-1999, та містить понад 1500 сторінок англійського рукопису із 13353 рядками та 115320 словами [4].

Поява таких датасетів значно прискорила розвиток машинного рукописного OCR та стимулювала підвищений інтерес до досліджень у цій сфері.

1.1.2 Сучасні методи розпізнавання тексту

На відміну від традиційних методів, які зосереджені на сегментації окремих символів, сучасні методи фокусують свою увагу на розпізнаванні всіх символів у сегментованому рядку або слові тексту.

Поява двонаправлених рекурентних нейронних мереж (BiLSTM) у поєднанні з функцією втрат Connectionist Temporal Classification (CTC) стала важливим етапом у розвитку систем оптичного розпізнавання тексту. Такий підхід уперше був представлений у роботі Алекса Ґрейвза [5], де було запропоновано ефективне розпізнавання послідовностей змінної довжини без необхідності попереднього сегментування на символи.

Двонаправленість у LSTM-мережах дозволяє враховувати контекст як зліва, так і справа від символу, що особливо важливо для рукописного тексту, де межі між літерами нечіткі. Алгоритм CTC забезпечує відповідність між вхідними часовими ознаками та послідовністю символів на виході без необхідності точної розмітки по символах.

Подальший розвиток технологій привів до появи моделей, що базуються на архітектурі трансформерів. У 2021 році Li та співавт. представили TrOCR – повністю трансформерну архітектуру для оптичного розпізнавання тексту [6]. Модель складається з Vision Transformer (ViT) як енкодера та стандартного Transformer-декодера, попередньо навчена на великих обсягах синтетичних та реальних даних. У своїй роботі автори продемонстрували перевагу TrOCR над класичними підходами CNN+RNN у задачах розпізнавання друкованого, рукописного та сценічного тексту, що підтверджує ефективність трансформерів у цій галузі.

Останнім часом активно розвивається напрям використання великих мовних моделей (LLM) безпосередньо для OCR. Нещодавні роботи демонструють, що моделі на зразок GPT-4V та Gemini можуть виконувати OCR із високою якістю контекстуального розуміння [7]. Це відкриває нову еру «OCR як частина візуального знання» – де текст і контекст трактуються

єдиним модулем, що веде до значного підвищення точності та гнучкості застосувань.

Таким чином, сучасний OCR пройшов шлях від класичних моделей, що потребували попереднього сегментування, до універсальних архітектур, здатних працювати з неструктурованими даними. Комбінація BiLSTM та CTC продовжує залишатися ефективним рішенням для багатьох застосунків, водночас моделі на основі трансформерів демонструють перспективи в складних умовах розпізнавання.

1.1.3 Виклики та проблеми в обробці рукописного тексту

Обробка рукописного тексту все ще залишається однією з найскладніших задач у сфері оптичного розпізнавання символів. Основна складність полягає в тому, що рукописний текст значно варіативніший, ніж друкований. Індивідуальні особливості почерку, різні стилі написання букв, злиття символів, пропуски та викривлення створюють додаткові труднощі при спробі автоматичного розпізнавання.

Особливої уваги також потребує якість вхідного зображення: низька роздільна здатність, погане освітлення, тіні, шуми, розмиття, неоднорідність фону на зображенні, наприклад тексти в зошитах – все це суттєво впливає на точність розпізнавання. У мобільному середовищі, де документи часто фотографуються на ходу, ці проблеми виникають регулярно.

Ще одним викликом є відсутність явної структури у рукописному тексті. В таких документах часто відсутні чіткі розділення між абзацами, реченнями або заголовками, що ускладнює побудову логічної структури тексту після розпізнавання. Крім того, у рукописах можуть поєднуватися різні типи інформації – текст, формули, рисунки, що вимагає попереднього аналізу типу вмісту.

1.1.4 Особливості україномовного OCR

Попри загальний поступ у розвитку систем оптичного розпізнавання рукописного тексту, україномовне OCR довгий час залишалося недостатньо дослідженим напрямом. Причиною цього були як обмеженість мовних ресурсів, так і нечисельність високоякісних рукописних україномовних датасетів. Оскільки більшість комерційних або академічних систем були орієнтовані на англомовні або китайсько-японські тексти, українська мова опинялася поза основним фокусом розробників.

З переходом до глибинного навчання підтримка української мови почала покращуватися, однак на академічному рівні прогрес був обмежений через відсутність відкритих датасетів. Наприклад, такі широко використовувані набори, як IAM чи RIMES, не містять кириличних записів. У відповідь на цю проблему були спроби створення україномовних наборів, однак вони, як правило, не публікувались у відкритому доступі або мали обмежене використання.

1.1.5 Контекстна автокорекція та її важливість

Навіть за умови використання сучасних алгоритмів OCR, результат автоматичного розпізнавання рукописного тексту рідко буває ідеальним. Виявлення та виправлення помилок, що виникають у процесі розпізнавання, є обов’язковим етапом обробки тексту. Проте ручне редагування вимагає часу та зусиль користувача, особливо коли йдеться про великі обсяги тексту або погану якість почерку. Саме тому з’являється потреба у використанні методів інтелектуальної післяобробки, зокрема контекстної автокорекції.

Контекстна автокорекція – це процес виправлення помилок у тексті на основі мовного контексту, тобто аналізу навколишніх слів, граматичних конструкцій, частотності вживання слів та фраз. На відміну від простих словникових перевірок, які виявляють лише орфографічні невідповідності,

контекстні моделі можуть знаходити та виправляти граматичні помилки, омоніми, неправильне вживання форм слова тощо.

Застосування таких підходів базується на використанні моделей обробки природної мови (NLP), серед яких особливу роль відіграють трансформери – архітектури, здатні враховувати широкий контекст завдяки механізму уваги. Наприклад, моделі на зразок BERT чи GPT можуть ефективно передбачати ймовірні слова на основі контексту, допомагаючи замінити невірно розпізнані фрагменти тексту на правильні.

Особливо важливою є контекстна автокорекція для мов, які мають складну морфологію та гнучкий порядок слів – саме до таких належить українська мова. У таких випадках помилки OCR можуть бути складними для виявлення без урахування ширшого контексту – наприклад, коли помилково розпізнане слово має правильну орфографію, але не відповідає змісту речення або коли результат розпізнавання настільки зашумлений, що може бути складно визначити розпізнане слово простою словниковою перевіркою.

1.2 Огляд наявних рішень та їхніх обмежень

З розвитком мобільних платформ та хмарних сервісів, у користувачів з’явилась можливість розпізнавати рукописний текст безпосередньо зі смартфона або через вебінтерфейси. У цьому підрозділі розглянуто п’ять актуальних рішень – два мобільні застосунки, та три мультимодальні LLM-сервіси.

Handwriting to Text (Apptico) – мобільний застосунок, безпосередньо направлений на розпізнавання рукописного тексту. Він також підтримує розпізнавання математичних формул у форматі LaTeX. Має хорошу точність та швидкість роботи, однак головним недоліком є обмеженість безкоштовного варінту. Застосунок має обмеження на кількість

безкоштовних сканувань, також якість розпізнавання помітно знижується за наявності дефектів зображення, таких як тіні та перекоси.

На рисунку 1.1 зображено модальне вікно, яке з’являється після вичерпання всіх базкоштовних сканів.

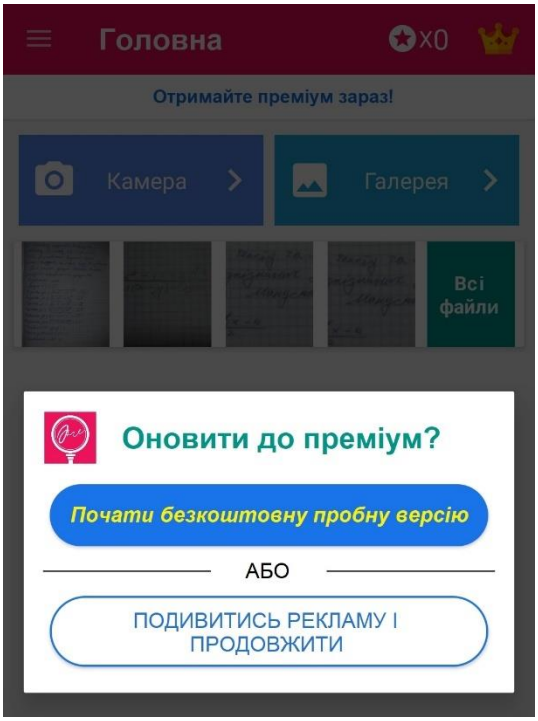

Рисунок 1.1 – Модальне вікно застосунку Handwriting to Text

Pen to Print (Serandi Ltd.) – один із найпопулярніших застосунків серед користувачів Android та iOS для розпізнавання рукописного тексту. Застосунок показує задовільні результати на текстових фрагментах, хоча поступається попередньому за точністю. Основним обмеженням є відсутність підтримки математичних формул, а також наявність платного тарифного плану для експорту результатів у зовнішні формати. На рисунку 1.2 продемонстровано результат обробки зображення, що містить текст та формулу.

Наступними було розглянуто використання мультимодальних моделей, здатних сприймати зображення та витягати текст, включаючи рукописний. Було розглянуто три популярні AI сервіси: ChatGPT, Google

Gemini та Mistral. Результати їх роботи можна охарактеризувати наступним чином: Google Gemini мала найгірші результати, іноді із доволі значною кількістю помилок. ChatGPT та Mistral забезпечили надзвичайно високу точність розпізнавання як текстових фрагментів, так і частково математичних виразів. При цьому варто зазначити, що Mistral має перевагу у відкритості моделі та можливості автономного запуску, тоді як ChatGPT доступний переважно як хмарний сервіс. Основним обмеженням усіх LLM-рішень є наявність лімітів у безкоштовних версіях, що може обмежити їхнє практичне використання в мобільному застосунку без спеціальної інтеграції або передплати.

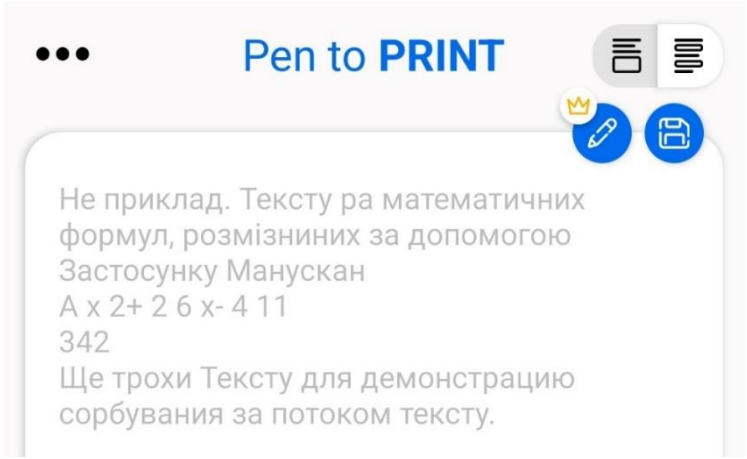

Рисунок 1.2 – Результат обробки зображення застосунком Pen to Print

Отже, наявні продукти на ринку не виконують свої функції у повному обсязі, часто демонструють недостатню точність розпізнавання та майже завжди обмежені умовами безкоштовного використання. Таким чином, існує потреба у створенні рішень, здатних ефективно працювати з україномовним рукописним текстом, забезпечуючи як якісне розпізнавання, так і інтелектуальну післяобробку із можливістю локального запуску серверу без необхідності до доступу до хмарних OCR-сервісів. У цьому контексті розробка мобільного застосунку з підтримкою контекстної автокорекції та розпізнаванням математичних формул є важливим

завданням, яке допоможе спростити доступ звичайних людей до технології OCR. Подібний застосунок дозволить розширити перспективи для спрощення щоденної роботи студентів, викладачів, науковців та всіх, хто має справу з рукописним введенням тексту.

1.3 Аналіз технологічних викликів

1.3.1 Відсутність якісних україномовних датасетів

Як вже було зазначено вище, однією з головних проблем розвитку систем оптичного розпізнавання рукописного тексту українською мовою є брак відкритих та якісно розмічених датасетів. Існуючі датасети (як-от IAM, MNIST) орієнтовані переважно на англійську мову або цифрові символи, а для української мови або складних документів із формулами відповідних наборів даних практично немає. Це суттєво ускладнює навчання та тестування моделей, а також обмежує можливості глибокого навчання. Тому залишається декілька способів вирішення даної проблеми, таких як ручний збір прикладів україномовного рукопису та подальша їх обробка, або генераця синтетичного датасету із використанням різноманітних українських рукописних шрифтів. Також доступний гібридний підхід.

1.3.2 Обробка вхідних даних

Розпізнавання починається з отримання якісного вхідного сигналу – зображення рукописного тексту. Вхідні зображення часто мають шум, нерівномірне освітлення, перекоси, тіні, розмиття тощо. Тому важливими етапами є попередня обробка (preprocessing): корекція геометричних викривлень, шумозаглушення, нормалізація яскравості та контрасту. Ці операції суттєво впливають на точність подальшого розпізнавання. Після обробки зображення відбувається сегментація тексту – розбиття на рядки,

слова або символи. Вибір стратегії сегментації залежить від архітектури моделі: класичні методи потребують чіткої сегментації символів, сучасні – можуть працювати із рядками чи навіть блоками тексту без сегментації на символи. Сортування сегментованих елементів у правильному порядку – окремий виклик, особливо для рукописного тексту, де орієнтація та розміщення символів можуть бути нерегулярними. Потрібно розробити алгоритми, які відновлюють правильну послідовність символів та рядків.

1.3.3 Архітектурні варіанти моделей

Згорткові нейронні мережі, також відомі як Convolutional Neural Networks (CNN), спеціалізуються на обробці зображень [8]. Головною особливістю, що відрізняє CNN від інших нейронних мереж, є операція згортки. Ця операція є потужним інструментом, який дозволяє мережі ефективно виявляти важливі ознаки на зображеннях, незалежно від їхнього розташування. Процес згортки в CNN полягає у застосуванні фільтра, також відомого як «ядро згортки», до зображення. Фільтр являє собою невелику матрицю, яка множиться на пікселі зображення. Переміщуючись по зображенню, фільтр аналізує невеликі ділянки зображення послідовно, що дозволяє виявляти локальні особливості (рисунок 1.3).

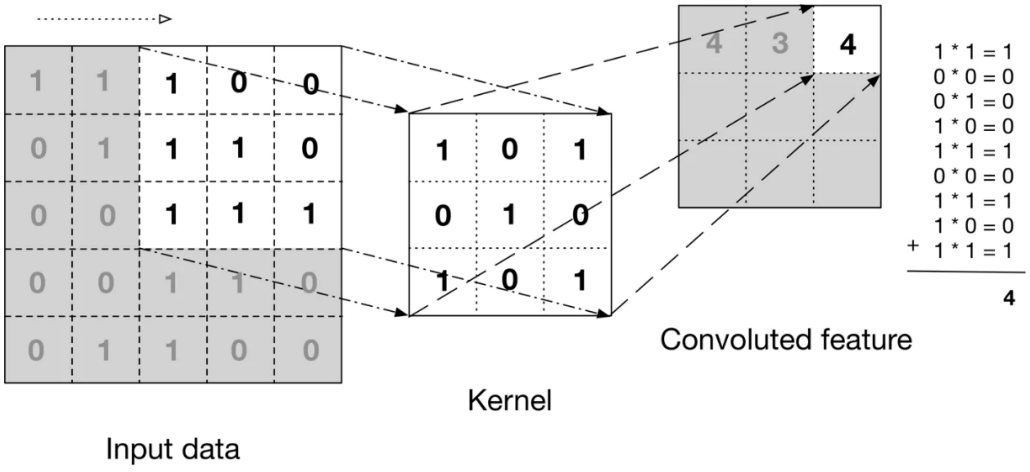

Рисунок 1.3 – Приклад роботи CNN

LSTM (Long Short-Term Memory) – це різновид рекурентної нейронної мережі, який вирішує проблему короткої пам’яті, притаманну звичайним RNN (рисунок 1.4). Його головною особливістю є механізм збереження інформації у вигляді стану пам’яті (cell state), який передається від кроку до кроку майже без змін, за винятком впливу вентилів. Ці вентилі контролюють потік інформації: вони визначають, що слід залишити, що стерти, і що додати до пам’яті. Завдяки цьому модель може зберігати важливу інформацію протягом багатьох часових кроків, уникаючи затухання або вибуху градієнтів.

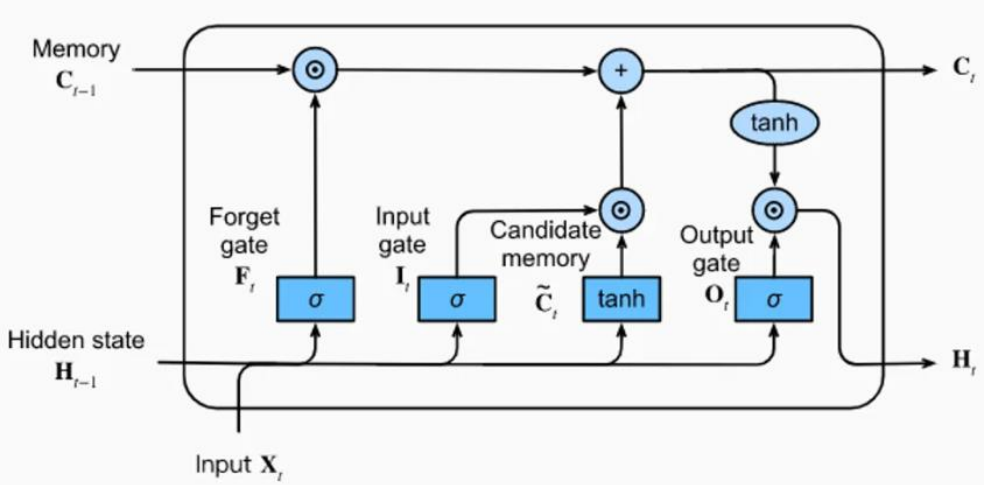

Рисунок 1.4 – Приклад архітектури LSTM

На кожному кроці роботи LSTM попередній прихований стан і поточне значення на вході використовуються для того, щоб вирішити, яку частину попередньої пам’яті слід забути, яку нову інформацію зберегти, і що передати як вихід поточного кроку. Таким чином, мережа динамічно навчається зберігати контекст, який є важливим, і ігнорувати те, що є зайвим. Це дозволяє LSTM ефективно працювати з довгими послідовностями, наприклад, реченнями чи рядками рукописного тексту, і зберігати смисл навіть за великої кількості попередніх елементів [9].

Алгоритм Connectionist Temporal Classification (CTC) дозволяє навчати нейромережі зі змінною довжиною вхідних та вихідних послідовностей без потреби у точному вирівнюванні. Ідея полягає в тому, що на кожному часовому кроці мережа отримує розподіл ймовірностей над усіма символами разом із спеціальним пустим токеном, що позначає відсутність символу. Потім розглядаються всі можливі вирівнювання, які після видалення цих пустих символів і злиття повторів можуть породити бажану послідовність символів. Шляхом сумування ймовірностей по всіх таких вирівнюваннях отримується ймовірність вихідної послідовності. При виведенні використовують пошук за променем(beam search) або просте детектування максимальної ймовірності на кожному кроці, після чого видаляють пусті символи і повтори [10].

Трансформери – це архітектура, запропонована Васьвані у 2017 році у статті «Attention Is All You Need»[11] , яка відмовилася від традиційної рекурентної або згорткової обробки послідовностей на користь механізму самоуваги (self-attention). У моделі є дві основні частини: енкодер і декодер. Енкодер складається із стеку ідентичних блоків, кожен з яких має шар самоуваги, позиційне нормування, а також позиційно-залежну мережу (feed-forward). Декодер додатково використовує шар крос-уваги, що дозволяє звертатися до виходів енкодера під час генерації відповіді. Цей підхід усуває залежності від порядку обробки, дозволяючи значно паралелізувати навчання й обробку даних. На рисунку 1.5 зображено вигляд базової архітектури трансформів.

Важливою перевагою трансформерів є їх особливості навчання. Спочатку модель переднавчають. Цей етап є дорогим як за часом, так і за коштами. Переднавчання зазвичай виконується на величезних обсягах даних, а сам процес може тривати кілька тижнів. Після цього здійснюється донавчання для спеціалізації моделі для роботи із конкретним завданням. Оскільки переднавчена модель уже «бачила» велику кількість даних, процес донавчання потребує значно меншого обсягу нових даних для досягнення

прийнятних результатів. З цієї ж причини донавчання потребує набагато менше часу для отримання хороших результатів.

Таким чином, донавчання вимагає менше часу, менше даних, менше фінансових і навіть екологічних витрат. Воно також дає змогу швидше і з меншими зусиллями експериментувати з різними варіантами навчання, оскільки не потребує повного переднавчання з нуля. Цей підхід також зазвичай забезпечує кращі результати, ніж навчання з нуля (якщо тільки у вас немає надзвичайно великого набору даних). Тому завжди варто намагатись використовувати вже переднавчену модель – максимально близьку до бажаного завдання – і донавчати її.

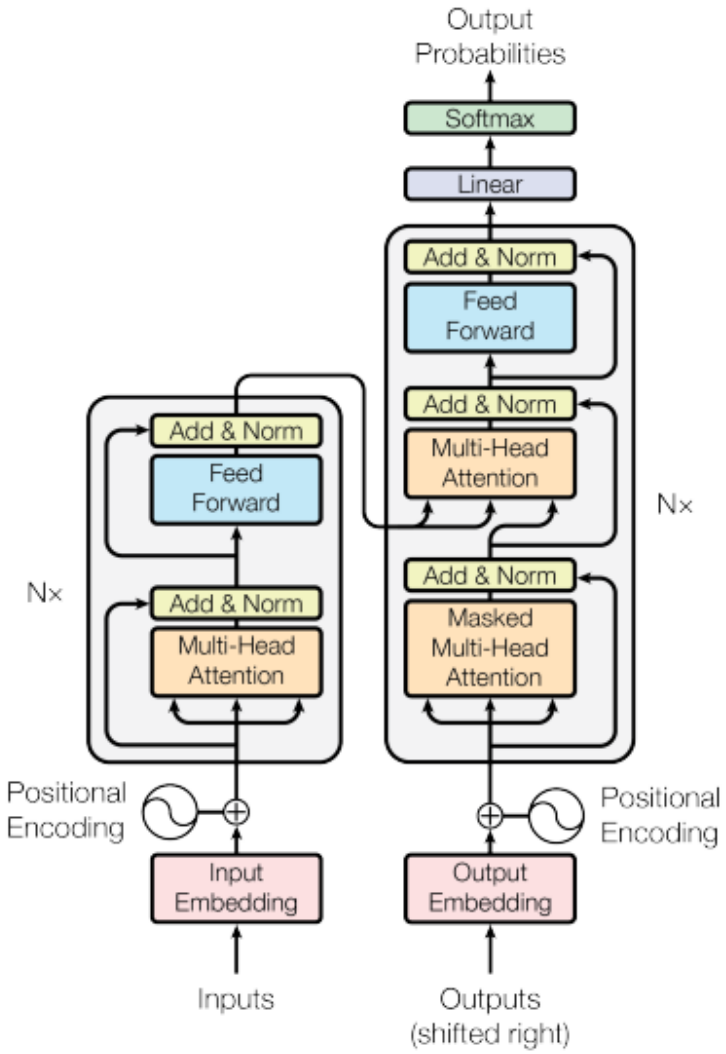

Рисунок 1.5 – Базова архітектура трансформера

1.4 Постановка задачі

Отже, проаналізувавши предметну область, протестувавши аналоги та визначивши всі технологічні виклики, можна чітко поставити задачу дослідження.

В контексті розглянутих викликів, актуальною є задача розробки мобільного застосунку, здатного ефективно розпізнавати україномовний рукописний текст (у тому числі математичні формули) з використанням сучасних підходів комп'ютерного зору та природньої мовної обробки. Для даної системи має бути розроблено:

– автоматичну сегментацію тексту для подальшої передачі до OCR моделі;

– модель розпізнавання україномовного тексту;

– модель розпізнавання базових математичних формул

– модель корекції тексту після розпізнавання;

– клієнт-серверний застосунок, що керуватиме роботою застосунку;

– збереження даних користувачів у база даних;

– реалізацію захищених інструментів реєстрації та авторизації користувачів.

1.5 Висновки за розділом

У першому розділі було здійснено комплексний аналіз предметної області оптичного розпізнавання рукописного тексту, що охоплює історичний процес розвитоку технології, було проаналізовано сучасні мобільні застосунки та мультимодальні моделі, а також вивчено технічні виклики, які постають при розробці систем розпізнавання, орієнтованих на українську мову.

Результати аналізу засвідчили, що попри наявність комерційних рішень для розпізнавання тексту, жодне з них не задовольняє повною мірою

потреб користувачів, які працюють з українським рукописним текстом і математичними формулами. Наявні інструменти мають обмежену функціональність, залежать від хмарних сервісів і часто вимагають платної підписки для повноцінного використання.

Було виокремлено ключові технологічні виклики, зокрема: відсутність якісних відкритих україномовних датасетів, складність сегментації рукописного тексту, вибір ефективної архітектури для розпізнавання, а також необхідність контекстної післяобробки результатів. На основі отриманих висновків сформульовано чітке завдання дослідження та визначено основні етапи реалізації проєкту.

Таким чином, розробка автономного мобільного застосунку для розпізнавання українського рукописного тексту з підтримкою формул і контекстної автокорекції є актуальною та технічно доцільною задачею, що має як наукову, так і практичну цінність.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ЗАСТОСУНКУ ТА СЕРВЕРНОЇ ЧАСТИНИ

2.1 Архітектура застосунку

Проєктування архітектури застосунку є критичним етапом у реалізації системи для розпізнавання рукописного тексту з контекстною автокорекцією. У межах цієї роботи буде розроблено прототип, що реалізує основні функціональні компоненти системи, зосереджуючись на практичній придатності, стабільності та розширюваності.

Система розпізнавання рукописного тексту з контекстною автокорекцією проєктується за класичною клієнт-серверною архітектурою. Основними її компонентами є мобільний клієнт, сервер обробки даних та допоміжні програмні модулі, відповідальні за розпізнавання тексту (OCR) і мовну післяобробку. Зокрема, необхідно буде впровадити наступні пов’язані з OCR модулі: модуль сегментації тексту, модуль розпізнавання (OCR), модуль контекстної автокорекції та модуль розпізнавання математичних формул.

Оскільки основною метою проєкту є створення наукового прототипу з можливістю локального розгортання, допускаються певні спрощення в порівнянні з комерційними бізнес-рішеннями – зокрема, обмежена багатокористувацька підтримка, спрощені процедури захисту API, відсутність масштабування через мікросервіси та спрощене управління сесіями. У майбутньому ці компоненти можуть бути вдосконалені, однак на даному етапі акцент зроблено на функціональності та автономності.

Для кращого розуміння та документування архітектури системи буде використано підхід C4 [12], який дозволяє послідовно моделювати систему на чотирьох рівнях абстракції: контекст, контейнери, компоненти та код.

У цій роботі будуть представлені перші три рівні діаграм: контекст, контейнери та компоненти. На рисунку 2.1 наведено діаграму системного контексту застосунку.

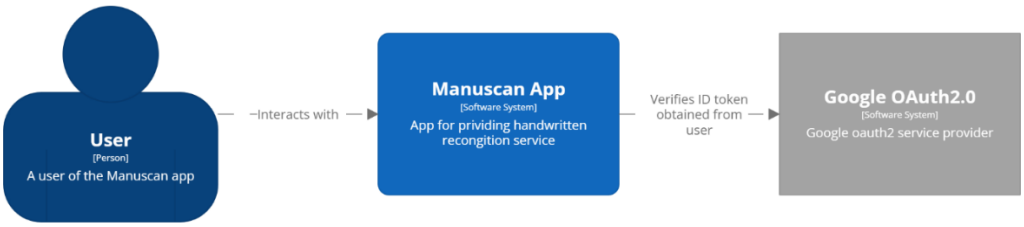

Рисунок 2.1 – Діаграма системного контексту

Дизайн затосунку направлений на забезпечення максимальної автономності, тому об’єм взаємодії із зовнішніми сервісми зведений до мінімуму. Із діаграми видно, єдиний використаний зовнішній сервіс – це Google OAuth2.0 для полегшення авторизації в застосунку.

Щоб глибше та дітальніше описати систему необхідно перейти на рівень нижче (рисунок 2.2).

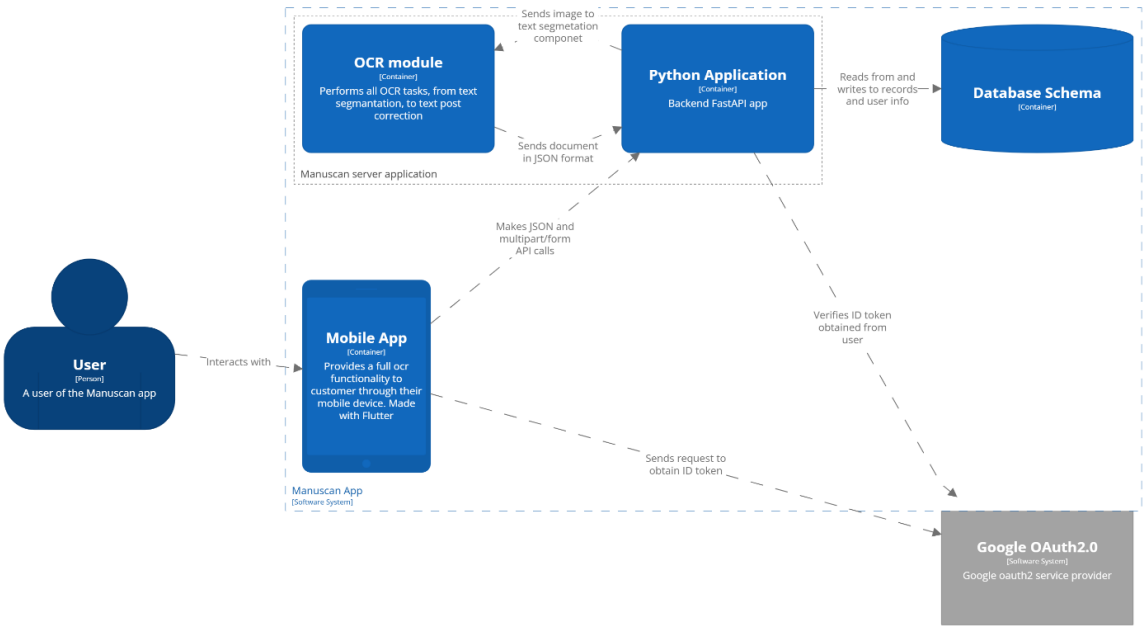

Рисунок 2.2 – Контейнерна діаграма

На контейнерному рівні діаграми C4 представлено архітектуру основних логічних блоків (контейнерів), що реалізують ключові функціональні можливості застосунку. Цей рівень деталізує, як основна система взаємодіє з користувачами та сторонніми сервісами, а також як внутрішні модулі обмінюються даними між собою.

Центральним елементом є Manuscan App – система для розпізнавання рукописного тексту з контекстною автокорекцією. Вона складається з декількох основних контейнерів.

Mobile App – клієнтська частина, що встановлюється на пристрій користувача. Цей контейнер забезпечує користувацький інтерфейс, через який користувач може завантажувати зображення рукописного тексту, переглядати результати розпізнавання, керувати історією обробок та здійснювати автентифікацію. Він ініціює запити до серверної частини та приймає відповіді у структурованому форматі.

Серверна частина – центральний логічний контейнер, який забезпечує обробку клієнтських запитів, автентифікацію користувачів, керування сесіями, зберігання та отримання даних, а також маршрутизацію запитів до спеціалізованих модулів розпізнавання. Саме сервер забезпечує координацію між усіма компонентами системи.

Модуль OCR – окремий контейнер, відповідальний за повний цикл обробки зображень: від вхідного зображення і до вихідного структурованого тексту. Цей модуль виконує інтенсивні обчислення та в реальних умовах може бути розгорнутий ізольовано для підвищення продуктивності. В даній роботі цей модуль було спрощено та безпосередньо вбудовано в серверну частину застосунку.

База даних – контейнер для постійного зберігання інформації: облікових записів користувачів, результатів розпізнавання та пов’язаних метаданих. Вона використовується виключно серверною частиною для запису та читання даних, необхідних для функціонування системи.

Також система взаємодіє з зовнішнім сервісом автентифікації, який забезпечує перевірку особи користувача. Мобільний застосунок отримує токен доступу через цей сервіс, а серверна частина перевіряє його достовірність перед тим, як дозволити подальші дії.

Наступний крок роботи за допомогою підходу C4 – перехід до діаграми компонентів. Важливо детально розглянути два найбільш складних та об’ємних контейнери, а саме серверний застосунок та OCR модуль.

2.1.1 Склад компонентів серверного застосунку

Отже, на рисунку 2.3 наведено діаграму для серверного застосунку. Вона складається із п’яти компонентів. Кожен компонент відповідає за окрему функцію застосунку і дозволяє користувачу ефективно взаємодіяти з системою.

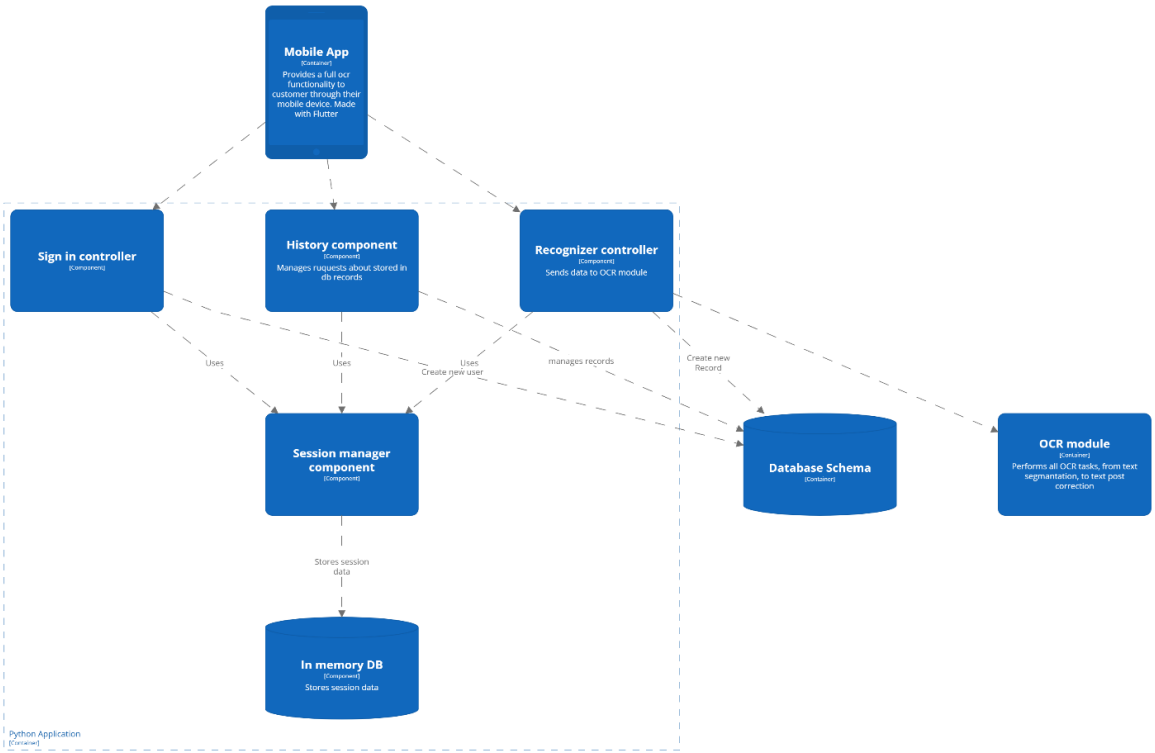

Рисунок 2.3 – Компонентна діаграма серверного затосунку

На діаграмі представлено наступні компоненти:

– модуль авторизації, що відповідає за вхід та вихід користувачів із системи, додавання нових користувачів;

– модуль управління сесіями займається валідацією сесійних cookie, створенням та видаленням сесій. Саме цей модуль робить запити до Google OAuth2.0 для валідації ID токена від користувача. Сесійні дані зберігаються в тимчасовому сховищі сесій. Воно реалізоване як легкий внутрішній модуль, який дозволяє уникнути зайвих звернень до постійної бази даних. Це може покращити швидкодію при перевірках доступу;

– компонент обробки історії пошуку. Цей компонент обробляє запити на отримання списку попередніх обробок (історії) користувача. Він звертається до постійної бази даних, фільтрує записи за користувачем і повертає метаінформацію та результати у зручному форматі. Також він піклується обробкою змін, що були внесені до результатів OCR користувачем;

– контролер розпізнавання. Цей має єдину функцію – взаємодія із OCR модулем. Тобто, цей компонент відповідає за основну бізнес-логіку системи – обробку зображень. Він приймає зображення, передає їх до OCR-модуля, отримує результат у форматі структурованого документа (наприклад, JSON), створює новий запис у базі даних і повертає результат користувачу.

2.1.2 Склад компонентів OCR модулю

OCR-модуль є спеціалізованою частиною системи, відповідальною за повний цикл обробки зображень рукописного тексту (рисунок 2.4).

Його робота починається з компонента сегментації, який виділяє окремі текстові блоки або рядки зі вхідного зображення. Отримані фрагменти впорядковуються компонентом сортування, що забезпечує правильний порядок проходження елементів – це важливо для формування

змістовного тексту у випадках з багаторядковим або несиметричним рукописом.

Після сортування сегментів тексту, кожен з них передається до компонента розпізнавання, який виконує перетворення зображення в машинний текст. На цьому етапі результати ще можуть містити помилки, спричинені неоднозначністю почерку, тому наступним етапом є контекстна післяобробка. Цей модуль виконує автокорекцію розпізнаного тексту, спираючись на мовну модель, яка дозволяє виявляти та виправляти граматичні й синтаксичні помилки.

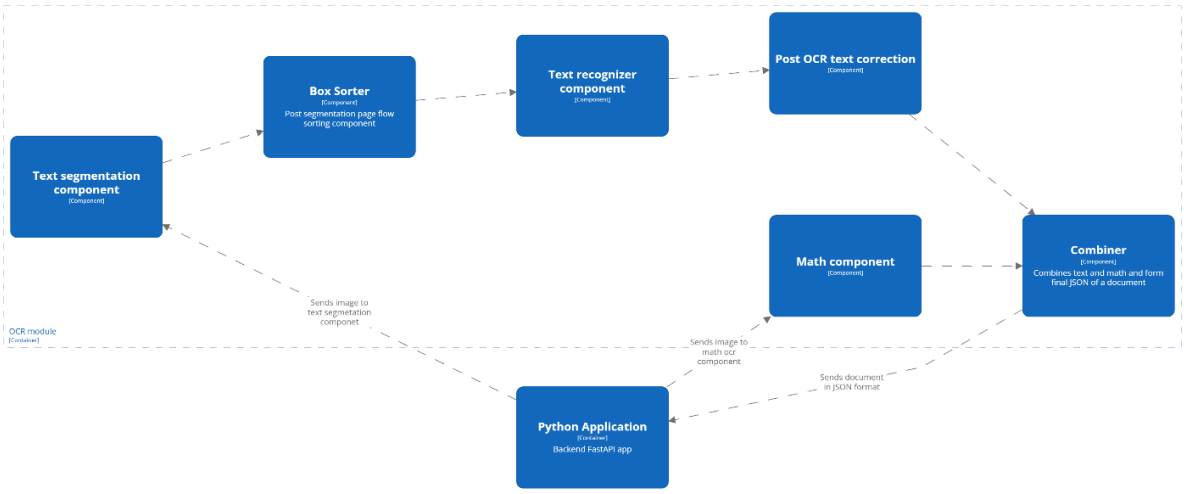

Рисунок 2.4 – Компонентна діаграма модулю OCR

Паралельно із текстовою обробкою працює модуль розпізнавання математичних формул, який виділяє та аналізує ділянки з математичним записом, повертаючи результат у вигляді структурованого представлення, наприклад, у форматі LaTeX.

Завершальний компонент OCR-модуля – комбінатор. Він відповідає за об’єднання результатів роботи всіх попередніх модулів у цілісний документ. Комбінатор формує підсумковий JSON, що включає текст, математичні вставки та координати рядків та математичних виразів, забезпечуючи цілісне представлення розпізнаного вмісту.

2.2 Проєктування бази даних

У межах даної роботи проєктування бази даних відбувалося з урахуванням основної цілі системи – розпізнавання рукописного тексту з можливістю контекстної автокорекції. Збереження результатів таких розпізнань є другорядною функціональністю, яка має на меті підвищити зручність для користувача, дозволяючи переглядати історію оброблених документів. Саме тому процес проєктування бази даних свідомо був спрощений на користь легкості реалізації, прозорості структури та мінімальної кількості сутностей.

Із цієї причину було прийнято рішення спростити багатоетапну процедуру проєктування із послідовності концептуальне – логічне – фізичне моделювання, на процедуру проєктування у вигляді концептуальне – фізичне моделювання. Це дозволило скоротити час розробки, зберігши при цьому достатню функціональність для зберігання основних даних – зокрема результатів розпізнавання, даних користувача, а також метаданих щодо часу й типу запитів. Для реалізації цього проєкту буде використано реляційну базу даних PostgreSQL.

На етапі концептуального моделювання було визначено основні сутності, які забезпечують зберігання необхідної інформації про користувачів і результати розпізнавання (рисунок 2.5).

Модель містить п’ять ключових сутностей: користувачі, OCR-записи, та в цих записах текст, лінії та математичні рівняння. Ключова ідея моделі – логічно розділити основний текст і формули, що дозволяє точно зберігати порядок їхнього розташування в документі та виконувати подальший аналіз, обробку чи візуалізацію результатів. Уся модель є достатньо простою для реалізації, але при цьому зберігає можливість гнучкої роботи з текстовими та математичними даними у подальших етапах обробки.

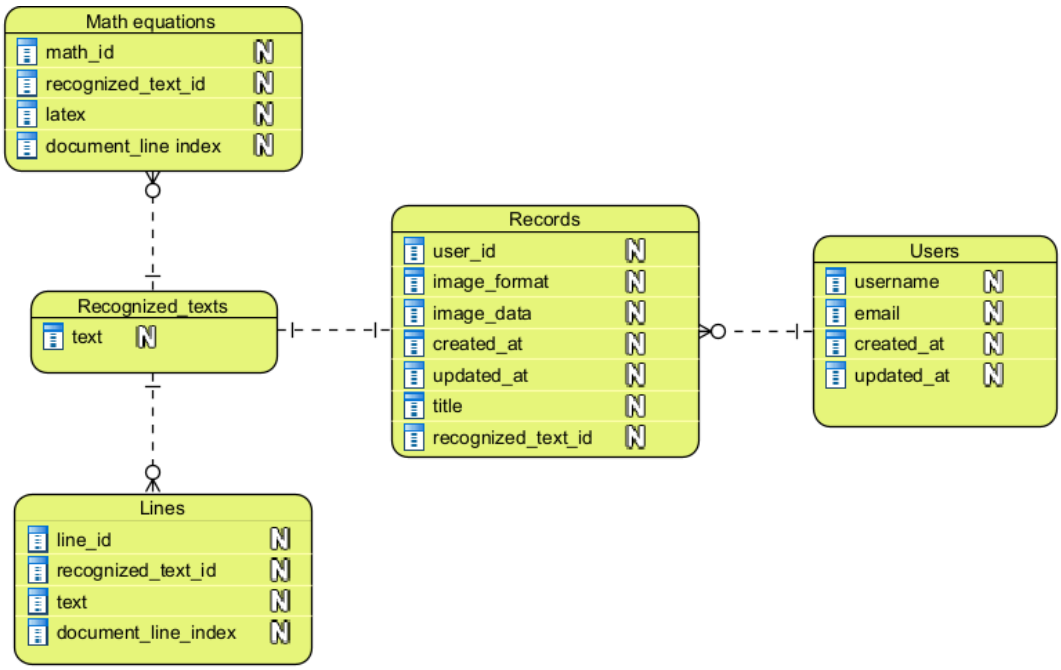

Рисунок 2.5 – Концептуальна модель даних

У процесі переходу від концептуальної до фізичної моделі даних (рисунок 2.6) було прийнято рішення ще більше спростити структуру зберігання результатів розпізнавання. Основним мотивом такого рішення стала необхідність зменшити складність реалізації, підвищити швидкодію при зчитуванні даних і спростити логіку взаємодії між компонентами серверної частини системи.

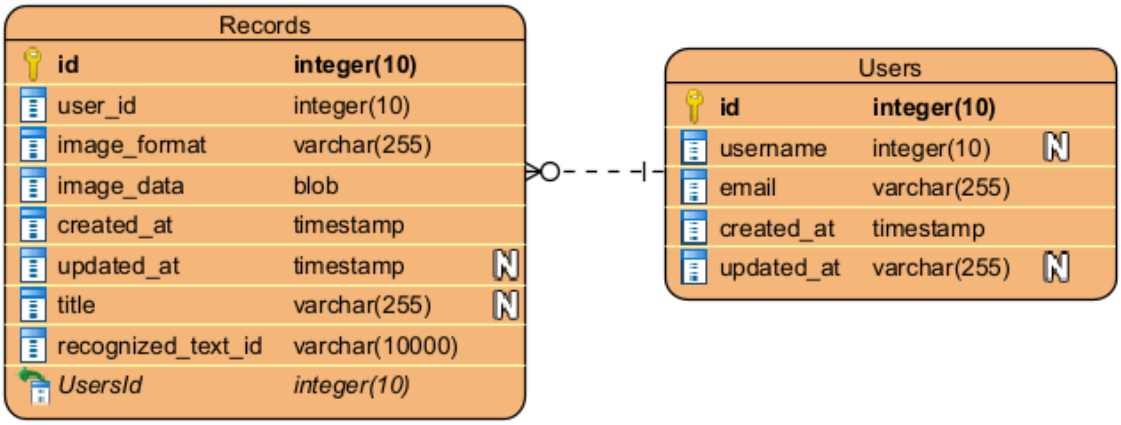

Рисунок 2.6 – Фізична модель даних

Замість реалізації окремих таблиць Recognized_texts, Lines і Math_equations, усю відповідну інформацію було зведено до одного JSON-поля в таблиці Records. Цей підхід дозволив уникнути складних зв’язків між таблицями, при цьому зберігши всю потрібну структуровану інформацію.

JSON-структура включає наступні п’ять полів:

– text – суцільний розпізнаний текст у вигляді одного рядка;

– lines – масив рядків (string), які відповідають окремим текстовим рядкам документа;

– math – масив рядків (string) із LaTeX-представленнями математичних виразів;

– lines_y_coords – масив чисел, які задають вертикальне розташування кожного текстового рядка;

– math_y_coords – аналогічний масив координат для математичних виразів.

Ця структура дозволяє зберігати інформацію у зручному для обробки вигляді та прямо передавати її у фронтенд для подальшого відображення або редагування без необхідності додаткових запитів до БД. Незважаючи на втрату нормалізованості, обрана схема є цілком виправданою в контексті наукового прототипу, де головний акцент зроблено на функціональності системи розпізнавання, а не на складній системі управління даними. У разі розширення проєкту така модель може бути трансформована у більш розгалужену реляційну структуру.

2.3 Розробка серверної частини застосунку

Серверна частина системи реалізується як окремий застосунок, що відповідає за обробку вхідних даних, взаємодію з модулями OCR, управління сесіями користувачів і збереження результатів розпізнавання. Для розробки серверної логіки було обрано мову програмування Python, яка забезпечує високу швидкість розробки, широку підтримку сучасних

бібліотек для машинного навчання та обробки зображень, а також має гнучку екосистему для створення веб-серверів.

Основою серверного застосунку є FastAPI – сучасний веб-фреймворк для створення API, який підтримує асинхронну обробку запитів, автоматичну генерацію документації (Swagger UI), валідацію вхідних даних і зручну інтеграцію з популярними бібліотеками. Завдяки FastAPI серверна частина системи лишається масштабованою, легкою у підтримці та придатною для розгортання як на локальному, так і на віддаленому сервері.

Застосунок також реалізує підтримку сесій користувачів, базується на REST-підході для побудови ендпоінтів і дозволяє обробляти як звичайні JSON-запити, так і запити з файлами (multipart/form-data), що життєво необхідно для надсилання зображень з клієнтського застосунку.

2.3.1 Автентифікація та авторизація користувачів

Автентифікація та авторизація користувачів у системі реалізується за допомогою спрощеного механізму, заснованого на ID токенах, які видаються сервісом Google під час автентифікації через OAuth 2.0. Коли користувач вперше відкриває мобільний застосунок і проходить авторизацію, клієнтська частина отримує ID токен, який передається на сервер разом із технічною інформацією про пристрій (ідентифікатор пристрою, платформа, модель).

Сервер приймає запит, перевіряє наявність обов’язкових полів, таких як токен і унікальний ідентифікатор пристрою. Якщо дані відсутні, запит відхиляється. У випадку наявності всіх необхідних параметрів, токен верифікується за допомогою запиту до Google сервісу, як наведено у лістингу 2.1. Якщо токен дійсний, з нього витягується електронна адреса та ім’я користувача. Система перевіряє, чи існує в базі даних запис із цією електронною адресою. Якщо користувач новий, створюється новий запис, у який зберігаються електронна адреса й ім’я.

Лістинг 2.1 – Процес верифікації ID токену зо допомогою Google

```
try:
     idinfo = id_token.verify_oauth2_token(token,
grequests.Request(), GOOGLE_CLIENT_ID)
except Exception:
     raise HTTPException(status_code=401, detail="Invalid
ID token")
```

Після цього створюється унікальний ідентифікатор сесії, який зв’язується з інформацією про користувача та його пристрій. Ці дані зберігаються у внутрішньому тимчасовому сховищі, а у відповідь на запит сервер прикріплює cookie з ідентифікаційним номером сесії, що дозволяє пов’язати подальші запити з конкретним користувачем.

Кожен запит, який потребує ідентифікації користувача, проходить перевірку автентичності за допомогою механізму ін’єкції залежностей, що реалізується через конструкцію Depends у FastAPI (лістинг 2.2).

Лістинг 2.2 – Приклад перевірки автентичності користувача за допомогою механізму ін’єкції залежностей

```
@app.post("/ocr_all", dependencies=[Depends(cookie)])
async def recognize_text_and_math(
    session_data: SessionData = Depends(verifier),
    ...
    ):
    ...
```

Такий підхід дозволяє автоматизувати обробку сесій та перевірку авторизаційної інформації без необхідності вручну передавати сесійні дані до кожного обробника. Зокрема, при виклику захищених ендпоінтів система автоматично витягує відповідні cookie або заголовки, перевіряє дійсність сесії, та у разі успіху передає до функції обробки даних інформацію про користувача.

У разі завершення роботи з додатком користувач або система може ініціювати видалення сесії. Для цього сервер ідентифікує сесію за cookie, очищує пов’язані з нею дані зі сховища й видаляє cookie у відповіді.

2.3.2 Основні едпойнти застосунку

На рисунку 2.7 за допомогою Swagger UI наведено основні едпойнти, які забезпечують функціонування ключових можливостей системи.

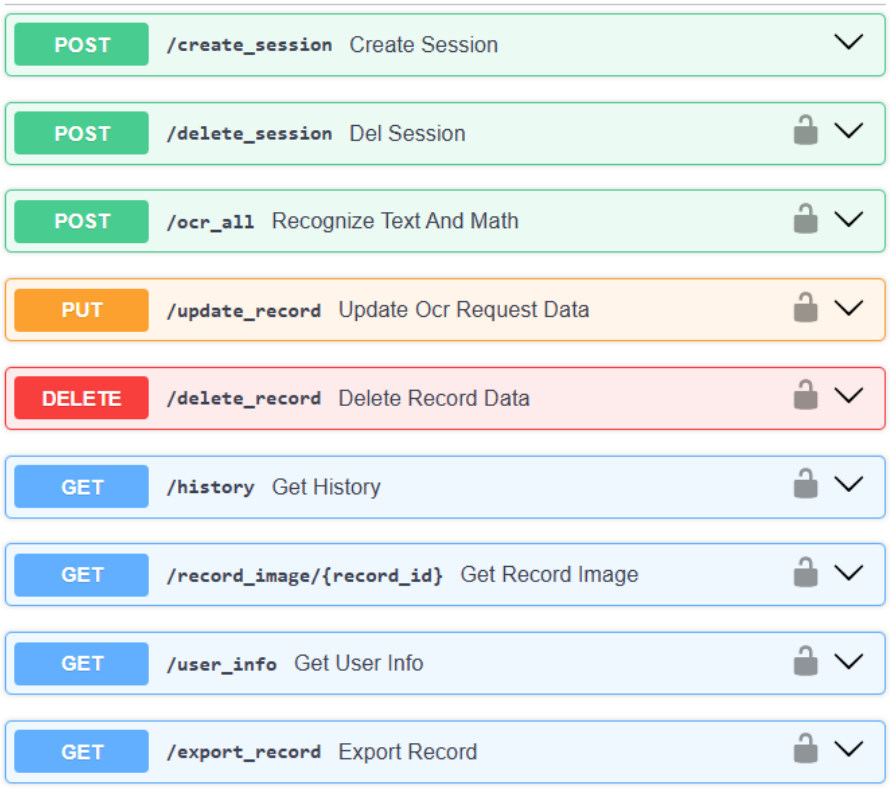

Рисунок 2.7 – Основні серверні ендпойнти

До них належать:

– `/create_session` – використовується для створення нової сесії користувача після перевірки ID токена від зовнішнього постачальника автентифікації. Цей єдиний ендпойнт є відкритим і не вимагає наявності

автентифікованої сесії. Він є єдиним ендпойнтом, який не вимагає сесійного токена;

– `/delete_session` – завершує сесію користувача, видаляючи її з внутрішнього сховища;

– `/ocr_all` – приймає зображення, запускає повний процес обробки OCR, включно з текстовим і математичним розпізнаванням, та зберігає результат;

– `/update_record` – дозволяє оновити вже існуючий запис, наприклад, виправити вміст або змінити назву;

– `/delete_record` – видаляє один із раніше створених записів користувача;

– `/history` – повертає перелік усіх записів поточного користувача;

– `/record_image/{record_id}` – повертає зображення, прикріплене до певного запису, за його ідентифікатором;

– `/user_info` – повертає основну інформацію про автентифікованого користувача;

– `/export_record` – повертає запис, конвертований у DOCX або PDF формат.

Завдяки використанню механізму перевірки сесії через cookie, сервер гарантує, що доступ до чутливої інформації або функцій мають лише авторизовані користувачі.

2.3.3 Структура OCR модулю

OCR-модуль є ядром системи і відповідає за обробку вхідного зображення, виділення текстових та математичних фрагментів, розпізнавання символів, а також контекстну післяобробку результату. Його структура реалізована у вигляді послідовності окремих компонентів, кожен з яких виконує конкретну функцію в обробному конвеєрі.

Сегментація тексту є першим і ключовим етапом у процесі обробки зображення, який дозволяє виділити текстові області, придатні для подальшого розпізнавання. У межах цієї роботи для вирішення задачі сегментації було обрано підхід, заснований на глибокому навчанні. Зокрема, використано модель CRAFT (Character Region Awareness for Text detection) – один з найефективніших сучасних алгоритмів локалізації тексту на зображеннях [13].

CRAFT побудоваий на основі згорткових нейронних мереж і забезпечує виявлення текстових регіонів з високою точністю завдяки тому, що прогнозує не лише області символів, але й зв’язки між ними. Це дає змогу виділяти текст у формі гнучких, неправильної форми блоків, що особливо актуально для обробки рукописного тексту, де часто порушено регулярність та вирівнювання.

У цьому випадку модель приймає на вхід зображення, масштабоване до відповідного розміру, і повертає координати прямокутників, які охоплюють фрагменти, що ймовірно містять текст.

Застосування CRAFT дозволило уникнути складних евристик, заснованих на морфологічних операціях або ручному налаштуванні порогів, і забезпечити універсальність при роботі з різними типами рукописних зображень.

Результати, повернені CRAFT необхідно обробити перед використання OCR моделі. Необхідно сортувати блоки відповідним до потоку сторінки чином. Це відбувається евристично. Спочатку блоки розділяються на сторінкові фрагменти. Оскільки система налаштована на роботу із рукописним текстом, то вхідні зображення зазвичай містять фото зошитів та нотаток. В такому випадку, користувач може сфотографувати як одну сторінку, так і розворот зошиту, тому необхідно це передбачити. Для цього використовується евристика, яка аналізує щільність тексту вздовж різних осей. Алгоритм виконує проєкцію сегментованих координат на набір осей у діапазоні від –20° до +20°, обчислює гістограму та шукає локальні

«провали», які можуть свідчити про розділення між сторінками. Після цього знаходяться найглибші провали та з них, як фінальний, обирається той, який має найменший кут нахилу, Опісля, блоки розділяються за знайденою віссю (рисунок 2.8).

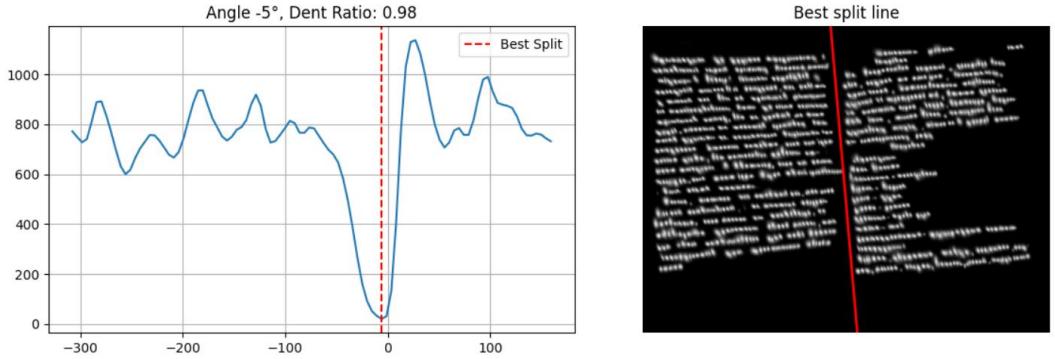

Рисунок 2.8 – Демонстрація результатів роботи алгоритму із розділення двосторінкового документа

Далі виконується сортування блоків у межах кожної групи, отриманої після розділення за віссю. Сортування здійснюється за принципом логічного порядку читання – зверху вниз і зліва направо. Для цього спочатку обчислюється вертикальна координата верхньої межі кожного блоку, і всі блоки впорядковуються за зростанням цієї координати. Після цього, всередині кожного сформованого рядка, блоки додатково сортуються за горизонтальною координатою лівої межі.

У випадках, коли деякі блоки мають подібні вертикальні координати (наприклад, через нахил або хвилясту лінію тексту), використовується допустиме порогове значення відхилення, яке дозволяє вважати такі блоки частиною одного рядка. Це значення підбирається емпірично і залежить від середньої висоти блоків.

Якщо ж рядки або блоки виявляються дуже близько один до одного (наприклад, міжрядковий інтервал малий), алгоритм додатково

враховує співвідношення розмірів блоків і відстані між ними, щоб уникнути неправильного злиття окремих рядків. Таким чином досягається стабільне і надійне впорядкування навіть у випадках нерівномірно написаного або перекошеного рукописного тексту. На рисунку 2.9 зображено приклад результату розділу документа за лініями.

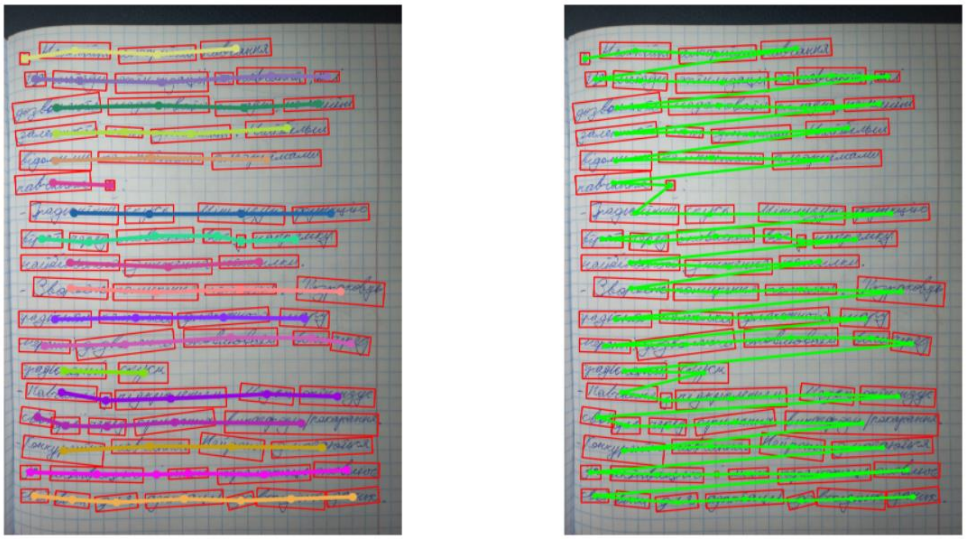

Рисунок 2.9 – Результат розподілу документа на лінії

Наступним кроком із зображення вирізаються всі розпізнані блоки, які відповідним чином обробляються до необхідного для OCR-моделі формату, та передаються туди для подальшого розпізнавання. Докладніше про архітектуру та процес тренування даної моделі наведено в розділі 3.

Після завершення обробки OCR-моделлю, кожен з блоків повертається у вигляді текстового фрагмента. Ці фрагменти потім об'єднуються відповідно до порядку, встановленого на попередньому етапі сортування. На цьому етапі також можуть бути виявлені типові помилки розпізнавання, притаманні рукописному тексту – наприклад, заміна подібних символів, пропущені літери або випадкові розриви у словах. Для усунення таких похибок застосовується окремий модуль контекстної автокорекції, який розглядає навколишній текст кожного фрагмента та

вносить відповідні правки. Принцип його роботи та особливості реалізації буде розглянуто в наступному розділі.

Паралельно з обробкою текстових блоків виконується обробка математичних формул. Для цього використовується попередньо натренована модель трансформерів TrOCR [6], адаптована для розпізнавання рукописних математичних виразів та перетворення їх у формат LaTeX [14]. На відміну від автоматичного виділення текстових фрагментів, блоки з формулами позначаються вручну користувачем безпосередньо у мобільному застосунку. Це дозволяє точно вказати місце розташування математичних виразів навіть у складних або нестандартних випадках. Після цього виділені області обробляються згаданою моделлю, а результати інтегруються до загальної структури документа разом з текстовими фрагментами.

Таким чином, структура OCR-модуля є гнучкою та модульною, що дозволяє адаптувати її під специфіку різних документів, забезпечуючи при цьому високу точність та якість результату. Завдяки поєднанню методів сегментації, алгоритмів впорядкування блоків, моделі для розпізнавання рукописного тексту та контекстної автокорекції, система здатна ефективно працювати з реальними зображеннями низької якості, зробленими за допомогою мобільних пристроїв.

Узагальнена логіка роботи OCR-модуля реалізована у вигляді окремої функції «run_ocr_pipeline», яка поєднує усі ключові компоненти: маскування зображення, обробку текстових та математичних блоків, сортування, а також контекстну корекцію. Її структура наведена у лістингу 2.3. Підфункції run_text_ocr та run_math_ocr наведено в додатку А.

Лістинг 2.3 – Вигляд головної функції комбінованої OCR системи

```
def run_ocr_pipeline(self, image, boxes=[]):
    if boxes:
        boxes = sorted(boxes, key=lambda b: (b[1] + b[3]) / 2)
```

Продовження лістингу 2.3

```
    mask = self.mask_image(image, boxes)
    crops, valid_boxes = self.crop_boxes(image, boxes)

    text, lines, lines_boxes, _ = self.run_text_ocr(mask)
    latexes, math_boxes = self.run_math_ocr(crops, valid_boxes)
       text_y,  math_y  =  self.get_lines_indexes(lines_boxes,
math_boxes)
    return text, lines, text_y, latexes, math_y
```

2.4 Висновки за розділом

У другому розділі було детально розглянуто архітектуру розроблюваного застосунку та сервісної інфраструктури, що забезпечує обробку рукописного тексту. Проведено аналіз ключових функціональних компонентів як клієнтської частини (мобільного застосунку), так і серверної частини, яка виконує розпізнавання, післяобробку та зберігання результатів. Особливу увагу приділено побудові взаємодії між фронтендом і бекендом, використанню сесій, авторизації, обміну даними та форматам запитів.

Також було описано модулі попередньої обробки зображень, етапи сегментації та впорядкування текстових блоків, а також механізм інтеграції OCR-моделі. Розглянуто спосіб розпізнавання математичних формул за допомогою окремої моделі TrOCR. Усі ці компоненти спроектовані таким чином, щоб забезпечити злагоджену роботу системи, зручний користувацький інтерфейс і високу якість результатів, що є основою для реалізації ефективного і надійного застосунку.

3 РОЗРОБКА МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ

3.1 Формування наборів даних для розпізнавання та корекції

Для досягнення мети, визначеної у постановці задачі, необхідно розробити та натренувати дві моделі штучного інтелекту: модель для оптичного розпізнавання рукописного тексту (OCR) та модель для контекстної автокорекції розпізнаного тексту.

Ефективність таких моделей значною мірою залежить від якості та репрезентативності початкового набору даних. Зважаючи на специфіку задачі – обробка українського рукописного тексту – жоден з існуючих публічних датасетів не задовольняє вимоги повною мірою.

Найбільш відомими відкритими наборами даних для розпізнавання рукописного тексту є:

– IAM Handwriting Database – англомовний датасет, що містить понад 1 500 сканів рукописних текстів із детальною розміткою на рівні слів та рядків. Широко використовується для тренування та тестування OCR-моделей, але не підтримує кирилицю;

– Cyrillic Handwritten Dataset – один з небагатьох публічних наборів даних із рукописними словами кирилицею. Містить слова з різних слов'янських мов, однак більшість слів є на російській мові, тому датасет вимагає фільтрації для виокремлення українських прикладів;

– HKR Dataset (Handwritten Kazakh & Russian) – датасет для розпізнавання кириличного тексту казахською та російською мовами.

3.1.1 Датасет для розпізнавання рукописного тексту (OCR)

Для навчання моделі оптичного розпізнавання нобхідно створити набір даних, який матиме достатню кількість прикладів, буде достатньо репрезентативним та включатиме різноманітні дані. Таким чином, було

обрано стратегію створення комбінованого датасету, що включає як приклади рукописного тексту, написаного людиною, так і синтетично згенеровані зображення слів українською мовою.

Для наповнення набору даних було зібрано 280 фотографій фрагментів зошитів, конспектів та нотаток, написаних українською мовою. Із цих зображень вручну виділено та анотовано близько 10 000 окремих слів. Розмітка здійснювалась шляхом позначення координат прямокутників навколо кожного слова з подальшим призначенням відповідного текстового підпису. Ця частина датасету відображає реальні умови сканування та фотографування рукописного тексту. Другою частиною став публічно доступний Cyrillic Handwritten Dataset з платформи Kaggle [15].

З метою відповідності задачі було попередньо видалено всі слова, що містять неукраїнські символи. У результаті було отримано близько 50000 зображень українських слів, записаних рукописним шрифтом. Останню та найбільшу частину датасету склали синтетично згенеровані слова та частини слів. Для цього було використано різноманітні українські рукописні шрифти. Для досягнення візуальної варіативності та наближення до реальних умов було реалізовано низку трансформацій:

– випадкове обертання та нахил зображень;

– elastic distortion, за якої координати пікселів зміщуються відповідно до заздалегідь згенерованого шумового поля.

– зміна яскравості, імітація нерівномірного освітлення (наприклад, затемнення однієї частини зображення);

– накладання фону у вигляді ліній або клітинок, подібно до зошитних аркушів.

На рисунку 3.1 зображено приклади синтетично згенерованих слів.

Для оцінки репрезентативності сформованого датасету було проведено аналіз частоти входження окремих символів у текстових підписах. З цією метою усі мітки датасету були оброблені для отримання сукупної частотної розподіленості символів, з урахуванням як малих, так і

великих літер (останні попередньо було зведено до відповідних малих). Отриману частотну гістограму було порівняно з еталонним розподілом символів української мови, заснованим на корпусних лінгвістичних даних [16].

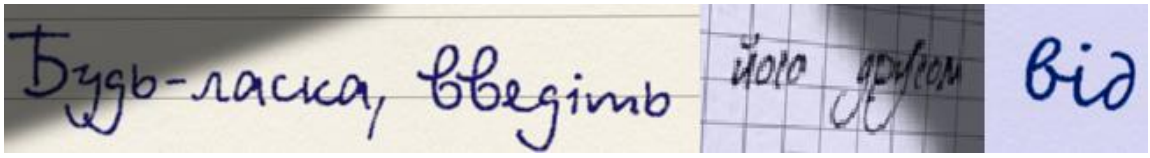

Рисунок 3.1 – Приклади синтетично згенерованих слів

Результати такого порівняння наведено на рисунку 3.2. Як видно з гістограми, отриманий розподіл достатньо добре відображає типову частоту входження символів в українському тексті.

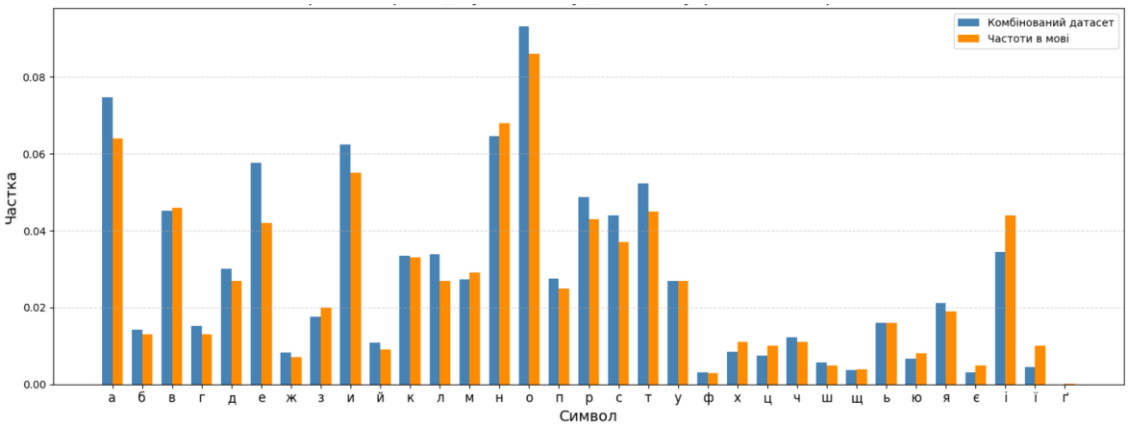

Рисунок 3.2 – Порівняльна діаграма розподілу літер

В наборі даних добре узгоджуються частоти найуживаніших букв, таких як «о», «а», «н», «и» та «е». Це підтверджує, що сформований набір є не лише візуально, але й статистично репрезентативним для задачі розпізнавання рукописного тексту українською мовою.

Перед використання моделлю всі зображення проходили попередню обробку у вигляді приведення зображення до одноканального формату, а також масштабування до фіксованого розміру 256×32 пікселів без викривлення пропорцій. Для цього використовувалося доповнення нульовими пікселями по краях (рисунок 3.3).

Рисунок 3.3 – Приклади оброблених зображень

Такий підхід до попередньої обробки дозволив забезпечити уніфіковану форму вхідних даних, придатну для батч-обробки в рамках глибокої нейромережевої моделі.

3.1.2 Датасет для контекстної автокорекції

Щоб донавчати модель контекстної автокорекції необхідно сформувати пару речень у форматі «пошкоджений текст – правильний текст». Оскільки помилки, які виникають у процесі оптичного розпізнавання рукописного тексту, часто є систематичними або повторюваними, було прийнято рішення створити штучно пошкоджений датасет, базуючись на великому корпусі правильно написаних речень українською мовою. Як джерело правильного тексту було обрано корпус OSCAR (версія unshuffled_deduplicated_uk) – відкритий корпус українських текстів, отриманий шляхом фільтрації веб-даних. Із загального обсягу було відібрано 500 000 речень, після чого виконано попереднє очищення та фільтрацію, в ході яких було видалено дубльовані зразки, вилучено занадто

короткі приклади, а також залишено лише ті рядки, що мали українські літери, цифри та базові розділові знаки.

Для моделювання типових OCR-помилок було створено спеціальну структуру даних, яка визначає відповідності між схожими за виглядом символами. Наприклад, літера н може бути помилково розпізнана як п або м, о – як а, і – як 1, тощо. Усього було створено близько сотні таких правил, які враховують рукописні особливості написання.

Застосовуючи цю карту помилок, кожне речення з корпусу було модифіковано наступним шляхом:

– випадкові заміни символів згідно з картою помилок;

– вставки випадкових символів;

– видалення випадкових символів;

– дублювання випадкових символів;

– поділ випадкових слів на дві частини.

Рівень спотворення регулювався параметрами, що задавали ймовірності кожної з модифікацій на трьох рівнях інтенсивності (легкий, середній, сильний), із перевагою на користь більш інтенсивних. У результаті для кожного оригінального речення було згенеровано штучно спотворений аналог, що імітує типові помилки систем оптичного розпізнавання.

3.2 Розробка та тренування OCR-моделі

У процесі розробки системи розпізнавання рукописного тексту було розглянуто низку архітектурних підходів. Серед них – сучасні трансформерні моделі (зокрема, TrOCR), гібридні варіанти з елементами уваги, легші CNN+BiLSTM+CTC-моделі з меншою кількістю згорткових шарів, а також готові рішення, як-от Tesseract OCR. Однак найкращі результати в умовах наявного набору даних вдалося досягти при використанні глибокої архітектури, що поєднує згорткові шари (CNN), бінаправлені рекурентні шари (BiLSTM) та функцію втрат Connectionist

Temporal Classification (CTC). Перевага такого підходу полягає у відмові від необхідності попереднього сегментування тексту на символи, що є особливо важливим при роботі з рукописними зображеннями, де межі між символами часто розмиті або взагалі відсутні.

На рисунку 3.4 представлено архітектуру створеної моделі. Вхідним даним слугує зображення розміром 256×32 пікселі у відтінках сірого, після чого застосовується послідовність згорткових шарів, нормалізації та підвибірки (MaxPooling), що дозволяє виділити ознаки високого рівня. Далі, через операцію Reshape, просторові ознаки перетворюються у часові послідовності, які подаються на два бінаправлені шари LSTM. Після рекурентної обробки використовується шар Dropout для регуляризації та повнозв’язний шар, який формує ймовірнісні розподіли для кожного кроку. Остаточне формування послідовності виконується за допомогою CTC-декодера.

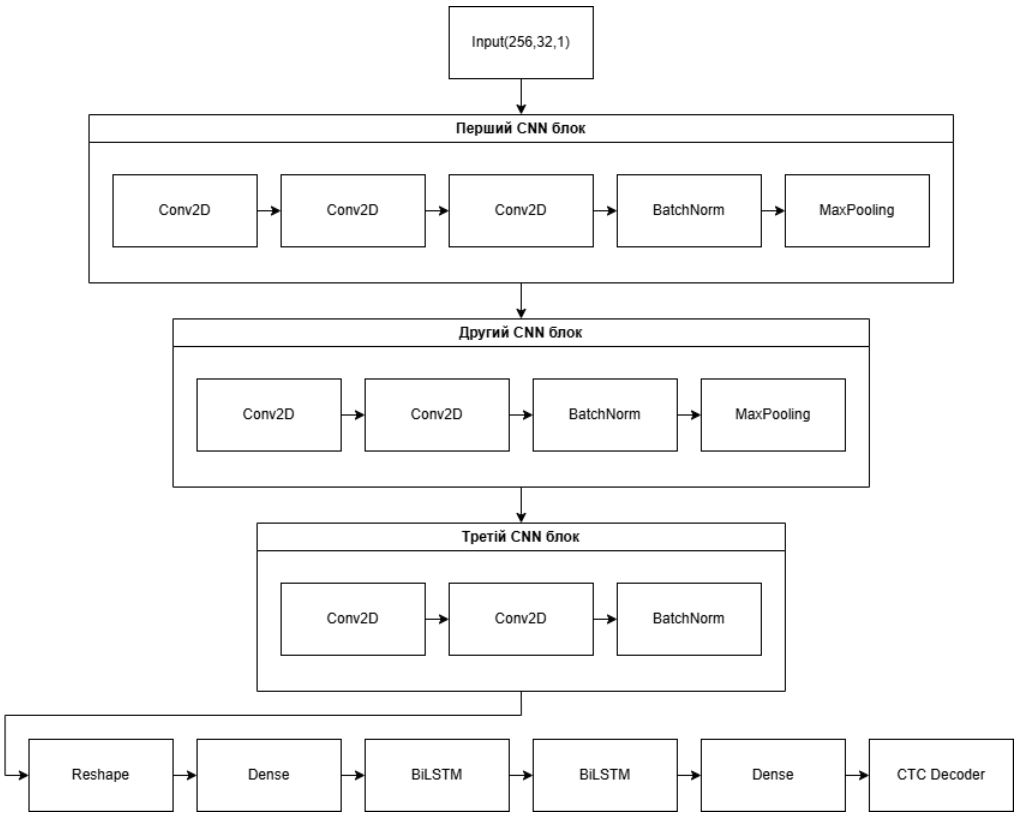

Рисунок 3.4 – Архітектура OCR моделі

Епоха – в контексті штучного інтелекту це один повний прохід усіх навчальних даних через модель під час її тренування. Модель навчалась протягом 10 епох із використанням оптимізатора Adam та розміром групи 128. На рисунку 3.5 зображено графік зміни функції втрат для навчального та валідаційного наборів. Як видно з графіку, вже після перших трьох епох модель демонструє різке покращення, а в подальшому – поступову стабілізацію. Це свідчить про ефективне узгодження параметрів та відсутність значного перенавчання.

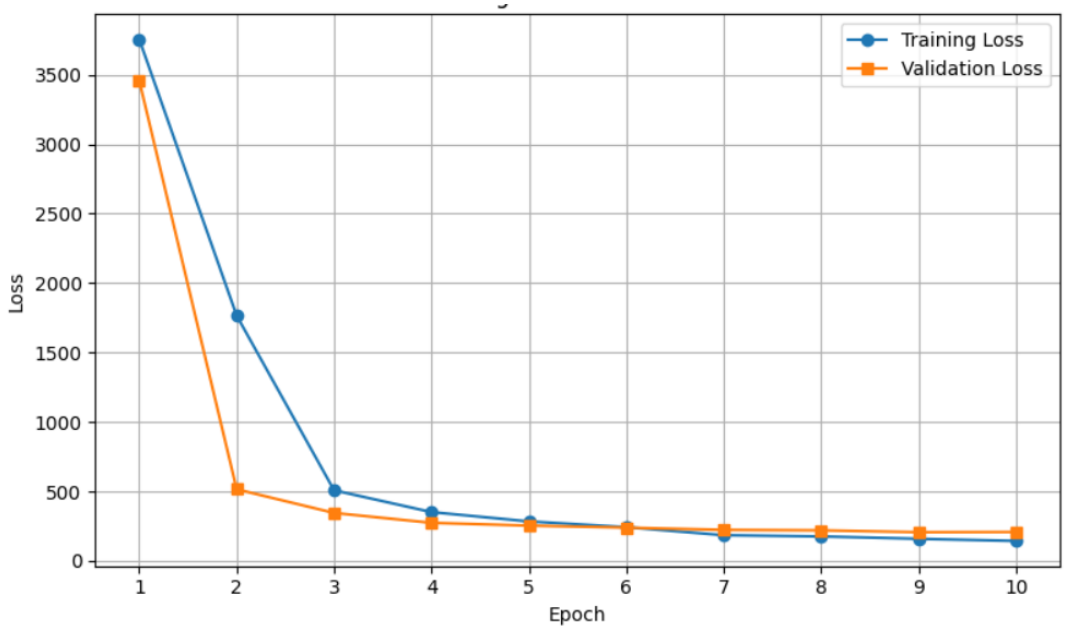

Рисунок 3.5 – Графік функції втрат для OCR моделі

Для оцінки практичної ефективності модель було протестовано на тій частині датасету, яка була зібрана вручну. Незважаючи на складні умови та значну зашумленість зображень, модель досягла значення метрики Character Error Rate (CER) = 14,97%, тобто всього 15% результатів були помилковими, що свідчить про її високу стійкість до зовнішніх факторів і придатність для використання в реальних сценаріях.

3.3 Розробка та тренування моделі корекції

Для контекстної автокорекції рукописного тексту було обрано трансформерну архітектуру mBART, яка є багатомовним розширенням моделі BART, що базується на принципах BERT. Модель BERT (Bidirectional Encoder Representations from Transformers) була запропонована в роботі Якоба Девліна [17], і є однією з перших моделей, що здійснює двонаправлене кодування тексту, завдяки чому може враховувати як лівий, так і правий контекст слів у реченні. Архітектура BERT використовує лише енкодери трансформера, навчені на завданні маскування слів (Masked Language Modeling) та прогнозу наступного речення (Next Sentence Prediction).

Безпосередньо для донавчання було використано архітектуру mBART (Multilingual BART), описану в роботі [18], яка поєднує енкодер-декодерну структуру та підтримує десятки мов, включаючи українську. У рамках цього дослідження для подальшої адаптації під завдання граматичної корекції українського тексту було використано попередньо натреновану модель «schhwmn/mbart-large-50-finetuned-ukr-gec з бібліотеки» Hugging Face. Ця модель була донавчена на помилкових реченнях із навчального піднабору корпусу UA-GEC [19]. Набір даних UA-GEC включає речення з реальними граматичними помилками та їхні виправлення, що робить його особливо цінним для задачі автоматичної корекції.

Дані для навчання були завантажені у форматі Dataset, та містили пари текстів: початковий (помилковий) та цільовий (коректний) варіанти. Тексти перед передачею на вхід до моделі оброблювались у форматі із спеціальними токенами, а токенізація виконувалась за допомогою MBart50Tokenizer з максимальним розміром послідовності 640 токенів. Дотренування було проведено із швидкістю навчання в $5 \cdot 10^{-7}$ ступені, на протязі 3 епох, та із розміром пакету в 4 приклади і склало 37 тис. кроків.

Процес навчання супроводжувався оцінкою функції втрат на тренувальній та валідаційній вибірках (рисунок 3.6), де спостерігалось стабільне зниження втрат без явного перенавчання. Результати тренування показали стабільне зниження втрат (≈0.15 після 3 епох), що свідчить про ефективне навчання моделі автокорекції.

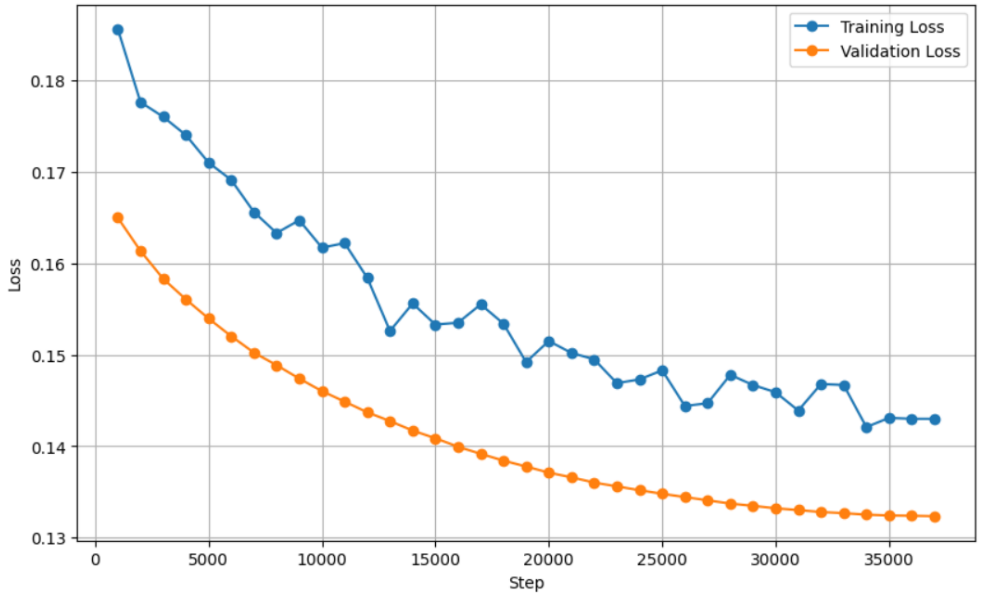

Рисунок 3.6 – Графік функції втрат для моделі корекції тексту

3.4 Висновки за розділом

У цьому розділі було розглянуто процес підготовки даних та навчання двох ключових моделей, які забезпечують функціональність системи розпізнавання рукописного тексту: моделі оптичного розпізнавання (OCR) та моделі контекстної автокорекції.

Під час формування датасетів було враховано специфіку української мови та відсутність повноцінних публічних наборів даних, що призвело до створення комбінованого набору: частково зібраного вручну, частково очищеного з відкритих джерел і синтетично згенерованого. Це дозволило досягти високого рівня різноманітності та реалістичності вхідних прикладів для навчання моделі OCR. Для завдання автокорекції сформовано

спеціальний корпус із пар «пошкоджений – правильний» текст, що імітує типові помилки систем розпізнавання.

У ході дослідження було реалізовано та навчено дві моделі: CNN+BiLSTM+CTC для розпізнавання зображень тексту, а також здійснено донавчання трансформерної моделі mBART для граматичної та контекстної корекції результатів. Обидві моделі продемонстрували стабільне навчання, низькі втрати на валідаційних вибірках і задовільну якість результатів у межах цільового домену.

4 РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ

4.1 Загальна характеристика клієнтської частини

Мобільний застосунок, розроблений у рамках цієї кваліфікаційної роботи, призначений для платформи Android, що зумовлено її високою популярністю серед користувачів у світі, зокрема в Україні. Підтримка Android дозволяє забезпечити доступність застосунку для широкого кола цільових користувачів: студентів, викладачів та дослідників.

Для реалізації клієнтської частини було обрано Flutter – кросплатформену технологію від Google, що базується на мові програмування Dart [20]. Це рішення дозволяє створювати сучасні, швидкі та візуально привабливі інтерфейси з високим рівнем продуктивності. Flutter надає великий набір вбудованих віджетів і активну спільноту, що сприяє швидкому розвитку застосунку та його гнучкому масштабуванню.

Серед додаткових переваг Flutter – кросплатформенність, що надає можливість одночасного створення застосунків, що будуть працювати із Android та iOS, а також підтримка нативної інтеграції з платформеними API. Хоча на поточному етапі реалізується лише Android-версія, використання Flutter забезпечує потенційну можливість розширення підтримки на інші платформи у майбутньому без необхідності повного переписування коду.

4.2 Опис візуального інтерфейсу

Мобільний застосунок передбачає простий та інтуїтивно зрозумілий процес автентифікації користувача за допомогою Google OAuth 2.0. Після надання згоди користувачем на авторизацію, клієнтська частина отримує ID токен, який надсилається на сервер для валідації. Серверна частина перевіряє справжність ID окену через відповідний Google API та, у разі успішної перевірки, створює новий запис сесії з унікальним

ідентифікатором. Цей ідентифікатор зберігається у cookie, який автоматично додається до HTTP-відповіді й зберігається в застосунку. Надалі клієнтський застосунок зберігає його в захищеному сховищі, і автоматично надсилає цей сесійний cookie з кожним запитом до серверу. Сервер ідентифікує користувача, перевіряючи дійсність сесії у внутрішньому сховищі. Таким чином реалізується механізм безпечного контролю доступу до персональних даних користувача. У лістингу 4.1 наведено приклад виконання запиту до серверної частини застосунку із метою отримання інформації користувача та із включеним до заголовку сесійним cookie.

Лістинг 4.1 – Верифікація сесійного cookie в одному з ендпоінтів

```
final response = await _httpClient.get(
      ApiConfig.uris.userInfo,
      headers: {
        'Content-Type': 'application/json',
        if (_cookies.isNotEmpty) 'Cookie':
_formatCookies(),
      },
    );
```

Мобільний застосунок реалізує кілька основних сторінок, кожна з яких відповідає певному етапу роботи користувача з рукописним текстом. На рисунку 4.1 зображено головну сторінку. Вона є основним елементом є застосунку, яка слугує точкою входу до функціональності розпізнавання. Звідси користувач може зробити нове фото або вибрати існуюче з галереї, після чого отримане зображення проходить етап попередньої обробки: його можна повернути або обрізати відповідно до потреб користувача.

Після редагування зображення користувач переходить на сторінку підготовки до розпізнавання (рисунок 4.2), де йому надається можливість вручну виділити області, які слід розпізнавати як математичні формули. Ці

області згодом обробляються окремою моделлю, спеціалізованою на перетворенні математичних виразів у формат LaTeX. Решта зображення інтерпретується загальною OCR-моделлю.

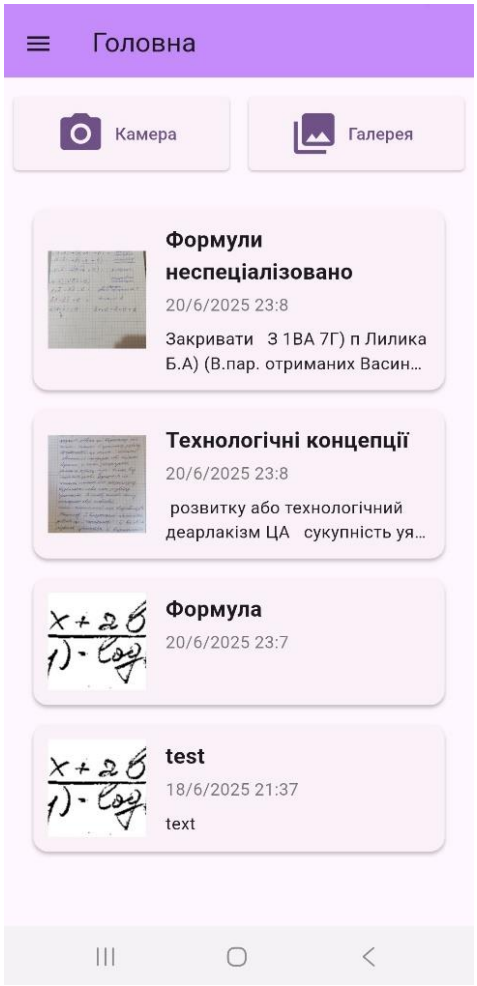

Рисунок 4.1 – Головна сторінка

Після обробки зображення сервер надсилає результат у структурованому форматі. Відповідь містить розпізнаний текст та його сегментацію, у структурованому JSON форматі. Приклад форматування наведено у лістингу 4.2.

Лістинг 4.2 – Форматування розпізнаного тексту у відповіді з сервера

```
recognized_text = {
    "text": text,
```

Продовження лістингу 4.2

```
    "lines": lines,
    "lines_y_coords": lines_y_coords,
    "math": latexes,
    "math_y_coords": math_y_coords
}
```

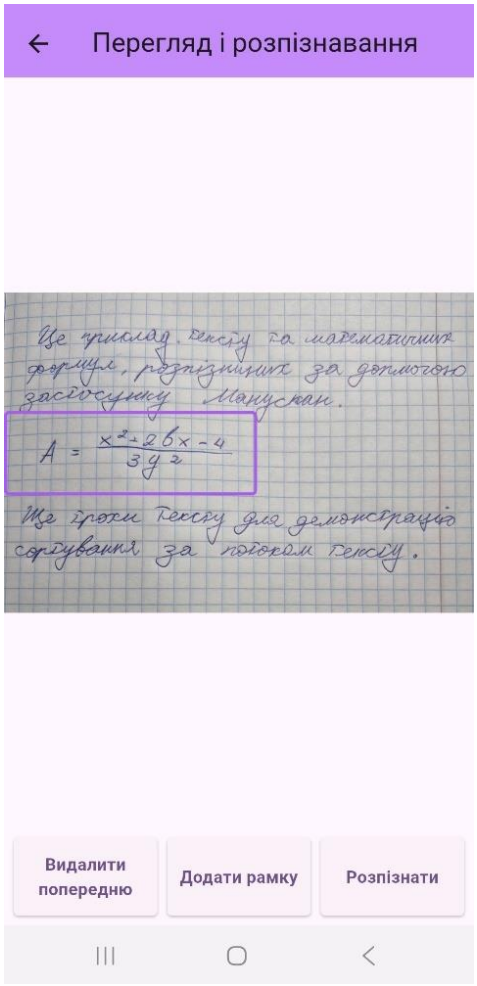

Рисунок 4.2 – Сторінка OCR

У даному об’єкті поле «text» є зібраним суцільним текстом, що утворений об'єднанням рядків через пробіл. Координати «lines_y_coords» та «math_y_coords» є покажчиками на рядки у документі. Важливо, що множини індексів «lines_y_coords» та «math_y_coords» є

взаємовиключними, тобто не перетинаються: кожен індекс відповідає або текстовому рядку, або математичному виразу.

Після завершення розпізнавання користувач може перейти до історії запитів (рисунок 4.3), де відображаються всі попередні OCR-сеанси.

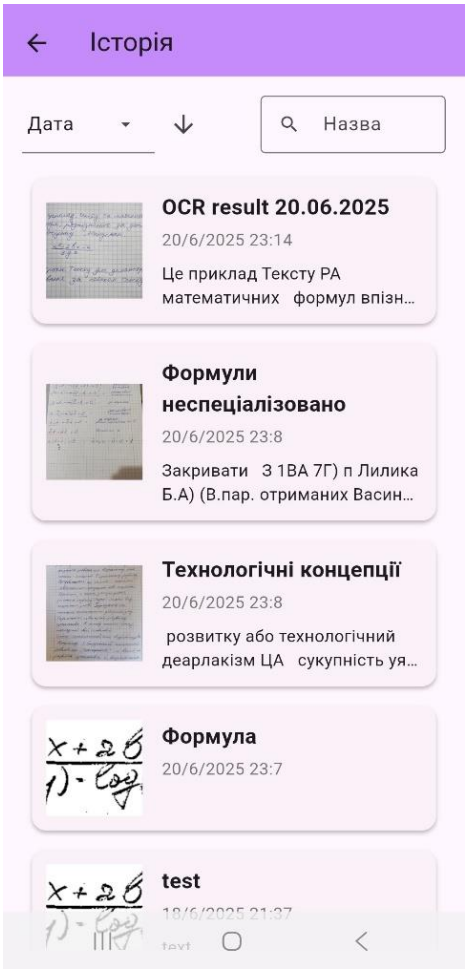

Рисунок 4.3 – Сторінка історії

Ця сторінка включає інструменти для пошуку та фільтрації записів: можна здійснювати пошук за назвою документа (за допомогою нечутливого до регістру пошуку за підрядком у назві документа), сортувати записи за назвою або датою створення, а також змінювати порядок сортування між висхідним та низхідним. Крім того, реалізовано пагінацію результатів, що забезпечує ефективну навігацію при великій кількості документів.

Кожен запит можна переглянути на сторінці редагування, макет якої наведено на рисунку 4.4, де користувач має можливість:

– переглянути результат розпізнавання;
– внести правки до текста;
– експортувати документ;
– видалити запис із сервера, якщо він більше не потрібен.

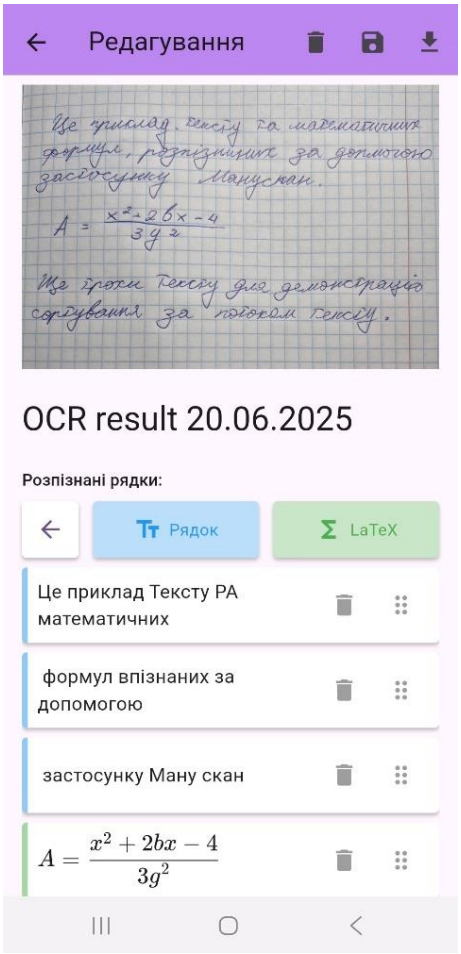

Рисунок 4.4 – Сторінка редагування

Застосунок надає можливість експортувати оброблений документ у форматах DOCX або PDF. Після вибору відповідної опції формується запит на сервер, де на основі раніше розпізнаного тексту відбувається генерація документа. Форматування тексту здійснюється лінія за лінією відповідно до структури, описаної у відповіді з сервера (лістинг 4.2): окремо зберігаються

звичайні текстові рядки та математичні формули з їхніми координатами у документі. Після формування файл стає доступним для завантаження користувачем безпосередньо на пристрої.

4.3 Висновки за розділом

У цьому розділі було розглянуто реалізацію мобільного застосунку, що є ключовим компонентом системи розпізнавання рукописного тексту. Застосунок розроблено для платформи Android із використанням фреймворку Flutter, що забезпечує кросплатформеність, гнучкість дизайну та ефективну інтеграцію з серверною частиною. Було впроваджено зручний візуальний інтерфейс, який охоплює всі етапи взаємодії користувача із системою: авторизацію, вибір або створення зображення, його попередню обробку, сегментацію, розпізнавання тексту та подальшу роботу з результатом. Особливу увагу приділено сторінці історії, яка дозволяє швидко знаходити попередні запити завдяки сортуванню, пошуку за назвою, фільтрації та підтримці пагінації. Застосунок реалізує зручний і надійний механізм аутентифікації через Google OAuth 2.0, із подальшим збереженням сесії у вигляді HTTP-only cookie. Це забезпечує безпеку взаємодії та захист персональних даних користувача. Таким чином, створене програмне рішення охоплює весь цикл обробки рукописного тексту – від завантаження зображення до експорту готового документа, при цьому залишаючись інтуїтивно зрозумілим та зручним у використанні.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено повнофункціональний мобільний застосунок, що забезпечує конвертацію рукописного тексту у цифровий формат із подальшою контекстною автокорекцією. Реалізована система складається з двох основних частин – клієнтської – мобільний застосунок для Android, та серверної – FastAPI-застосунок із модулями обробки зображень, розпізнавання тексту та автокорекції. Такий підхід дозволив поєднати зручність користування з ефективністю сучасних моделей штучного інтелекту, реалізованих на сервері.

В рамках роботи було здійснено глибокий аналіз предметної області. Було визначено ключові проблеми, з якими стикаються користувачі під час оцифрування рукописних документів, серед них: нечіткість написання, перекошені рядки, неоднорідне освітлення зображень, а також значна кількість помилок після первинного розпізнавання. Було також визначено виклики, пов’язані із розробкою OCR систем.

Під час проектування архітектури застосунку було обрано модульну структуру з чітким поділом на компоненти, що взаємодіють через API. Клієнтська частина на Flutter забезпечує зручний інтерфейс для завантаження фотографій, виділення формул та перегляду результатів. Серверна частина реалізує обробку зображення, сегментацію, вирівнювання, розділення на блоки, а також передачу цих блоків до моделей OCR та корекції. У межах архітектурного рішення передбачено можливість локального розгортання серверної частини, що забезпечує конфіденційність даних користувача та можливість автономної роботи без підключення до хмарних сервісів.

У межах дослідження було створено два повноцінних датасети. Для OCR-моделі – комбінований набір з реальних рукописів, очищених публічних джерел та синтетичних прикладів. Для моделі автокорекції –

корпус із речень українською мовою, до яких були штучно додані помилки, що імітують типові збої при розпізнаванні рукописного тексту. Такий підхід дозволив забезпечити прийнятний рівень узагальнення та адаптованості моделей до реальних умов використання.

У процесі розробки було реалізовано дві моделі штучного інтелекту. Перша модель спрямована на розпізнавання рукописного тексту, що поєднує згорткову нейромережу (CNN), двонаправлені рекурентні шари (BiLSTM) та CTC-декодер. Вона дозволяє розпізнавати рядки або слова тексту без попереднього розділення на символи. Друга модель спрямована на контекстну автокорекцію на основі трансформерної архітектури mBART, яка враховує граматичні, лексичні та орфографічні залежності в межах речення.

Обидві моделі було успішно навчено та інтегровано в загальний сервіс. OCR-модель досягла Character Error Rate близько 15% на валідаційному наборі з реальних прикладів, що є прийнятним рівнем для задач з високим рівнем шуму та спотворень. Модель автокорекції демонструє незначне зниження втрат на етапі донавчання та генерує виправлення, що певним чином покращують підсумковий текст.

Крім того, у застосунку передбачено окрему обробку математичних формул. Користувач може вручну виділяти відповідні області, які надалі передаються на обробку за допомогою готової моделі TrOCR, натренованої на рукописних формулах і здатної повертати результат у форматі LaTeX.

У результаті реалізоване рішення дозволяє зручно оцифровувати рукописні нотатки українською мовою, автоматично виправляти частину помилок розпізнавання завдяки контекстній корекції, виділяти математичні формули для окремого розпізнавання, отримувати фінальний результат у вигляді цифрового тексту, придатного для подальшого редагування, експорту в DOCX або PDF та збереження в особистому архіві.

Незважаючи на досягнуті результати, розроблена система має значний потенціал для подальшого вдосконалення. Одним із можливих напрямків є заміна або донавчання моделі OCR на сучасніші архітектури, зокрема на основі трансформерів (наприклад, TrOCR або Donut), які демонструють вищу точність на складних даних і не потребують ручного формування окремих ознак. Це дозволило б ще більше знизити рівень помилок при розпізнаванні та покращити стійкість до шумів.

Ще одним напрямком є вдосконалення алгоритму сортування розпізнаних фрагментів тексту. Наразі сортування відбувається за евристичними метриками, однак цей підхід має обмеження при обробці нестандартних макетів. Використання навченої моделі або алгоритмів графової класифікації для визначення порядку читання фрагментів могло б значно покращити якість відновлення логічної структури документа.

З боку клієнтського застосунку перспективним виглядає впровадження автоматичного генерування назви документів на основі їх змісту, а також автоматичного створення короткого змісту або анотації, що було б корисним для подальшої організації, пошуку та фільтрації документів.

Крім того, можливим є розширення функціональності шляхом додавання підтримки багатомовного розпізнавання та перекладу. Зокрема, користувач міг би обрати бажану мову перекладу після розпізнавання, а система – автоматично виконати переклад за допомогою відповідної моделі трансформера, наприклад, mBART чи MarianMT. Це дозволило б зробити застосунок корисним не лише в академічному або освітньому контексті, а й у міжнародному середовищі.

Отже, створене рішення не лише демонструє практичну ефективність, але й формує основу для майбутнього розширення та дослідження в напрямах мультимодальної обробки рукописних документів, інтеграції з іншими сервісами та підвищення рівня інтелектуальності користувацького досвіду.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. d'Albe E. E. F. The Type-Reading Optophone. *Scientific american*. 1914. Т. 78. № 2032supp. С. 371. URL: https://doi.org/10.1038/scientificamerican12121914-371asupp (дата звернення: 01.06.2025).

2. IBM archives: rochester chronology. *Wayback Machine*. URL: https://web.archive.org/web/20190105095747/https://www.ibm.com/ibm/history/exhibits/rochester/rochester_chronology2.html (дата звернення: 05.06.2025).

3. Gradient-based learning applied to document recognition / Y. Lecun та ін. *Proceedings of the IEEE*. 1998. Т. 86, № 11. С. 2278–2324. URL: https://doi.org/10.1109/5.726791 (дата звернення: 06.06.2025).

4. Marti U. V., Bunke H. A full English sentence database for off-line handwriting recognition. *Proceedings of the fifth international conference on document analysis and recognition. ICDAR '99 (cat. no.pr00318)*, м. Bangalore, India, 22 верес. 1999 р. 1999. URL: https://doi.org/10.1109/icdar.1999.791885 (дата звернення: 06.06.2025).

5. Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks / A. Graves та ін. *Proceedings of the 23rd International Conference on Machine Learning*. 2006. С. 369–376.

6. TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models / M. Li та ін. *arXiv preprint arXiv:2109.10282*. 2021. URL: https://arxiv.org/abs/2109.10282. (дата звернення: 07.06.2025).

7. OCRBench: on the hidden mystery of OCR in large multimodal models / Y. Liu та ін. *Science China Information Sciences*. 2024. Т. 67, № 12. URL: https://doi.org/10.1007/s11432-024-4235-6 (дата звернення: 07.06.2025).

8. Малишев О. Використовуємо CNN для обробки зображень. Частина перша. *DOU.ua*. URL: https://dou.ua/forums/topic/48368/ (дата звернення: 21.06.2025).

9. Calzone O. An Intuitive Explanation of LSTM. *Medium*. URL: https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c (дата звернення: 09.06.2025).

10. Sequence Modeling with CTC. *Distill*. URL: https://distill.pub/2017/ctc/ (дата звернення: 09.06.2025).

11. Attention Is All You Need / A. Vaswani та ін. *arXiv.org*. URL: https://arxiv.org/abs/1706.03762 (дата звернення: 21.06.2025).

12. Home. *C4 model*. URL: https://c4model.com/ (date of access: 16.06.2025).

13. Character Region Awareness for Text Detection / Y. Baek та ін. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, м. Long Beach, CA, USA, 15–20 черв. 2019 р. 2019. URL: https://doi.org/10.1109/cvpr.2019.00959 (дата звернення: 16.06.2025).

14. TrOCR model fine-tuned on a part of the mathwriting dataset converted from InkML files into images. *Hugging Face – The AI community building the future.* URL: https://huggingface.co/fhswf/TrOCR_Math_handwritten (дата звернення: 19.06.2025).

15. Cyrillic Handwriting Dataset. *Kaggle*. URL: https://www.kaggle.com/datasets/constantinwerner/cyrillic-handwriting-dataset (дата звернення: 21.06.2025).

16. Частоти повторюваності українських літер. *Друкарня Друкарик*. URL: https://drukaryk.com/chastoty-povtoruvannosti-ukrainskyh-bukv/?srsltid=AfmBOoqipxO1n5waJflXIB7PJ75KjBt2E1qfEdzRP9z5h0dgxUBRAgon (дата звернення: 21.06.2025).

17. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin та ін. 2018.

18. Multilingual Translation with Extensible Multilingual Pretraining and Finetuning / Y. Tang та ін. *arXiv.org*. URL: https://arxiv.org/abs/2008.00401 (дата звернення: 17.06.2025).

19. UA-GEC: Grammatical Error Correction and Fluency Corpus for the Ukrainian Language / O. Syvokon та ін. *Proceedings of the Second Ukrainian Natural Language Processing Workshop (UNLP)*, м. Dubrovnik, Croatia. Stroudsburg, PA, USA, 2023. URL: https://doi.org/10.18653/v1/2023.unlp-1.12 (дата звернення: 19.06.2025).

20. Flutter documentation. *Docs | Flutter*. URL: https://docs.flutter.dev/ (дата звернення: 21.06.2025).