Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ЗАСТОСУНКУ ОПТИЧНОГО РОЗПІЗНАВАННЯ СИМВОЛІВ ДЛЯ ОТРИМАННЯ ТЕКСТУ З ФОТОГРАФІЙ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-19-1
Бутенко П.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Вечірська І.Д.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Бутенку Павлу Владленовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка застосунку оптичного розпізнавання символів для отримання тексту з фотографій

затверджена наказом університету від 15 травня 2023 року № 474 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 29 травня 2023 р.

3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека для оптичного розпізнавання символів з відкритим кодом Tesseract, мова програмування PHP, framework Laravel, середовище автоматизації розгортання Docker.

4. Перелік питань, що потрібно опрацювати в роботі

1. Аналіз існуючих рішень та бібліотек оптичного розпізнавання символів для отримання тексту з фотографій.
2. Аналіз методів обробки зображень.
3. Розробка застосунку для оптичного розпізнавання символів.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність проблеми обробки тексту з зображень, постановка задачі, реалізація розпізнавання тексту, розпізнавання тексту.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Творошенко І.С.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	10.04.2023	
2	Аналіз завдання, підбір літератури	11.04.23-15.04.23	
3	Аналіз літератури з досліджуваної проблеми	16.04.23-23.04.23	
4	Аналіз технічних засобів і програмних засобів	24.04.23-30.04.23	
5	Моделювання застосунку	1.05.23-11.05.23	
6	Програмна реалізація	12.05.23-26.05.23	
7	Оформлення пояснювальної записки	27.05.23-2.06.23	
8	Перевірка на плагіат	03.06.23	
9	Рецензування	04.06.23	
10	Підготовка презентації та доповіді	05.06.23-06.06.23	
11	Занесення роботи в електронний архів	07.06.23	
12	Попередній захист кваліфікаційної роботи	07.06.23	

Дата видачі завдання 10 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ _____доц. Вечірська І.Д._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 74 с., 2 табл., 26 рис., 1 дод., 30 джерел.

ЗАСТОСУНОК, ОТОЧЕННЯ DOCKER ТА DOCKER COMPOSE, МОВА ПРОГРАМУВАННЯ PHP, FRAMEWORK LARAVEL, ОПТИЧНЕ РОЗПІЗНАВАННЯ СИМВОЛІВ, МЕТОДИ ОПТИМІЗАЦІЇ ФОТОГРАФІЇ.

Об’єтом роботи є дослідження методу оптичного розпізнавання символів.

Метою роботи є розробка програми для оптичного розпізнавання символів з використанням технології Tesseract OCR. Основним завданням є створення програмного забезпечення, яке здатне обробляти фотографії з текстом та автоматично розпізнавати символи на зображенні, що дозволить підвищити ефективність та точність обробки інформації.

Використано мову програмування PHP, фреймворк Laravel та контейнеризацію програми за допомогою Docker. Створена програма має простий і зручний інтерфейс для завантаження зображень та виведення результатів розпізнавання. Вибрано оптимальні датасети для покращення якості розпізнавання.

Результатом цієї роботи є повноцінне програмне забезпечення, здатне ефективно обробляти фотографії з текстом, що може бути корисним у різних сферах діяльності, пов’язаних з обробкою великих обсягів інформації.

APPLICATION, DOCKER ENVIRONMENT AND DOCKER-COMPOSE, PHP PROGRAMMING LANGUAGE, LARAVEL FRAMEWORK, OPTICAL CHARACTER RECOGNITION, PHOTO OPTIMIZATION METHODS.

The object of the work is to study the method of optical character recognition.

The purpose of the work is to develop a program for optical character recognition using the Tesseract OCR technology. The main task is to create software that can process photos with text and automatically recognize symbols on the image, which will increase the efficiency and accuracy of information processing.

PHP programming language, Laravel framework and application containerization using Docker are used. The created program has a simple and convenient interface for uploading images and displaying recognition results. Optimal datasets are selected to improve recognition quality.

The result of this work is a full-fledged software capable of efficiently processing photos with text, which can be useful in various fields of activity related to the processing of large volumes of information.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

HMM – Hidden Markov Model (прихована модель Маркова)

OCR – Optical Character Recognition (оптичне розпізнавання символів)

AWS – Amazon Web Services (вебсервіси Amazon)

MVC – Model View Controller

API – Application Programming Interface (програмний інтерфейс застосунків)

HSV – Hue Saturation Value (значення насиченості відтінку)

MNIST – Modified National Institute of Standards and Technology (модифікований національний інститут стандартів і технологій)

ВСТУП

У сучасному світі інформаційні технології займають чільне місце у багатьох сферах діяльності, вимагаючи від розробників нових інструментів підвищення ефективності роботи та оптимізації процесів. Серед завдань, які стоять перед розробниками, найактуальнішою є розробка системи OCR, яка може отримувати текстову інформацію з зображень.

Мета цієї кваліфікаційної роботи полягає в розробці застосунку оптичного розпізнавання символів для отримання тексту з фотографії. Для досягнення цієї мети розглянуті та проаналізовані різні методи обробки зображень та алгоритми розпізнавання тексту, а також здійснено їх адаптацію для використання у вебдодатку.

Проблематика, що розглядається в роботі, пов’язана з необхідністю розробки ефективної системи OCR, здатної отримувати текстову інформацію із зображень з високою точністю та швидкістю. У сучасному інформаційному суспільстві обсяги даних постійно зростають і важливо мати ефективні інструменти для обробки та аналізу цієї інформації. Ручне введення даних із паперових документів або зображень є трудомістким та неточним процесом, який потребує значних витрат часу та ресурсів. Тому розробка системи OCR стає важливим завданням для автоматизації процесу отримання текстової інформації та поліпшення ефективності роботи в різних галузях.

Ця кваліфікаційна робота має високу практичну значущість, оскільки може бути використана як основа для створення подібних систем у різних сферах діяльності. Результати дослідження можуть бути застосовані в галузі документообігу, архівування та зберігання інформації, що дозволить підвищити ефективність роботи у цих сферах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Огляд оптичного розпізнавання символів та його застосування

В даний час OCR є однією з найважливіших технологій для автоматичної обробки зображень, і широко застосовується у різних галузях, таких як банківська справа, медична документація, урядова документація, бізнес та освіта. В цій роботі розглянуто, що таке OCR, основні методи, які використовуються для його реалізації та переваги і недоліки, що супроводжують цю технологію.

Основними методами оптичного розпізнавання символів є методи шаблонного зіставлення та методи машинного навчання. Методи шаблонного зіставлення засновані на порівнянні форми символу, що розпізнається, з відомим набором шаблонів символів, а методи машинного навчання використовують набори даних, щоб навчити моделі на основі зображень символів. Також важливо відзначити, що в процесі OCR використовуються не тільки методи розпізнавання тексту, але й методи обробки зображень, такі як бінаризація і сегментація.

Застосування оптичного розпізнавання символів зараз поширене. У банківській справі OCR використовується для автоматичного розпізнавання інформації з банківських документів, таких як рахунки, чеки і т.д. У медичній документації, OCR застосовується для розпізнавання та перетворення медичних даних та документів пацієнтів. В урядових документах OCR використовується для автоматичного розпізнавання текстової інформації, яка збережена в електронній формі. У бізнесі, OCR використовується для автоматичного розпізнавання та обробки інформації на рівні компаній, таких як звіти та фінансові документи. В освіті, OCR застосовується для розпізнавання інформації у навчальних матеріалах, тестах та екзаменаційних роботах [1-3].

Перевагами OCR є швидкість і точність, які дозволяють швидко та ефективно обробляти велику кількість інформації. Крім того, OCR дозволяє автоматизувати процеси обробки інформації та підвищити продуктивність та ефективність роботи компаній. Недоліки ж включають можливість помилок при розпізнаванні, особливо у випадках з поганою якістю зображень або нерозрізненими символами, а також потреба в витратному обладнанні і професійному навчанні персоналу.

Також важливо відзначити, що OCR є частиною ширшої області комп’ютерного зору, яка включає різні методи аналізу та обробки зображень. Разом з тим, розвиток технологій, таких як штучний інтелект та глибоке навчання, призводить до покращення якості розпізнавання символів та розширення можливостей застосування OCR.

Таким чином, оптичне розпізнавання символів є важливою технологією автоматичної обробки зображень і має широке застосування в різних галузях, таких як банківська справа, медична документація, урядова документація, бізнес та освіта. Однак необхідність у витратному обладнанні та професійному навчанні персоналу, а також можливість помилок при розпізнаванні символів є основними недоліками цієї технології. Разом з тим, розвиток нових технологій та методів обробки зображень дозволяють покращити якість розпізнавання символів та розширити можливості застосування OCR у майбутньому.

1.2 Аналіз існуючих рішень для оптичного розпізнавання символів

OCR є важливим напрямком у сфері комп’ютерного зору. OCR дозволяє розпізнавати текст на зображеннях та перетворювати його на електронний вигляд, що спрощує та прискорює обробку даних та покращує їх доступність. Рішення для оптичного розпізнавання символів широко використовуються у багатьох галузях.

Існує безліч рішень для оптичного розпізнавання символів, які можна використовувати з PHP. Деякі з них є комерційними продуктами, а деякі – безкоштовними та з відкритим вихідним кодом. Розглянемо найвідоміші рішення.

1.2.1 Tesseract OCR

Одним із найпопулярніших рішень для оптичного розпізнавання символів з відкритим вихідним кодом є Tesseract OCR [4]. Tesseract OCR був розроблений у 1985 році в США і згодом був придбаний Google. У 2006 році Google передав вихідний код Tesseract OCR у суспільне надбання, після чого розробкою стало займатися ком'юніті. Tesseract OCR підтримує понад 100 мов та має високу точність розпізнавання. Tesseract OCR підтримує PHP через бібліотеку PHP Tesseract.

Для використання Tesseract у PHP існує декілька бібліотек, таких як Tesseract OCR for PHP та PHP Tesseract OCR. Обидві бібліотеки дозволяють легко інтегрувати Tesseract до PHP-додатків та забезпечують зручний інтерфейс для роботи з ним.

Перевага Tesseract OCR полягає в тому, що це безкоштовне програмне забезпечення з відкритим кодом. До нього можна легко отримати доступ і використовувати його в різних операційних системах, без ліцензійних зборів. Ще однією важливою перевагою є його багатомовність, що дозволяє обробляти текст різними мовами. Це робить його особливо корисним для організацій і компаній, які працюють з багатомовним контентом.

Крім того, Tesseract OCR простий у використанні та інтеграції в існуючі проєкти. Він має добре задокументований API і бібліотеки, доступні для багатьох мов програмування, що полегшує його застосування в різних програмах. Tesseract OCR відомий своєю високою точністю розпізнавання символів, що важливо для обробки великих обсягів даних.

З іншого боку, є деякі обмеження щодо використання Tesseract OCR. Одним із істотних недоліків є недосконале розпізнавання символів. У деяких випадках Tesseract OCR може неправильно інтерпретувати символи, особливо на зображеннях із низькою роздільною здатністю або погано освітлених зображеннях. Tesseract OCR може зіткнутися з труднощами при роботі з деякими з найновіших форматів зображень, які не були включені в його базову версію.

Таким чином, Tesseract OCR є широко використовуваним і універсальним інструментом для розпізнавання тексту із зображень. Його переваги включають безкоштовне програмне забезпечення з відкритим вихідним кодом і багатомовність, а також високу точність і легкість інтеграції. Загалом Tesseract OCR є корисним інструментом, який вимагає ретельного розгляду його можливостей і обмежень для кожної програми.

1.2.2 OCRopus

Ще одним рішенням для оптичного розпізнавання символів із відкритим вихідним кодом є OCRopus. OCRopus був створений у 2006 році в Центрі комп’ютерного зору Німецького наукового центру інформатики та є інструментарієм для обробки текстових документів [5]. OCRopus має відкритий вихідний код та може бути використаний у комерційних проєктах. OCRopus підтримує кілька мов та має можливість адаптивного навчання. OCRopus підтримує PHP через бібліотеку PHP OCRopus.

OCRopus написана мовою Python, має більш складну архітектуру, що включає безліч модулів, які дозволяють виконати всі етапи обробки зображень, включаючи сегментацію сторінок, бінаризацію, розпізнавання тексту та виправлення помилок. OCRopus дозволяє використовувати різні алгоритми розпізнавання, такі як Tesseract та CuneiForm. Однак, через свою

складність, OCRopus може бути складним для використання та вимагає глибоких знань програмування.

Перевагою OCRopus є точність розпізнавання друкованого тексту. OCRopus використовує складний алгоритм, який може точно розпізнавати друковані символи навіть у відсканованих зображеннях низької якості. Це важливо для організацій, які мають справу з великою кількістю друкованого тексту.

OCRopus підтримує широкий спектр форматів зображень, що полегшує використання в різних налаштуваннях. OCRopus має простий інтерфейс користувача, який дозволяє користувачам легко інтегрувати його у свій робочий процес.

OCRopus можна налаштувати. Це дозволяє користувачам навчити його розпізнавати певні шрифти або символи, що робить його дуже придатним для завдань, які потребують спеціального розпізнавання символів, що особливо корисно для організацій, які мають особливі вимоги до розпізнавання символів.

З іншого боку, OCRopus має деякі недоліки. Одним з головних недоліків є його складність. Користуватися OCRopus може бути складно, особливо для тих, хто не має сильного технічного досвіду. Крім того, для ефективної роботи OCRopus потрібна значна кількість обчислювальних ресурсів, що може бути обмеженням для деяких організацій.

Іншим недоліком OCRopus є обмежена підтримка нелатинських шрифтів. OCRopus добре розпізнає латинські шрифти, але його підтримка нелатинських шрифтів, таких як арабська чи китайська, обмежена.

З точки зору сумісності з мовами програмування, OCRopus надає API для різних мов програмування, включаючи Python і Java, але його сумісність з PHP обмежена. Це може бути недоліком для організацій, які використовують PHP як основну мову програмування.

Отже, OCRopus – це потужне програмне забезпечення OCR, яке забезпечує високу точність і гнучкість обробки різних форматів зображень.

Його переваги включають точність, гнучкість і можливість налаштування. Однак його недоліки включають його складність, обмежену підтримку нелатинських скриптів і обмежену сумісність з PHP. Незважаючи на свої обмеження, OCRopus є цінним інструментом для організацій, яким потрібно розпізнавати друкований текст зі сканованих зображень.

1.2.3 Abbyy FineReader

Abbyy FineReader ще одне рішення для оптичного розпізнавання символів. Abbyy FineReader – це комерційне рішення, яке має широкий спектр функцій. Ця програма здатна розпізнавати текст та символи на зображеннях, сканованих документах та PDF-файлах за допомогою OCR. Abbyy FineReader підтримує понад 190 мов [6].

Abbyy FineReader має безліч функцій, які роблять його одним із найпопулярніших OCR-програм у світі. Він має точність розпізнавання символів понад 99%, що дозволяє обробляти навіть складні документи.

Abbyy FineReader має функцію управління документами, що дозволяє швидко та ефективно працювати з великими обсягами документів. Його функціональність включає можливість конвертувати документи у різні формати, такі як Word, Excel, HTML та інші. Abbyy FineReader також має можливість працювати із зображеннями, що робить його дуже корисним для фотографів та дизайнерів. Він може розпізнавати текст на фотографіях, обробляти зображення та покращувати їхню якість.

В цілому, Abbyy FineReader є потужним інструментом для розпізнавання тексту та символів, який може бути корисним у багатьох сферах, включаючи офісну роботу, фотографію, дизайн та багато іншого.

Перевагою Abbyy FineReader є висока точність розпізнавання друкованого тексту. Abbyy FineReader використовує вдосконалену технологію оптичного розпізнавання символів для точного розпізнавання друкованих

символів навіть у відсканованих зображеннях низької якості. Це робить його особливо корисним для організацій, які мають справу з великою кількістю друкованого тексту.

Зручний інтерфейс є ще однією перевагою Abbyy FineReader. Abbyy FineReader має простий та інтуїтивно зрозумілий інтерфейс користувача, який дозволяє користувачам легко орієнтуватися та використовувати програмне забезпечення.

Крім того, Abbyy FineReader має широкий набір функцій, які роблять його універсальним програмним забезпеченням OCR. Він підтримує різні формати зображень і надає потужні інструменти для редагування та форматування документів. Крім того, Abbyy FineReader має багатомовні можливості, що робить його придатним для організацій, які працюють з багатомовним контентом.

З іншого боку, Abbyy FineReader має деякі обмеження. Одним з основних обмежень є його вартість. Abbyy FineReader – комерційне програмне забезпечення, для використання якого потрібна ліцензія. Це може бути значним коштом для деяких організацій, особливо малих підприємств або окремих осіб.

Ще одним обмеженням Abbyy FineReader є його сумісність з мовами програмування. Хоча Abbyy FineReader надає API для різних мов програмування, він може бути не сумісний з усіма мовами програмування, включаючи PHP.

Таким чином, Abbyy FineReader – це потужне програмне забезпечення OCR, яке забезпечує високу точність і зручний інтерфейс. Його переваги включають точність, зручний інтерфейс і широкий набір функцій. Однак його обмеження включають вартість і сумісність з мовами програмування. Незважаючи на свої обмеження, Abbyy FineReader є цінним інструментом для організацій, яким потрібно розпізнавати друкований текст зі сканованих зображень.

1.2.4 GOCR

GOCR є ще одним рішенням для оптичного розпізнавання символів з відкритим кодом для розпізнавання символів, розроблене в рамках проєкту GNU. GOCR надає реалізацію OCR для кількох мов, включаючи англійську, французьку, німецьку та інші. Він може працювати як з відсканованими зображеннями, так і цифровими фотографіями [7].

Для використання GOCR з мовою PHP доступна низка бібліотек та інструментів. Одним із найпопулярніших способів є використання бібліотеки Tesseract PHP Wrapper, яка надає інтерфейс для роботи з GOCR через PHP.

Однак, варто відзначити, що GOCR не має такої високої точності, як комерційні продукти. GOCR залишається популярним інструментом для розпізнавання символів завдяки своїй свободі та відкритості, а також можливості розширення та налаштування функціоналу відповідно до потреб користувача.

Недоліком GOCR є те, що він не такий точний, як деякі інші рішення. Він не завжди правильно розпізнає символи у складних шрифтах і може допускати помилки за наявності шуму на зображенні.

1.2.5 Texttract

Texttract – це програмне забезпечення OCR, розроблене AWS, призначене для вилучення тексту та даних із різних документів.

Texttract має високу точність розпізнавання друкованого та рукописного тексту. Texttract використовує алгоритми машинного навчання, для точного розпізнавання тексту і даних з різних документів, включаючи скановані зображення та PDF-файли. Це робить його особливо корисним для організацій, які мають справу з великою кількістю тексту та даних [8].

Перевагою Texttract є його інтеграція з іншими сервісами AWS, такими як S3 і Lambda. Це дозволяє організаціям легко інтегрувати Texttract у свою

існуючу інфраструктуру AWS, що робить його цінним інструментом для організацій, які покладаються на AWS. Texttract має простий і зручний API, який дозволяє розробникам легко інтегрувати Texttract у свої програми. Texttract надає ряд функцій, які роблять його універсальним програмним забезпеченням OCR, включаючи підтримку різних форматів документів і можливість витягувати таблиці та форми.

Одним з основних обмежень Texttract є його вартість. Texttract - це комерційне програмне забезпечення, для використання якого потрібна підписка. Це може бути не зручно для деяких організацій, особливо малих підприємств або окремих осіб.

Ще одним обмеженням Texttract є його обмежена підтримка нелатинських мов. Texttract добре розпізнає латинський текст, але його підтримка нелатинських мов обмежена.

З точки зору сумісності з мовами програмування, Texttract надає API для різних мов програмування, включаючи PHP. Однак його інтеграція з PHP може вимагати додаткової конфігурації та налаштування, що може бути обмеженням для деяких організацій.

Отже, Texttract – це потужне програмне забезпечення OCR, яке забезпечує високу точність та інтеграцію з іншими сервісами AWS. Його переваги включають його точність, інтеграцію з іншими службами AWS, зручний API та різноманітні функції. Однак його обмеження включають вартість, обмежену підтримку нелатинських мов і потенційні обмеження в інтеграції з PHP. Незважаючи на свої обмеження, Texttract є цінним інструментом для організацій, яким потрібно отримувати текст і дані з різних документів.

1.3 Постановка задачі

При створенні додатка оптичного розпізнавання символів для отримання тексту з фотографіями, необхідно вирішити таке завдання: створення системи, яка здатна обробляти зображення, що містять символьну інформацію, і конвертувати її в текстовий формат. Це завдання складається з кількох етапів: створення оточення, попередньої обробки зображення, розпізнавання символів, аналізу розпізнаного тексту та його конвертації у потрібний формат.

Перий етап полягає в створенні оточення, на якому будуть проходити процеси обробки зображення. В результаті має бути універсальне оточення, яке можна буде запустити з будь-якого пристрою.

Другий етап полягає в попередній обробці зображення, яка може включати такі операції, як зменшення шуму, поліпшення контрастності, приведення до потрібного формату. В результаті цієї обробки зображення має бути готовим для подальшого розпізнавання символів.

Третій етап полягає у розпізнаванні символів на обробленому зображенні. Для цього можна використовувати різні методи, такі як нейромережні алгоритми, методи машинного навчання та ін. Основне завдання на цьому етапі – визначити, які символи знаходяться на зображенні та які саме це символи.

Четвертий етап полягає у конвертації розпізнаного тексту у потрібний формат. Наприклад, якщо потрібно скопіювати текст із фото і вставити їх у текстовий документ, його потрібно скопіювати і зберегти у відповідному форматі.

Таким чином, основне завдання розробки програми оптичного розпізнавання символів для отримання тексту з фотографії – створення системи, здатної обробити зображення з символьною інформацією, розпізнати символи на ньому і конвертувати їх у потрібний формат текстового документа.

Об’єтом роботи є дослідження методу оптичного розпізнавання символів.

Метою роботи є розробка програми для оптичного розпізнавання символів з використанням технології Tesseract OCR. Основним завданням є створення програмного забезпечення, яке здатне обробляти фотографії з текстом та автоматично розпізнавати символи на зображенні, що дозволить підвищити ефективність та точність обробки інформації.

Для досягнення мети необхідно вирішити такі завдання:

- розробити уніфіковану систему за допомогою Docker;
- інтегрувати Tesseract OCR для мови PHP;
- реалізувати кастомні методи для предобробки зображення;
- реалзувати обробку зображення за допомогою Tesseract OCR;
- реалізувати отримання файлу з текстом після обробки зображення.

2 МАТЕМАТИЧНЕ ОБҐРУНТУВАННЯ ВИБРАНИХ МЕТОДІВ

2.1 Огляд методів обробки зображень для покращення якості розпізнавання символів, включаючи видалення шуму, збільшення контрастності та підвищення різкості

Обробка зображень – це процес перетворення зображень у цифровому форматі для полегшення їх аналізу та обробки. В області OCR обробка зображень є ключовим кроком у процесі розпізнавання тексту. Вона допомагає покращити якість зображення, збільшити точність розпізнавання символів та знизити кількість помилок під час розпізнавання.

У цьому огляді розглянемо різні методи обробки зображень, які використовуються для покращення якості розпізнавання символів. Почнемо з огляду основних проблем, пов’язаних із зображеннями, які можуть вплинути на якість розпізнавання, а потім перейдемо до розгляду методів обробки зображень, які можуть вирішити ці проблеми.

Однією з основних проблем, пов’язаних із зображеннями, є низька контрастність. Зображення з низькою контрастністю не можуть бути розпізнані OCR-системами, оскільки символи можуть зливатися з фоном. Для вирішення цієї проблеми використовують різні методи обробки зображень, такі як збільшення контрастності, збільшення яскравості або зміна кольору зображення.

Ще однією проблемою, пов’язаною із зображеннями, є наявність шумів на зображенні. Шуми можуть включати різні артефакти, такі як крапки, лінії або інші несподівані об’єкти, які можуть заважати процесу розпізнавання. Для вирішення цієї проблеми використовують методи, такі як фільтрація зображень або видалення шуму за допомогою алгоритмів обробки зображень.

Крім того, існує проблема розмитих зображень, які можуть бути викликані рухом або недостатнім фокусуванням камери під час зйомки. Ця проблема вирішується за допомогою методів обробки зображень, таких як відновлення або покращення різкості.

Розглянемо методи обробки зображень, які використовуються для сегментації зображень та виділення текстових блоків на зображенні. Сегментація зображень є важливим кроком у процесі розпізнавання символів, оскільки вона дозволяє розбити зображення на окремі блоки тексту, що спрощує завдання OCR-системи.

Одним із методів сегментації зображень є порогова обробка, яка заснована на розподілі зображення на дві групи пікселів: тих, що є текстовими елементами, і тих, що не є текстовими елементами. Для цього застосовується пороговий фільтр, який відкидає всі пікселі, чий рівень яскравості менший за заданий поріг. Таким чином, на виході виходить зображення, яке складається лише з текстових елементів.

Ще одним методом сегментації є використання морфологічних операцій, таких як розширення, звуження, відкриття та закриття. Вони ґрунтуються на зміні форми об’єктів на зображенні, що може бути корисним для видалення шуму або об’єднання суміжних об’єктів в один блок тексту.

Існують також методи, які використовують машинне навчання для автоматичного виділення текстових блоків на зображенні. Вони ґрунтуються на навчанні класифікаторів, які можуть визначати, чи є область на зображенні текстовим блоком чи ні.

Крім сегментації, також важливим кроком у обробці зображень для розпізнавання символів є покращення якості зображення. Одним з методів покращення якості є фільтрація зображень, яка заснована на видаленні шуму та покращенні контрастності.

Також використовують методи обробки зображень для розпізнавання символів на поганих якостях, наприклад, на старих документах або факсимільних копіях. Для цього можуть бути використані методи відновлення

зображень, які дозволяють покращити якість зображення та зробити його більш придатним для розпізнавання символів.

Обробка зображень відіграє важливу роль у процесі розпізнавання символів і може значно покращити точність та якість роботи OCR-систем. Існує безліч методів обробки зображень, які можуть бути використані для вирішення різних проблем, пов’язаних із зображеннями, та вибір певного методу залежить від конкретних умов завдання [9-13].

2.1.1 Аналіз видалення шуму

Одним із найпопулярніших прийомів попередньої обробки зображення для підвищення точності ідентифікації символів є видалення шуму. Небажані пікселі, які з’являються внаслідок технологічних обмежень, поганої якості сигналу або інших аспектів, які можуть вплинути на якість зображення, називаються шумом зображення [14].

Усунення шуму часто передбачає використання фільтрів, таких як медіанний фільтр, фільтр Гауса, проміжний фільтр, двосторонній фільтраційний фільтр та інші. Вони використовуються для усунення різноманітних шумів, у тому числі шуму Пуассона, шуму Гауса та інші.

Медіанний фільтр – це один із найбільш популярних фільтрів для видалення шуму на зображенні. Він замінює значення кожного пікселя на медіану значень пікселів у його околиці. Це дозволяє видаляти шум, не видаляючи при цьому важливу інформацію із зображення.

Гаусов фільтр використовується для видалення гаусівського шуму, який є одним із найпоширеніших видів шуму на зображеннях. Він замінює значення кожного пікселя на виважену суму значень пікселів на його околиці, де ваги визначаються гаусовим розподілом.

Фільтр усереднення використовується для видалення шуму, що створюється випадковими флуктуаціями яскравості на зображенні. Він

замінює значення кожного пікселя на середнє значення пікселів у його околиці.

Фільтр білатеральної фільтрації використовується для видалення шуму, зберігаючи при цьому ребра на зображенні. Він замінює значення кожного пікселя на зважену суму значень пікселів у його околиці, де ваги визначаються як функція відстані між пікселями і різницею яскравості з-поміж них.

Ці методи та інші можуть бути використані для видалення шуму на зображеннях, що покращує якість розпізнавання символів

2.1.2 Аналіз збільшення контрастності

Одним із методів обробки зображень, що використовуються для покращення якості розпізнавання символів, є збільшення контрастності. Цей метод спрямований на підвищення різниці між яскравістю об’єкта та фону, що покращує читаність символів [15].

Збільшення контрастності можна досягти різними способами. Один із таких способів – зміна яскравості та насиченості зображення. І тому можна використовувати колірну модель HSV, у якій значення V відповідає яскравість зображення. Збільшення значення V призводить до збільшення яскравості зображення та підвищення контрастності.

Ще одним способом збільшення контрастності є лінійне розтягування гістограми. Гістограма показує розподіл яскравості пікселів на зображенні. Лінійне розтягування гістограми полягає у розтягуванні гістограми вздовж осі яскравості, таким чином, щоб використати весь діапазон яскравості зображення. Це може призвести до більш контрастного зображення, що покращує якість розпізнавання символів.

Важливо, що збільшення контрастності може збільшити шум на зображенні. Тому необхідно знаходити баланс між підвищенням контрастності та збільшенням шуму на зображенні.

2.1.3 Аналіз підвищення різкості

Підвищення різкості зображення – це один із методів обробки зображень для покращення якості розпізнавання символів. Цей метод полягає у збільшенні кількості кордонів на зображенні, що робить зображення чіткішим і різкішим. Більш чітке та різке зображення спрощує процес розпізнавання символів, оскільки символи стають чіткішими [16].

Один із способів підвищення різкості зображення – це використання фільтра збільшення різкості, який збільшує різницю в яскравості між пікселями, що створює враження різкішого зображення. Інший спосіб – це використання операторів градієнта, які виявляють зміни в яскравості зображення і допомагають посилити межі між об’єктами на зображенні.

При реалізації цього методу важливо пам’ятати, що занадто сильне збільшення різкості може призвести до появи артефактів і шумів на зображенні, що може негативно вплинути на якість розпізнавання символів. Тому необхідно підібрати оптимальний рівень підвищення різкості, який забезпечує достатню чіткість та різкість зображення, але не призводить до появи шумів та артефактів.

2.2 Огляд алгоритму оптичного розпізнавання символів Tesseract OCR

Tesseract OCR – програмний інструмент, що вільно розповсюджується, призначений для розпізнавання тексту на зображеннях. Tesseract OCR був створений у HP Labs у 1985 році та у 2005 році був випущений у відкритому доступі під ліцензією Apache.

Основним алгоритмом Tesseract OCR є нейронна мережа, яка навчається на великій кількості зображень із текстом. Важливо, що Tesseract OCR може розпізнавати текст різними мовами, включаючи складні системи письма, такі як китайська або японська.

Архітектура Tesseract OCR складається з трьох основних модулів: попередньої обробки зображення, розпізнавання тексту та постобробки результатів. Розглянемо кожен із цих модулів докладніше

Попередня обробка зображення. Цей модуль призначений для покращення якості зображення та підготовки його до подальшої обробки. Він включає такі операції, як зменшення шуму, збільшення контрастності і підвищення різкості.

У модулі розпізнавання тексту відбувається саме розпізнавання тексту. Він використовує нейронну мережу і словник слів знаходження найімовірніших варіантів розпізнавання символів і слів.

Для більш детального опису алгоритму розпізнавання можна подати формули. Позначимо X як вхідне зображення, а Y як результат розпізнавання. Алгоритм розпізнавання може бути представлений такою формулою:

$$Y = f(X), \tag{2.1}$$

де f – функція, яка зіставляє вхідне зображення Y з результатом розпізнавання.

Модуль постобробки результатів використовується для покращення якості розпізнавання тексту. Він включає такі операції, як виправлення помилок розпізнавання, фільтрацію неправильних символів і повторне розпізнавання тих частин зображення, які не вдалося розпізнати з першого разу.

Також для оцінки якості розпізнавання Tesseract OCR використовує метрику, яка називається Precision and Recall. Precision – це частка вірно розпізнаних символів від загальної кількості символів, розпізнаною системою. Recall – це частка чітко розпізнаних символів від загальної кількості символів, присутніх на зображенні. Тобто Precision показує, наскільки точно система розпізнала символи, а Recall – наскільки повно система змогла розпізнати всі символи, що були на зображенні.

Також у Tesseract OCR використовуються словники для покращення розпізнавання. Ці словники можуть містити список слів, які знаходяться у конкретній тематичній області. Наприклад, для розпізнавання текстів на медичних зображеннях можна використовувати словник, що містить терміни та назви медичних препаратів.

Однією з головних переваг Tesseract OCR є його відкритий вихідний код та активна розробка спільнотою. Це дозволяє постійно покращувати та вдосконалювати алгоритми розпізнавання символів, додавати нові функції та покращувати якість роботи системи в цілому.

Недоліком Tesseract OCR може бути його відносно низька швидкість роботи. Однак, при правильному налаштуванні та використанні оптимізації, можна досягти прискорення роботи системи.

Загалом Tesseract OCR є потужним та ефективним інструментом для оптичного розпізнавання символів, який може бути використаний у різних сферах, таких як медицина, банківська справа, наука та технічна документація.

2.2.1 Аналіз бінаризація зображення

Бінаризація зображення в Tesseract OCR – це процес перетворення кольорового зображення на зображення, що складається тільки з чорних та білих пікселів. Це робиться для спрощення подальшої обробки зображення та розпізнавання тексту на ньому.

У Tesseract OCR бінаризація зображення зазвичай виконується за допомогою алгоритму граничної обробки, який визначає поріг яскравості, при якому кожен піксель буде вважатися чорним або білим. Зазвичай цей поріг визначається з яскравості пікселів у зображенні. Якщо піксель яскравіший за поріг, він стає білим, а якщо темнішим – чорним.

Бінаризація зображення в Tesseract OCR є важливим кроком у процесі розпізнавання тексту і може підвищити точність розпізнавання, видаливши шуми та покращивши контрастність зображення [17].

У алгоритмі бінаризації зображення використовується порогова бінаризація, яка дозволяє розділити зображення на частини: чорну і білу. Для визначення порога бінаризації використовується формула Оцу:

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) , \quad (2.2)$$

де $\sigma_w^2(t)$ – дисперсія внутрішньокласового розподілу пікселів;

$q_1(t)$ – ймовірність того, що піксель належить першому класу (пікселі з інтенсивністю менше від порога);

$\sigma_1^2(t)$ – дисперсія інтенсивності пікселів першого класу;

$q_2(t)$ – ймовірність того, що піксель належить другому класу (пікселі з інтенсивністю більше за поріг);

$\sigma_2^2(t)$ – дисперсія інтенсивності пікселів другого класу.

2.2.2 Аналіз сегментація зображення

Сегментація зображення в Tesseract OCR – це процес розбиття зображення на окремі частини, які називають сегментами, які містять окремі символи або слова тексту.

Сегментація зображення в Tesseract OCR виконується після бінаризації зображення та використовується для того, щоб збільшити точність розпізнавання тексту. Кожен сегмент зображення проходить через процес розпізнавання тексту індивідуально, що дозволяє покращити точність розпізнавання, оскільки кожен сегмент містить лише один символ чи слово.

Для сегментації зображення в Tesseract OCR використовуються різні алгоритми, включаючи алгоритми, що базуються на кольорі, текстурі та формі об’єктів на зображенні. Ці алгоритми дозволяють автоматично визначати межі між символами та словами на зображенні.

Сегментація зображення є важливим кроком у процесі розпізнавання тексту в Tesseract OCR, оскільки вона дозволяє покращити точність розпізнавання та забезпечити більш точне вилучення інформації із зображення.

В алгоритмі сегментації зображення окремі символи використовується алгоритм connected components. Він заснований на пошуку зв’язкових компонентів на бінаризованому зображенні [18–20]. Для цього використовується алгоритм:

– ініціалізація порожнього списку компонентів;

– обхід усіх пікселів зображення по рядкам;

– якщо поточний піксель чорний і не належить до жодної компоненти, то створюється нова компонента і додається до списку компонентів. Потім виконується обхід углиб для пошуку всіх суміжних чорних пікселів, які належать цій компоненті;

– повторюється попередній пункт для всіх чорних пікселів, що не належать жодному компоненту.

У Tesseract OCR сегментація зображення на окремі символи реалізована таким чином:

– перетворення зображення у відтінки сірого та застосування порогової фільтрації для отримання бінаризованого зображення;

– застосування алгоритму connected components для пошуку зв’язкових компонентів на бінаризованому зображенні;

– для кожної знайденої зв’язкової компоненти визначається її межа та витягується як окремий символ.

Формули для цього алгоритму:

– перетворення зображення на відтінки сірого:

$$Gray = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B\,, \qquad (2.3)$$

де *R*, *G* і *B* – червоний, зелений та синій канали пікселя відповідно;

– порогова фільтрація:

$$binary = (gray < threshold)?\,0:1\,, \tag{2.4}$$

де *threshold* – заданий поріг;

– алгоритм connected components;

– визначення межі зв’язкової компоненти:

1) ініціалізація лівої, правої, верхньої та нижньої межі;

2) обхід усіх пікселів зв’язного компонента;

3) якщо поточний піксель чорний, то оновлюються ліва, права, верхня та нижня.

2.2.3 Аналіз алгоритму прихованих Марківських моделей

Алгоритм HMM використовується в Tesseract OCR для розпізнавання тексту на зображеннях. Він використовує імовірнісні методи визначення найбільш ймовірної послідовності символів на зображенні.

HMM – це статистична модель, яка є ймовірнісним автоматом, що складається з набору прихованих станів, що спостерігаються, і ймовірнісних переходів між ними. HMM можна використовувати для моделювання тимчасових послідовностей, таких як текст на зображеннях.

Основна ідея HMM полягає в тому, що дані (зображення символу) залежать тільки від прихованого стану (який символ знаходиться на зображенні) і не залежать від інших даних і станів, що спостерігаються. Для кожного символу на зображенні створюється своя модель HMM, яка містить набір прихованих станів, станів, що спостерігаються, і ймовірнісних переходів між ними [21–23].

Формули для алгоритму HMM:

– алгоритм прямого поширення:

$$a_j(t) = \sum_i a_i(t-1)a_{ij}b_j(y_t)\ , \qquad (2.5)$$

де $a_j(t)$ – ймовірність генерації часткової послідовності;

– алгоритм зворотного розповсюдження:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{j=1}^{N}\alpha_t(j)\beta_t(j)}\ , \qquad (2.6)$$

де $\gamma_t(i)$ – апостеріорна ймовірність перебування у прихованому стані і, в момент часу *t*;

$\alpha_t(i)$ – пряма ймовірність перебування у стані і, в момент часу *t*;

$\beta_t(i)$ – зворотна ймовірність перебування у стані і, в момент часу *t*;

N – загальна кількість прихованих станів у моделі HMM;

– алгоритм Вітербі:

$$\delta_t(t) = \max_i[\delta_i(t-1)a_{ij}b_j(y_t)]\ , \qquad (2.7)$$

де $\delta_t(t)$ – найкраща ймовірність попадання у прихований стан і та спостереження перших *t* кроків.

2.3 Опис вибору датасетів для покращення якості розпізнавання символів

Одним з важливих аспектів навчання OCR–систем, таких як Tesseract OCR, є вибір правильного датасета для навчання. Датасети є набором зразків символів, які використовуються для навчання OCR-системи та допомагають їй розпізнавати символи на зображеннях.

Існує безліч датасетів для навчання OCR-систем, і вибір конкретного датасета залежить від ряду факторів, таких як тип даних, на яких

використовуватиметься OCR-система, та якість зображень, які будуть оброблятися [24–30].

Одним із найпоширеніших датасетів є MNIST, який містить зображення рукописних цифр. Він часто використовується для навчання нейронних мереж, оскільки містить велику кількість зразків, які можуть бути використані для тренування мережі.

Однак при використанні Tesseract OCR рекомендується використовувати датасети, надані в офіційному репозиторії Tesseract OCR. Вони створені спеціально для використання з Tesseract OCR та містять велику кількість символів, які зазвичай зустрічаються у різних типах документів, таких як книги, статті та договори.

Крім того, датасети з офіційного репозиторію проходять ретельну перевірку та тестування, що гарантує їхню високу якість та точність при використанні з Tesseract OCR.

Важливо також враховувати, що не всі датсети підходять для всіх типів даних та зображень. Наприклад, датасет, який містить лише рукописний текст, може не відповідати розпізнаванню символів на друкованих документах.

Також можна використовувати додаткові методи, такі як аугментація даних для покращення якості навчання OCR-системи. Аугментація даних є методом розширення датасета за рахунок застосування різних трансформацій до зображень, наприклад, повороту, зміни масштабу або додавання шуму.

При виборі датасету для навчання OCR-системи, необхідно враховувати ряд факторів, таких як тип даних та якість зображень, а також використовувати датасети з офіційного репозиторію Tesseract OCR, які пройшли перевірку та тестування на відповідність вимогам OCR системи. Однак, варто враховувати, що датасети з офіційного репозиторію можуть бути обмежені в обсязі та змісті. У цьому випадку можна звернутися до інших ресурсів, таких як відкриті бази даних, наприклад MNIST, CIFAR-10, COCO та інші.

MNIST – це база даних, що містить зображення рукописних цифр, яка широко використовується для тестування алгоритмів розпізнавання символів.

У цій базі даних представлено понад 70 000 зображень, які можна використовувати для навчання OCR-системы.

CIFAR-10 – це база даних, що містить зображення об’єктів, що розмічені на 10 класів, таких як автомобілі, літаки, птахи і т.д. У цій базі даних представлено понад 60 000 зображень, які можуть бути використані для навчання OCR-системи.

COCO – це база даних, що містить зображення об’єктів розмічених на кілька класів, таких як люди, тварини, транспортні засоби тощо. Ця база даних містить понад 330 000 зображень, які можна використовувати для навчання OCR-системы.

Незважаючи на те, що використання відкритих баз даних може бути корисним для навчання OCR-системи, деякі з баз даних можуть містити зображення низької якості або не відповідають вимогам OCR-системи. Тому при виборі датасета для навчання OCR-системи необхідно проаналізувати його зміст і переконатися, що він відповідає вимогам OCR-системи.

Загалом, для навчання OCR-системи краще використовувати датасети з офіційного репозиторію Tesseract OCR, оскільки вони пройшли перевірку та тестування на відповідність вимогам OCR-системи. Однак, якщо потрібно використовувати інші датасети, необхідно проаналізувати їх зміст та якість зображень, щоб переконатися, що вони підходять для навчання OCR-системи.

2.4 Оцінка якості розпізнавання символів за допомогою вибраних методів

Для оцінки якості розпізнавання символів за допомогою Tesseract OCR існує кілька підходів, кожен з яких має свої переваги та недоліки. Перший підхід – візуальний аналіз результатів дозволяє користувачеві швидко оцінити якість розпізнавання шляхом візуального порівняння зображень та відповідних розпізнаних текстів. Однак, цей метод не є достатньо надійним,

тому що його результати можуть бути сильно залежати від досвіду та переваг конкретного користувача.

Другий підхід – аналіз помилок розпізнавання, може бути об’єктивнішим. У цьому випадку проводиться аналіз помилок, що здійснюються Tesseract OCR при розпізнаванні тексту зображень. Такий підхід може допомогти визначити області, в яких Tesseract OCR демонструє кращу і найгіршу якість розпізнавання.

Третій підхід – використання стандартних метрик якості може бути найбільш об’єктивним, оскільки він ґрунтується на математичних обчисленнях. Для цього використовують стандартні метрики якості, такі як Precision та Recall, які порівнюють результати, отримані Tesseract OCR, з очікуваними результатами.

Однак, для досягнення найкращої якості розпізнавання символів за допомогою Tesseract OCR, необхідно використовувати не лише офіційні датасети, але й кастомні методи видалення шуму, підвищення контрастності та збільшення різкості. Ці методи можуть значно підвищити якість розпізнавання, особливо у випадках, коли вхідні зображення мають погану якість.

При виборі датасетів для навчання OCR-системи необхідно враховувати ряд факторів, таких як тип даних та якість зображень. Однак найбільш рекомендованим варіантом є використання датасетів з офіційного репозиторію Tesseract OCR, які пройшли перевірку та тестування на відповідність вимогам системи.

Загалом оцінка якості розпізнавання символів за допомогою Tesseract OCR може бути досить складним завданням, яке може вимагати кількох підходів та інструментів. Однак, правильний вибір датасетів та методів обробки зображень може суттєво покращити результати, отримані за допомогою Tesseract OCR.

При виборі датасету на навчання OCR-системы необхідно враховувати низку чинників. По-перше, слід визначити тип даних, які

використовуватимуться для розпізнавання символів. Для цього може знадобитися використання спеціальних датасетів, наприклад, для розпізнавання рукописних символів або символів на фотографіях.

Крім того, важливо враховувати якість зображень, на яких будуть розпізнавати символи. Низька якість зображень може суттєво погіршити результати розпізнавання. Тому, якщо зображення мають погану якість, необхідно використовувати методи обробки зображень, такі як видалення шуму, підвищення контрастності і збільшення різкості, які можуть значно поліпшити результати розпізнавання.

Крім того, для отримання найкращих результатів слід використовувати датасети з офіційного репозиторію Tesseract OCR, які пройшли перевірку та тестування на відповідність вимогам. Офіційні датасети зазвичай містять велику кількість різних символів та мов, що дозволяє розпізнавати тексти різними мовами.

Однак, у деяких випадках, може знадобитися використання кастомних датасетів, які містять специфічні символи або мови, які не були представлені в офіційних датасетах. У такому випадку необхідно провести ретельну перевірку якості таких датасетів та використовувати методи обробки зображень, щоб покращити результати розпізнавання.

Крім датасетів, важливо враховувати методи обробки зображень для оцінки якості розпізнавання символів за допомогою Tesseract OCR. Ці методи можуть значно підвищити якість розпізнавання, особливо у випадках, коли вхідні зображення мають погану якість. Наприклад, методи видалення шуму, підвищення контрастності та збільшення різкості можуть суттєво покращити якість розпізнавання, дозволяючи Tesseract OCR правильно розпізнавати символи.

Також можна використовувати стандартні метрики якості, такі як Precision та Recall, для оцінки якості розпізнавання символів. Precision – це частка правильно розпізнаних символів щодо всіх символів, розпізнаних Tesseract OCR, а Recall – це частка правильно розпізнаних символів щодо всіх

символів у вихідному тексті. Ці метрики можуть бути особливо корисними у випадку, якщо потрібно порівняти результати розпізнавання Tesseract OCR з іншими OCR-системами.

Однак важливо розуміти, що використання стандартних метрик може бути недостатнім для повної оцінки якості розпізнавання символів за допомогою Tesseract OCR. Це з тим, що це метрики не враховують контекст розпізнавання, наприклад, порядок слів у реченні. Також ці метрики не описують якість розпізнавання окремих символів чи слів, лише оцінюють якість загалом.

Тому для більш повної оцінки якості розпізнавання символів Tesseract OCR рекомендується використовувати кілька підходів, включаючи візуальний аналіз результатів, аналіз помилок розпізнавання та використання стандартних метрик якості. Крім того, необхідно вибирати відповідні датасети та методи обробки зображень для покращення якості розпізнавання.

Офіційні датасети з репозиторію Tesseract OCR можуть бути хорошим вибором для початку, оскільки вони пройшли тестування та перевірку на відповідність вимогам Tesseract OCR. Однак для більш специфічних завдань може знадобитися створення власних датасетів, які будуть відповідати конкретним потребам користувачів.

Крім вибору датасетів, також важливо враховувати якість вхідних зображень. Для досягнення найкращої якості розпізнавання символів необхідно використовувати зображення високої роздільної здатності та хорошої якості. Якщо вхідні зображення мають шум, низьку контрастність або низьку різкість, можна використовувати різні методи обробки зображень, такі як фільтрація шуму, підвищення контрастності та збільшення різкості.

3 РЕАЛІЗАЦІЯ ПОСТАНОВКИ ЗАДАЧІ ТА ТЕСТУВАННЯ

3.1 Опис архітектури програми

Однією з найважливіших складових розробки програми є архітектура, яка визначає структуру та взаємодію всіх його компонентів. Архітектура додатка має бути гнучкою, масштабованою та легко розширюваною, щоб забезпечити його успішну роботу в довгостроковій перспективі.

У рамках цієї роботи було розроблено архітектуру програми для оптичного розпізнавання символів на базі Tesseract OCR. Архітектура програми включає кілька компонентів, які взаємодіють між собою для забезпечення коректної роботи програми.

Першим компонентом архітектури є Docker, який дозволяє упаковувати і запускати додаток в контейнері, що забезпечує його масштабованість і зручність розгортання. Docker дозволяє створювати контейнери, які містять усі необхідні компоненти програми та залежності, що дозволяє усунути проблеми сумісності та скоротити час налаштування оточення.

Другим компонентом архітектури є Docker Compose, який забезпечує управління контейнерами і запуск їх у потрібному порядку. Docker Compose дозволяє описувати конфігурацію програми у файлі docker-compose.yml, який включає інформацію про контейнери, їх залежності і налаштування. Це забезпечує зручність розгортання та оновлення програми, а також спрощує його масштабування.

Третім компонентом архітектури є Nginx, який є вебсервером і забезпечує маршрутизацію запитів. Nginx дозволяє налаштовувати проксіювання запитів між контейнерами програми, що дозволяє забезпечити його високу доступність і стійкість до збоїв.

Четвертим компонентом архітектури є Laravel, який є фреймворком для розробки вебдодатків на мові PHP. Laravel забезпечує структурування та організацію програми, що спрощує її розробку та підтримку.

3.1.1 Опис архітектури Docker та Docker Compose

Docker – це програмне забезпечення, що дозволяє створювати та керувати контейнерами для різних програм. Контейнери представляють ізольоване середовище, в якому програми та їх залежності можуть працювати незалежно від навколишнього операційної системи та інших факторів.

Контейнеризація додатків є одним із найбільш ефективних способів керування продуктивністю, безпекою та розгортанням додатків. Контейнери Docker дозволяють створювати легковажні та портативні програми, які можна запускати практично в будь-якому оточенні.

Docker використовує концепцію контейнеризації, яка ґрунтується на принципах ізоляції ресурсів та спільного використання ядра операційної системи хоста. Контейнери створюються на основі образів, які можуть бути створені або завантажені із загальнодоступних репозиторіїв.

Контейнери надають ізольований простір, у якому програми можуть працювати без впливу інші програми, запущені на хості. Вони можуть бути запущені на будь-якому хості, що підтримує Docker, і можуть бути упаковані та доставлені з додатками та їх залежностями в єдиному середовищі, що робить їх переносними та легко керованими.

Docker забезпечує керування контейнерами через Docker API, який забезпечує доступ до контейнерів та їх ресурсів. Контейнери можна керувати через Docker CLI або через Docker GUI.

Однією з ключових переваг Docker є його швидкість та ефективність. Docker контейнери легковагні і можуть бути створені та запущені швидко. Вони споживають менше ресурсів, ніж віртуальні машини, що дозволяє покращити продуктивність та знизити витрати на обладнання.

Іншою важливою перевагою Docker є його портативність. Контейнери Docker можуть бути розгорнуті на будь-якій платформі, яка підтримує Docker, без необхідності перекомпіляції програми. Це дозволяє розробникам швидко і легко доставляти програми на різні сервери, адаптувати їх до різних

операційних систем та забезпечувати їхню надійну роботу на різних платформах.

Ще одна перевага Docker полягає в тому, що він забезпечує високий рівень безпеки. Контейнери створюються з використанням ізольованих ресурсів, що дозволяє запобігти можливим конфліктам між додатками та зменшити ймовірність впровадження шкідливого коду. Docker також надає інструменти для моніторингу та керування безпекою контейнерів, що підвищує рівень безпеки програм.

Крім того, Docker забезпечує зручне середовище для розробки та тестування програм. Розробники можуть створювати контейнери, в яких запускатимуться тести, та переконатися в коректності роботи додатків в ізольованому середовищі. Це спрощує процес розробки та зменшує кількість помилок, пов’язаних із оточенням.

В цілому, Docker є потужним інструментом для розробки, розгортання та управління додатками, який дозволяє створювати легковагові та портативні контейнери, покращити продуктивність та знизити витрати на обладнання, забезпечити безпеку та надійність додатків, а також спростити процес розробки та тестування.

Dockerfile для NGINX – це файл, який використовується для створення образу Docker, який включає сервер NGINX, конфігураційний файл і робочу директорію. Dockerfile є текстовим файлом, який містить інструкції для збирання образу. У цьому випадку Dockerfile використовується для створення образу Docker для сервера NGINX (лістинг А.1).

Використовуючи інструкцію «FROM», Dockerfile починає з базового образу Docker, який є основою створення нового образу. В даному випадку базовим чином є NGINX. Інструкція «ADD» використовується для додавання конфігураційного файлу у створюваний образ. У цьому випадку конфігураційний файл знаходиться в директорії «docker/conf/vhost.conf» і буде поміщений в директорію «/etc/nginx/conf.d/» у створюваному образі.

Інструкція «WORKDIR» використовується для встановлення робочої директорії у створюваному образі. В даному випадку робоча директорія встановлюється на «/var/www/laravel». Це дозволяє встановити кореневу директорію для вебсайту, який буде розміщений на сервері NGINX.

Використовуючи цей Dockerfile, Docker може створювати образ для сервера NGINX, який включає настроєний конфігураційний файл і встановлену робочу директорію. Цей образ може бути використаний для запуску контейнерів Docker, що містять сервер NGINX, із встановленим конфігураційним файлом та настроєною робочою директорією.

Образ, створений за допомогою цього Dockerfile, може бути доповнений іншими інструкціями Dockerfile, щоб увімкнути додаткові компоненти та налаштування. Наприклад, образ може бути доповнений інструкціями для встановлення додаткових модулів NGINX або параметрів безпеки.

Docker файл для PHP використовується для створення середовища PHP на основі образу php:8.1-fpm-buster (лістинг А.2, А.3).

Першим кроком є встановлення базового зображення на php:8.1-fpm-buster. Цей образ містить PHP 8.1 і базується на операційній системі Debian Buster.

Наступним кроком є оновлення системи та встановлення необхідних залежностей. Це робиться за допомогою команди apt-get, яка встановлює такі пункти,як:

- zip;
- libfreetype6-dev;
- libjpeg62-turbo-dev;
- libpng-dev;
- libwebp-dev;
- libxpm-dev;
- libzip-dev;
- libicu-dev;
- libpq-dev;

- git;
- libtesseract-dev;
- tesseract-ocr;
- tesseract-ocr-rus;
- tesseract-ocr-ukr;
- pandoc;
- libmagickwand-dev;
- libopencv-dev;
- libopencv-core-dev;
- libopencv-imgproc-dev;
- libopencv-highgui-dev;
- libopencv-features2d-dev;
- libopencv-flann-dev;
- libopencv-calib3d-dev;
- libopencv-objdetect-dev.

Наступним кроком буде встановлення розширення zip для PHP за допомогою команди docker-php-ext-install.

Наступний крок налаштовує та встановлює розширення GD для PHP за допомогою команд docker-php-ext-configure і docker-php-ext-install. Це розширення надає функції для обробки зображень.

Наступний крок встановлює розширення Imagick для PHP за допомогою команди pecl і вмикає його за допомогою команди docker-php-ext-enable. Це розширення надає інтерфейс PHP для бібліотеки ImageMagick, яка використовується для створення та обробки зображень.

Наступний крок встановлює Composer, менеджер залежностей для PHP, використовуючи команду curl та інтерпретатор php.

Наступним кроком буде встановлено робочий каталог /var/www/laravel, де можна розмістити код програми Laravel.

На наступному кроці встановлюється система керування процесом Supervisor за допомогою команди apt-get, створюється каталог «/var/log/supervisor» і додається файл конфігурації supervisord.conf до каталогу «/etc/supervisor/conf.d/».

Останні кроки відкривають порт контейнера 9000 і встановлюють команду за замовчуванням для контейнера на php-fpm, яка запускає PHP FastCGI Process Manager.

Таким чином, цей Dockerfile використовується для створення середовища PHP для запуску програм Laravel, яке містить багато корисних розширень та інструментів. Отримане зображення можна використовувати для розгортання програм Laravel у контейнерному середовищі.

Docker Compose – це інструмент для опису та керування багатоконтейнерними програмами, що працюють у середовищі Docker. Цей інструмент дозволяє визначити всі контейнери, їх залежності та конфігурацію в єдиному файлі, який легко читати та підтримувати.

Docker Compose дозволяє визначити всі контейнери, їх залежності та налаштування в одному конфігураційному файлі. Кожен контейнер визначається як окремий сервіс із зазначенням образу Docker, який використовуватиметься для створення контейнера. Кожен сервіс може мати свої установки, такі як порти, змінні оточення, монтування томів, залежності та ін.

Docker Compose дозволяє легко масштабувати та керувати програмами, запущеними в середовищі Docker. За допомогою цього інструменту можна запускати та зупиняти контейнери, масштабувати сервіси та керувати залежностями між контейнерами.

Крім того, Docker Compose дозволяє зручно працювати з мережами та томами, що використовуються у додатках. За допомогою цього інструменту можна створювати та налаштовувати Docker-мережі, а також монтувати томи у контейнери.

Для опису багатоконтейнерних програм у Docker Compose використовується файл конфігурації, який складається з сервісів та їх налаштувань. Файл конфігурації повинен містити інформацію про версію Compose, яка використовується для створення файлу, а також список сервісів та їх залежностей.

Для роботи з Docker Compose необхідно встановити його на комп'ютер і запустити команду docker-compose up для запуску програми. Ця команда автоматично створить та запустить усі контейнери, вказані у файлі конфігурації, та зв'яже їх між собою відповідно до залежностей.

В цілому, Docker Compose є потужним та зручним інструментом для управління багатоконтейнерними програмами в Docker. Він полегшує створення, запуск та керування контейнерами, а також спрощує розгортання додатків у продакшені.

Docker-compose файл визначає конфігурацію Docker-контейнерів для розгортання програми на PHP, яка працює за допомогою вебсервера Nginx (лістинг А.4). Файл містить опис двох сервісів: nginx та app.

Сервіс nginx описує параметри для запуску контейнера, який базується на образі, створеному з файлу Docker Nginx (лістинг А.1), що знаходиться в директорії docker. Цей контейнер прослуховує порт 80 на хості та перенаправляє запити на вебсервер програми. Крім того, nginx залежить від сервісу app і поділяє з ним директорію /var/www/laravel, що дозволяє обмінюватись даними між контейнерами.

Сервіс app також описує параметри запуску контейнера, який заснований на образі, створеному з файлу Docker Fpm (лістинг А.2, А.3), також розташованому в директорії docker. Контейнер app прослуховує порт 9000 і має доступ до директорії /var/www/laravel, де розташована програма. Також контейнер app монтує файл конфігурації php.ini з директорії docker/conf всередину контейнера, що дозволяє налаштувати PHP-програму.

Для забезпечення взаємодії між сервісами app та nginx використовується мережа app-network, що створюється за допомогою драйвера bridge. Мережа

дозволяє контейнерам обмінюватися даними та здійснювати зв’язок між сервісами.

Таким чином, ця конфігурація дозволяє розгорнути програму на PHP з використанням вебсервера Nginx всередині Docker-контейнерів. За допомогою docker-compose можна легко створювати та керувати кількома контейнерами, що спрощує процес розробки та розгортання програм.

3.1.2 Опис вебсерверу Nginx

Nginx – це високопродуктивний вебсервер та проксі-сервер, розроблений для забезпечення швидкої та надійної доставки вебвмісту. Він був створений Ігорем Сисоєвим у 2002 році і тепер є одним із найпопулярніших вебсерверів у світі. Nginx має безліч функцій і налаштувань, що робить його дуже гнучким і налаштованим для різних завдань, таких як статичний та динамічний вміст, балансування навантаження, кешування та багато іншого.

Nginx працює на операційних системах Linux, BSD, Solaris, MacOS, Windows та інших. Він використовує асинхронну модель обробки запитів, яка дозволяє ефективно обслуговувати велику кількість запитів із невеликою кількістю ресурсів. Крім того, Nginx підтримує безліч протоколів, таких як HTTP, HTTPS, SMTP, POP3 та IMAP, що робить його універсальним та готовим для використання у різних сценаріях.

Однією з особливостей Nginx є його модульна архітектура, яка дозволяє додавати нові функції та можливості без необхідності зміни ядра програми. Це робить його дуже гнучким і дозволяє настроювати його для різних завдань та сценаріїв.

В цілому, Nginx є потужним та гнучким вебсервером та проксі-сервером, який широко використовується у всьому світі. Він має високу продуктивність

і настроюваність, що дозволяє використовувати його в різних завданнях і сценаріях.

Конфігураційний файл містить вказівку на порт, у якому сервер слухає запити. Крім того, вказано ім’я сервера «localhost» та шлях до директорії з кореневим каталогом документів «/var/www/laravel/public» (лістинг А.5, А.6).

Для обробки запитів до директорії кореневого каталогу використовується директива «location/», яка перенаправляє запити на файл index.php або, якщо він відсутній, на головну сторінку. Параметри запиту зберігаються.

Для обробки запитів на виконання PHP-скриптів використовується директива «location ~ .php$», яка вказує, що запити, що закінчуються на .php, повинні бути передані на обробку FastCGI-серверу на порт 9000. шлях до виконуваного файлу скрипта, параметр SCRIPT_FILENAME, та параметр PATH_INFO, який містить інформацію про шлях до скрипту.

У конфігураційному файлі також вказані шляхи до файлів журналів помилок та доступу до сервера. Журнали призначені для відстеження та аналізу подій, що відбуваються на сервері.

В цілому, ця конфігурація дозволяє використовувати сервер Nginx для обслуговування вебпрограми на базі PHP, з можливістю перенаправлення запитів до скриптів на FastCGI-сервер.

3.1.3 Опис архітектури Laravel

Laravel – це безкоштовний та відкритий фреймворк для веброзробки, написаний мовою програмування PHP. Він використовує архітектурний шаблон проєктування MVC і має широкий спектр інструментів та функціональних можливостей, які спрощують процес розробки вебдодатків.

Laravel включає безліч функцій, таких як система маршрутизації, управління сеансами, автентифікація користувачів, міграції баз даних і багато

іншого. Крім того, Laravel має набір інструментів для управління залежностями та складання проєкту, що робить його ідеальним вибором для створення складних та масштабованих вебдодатків.

Laravel був випущений у 2011 році Тейлором Отвеллом і швидко став одним із найпопулярніших фреймворків для веброзробки мовою PHP. Він має активну спільноту розробників, які постійно працюють над покращенням фреймворку та створенням нових інструментів, що робить його ще більш привабливим для розробників вебдодатків.

Розділення коду у рамках framework Laravel (рис. 3.1).

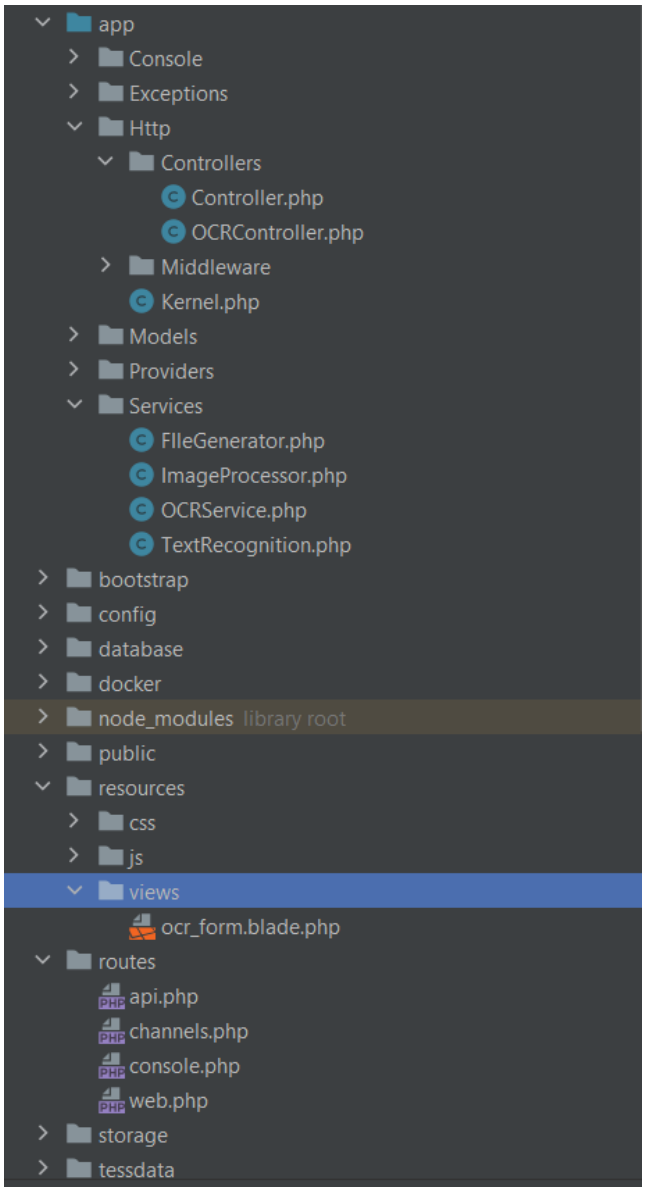

Рисунок 3.1 – Архітектура застосунку

Монолітний підхід до розробки програм у Laravel – це популярний підхід, який дозволяє розробляти програми в єдиному монолітному кодовій базі. Підхід передбачає поділ програми на шари та компоненти, де кожен шар чи компонент є окремою частиною функціональності програми.

У даному випадку використовується монолітний підхід для написання програми на Laravel. Додаток складається з одного Blade-шаблону, що містить форму, яка приймає фотографії з текстом, а також параметри для вибору мови та типу вихідного файлу.

Controller приймає параметри з форми і передає в Services, які обробляють запит і створюють вихідний файл. У цьому підході контролер не виконує бізнес-логіку, лише відповідає за прийняття запитів і передачу у відповідні сервіси.

Використовуються веброути які забезпечують відображення інформації та її обробку.

В цілому, монолітний підхід до розробки додатків на Laravel надає безліч переваг, таких як простота та зручність у розробці, легкість у тестуванні та підтримці, а також можливість швидкого масштабування програми у майбутньому.

3.2 Опис реалізації методів обробки зображень та оптичного розпізнавання символів

Вивчення методів обробки зображень для оптичного розпізнавання символів є актуальним завданням сучасної інформаційної технології. У цій роботі розглянуто основні методи обробки зображень, такі як фільтрація шуму, бінаризація, сегментація та розпізнавання ліній, а також їх застосування у системі оптичного розпізнавання символів.

Першим етапом для обробки зображення є отримання вхідних даних, які далі передаються по сервісам, роль приймача виконує Controller (лістинг А.7).

OCRController є класом, який реалізує методи обробки зображень для оптичного розпізнавання символів. Він успадковується від класу Controller і містить метод processImage, який приймає запит від клієнта і викликає відповідний метод OCRService (лістинг А.8) для обробки зображення.

Метод processImage отримує із запиту валідовані дані – зображення, тип файлу та мову розпізнавання. Потім створюється екземпляр класу OCRService, що містить реалізацію методів обробки зображень. Метод processImage викликає метод processImage з класу OCRService з передачею йому зображення, типу файлу та мови розпізнавання.

Після обробки зображення метод processImage повертає результат як шлях до файлу з розпізнаним текстом. Цей шлях передається методу sendOutputFile, який надсилає файл із розпізнаним текстом клієнту у вигляді відповіді на запит.

Метод sendOutputFile формує заголовки відповіді, що містять інформацію про тип контенту, ім’я файлу та розмір файлу. Потім метод download з бібліотеки Laravel викликається для надсилання файлу клієнту у форматі octet-stream.

OCRService, реалізую паттерн Фасад, він є одним із патернів проєктування, що використовуються для спрощення інтерфейсу складної системи. Він дозволяє приховати складність та деталі внутрішньої реалізації системи, надаючи простіший інтерфейс для використання системи.

У контексті реалізації OCR-сервісу патерн Фасад використовується для приховання деталей роботи з бібліотеками розпізнавання тексту, обробки зображень та генерації файлів, які використовуються всередині сервісу. Таким чином, замість того щоб безпосередньо використовувати всі компоненти, необхідні для виконання операції розпізнавання тексту на зображенні та генерації файлу в потрібному форматі, користувач взаємодіє тільки з простим інтерфейсом OCRService.

Конкретно, у цій реалізації OCRService надає метод processImage, який приймає параметри зображення, типу файлу та мови. Всередині методу

processImage використовуються компоненти TextRecognition, ImageProcessor та FileGenerator, які виконують розпізнавання тексту на зображенні, обробку зображення та генерацію файлу відповідно. Однак, користувачеві не потрібно знати про те, як ці компоненти працюють усередині сервісу, оскільки всі деталі реалізації приховані за простим інтерфейсом OCRService.

Таким чином, патерн Фасад у цій реалізації допомагає спростити інтерфейс сервісу та приховати складність його внутрішньої реалізації.

Передобробка зображення є частиною системи OCR, який приймає на вхід зображення та обробляє його за допомогою низки методів обробки зображень (лістинг А.9, А.10). Клас ImageProcessor відповідає за обробку зображення і містить такі методи: processImageFile, removeNoise, increaseContrast, і increaseSharpness.

Метод process викликається для кожного вхідного зображення та використовує методи обробки зображення у заданому порядку. Спочатку відбувається обробка зображення за допомогою методу processImageFile, який зчитує вхідне зображення, перетворює його формат у JPEG, збільшує контрастність та зберігає його у тимчасовий файл. Конвертація зображення у формат JPEG та встановлення максимальної якості стиснення необхідні для зменшення розміру файлу та прискорення його обробки.

Далі, метод removeNoise використовується для видалення шумів із зображення, використовуючи медіанний фільтр та гаусове розмиття, а також нормалізацію. Видалення шуму на зображенні дозволяє прибрати випадкові пікселі та покращити якість зображення, що може бути особливо корисним у випадках, коли якість вихідного зображення низька або коли воно містить артефакти, які можуть перешкодити розпізнаванню тексту.

Потім метод increaseContrast застосовує контрастне розтягування зображення, що покращує різницю між світлими і темними областями на зображенні.

Нарешті, метод increaseSharpness застосовує збільшення різкості за допомогою фільтра нерізкого маскування, що дозволяє покращити чіткість та деталізацію зображення.

Контрастування зображення та збільшення різкості допомагають покращити якість зображення та зробити текст на ньому більш читаним.

Методи обробки зображення, що використовуються в даному сервісі, є набором дій, що виконуються з використанням бібліотеки Imagick, які дозволяють покращити якість зображення і зробити його більш придатним для розпізнавання тексту. Кожен метод виконує певну операцію обробки зображення, яка покращує його якість. Крім того, кожен метод повертає шлях до тимчасового файлу, який містить оброблене зображення.

У результаті сервіс ImageProcessor використовує комбінацію методів обробки зображень для поліпшення якості вхідного зображення та готує його для подальшого розпізнавання тексту.

Головним сервісом, який виконує перетворення фото у текст є сервіс TextRecognition, цей сервіс приймає на вхід зображення, що обробляється ImageProcessor, та мовну модель, яка використовується для розпізнавання тексту (лістинг А.11, А.12). Внаслідок роботи сервісу виходить розпізнаний текст.

Для роботи TextRecognition необхідна мовна модель TesseractOCR, яка зберігається в каталозі tessdata. У даному сервісі використовується модель для англійської та української мов.

Для перетворення зображення на текст, Tesseract OCR використовує методи комп’ютерного зору та машинного навчання. Спочатку зображення перетворюється на чорно-біле, а потім перебувають контури символів, використовуючи методи сегментації зображення. Контур кожного символу порівнюється із шаблоном, збереженим у датасеті. Якщо збіг знайдено, символ розпізнається.

Підключення датасетів, таких як «eng.traineddata» та «ukr.traineddata», дозволяє Tesseract OCR розпізнавати символи різними мовами. Датасет

містить інформацію про форми символів, ідентифікатори та інші характеристики для кожної мови.

Таким чином, Tesseract OCR обробляє зображення, знаходить символи, розпізнає їх та перетворює їх у текст, використовуючи методи комп’ютерного зору та машинного навчання. Підключення датасетів дозволяє Tesseract OCR розпізнавати символи різними мовами, що робить його дуже корисним для багатьох програм, пов’язаних з обробкою тексту.

Застосування методів методів обробки зображень було б не можливе без роутінгу та візуальнії частини.

Застосовується стандартний роутінг для Laravel, який допомагає визначати, який код повонен виконатися для обробки запиту, що надходить від клієнтів (лістинг А.13).

Коли програма Laravel отримує запит, вона використовує певний маршрут, щоб визначити, який контролер і який метод повинні опрацювати цей запит.

Візуальна частина виконана за допомогою blades від Laravel (лістинг А.14, А.15).

Використовується Blade для створення HTML-форми для завантаження зображення та вибору мови та типу файлу для подальшої обробки за допомогою OCR. У цьому прикладі Blade дозволяє вставляти змінні всередину HTML-коду, використовуючи синтаксис фігурних дужок разом із символом @. Разом з Bootstrap CSS і JavaScript, що використовуються у шаблоні, Blade спрощує процес створення чуйних та привабливих інтерфейсів для вебдодатків.

3.3 Тестування програми, включаючи тестування якості розпізнавання та продуктивності

Мануальне тестування – це процес перевірки програмного продукту на відповідність заданим вимогам та специфікаціям шляхом виконання певних тестових сценаріїв вручну. Цей процес проводиться з метою виявлення помилок, дефектів та недоліків, які можуть виникнути під час роботи програми, а також для перевірки її функціональності та відповідності вимогам замовника.

Мануальне тестування передбачає використання спеціальних тестових випадків, які включають широкий діапазон вхідних даних, а також здатність перевіряти різні аспекти програми, включаючи інтерфейс користувача, правильність роботи алгоритмів і видачу результатів. Вона може бути проведена як професійними тестувальниками, і звичайними користувачами продукту.

Основна мета мануального тестування – забезпечити високу якість програмного продукту та задоволення потреб користувачів. Для досягнення цієї мети тестувальники повинні мати гарне розуміння вимог до продукту, глибокі знання у галузі тестування та досвід у використанні різних методик та інструментів тестування.

Мануальне тестування є невід’ємною частиною процесу розробки програмного забезпечення та може бути використане у поєднанні з автоматизованим тестуванням для забезпечення максимальної ефективності та якості продукту.

Тестування проводилося на застосунку, написаному на фреймворку Laravel, з використанням Tesseract OCR для розпізнавання тексту на зображеннях (рис. 3.2).

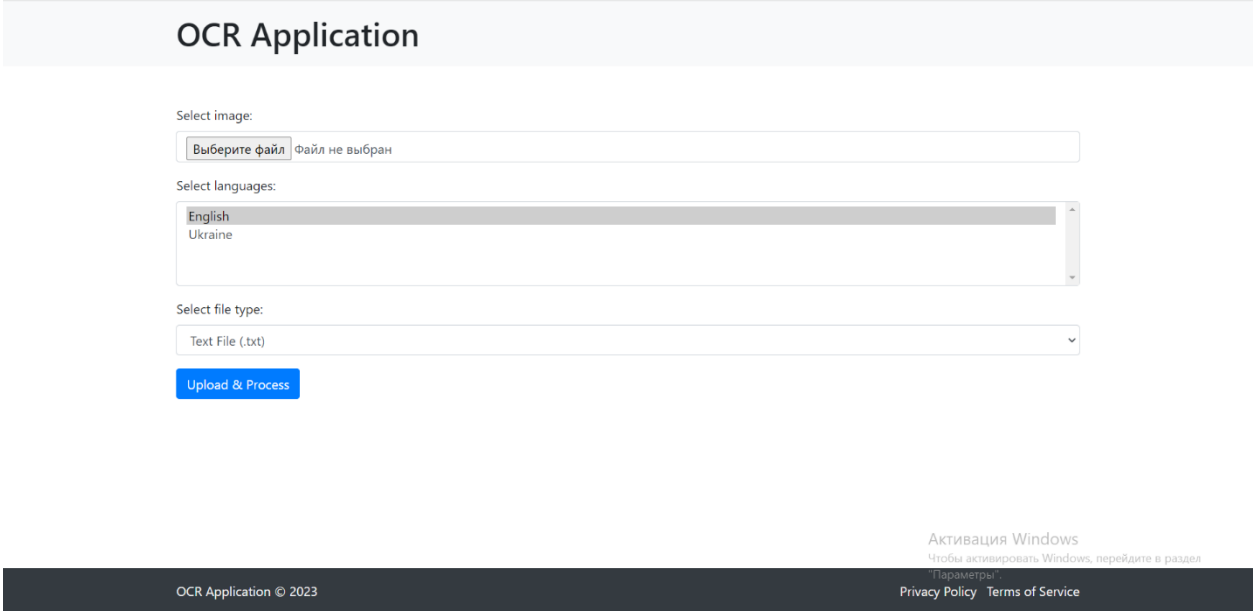

Рисунок 3.2 – Основний функціонал програми

Першим кроком є перевірка завантаження файлу (рис. 3.3).

Рисунок 3.3 – Поле для завантаження файлу

Для цього потрібно натиснути кнопку «Завантажити файл», потім вибрати файл для завантаження та натиснути кнопку «Відкрити» (рис. 3.4).

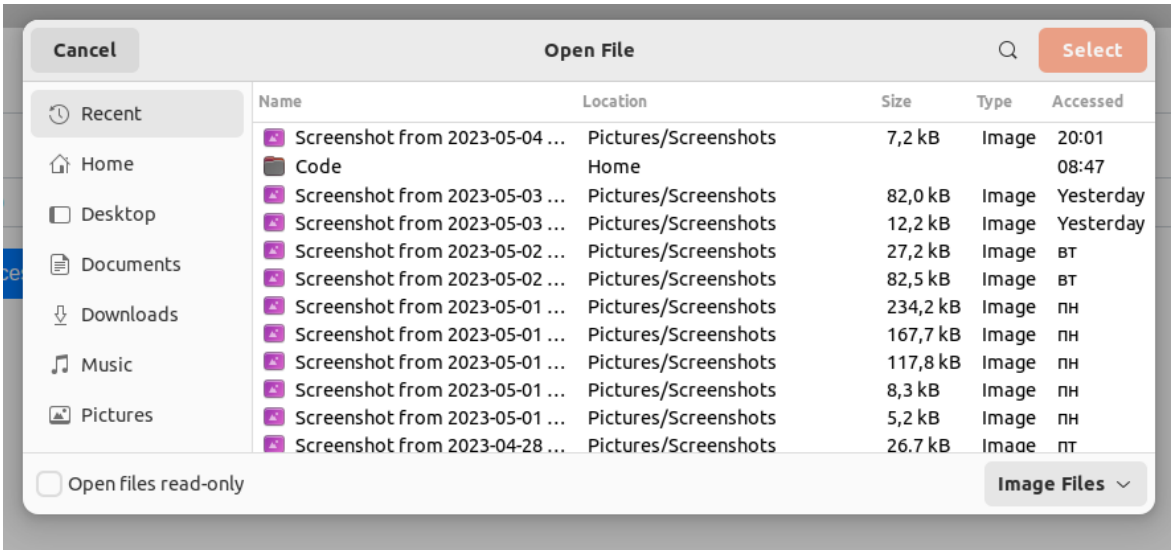

Рисунок 3.4 – Форма вибору файлу для завантаження

Після вибору файлу, він успішно завантажений та відображається у списку завантажених файлів (рис. 3.5).

Рисунок 3.5 – Обраний файл

Наступним кроком тестування є вибір мови розпізнавання із запропонованих (рис. 3.6).

Рисунок 3.6 – Вибір мови

Після вибору мови, вона успішно зберігається та відодображається. Далі йде тестування вибору типу вихідного файлу (рис. 3.7).

Рисунок 3.7 – Вибір типу вихідного файлу

Результатом корректної роботи є вібір та відобраення обранного типу (рис. 3.8).

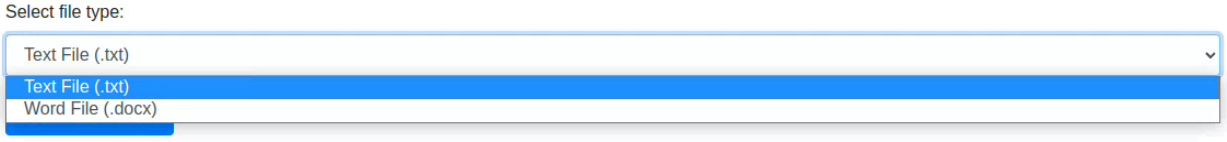

Рисунок 3.8 – Можливі типи вихідних файлів

Для того щоб почати розпізнавання тексту з фотографії, потрібно натиснути кнопку «старту», у результаті видбулося завантаження файлу у локальну систему користуача (рис. 3.9).

Рисунок 3.9 – Оброблений файл у локальній системі

У вихідному файлі є оброблений текст з фотографії. Це свідчить про успішне тестування основного функціоналу (рис. 3.10).

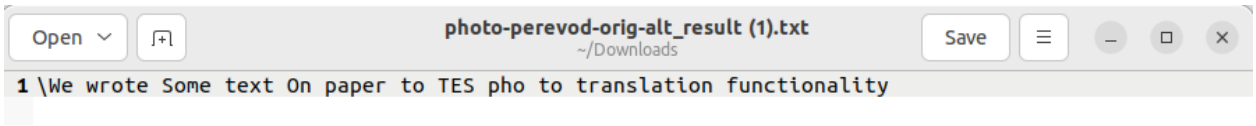

Рисунок 3.10 – Оброблений текст

Дані, отримані в результаті мануального тестування систематизовані у таблиці (табл. 3.1). Таблиця містить інформацію про всі процедури тестування, отримані результати та інші характеристики, які є важливими для оцінки якості продукту.

Таблиця є важливим джерелом даних для подальшої аналітики та звітності, які необхідні для оцінки ефективності процесу тестування та якості продукту в цілому. Аналіз отриманих даних дозволяє приймати обґрунтовані рішення про додаткові тести та коригування продукту.

Документація результатів мануального тестування у таблиці є обов’язковим етапом процесу тестування і є необхідною основою подальшої роботи над поліпшенням якості продукту.

Таблиця 3.1 – Результати тестування основного інтерфейсу

Опис тест-кейсу	Передумови	Очікуваний результат	Фактичний результат	Оцінка
завантаження файлу	додаток запущено	файл успішно завантажено та відображається у списку завантажених файлів	файл успішно завантажено та відображається у списку завантажених файлів	пройдено
вибір мови	додаток запущено, файл завантажений	мова розпізнавання успішно змінена на вибрану	мова розпізнавання успішно змінена на вибрану	пройдено
вибір типу вихідного файлу	додаток запущено, файл завантажений	тип вихідного файлу успішно змінено на вибраний	тип вихідного файлу успішно змінено на вибраний	пройдено
розпочати розпізнавання файлу	додаток запущено, файл завантажений, мову та тип вихідного файлу вибрано	процес розпізнавання успішно запущено	процес розпізнавання успішно запущено	пройдено
перегляд результатів	додаток запущено, файл завантажений, мову та тип вихідного файлу вибрано, розпізнавання завершено	результат розпізнавання відображається у завантаженнях	результат розпізнавання відображається у завантаженнях	пройдено

Точність розпізнавання тексту на зображеннях може сильно змінюватись в залежності від різних факторів, таких як якість зображення, розмір і шрифт

тексту, наявність шуму і т.д. Тому розробка ефективних алгоритмів для оцінки якості обробки тексту з зображень є важливим завданням у сфері комп'ютерного зору.

У цьому тестуванні використана формула для обчислення якості обробки тексту з зображень, що базується на стандартному підході для оцінки точності розпізнавання тексту.

Розрахунок якості, точності та продуктивності програми, відбувається за формулою:

$$R = (n \div m) \times 100\%, \quad (3.1)$$

де R – результат точності;

n – кількість правильно розпізнаних символів;

m – загальна кількість символів.

В рамках цього дослідження проведено тестування якості розпізнавання тексту з використанням зображень. Тестування проведене на 8-ми зображеннях.

Перше зображення загально містить 57 символів, які були розпізнані вручну (рис. 3.11).

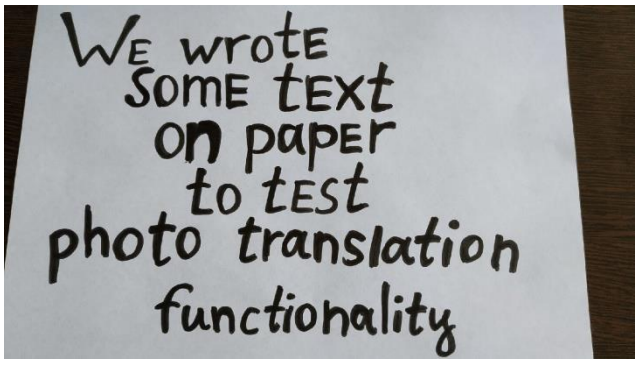

Рисунок 3.11 – Приклад фото з текстом

Програма розпізнала 56 символів на зображенні (рис. 3.12). Точність розпізнавання становить 98%.

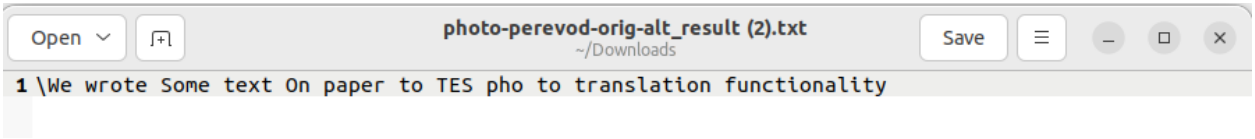

Рисунок 3.12 – Оброблений текст з фото

Друге зображення загально містить 14 символів, які були розпізнані вручну (рис. 3.13).

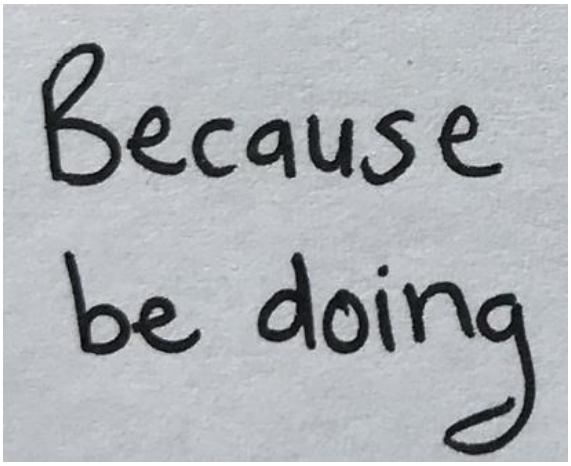

Рисунок 3.13 – Приклад фото з текстом

Програма розпізнала 8 символів на зображенні (рис. 3.14). Точність розпізнавання становить 57%.

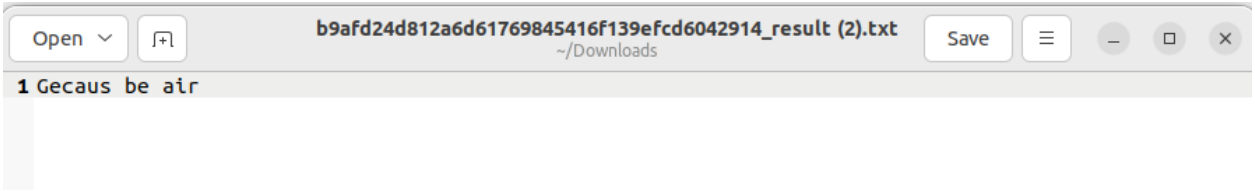

Рисунок 3.14 – Оброблений текст з фото

Трете зображення загально містить 61 символів, які були розпізнані вручну (рис. 3.15).

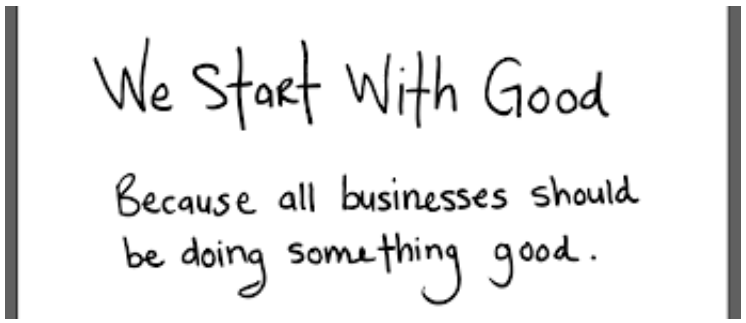

Рисунок 3.15 – Приклад фото з текстом

Програма розпізнала 55 символів на зображенні (рис. 3.16). Точність розпізнавання становить 90%.

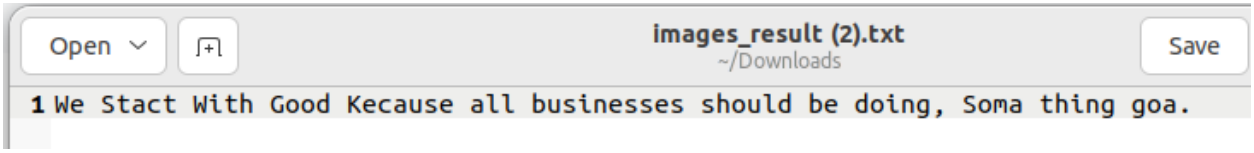

Рисунок 3.16 – Оброблений текст з фото

Четверте зображення загально містить 44 символа, які були розпізнані вручну (рис. 3.17).

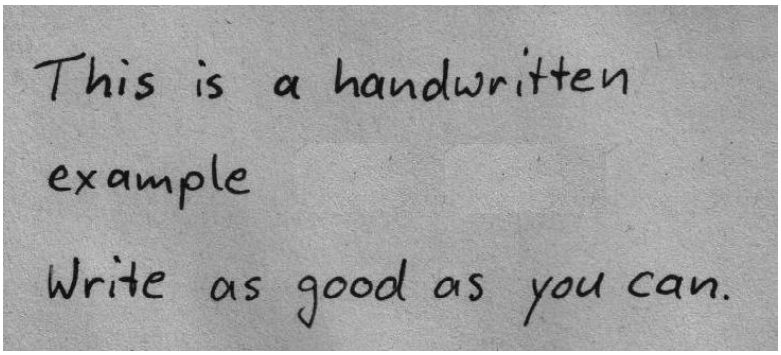

Рисунок 3.17 – Приклад фото з текстом

Програма розпізнала 44 символа на зображенні (рис. 3.18). Точність розпізнавання становить 100%.

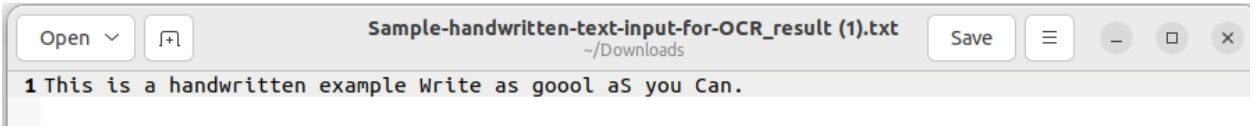

Рисунок 3.18 – Оброблений текст з фото

П’яте зображення загально містить 60 символів, які були розпізнані вручну (рис. 3.19).

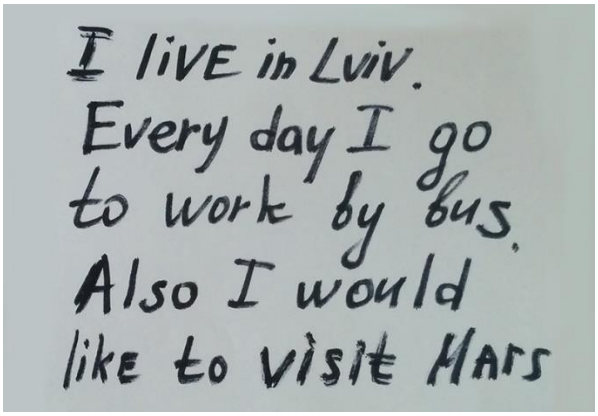

Рисунок 3.19 – Приклад фото з текстом

Програма розпізнала 48 символів на зображенні (рис. 3.20). Точність розпізнавання становить 80%.

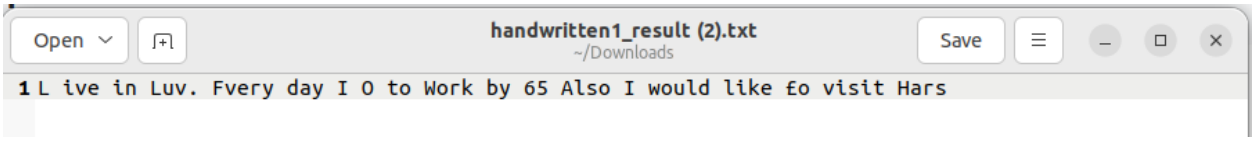

Рисунок 3.20 – Оброблений текст з фото

Шосте зображення загально містить 272 символів, які були розпізнані вручну (рис. 3.21).

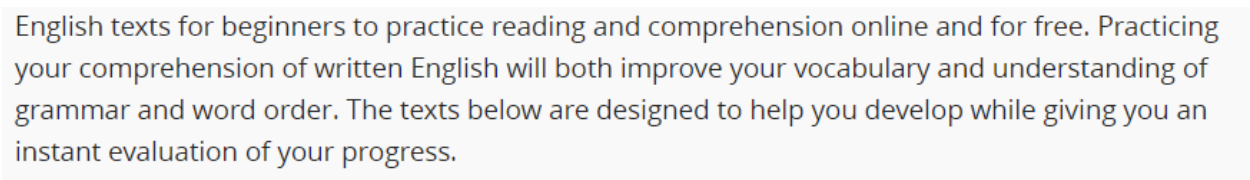

Рисунок 3.21 – Приклад фото з текстом

Програма розпізнала 272 символів на зображенні (рис. 3.22). Точність розпізнавання становить 100%.

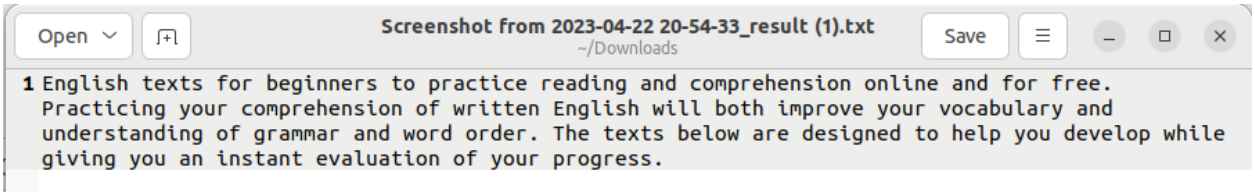

Рисунок 3.22 – Оброблений текст з фото

Сьоме зображення загально містить 649 символів, які були розпізнані вручну (рис. 3.23).

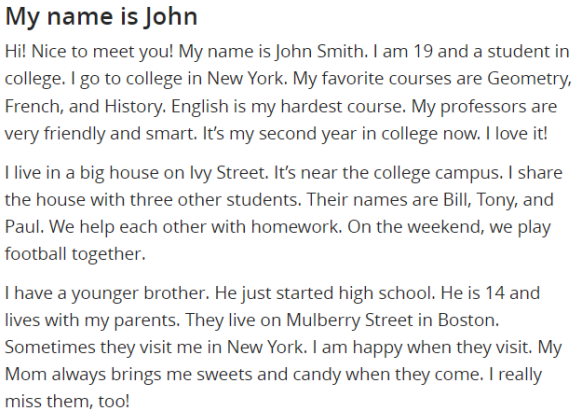

Рисунок 3.23 – Приклад фото з текстом

Програма розпізнала 649 символів на зображенні (рис. 3.24). Точність розпізнавання становить 100%.

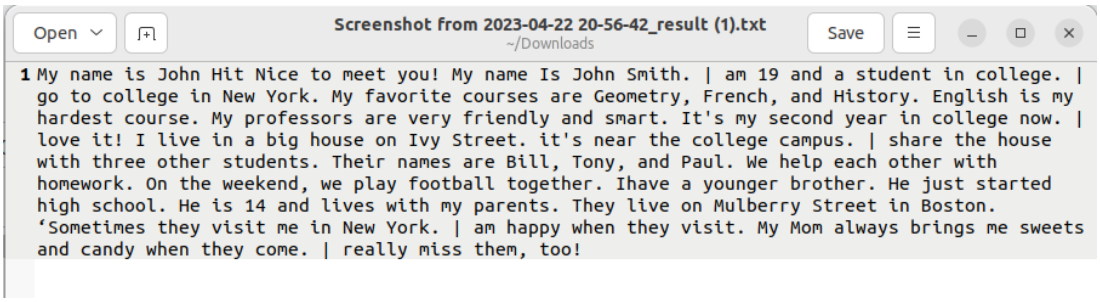

Рисунок 3.24 – Оброблений текст з фото

Восьме зображення загально містить 36 символів, які були розпізнані вручну (рис. 3.25).

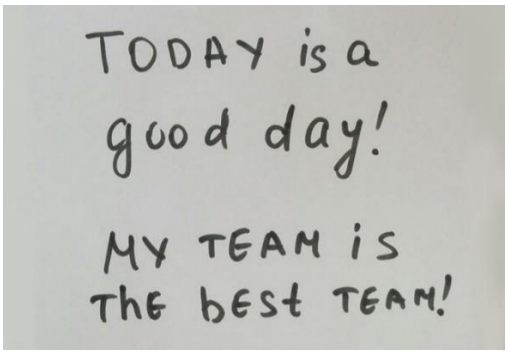

Рисунок 3.25 – Приклад фото з текстом

Програма розпізнала 29 символів на зображенні (рис. 3.26). Точність розпізнавання становить 80%.

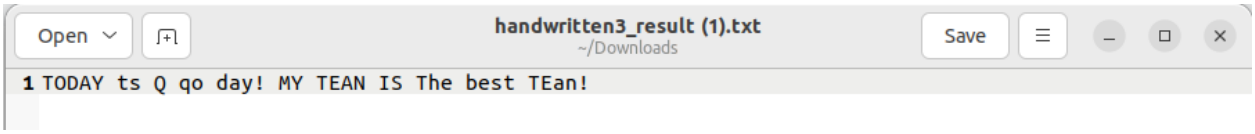

Рисунок 3.26 – Оброблений текст з фото

Отримані результати систематизовані та занесені до таблиці (табл. 3.2). Таблиця містить інформацію про використовувані зображення, а також результати процедури розпізнавання тексту.

У таблиці представлені дані, необхідні для аналізу ефективності процесу розпізнавання тексту з зображень. Кожен рядок таблиці відповідає одному тесту, а кожен стовпець таблиці відображає певний аспект тестування, такий як точність розпізнавання та кількість символів.

Ця таблиця є необхідною основою для проведення подальшого аналізу отриманих результатів та визначення причин можливих помилок. При цьому результати тестування в таблиці будуть використовуватися для оцінки ефективності процесу розпізнавання тексту та прийняття обґрунтованих рішень щодо покращення даної функціональності продукту.

Таблиця 3.2 – Результати тестування якості розпізнавання символів

Номер рисунку	Кількість символів	Кількість розпізнанних символів	Точність
Рисунок 3.11	57	56	98%
Рисунок 3.13	14	8	57%
Рисунок 3.15	61	55	90%
Рисунок 3.17	44	44	100%
Рисунок 3.19	60	48	80%
Рисунок 3.21	272	272	100%
Рисунок 3.23	649	649	100%
Рисунок 3.25	36	29	80%

Після використання формули для оцінки точності розпізнавання символів, отриманий результат точності визначений у відсотках і є відношенням числа правильно розпізнаних символів до загального числа символів у тексті.

Тестування показало, що точність розпізнавання тексту з фотографії залежить від різних факторів, таких як шрифт, розмір, орієнтація та мова. У середньому точність розпізнавання становить близько 86%.

Такий підхід до оцінки точності розпізнавання тексту має широке застосування в області комп’ютерного зору і використовується в різних завданнях, пов’язаних з обробкою зображень.

Результат точності дозволяє оцінити якість роботи алгоритмів обробки зображень та розпізнавання тексту та визначити, наскільки точно система справляється з поставленим завданням. Однак, слід зазначити, що в деяких випадках даний підхід до оцінки точності може бути недостатньо інформативним, оскільки деякі помилки розпізнавання можуть бути критичними та впливати на загальну оцінку якості роботи системи.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований застосунок для оптичного розпізнавання символів із фотографій. Ця програма успішно здійснює перетворення зображення в текст, при цьому точність розпізнавання залежить від читальності символів на фотографії та датасетів. Машинописний текст розпізнається з високою точністю, тоді як рукописний текст можна розпізнати з деякою похибкою.

У ході роботи були використані Laravel і Tesseract OCR, а також були дописані методи зменшення шуму, збільшення контрастності і підвищення різкості зображення. Ці методи дозволили покращити якість зображення та підвищити точність розпізнавання.

Важливим фактором, що впливає на точність розпізнавання, є якість фотографії та розмір зображення. Чим краща якість фотографії і чим більший її розмір, тим вища точність розпізнавання символів.

Таким чином, розроблений застосунок є ефективним інструментом для оптичного розпізнавання символів. Його використання може спростити та прискорити роботу у різних сферах, пов’язаних з обробкою текстової інформації. Однак, для досягнення максимальної точності розпізнавання символів необхідно забезпечити високу якість фотографій, оптимальні датасети за необхідною тематикою а також оптимальні умови освітлення та контрастності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Awel, M. A., & Abidi, A. I. (2019). Review on optical character recognition. International Research Journal of Engineering and Technology (IRJET), 6(6), 3666-3669.

2. Memon, J., Sami, M., Khan, R. A., & Uddin, M. (2020). Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR). IEEE Access, 8, 142642-142668.

3. Iryna, V., & Oleksii, T. (2021). RESEARCH AND IMPLEMENTATION OF THE VIDEO STREAM TEXT RECOGNITION METHOD. EDITORIAL BOARD, 412.

4. Adjetey, C., & Adu-Manu, K. S. (2021). Content-based image retrieval using Tesseract OCR engine and levenshtein algorithm. International Journal of Advanced Computer Science and Applications, 12(7).

5. Breuel, T. M., Ul-Hasan, A., Al-Azawi, M. A., & Shafait, F. (2013, August). High-performance OCR for printed English and Fraktur using LSTM networks. In 2013 12th international conference on document analysis and recognition (pp. 683-687). IEEE.

6. Bagirzade, A. R., Najafova, A. S., Yessirkepova, S. M., & Albert, E. S. (2021). OCR/ICR Text Recognition Using ABBYY FineReader as an Example Text. International Journal of Computer and Information Engineering, 15(2), 126-130.

7. Dhiman, S., & Singh, A. (2013). Tesseract vs gocr a comparative study. International Journal of Recent Technology and Engineering, 2(4), 80.

8. Hegghammer, T. (2022). OCR with Tesseract, Amazon Textract, and Google Document AI: a benchmarking experiment. Journal of Computational Social Science, 5(1), 861-882.

9. Кобилін, О. А., & Творошенко, І. С. (2021). Методи цифрової обробки зображень.

10. Творошенко, І. С. (2018). Основи цифрової обробки зображень: конспект лекцій для студентів 4 курсу денної форми навчання напряму 6.080101–Геодезія, картографія та землеустрій.

11. Путятін, Є. П., Гороховатський, В. О., & Матат, О. О. (2006). Методи та алгоритми комп'ютерного зору: навч. посіб. Харків: ТОВ «Компанія СМІТ.

12. Руденко, Д. О., & Кухарчук, В. А. (2022, October). СУЧАСНІ НАПРЯМКИ ЗАСТОСУВАННЯ МЕТОДІВ СУПЕР РОЗДІЛЬНОСТІ ЗОБРАЖЕННЯ. In The 8 th International scientific and practical conference "Modern research in world science"(October 29-31, 2022) SPC "Sci-conf. com. ua", Lviv, Ukraine. 2022. 1828 p. (p. 393).

13. Daradkeh, Y.I., Tvoroshenko, I., Gorokhovatskyi, V., Latiff, L.A., and Ahmad, N. (2021) Development of Effective Methods for Structural Image Recognition Using the Principles of Data Granulation and Apparatus of Fuzzy Logic, *IEEE Access*, 9, pp. 13417-13428.

14. Sheeba, T. M., & Raj, S. A. A. (2022). Analysis of Noise Removal Techniques on Retinal Optical Coherence Tomography Images. International Journal of Advanced Computer Science and Applications, 13(9).

15. Agrawal, S., Panda, R., Mishro, P. K., & Abraham, A. (2022). A novel joint histogram equalization based image contrast enhancement. Journal of King Saud University-Computer and Information Sciences, 34(4), 1172-1182.

16. Fu, X., Wang, W., Huang, Y., Ding, X., & Paisley, J. (2020). Deep multiscale detail networks for multiband spectral image sharpening. IEEE Transactions on Neural Networks and Learning Systems, 32(5), 2090-2104.

17. Zhao, J., Shi, C., Jia, F., Wang, Y., & Xiao, B. (2019). Document image binarization with cascaded generators of conditional generative adversarial networks. Pattern Recognition, 96, 106968.

18. Rabotiahov, A., Kobylin, O., Dudar, Z., & Lyashenko, V. (2018, February). Bionic image segmentation of cytology samples method. In 2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET) (pp. 665-670). IEEE.

19. Kobylin, O., & Lysenko, A. (2021). Research of superpixel image segmentation method.

20. Liu, X., Song, L., Liu, S., & Zhang, Y. (2021). A review of deep-learning-based medical image segmentation methods. Sustainability, 13(3), 1224.

21. Deng, Q., & Söffker, D. (2021). A Review of the current HMM-based Approaches of Driving Behaviors Recognition and Prediction. IEEE Transactions on Intelligent Vehicles.

22. Yang, G., Lu, Z., Yang, J., & Wang, Y. (2019). An adaptive contourlet HMM–PCNN model of sparse representation for image denoising. IEEE Access, 7, 88243-88253.

23. Li, H., Wang, P., Shen, C., & Zhang, G. (2019, July). Show, attend and read: A simple and strong baseline for irregular text recognition. In Proceedings of the AAAI conference on artificial intelligence (Vol. 33, No. 01, pp. 8610-8617).

24. Kobylin O., Gorokhovatskyi V., Tvoroshenko I., and Peredrii O. (2020) The application of non-parametric statistics methods in image classifiers based on structural description components, *Telecommunications and Radio Engineering*, 79(10), pp. 855-863.

25. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., and Al-Dhaifallah, M. (2021) Methods of Classification of Images on the Basis of the Values of Statistical Distributions for the Composition of Structural Description Components, *IEEE Access*, 9, pp. 92964-92973.

26. Gorokhovatskyi, V.O., Tvoroshenko, I.S., and Peredrii O.O. (2020) Image classification method modification based on model of logic processing of bit description weights vector, *Telecommunications and Radio Engineering*, 79(1), pp. 59-69.

27. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Cluster representation of the structural description of images for effective classification, *Computers, Materials & Continua*, 73(3), pp. 6069-6084.

28. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5-1

29. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Al-Dhaifallah M. (2022) Classification of Images Based on a System of Hierarchical Features, *Computers, Materials & Continua*, 72(1), pp. 1785-1797.

30. Творошенко, І. С., & Зеленський, М. О. Дослідження гібридних методів для класифікації складноструктурованих зображень. Сучасний рух науки: тези доп. VIII міжнародної науково-практичної інтернет-конференції, 3-4 жовтня 2019 р.–Дніпро, 2019.–Т. 3.–724 с., 382.