Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ТА РЕАЛІЗАЦІЯ МЕТОДУ РОЗПІЗНАВАННЯ ДОКУМЕНТІВ ДЛЯ АВТОМАТИЧНОЇ РЕАЛІЗАЦІЇ ІЗ ЗАСТОСУВАННЯМ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-21-3
Фурсов А. Д.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Любченко В. А.
(посада, прізвище, ініціали)

Допускається до захисту

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

2025 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Фурсову Андрію Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка та реалізація методу розпізнавання документів для автоматичної реалізації із застосуванням моделей штучного інтелекту

затверджена наказом університету від 19 травня 2025 року № 381Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 25 травня 2025 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, інтернет джерела, платформа для розміщення вебзастосунків HuggingFace Spaces, мова програмування Python, файлове сховище, мовні моделі GPT, Gemini, Claude, Tesseract OCR, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом OpenCV.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів попередньої обробки зображень у задачах.
2. Аналіз сучасних методів OCR та можливості їх адаптації до документів.
3. Побудова та обґрунтування математичних моделей фільтрації зображень, що покращують якість вхідних даних для розпізнавання.
4. Розробка програмної системи на мові Python із графічним інтерфейсом.
5. Тестування системи на прикладах документів різної якості.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Актуальність задачі автоматичного розпізнавання документів, схема логіки роботи програми, приклади обробки зображень на різних етапах (оригінал, після фільтрації, після бінаризації), демонстрація результатів розпізнавання ID-картки, фрагмент JSON виводу, інтерфейс графічної програми, порівняльна таблиця результатів тестування системи на різних зображеннях.

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	07.04.2025	
2	Аналіз завдання, підбір літератури	08.04.25-10.04.25	
3	Аналіз літератури з досліджуваної проблеми	11.04.25-14.04.25	
4	Аналіз технічних засобів	15.04.25-20.04.25	
5	Розробка методу	21.04.25-27.04.25	
6	Програмна реалізація	28.04.25-11.05.25	
7	Оформлення пояснювальної записки	12.05.25-20.05.25	
8	Перевірка на нормоконтроль	21.05.25-01.06.25	
9	Перевірка на плагіат	21.05.25-01.06.25	
10	Рецензування	21.05.25-01.06.25	
11	Підготовка презентації та доповіді	21.05.25-18.06.25	
12	Занесення роботи в електронний архів	02.06.25-18.06.25	
13	Попередній захист кваліфікаційної роботи	02.06.25-18.06.25	

Дата видачі завдання 7 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____ доц. Любченко В. А._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 67 с., 37 рис., 33 джерела.

ВЕБПЛАТФОРМА, МІКРОСЕРВІСИ, ДОКУМЕНТНЕ РОЗПІЗНАВАННЯ, АВТОМАТИЗОВАНА ОБРОБКА, AI-МОДЕЛІ, SAGA, PROMPT-ІНЖИНІРИНГ, ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ.

Об’єктом роботи є різнорідні документи.

Метою роботи є розробка та впровадження методу автоматизованого розпізнавання документів із подальшою класифікацією та витягуванням ключових даних для безшовної інтеграції в бізнес-процеси.

Особлива увага приділена prompt-інжинірингу для генерації оптимальних запитів до LLM-моделей, що підвищує точність інтерпретації складних текстових сегментів. У реалізації застосовано стек технологій OpenCV та Tesseract OCR для попередньої обробки зображень, Google Cloud Functions для серверлес-логіки, а також GPT-4 через API для семантичного аналізу і перевірки контексту. Побудовано мікросервіси, які виконують прийом і нормалізацію вхідних файлів, модуль розпізнавання й екстракції, компонент валідації даних та службу управління транзакційними подіями за патерном SAGA. Кінцевим результатом є реалізація модульної платформи, що надає користувачам інтерфейс для завантаження документів і отримання рядкового JSON-висновку з категоризованими полями, відкриваючи нові можливості для автоматизації документообігу та аналітики.

WEB PLATFORM, MICROSERVICES, DOCUMENT RECOGNITION, AUTOMATION, AI MODELS, SAGA, PROMPT ENGINEERING, INTELLIGENT ANALYSIS.

The object of this work comprises heterogeneous documents.

The aim is to develop and implement a method for automated document recognition with subsequent classification and extraction of critical data, seamlessly integrating into business workflows.

Special emphasis is placed on prompt engineering to formulate optimal queries for LLM models, enhancing the precision of interpreting complex textual segments. The implementation leverages a technology stack including OpenCV and Tesseract OCR for image preprocessing, Google Cloud Functions for serverless logic, and GPT-4 via API for semantic analysis and context validation. Microservices were architected to handle file ingestion and normalization, recognition and extraction modules, data validation components, and a transaction orchestration service following the SAGA pattern. The final outcome is a modular platform providing users an interface to upload documents and receive a structured JSON output with categorized fields, unlocking advanced automation of document workflows and analytics.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – Application Programming Interface (інтерфейс програмування застосунків)

ACID – Atomicity, Consistency, Isolation, Durability (Атомарність, Послідовність, Ізоляція, Довговічність)

GCP – Google Cloud Platform (платформа Google Cloud)

OCR – Optical Character Recognition (оптичне розпізнавання символів)

LLM – Large Language Models (великі моделі даних)

SAGA – Saga Pattern (паттерн для управління транзакціями у мікросервісних архітектурах)

JSON – JavaScript Object Notation (формат обміну даними)

DB – Database (база даних)

HTTP – Hypertext Transfer Protocol (протокол передачі гіпертексту)

REST – Representational State Transfer (архітектурний стиль взаємодії компонентів у мережі)

GPT – Generative Pre-trained Transformer (генеративна попередньо натренована трансформерна модель)

MQ – Message Queue (черга повідомлень)

CRUD – Create, Read, Update, Delete (створення, читання, оновлення, видалення)

AI – Artificial Intelligence (штучний інтелект)

BPMN – Business Process Model and Notation (модель бізнес–процесу та нотація)

CRNN – Convolutional Recurrent Neural Networks (згорткові рекурентні нейронні мережі)

CDF – Cumulative Distribution Function (кумулятивна функція розподілу)

ВСТУП

У сучасному цифровому середовищі автоматизація обробки документів є ключовим чинником підвищення ефективності бізнес-процесів та зменшення операційних витрат. Особливо гостро стоїть завдання швидкого й точного розпізнавання різноманітних форм документів від сканованих договорів до електронних рахунків, адже ручна обробка призводить до затримок і помилок.

Актуальність розробки методу автоматичного розпізнавання документів із застосуванням моделей штучного інтелекту зумовлена потребою обробляти великі обсяги інформації з максимальною точністю. Використання мікросервісної архітектури забезпечує гнучкість і масштабованість рішення, а інтеграція OpenCV і Tesseract OCR на етапі попередньої обробки зображень дозволяє підвищити якість витягування тексту. Далі, семантичний аналіз із залученням LLM-моделей та промпт-інжиніринг оптимізує формулювання запитів, що гарантує коректну інтерпретацію контексту й автоматизовану класифікацію документів за заздалегідь визначеними категоріями.

1 ОСНОВНІ МЕТОДИ РОЗПІЗНАВАННЯ ДОКУМЕНТІВ ДЛЯ АВТОМАТИЧНОЇ РЕАЛІЗАЦІЇ ІЗ ЗАСТОСУВАННЯМ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Оптичне розпізнавання символів

OCR є одним з ключових етапів в автоматизованій обробці документів. Метод передбачає трансформацію зображень друкованого або рукописного тексту у формат, який може бути оброблений комп’ютером. На цьому рівні використовуються як традиційні алгоритми, так і новітні нейронні моделі. Обробка зазвичай розпочинається з підготовки зображення, що охоплює вирівнювання, регулювання контрасту та усунення шуму. Приклад наведено на рисунку 1.1. Після цього система переходить до розпізнавання символів. Для цього застосовуються методи зіставлення із шаблонами або моделі глибинного навчання, які пройшли попереднє тренування. Підсумком є текст, який можна далі аналізувати за змістом [1].

Рисунок 1.1 – Підготовка зображення документа. Підвищення контрастності для виділення тексту для OCR

Технологія OCR активно використовується у банківських системах, сфері документообігу, освітніх платформах, а також в медицині. Незважаючи на високу продуктивність та здатність обробляти великі обсяги інформації, метод має деякі обмеження. Наприклад точність залежить від якості початкового зображення, розбірливості шрифтів та наявності артефактів. Це обумовлює необхідність додаткових етапів обробки для покращення вхідних даних перед розпізнаванням.

1.2 Обробка зображень у системах автоматичного розпізнавання документів

Якість роботи OCR значною мірою визначається якістю вхідного зображення. Для цього застосовуються методи попередньої обробки, що покращують чіткість тексту, усувають нахили, знижують рівень шуму та дозволяють відділити текст від фону. Геометрична корекція допомагає позбутися спотворень перспективи і нахилів, що часто зустрічаються на сканах або фотографіях з мобільних пристроїв.

Додатково застосовуються методи аналізу контурів, які дозволяють локалізувати текст на зображенні. Це сприяє точнішому поділу на смислові одиниці та зменшує ризик некоректного розпізнавання фрагментів, коли після редагування зображення залишаються артефакти, які програма може прийняти за текст. Нормалізація освітлення і контрасту реалізується шляхом адаптивної бінаризації. Вона забезпечує точне розділення фону і символів. Для зменшення артефактів використовуються згладжувальні фільтри. У результаті формується зображення високої якості, що суттєво підвищує точність наступного розпізнавання [2].

Автоматичне виявлення областей інтересу дозволяє системі зосередитись на важливих зонах документа. Це підвищує ефективність обробки та зменшує ризик помилок при наявності зайвих елементів, таких як логотипи, печатки або підписи [3].

1.2.1 Геометрична та візуальна нормалізація документа

Початкова якість зображення має ключове значення для ефективного розпізнавання тексту. Першим кроком є виправлення геометричних спотворень і візуальне вирівнювання. До цього належать операції з корекції нахилу та перспективи, що особливо актуально при використанні мобільних пристроїв. На цьому ж етапі проводиться обрізання фону та виділення меж документа (рис. 1.2).

Далі здійснюється покращення чіткості зображення за допомогою адаптивної бінаризації. Вона дозволяє ефективно виділити текст навіть при неоднорідному освітленні. Шумозаглушення досягається через фільтрацію, використовуються гаусове розмиття або медіанні фільтри. Це дозволяє покращити сприйняття символів. Результатом стає рівне та контрастне зображення, придатне до точного аналізу методом OCR [4].

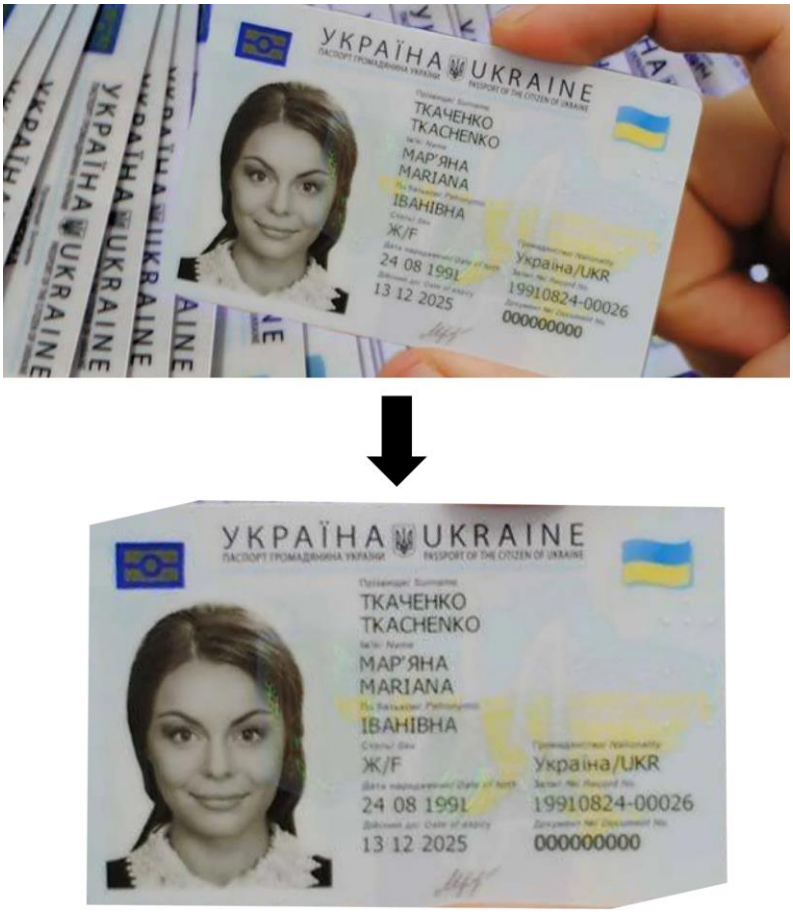

Рисунок 1.2 – Підготовка зображення документа. Геометрична нормалізація документа шляхом повороту і кадрування

1.2.2 Підготовка до розпізнавання

Після очищення та нормалізації зображення важливо зосередити обробку лише на змістовно значущих ділянках. Для цього здійснюється автоматичне виявлення області інтересу, яка дозволяє виключити з аналізу такі візуальні об’єкти, як логотипи, підписи або печатки. Виокремлення саме текстових блоків пришвидшує роботу системи та зменшує кількість помилок при подальшому розпізнаванні.

У деяких випадках проводиться додаткове зонування структури документа. Наприклад, відокремлюють таблиці, заголовки або пояснення. Це забезпечує адаптацію системи до складних макетів. Результатом є підготовлене зображення з визначеними текстовими зонами, яке передається до модуля розпізнавання. Для підвищення ефективності можуть застосовуватись правила, що враховують просторові взаємозв’язки між елементами документа. Це дозволяє точніше визначити межі текстових блоків і зменшити перетин з нерелевантними областями [5].

1.3 Глибинне навчання розпізнавання тексту та структури документа

Завдяки сучасним обчислювальним потужностям і відкритим наборам даних методи глибинного навчання широко впроваджуються в задачі OCR. Замість фіксованих правил використовуються нейронні мережі. Вони здатні вивчати зразки і на цій основі розпізнавати символи, слова та структуру документа (рис. 1.3). Це дозволяє ефективно працювати з різноманітними шрифтами, кольорами та неідеальною якістю зображень.

Особливо ефективні згорткові рекурентні нейронні мережі (CRNN). Вони поєднують здатність до візуального аналізу з обробкою послідовностей. Завдяки цьому можливо зберігати порядок символів та враховувати залежності між ними. Ще один напрямок розвитку полягає у використанні моделей, які дозволяють розпізнати не тільки текст, а й логічну

структуру документа. Це дозволяє автоматично виділяти таблиці, зони введення і подібні елементи.

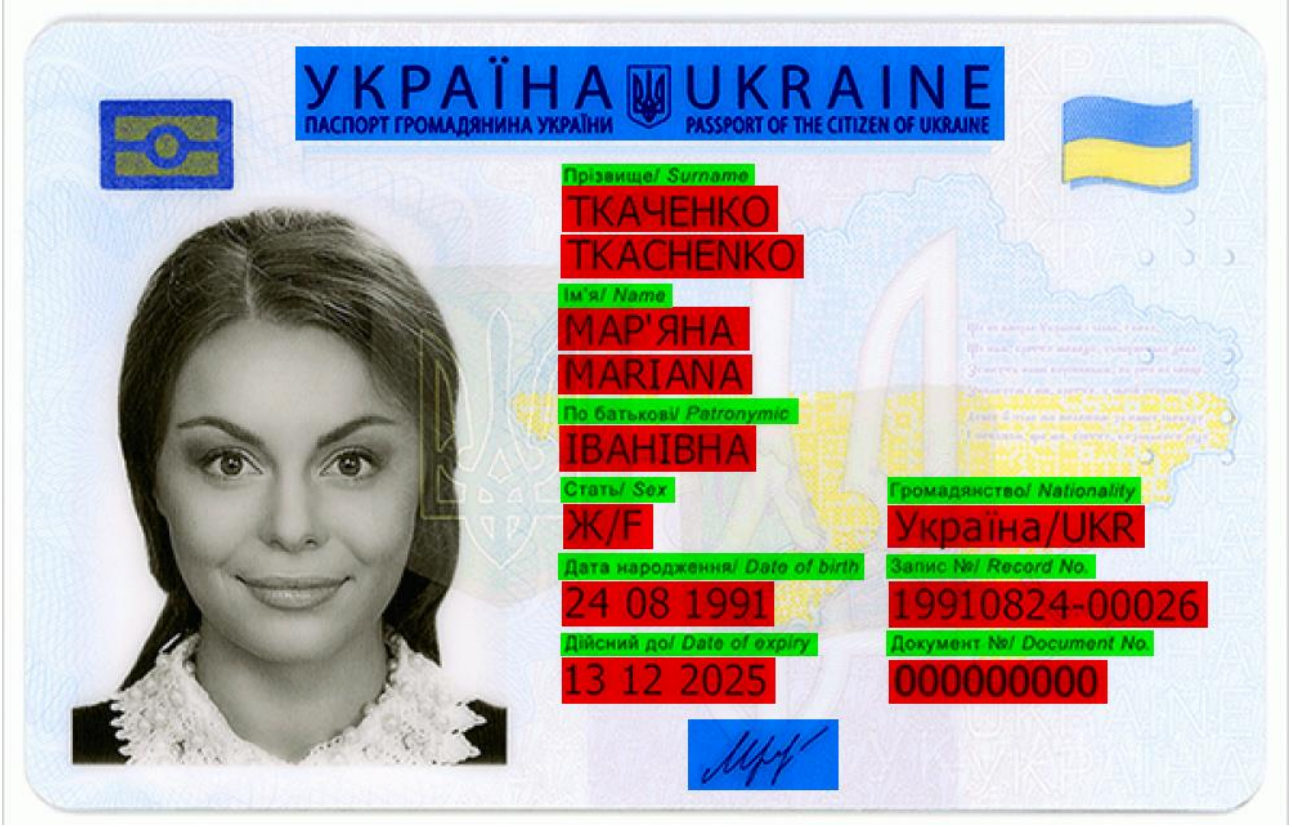

Рисунок 1.3 – Визначення та виділення потрібних блоків тексту на фотографії паспорта

Сучасні моделі підтримують багатомовність. Вони можуть розпізнавати документи, що містять тексти українською та англійською одночасно. Крім того, можливе навчання на спеціалізованих даних. Це покращує точність в окремих галузях, наприклад у юридичній чи медичній сфері. Останні дослідження у сфері OCR активно просувають концепцію «end-to-end learning», де весь процес від пікселя до структурованих даних виконується в межах однієї моделі. Такий підхід дозволяє уникнути втрат інформації, які виникають у класичних багатокрокових схемах [6].

1.3.1 Архітектури навчання для розпізнавання тексту

Методи глибинного навчання, зокрема згорткові нейронні мережі (CNN), відіграють провідну роль у сучасних системах OCR. Такі моделі здатні самостійно виявляти важливі характеристики символів, не покладаючись на попередньо задані шаблони. CNN добре справляються з шумами, геометричними спотвореннями, а також демонструють стійкість до змін у шрифтах і кольорових схемах. Це дозволяє застосовувати їх навіть для обробки зображень із низькою якістю (рис. 1.4).

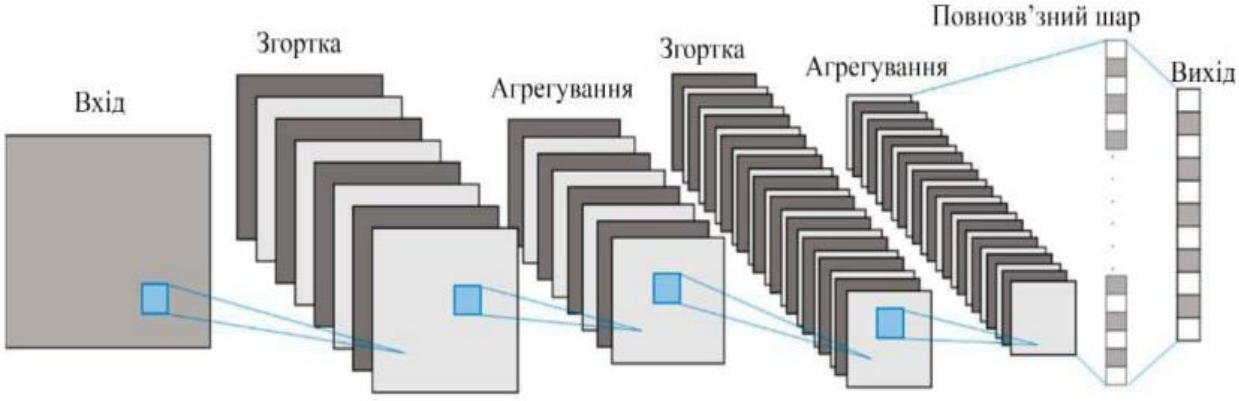

Рисунок 1.4 – Архітектура згорткової нейронної мережі

Особливої уваги заслуговують згорткові рекурентні нейронні мережі (CRNN). Такі архітектури поєднують аналіз просторових характеристик із обробкою послідовностей, що забезпечує збереження порядку символів у текстовому рядку. Ця здатність враховувати контекст допомагає досягати більшої точності при розпізнаванні складних мовних конструкцій. Саме тому CRNN широко застосовуються в задачах, де важливо не лише ідентифікувати символи, але й зберегти логічну послідовність даних [7].

1.3.2 Розпізнавання структури, підхід «end-to-end»

Сучасні моделі глибинного навчання здатні не лише визначати текст на зображенні, але й ідентифікувати логічну структуру всього документа. Такі моделі, як LayoutLM (рис. 1.5), DocFormer або Donut, поєднують текстову та візуальну інформацію, дозволяючи розпізнавати таблиці, зони введення, підписи, заголовки тощо. У результаті система розпізнає не просто слова, а цілісні змістовні фрагменти, як наприклад підписи, поля введення або таблиці, що особливо корисно для автоматизації обробки складних форм.

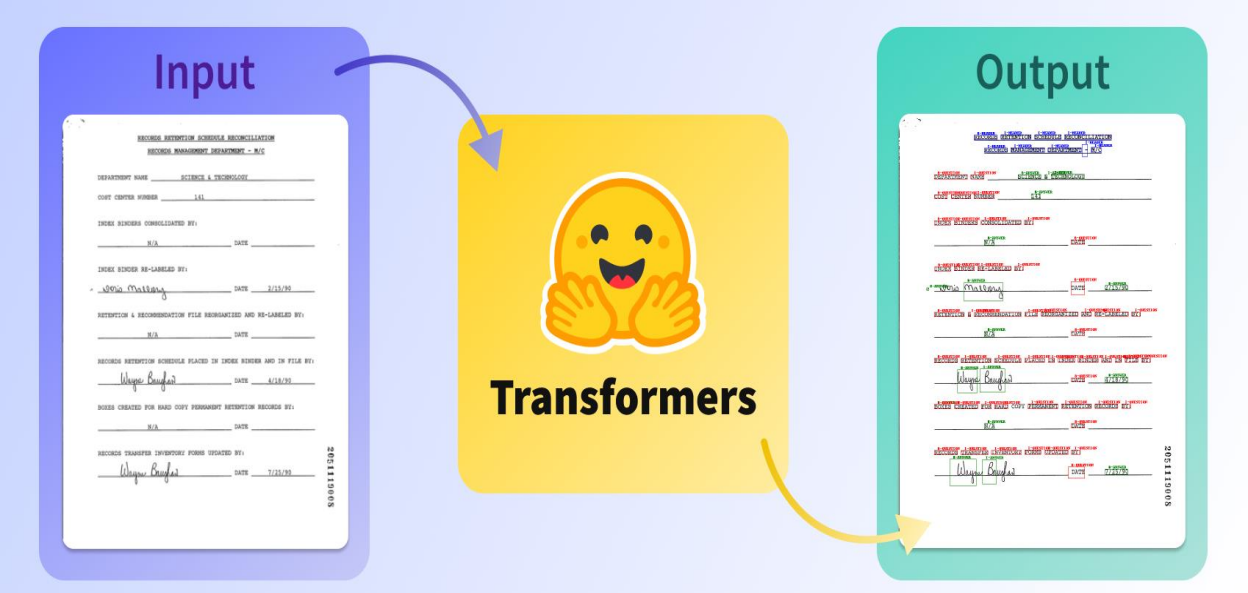

Рисунок 1.5 – Приклад роботи моделі LayoutLM

Суттєвою перевагою таких моделей є здатність працювати з багатомовним текстом без потреби перемикати налаштування. Система може розпізнавати фрагменти українською, англійською або іншими мовами одночасно, що спрощує її використання у міжнародних документах. Крім того, можливо донавчати модель на спеціалізованих галузевих даних, що забезпечує точність навіть у випадках із дуже специфічною термінологією [8].

Моделі глибинного навчання також демонструють здатність адаптуватися до різноманітних форматів документів. Вони можуть коректно працювати із структурованими, напівструктурованими та неструктурованими даними, що надходять у вигляді таблиць, текстових блоків або змішаних типів макетів. Це забезпечується за рахунок комбінованої обробки просторових і текстових ознак, що дозволяє точно локалізувати як ключові поля, так і зв’язки між ними. Завдяки цьому система зберігає стабільну точність навіть у випадках складних макетів, нестандартної верстки або наявності додаткових елементів, таких як печатки чи підписи [9].

Особливої уваги заслуговує підхід «end-to-end learning». Він передбачає, що весь процес від завантаження зображення до отримання готових структурованих даних виконується єдиною нейронною моделлю. Такий підхід дозволяє мінімізувати втрати інформації, уникнути накопичення помилок між окремими етапами і досягти більш узгоджених результатів (рис. 1.6).

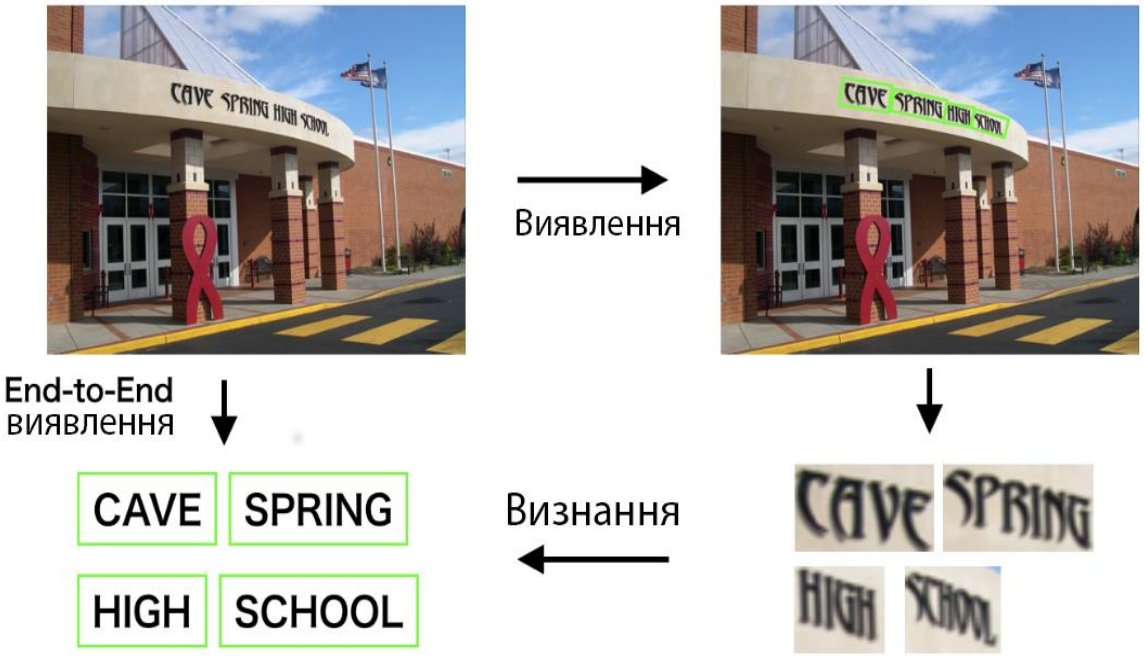

Рисунок 1.6 – Приклад роботи підходом «end-to-end»

1.3.3 Практичне застосування та адаптація моделей

Однією з ключових переваг глибинного навчання є гнучкість у налаштуванні моделей для специфічних задач. На відміну від універсальних рішень, які не завжди забезпечують стабільну точність при обробці складних шаблонів, спеціалізовані моделі адаптуються до особливостей вхідних даних завдяки навчанню на галузевих вибірках. Це дозволяє досягти високої ефективності навіть у нестандартних умовах.

Окрім галузевої адаптації, важливу роль відіграє можливість кастомізації моделей під конкретні бізнес-процеси. Наприклад, у сфері логістики документи можуть мати унікальні маркування, нестандартні поля або коди товарів, які не зустрічаються в загальнодоступних наборах даних. Адаптація моделі до таких умов передбачає включення додаткових рівнів валідації та оновлення словникової бази з урахуванням внутрішньої документації підприємства.

Ще одним прикладом є страхова галузь, де зустрічаються документи, що містять як заповнені вручну форми, так і додатки з багатьма сторінками. У таких випадках моделі мають справлятися з великим обсягом інформації, виявляти ключові поля, а також коректно інтерпретувати структуру багатосторінкових документів. Це вимагає поєднання просторового аналізу з контекстною обробкою, а також застосування механізмів розпізнавання логічних блоків даних.

Юридичні документи часто мають складну структуру, чітко визначену мову і повторювані текстові блоки. Такі особливості вимагають індивідуального підходу до побудови моделі, яка має бути чутливою до формальних конструкцій. У медичній сфері зустрічаються інші виклики, зокрема наявність скорочень, специфічної термінології та форм, які важко розпізнати без попереднього розширення мовної бази.

У документах бухгалтерського обліку особливе значення має точність числових значень. Помилка навіть в одному символі може мати серйозні наслідки. Саме тому в таких випадках важливою є не лише здатність моделі до розпізнавання тексту, а й її стійкість до помилок при обробці чисел.

Моделі глибинного навчання можуть бути донавчені на внутрішніх даних конкретної організації. Це дозволяє створити систему, яка буде максимально відповідати потребам підприємства і працюватиме стабільно навіть у складних виробничих умовах.

1.4 Великі мовні моделі для аналізу розпізнаного тексту

Після отримання тексту з документа виникає необхідність його подальшого аналізу. Цю функцію виконують великі мовні моделі (LLM), такі як GPT, BERT або їх спеціалізовані варіанти. Завдяки глибинному розумінню контексту ці моделі можуть встановлювати логічні зв’язки між різними частинами тексту та автоматично витягувати необхідні поля. Наприклад, можна визначити номер документа, дату, тип запису та інші сутності.

Після отримання тексту за допомогою OCR, його структура часто буває нечіткою або фрагментарною. У таких випадках великі мовні моделі відіграють ключову роль у впорядкуванні та осмисленні інформації. Вони здатні здійснювати постобробку, уточнювати формулювання та відновлювати контекст, втрачені на попередніх етапах [9].

Крім того, LLM можуть виявляти приховані логічні зв’язки та виконувати класифікацію за змістом. Наприклад, моделі автоматично розподіляють текстові блоки за типами. Наприклад, реквізити, заголовки, службова інформація, або визначають стиль написання, що може бути корисним для подальшої автоматизованої маршрутизації документа.

Для забезпечення точності витягування інформації використовується техніка промпт інжинірингу. Цей процес полягає у створенні запиту до

моделі, в якому описується, яку саме інформацію потрібно отримати, а також наводяться приклади бажаного результату. Такий підхід дозволяє моделі краще інтерпретувати зміст документа та повертати структуровану відповідь у зручному форматі, наприклад у вигляді JSON або XML.

Крім витягування інформації, мовні моделі здатні перетворювати її в більш зрозумілу форму. Наприклад, вони можуть автоматично сформувати стислий огляд документа або надати пояснення до конкретних полів. Це значно підвищує зручність подальшої роботи з даними (рис. 1.7).

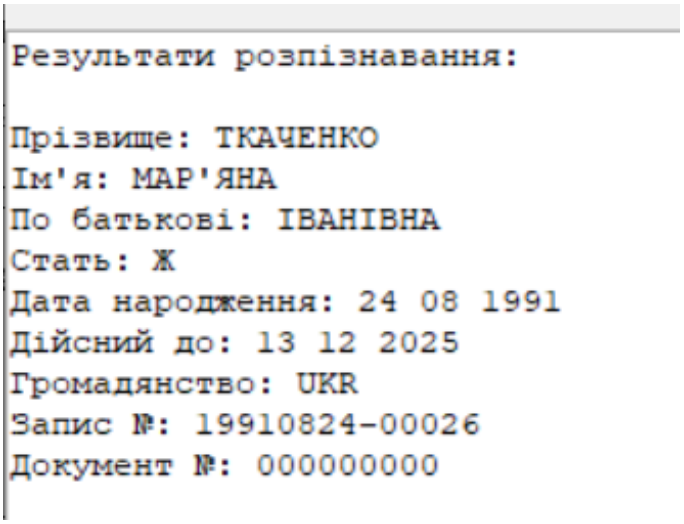

Рисунок 1.7 – Згенероване коротке оформлене пояснювання полів після сканування документу

1.5 Автоматизована категоризація документів

Після розпізнавання тексту наступним етапом є класифікація документа та перевірка коректності отриманих даних. Для цього застосовуються моделі машинного навчання, які аналізують вміст документа, кількість заповнених полів, характерні слова і структурні елементи. Завдяки цьому система може автоматично визначити, до якого типу належить документ, наприклад чи це рахунок, акт, накладна або договір [10].

Також виконується перевірка правильності витягнутої інформації. Система встановлює, чи відповідають значення певним форматам, наприклад, чи є дата коректною або чи не містить поле помилок. Якщо результат розпізнавання не дає достатньої впевненості, документ може бути переданий на ручну перевірку або до уточнення оператором.

Системи категоризації можуть функціонувати повністю автономно або із залученням користувача. У такому разі використовується активне навчання, коли модель поступово покращує свої результати на основі дій оператора. Такий підхід дає змогу досягти високої точності в умовах змінних або нестандартних даних.

Додатково реалізується можливість перевірки реквізитів через зовнішні джерела. Це може бути підключення до баз даних чи API для валідації ІПН, номерів рахунків або інших полів. Така перевірка підвищує достовірність розпізнаних даних.

У більш складних випадках використовується схематичне групування інформації. Наприклад, дані поділяються за блоками для спрощення їх подальшої інтеграції з ERP системами або іншим програмним забезпеченням. Перевірка може враховувати бізнес правила. Наприклад, документ не може мати дату, що перевищує поточну. На цьому етапі критично важливо не лише витягти текст, а й впевнено структуризувати його та перевірити на відповідність логіці і фактам. Саме для цього впроваджується система автоматизованої категоризації та валідації.

1.5.1 Класифікація типів документів машинним навчанням

Класифікація документів передбачає автоматичне визначення їх типу на основі аналізу текстового вмісту та його структурних ознак. Моделі машинного навчання використовують різні характеристики, серед яких ключові слова, кількість і тип полів, наявність шаблонних елементів або

заголовків. Це дає змогу встановити, до якого типу належить документ, наприклад рахунок фактура, акт виконаних робіт, договір чи накладна.

У процесі класифікації важливим є також врахування контексту документа. Деякі типи документів можуть мати подібні заголовки або формати, але різнитися за функціональним призначенням. Наприклад, акт приймання виконаних робіт і акт звірки можуть містити схожі блоки тексту, однак їхнє призначення та структура взаємозв'язаних полів різні. Для таких випадків моделі навчання використовують векторне представлення тексту або контекстну сегментацію, що дозволяє враховувати глибші відмінності.

Крім текстових і структурних ознак, сучасні підходи передбачають використання візуальних характеристик документа. Це можуть бути позиції полів, розмітка, шрифт, колір, наявність підписів або штампів. Застосування таких ознак підвищує точність класифікації, особливо при роботі зі сканованими копіями документів, де текстовий вміст може бути частково пошкоджений або неповний.

Системи класифікації можуть працювати у повністю автономному режимі або з інтерактивним компонентом. У другому випадку застосовується механізм активного навчання, коли модель покращується завдяки зворотному зв'язку від користувача. Це є особливо ефективним у ситуаціях із невідомими шаблонами або новими видами документів, які раніше не зустрічались у навчальній вибірці. Такий підхід дозволяє підтримувати високу точність навіть у мінливому середовищі [11].

Інтерактивне навчання також передбачає можливість поступового оновлення моделі без необхідності повного перенавчання. Це суттєво скорочує час адаптації системи до нових умов і забезпечує безперервне вдосконалення без переривання роботи. Крім того, класифікатори можуть бути інтегровані в більші системи управління контентом або документообігом, де визначення типу документа є лише першим етапом у більш складному процесі автоматизації.

1.5.2 Валідація даних і перевірка логічної узгодженості

Окрім класифікації документа, важливо переконатися у правильності витягнутих даних. На початковому етапі виконується базова перевірка формату. Наприклад, перевіряється, чи є значення в полі «Дата» справжньою датою, чи «Сума» є числом, чи правильно вказані ІПН або номер банківського рахунку. Після цього проводиться логічна перевірка узгодженості полів між собою. Система може порівнювати підсумкову суму з сумами в рядках, перевіряти хронологію дати або перевіряти код організації в офіційних реєстрах.

Для підвищення рівня точності перевірок можуть застосовуватись складні логічні правила, що враховують залежності між кількома полями. Наприклад, якщо дата акта пізніша за дату договору, це вважається помилкою. Або якщо номер банківського рахунку не відповідає банківському коду країни, такий документ маркується як потенційно некоректний. Подібні перевірки реалізуються через декларативні бізнес-правила, що задаються на рівні конфігурації системи.

Окремий напрям валідації це перевірка на дублікати. У великих системах документообігу ймовірність повторної подачі одного й того самого документа з різними файлами досить висока. Тому реалізуються механізми виявлення дублікатів за контрольними сумами, метаданими або вмістом основних полів. У разі збігу система автоматично блокує повторну обробку або маркує документ для ручної перевірки.

Для розширеної перевірки система може бути інтегрована з зовнішніми інформаційними ресурсами. Це дозволяє звіряти реквізити з державними базами або іншими надійними джерелами. Якщо результат перевірки не є остаточним, документ направляється на додаткову перевірку оператором.

Застосування комбінованої перевірки як внутрішньої логіки, так і зовнішньої валідації забезпечує високий рівень достовірності фінального результату. Це особливо важливо в таких сферах, як банківська діяльність,

держзакупівлі чи аудит, де навіть незначні помилки можуть призвести до серйозних наслідків. Завдяки сучасним підходам до валідації системи здатні значно зменшити кількість людських втручань і підвищити загальну надійність процесу обробки документів [12].

1.6 Мікросервісна архітектура

Щоб забезпечити гнучкість, масштабованість та зручність підтримки, системи автоматизованого розпізнавання документів будуються за мікросервісною архітектурою. У цьому підході кожен сервіс відповідає за окрему функцію [13]. Наприклад, обробка зображень, OCR, аналіз змісту, валідація, збереження даних та взаємодія з користувачем реалізуються як незалежні компоненти (рис. 1.8).

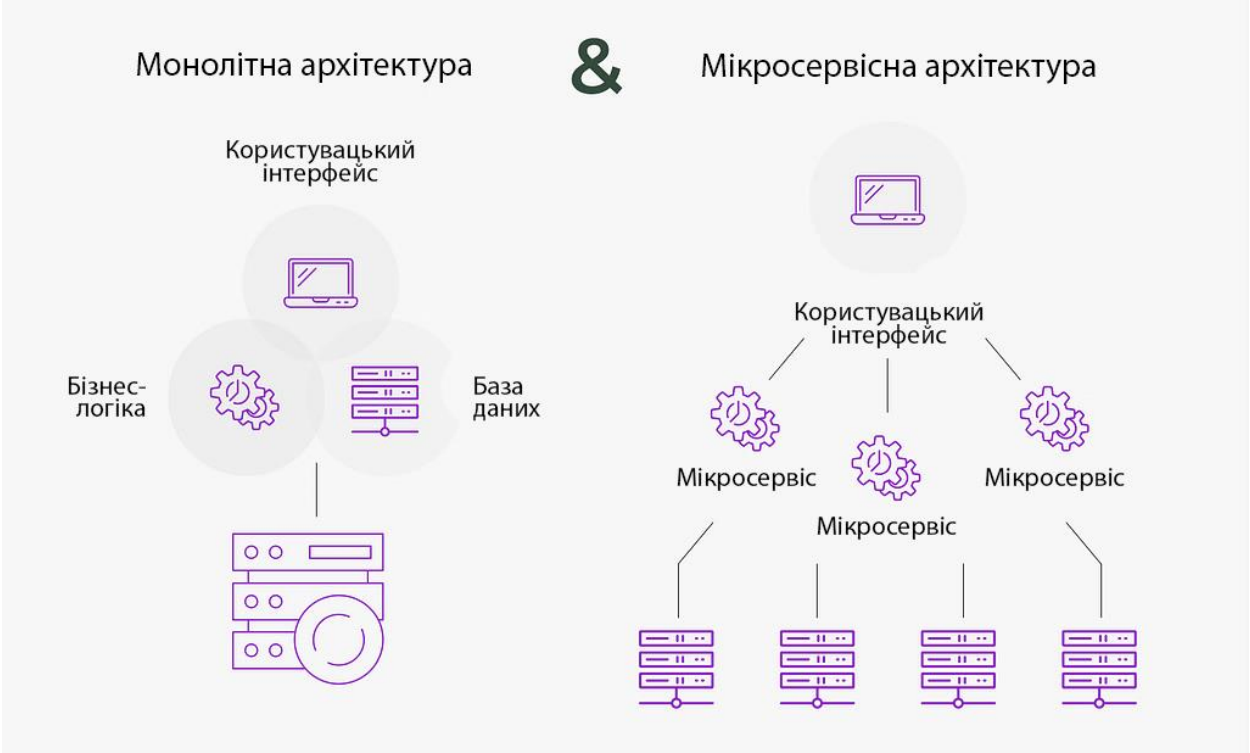

Рисунок 1.8 – Порівняння монолітної архітектури з мікросервісною

Взаємодія між компонентами координується за допомогою патерна SAGA. Це дозволяє організувати складні транзакції у розподіленому середовищі з можливістю відновлення у разі збоїв. Такий підхід дозволяє зберігати цілісність обробки навіть при помилках окремих частин системи.

Мікросервіси розгортаються у хмарному середовищі з використанням контейнерів, таких як Docker, і систем управління, таких як Kubernetes [14]. Це забезпечує гнучкість адаптації до різних рівнів навантаження. Від невеликих локальних інсталяцій до великих підприємств, які обробляють тисячі документів щодня.

Архітектура також дозволяє масштабувати сервіси горизонтально. Це означає, що окремі модулі можуть запускатись паралельно у кількох копіях. Оркестрація подій забезпечує поетапну обробку, коли кожен крок виконується окремим сервісом, а далі передає результат наступному.

Сучасні рішення включають також безсерверну архітектуру. У такій системі функції запускаються на вимогу, що дозволяє економити ресурси при нестабільному навантаженні. Використання моніторингу в реальному часі дозволяє виявляти слабкі місця і оперативно реагувати на зміни.

1.7 Безпека та конфіденційність даних

У контексті обробки документів, що містять конфіденційну інформацію, забезпечення надійного рівня захисту даних набуває критичного значення. Системи автоматизованого розпізнавання часто працюють з персональними відомостями, фінансовими записами або юридично значущими полями, що зобов’язує розробників та адміністраторів дотримуватись високих стандартів інформаційної безпеки.

Захист даних реалізується на кількох рівнях. Передусім, здійснюється шифрування як при зберіганні, так і під час передачі між модулями. Шифрування реалізується через застосування протоколу захисту

транспортного рівня (TLS) для API-запитів, а також через використання зашифрованих файлових систем або спеціалізованих сховищ із підтримкою апаратного шифрування. Це запобігає перехопленню або викривленню інформації з боку третіх осіб [15].

Другим рівнем захисту виступає контроль доступу. Кожен компонент системи повинен працювати лише в межах чітко визначених дозволів. Для цього застосовується рольова авторизація, яка ділить права між адміністраторами, користувачами та технічними службами. Усі дії, що стосуються перегляду, редагування або передачі документів, реєструються в журналах аудиту. Це дає змогу забезпечити прозорість операцій, відслідковувати помилки та формувати базу для ретроспективного аналізу.

Окремо слід зазначити важливість захисту резервних копій. Навіть за умов надійного поточного захисту, резервні збереження повинні відповідати тим самим стандартам, що й основні дані. Це стосується як шифрування, так і обмеження доступу до резервних реплік або зображень у хмарному середовищі. Безпека даних є не факультативною умовою, а фундаментальним принципом проєктування сучасної системи розпізнавання документів [16].

1.8 Логування та моніторинг системи

Функціональна надійність великих інформаційних систем напряму залежить від ефективності логування та моніторингу. У системах обробки документів, де щодня відбуваються тисячі транзакцій, своєчасне виявлення помилок, уповільнень або збоїв критично важливе для стабільної роботи.

Логування виконується на кількох рівнях. На інфраструктурному рівні фіксується інформація про використання ресурсів, зокрема процесору, оперативної пам’яті, мережеву активність та обсяг вхідних запитів. На рівні сервісів ведуться журнали про кожен крок обробки документа, включаючи

статус OCR, результат валідації, класифікації та збереження. Бізнес рівень логування забезпечує аналітику щодо типів оброблених документів, кількості помилок за категоріями та продуктивності окремих етапів.

Моніторинг передбачає збір метрик у реальному часі. Це дозволяє адміністраторам виявляти перевантаження окремих мікросервісів, вчасно масштабувати інфраструктуру або застосовувати механізми автоматичного перезапуску. Візуалізація даних моніторингу реалізується через доски, де виводяться ключові показники, такі як середній час обробки документа, кількість помилок за годину та статус підключень до зовнішніх сервісів.

У разі виникнення помилок, таких як зростання помилок в OCR або нестача ресурсів на одному з вузлів, система може автоматично реагувати через тригери [17]. Наприклад, виконати масштабування контейнерів, активувати додаткові обчислювальні потужності або відключити неактивні модулі. Логування та моніторинг не лише дозволяють підтримувати стабільність, але й формують основу для прийняття управлінських рішень, планування навантаження та удосконалення системи в цілому.

1.9 Забезпечення якості та тестування

Якість системи обробки документів визначається не лише точністю розпізнавання, але й стабільністю виконання кожної окремої операції, від завантаження зображення до експорту структурованих даних. Для цього впроваджується багаторівнева система тестування, яка охоплює модулі, API, сценарії та інтегровані середовища.

Тестування починається з модульного рівня, де кожен окремий компонент перевіряється на відповідність очікуваній логіці. Наприклад, OCR модуль повинен коректно обробляти зображення різних розмірів, типів і роздільностей. Далі слідує інтеграційне тестування, яке перевіряє взаємодію

модулів, чи передається результат OCR до класифікатора, чи коректно виконується валідація, чи зберігаються дані в базі.

Ще одним важливим етапом є навантажувальне тестування, яке дозволяє оцінити стійкість системи при високому обсязі запитів. Імітація пікового навантаження показує, як поводять себе мікросервіси, чи виникають затримки або втрати даних під час обробки тисяч документів одночасно. Такі тести допомагають виявити вузькі місця в архітектурі, налаштувати балансування навантаження і вдосконалити механізми масштабування [18].

Крім технічного тестування, проводяться також перевірки з точки зору кінцевого користувача, «end-to-end» тестування. Тут перевіряється повний ланцюг взаємодії від завантаження документа в інтерфейсі до отримання остаточного результату. Такий підхід дозволяє виявити логічні помилки або некоректну поведінку системи в реальних сценаріях, що особливо важливо для забезпечення високої якості користувацького досвіду [19].

Окремо проводиться тестування якості розпізнавання. Для цього створюються датасети із наперед відомим вмістом. Результати OCR порівнюються з ідеальними значеннями, після чого розраховуються показники. Погіршення одного з показників є сигналом до перегляду моделі або корекції підготовки зображення. Також важливим є регресійне тестування, яке дозволяє перевірити, чи не зламалась функціональність після внесення змін до коду або оновлення бібліотек. Результати всіх типів тестування фіксуються, зберігаються в системі контролю якості і використовуються як доказ стабільності при розгортанні в продуктивне середовище. Процес забезпечення якості є безперервним і необхідним для гарантування надійності системи на всіх етапах її життєвого циклу.

1.10 Постановка задачі

Таким чином, на сьогодні існує потреба у системах, здатних автоматично розпізнавати дані з документів, зокрема ID-карток, паспортів та інших офіційних бланків. Такі системи можуть використовуватися в державному секторі, банківських установах, сервісах електронної ідентифікації та цифрового документообігу. Автоматизація цього процесу дає змогу значно скоротити час введення даних, зменшити ймовірність помилок та підвищити загальну ефективність обробки документів.

Об’єктом роботи є процес автоматичного розпізнавання текстових даних з графічних зображень документів.

Метою роботи є розробка методу розпізнавання документів на основі сучасних технологій штучного інтелекту, який дозволяє автоматизувати витяг інформації з документів на зображеннях.

Для досягнення мети необхідно вирішити такі завдання:

– аналіз існуючих методів виявлення документів на зображеннях і обрати найефективніший підход;

– дослідити інструменти комп’ютерного зору для попередньої обробки зображень і нормалізації області документа;

– проаналізувати можливості сучасних систем OCR для читання тексту з документів;

– реалізувати обробку зображення, нормалізацією, покращення якості фільтрацією;

– реалізувати модуль розпізнавання тексту на основі Tesseract OCR;

– створити систему витягування структурованих даних із тексту документа;

– розробити графічний інтерфейс користувача для завантаження зображення та перегляду результатів.

2 ПРОЄКТУВАННЯ СИСТЕМИ АВТОМАТИЧНОГО РОЗПІЗНАВАННЯ ДОКУМЕНТІВ ІЗ ЗАСТОСУВАННЯМ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ

2.1 Проєктування системи роботи

Для реалізації функціональності системи було розроблено покрокову модель. Вона охоплює всі ключові етапи. Завантаження зображення, виявлення й нормалізацію області документа, попередню обробку для підвищення якості, розпізнавання текстової інформації та її подальше представлення у придатному для використання форматі (рис. 2.1).

Рисунок 2.1 – Блок-схема виконання роботи

Такий підхід дозволяє досягти високої точності та стабільності при зчитуванні персональних даних з документів, що є критично важливим для автоматизованих систем обробки документів.

Для кожного етапу буде обрано свій метод.

2.2 Знаходження документа

У цій системі користувач буде сам заливати зображення з документом, і часто це буде не ідеальна фотографія. Для того щоб модель змогла зчитати дані з документа, насамперед зображення потрібно нормалізувати. Тобто розгорнути, щоб модель змогла прочитати текст, і маштабувати, щоб на виході була чисте зображення документа, без зайвого. Цей етап є критичним, адже точність подальшого розпізнавання тексту значною мірою залежить від якості обрізаного фрагмента документа. Якщо передати до OCR зображення, яке містить багато зайвого, наприклад фон, руки, стіл або інші предмети, точність розпізнавання різко падає. Було проведено аналіз двох сучасних рішень з відкритими API: Grounding DINO та CLIP (рис. 2.2).

Рисунок 2.2 – Логотипи Grounding DINO та OpenAI CLIP

2.2.1 Grounding DINO

Grounding DINO це сучасна модель об’єктного детектування, яка здатна знаходити на зображеннях об’єкти, задані у вигляді текстового запиту. Тобто, достатньо просто вказати «знайди документ» або будь-який інший об’єкт, і модель здійснить пошук відповідних об’єктів без необхідності попереднього навчання на цьому класі. Grounding DINO поєднує підхід трансформерів, обробку природної мови і зображень, та використовує попередньо натреновану модель для формування зв’язків між текстом і візуальними об’єктами (рис 2.3). На відміну від класичних об’єктних детекторів, які потребують навчання на фіксованому наборі класів, Grounding DINO універсальніший. Це дає змогу працювати з довільними запитами, зокрема тими, що не були передбачені під час тренування моделі [20].

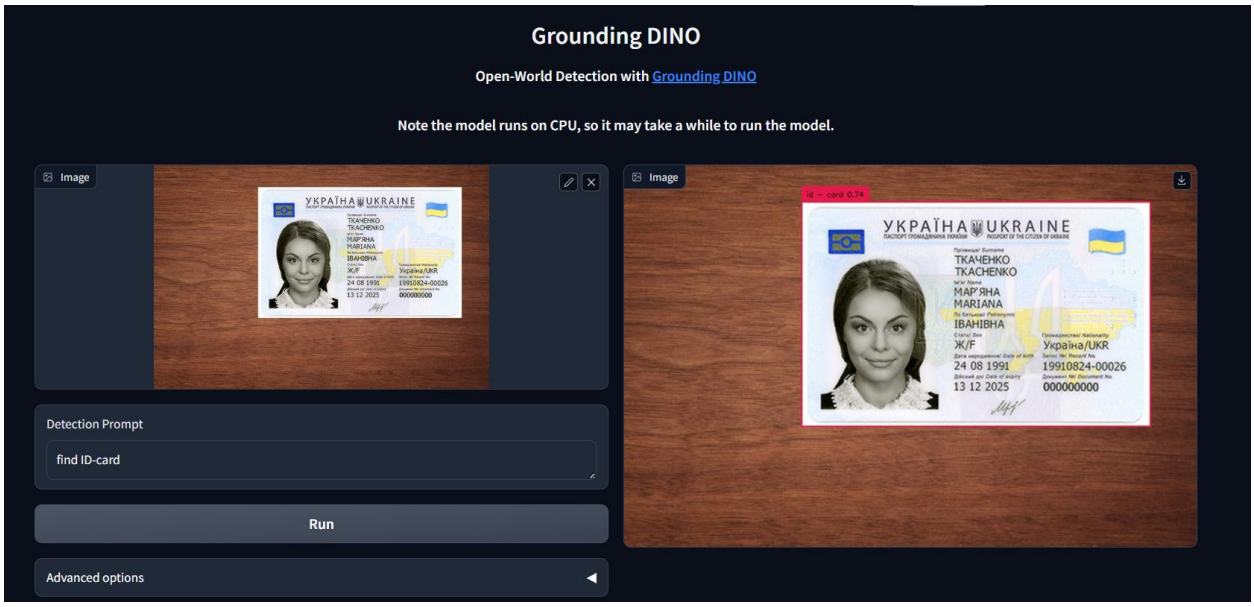

Рисунок 2.3 – Праця Grounding DINO

У демонстраційному просторі на Hugging Face модель показала хороші результати при пошуку документів на зображеннях. Вона стабільно виділяє прямокутник навколо документа. Особливо корисною є властивість моделі виділяти не просто зону з текстом, а саме об’єкт, який найбільше відповідає

заданому семантичному запиту. Це дає змогу більш точно обрізати зображення для подальшого аналізу. У процесі тестування модель повертала координати знайдених об’єктів, які можна було легко інтегрувати з OpenCV для виділення документа, навіть якщо на зображенні були й інші об’єкти, обличчя, руки, меблі тощо.

2.2.2 CLIP

CLIP це модель яка створена OpenAI. Вона також поєднує обробку тексту та зображення, але за іншим принципом. Вона не є об’єктним детектором, а натомість створює векторні представлення як для тексту, так і для зображень, після чого обчислює схожість між ними. Таким чином, CLIP дає змогу визначити, наскільки конкретне зображення відповідає заданому текстовому опису. Цей підхід дуже добре працює у задачах класифікації, фільтрації або підбору відповідного зображення серед великої вибірки (рис. 2.4).

CLIP не повертає координати об’єкта, вона працює з цілим зображенням або окремими частинами, які потрібно заздалегідь нарізати вручну. Для задачі знаходження документа це означає, що доведеться розбивати зображення на багато фрагментів, обчислювати схожість кожного з них із запитом і на основі найвищого результату визначати потрібну область. Такий підхід хоч і можливий, але набагато складніший у реалізації, менш точний та має значно нижчу ефективність у порівнянні з прямим виводом, який дає Grounding DINO. Під час тестування CLIP показував добру відповідність при порівнянні «ID-картка» та «не ID-картка», але саме задача локалізації документа залишалася для нього складною без додаткових інструментів [21].

Рисунок 2.4 – Принцип роботи моделі CLIP

2.2.3 Вибір моделі для знаходження документа

Після аналізу та порівняння обох моделей маємо, що Grounding DINO значно краще справляється саме із завданням знаходженням документа, тоді як CLIP більше підходить для задач класифікації чи фільтрації зображень. Grounding DINO дозволяє працювати із запитами природної мови, точно визначає координати документа навіть у складних умовах, таких як слабке освітлення або наявність сторонніх предметів, і без потреби в ручному виділенні регіонів. У той час як CLIP, попри свою загальну потужність, не надає інформації про розташування документа, що критично важливо для подальшого OCR розпізнавання.

Таким чином, для роботи було обрано Grounding DINO як більш точне, гнучке та придатне рішення для знаходження документа на зображеннях.

2.3 Фільтрування зображення

У процесі автоматичного розпізнавання документів важливу роль відіграє попередня обробка зображення, яка передбачає застосування математичних методів фільтрації. Ці методи дозволяють усунути шум, покращити контраст, вирівняти освітлення та виділити релевантні області документа для подальшого OCR аналізу.

Метою цієї обробки є забезпечення максимально чіткого й контрастного відображення текстових елементів на зображенні документа, що дозволяє системі розпізнавання досягати високої точності та знижувати частоту помилок. Особливо це актуально в умовах низької якості вхідних зображень, тіней, перекосів, неідеального освітлення або наявності шуму.

Просторові фільтри працюють шляхом аналізу локального оточення кожного пікселя зображення. Вони використовують спеціальні ядра, які ковзають по зображенню, змінюючи значення пікселів відповідно до їхніх

сусідів. Результатом є згладжене зображення, у якому зменшено рівень шуму та приглушено різкі переходи. Одним із найбільш поширених підходів є використання середнього значення пікселів у межах локального вікна, що дозволяє зменшити дрібні випадкові коливання, однак може розмивати тонкі контури символів [22].

Застосування згладжувальних фільтрів істотно підвищує якість подальшої бінаризації, забезпечує кращу сегментацію тексту, а також підвищує точність символів, розпізнаних OCR. Таким чином, це є важливим кроком у підготовці вхідних даних і гарантує стабільну роботу всієї системи розпізнавання документів [23].

2.3.1 Адаптивна бінаризація зображення

Одним із ключових етапів попередньої обробки зображення в системах розпізнавання документів є його бінаризація, це перетворення з повнокольорового або градацій сірого у двоколірне, чорно-біле зображення. Це необхідно для того, щоб виділити текстові символи з фону та забезпечити максимальну точність при їх подальшому аналізі алгоритмами OCR (рис. 2.5).

Звичайна бінаризація встановлює одне порогове значення для всього зображення, що працює лише у випадках, коли фон є рівномірним. Проте в реальних умовах при фотографуванні документів за допомогою мобільних пристроїв, у присутності тіней, змінного освітлення чи артефактів глобального підходу недостатньо. У таких випадках застосовуються методи адаптивної бінаризації, які розраховують поріг окремо для кожної області або навіть для кожного пікселя зображення.

Рисунок 2.5 – Адаптивна бінаризація зображення

2.3.2 Метод Отсу

Метод Отсу є одним з найбільш відомих підходів до глобальної бінаризації зображення. Він автоматично визначає оптимальне порогове значення на основі аналізу гістограми яскравості пікселів. Основна ідея полягає в тому, щоб знайти поріг, який мінімізує внутрішньокласову дисперсію між двома сегментами: фоном і об’єктами, наприклад, текстом. Критерій оптимальності має вигляд формули (2.1).

$$\sigma_w^2(T) = q_1(T) \cdot \sigma_1^2(T) + q_2(T) \cdot \sigma_2^2(T), \quad (2.1)$$

де T – значення порогу;

$q1$, $q2$ – частки пікселів, що належать до класів;

$\sigma1$, $\sigma2$ – дисперсії всередині класів.

Метод ефективний при обробці документів з рівномірним освітленням і чітким фоном. Він часто використовується для сканованих чорно-білих документів, а також у випадках, коли швидкість обробки є пріоритетом [24].

2.3.3 Метод «Sauvola»

Метод «Sauvola», або локальна адаптивна бінаризація, застосовується у випадках, коли освітлення зображення є нерівномірним, а фон неоднорідним (рис. 2.6).

Рисунок 2.6 – Приклад методу «Sauvola»

Це характерно для документів, сфотографованих з мобільного пристрою або за поганого освітлення. Порогове значення для кожного пікселя визначається індивідуально на основі локальних статистик за формулою (2.2).

$$T(x,y) = M(x,y) \cdot \left[1 + k \cdot \left(\frac{S(x,y)}{R} - 1\right)\right], \quad (2.2)$$

де $M(x,y)$ – локальне середнє значення;

$S(x,y)$ – стандартне відхилення в локальному вікні;

R – динамічний діапазон (часто 128);

k – параметр чутливості (зазвичай 0.5).

Метод дозволяє зберігати дрібні елементи тексту й ефективно відокремлювати символи від нерівномірного фону. Він є стандартом для адаптивної обробки зображень у сучасних OCR-системах [25].

2.3.4 Порівняння методів, вибір оптимального варіанту

Адаптивна бінаризація це не лише технічна необхідність, а й важливий крок для підвищення точності розпізнавання документів у нестабільних реальних умовах. Вибір конкретного методу визначається вимогами до продуктивності, типом вхідних зображень та обмеженнями апаратного забезпечення. У багатьох промислових застосуваннях спостерігається тенденція до використання адаптивних або гібридних методів, які поєднують точність із швидкодією.

Вибір між методом Отсу і «Sauvola» залежить від конкретних умов. Якщо маємо справу з однорідно освітленими сканами, то метод Отсу забезпечує високу швидкість і добру точність. Натомість у випадку фото-документів метод «Sauvola» забезпечує значно вищу якість бінаризації, хоча й вимагає більше обчислювальних ресурсів.

У практиці розпізнавання документів часто використовуються комбіновані підходи. У системі спочатку відбувається попередня фільтрація за Отсу, а потім уточнення методом «Sauvola» у зонах низької контрастності. Такий гібрид дозволяє досягати стабільної точності за різних умов зйомки й формату документа.

2.3.5 Ерозія та розширення

Так само в системі використовуються ерозія та розширення, це фундаментальні операції морфологічної фільтрації (рис. 2.7).

Рисунок 2.7 – Ерозія та розширення на практиці

Коли під час фільтрації літери виходять злиплими, нечитабельними, використовується ерозія. Вона зменшує об’єкти на зображенні, стирає пікселі по контуру, що призводить до звуження символів і видалення дрібного шуму. Формально, ерозію можна описати за формулою (2.3).

$$A \ominus B = \{z \mid B_z \subseteq A\}, \tag{2.3}$$

де *A* – бінарне зображення;

B – структурний елемент;

Bz – зсув структурного елементу.

Розширення, навпаки, використовується в тих випадках, коли літери після фільтрації виходять розривчастими. Воно додає пікселі до меж об’єктів, це дозволяє заповнити дрібні розриви в літерах і покращити з’єднання частин тексту. Розширення робиться за формулою (2.4)

$$A \oplus B = \{z \mid (B_{-z} \cap A) \neq \emptyset\}. \quad (2.4)$$

Розширення відновлює цілісність символів після попередніх етапів обробки. Комбінація цих двох операцій дозволяє виконувати більш складні трансформації зображення [26].

2.3.6 Методи підвищення чіткості

У складних випадках комбінують кілька методів покращення зображення, оскільки жоден із них окремо не забезпечує стабільно високої якості для всіх типів документів. Найбільш ефективним підходом є побудова каскадних або модульних конвеєрів обробки, у яких кожен етап виконує вузько спеціалізовану задачу.

Окремої уваги заслуговує послідовне застосування фільтраційних операцій із проміжною візуалізацією, яка дозволяє на кожному етапі контролювати якість перетворень.

В окремих випадках доцільно використовувати фільтри з різними розмірами вікна або ядра, адаптуючи їх під конкретну ділянку зображення. Наприклад, великі структури, як фонові плями або тіні, ефективно згладжуються великим ядром, у той час як для обробки тексту з тонкими лініями доцільніше використовувати малі розміри фільтра, щоб уникнути розмиття символів. Динамічне масштабування параметрів фільтра залежно

від локального контексту дозволяє збалансувати усунення шуму з одночасним збереженням деталей.

Сучасні підходи також включають машинне навчання для оптимізації параметрів кожного етапу. Наприклад, глибокі згорткові нейронні мережі можуть передбачати, які фільтри слід застосувати до конкретного регіону документа. Це дозволяє зменшити кількість ручного налаштування й покращити узгодженість результатів.

Деякі сучасні системи йдуть ще далі та застосовують методи сегментації зображення за допомогою попередньо навчених моделей, які автоматично розпізнають тип вхідного документа, наприклад, паспорт, посвідчення, рахунок, і адаптують схему фільтрації відповідно до його структури. Такі системи можуть не лише динамічно підбирати параметри обробки, а й пропускати зайві етапи у випадках, коли зображення вже має прийнятну якість. Це дозволяє оптимізувати продуктивність і уникнути надлишкових перетворень, які могли б спотворити важливу інформацію.

У результаті підвищується точність сегментації символів, зменшується кількість артефактів, а якість вхідного зображення стає придатною для обробки навіть у складних умовах. Комбіновані методи часто реалізуються у вигляді попередньо налаштованих конвеєрів обробки зображень у бібліотеках типу OpenCV, PIL або Tesseract OCR API, що значно спрощує їх впровадження в готові системи розпізнавання документів. Такі гібридні підходи дозволяють адаптувати систему до різних типів вхідних джерел і забезпечують високу гнучкість при зміні умов обробки [27].

2.4 Знаходження тексту

Після успішної локалізації документа і його фільтрацію, наступним етапом є розпізнавання текстової інформації, що міститься в ньому. У

процесі реалізації були протестовані декілька підходів, які сьогодні активно використовуються в задачах аналізу зображень і сканованих документів.

2.4.1 OpenCV + Contour Analysis

Одним з базових підходів до знаходження тексту є використання бібліотеки OpenCV, яка дає змогу обробити зображення й виявити області, що потенційно містять текст. Зокрема, застосовується метод обробки контурів, тобто зображення переводиться у відтінки сірого, порогове значення або адаптивна бінаризація дозволяє виділити темні символи на світлому фоні, а потім за допомогою функцій можна знайти прямокутники, у межах яких ймовірно розміщено текст (рис. 2.8).

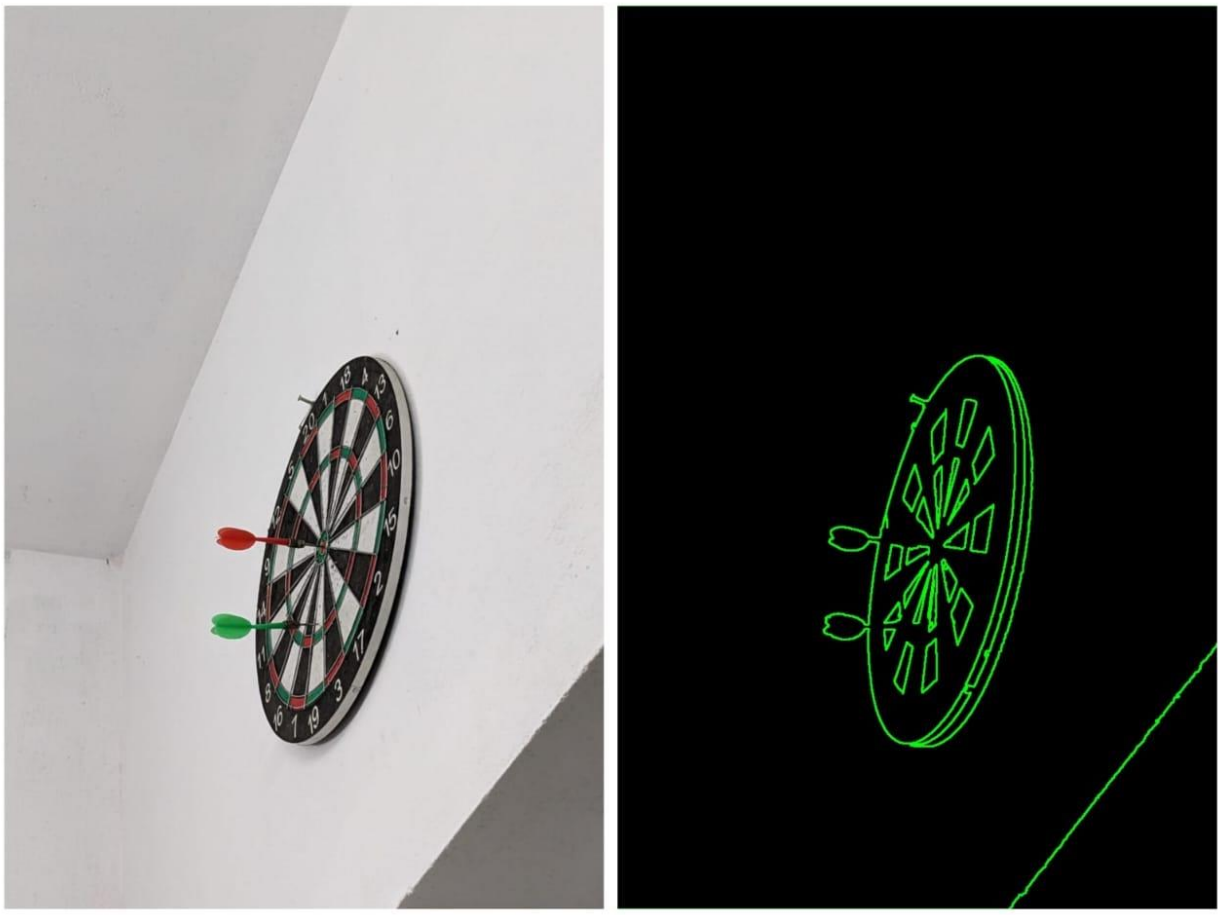

Рисунок 2.8 – Метод Contour Analysis у OpenCV

Цей метод добре працює у випадках, коли фон чіткий, контраст високий, а шрифт стандартний. Проте він не виконує самого розпізнавання тексту, а лише вказує області з текстом. Для його зчитування потрібна окрема OCR система. Крім того, при слабкому освітленні, складному фоні чи перекошених документах точність різко знижується [28].

2.4.2 EasyOCR

EasyOCR це сучасна Python бібліотека, яка поєднує традиційні та глибокі нейронні методи. Вона підтримує багато мов, включаючи українську, й здатна працювати без складного налаштування. EasyOCR добре справляється з друкованими текстами, навіть якщо вони розміщені під кутом або мають артефакти (рис. 2.9).

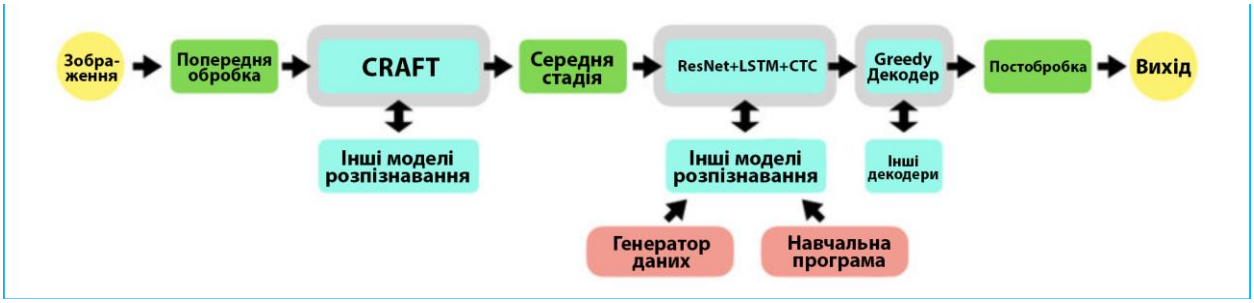

Рисунок 2.9 – Структура EasyOCR

У ході експериментів EasyOCR показав хороші результати на зразках паспортів. Однак головним недоліком стало те, що модель має досить громіздку структуру, важче адаптується під конкретні документи, і потребує більше ресурсів (особливо GPU). Також деякі специфічні символи кирилицею були розпізнані некоректно. Попри загальну зручність, вона поступилася в точності при читанні важливих полів документа [29].

2.4.3 Tesseract OCR

Tesseract OCR це одна з найпопулярніших систем для розпізнавання тексту, підтримувана компанією Google. Вона є відкритою, безкоштовною, активно оновлюється, підтримує понад 100 мов, зокрема українську та англійську, а також дозволяє об’єднувати декілька мов одночасно.

Tesseract було інтегровано в систему з використанням бібліотеки pytesseract, що надає зручний інтерфейс для Python. Для покращення точності була реалізована спеціальна обробка зображення перед передачею в OCR зокрема масштабування, згладжування, бінаризація та шумозниження за допомогою OpenCV. Після цих кроків Tesseract стабільно розпізнає навіть слабоконтрастні фрагменти документа [30]. Також важливо, що система дозволяє витягувати текст у форматі звичайного рядка, що легко обробляється регулярними виразами для подальшої структуризації (рис. 2.10).

Рисунок 2.10 – Структура Tesseract OCR

Після порівняння всіх розглянутих методів було обрано Tesseract OCR як найбільш просте, надійне, гнучке та адаптивне рішення. Воно поєднує точність, підтримку необхідних мов, простоту інтеграції та високу

стабільність при роботі з реальними зображеннями. На відміну від EasyOCR, Tesseract краще справляється з українськими паспортами, де частина тексту дублюється англійською. У поєднанні з попередньою обробкою зображень Tesseract дозволяє з високою точністю зчитувати прізвище, ім’я, дату народження, стать та інші поля документа.

2.5 Підсумкова система роботи та основні етапи роботи програми

Після порівнянь і виборів потрібних для роботи модулів, на рисунку 2.11 ми маємо таку блок-схему системи роботи.

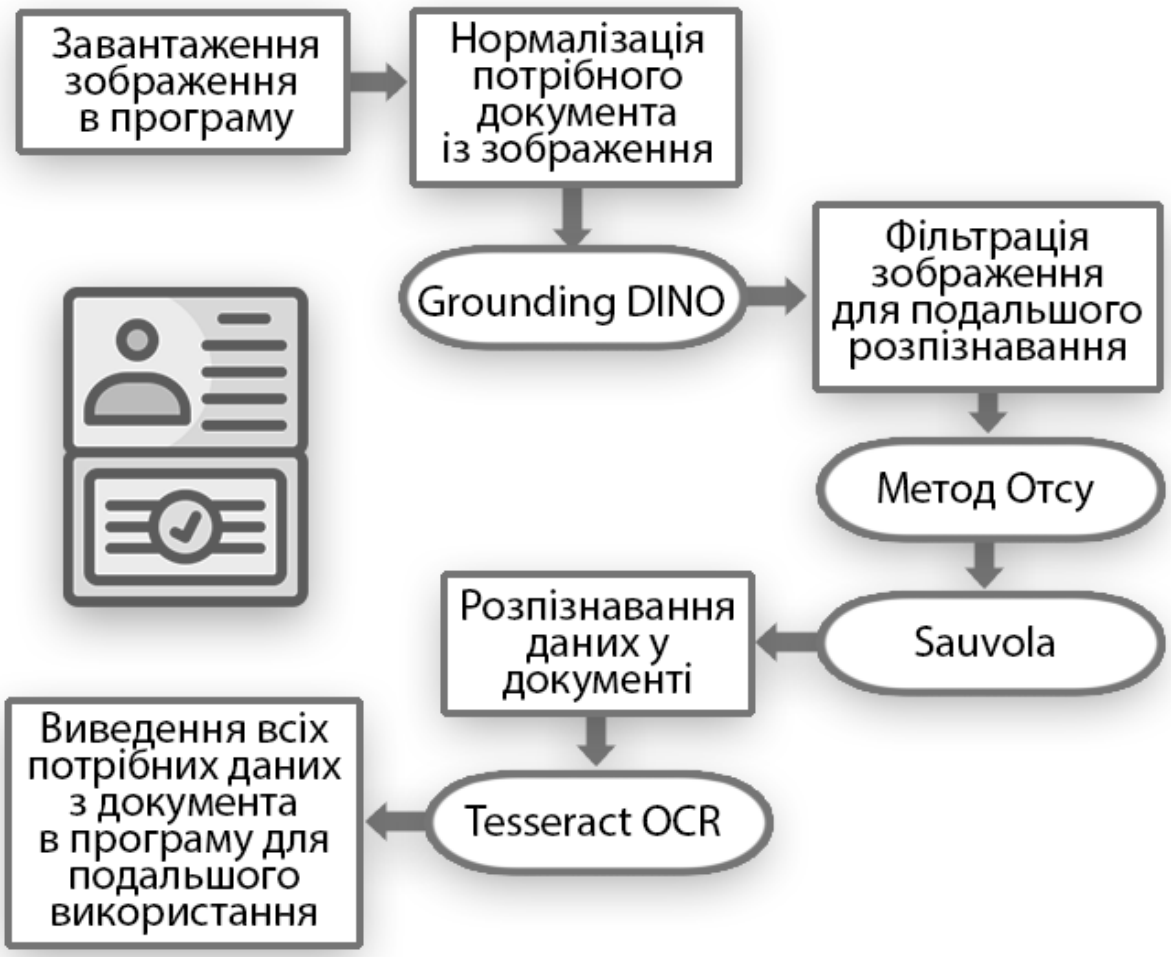

Рисунок 2.11 – Повна блок-схема виконання роботи

У результаті маємо, що спочатку йде завантаження та масштабування зображення, яке відбувається через PIL.Image, після чого воно виводиться у вікно програми. Покращення зображення виконується у функції «enhance_image()», де застосовується перетворення у відтінки сірого, масштабування, бінаризація методом Отсу та фільтрація шуму. На рисунку 2.12 відображено кілька прикладів обробки документів.

Рисунок 2.12 – Обробка документів до та після

Оптичне розпізнавання тексту виконується бібліотекою Tesseract OCR із зазначенням української та англійської мови. Це дозволяє витягувати як українські, так і англійські дані з ID-картки, але в цій програмі це використовується для визначення та виведення лише інформації українською. Парсинг результату відбувається через регулярні вирази. Ключові слова співставляються зі словником, після чого витягуються такі атрибути, як прізвище, ім’я, по батькові, дата народження, номер документа тощо [31]. Форматування дат виконується за допомогою окремої функції «format_date()», що дозволяє привести всі знайдені дати до стандартного вигляду «DD.MM.YYYY» (рис. 2.13).

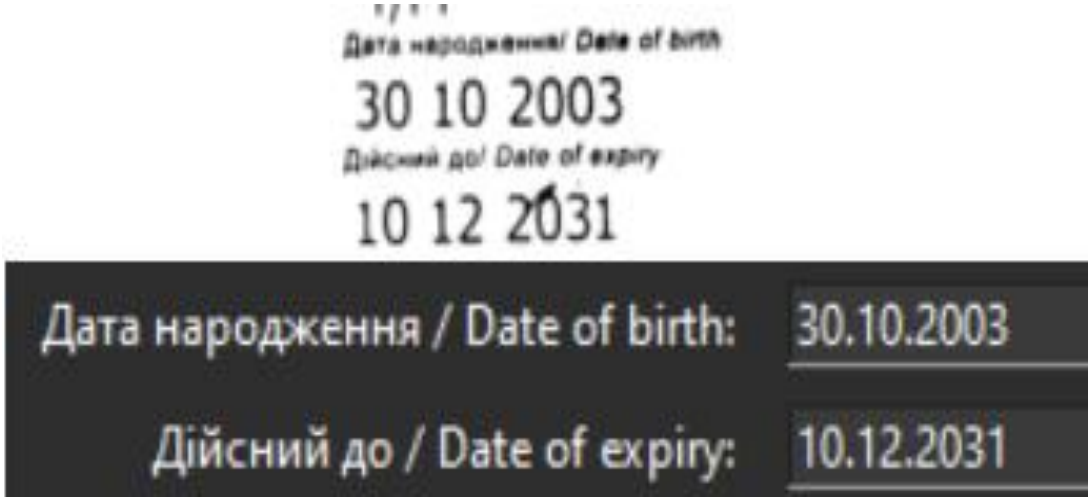

Рисунок 2.13 – Приклад наведення дат до стандартного виду у програмі

Робота зберігає архітектурну чистоту. Логіка обробки зображення і розпізнавання відокремлена від інтерфейсу. Це спрощує подальше тестування, масштабування та можливе використання модуля OCR в інших роботах. Всі етапи повністю автоматизовані, користувачеві потрібно лише завантажити зображення, після чого програма виконує весь процес без ручного втручання.

3 РОЗРОБКА ЗАСТОСУНКУ ДЛЯ АВТОМАТИЧНОГО РОЗПІЗНАВАННЯ ДОКУМЕНТІВ

3.1 Обґрунтування вибору середовища програмної реалізації

У рамках кваліфікаційної роботи розробки методу фільтрації зображень, який є складовою процесу розпізнавання документів, було обрано мову програмування Python та середовище розробки PyCharm (рис. 3.1). Такий вибір зумовлений як технічними, так і практичними чинниками. Вони охоплюють зручність у реалізації, доступ до сучасних бібліотек та здатність ефективно працювати із зображеннями різного типу і складності.

Рисунок 3.1 – Логотип середовища розробки PyCharm

Python на сьогоднішній день є однією з найбільш поширених мов у галузі комп’ютерного зору, машинного навчання й цифрової обробки зображень. Вона має потужну екосистему, в якій особливе місце займає бібліотека OpenCV. Саме вона дає змогу реалізовувати широкий спектр алгоритмів фільтрації, перетворень, морфологічних операцій і багато іншого. У поєднанні з нею ефективно працює бібліотека NumPy, яка значно спрощує маніпуляції з багатовимірними масивами і дозволяє будувати математичні моделі на рівні обробки піксельних структур. Також «scikit-image», що орієнтована на роботу із зображеннями та включає значну кількість готових

реалізацій фільтрів, засобів покращення якості та трансформацій. Для представлення результатів до і після обробки широко використовується Matplotlib, який забезпечує гнучкі інструменти візуалізації.

Додатковою перевагою Python є простота його синтаксису, що значно скорочує час розробки та полегшує відлагодження створених алгоритмів. Водночас ця мова добре інтегрується з популярними фреймворками, такими як TensorFlow чи PyTorch, що відкриває можливість включення елементів штучного інтелекту до проєкту вже на пізніших етапах його розвитку.

Що стосується середовища розробки, PyCharm зарекомендував себе як інструмент, що забезпечує високу ефективність і комфорт при написанні та тестуванні коду. Його функціонал охоплює автоматичне завершення, зручну систему налагодження, роботу з віртуальними середовищами, а також інтеграцію з системами контролю версій. Крім того, можливість швидкого перегляду стану змінних і масивів, а також графічний дебагер суттєво спрощують роботу з візуальними даними, що є надзвичайно корисним при створенні систем на основі комп’ютерного зору. Це надає широкі можливості для реалізації поточних завдань, та створює гнучке підґрунтя для масштабування системи у майбутньому, враховуючи динамічний розвиток бібліотек та інструментів у цій екосистемі [32].

3.2 Програмна реалізація

У рамках реалізації методу автоматичного розпізнавання документів була створена графічна програма, що дозволяє завантажувати фотографії ID-карток, здійснювати їх попередню фільтрацію, розпізнавати текст за допомогою OCR, а також зручно виводити результат у вигляді заповнених полів і JSON структури. Реалізація побудована на бібліотеках tkinter, OpenCV, Tesseract OCR, PIL, NumPy та re.

Tkinter це стандартна бібліотека для створення графічного інтерфейсу для користувача в Python. Вона використовується для створення вікон, кнопок, текстових полів, написів та інших елементів, які бачить користувач. У цій програмі tkinter відповідає за інтерфейс завантаження зображення, відображення його віконно, введення/виведення даних та режиму debug, який буде виводити відредаговане зображення для OCR.

OpenCV це потужна бібліотека для комп'ютерного зору, обробки та аналізу зображень. Тут вона використовується для попередньої обробки зображення, перетворення в градації сірого, масштабування, згладжування, бінаризація методом Отсу та видалення шумів. Це критично важливо для того, щоб підготувати зображення до точного OCR (рис 3.2).

Рисунок 3.2 – Логотип бібліотеки «OpenCV»

Tesseract OCR від Google відповідає за перетворення обробленого зображення в текст. Використовується одночасно українська й англійська мовна модель, що дозволяє розпізнавати документи з двомовними надписами (рис 3.3).

Рисунок 3.3 – Логотип «Google Tesseract OCR»

PIL це бібліотека для базової обробки зображень у Python. Вона використовується для відкриття зображень, зміни їхнього розміру, створення мініатюр, а також передачі зображення в tkinter. Без неї не працювало б візуальне відображення фотографії у вікні програми.

NumPy потрібен для обробки числових масивів, особливо багатовимірних. У OpenCV вона використовується як основа для роботи з зображеннями, адже кожне зображення це масив пікселів. Вона необхідна для коректної роботи зображень.

A re це стандартна бібліотека Python для роботи з регулярними виразами. Вона дозволяє знаходити дати, імена, шаблони документів у тексті, що був витягнутий з OCR. Усі ці бібліотеки працюють як єдиний механізм, одні покращують зображення, інші розпізнають текст, та показує це все користувачу.

3.2.1 Інтерфейс користувача

Графічний інтерфейс створено за допомогою «tkinter». Основне вікно дозволяє користувачу завантажити зображення паспорта (рис. 3.4), та переглянути зображення завантаженого документа.

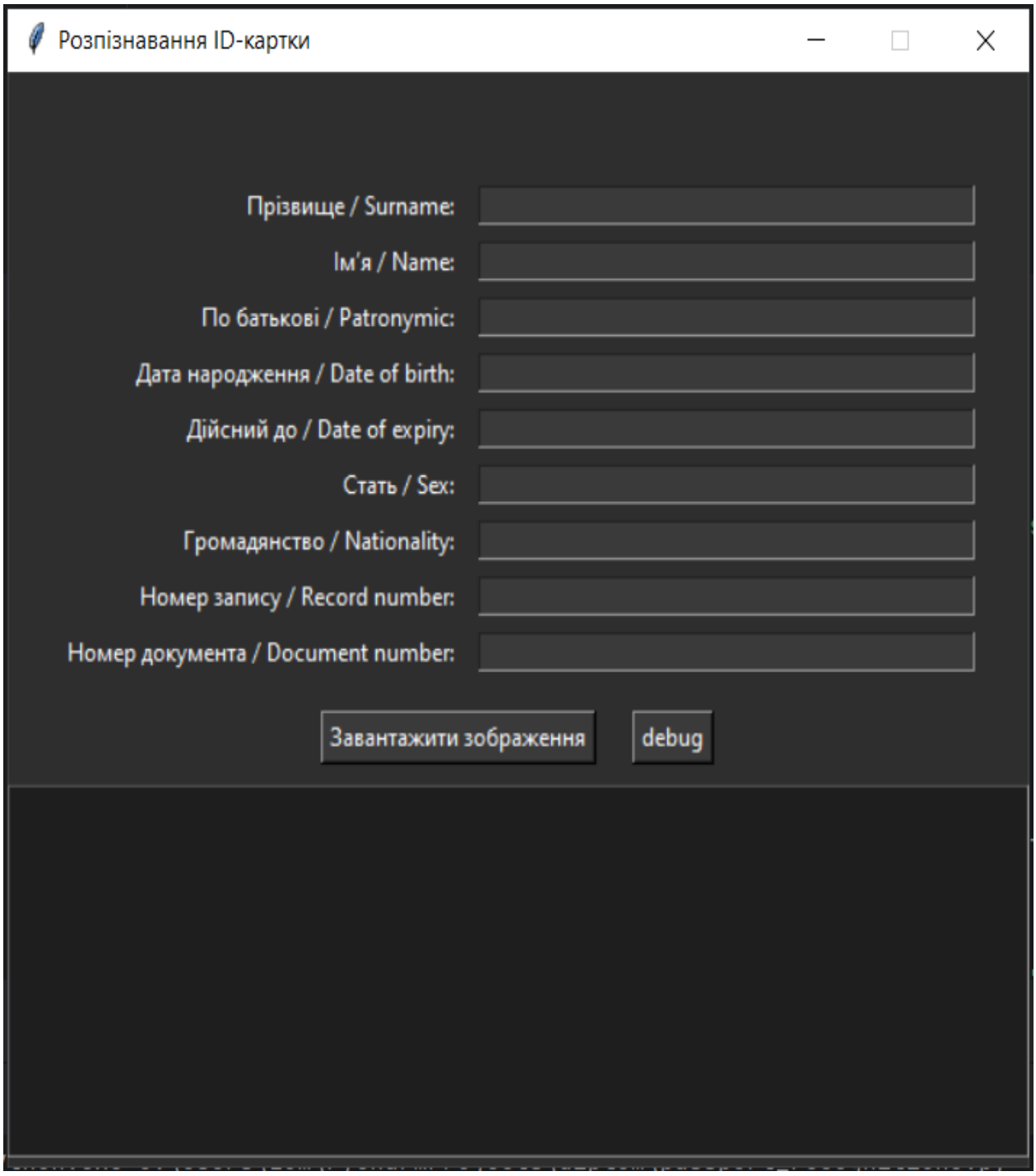

Рисунок 3.4 – Інтерфейс програми під час запуску

Після завантаження зображення відразу автоматично виводяться розпізнані дані у вигляді текстових полів та JSON результат для подальшої інтеграції (рис. 3.5).

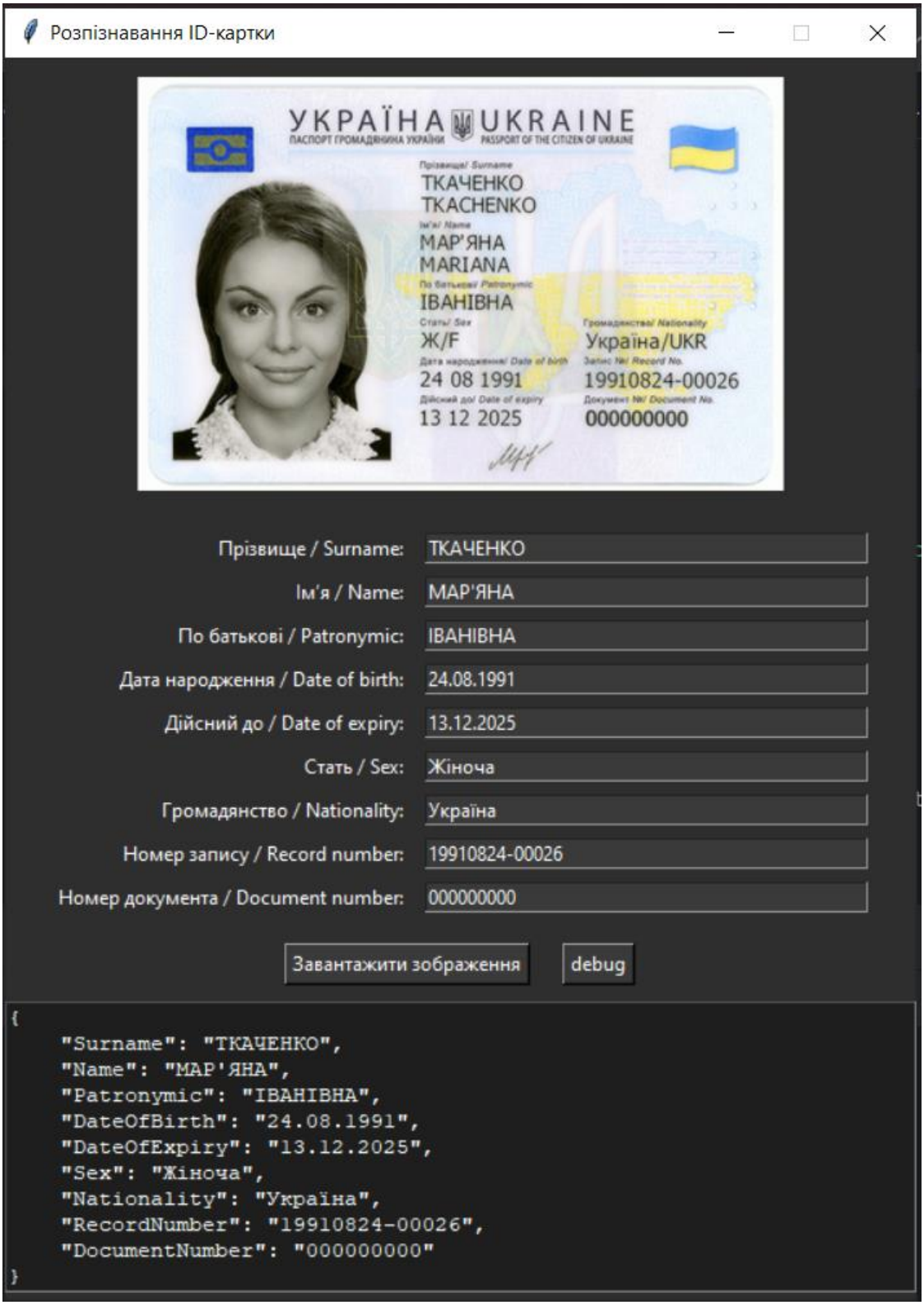

Рисунок 3.5 – Висновки програми після завантаження зображення

Також є кнопка «режим дебагу» при перемиканні якої показується попередньо оброблене зображення, яке використовується для OCR (рис. 3.6).

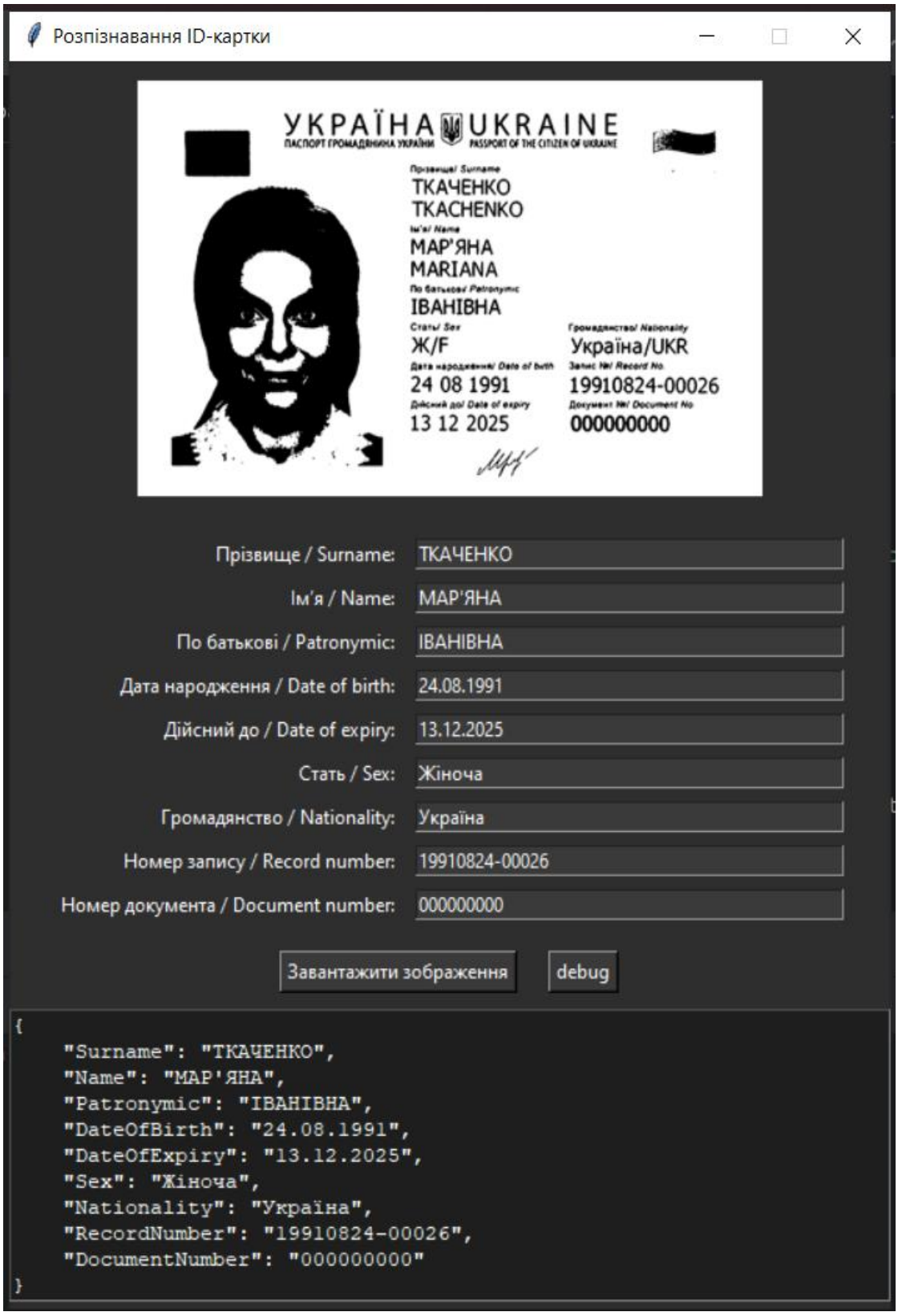

Рисунок 3.6 – Включений «режим дебагу»

3.3 Інструкція користувача

Програма призначена для розпізнавання текстової інформації з ID картки громадянина України. Вона реалізована у вигляді зручного графічного додатку з інтуїтивно зрозумілим інтерфейсом.

Після запуску програми у головному вікні буде всього дві кнопки, слід натиснути кнопку «Завантажити зображення». Відкриється стандартне вікно для вибору файлу (рис. 3.7). Потрібно обрати фотографію ID картки у форматі JPG або PNG. Після завантаження зображення автоматично з’явиться у верхній частині вікна програми (рис. 3.8).

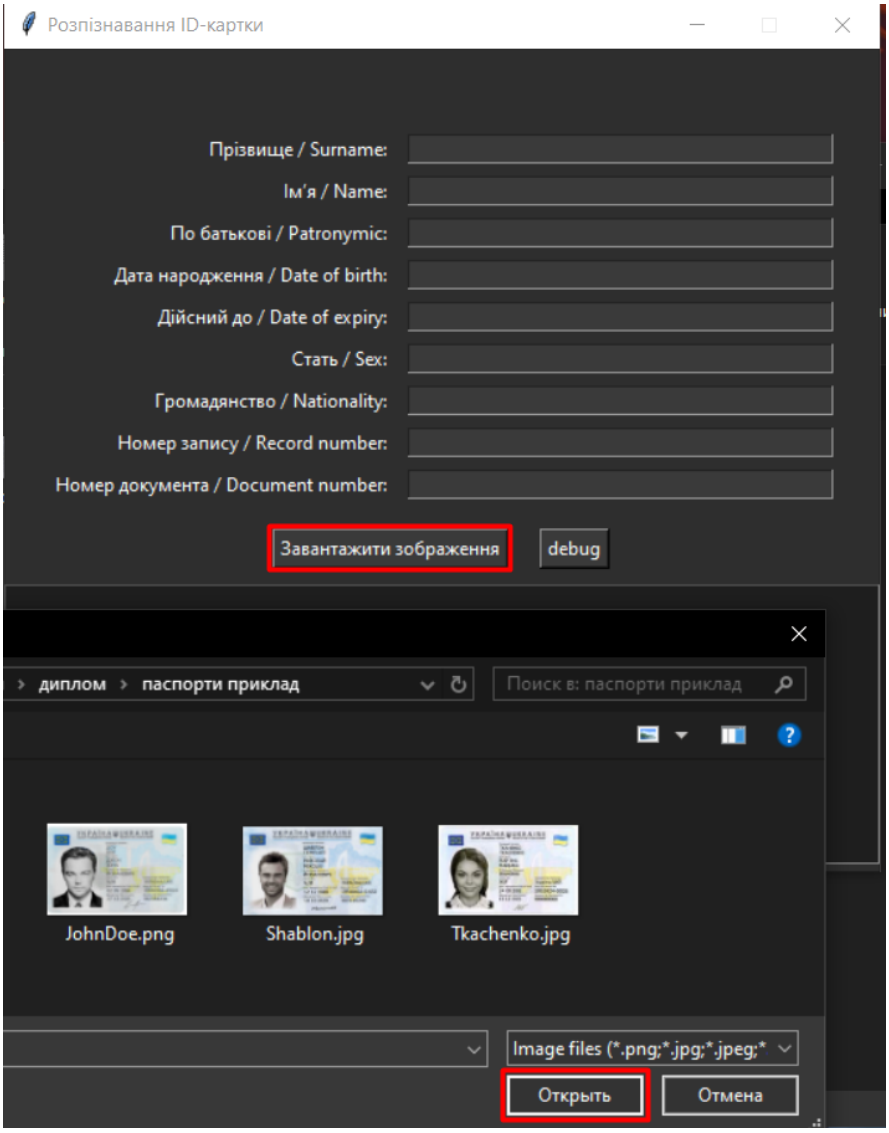

Рисунок 3.7 – Вибір зображення у програмі

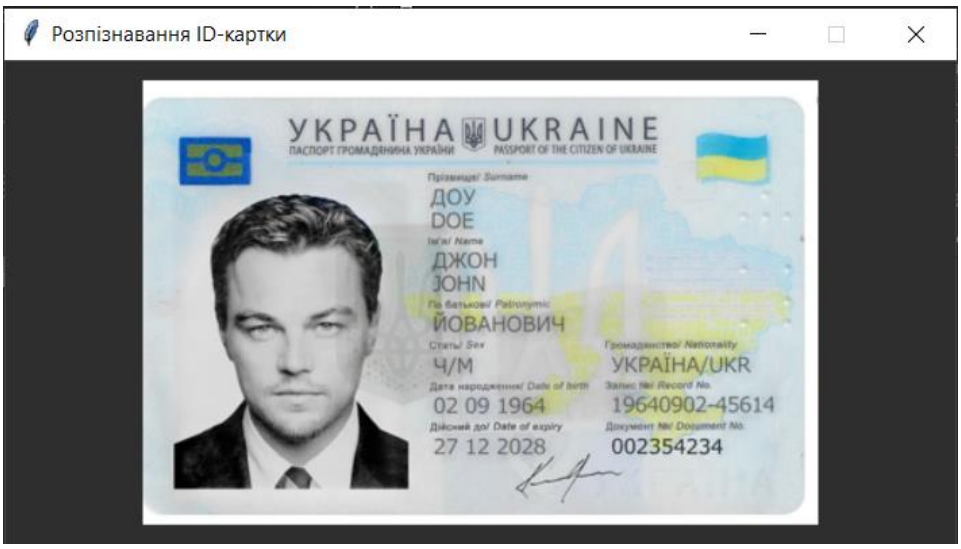

Рисунок 3.8 – Відображення вибраного зображення

Програма автоматично виконає всі етапи обробки зображення та розпізнавання тексту без додаткових дій. Отримані значення будуть відображені у відповідних текстових полях у центрі вікна (рис. 3.9). Якщо певне поле залишилось порожнім, це може бути пов’язано з якістю зображення або відсутністю відповідного тексту.

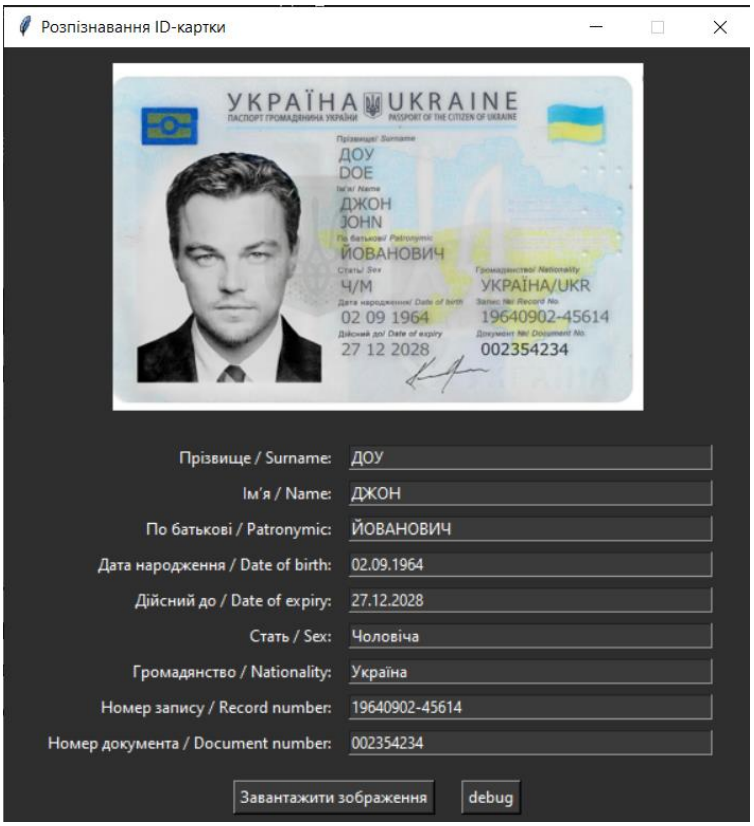

Рисунок 3.9 – Автоматичний вивід усієї інформації в потрібних полях

У нижній частині вікна буде автоматично згенерований JSON файл, що містить ті самі дані у структурованому вигляді (рис. 3.10). Це зручно для збереження або інтеграції результату у зовнішні системи.

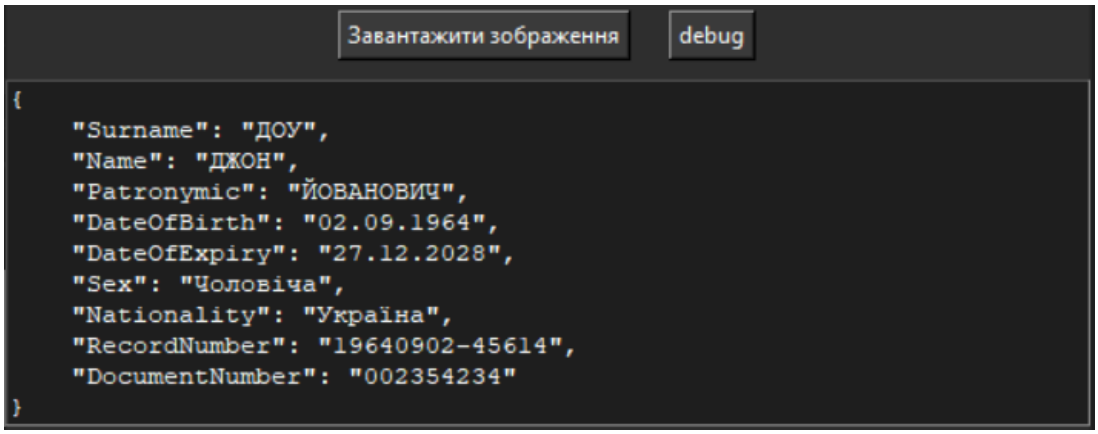

Рисунок 3.10 – Автоматичний вивід усієї інформації у форматі JSON

Також можна натиснути кнопку «debug» для переключення зображення на оброблену версію, яка була використана для OCR (рис. 3.11). Це дозволяє візуально оцінити ефективність фільтрації та підготовки зображення. Повторне натискання повертає початковий вигляд.

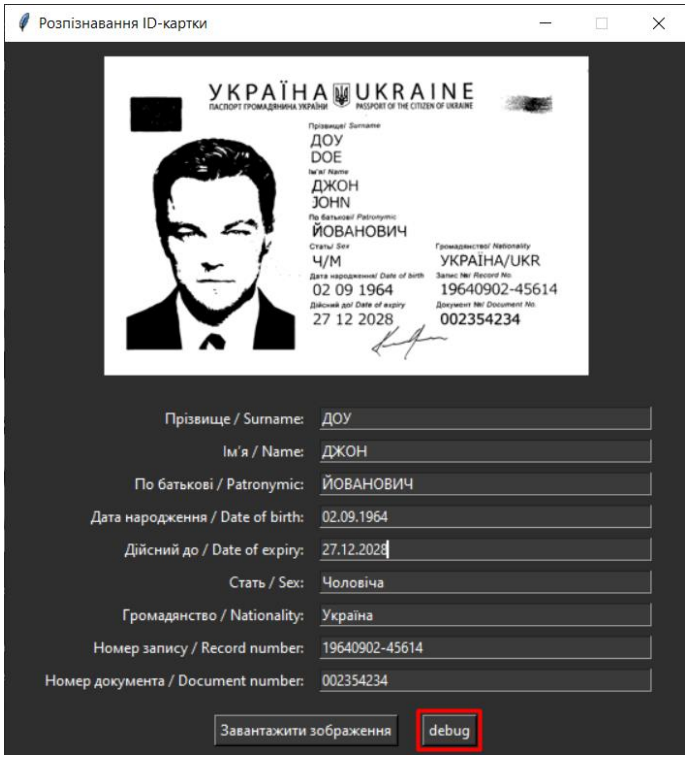

Рисунок 3.11 – Відредаговане зображення після натискання кнопки «debug»

Якщо будь-яке з полів з якоїсь причини заповнено некоректно, його можна вручну відредагувати перед копіюванням або збереженням JSON. Програма працює офлайн, не потребує доступу до мережі інтернет, так що ніякі файли, інформація, документи нікуди не йдуть, все залишається в рамках комп’ютера, на якому запущена програма. Всі оброблені зображення для тесту зберігаються у кореневій папці під назвою «debug_passport.png».

Інтерфейс є максимально простим і не вимагає спеціальних навичок. Ще один приклад роботи програми показано на рисунку 3.12.

Рисунок 3.12 – Повний скріншот програми з іншим завантаженим прикладом

3.4 Проблеми які можуть виникнути

Звичайно, програма не ідеальна. Вона може помилятися або зовсім не читати документ, якщо є якісь проблеми зі сканом або фотографією документа, який завантажує користувач.

Здебільшого всі проблеми з’являються через занадто низьку якість завантажуваного зображення. На прикладі рисунка 3.13 текст після редагування зображення зовсім нечитабельний для моделі.

Рисунок 3.13 – Приклад непрацюючої моделі через дуже погану якість зображення

Іноді, через погане освітлення або перешкоди на завантаженому зображенні перекривається текст після його редагування, що теж створює проблему для моделі, яка не може визначити те, що їй потрібно (рис 3.14).

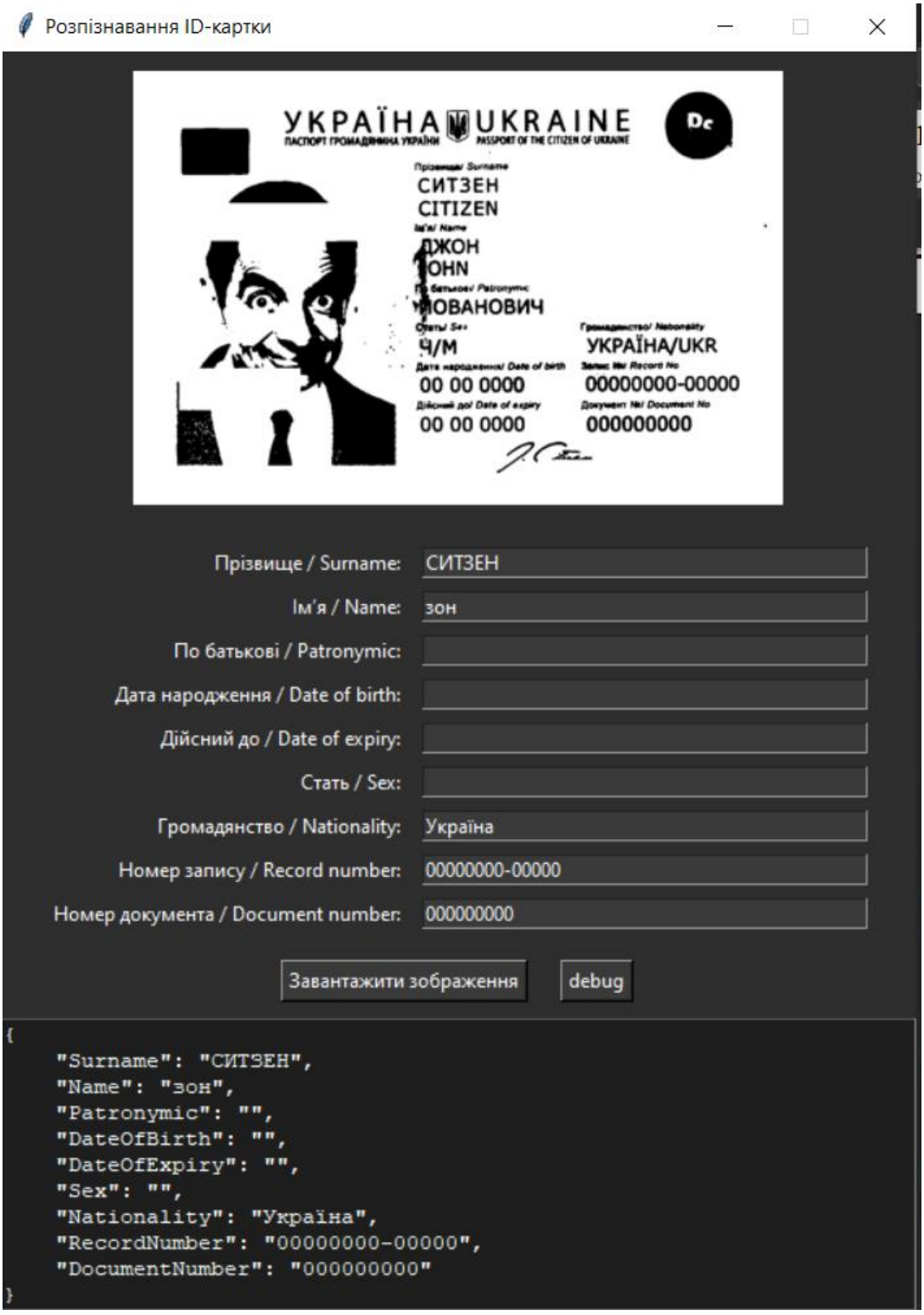

Рисунок 3.14 – Приклад непрацюючої моделі погане освітлення або перешкоди

На рисунку 3.15 показано як модель не може визначити текст із завантаженої картинки через дуже погану якість, і виводить у програму тільки ті ділянки, які може прочитати.

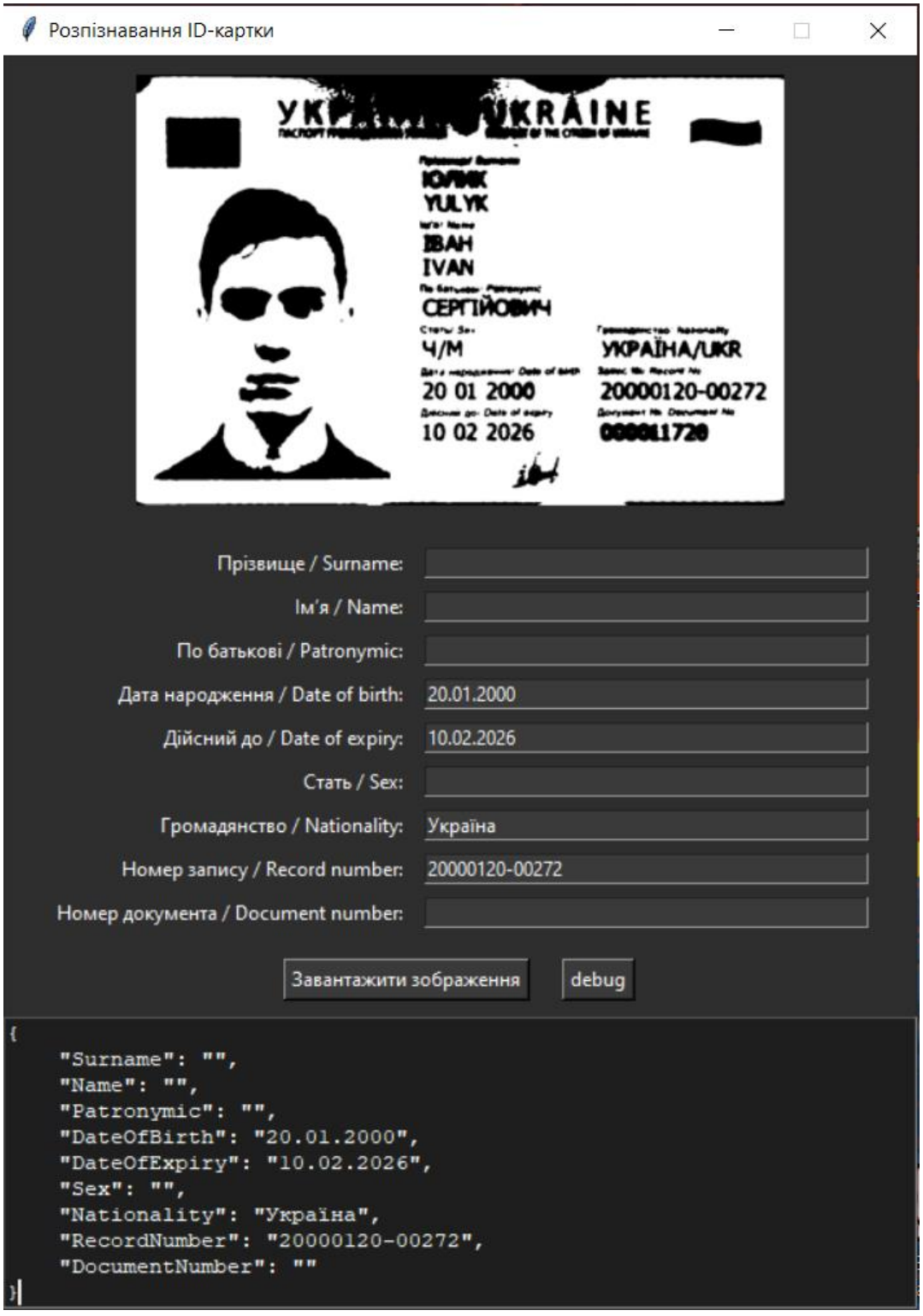

Рисунок 3.15 – Приклад неповноцінного розпізнавання документа через погану якість початкового зображення

На рисунку 3.16 показаний цей самий приклад, але в кращій якості початкового зображення після його «апскейлу» (підвищення якості нейромережами через сторонній сервіс). Як видно, тепер модуль може розібрати весь текст, і виводить усі потрібні дані в поля програми.

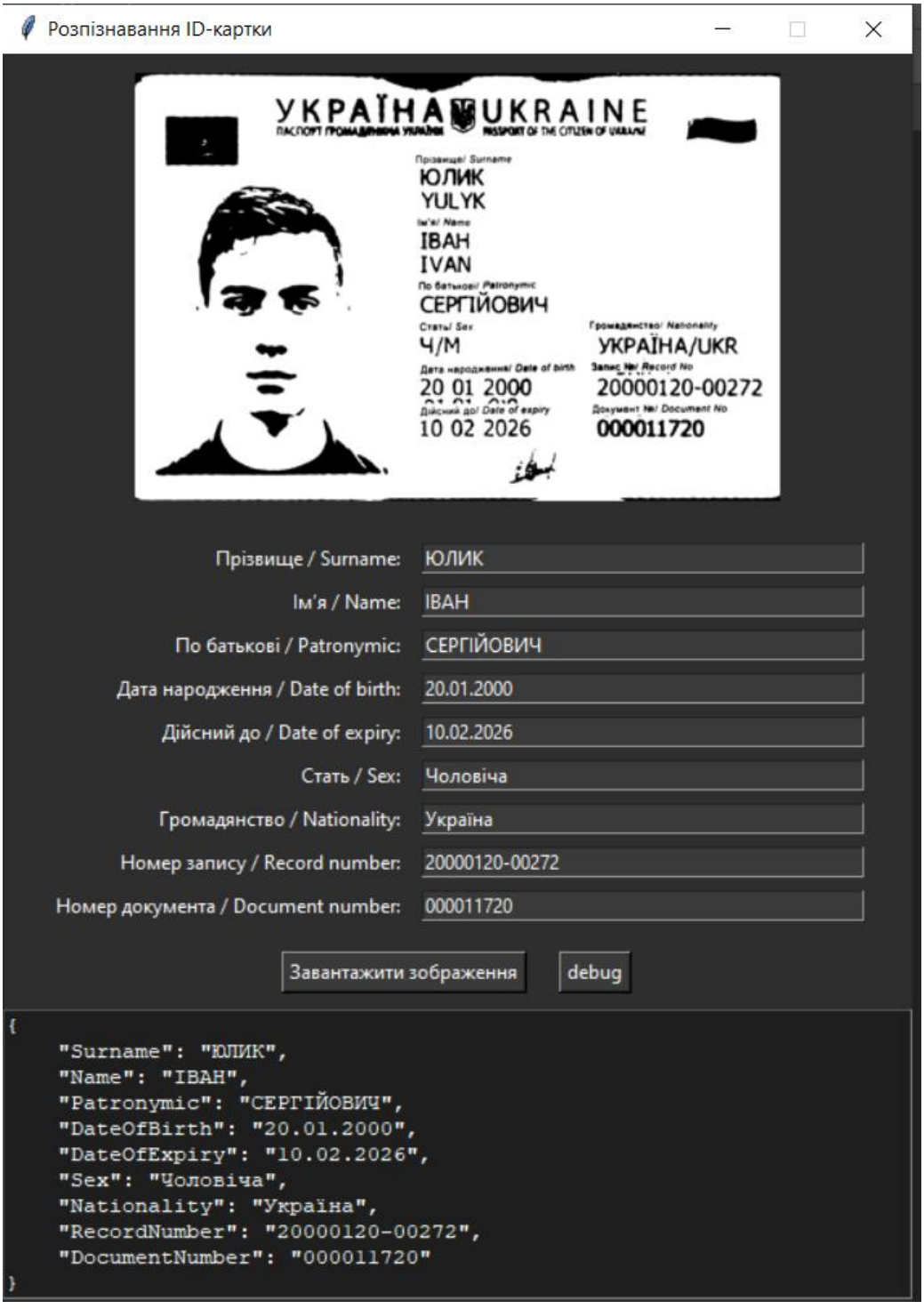

Рисунок 3.16 – Приклад повноцінного розпізнавання документа після поліпшення якості початкового зображення

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено застосунок для автоматичного розпізнавання даних із документів, зокрема зображень ID-карток, із використанням методів фільтрації зображень та моделей штучного інтелекту. Застосунок забезпечує повний цикл обробки від завантаження зображення до отримання структурованих даних у форматі JSON, що може бути інтегровано в зовнішні системи.

Для створення додатку були вирішені наступні завдання:

– проведено аналіз методів фільтрації зображень, що використовуються в задачах OCR та досліджено сучасні математичні моделі;

– розроблено повнофункціональний графічний застосунок;

– реалізовано систему попередньої обробки зображення для підвищення якості розпізнавання;

– забезпечено витягування основних даних з ID-картки;

– реалізовано механізм виводу у форматі JSON;

– проведено тестування моделі на прикладах тестових та справжніх зображень паспортів для оцінки точності та надійності моделі;

Тестування показало, що для кращої роботи моделі потрібно завантажувати скани або фотографії документів з непоганою роздільною здатністю і хорошим освітленням, інакше не факт, що всі дані виведе правильно. Але в додатку спокійно можна редагувати, доповнювати виведену інформацію. Наприклад, для уточнення у форматі JSON, якщо для роботи є тільки одне зображення в не найкращої якості.

У майбутньому система може бути розширена шляхом поліпшення інтеграції моделі LLM для перевірки та уточнення контексту, підтримки інших типів документів та створення інтерфейсу для мультиплатформенного використання, наприклад для браузеру або додатку на мобільний телефон.

Результати роботи можуть бути використані в адміністративних та комерційних сферах, де існує потреба в автоматизованому обробленні персональних документів громадян.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ковтуненко, А. Р., Яковлева, О. В., Любченко, В. А., & Янголенко, О. В. (2020). Дослідження сумісного використання математичної морфології та згорткових нейронних мереж для вирішення задачі розпізнавання цінників.

2. Lyashenko, V., Babker, A., & Lyubchenko, V. (2017). Wavelet Analysis of Cytological Preparations Image in Different Color Systems.

3. Ababneh, J., Abu-Jassar, A., Abuowaida, S., Liubchenko, V,. Lyashenko, V. (204). Evaluation of Three Different Operators for Object Highlighting in Medical RGB Images: Canny, Roberts, and LoG in Independent Color Spaces, 2024 25th International Arab Conference on Information Technology (ACIT), 1-7.

4. Hezretov, M. (2021). Budget Tracker Highly Customizable Budgeting Mobile Application (Doctoral dissertation).

5. Kaye, J. J., McCuistion, M., Gulotta, R., & Shamma, D. A. (2014, April). Money talks: tracking personal finances. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 521–530).

6. Кобилін, О.А., & Творошенко, І.С. (2021). Методи цифрової обробки зображень: навч. посібник. *Харків: ХНУРЕ*.

7. Lyubchenko, V., Veretelnyk, K., Kots, P., Lyashenko, V. (2024). Digital image segmentation procedure as an example of an NP-problem, Multidisciplinary Journal of Science and Technology 4 (4) 170-177.

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

9. Яковлева, О. В., & Ковтуненко, О. Р. (2019). Пошук та розпізнавання цінників товарів на зображеннях.

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

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

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

13. Tvoroshenko I., and Gorokhovatskyi V. (2022) The Application of Hybrid Intelligence Systems for Dynamic Data Analysis, *International Journal of Engineering and Information Systems*, 6(2), pp. 40–48.

14. White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., ... & Schmidt, D. C. (2023). A prompt pattern catalog to enhance prompt engineering with chatgpt.

15. Yevstratov, M., Lyubchenko, V., Amer, AJ., Lyashenko, V. (2024). Color correction of the input image as an element of improving the quality of its visualization.

16. Taibi, D., Lenarduzzi, V., & Pahl, C. (2020). Microservices anti–patterns: A taxonomy. Microservices: Science and Engineering, 111–128.

17. Speth, S., Stieß, S., & Becker, S. (2022, March). A saga pattern microservice reference architecture for an elastic SLO violation analysis. In 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA–C) (pp. 116–119). IEEE.

18. Mustafa, SK., Kopot, M., Ahmad, MA., Lyubchenko, V., Lyashenko, V. (2020). Interesting applications of mobile robotic motion by using control algorithms, WARSE.

19. Bosch, O. J., Revilla, M., & Paura, E. (2019). Answering mobile surveys with images: an exploration using a computer vision API. Social Science Computer Review, 37(5), 669–683.

20. Grounding DINO. URL: https://github.com/IDEA-Research/GroundingDINO (дата звернення 07.05.2025).

21. OpenAI CLIP. URL: https://github.com/openai/CLIP (дата звернення 07.05.2025).

22. Brynjolfsson, E., Li, D., & Raymond, L. R. (2023). Generative AI at work (No. w31161). National Bureau of Economic Research.

23. Feuerriegel, S., Hartmann, J., Janiesch, C., & Zschech, P. (2024). Generative ai. Business & Information Systems Engineering, 66(1), 111–126.

24. Carlà, M. M., Gambini, G., Baldascino, A., Giannuzzi, F., Boselli, F., Crincoli, E., ... & Rizzo, S. (2024). Exploring AI–chatbots' capability to suggest surgical planning in ophthalmology: ChatGPT versus Google Gemini analysis of retinal detachment cases. British Journal of Ophthalmology.

25. Floridi, L., & Chiriatti, M. (2020). GPT–3: Its nature, scope, limits, and consequences. Minds and Machines, 30, 681–694.

26. Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., & Tang, J. (2023). GPT understands, too.

27. Uchqun o‘g‘li, BS., Lyubchenko, V., Lyashenko, V. (2023). Pre-processing of digital images to improve the efficiency of liver fat analysis, Center for Tech and Media Research.

28. Uchqun o‘g‘li, BS., Lyubchenko, V., Lyashenko, V. (2023). Image Processing Techniques as a Tool for the Analysis of Liver Diseases, Research Science and Innovation house.

29. Nasution, W. S. L., & Nusa, P. (2021). UI/UX design web–based learning application using design thinking method. ARRUS Journal of Engineering and Technology, 1(1), 18–27.

30. Yakovleva, O., Kovtunenko, A., Liubchenko, V., Honcharenko, V., Kobylin, O. (2023). Face Detection for Video Surveillance-based Security System.

COLINS (3), 69-86 4th International Conference on New Media Studies (CONMEDIA) (pp. 170– 173). IEEE.

31. Zeleniy, O., Rudenko, D., Lyubchenko, V., Lyashenko, V. (2022). Image Processing as an Analysis Tool in Medical Research, IJAAR.

32. Miell, I., & Sayers, A. (2019). Docker in practice. Simon and Schuster.

33. Фурсов А. Д. (2025) Аналіз методів розпізнавання документів для автоматичної авторизації. Радіоелектроніка та молодь у XXI столітті: тези доповідей 29-го Міжнародного молодіжного форуму (Харків, 16–19 квітня 2025 р.). Харків: ХНУРЕ, 2025. Т. 7. С. 158-159.