Міністерство освіти і науки України
Університет митної справи та фінансів

Факультет інноваційних технологій
Кафедра комп'ютерних наук та інженерії програмного забезпечення

# Кваліфікаційна робота магістра

на тему *«Системи розпізнавання алфавітно-цифрової інформації методами інтелектуального аналізу даних»*

Виконав: студент групи К22-1м
Спеціальність 122 «Комп'ютерні науки»
Мірзабеков Ельдар Айвазович
(прізвище та ініціали)
Керівник к.т.н., доц. Мала Ю.А.
(науковий ступінь, вчене звання, прізвище та ініціали)
Рецензент _____
(місце роботи)
_____
_____
(посада)
_____
(науковий ступінь, вчене звання, прізвище та ініціали)

Дніпро – 2024

# АНОТАЦІЯ

*Мірзабеков Е.А.* Системи розпізнавання алфавітно-цифрової інформації методами інтелектуального аналізу даних.

Дипломна робота на здобуття освітнього ступеня магістр за спеціальністю 122 «Комп'ютерні науки». – Університет митної справи та фінансів, Дніпро, 2024.

Дипломна робота присвячена теоретичному та практичному аналізу існуючих систем розпізнавання алфавітно-цифрової інформації за допомогою методів інтелектуального аналізу даних. Робота розглядає актуальні проблеми та виклики, пов'язані з розпізнаванням текстової та числової інформації в сучасному інформаційному середовищі. Автор систематизує і аналізує різноманітні підходи та методи, що використовуються в існуючих системах розпізнавання. Дослідження спрямоване на визначення переваг та обмежень різних методів, таких як машинне навчання, нейронні мережі, статистичні методи тощо.

Результати дослідження можуть бути використані для покращення ефективності систем розпізнавання алфавітно-цифрової інформації та визначення перспектив розвитку цієї області в майбутньому.

Метою дослідження є розробка додатка для розпізнавання тексту на зображеннях, використовуючи сучасні методи інтелектуального аналізу даних, та оцінка його влучності та повноти.

Для досягнення мети дипломного проекту було створено програму, розроблену на мові Python з використанням Tesseract OCR та OpenCV.

ОПТИЧНЕ РОЗПІЗНАВАННЯ СИМВОЛІВ, АЛФАВІТНО-ЦИФРОВА ІНФОРМАЦІЯ, МЕТОДИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ, TESSERACT OCR, ОБРОБКА ЗОБРАЖЕНЬ

# ABSTRACT

*Mirzabekov E.A.* Systems of recognition of alphanumeric information by methods of intelligent data analysis.

Diploma thesis for the degree of Master's Degree in speciality 122 "Computer Science." - University of Customs and Finance, Dnipro, 2024.

The thesis is devoted to the theoretical and practical analysis of existing alphanumeric information recognition systems using data mining methods. The work considers current problems and challenges related to the recognition of text and numeric information in the modern information environment. The author systemises and analyzes various approaches and methods used in existing recognition systems. The study aims to identify the advantages and limitations of various methods, such as machine learning, neural networks, statistical methods, etc.

The results of the study can be used to improve the efficiency of alphanumeric information recognition systems and determine the prospects for the development of this area in the future.

The aim of the research is to develop and optimize an application for text recognition in images using modern methods of data mining and to evaluate its precision and recall.

To achieve the goal of the thesis project, we created an application developed in Python using Tesseract OCR and OpenCV.

TEXT RECOGNITION, ALPHANUMERIC INFORMATION, DATA MINING METHODS, TESSERACT OCR, IMAGE PROCESSING.

# ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

| | | |
|---|---|---|
| ІАД | – | інтелектуальний аналіз даних |
| OCR | – | оптичне розпізнавання символів |
| CNN | – | Convolutional neural network |
| RNN | – | Recurrent neural networks |
| BERT | – | Bidirectional Encoder Representations Transformers |
| GPT | – | Generative Pre-trained Transformer |
| LSTM | – | Long Short-Term Memory |
| GRU | – | Gated Recurrent Unit |
| NLTK | – | Natural Language Toolkit |
| TF-IDF | – | Term Frequency - Inverse Document Frequency |

# ВСТУП

Розпізнавання тексту, також відоме як оптичне розпізнавання символів (Optical Character Recognition, далі по тексту OCR), відіграє важливу роль в областях комп'ютерного зору та розпізнавання образів. Це охоплює процес перетворення зображень або рукописного тексту в редагований та можливий до пошуку текст. Основна мета розпізнавання тексту полягає в використанні автоматизованих технік, які можуть перетворити друкований або рукописний текст у формат, який комп'ютери можуть легко читати та розуміти.

Досягнення в машинному навчанні та штучному інтелекту (ШІ) значно внесли внесок у розвиток розпізнавання тексту. Методи глибокого навчання, такі як згорткові нейронні мережі (ЗНМ, англ. Convolutional neural network) і рекурентні нейронні мережі (РНМ, англ. Recurrent neural networks), зокрема, продемонстрували значні досягнення в цій галузі. Ці моделі можуть навчитися розпізнавати та транскрибувати складні шаблони та структури в текстових даних. Для розв'язання проблеми покращення точності розпізнавання тексту потрібно враховувати кілька факторів. По-перше, якість і різноманітність навчальних даних мають вирішальне значення. Моделі стає більш стійкою та адаптованою до різних сценаріїв завдяки різноманітному та добре анотованому набору даних, який допомагає їй вивчати різноманітні стилі тексту, шрифти та мову. Крім того, оптимізація великомасштабних процесів навчання та виведення може значно підвищити ефективність та продуктивність моделей розпізнавання тексту. [1]

Крім того, система розпізнавання текстів повинна підтримувати кілька мов і шрифтів, щоб вони могли працювати в різних мовних і культурних контекстах. У процесі розпізнавання різні мови та шрифти мають свою структуру та особливості. Додатково, враховуючи зміни розміщення та розташування тексту, такі як нахил або поворот, потрібні особливі методи для ефективного вилучення текстового матеріалу.

Розпізнавання рукописного тексту є ще однією важливою проблемою в розпізнаванні тексту. Різноманітність стилів почерку, індивідуальні стилі письма та відсутність стандартизованих шрифтів ускладнюють написання текстів. Розробка моделей, здатних ефективно розпізнавати та транскрибувати рукописний текст, вимагає використання алгоритмів і спеціалізованих методів навчання. Покращення інтерпретованості та зрозумілості моделей розпізнавання тексту також має вирішальне значення.

Користувачі повинні розуміти та бути впевнені в результатах, які отримують ці моделі. Такі методи, як механізми уваги й візуалізації, можуть дати уявлення про те, як модель робить прогнози, підвищуючи прозорість і довіру користувачів. Застосування методів штучного інтелекту і машинного навчання, зокрема глибокого навчання, зробило революцію в розпізнаванні тексту. Завдяки вирішенню проблем, пов'язаних з якістю та різноманітністю даних, оптимізацією великомасштабного навчання та висновків, підтримкою мов і шрифтів, варіаціями розташування тексту та макетів, а також розпізнаванням рукописного тексту, було досягнуто значного прогресу в точності та сфері застосування моделей розпізнавання тексту.

Тема дослідження спрямована на вирішення труднощів, пов'язаних із розпізнаванням тексту на зображеннях, за допомогою інтелектуального аналізу даних. Інтеграція методів машинного навчання та алгоритмів розпізнавання може суттєво покращити точність та швидкість таких систем. Дослідження також відзеркалює сучасні тенденції в галузі комп'ютерного зору, машинного навчання та обробки зображень.

Врахування новітніх досягнень у цих областях дозволяє розробити ефективні та інноваційні методи розпізнавання, що мають широкий спектр можливих застосувань.

У дипломній роботі було розглянуто фундаментальні аспекти розпізнавання алфавітно-цифрової інформації методами інтелектуального аналізу даних.

Мета дослідження: Розробка додатка для розпізнавання тексту на зображеннях, використовуючи сучасні методи інтелектуального аналізу даних, та оцінка його влучності та повноти.

Об'єкт дослідження: системи розпізнавання алфавітно-цифрової інформації.

Задачі, які необхідно вирішити для досягнення поставленої мети:

а) Аналіз сучасних підходів:

1) Вивчити та проаналізувати існуючі методи розпізнавання тексту на зображеннях;
2) Визначити переваги та недоліки існуючих підходів;
3) Сформулювати вимоги до нової системи розпізнавання.

б) Обґрунтування принципів та підходів:

1) Визначити принципи отримання алфавітно-цифрової інформації зображень;
2) Визначити використані методи інтелектуального аналізу даних у контексті розпізнавання тексту.

в) Розробка системи розпізнавання:

1) Вибрати підходящий алгоритм для розпізнавання текстової інформації;
2) Розробити програмне забезпечення для реалізації системи розпізнавання.

г) Оптимізація та вдосконалення:

1) Визначити можливі шляхи оптимізації роботи системи;
2) Розглянути можливості вдосконалення алгоритмів розпізнавання;
3) Оцінити швидкість та точність роботи системи та внести необхідні корективи.

Методи дослідження. Теорія інформації стала ключовим методом аналізу з погляду оптимізації алгоритмів та забезпечення ефективного оброблення алфавітно-цифрової інформації на зображеннях. Методи теорії

обробки сигналів використовувались для розгляду акустичної та візуальної інформації, сприяючи поліпшенню алгоритмів розпізнавання тексту на зображеннях. Теорія математичної морфології виявилася корисною при розв'язанні проблеми обробки та аналізу зображень. Також в роботі використовувалися методи об'єктно-орієнтованого програмування для створення додатка для розпізнавання тексту. Ці методи сприяли створенню програмної архітектури, яка дозволяє ефективно взаємодіяти з іншими компонентами системи та забезпечувати легкість розширення функціоналу додатку у майбутньому.

Практичне значення отриманих результатів дослідження розкривається у трьох аспектах: теоретичному, методичному та прикладному, що визначає їхню важливість у практиці та можливість впровадження в різні сфери.

У теоретичному аспекті результати дослідження сприяють поглибленню знань у галузі розпізнавання алфавітно-цифрової інформації методами інтелектуального аналізу даних. Вони розкривають нові підходи та перспективи використання інтелектуальних методів для вдосконалення точності та швидкості розпізнавання на зображеннях.

Методичний аспект полягає в розробці конкретних методів і підходів, які можуть бути використані для практичного впровадження в системи розпізнавання. Рекомендації та розроблені підходи можуть служити основою для подальших досліджень та практичного застосування в інших областях.

У прикладному аспекті результати дослідження мають безпосереднє застосування в розробці та вдосконаленні систем розпізнавання для різних галузей, включаючи технології безпеки, автоматизацію та інтелектуальні системи. Це дозволяє підвищити ефективність та надійність програмного забезпечення в умовах сучасного інформаційного суспільства.

Робота складається зі вступу, 3 розділів, висновків; містить 90 сторінок тексту, 29 рисунків, 9 таблиць, 2 додатки. Список використаних джерел включає 17 найменувань.

# РОЗДІЛ 1 ТЕОРЕТИЧНІ ОСНОВИ

## 1.1 Основні принципи систем оптичного розпізнавання алфавітно-цифрової інформації

Оптичне розпізнавання тексту (Optical Character Recognition) – це технологія, яка використовується для автоматичного перетворення зображень тексту в електронний текст.

Основні принципи OCR включають в себе наступне (рис. 1.1):

а) Зчитування зображення;

б) Обробка зображення;

в) Сегментація тексту;

г) Розпізнавання символів.


Рисунок 1.1 – Основні етапи роботи системи оптичного розпізнавання тексту

### 1.1.1 Зчитування зображення

Процес зчитування зображення в OCR включає кілька етапів, які допомагають підготувати та опрацювати вхідне зображення для подальшого розпізнавання тексту.

Основні етапи зчитування зображення в OCR виглядають приблизно так:

а) Отримання вихідного зображення;

б) Перетворення в формат, зрозумілий для OCR.

### 1.1.2 Обробка зображення

Обробка зображення дозволяє покращити якість вхідних зображень та сприяти точнішому розпізнаванню тексту. Попередня обробка може включати різні техніки і методи:

Корекція контрасту: Збільшення або зменшення контрасту може поліпшити видимість тексту та підвищити якість зображення.

Фільтрація шуму: Зображення може містити різні види шуму, такі як артефакти сканування або мерехтіння. Використання фільтрів дозволяє зменшити цей шум і збільшити чіткість тексту.

Видалення блисків і відблисків: Яскравість або відблиску можуть заважати правильному розпізнаванню тексту. Техніки видалення блисків можуть бути використані для зменшення цього впливу.

Нормалізація розмірів: Вирівнювання розмірів та пропорцій елементів зображення може поліпшити сприйняття та розпізнавання тексту.

Усунення тіней: Тіні на зображенні можуть впливати на якість розпізнавання. Техніки усунення тіней можуть включати зменшення їх інтенсивності або видалення.

Узагальнення кольорів: В деяких випадках, конвертація кольорових зображень в чорно-білі або в градації сірого може полегшити обробку та розпізнавання тексту.

Корекція рівня яскравості: Забезпечення рівномірного розподілу яскравості на зображенні сприяє більш точному розпізнаванню тексту та підвищує чіткість зображення. [2]

### 1.1.3 Переведення в градації сірого або чорно-білого

Для спрощення подальшого аналізу тексту необхідно змінити градації сірого або чорно-білого під час процесу обробки зображення OCR. При цьому враховуються деякі важливі елементи.

Важливим кроком є *бінаризація*, яка ділить зображення на два окремі кольори: чорний і білий. Це зберігає лише дві потенційні вартості для кожного пікселя, що полегшує подальшу обробку та розпізнавання тексту. Градації сірого можуть бути необхідні для деяких завдань. У цих ситуаціях кожен піксель буде представлений числовим значенням, що стосується його яскравості. Хоча він надає більше деталей, він може вимагати більше пам’яті.

Перетворення зображення в градації сірого зазвичай використовується для спрощення подальшої роботи з ним. Значення яскравості для кожного пікселя обчислюється в діапазоні від 0 до 255. Білий має 255 рівнів яскравості, тоді як чорний має 0. Такі розрахунки виконуються за допомогою цієї формули [3]:

$$I = 0{,}299R + 0{,}587G + 0{,}114B, \qquad (1.1)$$

де R - значення червоного
G - значення зеленого
B - значення синього

Важливо пам’ятати, що вибір між бінаризацією та градаціями сірого може залежати від конкретного завдання та особливостей зображення. Крім

того, поріг бінаризації визначає, чи вважаються пікселі чорними чи білими, і цей параметр можна оптимізувати для певних умов зображення.

Сучасні методи OCR дозволяють легко вибирати між бинаризацією та градаціями сірого відповідно до вимог конкретного завдання, що гарантує найкращі результати розпізнавання тексту при різних умовах вхідних даних.

### 1.1.4 Сегментація тексту

Сегментація тексту допомагає розрізнити окремі текстові елементи на зображенні. Подальша обробка та розпізнавання тексту полегшується за допомогою цього процесу.

На початку процесу сегментації текст зображення розділяється на окремі частини, такі як символи, слова чи рядки. У цьому випадку може знадобитися відокремити текстові області від інших елементів зображення. Техніки сегментації можуть брати до уваги різні елементи тексту, такі як форма, розмір, інтенсивність або розташування в просторі. Деякі методи використовують алгоритми машинного навчання, щоб автоматично визначити межі текстових елементів на зображенні.

В залежності від конкретного завдання текст після сегментації може бути оброблено на рівні рядків, символів або слів. Цей етап є важливим для того, щоб система OCR могла правильно розпізнавати та ідентифікувати різні частини тексту на зображенні. [2]

Під час ефективної сегментації тексту можна підвищити точність розпізнавання та забезпечити правильну ізоляцію текстових елементів для обробки. Врахування контексту та особливостей мови також може покращити результати сегментації, особливо коли йдеться про різноманітні види зображень і документів.

### 1.1.5 Розпізнавання алфавітно-цифрової інформації

Основним етапом оптичного розпізнавання алфавітно-цифрової інформації (OCR) є розпізнавання символів, коли система намагається

розрізнити окремі символи на підготовленому для обробки текстовому зображенні. Цей процес використовує моделі машинного навчання та алгоритми для виявлення літер, цифр або інших символів на зображенні.

Шаблонне визначення, нейронні мережі та статистичні моделі — це деякі з багатьох методів, які можуть бути використані в методах розпізнавання символів. В той час як нейронні мережі можуть використовувати глибоке навчання для автоматичного визначення ознак і зв'язків між символами, шаблонне визначення базується на порівнянні вхідного символу з попередньо збереженими шаблонами. Оскільки різні алгоритми можуть працювати на різних мовах і складнощах, важливо враховувати особливості мови та тип символів, які розпізнаються. Для підвищення точності розпізнавання символів деякі системи OCR можуть використовувати контекстну інформацію та інші характеристики зображення.

Корекція помилок дозволяє виправити помилки та помилки, які можуть виникнути під час процесу розпізнавання. На цьому етапі метою є підвищення точності та надійності результатів. Контекстуальний аналіз, використання словників, статистичні моделі та методи машинного навчання — це лише деякі з методів, які можуть бути використані для корекції помилок алгоритмів. Щоб виправити помилки на рівні слів, контекстний аналіз може враховувати лінгвістичний контекст і взаємозв'язки між словами.

Для виправлення найбільш ймовірних помилок можна порівняти розпізнані слова з мовним словником або корпусом тексту за допомогою статистичних моделей і словників. Це може включати аналіз контексту та оцінку того, наскільки ймовірно, що конкретне слово буде використано в певному контексті. На основі великої кількості правильних текстів і їх відповідних розпізнаних варіантів методи машинного навчання можуть бути використані для автоматичного навчання алгоритмів. Такі моделі можуть бути адаптовані до особливостей тексту та виправлення помилок, які можуть бути специфічними для певних документів чи жанрів.

## 1.2 Методи розпізнавання алфавітно-цифрової інформації

До традиційних методів розпізнавання тексту в основному відносяться: Шаблонні методи оптичного розпізнавання символів (OCR) передбачають створення бібліотеки шаблонів символів, які потім зіставляються з символами на вхідному зображенні для їх розпізнавання. Однак ці методи чутливі до таких факторів, як деформація символів і варіації освітлення, а також вимагають попередньої побудови великої кількості шаблонів символів.

Методи розпізнавання алфавітно-цифрової інформації, засновані на виділенні ознак, виділяють різні ознаки із зображень символів, наприклад, ознаки країв та проекційні ознаки. Ці ознаки потім використовуються в поєднанні з класифікаторами для розпізнавання символів. Однак ці методи можуть бути недостатньо надійними, щоб впоратися з такими факторами, як поворот символів і зміна масштабу, які можуть вплинути на точність розпізнавання. можуть вплинути на точність розпізнавання. Методи на основі статистичних моделей в OCR використовують статистичні моделі, такі як приховані Маркова (HMM) або умовні випадкові поля (CRF), щоб встановити зв'язок між послідовністю символів і зображенням для розпізнавання тексту. Ці методи можуть впоратися з варіаціями в розміщенні тексту і забезпечити кращі результати в складних сценаріях. Однак вони вимагають складного навчання моделі та налаштування параметрів, що може бути можуть займати багато часу та вимагати значних обчислювальних ресурсів. В останні роки багатообіцяючі результати в розпізнаванні текстів показали методи, засновані на глибокому навчанні.

Ці методи використовують глибокі нейронні мережі для автоматичного розпізнавання релевантних ознак зображень символів і прогнозування на основі вивчених репрезентацій. Моделі глибокого навчання продемонстрували покращену ефективність у вирішенні різноманітних проблем розпізнавання, зокрема деформації символів, варіацій освітлення, обертання та зміни масштабу. Вони також мають потенціал для роботи з різними мовами та шрифтами.

Загалом, хоча методи на основі шаблонів, вилучення ознак і статистичних моделей широко використовуються в розпізнаванні текстів, методи на основі глибокого навчання стали потужним підходом (див. таблицю 1.1), який долає багато обмежень традиційних методів.

Ці моделі глибокого навчання можуть навчатися на великих масивах даних і здатні вивчати складні шаблони, що призводить до більш точного та надійного розпізнавання тексту.

Таблиця 1.1

Методи розпізнавання алфавітно-цифрової інформації

| **Назва методу** | **Переваги** | **Обмеження** |
|---|---|---|
| Шаблонний метод | Легко та ефективно розпізнавати однакові або схожі шрифти | Не може розпізнавати символи різних шрифтів, масштабів і нахилів |
| Ознаковий метод | Дозволяє швидко розпізнавати символи з широким діапазоном шрифтів і нахилів | Необхідно точно визначити ознаки та їхні значення |
| Структурний метод | Чудовий для розпізнавання символів комплексної форми та змінної структури. | Необхідні складні алгоритми для аналізу зв’язків між структурами. |

### 1.2.1 Шаблонний метод

Шаблонні методи перетворюють зображення окремого символу в растрове, порівнюють його зі всіма шаблонами, наявними в базі і вибирають шаблон з найменшою кількістю крапок, відмінних від вхідного зображення. Шаблонні методи досить стійкі до дефектів зображення і мають високу

швидкість обробки вхідних даних, але надійно розпізнають тільки ті шрифти, шаблони яких їм «відомі». І якщо розпізнаний шрифт хоч трохи відрізняється від еталонного, шаблонні методи можуть робити помилки навіть при обробці дуже якісних зображень. [4]

Ця стратегія використовує попередньо збережені шаблони для порівняння вхідних зображень. Порівнявши кожен символ із шаблонами, було визначено найбільш схожий символ.

Переваги: легко та ефективно розпізнавати однакові або схожі шрифти.

Обмеження: не може розпізнавати символи різних шрифтів, масштабів і нахилів.

### 1.2.2 Ознаковий метод

Ознакові методи базуються на тому, що зображенню ставиться у відповідність N-мірний вектор ознак. Розпізнавання полягає в порівнянні вектора ознак з набором еталонних векторів тієї ж розмірності. Переваги методу – простота реалізації, хороша узагальнююча здатність, висока швидкість розпізнавання. Недолік методу – висока чутливість до дефектів зображення. Крім того, ознакові методи мають інший недолік — на етапі виділення ознак відбувається незворотня втрата частини інформації про символ. Виділення ознак проходить незалежно, тому інформація про взаємне розташування елементів символів втрачається. [4]

Аналізує кілька ознак або характеристик, які можуть мати символи, наприклад форму, розмір, кут нахилу тощо.

Використовує вектори ознак, щоб порівнювати та класифікувати символи.

Переваги: дозволяє швидко розпізнавати символи з широким діапазоном шрифтів і нахилів.

Обмеження: необхідно точно визначити ознаки та їхні значення.

### 1.2.3 Структурний метод

Структурні методи розпізнавання зберігають інформацію не про поточкове написання символу, а про його топологію. Еталон містить інформацію про взаємне розташування окремих складових частин символу. Перевага методу – стійкість до зсуву і повороту символу на невеликий кут, до різних стильових варіацій шрифтів. Однак, при повороті на кут, більший десяти градусів, даний метод не може бути використаний для розпізнавання символів. При застосування цього методу неважливими стають такі ознаки як розмір букви, що розпізнається і навіть шрифт, яким вона надрукована. Проте, основною проблемою цього методу є ідентифікація знаків, які містять певні дефекти (наприклад, розрив ліній або з'єднання сусідніх ліній). [4]

Використовує аналіз структурних і геометричних відносин, які існують між елементами символу. Встановлення структур, які пояснюють, як символ складається з основних елементів.

Переваги: Чудовий для розпізнавання символів комплексної форми та змінної структури.

Обмеження: необхідні складні алгоритми для аналізу зв’язків між структурами.

## 1.3 Типи систем розпізнавання алфавітно-цифрової інформації

Існує безліч напрямків, в яких дослідження в галузі розпізнавання текстів проводились протягом останніх років. У цьому розділі розглядаються різні типи систем розпізнавання тексту, що з'явилися в результаті цих досліджень. з'явилися в результаті цих досліджень. Ми можемо класифікувати ці системи ці системи на основі способу отримання зображення, символів підключення, шрифтові обмеження тощо. На рис. 1.2 показано класифікацію системи розпізнавання символів.

Залежно від типу введення, системи розпізнавання можна розділити на системи розпізнавання машинних символів та системи розпізнавання рукописного тексту. Перше є відносно простішим завданням, оскільки

символи, як правило, мають однакові розміри, і їхнє розташування на сторінці можна передбачити. Розпізнавання рукописних символів є дуже складним завданням через різний стиль письма користувачів, а також різні рухи пера для написання одного і того ж символу.

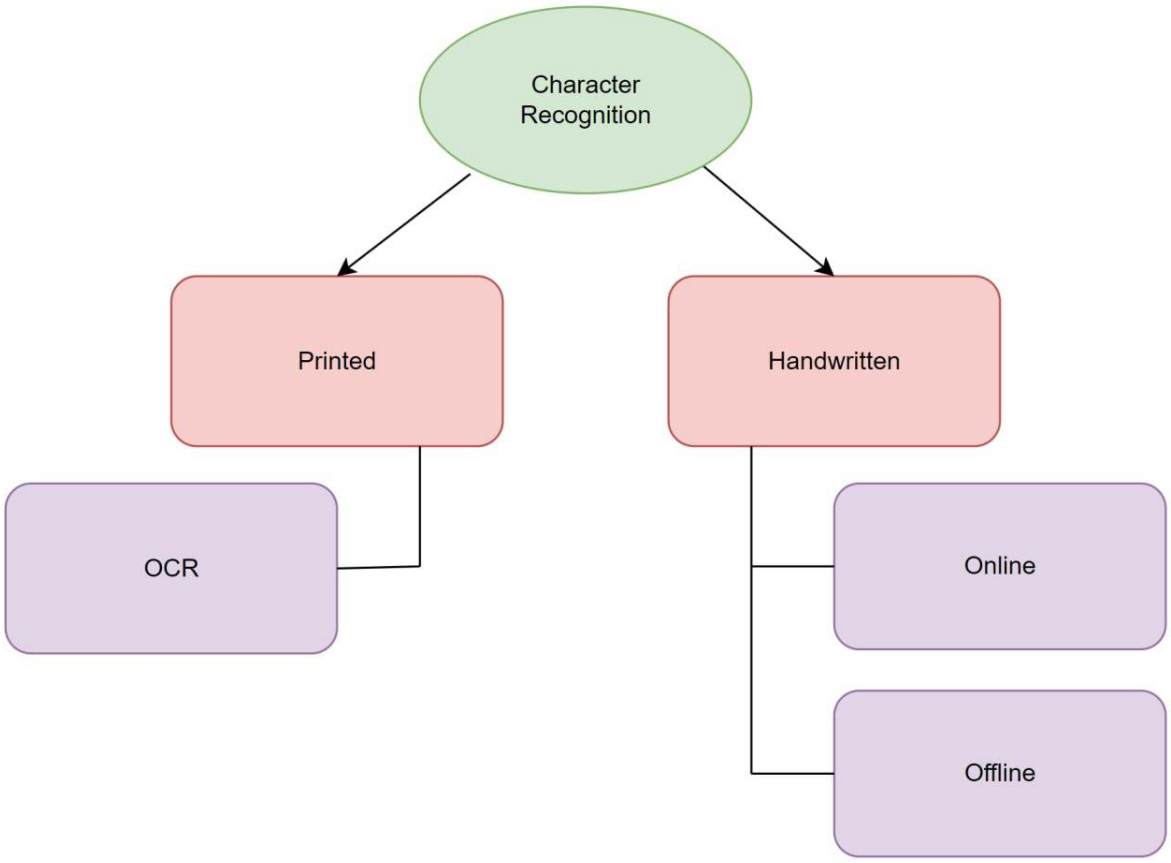


Рисунок 1.2 – Типи систем розпізнавання символів

Ці системи можна розділити на дві підкатегорії: онлайн та офлайн системи. Перші працюють у режимі реального часу, поки користувач пише символ. Вони менш складні, оскільки можуть фіксувати часову або часову інформацію, тобто швидкість, швидкість, кількість зроблених штрихів, напрямок написання штрихів тощо. Крім того, немає необхідності в техніці проріджування, оскільки слід пера має ширину в декілька пікселів. Офлайн-системи розпізнавання працюють зі статичними даними, тобто на вхід подається растрове зображення. Отже, виконати розпізнавання дуже складно.

Одним з передових напрямків у сучасних системах розпізнавання символів є використання нейронних мереж. Ці системи використовують глибоке навчання для автоматичного визначення особливостей та патернів у зображеннях символів. Нейронні мережі можуть працювати як з шрифтовими, так і безшрифтовими символами, завдяки їх здатності автоматично вивчати різноманітні структури. Вони базуються на глибоких конволюційних мережах або рекурентних нейронних мережах, які дозволяють виявляти складні залежності між пікселями зображення.

Перевагою використання нейронних мереж є їх здатність автоматично адаптуватися до різних стилів та типів шрифтів без необхідності створення окремого модулю для кожного шрифту. Це робить їх універсальними та ефективними для розпізнавання символів в різноманітних сценаріях. Незважаючи на переваги, системи на основі нейронних мереж можуть вимагати значної кількості даних для тренування та потребують великої обчислювальної потужності для ефективної роботи. Також, необхідно враховувати аспекти безпеки та конфіденційності при використанні глибоких моделей для розпізнавання символів, особливо якщо вони застосовуються в сферах, пов'язаних із конфіденційною інформацією.

Без шрифтові алгоритми вимірюють та аналізують різні характеристики окремих символів і на основі цих даних проводять розпізнавання символів.

Недоліками даного методу є низька якість розпізнавання, що нижча, ніж у шрифтового методу та недостатній коефіцієнт надійності розпізнавання символу.

Перевагами є універсальність, незалежність від користувача та простота в навчанні алгоритму. Тобто одна база даних може ефективно працювати на доволі різних шрифтах.

В більшості без шрифтових систем оптичного розпізнавання символів використовується принцип векторизації растрового зображення, тобто опис растрового зображення математичними формулами.  Серед алгоритмів векторизації найбільшу популярність мають хвильовий метод та метод

векторизації растрового скелету. В основі хвильового методу лежить хвиля, яка поширюється по пікселях символу. При проходженні хвилі по символу будується векторний скелет (граф). Вершини цього графа ставляться через певний крок і відповідають середині фронту хвилі. Після повного проходження символу, вершини створеного графа підлягають коригуванню.

Другий метод без шрифтового розпізнавання символів являє собою процедуру скелетизації растрового зображення.

Скелетизація представляє собою покрокове зняття зовнішнього шару пікселів з зображення, поки не залишаться лінії товщиною в один піксель. Після скелетизації отримане зображення підлягає векторній обробці, на основі якої і формується впізнання символу.

На практиці було реалізовано та досліджено два методи растрової скелетизації, а саме MB2 і алгоритм Занга-Суня. В MB2 алгоритмі для видалення пікселя використовується інформація по 24 сусіднім пікселям і для того щоб піксель був видалений його сусіди повині відповідати певним патернам. В алгоритмі Занга-Суня використовується 8 сусідніх пікселів і для видалення пікселя достатньо провести перевірку по деяким умовам.

Для самої векторизації растрового скелету використовується метод плаваючого вектора, який дозволяє проводити попередню корекцію вершин скелету в процесі його створення. [5, с. 1]

## 1.4 Аналіз сучасних методів інтелектуального аналізу даних для ідентифікації алфавітно-цифрової інформації

Сучасні методи інтелектуального аналізу даних для OCR використовують алгоритми визначення тексту як відправну точку. Найсучасніші нейронні мережі стали надзвичайно вдалими для визначення тексту в документах і зображеннях, навіть якщо він нахилений, повернутий або перекошений.

Хоча традиційні підходи, засновані на машинному навчанні, швидко розвиваються, вони займають значно більше часу на виконання і легко

випереджають алгоритми глибокого навчання як за точністю, так і за швидкістю висновку. Традиційні підходи до розпізнавання тексту проходять через серію етапів попередньої обробки, де перевіряється документ очищається від шуму і приводиться до вдалого виду.

Після цього документ двійковується для подальшого виявлення контурів, щоб допомогти у виявленні рядків і стовпців. Нарешті, символи, що будують лінії, витягуються, сегментуються та ідентифікуються за допомогою різних алгоритмів машинного навчання, таких як найближчі сусіди та машини опорних векторів.

Хоча алгоритми чудово працюють з простими наборами даних OCR, такими як легко розпізнані друковані та рукописні дані, вони упускають багато функцій, тому вони не працюють зі складними наборами даних.

Методи глибокого навчання є кращими за аналоги машинного навчання через їхню здатність ефективно витягувати велику кількість завдань. Алгоритми, які поєднують підходи на основі штучного мови (NLP), були особливо ефективними в розпізнаванні та виявлення тексту в дикій природі. Ці методи також пропонують наскрізний конвеєр виявлення, що звільняє їх від тривалих етапів попередньої обробки. [5, с. 1]

На сьогоднішній день існує багато сучасних методів розпізнавання тексту, які використовуються в різних областях, таких як комп'ютерний зор, обробка природної мови, і штучний інтелект.

Метод активного навчання: К-найближчих сусідів (k-NN): Використовується для класифікації тексту на основі його схожості з іншими текстами.

Технології обробки природної мови (NLP): Spacy, NLTK, TextBlob: Бібліотеки для обробки природної мови, які мають різноманітні функції, такі як токенізація, аналіз синтаксису, визначення частин мови та інше.

Методи машинного навчання:

а) Нейронні мережі: Глибокі нейронні мережі, зокрема рекурентні та трансформери, використовуються для розпізнавання тексту в

різних завданнях, таких як машинний переклад, розпізнавання іменованих сутностей та інше.

б) Метод опорних векторів (SVM): Використовується для класифікації тексту, інколи в комбінації з векторними представленнями слів (Word Embeddings).

Обробка природної мови (NLP):

а) Bag of Words (BoW) та TF-IDF: Класичні методи, які конвертують текст у вектори для подальшого аналізу та класифікації.

б) Word Embeddings: Сучасні техніки, такі як Word2Vec, GloVe та FastText, дозволяють представляти слова у векторній формі, зберігаючи семантичні відносини.

Глибоке навчання:

а) Трансформери: Моделі, такі як BERT та GPT (Generative Pre-trained Transformer), є потужними для розпізнавання тексту завдяки контекстно-залежним представленням слова та контексту.

б) Long Short-Term Memory (LSTM) та GRU: Рекурентні нейронні мережі, які добре справляються з моделюванням послідовностей та роблять їх придатними для розпізнавання тексту.

## 1.5 Програми розпізнавання алфавітно-цифрової інформації

Тексти можна розпізнати або перетворити з графічного зображення в текстовий формат за допомогою програмного забезпечення для оптичного розпізнавання символів (OCR). Механічне або електронне перетворення збереженого рукописного, машинописного або друкованого тексту в серію кодів, які можуть бути відображені в текстовому редакторі, називається оптичним розпізнаванням символів або OCR.

За допомогою оптичного розпізнавання тексту ви можете:

а) змінювати текст;

б) шукати слова або фрази;

в) стискати його до меншого формату;

г) друкувати або показувати вміст без втрати якості;

д) аналізувати дані та перетворювати текст на мову;

е) формат або електронний переклад.

Усі функції розпізнавання тексту, необхідні для ефективного керування документами та оптимізації процесів, включені в Acrobat Pro. Можна порівнювати два документи, додавати коментарі та відгуки до документів, сканувати таблиці за допомогою унікального інструменту, а також мати доступ до всіх основних інструментів розпізнавання, які вам потрібні в Pro-версії Acrobat. Через кілька секунд після сканування документи можна редагувати на екрані комп'ютера. Використовуючи безкоштовне програмне забезпечення Adobe Scan, Acrobat OCR дозволяє легко сканувати документи та конвертувати їх у PDF. Текст буде ідентифіковано автоматично та можна використовувати інструменти Adobe OCR для внесення необхідних змін.

OmniPage - це програма оптичного розпізнавання символів (OCR) від Kofax Incorporated.

OmniPage була однією з перших програм оптичного розпізнавання символів для персональних комп'ютерів. Її було розроблено наприкінці 1980-х років і продано компанією Caere Corporation, яку очолював Роберт Нойс (Robert Noyce). Першими розробниками були Філіп Бернзотт, Джон Ділворт, Девід Джордж, Брайан Хіггінс і Джеремі Найт. Caere була придбана компанією ScanSoft у 2000 році. У 2005 році ScanSoft придбала Nuance Communications і перейняла її назву. 2019 року OmniPage було продано компанії Kofax Inc.

OmniPage підтримує понад 120 різних мов. OmniPage надає набори для розробки програмного забезпечення для інтеграції функцій розпізнавання тексту в інші програми, такі як Microsoft Office Document Imaging та UiPath.

ABBYY FineReader PDF - програма для оптичного розпізнавання символів (OCR), розроблена компанією ABBYY, з підтримкою редагування PDF-файлів, починаючи з версії 15. Програма працює під управлінням

Microsoft Windows 7 або новішої версії, а також (без редагування PDF) Apple macOS 10.12 Sierra або новішої версії. Перша версія була випущена у 1993 році.

Програма дозволяє конвертувати графічні документи (фотографії, скани, PDF-файли) і знімки екрану в редаговані формати файлів, включаючи Microsoft Word, Microsoft Excel, Microsoft PowerPoint, Rich Text Format, HTML, PDF/A, PDF з можливістю пошуку, CSV і txt (звичайний текст). Починаючи з версії 11 файли можна зберігати у форматі DjVu. Версія 15 підтримує розпізнавання тексту 192 мовами і має вбудовану перевірку орфографії для 48 з них.

FineReader розпізнає нові символи шляхом: навчання символів, щоб вони були додані до алфавіту розпізнавання; вибору додаткових символів зі списку і додавання їх до алфавіту обраної мови (наприклад, додавання певних ісландських символів до німецького алфавіту для німецького тексту, що описує Ісландію); додавання специфічної для домену лексики до вбудованого лексикону FineReader. Програма також дозволяє користувачам порівнювати документи, додавати анотації та коментарі, а також планувати пакетне опрацювання.

## 1.6 Методи інтелектуального аналізу даних для розпізнавання алфавітно-цифрової інформації

### 1.6.1 Метод глибокого навчання для інтелектуального аналізу даних

Глибоке навчання - це ефективний метод інтелектуального аналізу даних машинного навчання, який використовує багатошарові нейронні мережі для розуміння та аналізу складних даних.

Методи глибокого навчання значно просунулися у сфері розпізнавання тексту. Ці методи автоматично виділяють релевантні елементи з текстового введення і роблять точні прогнози за допомогою глибоких нейронних мереж, таких як згорткові нейронні мережі (CNNs) і рекурентні нейронні мережі

(RNNs). Ці моделі здатні виконувати складні завдання розпізнавання з високою точністю, оскільки вони здатні вловлювати складні закономірності та залежності в тексті, використовуючи ієрархічну структуру глибоких нейронних мереж. Здатність глибокого навчання вивчати ієрархічні уявлення та адаптуватися до різних форм введення тексту зробила революцію в галузі розпізнавання тексту і відкрила нові можливості для застосування в таких сферах, як аналіз документів, автоматична транскрипція та обробка природної мови. [6]

### 1.6.2 Метод навчання з перенесенням (Transfer Learning) для ІАД

Завдяки застосуванню попередніх знань до нової діяльності, навчання з перенесенням є потужною стратегією, яка прискорює навчання. У сфері розпізнавання тексту трансферне навчання можна використовувати для прискорення навчання для нового завдання, застосовуючи знання, отримані з попередніх моделей розпізнавання тексту. [6]

Основна ідея, що лежить в основі трансферного навчання, полягає в тому, щоб почати з параметрів раніше навченої моделі і пристосувати їх до поточного завдання. Завдяки цьому методу потреба в значній кількості анотованих даних ефективно зменшується, а здатність моделі узагальнювати нові, неспостережувані приклади покращується. Навчання з перенесенням допомагає моделі швидко адаптуватися до поточного завдання розпізнавання тексту, використовуючи знання та уявлення, отримані під час виконання попередніх завдань, що підвищує продуктивність та ефективність. Цей метод виявився особливо корисним у ситуаціях, коли отримання маркованих даних для цільового завдання є дорогим або складним.

### 1.6.3 Метод навчання з підкріпленням (Reinforcement Learning) для ІАД

Динамічний метод навчання оптимальній поведінці через багаторазову взаємодію з навколишнім середовищем - це навчання з підкріпленням.

Навчання з підкріпленням можна використовувати в контексті розпізнавання тексту для оптимізації процесу розпізнавання, дозволяючи моделі самостійно налаштовувати свої параметри для підвищення точності розпізнавання. [6]

Основна ідея навчання з підкріпленням полягає в побудові інтерактивного зв'язку між агентом і його середовищем, яке визначається станами агента, його діями та функцією винагороди. Виходячи з поточного стану, агент може вибирати відповідні дії в контексті розпізнавання тексту, включаючи зміну параметрів моделі, і оцінювати ефективність цих дій за допомогою функції винагороди. Навчання з винагородою дозволяє моделі багаторазово налаштовувати свої параметри через постійну взаємодію з навколишнім середовищем, зрештою покращуючи точність розпізнавання тексту. [6]

### 1.6.4 Метод рекурентних нейронних мереж (RNN) для ІАД

Топології нейронних мереж з об'ємом пам'яті, або рекурентні нейронні мережі (RNN), ідеально підходять для моделювання та обробки послідовних даних. РНМ продемонстрували ефективність в управлінні послідовностями символів з часовими залежностями в контексті розпізнавання тексту. Рекурентні зв'язки є основною ідеєю РНМ; ці зв'язки дозволяють мережі інтерпретувати кожен символ, включаючи контекстну інформацію. РНМ можуть краще розпізнавати текст, вловлюючи залежності між символами, завдяки своїй здатності враховувати послідовний контекст. РНМ використовуються для різноманітних завдань розпізнавання тексту, в тому числі для створення текстів і моделювання мови. Ці моделі створюють зв'язний і контекстуально релевантний текст, використовуючи властивість пам'яті РНМ, що робить їх корисними інструментами в обробці природної мови та суміжних галузях. [6]

### 1.6.5 Метод машинного навчання для інтелектуального аналізу даних

У сфері розпізнавання алфавітно-цифрової інформації методи інтелектуального аналізу даних такі як машинне навчання та штучний інтелект мають вирішальне значення. Аналізуючи величезні обсяги навчальних даних, алгоритми машинного навчання навчають комп'ютери закономірностям і характеристикам символів, що призводить до точного розпізнавання тексту. [6] У цій галузі особливо гарні результати показали методи глибокого навчання, такі як рекурентні нейронні мережі (RNN) та згорткові нейронні мережі (CNN). Ці моделі глибокого навчання мають велику стійкість і навички узагальнення, що дозволяє їм вирішувати такі проблеми, як деформація символів і коливання освітленості. Вони також можуть автономно вивчати зображення об'єктів на фотографіях.

Крім того, системи розпізнавання тексту можуть працювати краще і ефективніше за допомогою методів штучного інтелекту, таких як навчання з підкріпленням і навчання з перенесенням.

Отже, машинне навчання та штучний інтелект пропонують потужні методи та інструменти для розпізнавання тексту, що сприяє розвитку та використанню технологій розпізнавання тексту.

### 1.6.5 Метод згорткових нейронних мереж (CNN) для ІАД

Згорткові нейронні мережі (Convolutional Neural Network, CNN) - це специфічна архітектура нейронних мереж, яка чудово справляється з вилученням ознак із зображень. CNN ефективно використовуються в розпізнаванні тексту для виконання завдань класифікації та вилучення релевантної інформації з символів. Основна ідея ЗНМ полягає в поступовому вилученні характеристик із зображень шляхом застосування багатьох згорткових шарів і об'єднання шарів, а потім класифікації зображень за допомогою повністю з'єднаних шарів. CNN може точно розпізнавати текст, фіксуючи як локальні, так і глобальні шаблони в тексті за допомогою цього

ієрархічного методу вилучення ознак. Високий рівень точності був досягнутий за допомогою ШНМ у різних програмах розпізнавання тексту, таких як розпізнавання рукописних цифр і оптичне розпізнавання символів (OCR). Використання ЗНМ у розпізнаванні тексту суттєво вдосконалило цю галузь і проклало шлях до кількох практичних застосувань в обробці документів, автоматизованому транскрибуванні та інших суміжних дисциплінах.

### 1.6.6 Трансформер (архітектура глибокого навчання)

Трансформер - це архітектура глибокого навчання, заснована на механізмі багатоголової уваги, запропонована в статті 2017 року "Attention Is All You Need". Вона не має рекурентних блоків, а отже, потребує менше часу на навчання, ніж попередні рекурентні нейронні архітектури, такі як довга короткочасна пам'ять (LSTM), і її пізніша варіація була прийнята для навчання великих мовних моделей на великих (мовних) наборах даних, таких як корпус Вікіпедії та Common Crawl. Вхідний текст розбивається на n-грами, закодовані як токени, і кожен токен перетворюється на вектор, шукаючи його в таблиці вбудовування слів. На кожному рівні кожна лексема контекстуалізується в межах контекстного вікна з іншими (незамаскованими) лексемами за допомогою паралельного багатоголового механізму уваги, що дозволяє посилити сигнал для ключових лексем і послабити менш важливі лексеми. Стаття про трансформер, опублікована в 2017 році, базується на механізмі уваги на основі softmax, запропонованому Багданау та ін. у 2014 році для машинного перекладу, а Fast Weight Controller, схожий на трансформер, був запропонований у 1992 році. [6]

Трансформер усередині складається з компонента, що кодує, компонента, що декодує, і зв'язку між ними (рис. 1.3). Етап кодування складається з попередньо навченої моделі трансформатора зору. А стадія декодера складається з попередньо навченої моделі мовного трансформатора.

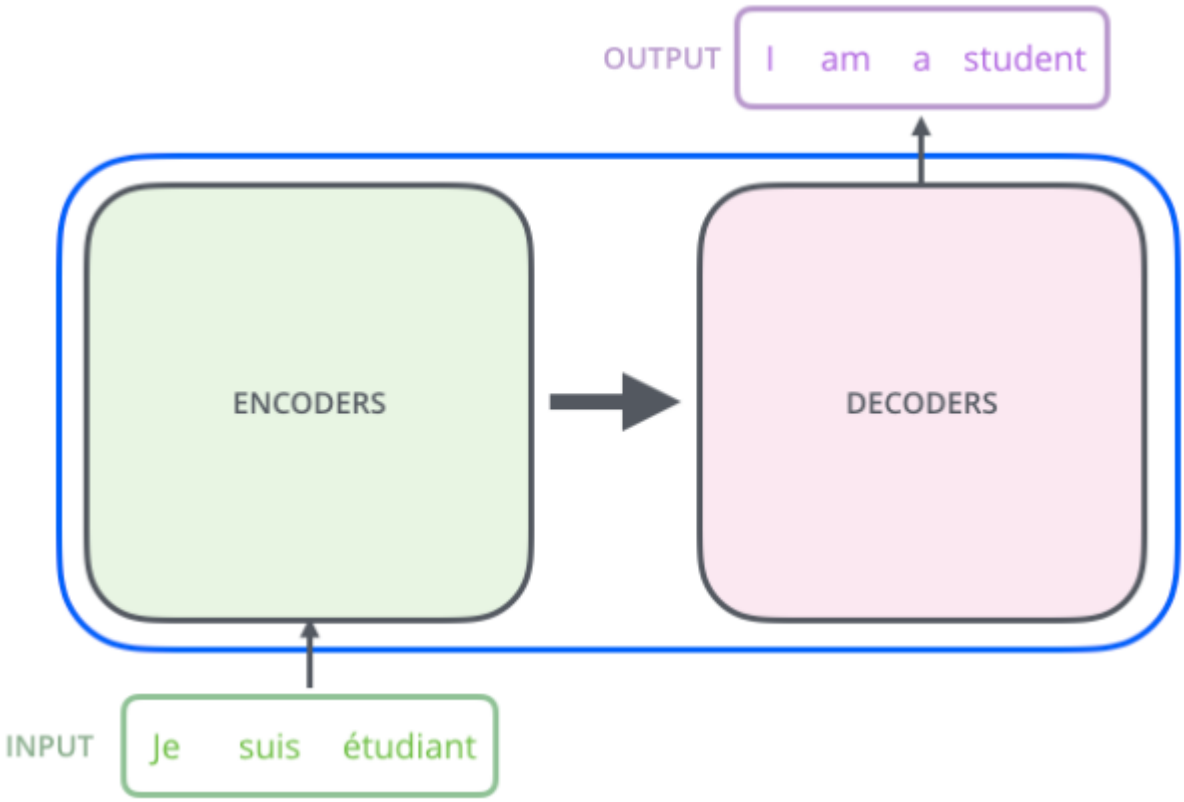


Рисунок 1.3 – Компоненти моделі Трансформер

1.7 Висновки

Таким чином, у цьому розділі представлено основні принципи та методи оптичного розпізнавання алфавітно-цифрової інформації, включаючи зчитування зображень, обробку зображень, переведення в градації сірого та чорно-білого, сегментацію тексту та розпізнавання символів.

Проаналізовано методи розпізнавання, такі як шаблонний, ознаковий та структурний.

Виявлено роль машинного навчання та штучного інтелекту для розпізнавання алфавітно-цифрової інформації, а також описано застосуваня методів машинного навчання штучного інтелекту.

Проведено аналіз сучасних методів ідентифікації алфавітно-цифрової інформації.

# РОЗДІЛ 2 ДОСЛІДЖЕННЯ ТА ОПИС ЗАСОБІВ РЕАЛІЗАЦІЇ СИСТЕМИ

У даному розділі проведено аналіз проблем систем розпізнавання алфавітно-цифрової інформації. Розглядаються проблеми та виклики, які виникають при реалізації систем розпізнавання, зокрема в аспектах точності та швидкості роботи.

Після визначення проблематики систем розпізнавання обговорюється вибір інструментальних засобів для реалізації проекту. Особлива увага приділяється OpenCV та Tesseract OCR як потенційним технологічним рішенням для вирішення завдань оптичного розпізнавання символів. Проводиться ретельний аналіз їхніх можливостей, переваг та недоліків з метою досягнення оптимальних результатів в контексті системи розпізнавання.

## 2.1 Проблеми систем розпізнавання алфавітно-цифрової інформації

Для точного та якісного розпізнавання символів потрібні зображення високої якості або з високою роздільною здатністю з певними базовими структурними якостями, такими як відмінна диференціація тексту і фону. Враховуючи, що методи створення зображень часто мають значний вплив на їхню якість, ці методи також відіграють вирішальну роль в успіху і точності розпізнавання.

Розпізнавання символів часто демонструє чудову продуктивність і високу точність, коли застосовується до зображень, створених за допомогою сканерів. З іншого боку, через проблеми, пов'язані з камерою або навколишнім середовищем, фотографії з камер зазвичай не такі якісні, як відскановані зображення, що використовуються для розпізнавання.

Погана якість зображення може значно впливати на точність оптичного розпізнавання тексту. Зображення може бути розмитим, містити шум, тіні, артефакти або мати низький контраст. Ці аспекти можуть ускладнювати

виділення тексту від тла та спотворювати форму літер. Може виникнути багато помилок, деякі з них пояснюються нижче.

Шрифти. Курсивні та скриптові шрифти символів можуть перекривати один одного, що ускладнює виконання деяких основних процесів розпізнавання, наприклад, сегментацію. Символи різних шрифтів мають великі варіації всередині класу і утворюють багато підпросторів шаблонів, що ускладнює точне розпізнавання, коли номер класу символів великий.

Розмиття. Цифрові камери призначені для роботи на різних відстанях, тому їх фокусування є важливим. Різкість символів необхідна для найвищої точності розпізнавання та сегментації символів. Невелика зміна точки зору може призвести до нерівномірного фокусування на коротких відстанях і великих діафрагмах. Нечіткість у фокусі та нечіткість у русі — це два види нечіткості, які часто зустрічаються у фотографіях. У точці зйомки об’єкта, що рухається, коли коефіцієнт затінення камери недостатньо високий, сенсор потрапляє на сцену, яка постійно змінюється. Таким чином, частини, що рухаються, демонструватимуть розмиття.

Багатомовне середовище. У той час як багато латинських мов мають велику кількість символів, такі мови, як японська, китайська та корейська, мають багато класів символів. Арабські мови мають сполучені символи, які змінюють стиль написання залежно від контексту. Склади гінді представлені поєднанням сотень різних алфавітних комбінацій літер. Розпізнавання тексту у відсканованих документах залишається основним питанням досліджень у багатомовному середовищі [8], оскільки розпізнавання складної символіки є більш проблематичним.

Співвідношення сторін. Текст має різні співвідношення сторін. Деякі тексти, наприклад, підписи на відео, можуть бути набагато довшими, ніж інші, наприклад, текст на дорожніх знаках. Для того, щоб виявити текст, процес пошуку повинен враховувати розташування, масштаб і довжину тексту, що додає значної обчислювальної складності.

Нахил (спотворення перспективи). Зображення, записане портативною камерою, може не завжди бути паралельним площині форми, тому це не завжди стосується зображень документів, отриманих сканерами, які завжди паралельні площині сенсора. Текстові рядки, що знаходяться далі від камери, виглядають меншими, ніж ті, що знаходяться ближче до камери, які виглядають більшими. Нахилені зображення є результатом цієї умови. Якщо розпізнавач не враховує перспективу, це призводить до зниження швидкості та точності розпізнавання.

Мобільні телефони мають перевагу завдяки датчикам орієнтації. Вони можуть розпізнати, чи пристрій нахилений, і коли відбувається скручування вони можуть заборонити клієнтам фотографувати. Дозволяючи користувачеві користувачеві вирівняти площину форми з площиною камери також ця функція також дозволяє користувачеві вирівнювати площину форми з площиною камери. Таким чином, датчики орієнтації можуть гарантувати що отримані знімки задовольняють певному ступеню рівності.

Для систем оптичного розпізнавання символів точка зору на вхідне зображення, отримане з камери портативного пристрою або іншого гаджета, який використовувався для зйомки, не є фіксованою, як на вході сканера, що може призвести до перекосу текстових рядків від їх унікальної орієнтації. При подачі такого перекошеного зображення на OCR-класифікатор будуть спостерігатися вкрай незадовільні результати.

Складність сцени. У звичайному середовищі ми можемо бачити велику кількість штучних об'єктів, які потрапляють на зображення, зроблені камерою, наприклад, картини, будівлі та символи. Ці об'єкти мають схожу структуру та зовнішній вигляд з текстом, що робить розпізнавання тексту на обробленому зображенні дуже складним завданням.  Сам текст регулярно розміщується, щоб полегшити його розпізнавання. Складність сцени полягає в тому, що навколишній пейзаж ускладнює відокремлення тексту від штучних об'єктів.

Умови нерівномірного освітлення. Часто зйомка в природних умовах призводить до того, що нерівномірне освітлення та тіні. Це створює певні

труднощі для розпізнавання тексту, оскільки це погіршує бажані характеристики зображення, а отже призводить до менш точного виявлення, сегментації та розпізнавання результатів.

Саме нерівномірність освітлення відрізняє відскановане зображення від зображення, зробленого камерою. Відскановані зображення є кращими за фотознімки через їхню вищу якість і відсутність нерівномірності освітлення та тіні. Хоча використання спалаху на камері може допомогти вирішити деякі з цих проблем нерівномірного освітлення, він також створює нові труднощі [9].

## 2.2 Вибір інструментальних засобів для реалізації системи розпізнавання алфавітно-цифрової інформації

У контексті створення системи розпізнавання алфавітно-цифрової інформації, вибір інструментів визначається необхідністю обробки та аналізу зображень, а також взаємодії з алгоритмами розпізнавання тексту. Для цієї задачі використовуються Python, Tesseract та OpenCV.

Однією з ключових переваг обраного стеку інструментів є їхня взаємна сумісність та можливість ефективної інтеграції. Python, як мова програмування, забезпечує зручний інтерфейс для взаємодії з бібліотеками Tesseract та OpenCV. Це робить можливим плавний обмін даними та результатами між компонентами системи.

Обрані інструменти дозволяють розробникам забезпечити гнучкість та розширюваність системи. Python має велику спільноту розробників, яка активно підтримує та розвиває мову, що визначається швидкістю вирішення проблем та постійним оновленням. Також, інтеграція нових алгоритмів розпізнавання чи розширення функціоналу може бути виконана безпроблемно завдяки робочій взаємодії із зовнішніми бібліотеками.

Використання OpenCV та Tesseract у поєднанні з Python дозволяє створити ефективну та швидкодіючу систему розпізнавання. Оптимізовані алгоритми обробки зображень в OpenCV сприяють швидкості аналізу великої

кількості зображень, тоді як Tesseract відомий своєю високою швидкістю та точністю оптичного розпізнавання тексту.

Обираючи Python, Tesseract та OpenCV для реалізації системи розпізнавання алфавітно-цифрової інформації, отримуємо потужний інструментарій, який поєднує в собі ефективність, гнучкість та зручність розробки. Цей набір інструментів визначається високою продуктивністю та здатністю ефективно працювати у різних сценаріях застосування системи.

### 2.2.1 Python

Python - це високорівнева, інтерпретована мова програмування, яка набула широкої популярності завдяки своїй простоті та ефективності. Вона володіє чистим і зрозумілим синтаксисом, що дозволяє легко читати і писати код, що робить її ідеальним вибором для початківців та досвідчених розробників.

Одна серед головних переваг Python є його багатий набір бібліотек, які допомагають розробникам реалізовувати різноманітні завдання. У контексті системи розпізнавання алфавітно-цифрової інформації, бібліотеки, такі як OpenCV, TensorFlow, та PyTorch, можуть бути використані для реалізації алгоритмів машинного навчання та обробки зображень.

Незважаючи на свої переваги, Python має певні недоліки, такі як обмежена швидкодія та високе споживання пам'яті порівняно з іншими мовами програмування, такими як C++ або Java. Це може бути важливим фактором при роботі з великими обсягами даних чи завданнями, де важлива висока продуктивність. Однак з урахуванням постійного розвитку і оптимізацій мови, ці недоліки поступово зменшуються, а вигоди Python залишаються значущими для багатьох проектів.

Вибір мови програмування для реалізації системи розпізнавання алфавітно-цифрової інформації, особливо в контексті Python, може бути обґрунтованим з кількох причин:

а) Простота та читабельність коду: Python відомий своєю лаконічністю та чистим синтаксисом, що робить код зрозумілим та легким у супроводженні. Це особливо важливо у задачах розпізнавання образів, де розуміння та модифікація коду можуть бути ключовими;
б) Багата кількість бібліотек: Python має широкий вибір бібліотек для обробки зображень та машинного навчання, які значно полегшують реалізацію алгоритмів розпізнавання. Бібліотеки, такі як OpenCV, TensorFlow, PyTorch, і scikit-learn, дозволяють ефективно використовувати різноманітні техніки та моделі;
в) Швидкість розробки: Python відомий своєю високою швидкістю розробки завдяки простоті та гнучкості мови. Розробники можуть швидко випробовувати та вдосконалювати алгоритми без необхідності великої кількості коду;
г) Можливість інтеграції: Python легко інтегрується з іншими мовами та технологіями, що дозволяє використовувати його в різноманітних системах та середовищах.

Обираючи Python для системи розпізнавання алфавітно-цифрової інформації, можна скористатися його зручністю, ефективністю та широким спектром інструментів для досягнення успішних результатів у вашому проекті.

### 2.2.2 Tesseract OCR

Tesseract - це OCR-движок з відкритим вихідним кодом, який витягує друкований або написаний текст із зображень. Спочатку його розробила компанія Hewlett-Packard, а згодом розробку перейняла компанія Google.

Одним із найточніших двигунів розпізнавання тексту з відкритим кодом на той час був Tesseract у 2006 році.

Історія розробки Tesseract OCR Tesseract спочатку розроблявся як власне програмне забезпечення в лабораторіях Hewlett-Packard у Брістолі, Англія, та Грілі, Колорадо, між 1985 та 1994 роками, з подальшими змінами, зробленими

у 1996 році для перенесення на Windows, та деякими міграціями з C на C++ у 1998 році. Більша частина коду була написана на C, а потім ще частина була написана на C++. Відтоді весь код було перетворено, щоб його можна було принаймні компілювати за допомогою компілятора C++.[потрібне посилання] У наступному десятилітті було зроблено дуже мало роботи. Потім він був випущений з відкритим вихідним кодом у 2005 році компанією Hewlett-Packard та Університетом Невади, Лас-Вегас (UNLV). З 2006 року розробку Tesseract спонсорує компанія Google. [10]

У версії 4 додано рушій розпізнавання на основі LSTM і моделі для багатьох додаткових мов і скриптів, що збільшило загальну кількість мов до 116 [10]. Додатково підтримується 37 скриптів. Так, наприклад, можна розпізнати текст із сумішшю західноєвропейських і центральноєвропейських мов, використовуючи модель для латинського шрифту, яким він написаний.

Версію 5 було випущено у 2021 році, після більш ніж двох років тестування та розробки.

Можливості Tesseract OCR. Tesseract входив до трійки лідерів за точністю розпізнавання символів у 1995 р. Він доступний для Linux, Windows і Mac OS X.

Tesseract до версії 2 включно міг приймати на вхід лише зображення у форматі TIFF з простим одноколонковим текстом. У цих ранніх версіях не передбачався аналіз розмітки, тому введення багатоколонкового тексту, зображень або рівнянь призводило до спотворення вихідних даних. Починаючи з версії 3.00 Тессеракт підтримує форматування вихідного тексту, позиційну інформацію hOCR та аналіз розмітки сторінки. За допомогою бібліотеки Leptonica було додано підтримку низки нових форматів зображень. Tesseract може визначати, чи є текст моноширинним або пропорційним. [10]

Початкові версії Tesseract могли розпізнавати лише англомовний текст. У Tesseract v2 додано шість додаткових західних мов (французька, італійська, німецька, іспанська, бразильська португальська, голландська). Версія 3 значно розширила мовну підтримку, включивши ідеографічні мови (китайську та

японську) і мови з написанням справа наліво (наприклад, арабську, іврит), а також багато інших шрифтів.. У версії 3.04, випущеній у липні 2015 року, додано ще 39 комбінацій мова/шрифт, що збільшило загальну кількість підтримуваних мов до понад 100. [10]

Крім того, Tesseract можна навчити працювати з іншими мовами.

Tesseract може досить добре обробляти текст справа наліво, наприклад, арабську або іврит, багато шрифтів Indic, а також CJK. Показники точності показані в цій презентації для підручника з Tesseract на DAS 2016.

Tesseract підходить для використання як бекенд і може бути використаний для складніших завдань розпізнавання, зокрема для аналізу макетів за допомогою інтерфейсу, наприклад, OCRopus.

Вихідні дані Tesseract матимуть дуже низьку якість, якщо вхідні зображення не були попередньо оброблені відповідно до нього: Зображення (особливо скріншоти) повинні бути збільшені таким чином, щоб висота тексту x становила щонайменше 20 пікселів, будь-яке обертання або нахил повинні бути виправлені, інакше текст не буде розпізнаний, низькочастотні зміни яскравості повинні бути відфільтровані високими частотами, інакше етап бінаризації Тессеракта знищить більшу частину сторінки, а темні межі повинні бути видалені вручну, інакше вони будуть неправильно інтерпретовані як символи. [10].

### 2.2.3 OpenCV

OpenCV (Open Source Computer Vision Library) - це бібліотека програмного забезпечення для комп'ютерного зору та машинного навчання з відкритим вихідним кодом. OpenCV була створена, щоб забезпечити загальну інфраструктуру для додатків комп'ютерного зору і прискорити використання машинного сприйняття в комерційних продуктах. Будучи продуктом з ліцензією Apache 2, OpenCV дозволяє компаніям легко використовувати та модифікувати код.

Бібліотека налічує понад 2500 оптимізованих алгоритмів, що включає в себе повний набір як класичних, так і сучасних алгоритмів комп'ютерного зору та машинного навчання. Ці алгоритми можна використовувати для виявлення та розпізнавання облич, ідентифікації об'єктів, класифікації дій людини на відео, відстеження руху камери, відстеження рухомих об'єктів, вилучення 3D-моделей об'єктів, створення 3D-хмар точок зі стереокамер, зшивання зображень для отримання зображення всієї сцени з високою роздільною здатністю, пошуку схожих зображень у базі даних зображень, видалення червоних очей зі знімків, зроблених зі спалахом, відстеження рухів очей, розпізнавання пейзажу та встановлення маркерів для накладання на нього доповненої реальності, тощо. Спільнота користувачів OpenCV налічує понад 47 тисяч осіб, а кількість завантажень перевищує 18 мільйонів. Бібліотека широко використовується в компаніях, дослідницьких групах та урядових установах. [11]

Поряд з такими відомими компаніями, як Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota, які використовують бібліотеку, є багато стартапів, таких як Applied Minds, VideoSurf та Zeitera, які широко використовують OpenCV. OpenCV використовується для зшивання зображень з вуличних камер, виявлення вторгнень на відео з камер спостереження в Ізраїлі, моніторингу шахтного обладнання в Китаї, допомоги роботам у навігації та підборі об'єктів у Willow Garage, виявлення нещасних випадків утоплення в басейні в Європі, запуску інтерактивного мистецтва в Іспанії та Нью-Йорку, перевірки злітно-посадкових смуг на наявність сміття в Туреччині, перевірки етикеток на продуктах на фабриках по всьому світу, а також для швидкого розпізнавання облич в Японії. [11]

Він має інтерфейси C++, Python, Java та MATLAB і підтримує Windows, Linux, Android та Mac OS. OpenCV здебільшого орієнтований на додатки технічного зору в реальному часі і використовує переваги інструкцій MMX і SSE, коли вони доступні. Зараз активно розробляються повнофункціональні інтерфейси CUDA та OpenCL. Існує понад 500 алгоритмів і приблизно в 10

разів більше функцій, які складають або підтримують ці алгоритми. OpenCV написаний на мові C++ і має шаблонний інтерфейс, який без проблем працює з контейнерами STL. [11].

Призначення OpenCV. OpenCV було розроблено, щоб максимізувати продуктивність та ефективність для завдань технічного зору, які потребують великої обчислювальної потужності. Таким чином, він значною мірою орієнтований на додатки штучного зору в реальному часі. Програма з відкритим вихідним кодом може використовувати багатоядерні процесори (багатопотоковість), оскільки вона написана ефективною мовою C.

Маючи понад 500 функцій, що охоплюють багато аспектів зору, OpenCV має на меті створити зручну для користувача інфраструктуру комп'ютерного зору, яка дозволить будь-кому швидко розробляти складні програми технічного зору. OpenCV часто використовується в таких сферах, як медична візуалізація, аналіз безпеки, роботизований зір, стереозорість (3D-зір), калібрування камер, людино-машинний інтерфейс і заводський контроль продукції.

Широкі можливості обробки зображень полегшують різні операції попередньої обробки зображень, калібрування камер, обробку відеопотоків і зшивання зображень (об'єднання декількох камер). OpenCV включає комплексну універсальну бібліотеку машинного навчання з акцентом на статистичному розпізнаванні образів і кластеризації, оскільки машинне навчання має вирішальне значення для комп'ютерного зору. OpenCV використовує прискорення графічного процесора та підтримку NVIDIA CUDA для підвищення швидкості роботи. З 2011 року OpenCV працює зі встановленими пакетами, пакетами сайтів та двійковими файлами OpenCV. Він також пропонує явний контроль над передачею даних між пам'яттю CPU і GPU за допомогою модуля OpenCV GPU.

## 2.3 Опис використаного метода інтелектуального аналізу даних

Для написання системи розпізнавання алфавітно-цифрової інформації можна використовувати різні методи інтелектуального аналізу даних. Один з потенційних підходів - це використання методів машинного навчання, зокрема, нейронних мереж. Нейронні мережі можуть бути навчені розпізнавати шаблони та закономірності в алфавітно-цифрових даних і використовувати цю інформацію для подальшого розпізнавання.

У даній роботі для завдань розпізнавання тексту та роботи з українською мовою використовувався Tesseract OCR. Зокрема, для того, щоб він почав розпізнавати алфавітно-цифрову інформацію потрібно підготувати модель та навчити її.

Tesseract був навчений на спеціальному наборі даних – tessdata. Процес навчання включав у себе підготовку великого обсягу текстових даних українською мовою, а також коректне форматування цих даних для навчання моделі. Дата сет tessdata використовувався для надання необхідних знань Tesseract щодо лексики та граматики.

### 2.3.1 Архітектура глибокого навчання Transformer для розпізнавання алфавітно-цифрової інформації

Використання архітектури трансформера для розпізнавання алфавітно-цифрової інформації є доцільним підходом в інтелектуальному аналізі даних. Трансформери, такі як BERT чи GPT, здатні працювати із послідовними даними та можуть бути успішно адаптовані для завдань OCR.

Початкове навчання трансформера на великому об'ємі текстових даних дозволяє моделі засвоїти мовні особливості та структуру тексту. Після цього проводиться тонке налаштування на наборі даних OCR для адаптації до конкретних рис зображень тексту.

Інтеграція механізму уваги в трансформері дозволяє обробляти довгі послідовності даних, що є корисним для зображень тексту. Трансформер може

бути використаний для сегментації зображень тексту та подальшого розпізнавання на рівні символів чи слів.

Модель складається з двох етапів:

а) Етап кодера складається з попередньо навченої моделі трансформера зору.

б) Етап декодера складається з попередньо навченої моделі мовного трансформера.

Щоб використовувати модель трансформер для розпізнавання текстів, спочатку потрібно навчити модель на великому наборі даних зображень і відповідних текстових транскриптів. Процес навчання передбачає подання моделі вхідних зображень і правильних транскрипцій, а також налаштування внутрішніх параметрів моделі для мінімізації похибки між передбаченнями моделі та правильними транскрипціями.

Після того, як модель навчена, потрібно використовувати її для розпізнавання тексту на нових зображеннях, надавши моделі вхідне зображення і попросивши її згенерувати транскрипцію. Після цього модель використає свої механізми уваги, щоб зосередитися на відповідних частинах зображення і згенерувати транскрипцію на основі свого навчання.

Існує кілька підходів до навчання моделей-трансформерів для розпізнавання текстів, зокрема використання наскрізних методів навчання, використання гібридних моделей, які поєднують традиційні методи розпізнавання текстів із моделями-трансформерами, а також використання попередньо навчених моделей-трансформерів і їхнє тонке налаштування для виконання завдань розпізнавання текстів.

Таким чином, ми можемо розділити завдання на 2 частини:

а) Попереднє навчання моделі

б) Завдання специфічного тонкого налаштування моделей

Наразі 2 найпопулярніші моделі - це :

а) Donut (від Google)

б) Trocr

2.3.2 Принцип роботи архітектури Transformer

В основі архітектури трансформатора лежить механізм самоуваги, а саме масштабована скалярнодобуткова увага та багатоголова увага. Цей механізм дозволяє моделі ефективно розставляти пріоритети і пов'язувати різні частини вхідної послідовності при обробці кожного конкретного елемента.

Наступне рівняння описує масштабовану скалярнодобуткову увага (рис. 2.1):

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \quad (2.1)$$

де Q – матриця, що містить запит

K – складається з усіх ключів, векторних зображень усіх слів у послідовності.

V – представляє значення, які часто збігаються з K.

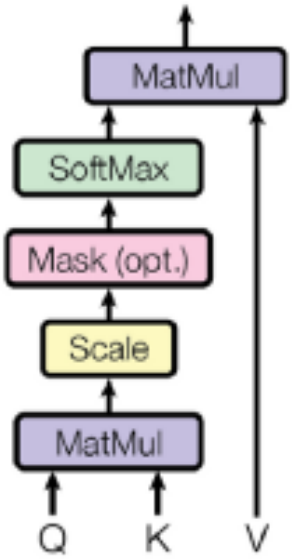


Рисунок 2.1 – Масштабована скалярнодобуткова увага

Значення V перемножуються і підсумовуються з вагами уваги a. Ваги визначаються за наступною формулою:

$$a = softmax(\frac{QK^T}{\sqrt{d_k}}) \quad (2.2)$$

де Q – матриця, що містить запит

K – складається з усіх Ключів, векторних зображень усіх слів у послідовності.

V – представляє значення, які часто збігаються з K.

Функція Softmax нормалізує значення a до шкали від 0 до 1. Далі модель застосовує ці ваги до всіх слів у значенні V.

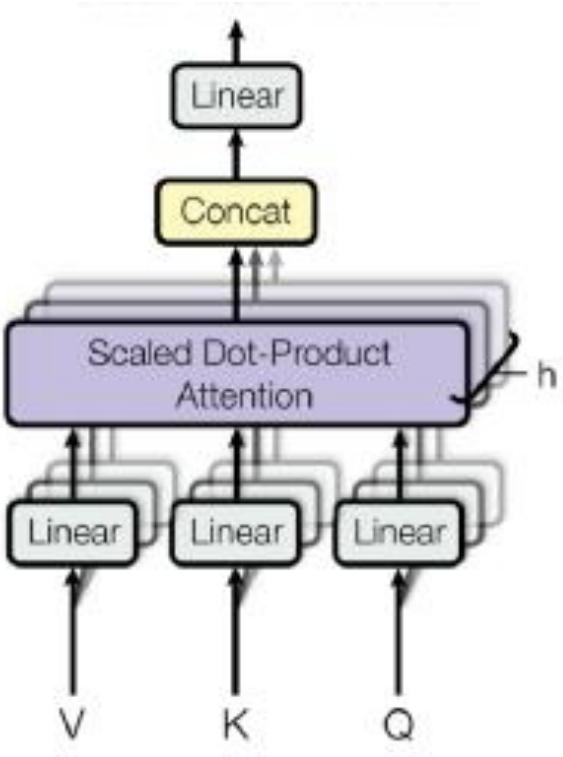


Рисунок 2.2 – Багатоголова увага складається з декількох рівнів уваги, що працюють паралельно

Багатоголова увага - це модуль для механізмів уваги, який пропускає через механізм уваги кілька разів паралельно. Незалежні результати уваги потім об'єднуються і лінійно трансформуються в очікувану розмірність. Інтуїтивно зрозуміло, що кілька головок уваги дозволяють по-різному приділяти увагу частинам послідовності (наприклад, довгостроковим залежностям порівняно з короткостроковими).

На рисунку 2.2, показано, як цей механізм уваги розпаралелюється. Багатоголовий механізм уваги дозволяє моделі звертати увагу на декілька частин одночасно.

$$MultiHead(Q, K, V) = [head_1, \dots, head_h]W_0 \quad (2.3)$$

де W – всі матриці параметрів, що вивчаються

$$head_i = Attention\ (QW_i^Q, KW_i^K, VW_i^V)$$

2.3.3 Вибір моделі Transformer для розпізнавання алфавітно-цифрової інформації

Для реалізації задачі розпізнавання алфавітно-цифрової інформації була вибрана модель від Google, а саме Transformer clovaai/donut.

Donut (Document understanding transformer) – це метод розуміння документів, який використовує наскрізну модель Transformer, що не потребує розпізнавання тексту. Donut не потребує готових OCR-API, але демонструє найсучасніші результати у вирішенні різних завдань візуального розуміння документів, таких як візуальна класифікація документів або вилучення інформації (так званий синтаксичний аналіз документів). [17]

Розуміння зображень документів (наприклад, рахунків-фактур) є основним, але складним завданням, оскільки воно вимагає складних функцій, таких як читання тексту і цілісного розуміння документа. Сучасні методи візуального розуміння документів передають завдання читання тексту готовим системам оптичного розпізнавання символів (OCR) і зосереджуються на завданні розуміння за допомогою результатів OCR.

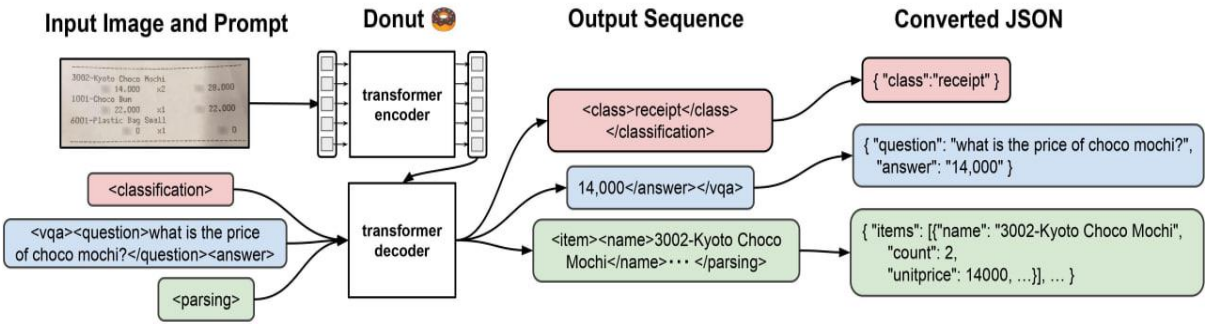


Рисунок 2.3 – Огляд архітектури моделі Donut

Хоча такі підходи на основі OCR показали перспективні результати, вони мають такі проблеми:

а) високі обчислювальних витрат на використання OCR;

б) негнучкість моделей OCR щодо мов або типів документів;

в) поширення помилок OCR на наступний процес.

Щоб вирішити ці проблеми, було представлено нову модель VDU без OCR, названу Donut. Як перший крок у дослідженні VDU без OCR. Архітектура моделі представлена на рисунку 2.3. [17]

Саме ця модель буде навчена і використана для розпізнавання алфавітно-цифрової інформації в подальшій практичній частині з додатковими інструментами, такими як Tesseract і OpenCV, опис яких представлено в цьому розділі.

## 2.4 Висновки

У цьому розділі було проведено аналіз проблем систем розпізнавання алфавітно-цифрової інформації. Розглядаються проблеми та виклики, які виникають при реалізації систем розпізнавання, зокрема в аспектах точності та швидкості роботи.

Після визначення проблем, пов’язаних із системами розпізнавання, було розглянуто інструменти, необхідні для програмної реалізації проекту, а також опис використаного метода інтелектуального аналізу даних. Особлива увага приділяється Python, OpenCV та Tesseract OCR як можливим технологічним рішенням для вирішення завдань оптичного розпізнавання символів. Було проведено детальний аналіз їхніх можливостей, переваг і недоліків.

У роботі розглянуті різні методи інтелектуального аналізу даних для розпізнавання алфавітно-цифрової інформації. Зазначено важливість використання методів машинного навчання, таких як нейронні мережі та архітектура Transformer, для ефективного розпізнавання шаблонів і тексту.

Обговорено застосування моделі Donut (Document understanding transformer) та висвітлено важливі аспекти вибору методів та моделей для

розпізнавання алфавітно-цифрової інформації, підкреслено роль попереднього навчання та тонкого налаштування для досягнення високих показників ефективності в завданнях розпізнаванні алфавітно-цифрової інформації.

# РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ ОТРИМАНИХ РЕЗУЛЬТАТІВ

У ході виконання практичної частини кваліфікаційної роботи магістра було реалізовано алгоритми попередньої обробки зображень, нормалізації рукописного тексту на зображенні, сегментування тексту на рядки, слова і літери, із застосуванням інструмента Tesseract OCR та OpenCV.

## 3.1 Етапи програмної реалізації

У цьому розділі автор описує основні етапи та архітектуру оптичного розпізнавання символів. Ці етапи включають попередню обробку, сегментацію, нормалізацію, подальше вилучення, класифікацію та постобробку. Для розробки ефективної програми, пов'язаної з оптичним розпізнаванням символів, ми повинні враховувати труднощі, які можуть виникнути на кожному етапі, щоб отримати високу швидкість розпізнавання символів.

### 3.1.1 Попередня обробка із застосуванням OpenCV

Попередня обробка зображення є ключовим етапом у визначенні якості результатів розпізнавання.

Цей процес включає в себе ряд технік, таких як фільтрація, бінаризація та корекція контрасту. Фільтрація допомагає видалити шум та випадкові елементи, що можуть перешкоджати подальшій обробці. Бінаризація перетворює зображення на чорно-біле, що полегшує виділення контурів та об'єктів. Корекція контрасту дозволяє зберегти деталі та чіткість зображення.

Для документів, які містять текст і графіку, цей етап особливо важливий. Бінарні або сірі зображення дозволяють зберігати необхідну інформацію, зменшуючи при цьому обчислювальні витрати. Попередня обробка вирішує завдання покращення чіткості тексту, усунення тіней та інших дефектів, що можуть виникнути під час зйомки або сканування документів.

Усі ці кроки попередньої обробки сприяють підвищенню точності систем розпізнавання символів та покращують якість фінальних результатів. Оптимізоване зображення стає більш доступним для подальшого розпізнавання тексту.

Ми можемо підвищити ефективність і простоту обробки зображення на наступних етапах (рис. 3.1)

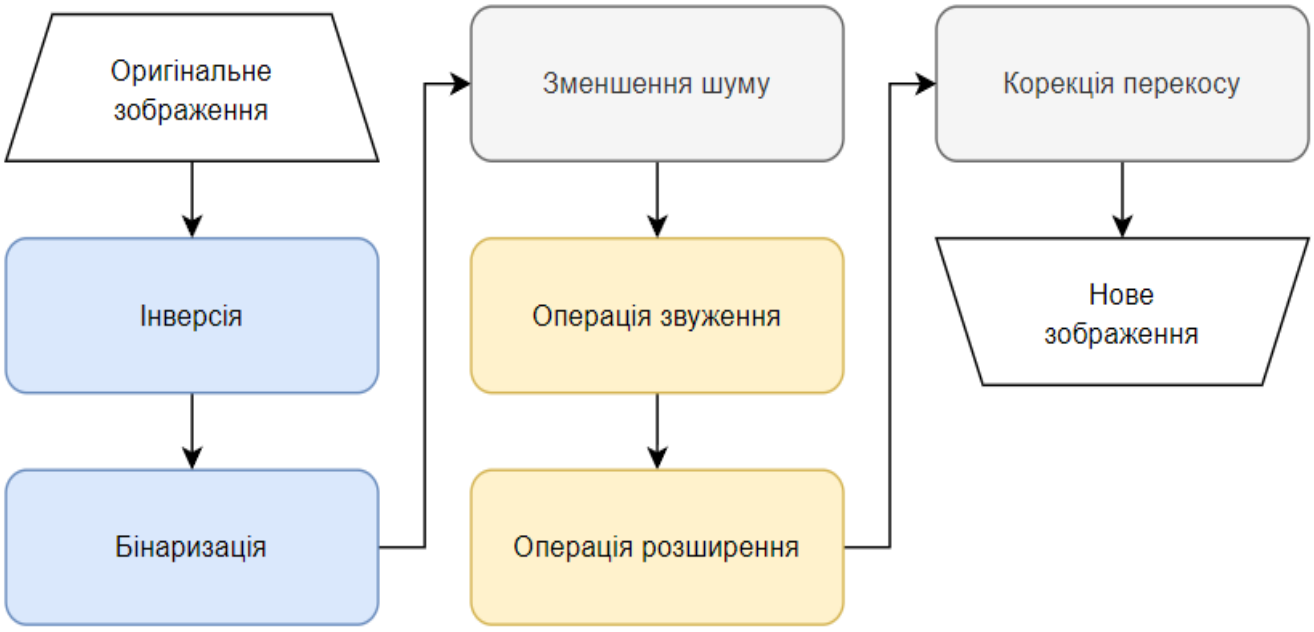


Рисунок 3.1 – Етапи попередньої обробки зображення

Таким чином, оскільки попередня обробка контролює придатність вхідних даних для наступних етапів, основним етапом, що стоїть перед етапом вилучення ознак, є етап попередньої обробки.

Більшість проблем, які ми перерахували в розділі «Проблеми розпізнавання текстів», потрібно вирішувати на етапі попередньої обробки.

Деякі операції, які ми можемо розглянути для виконання, можна перерахувати так: бінаризація, зменшення шуму, корекція перекосу, морфологічні операції, фільтрація, операції звуження, операції розширення.

Таблиця 3.1 надає короткий опис цих важливих процесів попередньої обробки. Вірно налаштована попередня обробка може значно поліпшити

точність та ефективність подальшого розпізнавання текстів у системах обробки інформації.

Таблиця 3.1

Опис процесів обробки зображення

| **Процес** | **Опис** |
|---|---|
| Бінаризація | Конвертація зображення в двійкове (чорне-біле), при цьому текст стає білим, а фон - чорним. Це полегшує подальший аналіз та розпізнавання тексту. |
| Зменшення шуму | Видалення непотрібних плям, випадкових пікселів або дрібних деталей, |
| Корекція перекосу | Виправлення скошеного положення тексту для забезпечення вертикального або горизонтального вирівнювання. |
| Морфологічні операції | Використання морфологічних операцій, таких як розширення, стискання, відкриття та закриття, для покращення форми та структури об'єктів на зображенні. |
| Корекція спотворень та іскажень | Корекція геометричних іскажень, таких як розтягнення, скручування, може бути важливою для точного OCR, особливо у випадках, коли зображення було захоплено під кутом або зі спотворенням |

Тепер розглянемо кожен з етапів та методи їх впровадження за допомогою OpenCV. Початкове зображення представлено на рисунку 3.2.

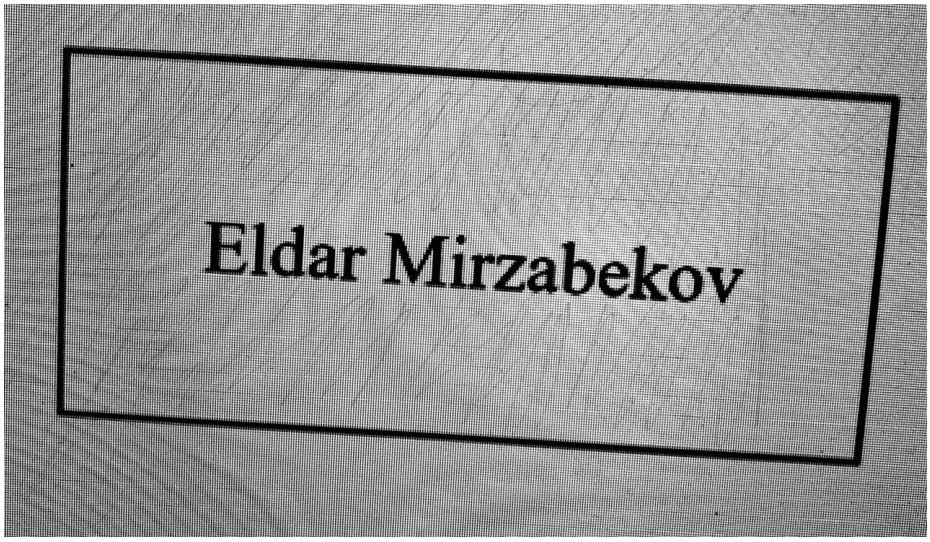


Рисунок 3.2 – Початкове зображення

Бінаризація.

Оскільки більшість алгоритмів розпізнавання потребують двоколірних зображень, наш перший крок передбачає перетворення кольорових або напівтонових зображень на чорно-білі аналоги - процес, який зазвичай називають «бінаризацією». Приклад зображено на рисунку 3.3.

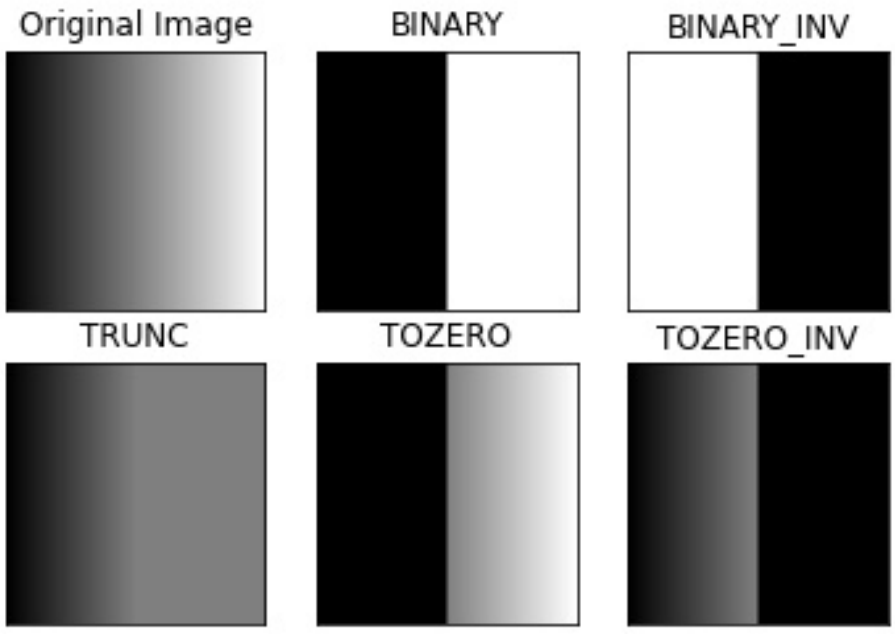


Рисунок 3.3 – Приклад бінаризації

Порівняння, зображене на рисунку 3.4, ілюструє відмінності між оригінальним зображенням і його чорно-білим відтворенням. Після перетворення початкового RGB-зображення у відтінки сірого процес бінаризації передбачає вибір порогового значення.

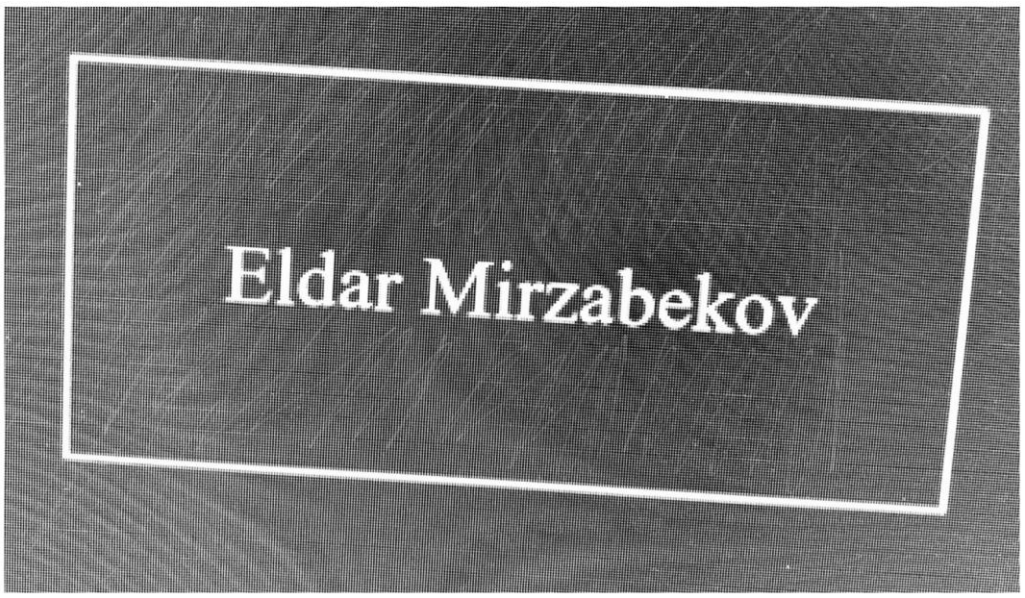


Рисунок 3.4 – Інвертоване та бінаризоване ісходне зображення

Зменшення шуму. Шум прийнято вважати випадковою величиною з нульовим середнім значенням. Розглянемо зашумлений піксель,

$$p = p_0 + n \tag{3.1}$$

де $p_0$ – істинне значення пікселя

$m$ – шум у цьому пікселі

Можливо взяти велику кількість однакових пікселів (скажімо, N) з різних зображень і обчислити їх середнє значення. В ідеалі, повинно отримати $p = p_0$, оскільки середнє значення шуму дорівнює нулю.

Тому ідея проста: потрібен набір схожих зображень для усереднення шуму. Розглянемо невелике вікно (скажімо, 5x5) на зображенні. Існує велика

ймовірність того, що така сама ділянка може бути десь в іншому місці зображення. Іноді в невеликій області навколо нього. [12]

OpenCV надає чотири варіації цього методу, вони представлені у таблиці 3.2. [12]

Таблиця 3.2

Різновиди деноїзації зображень у OpenCV

| **Назва методу** | **Опис** |
|---|---|
| cv.fastNlMeansDenoising() | Працює з одним зображенням у відтінках сірого |
| cv.fastNlMeansDenoisingColored() | Працює з кольоровим зображенням |
| cv.fastNlMeansDenoisingMulti() | Працює з послідовністю зображень, знятих за короткий проміжок часу (зображення у відтінках сірого) |
| cv.fastNlMeansDenoisingColoredMulti() | Те саме, що й fastNlMeansDenoisingMulti, але для кольорових зображень. |

Як згадувалося вище метод cv.fastNlMeansDenoising(), використовується для видалення шуму з кольорових зображень. (Передбачається, що шум буде гауссовим).

На рисунку 3.5 наведено збільшену версію результату. Вхідне зображення має гаусівський шум.

Тепер розглянемо, як власноруч можна реалізувати аналогічний метод для видалення гаусівського шуму з кольорових зображень. Для цього можна скористатися фільтрацією за Гауссом і використати власні розрахунки середнього значення для кожного пікселя на основі його сусідів.

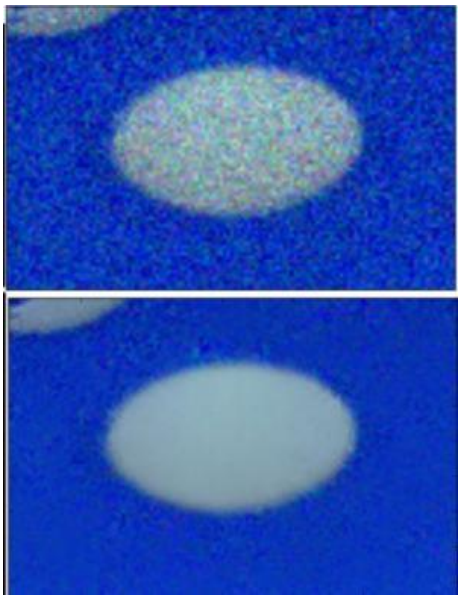

Рисунок 3.5 – Видалення шуму із застосуванням методу fastNlMeansDenoising

Проста реалізація зображена на рисунку 3.6 (з урахуванням того, що слід працювати з кольоровими зображеннями, тобто тривимірними масивами пікселів):

```
import cv2

def customDenoising(image, kernel_size=5, sigma=1.5):
    denoised_image = cv2.GaussianBlur(image, (kernel_size, kernel_size), sigma)

    return denoised_image

input_image = cv2.imread("input_image.jpg")

result_image = customDenoising(input_image)

cv2.imshow("Original Image", input_image)
cv2.imshow("Denoised Image", result_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Рисунок 3.6 – Проста реалізація видалення шуму з зображення

Результат виконання функції customDenoising на бінаризованому зображенні представлен на рисунку 3.7. Можна побачити що функція customDenoising успішно впоралася із завданням усунення шум. Порівняно з початковим станом, на рисунку видно, що шум та непотрібні деталі вдало

видалено, що призвело до покращення чіткості та виділення основних контурів об'єктів.

Зменшення шуму на таких зображеннях може поліпшити результати подальших аналізів, таких як визначення розмірів об'єктів, їх положення чи класифікація.

Важливо зазначити, що ефективність функції customDenoising може бути залежною від конкретного типу шуму та особливостей бінаризованого зображення. Доцільно провести аналіз властивостей шуму та оптимізацію параметрів функції для досягнення оптимальних результатів в конкретному випадку.


Рисунок 3.7 – Зображення після виконання функції зменшення шуму

Корекція перекосу.

Для вирішення поставленої задачі – корекції нахилу текстового блоку на зображенні – можна використовувати ряд алгоритмів обробки зображень.

Нижче розглянемо кожен з крокі:

а) Виявлення блоку тексту на зображенні;

б) Обчислення кута повороту тексту;

в) Поворот зображення для виправлення нахилу.

Маючи зображення зроблене на попередньому кроці, ми можемо обчислити мінімальний обернений обмежуючий прямокутник, який містить текстові області. Код представлен на рисунку 3.8.

```
16  coords = np.column_stack(np.where(thresh > 0))
17  angle = cv2.minAreaRect(coords)[-1]
18
19  if angle < -45:
20      angle = -(90 + angle)
21
22  else:
23      angle = -angle
24
```

Рисунок 3.8 – Визначення кута нахилу для області зображення

Рядок 17 знаходить усі x та y координати на растровому зображенні, які є частиною переднього плану.

Ми передаємо ці координати у функцію cv2.minAreaRect, яка обчислює мінімально повернутий прямокутник, що містить всю область тексту.

Функція cv2.minAreaRect повертає значення кута в діапазоні [-90, 0). При обертанні прямокутника за годинниковою стрілкою значення кута збільшується до нуля. Після досягнення нуля кут знову встановлюється на -90 градусів і процес продовжується.

Рядки 19 та 20 обробляють, якщо кут менше -45 градусів, в цьому випадку нам потрібно додати 90 градусів до кута і взяти обернену величину.

В іншому випадку, рядки 22 і 23 просто беруть обернену величину кута.

Тепер, коли ми визначили кут нахилу тексту, нам потрібно застосувати Афінне перетворення, щоб виправити нахил:

```
25  (h, w) = image.shape[:2]
26  center = (w // 2, h // 2)
27  M = cv2.getRotationMatrix2D(center, angle, 1.0)
28  rotated = cv2.warpAffine(
29      image, M, (w, h), flags=cv2.INTER_CUBIC, borderMode=cv2.BORDER_REPLICATE
30  )
31
```

Рисунок 3.9 – Поворот зображення за обчисленим кутом

Рядки 25 і 26 визначають координати центру (x, y) зображення. Ми передаємо координати центру та кут повороту у функцію cv2.getRotationMatrix2D. Ця матриця обертання M потім використовується для виконання перетворення у рядку 28.

Результат корекції зображення представлен на рисунку 3.10.

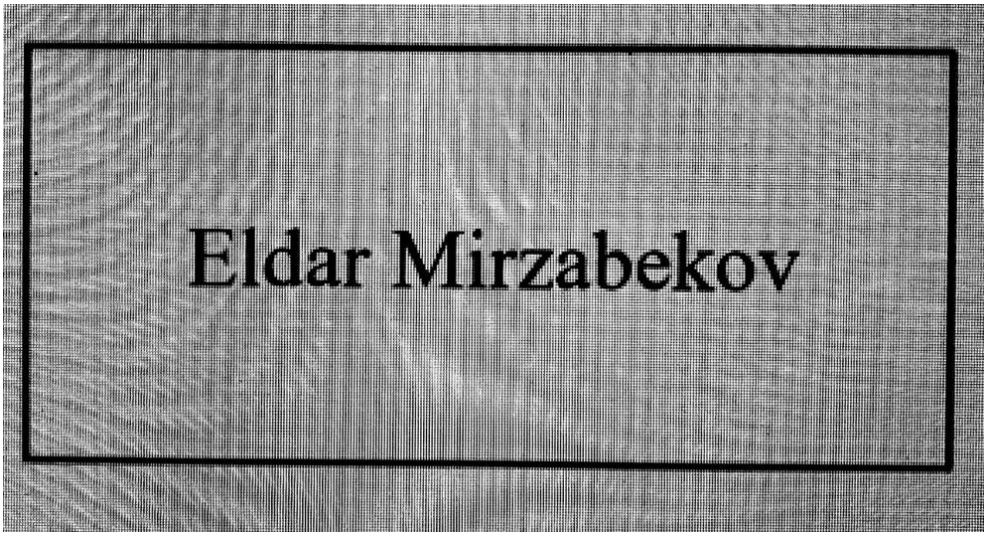


Рисунок 3.10 – Результат виконання функції корекції перекосу зображення

Морфологічні перетворення.

Морфологічні перетворення - це прості операції над формою зображення. Зазвичай вони виконуються над бінарними зображеннями. Вона потребує двох вхідних даних, один з яких є нашим вихідним зображенням, а другий називається структуроутворюючим елементом або ядром, який визначає характер операції. Основні морфологічні перетворення: Erode - розмивання (операція звуження) Dilate - розтягування (операція розширення) Розмиття (операція звуження) зображення з використанням фільтра (ядра)

один або кілька разів [13]. Ми розглянемо їх по черзі за допомогою наступного зображення (рис. 3.11):


Рисунок 3.11 – Ісходне зображення перед морфологічними перетвореннями

Морфологічні перетворення – Операція звуження.

Основна концепція ерозії така ж, як і ерозії ґрунту, за винятком того, що вона видаляє межі об'єкта на передньому плані (завжди намагайтеся тримати передній план білим). Далі, подібно до 2D-згортки, ядро рухається по зображенню. Якщо піксель на оригінальному зображенні (1 або 0) не розмитий (не перетворений на нуль), то він вважається 1, якщо всі пікселі під ядром мають значення 1.


Рисунок 3.12 – Результат виконання операції звуження

Відповідно, залежно від розміру ядра, всі пікселі, розташовані близько до межі, будуть відкинуті. Таким чином, товщина, розмір або просто кількість

білого простору об'єкта на передньому плані зображення зменшується (рис. 3.12). Як бачили у розділі «Простір кольорів», це корисно для усунення крихітних білих звуків, розділення двох пов'язаних об'єктів та інших речей.

Морфологічні перетворення – Операція розширення.

Операція розширення це протилежність ерозії. Піксельний елемент у цьому випадку дорівнює «1», якщо під ядром є хоча б один «1» піксель. Як результат, на зображенні більше білого кольору або об'єкт на передньому плані більший. Зазвичай розширення відбувається після ерозії в таких ситуаціях, як зменшення шуму. Тому що ерозія зменшує розмір нашого об'єкта, одночасно усуваючи білий шум. Таким чином, ми його збільшуємо (рис. 3.13). Оскільки шум зник, він більше не повернеться, але площа нашого об'єкта збільшиться. Це також корисно для з'єднання розірваних частин об'єкта.


Рисунок 3.13 – Результат виконання операції розширення

Код роботи представлено у Додатку А.

3.1.2 Розпізнавання тексту на обробленому зображенні з використанням Tesseract OCR.

Робочий процес розпізнавання складається з декількох послідовних кроків, які готують зображення до розпізнавання. Програмне забезпечення OCR використовує передові алгоритми для зчитування та аналізу зображення,

ідентифікації текстових рядків і символів та перетворення їх у машинний код. Цей машинний текст можна використовувати як вміст для пошуку.

Основою цього процесу є зображення, яке ми отримали на попередньому кроці.

Після завершення попередньої обробки програма розпізнавання використовує функцію вилучення ознак, щоб спробувати ідентифікувати та перетворити текст у цільовій області.

Це відбувається двома різними способами. Перший метод передбачає безпосереднє порівняння отриманих візуальних даних з інформацією, що вже зберігається в базі даних. Програма розпізнає і записує літеру «Р», відрізняючи її від інших літер, наприклад, «В», скануючи форми літер і знаходячи вектори, які відповідають шаблонам у її лексиконі - заздалегідь визначеній бібліотеці шрифтів, слів, речень тощо.

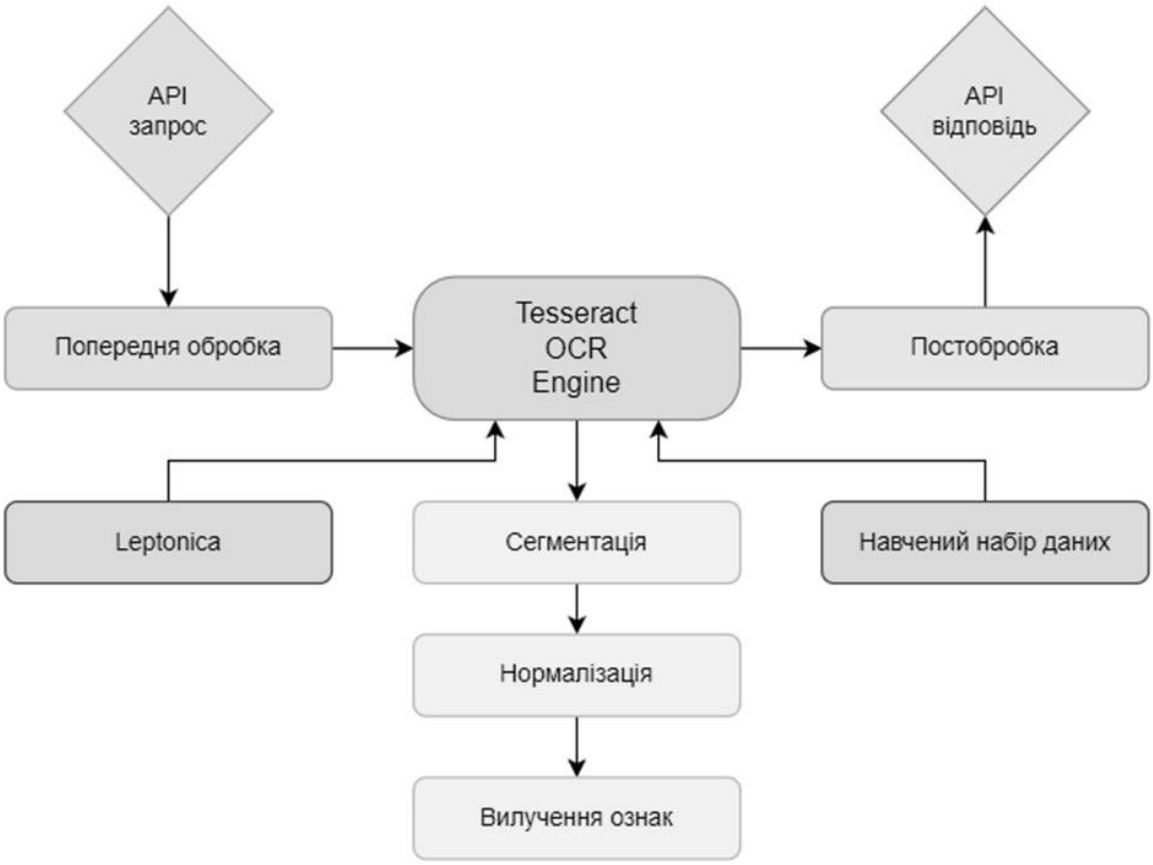


Рисунок 3.14 – Принцип роботи Tesseract OCR Engine

Цей підхід найбільш корисний для друкованого тексту, отриманого шляхом сканування документа або імпорту цифрового текстового файлу у форматі, який не сумісний з існуючими системами.

Другий метод передбачає більш просунуте розпізнавання символів за допомогою алгоритмів машинного навчання, які піднімають стандартне розпізнавання до рівня так званого інтелектуального розпізнавання символів, або ICR. Поєднуючи базове розпізнавання тексту з розширеним розпізнаванням образів, ICR може робити інтелектуальні висновки про те, які частини зображення є символами, словами та реченнями, і надавати їм значення на основі власного лексикону та ситуативного контексту.

Загальний принцип роботи оптичного розпізнавання символів (OCR) показано на рис. 3.14 вижче. Він включає в себе кілька процесів розпізнавання тексту. Сегментація, виділення ознак, розпізнавання і постобробка.

Фаза сегментації.

У комп'ютерному зорі сегментація - це процес поділу цифрового зображення на кілька сегментів (наборів пікселів, також відомих як суперпікселі). Метою сегментації є спрощення та/або зміна представлення зображення на більш змістовне і зручне для аналізу. Сегментація зображень зазвичай використовується для пошуку об'єктів і меж (ліній, кривих тощо) на зображеннях. Точніше кажучи, сегментація зображення - це процес присвоєння мітки кожному пікселю зображення таким чином, щоб пікселі з однаковими мітками мали певні візуальні характеристики.

Результатом сегментації зображення є набір сегментів, які в сукупності покривають все зображення, або набір контурів, виділених із зображення. Кожен з пікселів у регіоні схожий за певною характеристикою або обчислюваною властивістю, наприклад, кольором, інтенсивністю або текстурою. Сусідні області суттєво відрізняються за тією ж характеристикою (характеристиками). При застосуванні до стеку зображень, що є типовим для медичної візуалізації, отримані контури після сегментації зображень можна

використовувати для створення 3D-реконструкцій за допомогою інтерполяційних алгоритмів, таких як маршируючий куб.

Існує три категорії алгоритмів сегментації документів сегментації документів [5], а саме:

а) Методи «зверху-вниз»,

б) Методи «знизу-вгору»,

в) Гібридні методи.


Рисунок 3.15 – Результат сегментації тексту

Метод «зверху-вниз» рекурсивно ділить великі області на менші субрегіони. Процес сегментації документа завершиться, коли буде виконано певну умову, і тоді отримані діапазони представлятимуть остаточні результати сегментації. Однак методи «знизу вгору» починаються з пошуку пікселів, що становлять інтерес, а потім групують ці пікселі. Після цього вони маніпулюють цими пікселями, перетворюючи їх на взаємопов'язані літери, які потім об'єднуються у слова, рядки або текстові блоки. Гібридні підходи - це ті, що поєднують висхідні та низхідні методи.

Вважається, що речення в тексті організовані горизонтально і не накладаються одне на одне. Після цього сегментація є простою процедурою. Попередньо визначається середнє значення відстані між літерами слова. Потім

зображення розділяється на рядки шляхом ідентифікації всіх білих смуг. Далі, визначаючи білі смуги певної ширини, ці смуги розділяються на слова. Після цього виділені слова переходять на наступний етап і розділяються на літери. В результаті на виході модуля сегментації весь текст відображається у вигляді зображень його літер. (рис. 3.15)

Етап нормалізації. Перед розпізнаванням зображення нормалізується і приводиться до розмірів шаблонів, підготовлених заздалегідь.

В результаті сегментації отримують ізольовані символи, підготовлені для проходження етапу вилучення ознак; тому, залежно від використовуваних методів, ізольовані символи мінімізуються до певного розміру. Оскільки процедура сегментації перетворює зображення на матрицю $m * n$, вона є дуже важливою. Після цього ці матриці часто нормалізують, зменшуючи їх розмір і видаляючи будь-яку зайву інформацію з зображення, не пропускаючи жодної значущої інформації [12].

Нормалізація символів дозволяє зменшити кількість даних, що використовуються для класифікації символів. Інша мета нормалізації полягає в тому, щоб процес класифікації міг розпізнавати символи різного розміру.

Під час нормалізації кожен символ змінюється до розміру, який був заданий під час ініціалізації моделі (рис. 3.16). Всі подальші операції класифікатора будуть виконуватися над зміненими (нормалізованими) символами.


Рисунок 3.16 – Символи різного розміру до і після процесу нормалізації

Під час процесу нормалізації також важливо враховувати можливі артефакти або спотворення, які можуть виникнути в результаті сканування або інших етапів перед обробкою зображення. Такі аномалії можуть впливати на якість розпізнавання, тому необхідно встановити ефективні методи виявлення і корекції подібних спотворень. Забезпечення стабільності і чистоти даних на етапі нормалізації допомагає уникнути неправильностей у подальших етапах обробки.

Час навчання OCR-класифікатора збільшиться, якщо вибрати надто великий розмір нормалізації. І навпаки, якщо розмір замалий, значні деталі символів буде втрачено.

Вибраний розмір нормалізації повинен забезпечувати баланс між точністю розпізнавання та часом класифікації.

Розмір символу після нормалізації в ідеалі має бути схожим на його розмір до нормалізації, щоб досягти оптимального результату

Певні характеристики символів, такі як співвідношення сторін символу, будуть втрачені під час процесу нормалізації. Щоб компенсувати інформацію, втрачену під час процесу нормалізації, на етапі навчання може бути надано додаткову інформацію. Окрім того, важливо розглядати можливості автоматизації процесу навчання OCR-класифікатора з метою підвищення ефективності і зниження часу, необхідного для навчання моделі.

Етап вилучення ознак. Виділення ознак передбачає спрощення кількості ресурсів, необхідних для точного опису великого набору даних. При виконанні аналізу складних даних одна з головних проблем пов'язана з кількістю задіяних змінних. Аналіз з великою кількістю змінних, як правило, вимагає великого обсягу пам'яті та обчислювальної потужності або алгоритму класифікації, який надмірно відповідає навчальній вибірці і погано узагальнює нові вибірки. Виділення ознак - це загальний термін для методів побудови комбінацій змінних, які дозволяють обійти ці проблеми, при цьому описуючи дані з достатньою точністю. Методологія вилучення ознак, прийнята для розпізнавання символів, змогла забезпечити дуже хорошу точність для

широкого діапазону розмірів і стилів шрифтів. Максимальна отримана точність становить 97%. Основним фактором, що перешкоджає підвищенню точності, є схожість форм і особливостей окремих символів. Подальше вдосконалення системи можливе шляхом навчання OCR-рушія обробляти помилки, що найчастіше зустрічаються. Існує три типи таких помилок.

На цьому етапі виділяються різні ознаки символів. Ці ознаки однозначно ідентифікують символи. Вибір правильних ознак і загальної кількості ознак, які будуть використані, є важливим питанням дослідження. Можна використовувати різні типи ознак, такі як саме зображення, геометричні ознаки (петлі, штрихи) та статистичні ознаки (моменти). Нарешті, різні методи, такі як аналіз головних компонент, можуть бути використані для зменшення розмірності зображення.

Існує два основні методи виділення ознак в розпізнаванні текстів:

а) Алгоритм виявлення ознак визначає символ, оцінюючи його лінії та штрихи.

б) Розпізнавання працює шляхом ідентифікації всього символу.

Далі ми перетворюємо зображення символу у двійкову матрицю, де білі пікселі - це нулі, а чорні – одиниці (рис. 3.17)

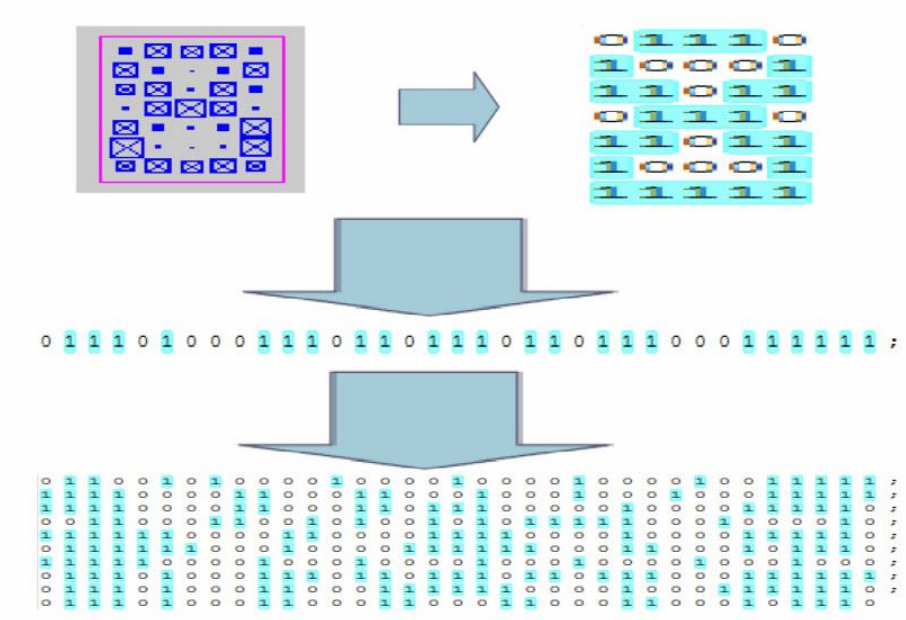

Рисунок 3.17 – Перетворення зображення символу у двійкову матрицю

Ми можемо розпізнати рядок тексту, шукаючи рядки білих пікселів, між якими є чорні пікселі. Аналогічно, ми можемо розпізнати, де починається і де закінчується символ.

Відстань між найвіддаленішою одиницею та центром матриці можна визначити за формулою відстані:

$$d = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2} \quad (3.2)$$

де d – відстань між двома точками у двовимірному просторі

$x_1, y_1$ – координати центру матриці

$x_2, y_2$ – координати найвіддаленішої точки зі значенням 1 в матриці

Етап класифікації. Системи розпізнавання текстів широко використовують методологію розпізнавання образів, яка відносить кожен приклад до попередньо визначеного класу. Класифікація - це процедура розподілу вхідних даних відповідно до виявленої інформації на класи порівняння з метою створення груп з однорідними якостями, одночасно розділяючи різні вхідні дані на різні класи. Класифікація здійснюється на основі виділення ознак. Можна сказати, що класифікація ізолює простір ознак на кілька класів з урахуванням правила прийняття рішення. Вибір класифікатора залежить від кількох факторів, таких як кількість вільних параметрів, наявна навчальна вибірка тощо. Вченими досліджуються різні процедури для розпізнавання текстів.

Процедура віднесення символу до відповідної категорії - це те, як він визначається. Кореляції, виявлені в компонентах зображення, слугують основою для структурного підходу до класифікації. Статистичні методи ґрунтуються на класифікації зображення за допомогою дискримінантної функції. Байєсівський класифікатор, класифікатор дерева рішень,

нейромережевий класифікатор, класифікатор найближчого оточення та інші підходи є прикладами статистичних методів класифікації [7]. Останнім типом класифікаторів є синтаксичний підхід, який передбачає граматичний метод складання зображення з його складових частин.

Методи класифікації OCR можна класифікувати як статистичні методи, нейронні мережі, зіставлення з шаблоном, алгоритми машин опорних векторів (SVM) та комбінація класифікаторів. Вони представлені на таблиці 3.3.

Таблиця 3.3

Техніки класифікації OCR

| **Категорія** | **Опис** |
|---|---|
| Статистичні техніки | Використовує ймовірнісні моделі, такі як приховані марківські моделі та методи байєсівського аналізу для класифікації OCR. |
| Нейронні мережі | Використовує зокрема згорткові та рекурентні нейронні мережі для ієрархічного вивчення рис. |
| Шаблонне відповідання | Зіставлення за шаблоном передбачає порівняння частин зображення із заздалегідь визначеними шаблонами або зразками. Цей метод ґрунтується на схожості між вхідним зображенням і набором еталонних шаблонів. |
| Support Vector Machine (SVM) | Використовує алгоритми SVM для пошуку гіперплощини для ефективної класифікації символів чи слів. |
| Комбінація класифікаторів | Інтегрує виходи кількох класифікаторів, покращуючи точність системи OCR за допомогою методів, таких як колективне навчання. |

Використовуючи сегментоване зображення. На етапі класифікації методом *pytesseract.image_to_data* було отримано набір даних (таблиця 3.4). Отримані дані містять інформацію про розпізнані текстові області, їхні розміри, положення та інші атрибути. Цей набір даних може бути використаний для подальшого аналізу та обробки текстової інформації на зображеннях, сприяючи вдосконаленню результатів класифікації та подальших завдань обробки даних.

Таблиця 3.4

Отриманий набір даних в результаті класифікації

| **line_num** | **word_num** | **left** | **top** | **width** | **height** | **conf** | **text** |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 510 | 250 | -1 | – |
| 0 | 0 | 122 | 112 | 269 | 26 | -1 | – |
| 0 | 0 | 122 | 112 | 269 | 26 | -1 | – |
| 1 | 0 | 122 | 112 | 269 | 26 | -1 | – |
| 1 | 1 | 122 | 112 | 80 | 26 | 93,28 | Eldar |
| 1 | 2 | 212 | 112 | 179 | 26 | 91,09 | Mirzabekov |
| 0 | 0 | 11 | 10 | 492 | 231 | -1 | – |
| 0 | 0 | 11 | 10 | 492 | 231 | -1 | – |
| 1 | 0 | 11 | 10 | 492 | 231 | -1 | – |
| 1 | 1 | 11 | 10 | 492 | 231 | 95 | – |

Етап постобробки. Етап постобробки включав в себе використання бібліотек OpenCV та Pillow для подальшої обробки зображення. Після використання OpenCV для опрацювання зображення була також використана бібліотека Pillow для виділення текстової інформації.

Дані отримані на етапі класифікації були використані для позначення прямокутників (rectangles) на зображенні, що визначають межі текстових областей. Також було виведено слова, які були розпізнані, під кожним відповідним прямокутником. (рис. 3.18).


Рисунок 3.18 – Результат постобробки зображення

Під час написання програми автор також зіткнувся з проблемою відображення українських літер в OpenCV. Це було пов'язано з кодуванням шрифтів, яке використовує бібліотека.

Для розв'язання проблеми відображення українських літер в OpenCV було обрано шрифт, який підтримує українські символи, такий як Arial Unicode MS. Було також встановлено необхідні бібліотеки, включно з Pillow, для обробки тексту та зображень.

Створене зображення за допомогою OpenCV було опрацьовано бібліотекою Pillow, де було використано вибраний шрифт для рендерингу українського тексту. Задавши позицію і колір тексту, а також переконавшись у підтримці обраним шрифтом українських символів, було успішно додано текст на зображення. Завершальним етапом стало відображення отриманого зображення за допомогою OpenCV. При цьому було важливо вказати правильний шлях до шрифту і упевнитися в наявності обраного шрифту в системі.

Після відображення результатів в окремому вікну, програма також виводить знайдений текст у консолі (рис. 3.19)

```
proge@mirzabekov MINGW64 /e/mirzabekov/projects/ocr_python_textbook (main)
$ C:/Users/proge/AppData/Local/Programs/Python/Python311/python.exe e:/mirzabekov/pro
jects/ocr_python_textbook/pytesseract_usage.py
Text result:  Eldar Mirzabekov
```

Рисунок 3.19 – Відображення результатів у консолі

На цьому етапі програма завершує своє виконання. Зобразивши результати обробки тексту на зображенні, програма зберігає остаточний варіант у папку «result» під назвою «output.png». Це дозволяє зручно відстежувати та зберігати візуалізовані результати обробки тексту для подальшого використання чи аналізу.

Код роботи представлено у Додатку Б.

## 3.2 Аналіз отриманих результатів системи розпізнавання алфавітно-цифрової інформації

В даному розділі проводиться аналіз ефективності розробленої системи розпізнавання з використанням ключових метрик – влучності (Precision) та повноти (Recall). У розпізнаванні образів, пошуку інформації, виявленні та класифікації об'єктів (машинному навчанні) влучність та повнота – це показники ефективності, які застосовуються до даних, отриманих з колекції, корпусу або простору зразків.

Ці метрики є фундаментальними вимірювальними інструментами, які дозволяють не лише кількісно оцінити рівень розпізнавання символів, а й здійснити більш глибокий аналіз його якості.

Влучність (Precision).

Влучність визначає, яка частина розпізнаних символів є правильними в порівнянні з усіма визначеними системою. Наприклад, для текстового пошуку на множині документів, влучність є числом правильних результатів, поділеним на число всіх повернених результатів.

Влучність бере до уваги всі знайдені документи, але її також можливо оцінювати на заданому рівні відсікання, враховуючи лише розташовані найвище результати, що повертає система. Таку міру називають «N-влучністю».

Влучність використовують разом із повнотою, відсотком всіх релевантних документів, який повертає пошук. Ці дві міри іноді

використовують разом в оцінці F1 (або F-мірі), щоби забезпечити єдине вимірювання для системи. [14]

Формула влучності виглядає наступним чином:

$$\text{Влучність } (Precision) = \frac{\text{Кількість розпознаних символів}}{\text{Загальна кількість символів}} \quad (3.3)$$

Повнота (Recall).

В інформаційному пошуку повнота є часткою релевантних документів, яку вдається успішно знайти.

Зосереджуючись на аспекті повноти, ми ставимо перед собою завдання визначити, наскільки ефективно система виявляє всі наявні символи на зображенні. Підвищення значення повноти вказує на здатність системи ефективно виявляти всі існуючі символи під час розпізнавання. [14]

Наприклад, для текстового пошуку на множині документів, повнота є числом правильних результатів, поділеним на число результатів, які мало би бути повернуто.

Влучність і повнота не є особливо корисними показниками, якщо їх використовувати ізольовано. Наприклад, можна мати ідеальну повноту, просто відновивши кожен елемент. Так само можна отримати майже ідеальну влучність, вибравши лише малу кількість дуже ймовірних елементів.

Повнота визначає, яка частина усіх існуючих символів була коректно розпізнана системою. Формула повноти має наступний вигляд:

$$\text{Повнота } (Recall) = \frac{\text{Кількість розпізнаних символів}}{\text{Загальна кількість символів}} \quad (3.4)$$

Влучність і повноту також можливо інтерпретувати не як відношення, а як оцінки ймовірностей:

а) Влучність є оцінкою ймовірності того, що документ, випадково вибраний з пулу знайдених документів, є релевантним;

б) Повнота є оцінкою ймовірності того, що документ, випадково вибраний з пулу релевантних документів, буде знайдено.

Іншою інтерпретацією є те, що влучність є усередненою ймовірністю релевантного знаходження, а повнота є усередненою ймовірністю повного знаходження, усереднені над багатократними запитами пошуку. [14]

Для задач класифікації, терміни істинно позитивні, істинно негативні, хибно позитивні та хибно негативні є порівняннями результатів тестованого класифікатора з надійними зовнішніми судженнями. Терміни позитивні та негативні стосуються передбачень класифікатора (які іноді називають очікуванням), а терміни істинно та хибно стосуються того, чи це передбачення відповідає зовнішньому судженню (іноді відомому як спостереження). [14]

### 3.2.1 Аналіз результатів розпізнавання української мови методами нейронних мереж

Для аналізу результатів візьмемо навчену українській мові даними tessdata модель Tesseract. Для того, щоб модель розпізнала текст вірно, вкажемо їй параметр мови (рис. 3.20). Значення параметра має дорівнювати мові, якої навчена модель.

```
language = "ukr"

boxes = pytesseract.image_to_data(input_image, lang=language)
```

Рисунок 3.20 – Заданий параметр мови для розпізнавання

Для прикладу, візьмемо вихідне зображення з українською мовою (рис. 3.21). Та запустимо процес попередньої обробки зображення OpenCV.

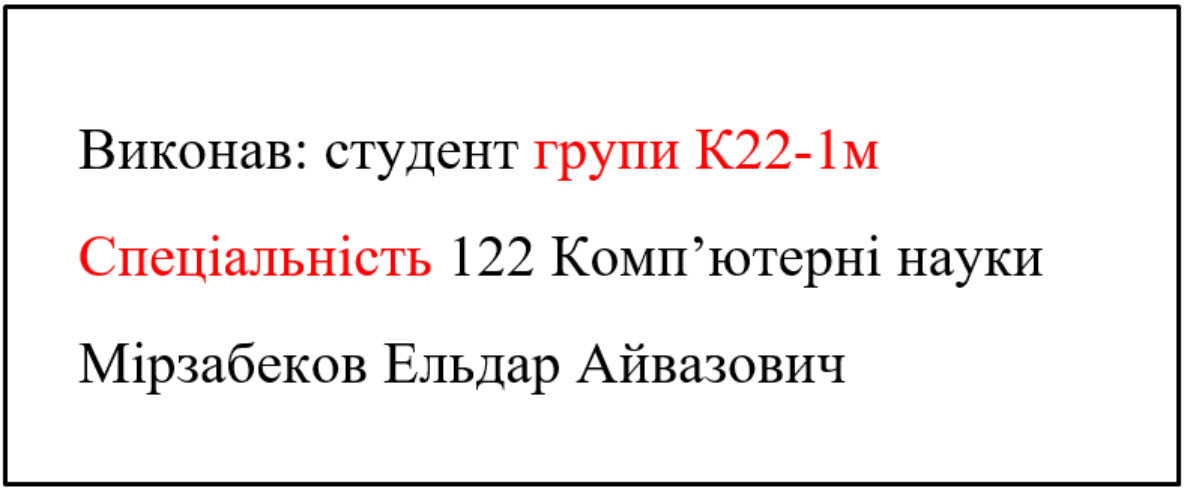


Рисунок 3.21 – Вихідне зображення з українською мовою

Після процесу попередньої обробки зображення OpenCV отримуємо зображення (рис. 3.22). Тепер ми можемо передати оброблене зображення моделі Tesseract для розпізнавання тексту. можна запускати процес розпізнавання тексту з вже встановленим параметром language.

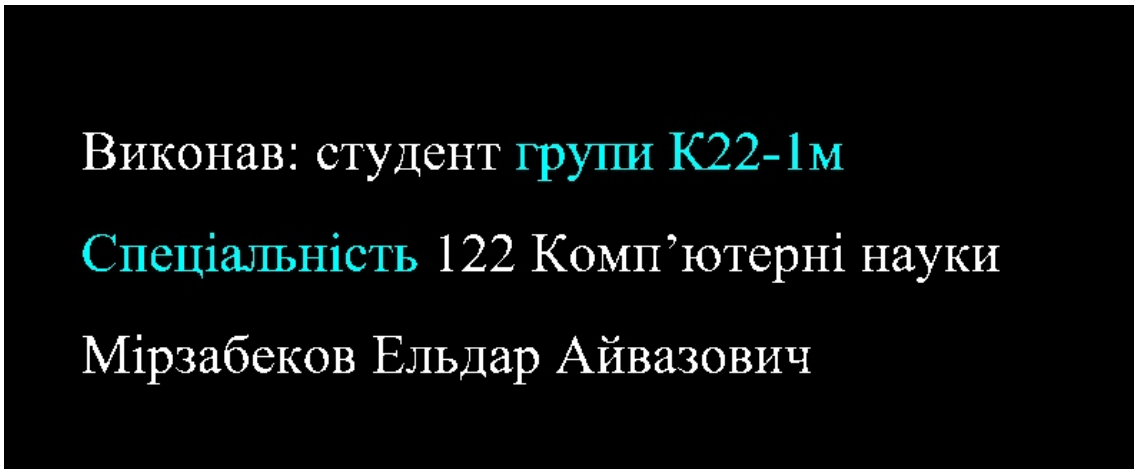


Рисунок 3.22 – Результат попередньої обробки зображення з українською мовою

Після процесу розпізнавання тексту отримуємо зображення, збережене у файлі output.png (рис. 3.23).

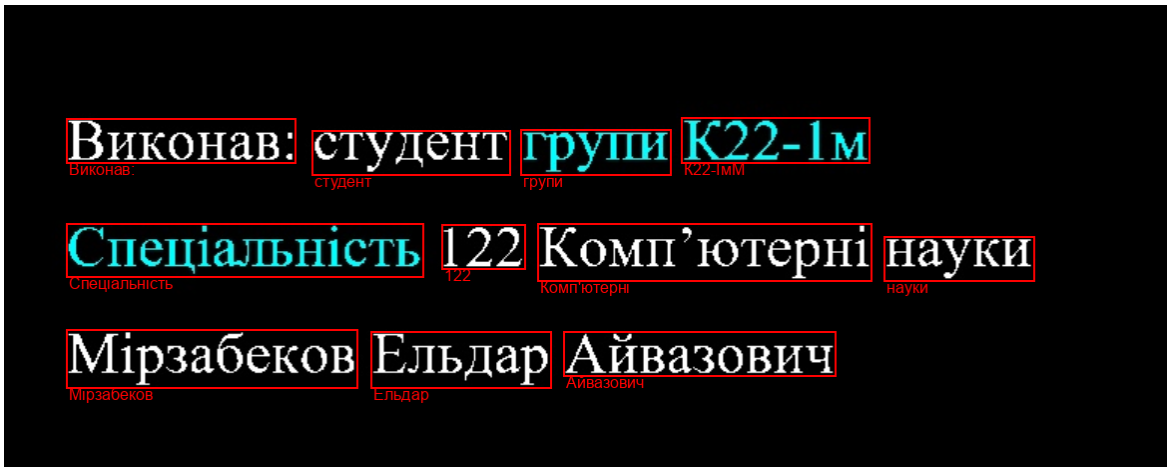


Рисунок 3.23 – Результат розпізнавання тексту з українською мовою

Результат розпізнавання показав, що майже всі слова української мови, за винятком одного, були розпізнані правильно. Аналіз влучності та повноти представлен у наступному підрозділі.

### 3.2.1 Аналіз влучності та повноти перед обробкою зображення

У цьому розділі буде проведений аналіз ефективності системи розпізнавання за допомогою ключових метрик - влучності та повноти. Для кращого розуміння результатів наведемо таблиці, які демонструють важливі характеристики системи перед та після обробки зображень.

Таблиця 3.5 відображає початковий стан результатів розпізнавання, перед застосуванням будь-яких методів обробки зображень. Загальна ефективність розпізнавання до будь-яких оптимізацій. Оцінка може включати помилкові класифікації та характеристики розпізнаних символів.

Пояснення щодо результатів таблиць:

а) Істинно позитивний (TP) – кількість правильно розпізнаних символів;

б) Хибно позитивний (FP) – кількість символів, які помилково визнані присутніми;

в) Хибно негативний (FN) – кількість символів, які були пропущені та не визнані.

Таблиця 3.5

Класифікація результатів розпізнавання перед обробкою зображення

| Символ | Істинно позитивний (TP) | Хибно позитивний (FP) | Хибно негативний (FN) | Всього |
|---|---|---|---|---|
| **А** | 45 | 2 | 410 | 457 |
| **Б** | 34 | 4 | 54 | 92 |
| **В** | 43 | 2 | 61 | 106 |
| **0** | 23 | 3 | 40 | 66 |
| **1** | 22 | 5 | 174 | 201 |
| **2** | 13 | 1 | 80 | 94 |
| **3** | 8 | 7 | 93 | 108 |
| **4** | 15 | 1 | 100 | 116 |
| **5** | 35 | 5 | 230 | 270 |
| **6** | 10 | 3 | 85 | 98 |
| **7** | 11 | 4 | 98 | 113 |
| **8** | 3 | 0 | 73 | 73 |
| **9** | 9 | 1 | 70 | 80 |

У таблиці 3.6 подані влучність та повнота для кожного окремого символу перед будь-якою обробкою зображень. Це дозволяє детально розглянути результати для кожного символу окремо та визначити, чи є які-небудь особливості чи виклики в розпізнаванні конкретних символів.

Таблиця 3.6

Влучність та повнота для кожного символу перед обробкою зображення.

| Символ | Влучність $\frac{TP}{TP + FP}$ | Повнота $\frac{TP}{TP + FN}$ |
|---|---|---|
| **А** | 0,96 | 0,10 |
| **Б** | 0,89 | 0,39 |
| **В** | 0,96 | 0,41 |
| **0** | 0,88 | 0,37 |
| **1** | 0,81 | 0,11 |
| **2** | 0,93 | 0,14 |
| **3** | 0,53 | 0,08 |
| **4** | 0,94 | 0,13 |
| **5** | 0,88 | 0,13 |

Продовження табл 3.6

| | | |
|---|---|---|
| **6** | 0,77 | 0,11 |
| **7** | 0,73 | 0,10 |
| **8** | 1,00 | 0,04 |
| **9** | 0,90 | 0,11 |
| **Всього** | ≈ 0,86 | ≈ 0,17 |

### 3.2.2 Аналіз влучності та повноти після обробки зображення

Враховуючи, що обробка зображення має суттєвий вплив на результати розпізнавання, введення етапів оптимізації, виявляється ключовим етапом у вдосконаленні функціональності системи. Зазначені в таблиці 3.7 покращення не лише підвищують точність, але і роблять систему більш стійкою до помилок та непередбачених викликів, забезпечуючи високий рівень надійності у розпізнаванні символів.

Таблиця 3.7

Класифікація результатів розпізнавання після обробки зображення

| **Символ** | **Істинно позитивний (TP)** | **Хибно позитивний (FP)** | **Хибно негативний (FN)** | **Всього** |
|---|---|---|---|---|
| **А** | 280 | 3 | 143 | 426 |
| **Б** | 50 | 4 | 34 | 88 |
| **В** | 43 | 6 | 18 | 67 |
| **0** | 42 | 1 | 49 | 92 |
| **1** | 144 | 14 | 41 | 199 |
| **2** | 84 | 6 | 34 | 124 |
| **3** | 83 | 4 | 36 | 123 |
| **4** | 91 | 15 | 29 | 135 |
| **5** | 204 | 5 | 72 | 281 |
| **6** | 53 | 3 | 26 | 82 |
| **7** | 85 | 4 | 26 | 115 |
| **8** | 40 | 0 | 21 | 61 |
| **9** | 69 | 12 | 14 | 95 |

Результати чітко свідчать про значущі покращення у якості розпізнавання після застосування методів обробки зображень, зокрема, використання OpenCV. Наприклад, для цифри «5» до проведення обробки

зображення система розпізнавала її в 35 із 270 випадків, а після оптимізації цей показник піднявся до 204 із 281 випадку. Це демонструє вражаючий приріст точності та ефективності розпізнавання конкретних символів.

У таблиці 3.8 наведено результати аналізу влучності та повноти для кожного окремого символу після застосування передових методів обробки зображень. Ці результати відображають значущий прогрес у точності та повноті розпізнавання символів, що грає важливу роль у покращенні ефективності системи.

Таблиця 3.8

Влучність та повнота для кожного символу після обробки зображення.

| **Символ** | **Влучність** $\frac{TP}{TP + FP}$ | **Повнота** $\frac{TP}{TP + FN}$ |
|---|---|---|
| **А** | 0,99 | 0,66 |
| **Б** | 0,93 | 0,60 |
| **В** | 0,88 | 0,70 |
| **0** | 0,98 | 0,46 |
| **1** | 0,91 | 0,78 |
| **2** | 0,93 | 0,71 |
| **3** | 0,95 | 0,70 |
| **4** | 0,86 | 0,76 |
| **5** | 0,98 | 0,74 |
| **6** | 0,95 | 0,67 |
| **7** | 0,96 | 0,77 |
| **8** | 1,00 | 0,66 |
| **9** | 0,85 | 0,83 |
| **Всього** | ≈ 0,94 | ≈ 0,69 |

Таблиця 3.8 надає детальний огляд покращень, отриманих для кожного символу. Наприклад, аналіз показав, що для цифри «3» влучність та повнота піднялися від 53% до 95% після впровадження методів обробки зображень. Це свідчить про значущий успіх у визначенні та розпізнаванні цього конкретного символу.

Високі показники влучності та повноти для різних символів підтверджують ефективність оптимізацій, впроваджених у систему розпізнавання. Наочно виявлені поліпшення свідчать про те, що розроблені методи обробки зображень сприяють не лише загальній точності, а й конкретним аспектам роботи системи, зокрема, у визначенні окремих символів.

## 3.3 Висновки

У ході виконання цього розділу було розроблено та описано етап попередньої обробки зображення а також етап розпізнавання тексту.

Для реалізації етапу попередньої обробки зображення був застосован ряд методів та алгоритмів OpenCV для оптимізації зображення перед передачею його до розпізнавання тексту. Етап розпізнавання тексту було виконано у декілька фаз.

Проведено оцінку влучності та повноти системи розпізнавання. Оцінка вказує на значний прогрес, особливо після застосування методів попередньої обробки. Це свідчить про ефективність обраного підходу. Необхідно також визначити можливості для подальших вдосконалень, роблячи акцент на оптимізації та покращеннях для досягнення ще вищої точності та продуктивності системи в майбутньому.

# ВИСНОВКИ

У ході виконання дипломної роботи були розглянуті та досліджені основні принципи та методи систем оптичного розпізнавання алфавітно-цифрової інформації. Було проаналізовано процеси зчитування та обробки зображень, переведення їх в градації сірого або чорно-білого, сегментації тексту та розпізнавання символів. Виокремлені та описані шаблонний, ознаковий та структурний методи розпізнавання.

У роботі розглянуті різні методи інтелектуального аналізу даних для розпізнавання алфавітно-цифрової інформації. Зазначено важливість використання методів машинного навчання, таких як нейронні мережі та архітектура Transformer, для ефективного розпізнавання алфавітно-цифрової інформації.

Обговорено застосування моделі Donut (Document understanding transformer) та висвітлено важливі аспекти вибору методів та моделей для розпізнавання алфавітно-цифрової інформації, підкреслено роль попереднього навчання та тонкого налаштування для досягнення високих показників ефективності в завданнях розпізнаванні алфавітно-цифрової інформації.

Окрема увага приділена аналізу сучасних методів ідентифікації алфавітно-цифрової інформації, а також програм розпізнавання. З'ясовано роль та застосування машинного навчання штучного інтелекту у розпізнаванні, розглянуто конкретні застосування цих методів.

Проведено дослідження засобів реалізації системи розпізнавання. Виокремлені проблеми, що виникають при роботі систем розпізнавання, вибрані та обґрунтовані інструментальні засоби для реалізації системи. Описана програмна реалізація системи з використанням інструментальних засобів, таких як Python, Tesseract OCR та OpenCV. Описані кроки попередньої обробки зображення, такі як бінаризація, морфологічна стандартизація та корекція шуму.

Проведено аналіз отриманих результатів, включаючи етап попередньої обробки та розпізнавання тексту. Виявлені та проаналізовані показники влучності та повноти до і після обробки зображення показали, що розроблена система розпізнавання алфавітно-цифрової інформації виявляє високу ефективність у виконанні своєї основної функції. На етапі попередньої обробки зображення за допомогою OpenCV вдалося досягти високого рівня точності та чіткості відображення символів. Це сприяло підвищенню влучності системи, забезпечуючи правильне розпізнавання алфавітно-цифрової інформації на початковому етапі обробки. Після застосування Tesseract OCR та моделі Donut для розпізнавання тексту на обробленому зображенні також спостерігалося значне покращення в показниках влучності та повноти. Алгоритм розпізнавання виявив високу здатність правильно ідентифікувати символи, навіть у складних умовах зображень.

Загальні висновки дозволяють стверджувати, що система розпізнавання алфавітно-цифрової інформації, реалізована з використанням обраного набору інструментальних засобів та методів, відзначається високою ефективністю та точністю. Розроблені та досліджені алгоритми можуть мати практичне застосування у сферах, де важлива автоматизована обробка та розпізнавання алфавітно-цифрової інформації.

Отримані результати є важливим внеском у розвиток та вдосконалення систем розпізнавання та машинного навчання штучного інтелекту. Робота виокремлюється актуальністю та практичністю своєї тематики, а отримані в ході дослідження знання можуть слугувати основою для подальших наукових досліджень та розробок у цьому напрямку.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. А. О. Олійник, С. О. Субботін, О. О. Олійник. Інтелектуальний аналіз даних: навч. посіб. Запоріжжя, 2012. 278 с.
2. Zhang X, Sun Y. Brand Name: An Intelligent Mobile-based Environmental Protection Rating and Suggestion Platform using Artificial Intelligence and Text Recognition, 2022.
3. Michael A. Nielsen. Neural Networks and Deep Learning. Determination Press, 2015.
4. В.О. Козел. Методи та етапи автоматичного розпізнавання тексту // Вісник Черкаського університету (науковий журнал) Випуск 172. Серія прикладна математика, 2020.
5. Алгоритми розпізнавання символів для систем штучного інтелекту. URL: https://elartu.tntu.edu.ua/bitstream/123456789/15097/2/Conf_2007_Herasimchuk_O-Alhorytmy_rozpiznavannia_117.pdf
6. О. І. Черняк, П. В. Захарченко. Інтелектуальний аналіз даних: підручник , 2014. 599 с.
7. Методи оптичного розпізнавання символів. URL: https://conf.ztu.edu.ua/wp-content/uploads/2022/04/120.pdf
8. Domingos, Pedro & Michael Pazzani (1997) «On the optimality of the simple Bayesian classifier under zero-one loss». Machine Learning, 29:103-137.
9. V. Wu, R. Manmatha and E. M. Riseman, «Finding Text in Images», In Proc. of Second ACM International Conference on Digital Libraries, Philadelphia, PA, pp. 23-26, 1997.
10. L. Agnihotri and N. Dimitrova, “Text Detection for Video Analysis”, In Proc. of the International Conference on Multimedia Computing and Systems, Florence, Italy, pp. 109-113, 1999.

11. Ye Q, Doermann D. Text detection and recognition in imagery: A survey. IEEE transactions on pattern analysis and machine intelligence. 2015 Jul 1;37(7):1480-500.
12. Tesseract (software) – Wikipedia. URL: https://en.wikipedia.org/wiki/Tesseract_(software)
13. About – OpenCV. URL: https://opencv.org/about
14. Image Denoising – OpenCV. URL: https://docs.opencv.org/3.4/d5/d69/tutorial_py_non_local_means.html
15. Morphological Transformations – OpenCV. URL: https://docs.opencv.org/3.4/d9/d61/tutorial_py_morphological_ops.html
16. Влучність та повнота – Wikipedia. URL: https://uk.wikipedia.org/wiki/Влучність_та_повнота
17. Документація по моделі Transformer Donut.URL: https://huggingface.co/docs/transformers/model_doc/donut

ДОДАТОК А

**Програмний код обробки зображення із застосуванням OpenCV**

```
import cv2
import os
import numpy as np

from matplotlib import pyplot as plt

def save_image(im_path, im_data):
    if not os.path.exists(os.path.dirname(im_path)):
        os.makedirs(os.path.dirname(im_path))

    cv2.imwrite(im_path, im_data)

# 01: Input image preview
input_image_file_path = "data/test.jpg"
output_directory = "preprocessed"

input_image = cv2.imread(input_image_file_path)

inverted_image = cv2.bitwise_not(input_image)
save_image(os.path.join(output_directory, "inverted.jpg"), inverted_image)

# 03: Binarization
def binarize_image(image, threshold_value=128, max_binary_value=255):
    return cv2.threshold(image, threshold_value, max_binary_value,
cv2.THRESH_BINARY)[1]
binary_image = binarize_image(inverted_image)
save_image(os.path.join(output_directory, "binary.jpg"), binary_image)
```

```
# 04: Noise Removal
def noise_removal(image):
    denoised_image = cv2.fastNlMeansDenoising(
        image, None, h=10, templateWindowSize=7, searchWindowSize=21
    )
    return denoised_image

denoised_image = noise_removal(binary_image)
save_image(os.path.join(output_directory, "denoised.jpg"), denoised_image)

# 05: Erosion
def thin_font(image):
    image = cv2.bitwise_not(image)
    kernel = np.ones((2, 2), np.uint8)
    image = cv2.erode(image, kernel, iterations=1)
    image = cv2.bitwise_not(image)

    return image

eroded_image = thin_font(denoised_image)
save_image(os.path.join(output_directory, "eroded.jpg"), eroded_image)

# 06: Dilation
def thick_font(image):
    image = cv2.bitwise_not(image)
    kernel = np.ones((2, 2), np.uint8)
    image = cv2.dilate(image, kernel, iterations=1)
    image = cv2.bitwise_not(image)

    return image
```

```
dilated_image = thick_font(eroded_image)
save_image(os.path.join(output_directory, "dilated.jpg"), dilated_image)

# Save result image
save_image(os.path.join(output_directory, "output.jpg"), dilated_image)
cv2.imshow("Preprocessing output", dilated_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

ДОДАТОК Б

**Програмний код розпізнавання тексту на зображенні**

```
import pytesseract
import cv2
import os
import numpy as np

from PIL import Image, ImageDraw, ImageFont
from matplotlib import pyplot as plt

def display(im_path):
    dpi = 80
    im_data = plt.imread(im_path)

    height, width = im_data.shape[:2]

    # What size does the figure need to be in inches to fit the image?
    figsize = width / float(dpi), height / float(dpi)

    # Create a figure of the right size with one axes that takes up the full figure
    fig = plt.figure(figsize=figsize)
    ax = fig.add_axes([0, 0, 1, 1])

    # Hide spines, ticks, etc.
    ax.axis("off")

    # Display the image.
    ax.imshow(im_data, cmap="gray")
```

```
    plt.show()

def save_image(im_path, im_data):
    if not os.path.exists(os.path.dirname(im_path)):
        os.makedirs(os.path.dirname(im_path))

    cv2.imwrite(im_path, im_data)

# Character Detection and Labeling image using OpenCV
input_image_file_path = "preprocessed/output.jpg"
output_directory = "result"

language = "eng+ukr"

input_image = cv2.imread(input_image_file_path)

boxes = pytesseract.image_to_data(input_image, lang=language)
print(boxes)

image_height, image_width = input_image.shape[:2]

box_gap = 3
# Convert OpenCV image to Pillow Image
pillow_image = Image.fromarray(cv2.cvtColor(input_image,
cv2.COLOR_BGR2RGB))
draw = ImageDraw.Draw(pillow_image)
font_size = 16
```

```
font = ImageFont.truetype("arial.ttf", font_size)

for x, box in enumerate(boxes.splitlines()):
    box = box.split()

    if x == 0 or len(box) < 12:
        continue
    print(box)

    box_x, box_y, box_width, box_height = (
        int(box[6]),
        int(box[7]),
        int(box[8]),
        int(box[9]),
    )

    word = box[11]

    draw.rectangle(
        [
            (box_x - box_gap, box_y - box_gap),
            (box_width + box_x + box_gap, box_height + box_y + box_gap),
        ],
        outline=(255, 0, 0),
        width=2,
    )

    # put a label under each rectangle with the corresponding character
```

```
    draw.text(
        (box_x, box_y + box_height),
        word,
        font=font,  # You can specify the font if needed
        fill=(255, 0, 0),
    )

output_image = cv2.cvtColor(np.array(pillow_image), cv2.COLOR_RGB2BGR)

save_image(os.path.join(output_directory, "output.png"), output_image)
cv2.imshow("Result", output_image)
cv2.resizeWindow("Result", 1200, 800)
cv2.waitKey(0)
cv2.destroyAllWindows()

# Text extraction on preprocessed image
input_image = cv2.imread(input_image_file_path)
result_text = pytesseract.image_to_string(input_image, lang=language)
print("Text result: ", result_text)
```