ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «СИСТЕМА ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ІНФОРМАЦІЇ У ДЕРЖАВНИХ БАЗАХ ДАНИХ НА БАЗІ ШТУЧНОГО ІНТЕЛЕКТУ»

на здобуття освітнього ступеня магістра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Євгеній ПРИЙМАК*
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. КСДМ-61
Євгеній ПРИЙМАК
Ім'я, ПРІЗВИЩЕ
Керівник: Віталій ВОЛОХІН
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Комп'ютерної інженерії

Ступінь вищої освіти Магістр

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
_____Ім'я, ПРІЗВИЩЕ
«_____» _____2025р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Приймак Євгеній Олександрович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Система інтелектуального аналізу інформації у державних базах даних на базі штучного інтелекту»
керівник кваліфікаційної роботи: Віталій ВОЛОХІН, доцент кафедри, к.т.н., доцент ,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «30» жовтня 2025р. №467
2. Строк подання кваліфікаційної роботи «26» грудня 2025р.
3. Вихідні дані до кваліфікаційної роботи: Науково-технічна література, технологічні рамки та протоколи, дизайн і архітектура системи, інтеграція та функціональність, тестування та оцінка.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
    1. Аналіз існуючих методів OCR та виявлення їх обмежень
    2. Дослідження архітектури нейронних мереж
    3. Розробка прототипу алгоритму обробки зображення документів
    4. Доопрацювання та тестування алгоритму, емуляція роботи оператора
5. Перелік ілюстративного матеріалу: *презентація*
    1. Актуальність роботи
    2. Мета, об'єкт, предмет дослідження
    3. Завдання дослідження
    4. Результати досліджень в теоретичній частині
    5. Реалізація прикладу алгоритму обробки даних (1)
    6. Реалізація прикладу алгоритму обробки даних (2)
    7. Реалізація прикладу алгоритму обробки даних (3)
    8. Реалізація прикладу алгоритму обробки даних (4)
    9. Реалізація прикладу алгоритму обробки даних (5)
    10. Висновки

11. Публікації та апробація роботи

6. Дата видачі завдання «01» вересня 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	09.10-11.10.2025	Виконано
2	Обробка матеріалу	11.10-16.10.2025	Виконано
3	Розробка теоретичної частини	16.10-19.10.2025	Виконано
4	Проектування архітектури системи та вибір технологічного стеку	19.10-21.10.2025	Виконано
5	Програмна реалізація модулів обробки документів	21.10-27.10.2025	Виконано
6	Тестування прототипу та аналіз перспектив розвитку	27.10-01.12.2025	Виконано
7	Висновки за результатами аналізу	01.12-06.12.2025	Виконано
8	Розробка презентації	06.12-15.12.2025	Виконано
9	Передзахист	15.12-17.12.2025	Виконано
10	Здача в деканат	20.12-26.12.2025	Виконано

Здобувач вищої освіти _____ Євгеній ПРИЙМАК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Віталій ВОЛОХІН
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістр: 66 стор., 17 рис., 50 джерел.

Мета роботи – розробити і протестувати програмну реалізацю системи інтелектуального аналізу документів на базі згорткових нейронних мереж для підвищення ефективності, швидкості та надійності процесів обробки неструктурованих даних у державних інформаційних системах.

Об'єкт дослідження – система автоматизованої обробки та семантичного аналізу графічних образів документів у інформаційно-аналітичних системах.

Предмет дослідження – моделі, методи та алгоритми комп'ютерного зору та машинного навчання для ідентифікації структури, класифікації типів та вилучення атрибутивної інформації з офіційних документів.

Короткий зміст роботи: У роботі розглянуто розробку та впровадження системи автоматизованої обробки документів для державного сектору, що поєднує технології комп'ютерного зору та глибокого навчання.

Проаналізовано проблематику обробки «темних даних» у державних реєстрах та обмеження класичних OCR-систем. Запропоновано гібридний підхід до розпізнавання тексту, що використовує комбінацію Tesseract OCR та нейромережевих рішень для роботи з українськими документами. Розроблено архітектуру системи, яка включає модуль класифікації на базі ResNet-50, підсистему препроцесингу зображень та веб-інтерфейс для операторів. Реалізований прототип дозволяє автоматизувати введення даних з ID-карток та паспортів, мінімізуючи вплив людського фактору.

КЛЮЧОВІ СЛОВА: ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ, КОМП'ЮТЕРНИЙ ЗІР, МАШИННЕ НАВЧАННЯ, OCR, НЕЙРОННІ МЕРЕЖІ, ДЕРЖАВНІ РЕЄСТРИ, ОБРОБКА ЗОБРАЖЕНЬ, RESNET, PYTHON, АВТОМАТИЗАЦІЯ ДОКУМЕНТООБІГУ.

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 66 pages, 17 figures, 50 sources.

The purpose of the work is to develop and test a software implementation of an intelligent document analysis system based on convolutional neural networks to increase the efficiency, speed, and reliability of unstructured data processing in state information systems.

The object of the research is the system of automated processing and semantic analysis of graphic images of documents in information and analytical systems.

The subject of research is models, methods, and algorithms of computer vision and machine learning for identifying structure, classifying types, and extracting attribute information from official documents.

Summary of the work: The paper discusses the development and implementation of an automated document processing system for the public sector, combining computer vision and deep learning technologies.

The problems of processing "dark data" in state registries and the limitations of classical OCR systems are analyzed. A hybrid approach to text recognition is proposed, using a combination of Tesseract OCR and neural network solutions for working with Ukrainian documents. The system architecture has been developed, which includes a classification module based on ResNet-50, an image preprocessing subsystem, and a web interface for operators. The implemented prototype allows automating data entry from ID cards and passports, minimizing the impact of the human factor.

KEY WORDS: INTELLIGENT DATA ANALYSIS, COMPUTER VISION, MACHINE LEARNING, OCR, NEURAL NETWORKS, STATE REGISTRIES, IMAGE PROCESSING, RESNET, PYTHON, DOCUMENT WORKFLOW AUTOMATION.

ЗМІСТ

ВСТУП 9
РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ДОКУМЕНТІВ 12
1.1 Еволюція підходів до OCR та проблеми обробки неструктурованих даних 12
1.2 Огляд сучасних архітектур нейромереж для задач комп'ютерного зору 17
1.3 Проблематика класифікації та сегментації різнотипних документів у державних реєстрах 21
1.4 Аналіз вимог до безпеки даних при використанні ШІ в держсекторі 24
1.5 Нормативно-правові засади впровадження систем штучного інтелекту в державному секторі України 27
1.6 Аналіз існуючих платформ інтелектуальної обробки документів 30
РОЗДІЛ 2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ПОТЕНЦІЙНОЇ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ 33
2.1 Проектування загальної архітектури системи та обґрунтування модульної структури 33
2.2 Моделювання бізнес-процесів та інформаційних потоків 35
2.3 Обґрунтування вибору технологічного стеку та інструментальних засобів ... 38
2.4 Алгоритмічне забезпечення попередньої обробки зображень 43
3 РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ 43
3.1 Проектування схеми даних та організації сховища інформації 43
3.2 Програмна реалізація серверної частини системи 46
3.3 Програмна реалізація клієнтського інтерфейсу 60
3.4 Тестування роботи програмного рішення 63
3.5 Перспективи розвитку та вдосконалення програмного рішення 69
ВИСНОВКИ 73
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 75
Додаток А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація) 82

ВСТУП

Актуальність теми. Сучасний етап розвитку державного управління в Україні характеризується стрімкою цифровою трансформацією, що втілюється у концепції «Держава у смартфоні». Впровадження екосистеми «Дія», системи електронної взаємодії «Трембіта» та модернізація державних реєстрів створили фундамент для переходу до сервіс-орієнтованої держави. Проте, незважаючи на успіхи у цифровізації фронт-офісів, бек-офіси державних установ досі стикаються з проблемою обробки значних масивів неструктурованої інформації.

Значна частина вхідних даних надходить до державних баз даних у вигляді скан-копій, фотографій документів, PDF-файлів без текстового шару або паперових архівів. Традиційні методи введення такої інформації, що базуються на ручній праці операторів, є часозатратними, фінансово обтяжливими та схильними до помилок внаслідок людського фактору. В умовах воєнного стану, міграції населення та необхідності швидкої верифікації внутрішньо переміщених осіб, навантаження на інформаційні системи значнозросло.

Вирішення цієї проблеми лежить у площині застосування методів штучного інтелекту, зокрема глибокого навчання та комп'ютерного зору. Створення інтелектуальної системи, здатної не лише зчитувати текст, а й класифікувати тип документа, виділяти семантичні зони та перевіряти дані в реальному часі, дозволить автоматизувати рутинні операції. Таким чином, тема магістерської роботи, присвячена розробці програмного рішення для аналізу інформації у державних базах даних.

Мета і завдання дослідження. Метою роботи є підвищення ефективності, швидкості та надійності процесів обробки неструктурованих даних у державних інформаційних системах шляхом розробки та програмної реалізації системи інтелектуального аналізу документів на базі згорткових нейронних мереж.

Для досягнення поставленої мети необхідно вирішити такі завдання:

– Провести системний аналіз існуючих методів оптичного розпізнавання символів (OCR) та виявити їх обмеження при роботі з документами.

– Дослідити сучасні архітектури нейронних мереж (CNN, RNN, Transformers) у контексті задач класифікації зображень та вилучення іменованих сутностей;

– Розробити архітектуру програмної системи, що забезпечує повний цикл обробки: від завантаження зображення до збереження верифікованих даних у реєстр;

– Розробити алгоритм попередньої обробки зображень для усунення дефектів сканування та нормалізації вхідних даних;

– Реалізувати механізм перехресної перевірки розпізнаних даних;

– Оцінити ефективність розробленої системи на основі метрик точності та швидкодії.

Об'єкт дослідження — процес автоматизованої обробки та семантичного аналізу графічних образів документів у інформаційно-аналітичних системах.

Предмет дослідження — моделі, методи та алгоритми комп'ютерного зору та машинного навчання для ідентифікації структури, класифікації типів та вилучення атрибутивної інформації з офіційних документів.

Методи дослідження. У роботі використано комплексний підхід з використанням методів цифрової обробки зображень — для препроцесингу, фільтрації шумів, бінаризації та корекції геометричних спотворень та теорії штучних нейронних мереж — для побудови моделей класифікації та розпізнавання тексту (архітектури CNN, LSTM);

Наукова новизна одержаних результатів. Удосконалено метод автоматичної класифікації різнотипних державних документів шляхом застосування гібридної моделі згорткової нейронної мережі, що дозволило підвищити точність

розпізнавання документів за умов часткового перекриття або низької якості зображення.

Апробація результатів. Основні положення роботи доповідалися на науково-практичних конференціях:

1. Волохін В. В., Приймак Є. О. МЕТОДИ ПОПЕРЕДНЬОЇ ОБРОБКИ ТА АДАПТИВНОЇ БІНАРИЗАЦІЇ ЗАШУМЛЕНИХ ЗОБРАЖЕНЬ В СИСТЕМАХ ЕЛЕКТРОННОГО УРЯДУВАННЯ «Проблеми комп'ютерної інженерії» : VI науково-практична конференція, м. Київ, 3 грудня 2025р. С. 1–3. URL: https://duikt.edu.ua/ua/news-1-823-14852-kafedra-kompyuternoi-inzhenerii-zaproshue-do-uchasti-u-vi-naukovo-praktichniy-konferencii-problemi-kompyuternoi-inzhenerii

2. Приймак Є. О., Зайцев Є. О., Лемешко А. В., Антоненко А. В. Стаття «ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ ОПТИМІЗАЦІЇ ПРОЦЕСУ ОБРОБКИ ДАНИХ В ДЕРЖАВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ ІЗ ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ». Журнал «ITSynergy» : номер 1, м. Київ, 2024 р. С. 6–15. URL: https://its.istu.edu.ua/index.php/ITS/article/view/52

РОЗДІЛ 1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ІНТЕЛЕКТУАЛЬНОЇ ОБРОБКИ ДОКУМЕНТІВ

1.1. Еволюція підходів до OCR та проблеми обробки неструктурованих даних

В епоху цифрової економіки дані трансформуються у новий стратегічний актив держави, що визначає ефективність управлінських рішень, прозорість процесів та якість надання адміністративних послуг. Однак, цінність інформаційних ресурсів прямо пропорційна рівню їх доступності, структурованості та придатності до автоматизованого машинного аналізу. Хоча державні реєстри та бази даних історично проектувалися як реляційні системи для зберігання строго типізованих записів, первинними джерелами наповнення цих реєстрів здебільшого виступають фізичні носії інформації. До цього масиву належать паперові заяви громадян, архівні довідки, паспортні документи різних років випуску, свідоцтва про акти цивільного стану, судові рішення та інше.

Процес цифровізації передбачає перетворення аналогового візуального сигналу, яким є зображення документа, у дискретний цифровий код. Ключовою технологією цього процесу виступає оптичне розпізнавання символів — OCR (Optical Character Recognition). З точки зору теорії розпізнавання образів та комп'ютерного зору, OCR визначається як процес трансформації растрового зображення рукописного, машинописного або друкованого тексту в послідовність символьних кодів, придатних для індексації, пошуку та семантичної обробки[1].

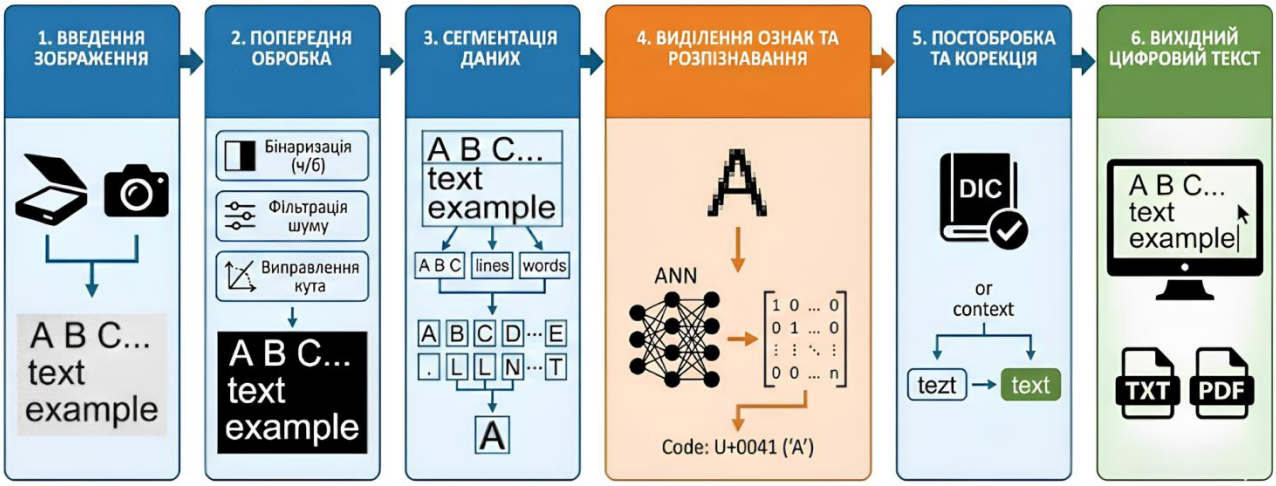


Рисунок 1.1 – Загальна схема роботи системи OCR

Історичний та технічний аналіз розвитку технологій розпізнавання дозволяє виділити кілька еволюційних етапів, кожен з яких характеризується зміною математичного апарату, підходів до вилучення ознак та рівня абстракції. Глибоке розуміння обмежень кожного покоління є критично важливим для обґрунтування необхідності переходу до методів глибокого навчання в рамках даного магістерського дослідження.

Перші спроби автоматизації читання тексту, що датуються 60-70-ми роками XX століття, базувалися на детермінованих алгоритмах матричного зіставлення[2]. Цей підхід, також відомий як "Pattern Matching", оперує зображенням як двовимірною матрицею інтенсивностей пікселів. В основі методу лежить гіпотеза, що зображення символу є статичним патерном, який можна ідентифікувати шляхом прямого накладання на відомий зразок.

Суть методу полягає у безпосередньому порівнянні растрового зображення вхідного символу з бібліотекою еталонних шаблонів, що зберігаються в пам'яті системи. Процес розпізнавання є суворо послідовним і починається з попередньої сегментації, де вхідне зображення всього документа розбивається на окремі області зв'язності шляхом аналізу гістограм проекцій. Це критичний етап, оскільки будь-яка помилка тут унеможливлює подальше розпізнавання.

Наступним кроком є нормалізація та масштабування, коли кожен виділений гліф приводиться до фіксованої піксельної сітки та бінаризується, що зводить зображення до монохромного вигляду. Завершується процес обчисленням кореляції, де алгоритм визначає метрику подібності між вхідним вектором зображення та кожним шаблоном у базі даних.

Математично міру розбіжності часто виражають через евклідову відстань між вектором вхідного зображення та вектором шаблону. Символ ідентифікується як такий, що відповідає шаблону з мінімальним значенням помилки. У більш просунутих реалізаціях того часу використовувався коефіцієнт нормалізованої взаємної кореляції, що дозволяло частково компенсувати зміни в освітленні.

Попри простоту реалізації та низькі вимоги до обчислювальних ресурсів, метод шаблонного зіставлення виявився тупиковою гілкою для задач масової обробки документів вільної форми. Головним обмеженням є жорстка залежність від гарнітури, адже система здатна розпізнати лише ті шрифти, точні піксельні еталони яких закладені в базу. Будь-яка зміна накреслення або використання нестандартного шрифту вимагає створення окремого набору шаблонів, що призводить до експоненційного зростання бази даних. Крім того, метод є вкрай нестійким до афінних перетворень: навіть незначний поворот зображення чи деформація паперу призводить до розбіжності піксельних сіток. Ще однією суттєвою проблемою є неможливість роботи зі злитими символами, оскільки алгоритм вимагає ідеальної ізоляції кожного знака, що практично неможливо забезпечити на реальних документах з печатками чи підписами.

Друге покоління систем OCR, що домінувало у 80-90-х роках, ознаменувало перехід від наївного аналізу пікселів до топологічного підходу[3]. Цей напрямок, часто званий "Omni-font OCR", базується на припущенні, що кожен символ алфавіту є не набором пікселів, а платонічною ідеєю, яку можна описати унікальним набором інваріантних ознак. Ці характеристики залишаються стабільними незалежно від гарнітури шрифту, його розміру чи товщини ліній.

У цьому підході процес розпізнавання трансформувався у задачу класифікації вектора ознак у багатовимірному просторі. До типових топологічних дескрипторів належать інваріанти, такі як кількість замкнених контурів, наявність кінцевих точок, вузлові перетини ліній та різноманітні геометричні примітиви. Наприклад, цифра "8" завжди матиме два замкнених контури, тоді як "1" — жодного, що дозволяє миттєво відсіяти значну частину хибних варіацій.

Алгоритмічна реалізація цього методу включає складний етап скелетизації, який ітеративно стоншує зображення символу до медіальної осі товщиною в один піксель. Отриманий "скелет" перетворюється на граф, який згодом порівнюється з графовими моделями літер за допомогою методів машинного навчання, таких як k-найближчих сусідів або дерева рішень.

Хоча структурний аналіз вирішив проблему різноманіття шрифтів, він виявився критично вразливим до шумів, що змінюють топологію об'єкта. В контексті українських державних документів це проявляється особливо гостро через деградацію носія. На старих документах, надрукованих на матричних принтерах, часто зустрічаються мікроскопічні розриви у штрихах, які змінюють топологічну структуру літери — наприклад, розірвана "О" класифікується як "С". Значною проблемою також є наявність гільйошних сіток та захисних елементів на бланках звітності. Алгоритми виділення країв часто не здатні відокремити корисний текст від високочастотного шуму захисної сітки, що створює хибні точки перетину та артефакти. Окрім того, структурні методи вимагають чіткої бінаризації, що ускладнює роботу з кольоровими печатками, які при переведенні в монохромний режим можуть зливатися з текстом.

Проміжним, але еволюційно необхідним етапом перед ерою нейромереж стало запозичення методів із суміжної галузі розпізнавання мовлення, що призвело до широкого застосування Прихованих Марковських Моделей (HMM)[4]. Цей підхід змінив метод розпізнавання: замість спроб ідеально сегментувати слово на літери, HMM дозволили аналізувати слово цілком як часовий ряд.

У моделі HMM сканує вікно рухається вздовж слова, витягуючи на кожному кроці вектор ознак, де внутрішні "приховані" стани моделі відповідають частинам літер. Головною перевагою цього підходу стала його стохастична природа: система оперує ймовірностями переходу між станами, а не жорсткими бінарними рішеннями. Це дозволило досягти значного прогресу в задачах розпізнавання рукописного тексту, де межі між літерами розмиті, а символи мають варіативну ширину. Проте суттєвим обмеженням залишається "марковське припущення", згідно з яким ймовірність наступного стану залежить лише від поточного. Це унеможливлює ефективне моделювання довгострокових залежностей, через що системі важко використовувати контекст цілого слова для уточнення окремих символів.

Революція у сфері комп'ютерного зору, пов'язана з розвитком глибоких нейронних мереж, докорінно змінила підходи до інтелектуальної обробки документів. Відбулася фундаментальна зміна парадигми від "інженерії ознак", коли правила опису літер програмувалися вручну, до "навчання представлень", коли нейронна мережа самостійно формує ієрархію абстрактних ознак на основі великих масивів даних. Чим глибше шар мережі, тим більш складні концепти він здатен розпізнавати.

Для задач аналізу державних документів найбільш ефективними зарекомендували себе гібридні архітектури. Згорткові нейронні мережі (CNN) виконують роль "очей" системи, забезпечуючи трансляційну інваріантність та ефективну фільтрацію структурного шуму, такого як текстура паперу чи гільйошні сітки. У свою чергу, рекурентні нейронні мережі (RNN), зокрема архітектури LSTM та GRU, виступають у ролі "читача", обробляючи послідовності ознак. Механізм довготривалої пам'яті дозволяє вирішувати проблему контексту, відновлюючи пошкоджені символи на основі вивченої статистичної моделі мови. Доповнення цих архітектур механізмами уваги дозволяє моделі фокусуватися на релевантних частинах зображення, реалізуючи

принцип End-to-End розпізнавання без необхідності попередньої посимвольної сегментації.

Окрім алгоритмічних викликів, необхідно враховувати інформаційну природу даних, якими оперують державні установи. Інформаційний простір держави є неоднорідним. Лише незначна частина даних є структурованою, тобто має чітко визначену схему та зберігається в реляційних базах.

Цей масив отримав назву "Dark Data" або "темні дані". Це інформаційні активи, які організації збирають та зберігають, але не можуть ефективно використовувати для аналітики, оскільки вони "замкнені" у нечитабельних для машини форматах, таких як скан-копії чи PDF-файли[5]. В українських реаліях проблема ускладнюється мультимодальністю архівів, де в одній справі можуть співіснувати документи різних історичних епох, від рукописних метрик до сучасних ID-карток. Ситуацію погіршує відсутність семантичної індексації, коли файли зберігаються під технічними іменами, перетворюючи архів на неструктуроване сховище. Крім того, перехід до мобільних сервісів призвів до появи потоку зображень "in the wild" — фотографій низької якості, зроблених у побутових умовах, що є критичним викликом для класичних систем. Таким чином, наукова проблема полягає у створенні адаптивної системи, здатної подолати цей семантичний розрив і перетворити "темні дані" на структурований державний актив.

1.2. Огляд сучасних архітектур нейромереж для задач комп'ютерного зору

Проектування ефективної системи інтелектуального аналізу документів вимагає глибокого розуміння математичного апарату, що лежить в основі сучасних методів штучного інтелекту. На відміну від класичних алгоритмів

комп'ютерного зору, які покладалися на ручне конструювання ознак, сучасні підходи базуються на автоматичному виявленні ієрархії репрезентацій даних. Вибір конкретної архітектури нейронної мережі є критичним етапом, оскільки різні типи документів та задач вимагають спеціалізованих підходів до обробки візуальної та текстової інформації. На сьогоднішній день у цій предметній області домінують три основні парадигми: згорткові мережі для вилучення візуальних ознак, рекурентні мережі для моделювання послідовностей та трансформерні моделі для глобального семантичного аналізу.

Фундаментальним будівельним блоком для будь-якої системи аналізу зображень залишаються згорткові нейронні мережі. Їхня архітектура інспірована будовою рецептивних полів зорової кори головного мозку, що дозволяє ефективно вирішувати проблему просторової інваріантності. Це означає здатність мережі розпізнавати об'єкт, наприклад гербову печатку чи фотографію власника паспорта, незалежно від їхнього розташування на скані, масштабу чи кута повороту.

Ключовим елементом таких мереж є операція згортки, під час якої навчені фільтри сканують вхідне зображення, формуючи карти ознак. На початкових шарах мережа навчається реагувати на примітивні елементи, такі як вертикальні та горизонтальні лінії, кути або градієнти кольору. Проходячи через глибші шари, ці примітиви комбінуються у складніші геометричні фігури, частини символів і, зрештою, у високорівневі семантичні об'єкти. Для задач державного сектору, де документи часто мають складний фон у вигляді гільйошних сіток, саме здатність CNN ігнорувати високочастотний шум і фокусуватися на структурних елементах є визначальною.

Особливу увагу в контексті даного дослідження варто приділити архітектурам із залишковими зв'язками, зокрема ResNet (Residual Networks). Впровадження механізму пропускання сигналу дозволило тренувати надзвичайно глибокі мережі без проблеми затухання градієнта. У розроблюваній системі такі мережі доцільно використовувати як «backbone» — основу для вилучення вектора

ознак з зображення перед його передачею на етап розпізнавання тексту або класифікації типу документа.

Якщо згорткові мережі відповідають за візуальне сприйняття структури документа, то задачу безпосереднього "читання" та формування текстового рядка вирішують рекурентні нейронні мережі. Специфіка текстових даних полягає в їхній секвенційній природі: значення кожного символу нерозривно пов'язане з попередніми та наступними елементами. Традиційні мережі прямого поширення не здатні ефективно обробляти такі залежності, оскільки вони розглядають кожен вхідний вектор як незалежну подію.

Класичні рекурентні мережі теоретично здатні утримувати інформацію про попередні стани, проте на практиці вони стикаються з проблемою короткочасної пам'яті. При обробці довгих послідовностей, таких як прізвище особи або номер органу видачі, градієнт помилки швидко затухає при зворотному поширенні, що призводить до втрати контексту початку слова. Вирішенням цієї проблеми стала розробка архітектури LSTM (Long Short-Term Memory). Впровадження спеціальних шлюзів — вхідного, вихідного та шлюзу забування — дозволило мережі керувати потоком інформації, вибірково зберігаючи або видаляючи дані з внутрішнього стану пам'яті[6].

У контексті розпізнавання українських документів, де зустрічаються довгі складені прізвища або складні назви установ, використання двонаправлених LSTM є стандартом. Така архітектура аналізує послідовність одночасно у прямому та зворотному напрямках, що дозволяє використовувати як попередній, так і наступний контекст для уточнення розпізнавання конкретного символу, значно підвищуючи точність роботи з пошкодженими або нечіткими фрагментами тексту.

Еволюція методів розпізнавання призвела до розуміння того, що ізольоване використання CNN або RNN не дозволяє досягти максимальної ефективності. Це зумовило появу гібридних архітектур типу CRNN (Convolutional Recurrent Neural Network), які об'єднують переваги обох підходів у єдиний диференційовний

конвеєр[7]. У такій системі згорткові шари трансформують вхідне зображення у послідовність векторів ознак, які зберігають просторову інформацію по горизонтальній осі, після чого ці вектори подаються на вхід рекурентним шарам для декодування контексту.

Критично важливим компонентом таких систем є функція втрат CTC (Connectionist Temporal Classification). Традиційні методи навчання вимагають наявності розмічених даних, де кожному символу тексту відповідає чітко визначена позиція на зображенні. Створення таких наборів даних є надзвичайно трудомістким процесом. Натомість CTC дозволяє навчати мережу на парах "зображення — текст" без явної розмітки координат кожного символу. Алгоритм самостійно знаходить найбільш ймовірне вирівнювання між послідовністю ознак та цільовим текстом, враховуючи можливі пропуски та дублювання символів. Це робить CRNN ідеальним вибором для розпізнавання рядків змінної довжини, таких як поля адреси або назви документів.

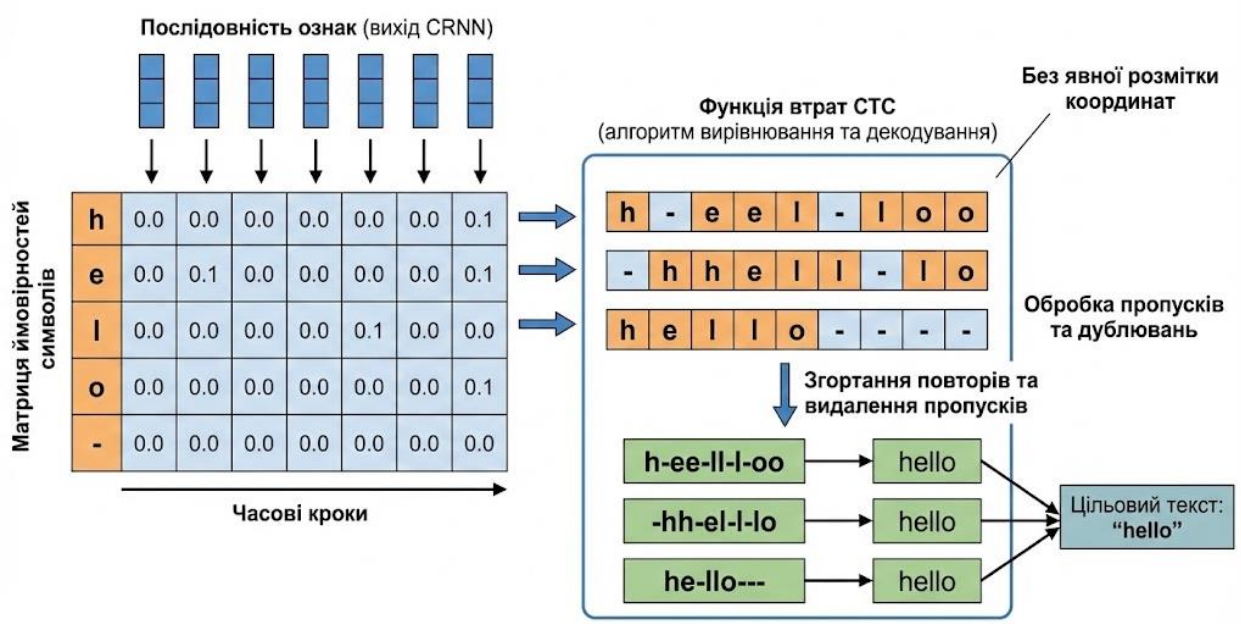


Рисунок 1.2 – Принцип роботи функції втрат CTC

Найновішим етапом розвитку технологій комп'ютерного зору стала подальша адаптація архітектури, яка спочатку була розроблена для задач

машинного перекладу. Ключовою інновацією тут виступає механізм уваги, який дозволяє моделі динамічно оцінювати важливість різних фрагментів вхідного зображення при формуванні вихідного результату. На відміну від рекурентних мереж, які обробляють інформацію послідовно, ці здатні аналізувати глобальні залежності між усіма частинами зображення одночасно.

У сфері аналізу документів проривним стало застосування мультимодальних трансформерів, таких як LayoutLM. Ці моделі навчаються не лише на візуальних образах символів, але й враховують їхнє просторове розташування на сторінці та семантичні зв'язки. Це дозволяє системі "розуміти" структуру документа на глибшому рівні. Наприклад, модель може відрізнити серію паспорта від дати народження не лише за форматом даних, але й за тим, що серія зазвичай розташована у верхній частині бланка поруч зі словом "Паспорт", а дата — у відповідній графі нижче.

Використання трансформерних моделей дозволяє досягти безпрецедентної точності в задачах вилучення інформації з неструктурованих та напівструктурованих документів. Однак суттєвим недоліком є висока обчислювальна складність таких алгоритмів, що вимагає значних апаратних ресурсів для їх розгортання. Тому при проектуванні архітектури державної інформаційної системи необхідно шукати баланс між потужністю трансформерних моделей та вимогами до швидкодії та ресурсоємності, можливо, застосовуючи методи дистиляції знань для створення полегшених версій моделей.

1.3. Проблематика класифікації та сегментації різнотипних документів у державних реєстрах

Специфіка національного документообігу та історичні особливості формування державних інформаційних ресурсів в Україні створюють унікальний ландшафт технічних викликів для систем штучного інтелекту, які часто

ігноруються або недостатньо глибоко опрацьовуються у стандартних комерційних рішеннях західного зразка. Головна складність полягає не стільки в самому процесі оптичного розпізнавання символів, скільки в коректній попередній ідентифікації типу документа, його семантичній сегментації та роботі зі специфічними поліграфічними елементами захисту, які для алгоритмів комп'ютерного зору виступають джерелом інтенсивного шуму. Фундаментальною проблемою є висока варіативність форматів, зумовлена тривалим перехідним періодом цифровізації та паралельним юридичним обігом документів різних історичних поколінь. В актуальному правовому полі України одночасно чинними залишаються паспорти громадянина зразка 1994 року у формі книжечки, які можуть мати як рукописне, так і машинописне заповнення кирилицею, та сучасні біометричні ID-картки зразка 2016 року, виконані згідно з міжнародними стандартами ICAO 9303. Для інтелектуальної системи це означає неможливість застосування єдиного уніфікованого шаблону зчитування та необхідність реалізації багаторівневого класифікатора, здатного ще на етапі попередньої обробки безпомилково визначити клас об'єкта для вибору релевантного сценарію екстракції даних, оскільки спроба застосувати алгоритм пошуку машинозчитуваної зони до старого паспорта-книжечки неминуче призведе до помилки сегментації та зупинки всього конвеєра обробки.

Суттєвим бар'єром для ефективної роботи згорткових нейронних мереж є наявність розвиненої системи фізичного захисту на бланках звітності. Голограми, кінеграми, рельєфне тиснення, водяні знаки та мікротекст, які для людини слугують підтвердженням автентичності документа, для алгоритмів комп'ютерного зору перетворюються на складний структурний шум, що перешкоджає бінаризації. Зокрема, голографічні плівки при скануванні або фотографуванні під певним кутом створюють яскраві світлові бліки та дзеркальні відображення, які можуть частково або повністю перекривати текстові поля, руйнуючи контури символів. Ще більшу алгоритмічну проблему створюють гільйошні сітки — складні аперіодичні візерунки з тонких ліній, що покривають всю поверхню бланка. Стандартні методи виділення країв, такі як оператор Канні

або Собеля, часто помилково інтерпретують лінії захисної сітки як частини літерних гліфів, що призводить до ефекту «склеювання» символів із фоном або появи фантомних об'єктів, критично знижуючи точність розпізнавання, особливо для дрібних шрифтів. Ситуація ускладнюється наявністю "шумів реального світу": печаток, штампів про реєстрацію місця проживання, які часто ставляться хаотично, перекриваючи основний текст або фотографію, а також рукописних підписів посадових осіб, що заходять на зони машинописного тексту.

Окремий кластер проблем пов'язаний із геометричними та фотометричними спотвореннями, характерними для документів, отриманих не з професійних планшетних сканерів, а через мобільні застосунки або веб-камери в центрах надання адміністративних послуг. Такі зображення неминуче страждають від перспективних викривлень, коли документ сфотографовано під кутом, що призводить до афінних деформацій форми літер, роблячи їх невпізнаваними для класичних OCR-двигунів, навчених на фронтальних проекціях. Нерівномірне освітлення, наявність глибоких тіней від рук оператора або мобільного пристрою створюють градієнти яскравості, які унеможливлюють використання глобальних методів бінаризації, вимагаючи застосування ресурсоємних адаптивних алгоритмів локальної порогової обробки. Крім того, значна частина архівних документів у державних реєстрах перебуває у стані фізичної деградації: пожовклий папір, вицвілі чорнила матричних принтерів, потертості на місці згинів та плями вологи суттєво знижують контрастність корисного сигналу, вимагаючи застосування методів цифрової реставрації зображень та супер-роздільної здатності перед етапом розпізнавання.

Додатковим викликом є лінгвістична та семантична гетерогенність контенту. Більшість відкритих моделей оптимізовані переважно для англійської мови та латинської графіки, тоді як якісна робота з українською кирилицею, особливо з розпізнаванням специфічних літер «ї», «є», «ґ» та апострофа, вимагає спеціалізованого донавчання на репрезентативних національних дата сетах[8]. Складність зростає через мультимовність сучасних документів (наприклад,

закордонних паспортів або водійських посвідчень), де дані дублюються латинською транслітерацією, що вимагає від системи здатності працювати в змішаному мовному режимі без втрати контексту. Також критичною є проблема семантичної неоднозначності полів: на старих документах відсутня чітка розмітка полів, і цифрова послідовність «12.05.1990» без розуміння контексту може бути інтерпретована і як дата народження, і як дата видачі документа, і як дата закінчення терміну дії, що вимагає застосування не просто методів OCR, а повноцінних моделей розуміння документів, здатних аналізувати взаємне просторове розташування текстових блоків та їх логічні зв'язки.

1.4. Аналіз вимог до безпеки даних при використанні ШІ в держсекторі

Впровадження новітніх інтелектуальних систем автоматизованої обробки інформації в екосистему державних баз даних та реєстрів є складним процесом, який неможливо реалізувати без суворого дотримання комплексних вимог інформаційної безпеки. В Україні ця сфера жорстко регулюється низкою нормативно-правових актів, зокрема Законом України «Про захист інформації в інформаційно-комунікаційних системах», Законом України «Про захист персональних даних», а також відомчою нормативною базою Державної служби спеціального зв'язку та захисту інформації України, яка регламентує порядок побудови та експертизи Комплексних систем захисту інформації (КСЗІ). Специфіка використання технологій штучного інтелекту, зокрема глибоких згорткових нейронних мереж, привносить у ландшафт кібербезпеки принципово нові, нетипові вектори загроз, які виходять за рамки традиційних вразливостей програмного забезпечення і вимагають детального наукового аналізу та розробки відповідних контрзаходів ще на етапі архітектурного проектування системи. Якщо класичні моделі загроз фокусуються переважно на мережевих атаках, перехопленні трафіку чи SQL-ін'єкціях, то для систем на базі машинного навчання критичними стають атаки, спрямовані безпосередньо на математичну

логіку роботи моделі та навчальні дані.

Однією з найбільш небезпечних та специфічних загроз для систем комп'ютерного зору, що використовуються для ідентифікації документів, є атаки змагальних прикладів, відомі в науковій літературі як «атака супротиву». Суть цього явища полягає у вразливості нейронних мереж до незначних, часто непомітних для людського ока збурень вхідних даних, які, однак, кардинально змінюють результат класифікації. Зловмисник, маючи на меті обійти систему безпеки або сфальсифікувати дані, може нанести на зображення документа спеціально розрахований цифровий шум або використати фізичні артефакти, наприклад, наклейки з певним візерунком на пластикову ID-картку. Такі модифікації можуть призвести до того, що система з високою впевненістю ідентифікує підроблений документ як валідний або помилково розпізнає цифри в номері документа, змінюючи особистість власника. Захист від подібних атак вимагає впровадження багаторівневих механізмів попередньої фільтрації зображення, використання технік змагального навчання, коли модель під час тренування спеціально ознайомлюють із такими прикладами, а також застосування робастних архітектур, стійких до шумового забруднення даних.

Окремий пласт проблем безпеки пов'язаний із конфіденційністю навчальних даних та самої моделі. Існує реальна загроза інверсії моделі та атак на приналежність. Оскільки глибокі нейронні мережі мають здатність "запам'ятовувати" патерни навчальної вибірки, існує ризик, що зловмисник, маючи доступ до програмного інтерфейсу (API) системи, зможе шляхом аналізу серії відповідей нейромережі на спеціально підібрані запити реконструювати фрагменти вихідних зображень. У контексті державних реєстрів, де моделі можуть навчатися на реальних скан-копіях паспортів громадян, це створює критичний ризик витоку персональних даних. Для нівелювання цієї загрози необхідно застосовувати методи диференційної приватності під час тренування алгоритмів, що дозволяє додавати математичний шум до градієнтів навчання, унеможливлюючи відновлення конкретного запису, а також суворо обмежувати

доступ до внутрішніх параметрів моделі.

Фундаментальним архітектурним обмеженням, продиктованим вимогами національної безпеки, є заборона на транскордонну передачу інформації з обмеженим доступом, до якої належить вміст державних реєстрів. Це робить категорично неприпустимим використання у промисловій експлуатації популярних публічних хмарних сервісів OCR, оскільки це передбачає фізичну передачу зображень документів на сервери, що знаходяться поза межами правового поля України. Система повинна безумовно базуватися на принципі локальної обробки даних або використовувати захищену приватну хмару, розгорнуту в атестованих дата-центрах на території України. При цьому всі канали передачі даних між автоматизованим робочим місцем оператора та сервером обробки мають бути захищені із використанням засобів криптографічного захисту інформації, які мають експертний висновок Держспецзв'язку та реалізують національні стандарти шифрування, такі як ДСТУ 7624:2014 «Калина» або ДСТУ 4145-2002.

Забезпечення цілісності та достовірності даних у системах на базі штучного інтелекту ускладнюється феноменом, який отримав назву «галюцинації» нейромереж. На відміну від класичних алгоритмів, які у випадку помилки зазвичай видають повідомлення про збій, нейронна мережа може згенерувати правдоподібний, але фактично невірний результат, наприклад, замінивши одну цифру в даті народження або літеру в прізвищі, при цьому супроводжуючи це високим показником впевненості. Внесення таких спотворених даних до державного реєстру може мати серйозні юридичні наслідки для громадян. Для мітигації цього ризику система повинна містити жорсткі детерміновані алгоритмічні перевірки, які діють як запобіжник після етапу нейромережевого розпізнавання. До таких перевірок належать валідація контрольних сум машинозчитуваної зони згідно зі стандартами ІСАО, перехресна звірка дат та логічний контроль полів. Крім того, всі дії системи, включаючи вхідні зображення, проміжні результати розпізнавання та остаточні дані, мають

фіксуватися у захищених від модифікації журналах аудиту, що дозволить у випадку інциденту провести розслідування та чітко розмежувати технічний збій алгоритму, помилку оператора чи зловмисне втручання.

Важливим аспектом, що вимагає врахування при проектуванні КСЗІ для інтелектуальних систем, є забезпечення доступності в умовах можливих атак на виснаження ресурсів. Алгоритми глибокого навчання є надзвичайно ресурсоємними, і обробка одного зображення потребує значно більше обчислювальних потужностей процесора або графічного прискорювача, ніж обробка звичайного веб-запиту. Це робить систему вразливою до специфічних DoS-атак, коли зловмисник може паралізувати роботу всього реєстру, надсилаючи відносно невелику кількість складних для обробки зображень. Захист від цього вектора атак вимагає впровадження інтелектуальних механізмів черг, квотування запитів та попередньої легковагової валідації файлів до моменту їх передачі на вхід важких нейронних мереж. Також необхідно враховувати проблему інтерпретованості рішень штучного інтелекту, відому як проблема «чорної скриньки», яка вступає у протиріччя з вимогами щодо прозорості та обґрунтованості адміністративних рішень у державному секторі. Відповідно, необхідною умовою впровадження є імплементація модулів пояснюваного штучного інтелекту, які здатні візуалізувати зони уваги мережі та надавати оператору зрозумілу технічну аргументацію результатів класифікації або причин відмови у верифікації документа, забезпечуючи таким чином можливість контролю з боку людини.

1.5. Нормативно-правові засади впровадження систем штучного інтелекту в державному секторі України

Проектування та імплементація інформаційних систем, що базуються на

технологіях штучного інтелекту, для потреб державного управління не може розглядатися виключно як інженерна задача. Ключовим фактором, що визначає архітектуру, функціональність та саму можливість експлуатації такої системи, є відповідність чинному законодавчому полю України. Враховуючи, що об’єктом обробки виступають персональні дані громадян, часто чутливого характеру, правовий аспект набуває критичного значення та потребує детального аналізу на етапі передпроектного обстеження.

Фундаментальним документом, що регулює цю сферу, є Закон України «Про захист персональних даних». Згідно з його положеннями, автоматизована обробка даних допустима лише за умови забезпечення належного рівня захисту від випадкової втрати або знищення, а також від незаконної обробки[9]. У контексті розроблюваної системи це накладає імперативну вимогу щодо реалізації механізмів логування всіх дій з даними (хто, коли і до якого документа отримав доступ), а також шифрування каналів передачі інформації. Особливу увагу слід приділити статті, яка регулює права суб'єкта даних, зокрема право на захист від автоматизованого рішення, яке має для нього правові наслідки. Це означає, що система інтелектуального аналізу не може бути "останньою інстанцією" у прийнятті рішення, наприклад, про відмову у видачі довідки. Архітектура повинна передбачати обов'язковий етап валідації результату відповідальною посадовою особою, що має бути технічно реалізовано через відповідні інтерфейси верифікації.

Важливим кроком у формуванні державної політики стала схвалена Кабінетом Міністрів України «Концепція розвитку штучного інтелекту в Україні»[10]. Цей документ визначає пріоритетним напрямком використання ШІ саме у сфері публічного адміністрування для автоматизації рутинних процесів. Концепція наголошує на необхідності дотримання принципів прозорості та пояснюваності алгоритмів. Технічно це означає, що використання нейронних мереж є ризикованим без додаткових механізмів інтерпретації. Тому при проектуванні системи необхідно надавати перевагу архітектурам, які дозволяють

візуалізувати ознаки, на основі яких було прийнято рішення про класифікацію документа, наприклад, шляхом підсвічування відповідних зон на зображенні.

Окремим пластом регулювання є Закон України «Про хмарні послуги», який набув чинності у 2022 році. Він впроваджує принцип «Cloud First» для державних органів, заохочуючи використання хмарних технологій. Однак для систем, що обробляють інформацію з обмеженим доступом або персональні дані, встановлюються жорсткі обмеження щодо локалізації центрів обробки даних. Використання публічних хмарних сервісів (наприклад, AWS, Google Cloud) можливе лише за умови наявності у провайдера відповідних сертифікатів відповідності КСЗІ (Комплексна система захисту інформації). Враховуючи складність отримання таких сертифікатів для закордонних платформ у контексті українського законодавства, найбільш доцільним архітектурним рішенням залишається розгортання системи у приватній хмарі або на власних серверах державної установи, що було враховано у розділі проектування архітектури.

Також слід враховувати вимоги щодо технічного захисту інформації, які регламентуються нормативними документами Державної служби спеціального зв'язку та захисту інформації України. Будь-яка система, що впроваджується в органах державної влади, підлягає державній експертизі КСЗІ. Це вимагає створення детальної технічної документації, моделі загроз та впровадження відповідних засобів захисту. На програмному рівні це диктує необхідність використання сертифікованих в Україні бібліотек криптографічного захисту (наприклад, бібліотек, що реалізують національні стандарти ДСТУ 4145-2002 для електронного цифрового підпису), а не лише загальноприйнятих світових стандартів.

Таким чином, нормативно-правове поле України створює чіткі рамки для розробки систем інтелектуального аналізу: обов'язковість локальної обробки або використання сертифікованих хмар, необхідність інтерпретованості рішень ШІ, суворе логування доступу та пріоритет національних криптографічних стандартів. Ігнорування цих вимог на етапі проектування призведе до неможливості введення

системи в промислову експлуатацію, незалежно від її технічної досконалості.

1.6. Аналіз існуючих платформ інтелектуальної обробки документів

Для обґрунтування доцільності розробки власної системи інтелектуального аналізу необхідно провести детальний аналіз ринку існуючих рішень класу IDP (Intelligent Document Processing). Сучасний ринок пропонує широкий спектр продуктів, від універсальних хмарних API до спеціалізованих корпоративних платформ. Аналіз їхніх можливостей та обмежень дозволить виявити нішу для власної розробки та уникнути "винахідництва велосипеда" там, де існують ефективні готові інструменти.

Одним із лідерів ринку є рішення компанії ABBYY, зокрема платформа ABBYY FlexiCapture. Це потужне корпоративне рішення, яке використовує комбінацію традиційних структурних методів та машинного навчання. Сильною стороною продуктів ABBYY є виняткова точність розпізнавання кириличних текстів та наявність готових шаблонів для багатьох типів документів. Однак впровадження FlexiCapture у державному секторі наштовхується на ряд суттєвих перешкод. По-перше, це висока вартість ліцензування, яка розраховується за кількістю оброблених сторінок, що створює значне навантаження на бюджет при великих обсягах архівних даних. По-друге, це закритість архітектури, що ускладнює аудит безпеки коду та інтеграцію специфічних національних криптографічних алгоритмів. Крім того, налаштування системи під нові, нестандартні типи архівних довідок часто вимагає залучення сертифікованих спеціалістів вендора, що знижує гнучкість системи.

Іншим популярним вектором є використання хмарних сервісів від технологічних гігантів: Google Cloud Vision API, Amazon Textract та Microsoft Azure Form Recognizer. Ці сервіси демонструють вражаючі результати завдяки

використанню величезних обчислювальних потужностей та передових трансформерних моделей, навчених на мільярдах зображень. Вони здатні ефективно розпізнавати рукописний текст та складні таблиці без попереднього налаштування шаблонів. Проте, як було зазначено у попередніх розділах, їх використання є неможливим для обробки персональних даних громадян України через вимоги законодавства щодо локалізації даних та захисту інформації. Передача сканів паспортів на сервери, розташовані у Франкфурті чи Вірджинії, створює неприпустимі ризики витоку інформації та залежності критичної інфраструктури держави від зовнішніх провайдерів.

У сегменті відкритого програмного забезпечення стандартом де-факто тривалий час залишався двигун Tesseract OCR, що підтримується компанією Google. Його головними перевагами є безкоштовність, можливість локального розгортання та підтримка української мови. Однак архітектура Tesseract, навіть з впровадженням LSTM у 4-й версії, залишається чутливою до якості зображень. Він погано справляється зі складним фоном, шумами та викривленнями перспективи, що вимагає розробки складного конвеєра попередньої обробки зображень. Крім того, Tesseract повертає лише "сирий" текст без розуміння його семантики, що перекладає задачу структурування даних на розробника кінцевої системи.

Останнім часом набувають популярності бібліотеки на базі глибокого навчання, такі як PaddleOCR та EasyOCR. Вони демонструють значно вищу стійкість до шумів та складних шрифтів порівняно з Tesseract. PaddleOCR, розроблений компанією Baidu, пропонує ультра-легкі моделі, здатні працювати навіть на мобільних пристроях, і показує відмінні результати на багатомовних текстах. Проте більшість цих рішень орієнтовані на універсальні задачі і не мають специфічних налаштувань для українських документів (наприклад, знання формату серії паспорта чи структури ІПН), що вимагає їх донавчання на власних наборах даних.

Узагальнюючи проведений аналіз, можна зробити висновок, що на ринку

відсутнє готове рішення, яке б одночасно задовольняло вимоги щодо безпеки, вартості, точності роботи з українськими "брудними" даними та глибокого розуміння контексту державних документів. Комерційні платформи є занадто дорогими та складними в інтеграції, хмарні API не відповідають вимогам КСЗІ, а чисті Open Source бібліотеки потребують суттєвої доробки та створення надбудови для постобробки даних. Це підтверджує актуальність теми магістерської роботи та обґрунтовує необхідність розробки гібридної системи, яка б поєднувала потужність сучасних Open Source нейромереж з власною логікою препроцесингу та семантичного аналізу, адаптованою до реалій українського документообігу.

РОЗДІЛ 2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ПОТЕНЦІЙНОЇ СИСТЕМИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ

2.1. Проектування загальної архітектури системи та обґрунтування модульної структури

Проектування програмного забезпечення для державного сектору вимагає дотримання складного балансу між гнучкістю впровадження нових функцій та суворими, часто консервативними вимогами до інформаційної безпеки та стабільності роботи. При розробці архітектури системи інтелектуального аналізу документів визначальним фактором стала імперативна необхідність обробки конфіденційних персональних даних громадян виключно у закритому контурі. Архітектурний патерн повинен не лише забезпечувати фізичну ізоляцію модулів обробки даних від публічних мереж, але й гарантувати можливість локального горизонтального масштабування при пікових навантаженнях, які є характерними для державних сервісів у періоди подання звітності або масової верифікації громадян.

Традиційні монолітні архітектури, де інтерфейс користувача, бізнес-логіка, модулі авторизації та шар доступу до даних тісно інтегровані в єдиний виконуваний модуль, історично вважалися стандартом для корпоративних систем завдяки простоті розгортання на початкових етапах. Однак для специфічних задач інтелектуального аналізу, що включають ресурсоємні операції комп'ютерного зору та використання глибоких нейронних мереж, монолітний підхід створює критичні перешкоди для розвитку системи. Проблема полягає у жорсткій зв'язності компонентів: будь-яка локальна зміна в модулі розпізнавання, наприклад, оновлення вагових коефіцієнтів нейромережі для підтримки нового зразка посвідчення водія або виправлення помилки в алгоритмі препроцесингу,

неминуче вимагала б повної перекомпіляції, тестування та перезапуску всього комплексу. Це створює неприпустимі ризики для доступності сервісу, оскільки оновлення допоміжного модуля аналітики може призвести до зупинки критично важливих функцій, таких як автентифікація користувачів або прийом заявок.

З огляду на виявлені ризики, для проектованої системи було обрано сервіс-орієнтовану архітектуру з елементами мікросервісного підходу, що дозволяє досягти необхідного рівня декомпозиції. Система розділяється на набір автономних, слабозв'язаних модулів, кожен з яких інкапсулює логіку конкретного етапу конвеєра обробки даних та взаємодіє з іншими через чітко визначені інтерфейси API. Центральним елементом цієї конструкції виступає модуль оркестрації, який керує послідовністю виконання завдань та потоками даних, не втручаючись у внутрішню реалізацію окремих сервісів. Такий підхід дозволяє розробляти, тестувати та, що найважливіше, масштабувати кожен компонент незалежно від решти системи. Наприклад, модуль попередньої обробки зображень, який виконує математично складні геометричні трансформації та фільтрацію шумів, може бути розгорнутий на спеціалізованих серверах з підтримкою GPU для апаратного прискорення, тоді як менш вимогливий модуль веб-інтерфейсу або модуль логування може ефективно функціонувати на стандартному серверному обладнанні, що забезпечує раціональне використання бюджетних коштів на інфраструктуру.

Важливим архітектурним рішенням, спрямованим на підвищення стійкості системи, є використання асинхронної, подійно-орієнтованої моделі взаємодії між компонентами. Обробка зображень високої роздільної здатності та їх прогонка через каскад глибоких нейронних мереж є операціями з високою латентністю, тривалість яких є непередбачуваною і може варіюватися від кількох секунд до хвилини залежно від якості вхідного файлу та поточної черги завдань. Застосування класичної синхронної моделі, за якої клієнтський додаток змушений чекати на відповідь сервера у реальному часі, призвела б до блокування інтерфейсу, тайм-аутів з'єднання та вкрай негативного досвіду користувача.

Натомість архітектура передбачає впровадження брокера повідомлень для буферизації вхідних запитів. Коли користувач завантажує документ, система миттєво повертає ідентифікатор транзакції, звільняючи клієнтське з'єднання, тоді як саме завдання потрапляє у чергу і обробляється у фоновому режимі пулом воркерів. Кількість активних воркерів може динамічно змінюватися залежно від довжини черги, що дозволяє системі згладжувати пікові навантаження без деградації продуктивності.

Модульна структура також ефективно вирішує складну проблему інтеграції з успадкованими державними реєстрами, які часто використовують застарілі протоколи обміну даними. Замість надання внутрішнім сервісам прямого доступу до зовнішніх баз даних, що створює суттєві ризики інформаційної безпеки та порушує принцип інкапсуляції, система використовує виділений шар абстракції — адаптер інтеграції. Цей модуль виступає єдиною точкою входу для взаємодії із зовнішнім світом, реалізуючи специфічні протоколи (наприклад, SOAP-запити через систему «Трембіта» або пропрієтарні API відомчих систем). Він транслює внутрішні уніфіковані структури даних системи у формати, зрозумілі зовнішнім реєстрам, і навпаки. Це дозволяє ізолювати ядро інтелектуального аналізу від нестабільності зовнішніх інтерфейсів, забезпечуючи легку адаптацію системи при зміні регламентів взаємодії без необхідності переписувати логіку нейромережевої обробки.сучасні системи відеоспостереження повинні еволюціонувати. Хоча централізація має свої переваги, коли всі дані зберігаються та обробляються в одному місці, така концепція є застарілою і потребує змін для ефективнішого управління сучасними системами безпеки.

2.2. Моделювання бізнес-процесів та інформаційних потоків

Ефективність проектування та подальшої експлуатації інформаційної системи значною мірою залежить від детального моделювання бізнес-процесів. Цей етап дозволяє не лише формалізувати логіку обробки даних, але й виявити

потенційні архітектурні колізії та вузькі місця ще до початку написання програмного коду. У контексті системи інтелектуального аналізу документів основним, наскрізним бізнес-процесом є трансформація неструктурованого вхідного зображення у структурований, юридично значущий та валідований запис бази даних. Для опису та стандартизації цього процесу доцільно використати нотацію BPMN 2.0 (Business Process Model and Notation), яка є міжнародним стандартом ISO/IEC 19510. Використання BPMN дозволяє створити візуальну модель, зрозумілу як технічним спеціалістам, так і замовникам з боку державних установ, чітко розмежувавши зони відповідальності між учасниками процесу: оператором, модулями системи розпізнавання та зовнішніми реєстрами.

Життєвий цикл обробки документа починається з етапу ініціалізації запиту, коли оператор через веб-інтерфейс або зовнішня інформаційна система через API завантажує цифровий образ документа. Запит проходить через захищений шлюз, де відбувається первинна технічна валідація файлу. На цьому рівні система виконує перевірку MIME-типів (відсіювання виконуваних файлів під виглядом зображень), аналіз заголовків файлу на пошкодження та валідацію розміру для запобігання DoS-атакам. Додатково має розраховуватись контрольна сума файлу для перевірки на дублікати: якщо аналогічний файл вже було оброблено системою раніше, процес може бути оптимізовано шляхом видачі збереженого результату з кешу. Успішно валідований файл отримує унікальний ідентифікатор транзакції і потрапляє у чергу повідомлень для асинхронної обробки.

Наступним критичним кроком є етап попередньої обробки та інтелектуальної класифікації. Спеціалізований модуль комп'ютерного зору аналізує глобальні візуальні характеристики зображення для визначення типу документа. Цей процес реалізує патерн "Маршрутизатор": залежно від результату класифікації потік управління розгалужується.

– Якщо система з високою ймовірністю класифікує зображення як паспорт громадянина України у формі ID-картки, активується сценарій пошуку машиночитабельної зони та біометричної фотографії власника.

– Якщо ідентифіковано паспорт-книжечку зразка 1994 року, запускається альтернативний алгоритм, орієнтований на сегментацію сторінки та пошук рукописного або машинописного тексту у фіксованих координатах бланка.

– У випадку, коли система не може однозначно визначити тип документа або виявляє ознаки суттєвого пошкодження носія, автоматичний процес переривається, і завдання отримує статус "Потребує уточнення". Такий запит перенаправляється у чергу ручної верифікації, де оператор приймає рішення про тип документа або відхиляє його як непридатний, що дозволяє уникнути накопичення помилок на наступних етапах екстракції.

Етап екстракції та нормалізації даних є технологічним ядром бізнес-процесу. Залежно від обраного сценарію, система виконує сегментацію зображення на семантичні блоки. Для кожного блоку застосовуються спеціалізовані моделі розпізнавання: рекурентні нейронні мережі для текстових полів, детектори облич для вилучення фотографій та алгоритми аналізу графічних примітивів для підписів і печаток. Отримані "сирі" дані проходять через модуль лінгвістичної нормалізації та пост-обробки. Цей модуль використовує словники імен, назв населених пунктів та адміністративних органів для виправлення типових помилок оптичного розпізнавання (наприклад, заміна цифри "0" на літеру "О" у серії паспорта, відновлення пошкоджених закінчень у прізвищах). Також відбувається приведення дат до єдиного формату та транслітерація полів за потреби.

Фінальним етапом є семантична валідація, перехресна перевірка та збереження. Система виконує логічний контроль даних: наприклад, дата видачі паспорта не може бути меншою за дату народження плюс 14 або 16 років (залежно від типу документа), а термін дії не може бути минулим. Крім того, здійснюється перехресна перевірка даних з різних зон документа: інформація, зчитана з MRZ-зони, порівнюється з візуальною зоною. Якщо дані співпадають, система формує пошуковий запит до локальної бази даних або, за наявності відповідних повноважень та сертифікатів, ініціює запит до зовнішніх державних

реєстрів через систему інтероперабельності «Трембіта». Цей крок дозволяє підтвердити юридичну чинність документа та перевірити його статус (наприклад, чи не значиться бланк як викрадений або втрачений). Результатом процесу є формування структурованого JSON-об'єкта, який містить розпізнані атрибути, метадані процесу обробки та рівень довіри до кожного поля. Цей об'єкт разом із посиланнями на оброблені зображення зберігається в захищеному сховищі, завершуючи цикл обробки.

2.3. Обґрунтування вибору технологічного стеку та інструментальних засобів

Вибір технологічного стеку для реалізації системи інтелектуального аналізу документів базувався на комплексному аналізі критеріїв продуктивності, наявності розвинених бібліотек для машинного навчання, активності професійної спільноти та відповідності суворим вимогам інформаційної безпеки. Враховуючи специфіку задачі, яка поєднує класичну веб-розробку, системне програмування та складні математичні обчислення в області комп'ютерного зору, було прийнято стратегічне рішення про використання гетерогенного стеку технологій. Такий підхід дозволяє використовувати найкращі інструменти для кожної підзадачі, нівелюючи недоліки окремих мов програмування чи фреймворків шляхом їх ефективної комбінації.

У якості основної мови програмування для бекенду та реалізації бізнес-логіки машинного навчання обрано Python. Цей вибір є безальтернативним з огляду на домінування Python у сфері Data Science, що забезпечує доступ до найпотужніших екосистем для роботи з тензорними обчисленнями та нейронними мережами, таких як TensorFlow, PyTorch та Keras. Незважаючи на те, що Python є інтерпретованою мовою, його роль у системі зводиться до висококрівневого "клею", який керує викликами оптимізованих бібліотек, написаних на C/C++. Це дозволяє поєднати високу швидкість розробки та читабельність коду з продуктивністю на рівні компільованих мов. Для побудови API та оркестрації

мікросервісів використовується сучасний фреймворк FastAPI. Його ключовою перевагою є нативна підтримка стандарту ASGI та асинхронного програмування, що дозволяє обробляти тисячі конкурентних з'єднань без блокування основного потоку виконання. Завдяки використанню бібліотеки Pydantic для валідації типів даних, FastAPI забезпечує автоматичну генерацію інтерактивної документації та гарантує цілісність даних на вході в систему, демонструючи продуктивність, співмірну з серверами на Go або Node.js, що є критично важливим для високонавантажених державних реєстрів.

Для вирішення задач комп'ютерного зору, геометричних трансформацій та попередньої обробки зображень ключовим інструментом обрано бібліотеку OpenCV. Вона надає високооптимізовані реалізації фундаментальних алгоритмів фільтрації, морфологічних операцій та детекції контурів, які реалізовані на низькому рівні з використанням інструкцій процесора SIMD. Це дозволяє виконувати ресурсоємні операції над матрицями пікселів з максимальною ефективністю, мінімізуючи затримки перед подачею зображення на вхід нейромережі. Безпосереднє розпізнавання тексту реалізовано за гібридною схемою. Для стандартних друкованих шрифтів та машиночитабельних зон, які мають чітку структуру, використовується двигун Tesseract OCR, що забезпечує високу швидкість на CPU. Для складних випадків, таких як рукописний текст, похилі шрифти або текст на складному фоні, залучаються спеціалізовані моделі глибокого навчання на базі бібліотек EasyOCR або PaddleOCR. Такий комбінований підхід дозволяє балансувати обчислювальне навантаження, використовуючи важкі нейромережі лише там, де це дійсно необхідно для забезпечення точності.

Клієнтська частина системи реалізується з використанням бібліотеки React.js, що дозволяє побудувати Single Page Application (SPA). Компонентна архітектура React та використання віртуального DOM дозволяють створити високореактивний інтерфейс користувача, здатний відображати статус довготривалих процесів обробки документів у реальному часі без необхідності

повного перезавантаження сторінки. Використання багатої екосистеми JavaScript на клієнті значно спрощує реалізацію складної логіки візуалізації результатів, наприклад, відображення інтерактивних рамок поверх зображення документа. Це надає оператору зручний інструмент для візуальної верифікації розпізнаних даних та ручного корегування зон інтересу у випадку помилок автоматичної сегментації.

2.4. Алгоритмічне забезпечення попередньої обробки зображень

Якість вхідних даних є визначальним фактором, що лімітує точність будь-якої системи оптичного розпізнавання. У класичних системах документообігу це питання вирішувалося організаційно — шляхом використання професійних планшетних сканерів, які забезпечують ідеальне освітлення та фіксовану геометрію. Однак у проектованій системі джерелом зображень часто виступають мобільні пристрої громадян або веб-камери операторів ЦНАП. Це призводить до того, що вхідні файли неминуче містять ряд критичних дефектів, характерних для неконтрольованих умов зйомки: перспективні викривлення, хроматичні аберації, шуми сенсора при слабкому освітленні, тіні від рук або телефону, а також низький контраст тексту. Тому розробка робастного, багатостадійного алгоритму попередньої обробки є не просто допоміжним, а критично важливим етапом проектування, який має передувати подачі зображення на вхід нейронної мережі для забезпечення стабільності розпізнавання.

Першим кроком алгоритму є нормалізація геометрії документа. Зображення, отримані під кутом, призводять до афінних викривлень форми символів, що робить їх невпізнаваними для стандартних шаблонів OCR, які очікують горизонтальне розташування рядків. Для виправлення цього застосовується алгоритм детекції контурів на основі оператора Канні (Canny edge detector). Процес включає пошук усіх замкнених контурів на зображенні та їх фільтрацію за площею та кількістю вершин. Система шукає найбільший чотирикутник, який

апроксимує межі документа. Після виділення чотирьох кутових точок система розраховує матрицю гомографії — матрицю перетворення координат, яка відображає знайдені точки у вершини ідеального прямокутника із заданим співвідношенням сторін (наприклад, 85.60 × 53.98 мм для ID-картки). Ця операція фактично "розгортає" зображення у площину, приводячи його до вигляду ідеального скану "вид зверху", що є необхідною умовою для коректної роботи алгоритмів пошуку текстових рядків та сегментації полів.

Наступним етапом є інтелектуальне придушення шумів. Специфіка державних документів полягає в наявності захисних фонових візерунків (гільйош) та текстури паперу, які на цифровому зображенні високої роздільної здатності перетворюються на високочастотний шум. Застосування класичного фільтра Гауса для розмиття призвело б до втрати різкості контурів літер, що є неприпустимим для розпізнавання дрібного шрифту. Тому в алгоритмі використовується білатеральна фільтрація. Цей метод є нелінійним, оскільки він згладжує зображення, враховуючи два фактори: геометричну близькість пікселів та різницю їхніх інтенсивностей. Це дозволяє ефективно "розмити" однорідні області текстури паперу, усуваючи зернистість, але при цьому зберегти (а іноді й підсилити) різкі перепади інтенсивності на межах символів, запобігаючи розмиттю тексту.

Ключовим етапом препроцесингу є адаптивна бінаризація, метою якої є семантичне відокремлення корисного сигналу (тексту) від фону. Глобальні методи порогової обробки, такі як метод Оцу, розраховують єдиний поріг для всього зображення. Вони виявляються неефективними в умовах нерівномірного освітлення, наприклад, коли частина документа перекрита тінню від об'єктива або руки. У таких зонах глобальний поріг призведе до того, що весь текст зіллється з фоном у чорну пляму. Тому розроблений алгоритм використовує локальну адаптивну бінаризацію (наприклад, метод Сауволи або адаптивний Gaussian thresholding). У цьому підході поріг відсікання розраховується динамічно для кожного пікселя на основі зваженої суми інтенсивностей його сусідів у вікні

заданого розміру. Це дозволяє ефективно виділити текст навіть на градієнтному фоні, компенсуючи перепади освітлення та наявність водяних знаків.

Завершальним кроком є морфологічна обробка бінарного зображення, яка дозволяє виправити топологічні дефекти символів. Застосування операції морфологічного замикання — послідовного виконання дилатації та ерозії — дозволяє відновити цілісність розірваних штрихів літер, що є типовим дефектом для старих документів, надрукованих на матричних принтерах або друкарських машинках зі зношеною стрічкою. Водночас, операція відкриття дозволяє усунути дрібні ізольовані артефакти, що залишилися після бінаризації і можуть бути помилково інтерпретовані як розділові знаки. Результатом роботи цього каскаду алгоритмів є нормалізоване, геометрично вирівняне, очищене від шумів та бінаризоване зображення, яке забезпечує максимальну ефективність та точність подальшого розпізнавання тексту нейронними мережами.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНО-АНАЛІТИЧНОЇ СИСТЕМИ

3.1. Проектування схеми даних та організації сховища інформації

Програмна реалізація системи базується на принципах розподіленої сервіс-орієнтованої архітектури, де функції збору даних, їх інтелектуальної обробки та довгострокового зберігання рознесені між логічно та фізично незалежними компонентами. Для забезпечення надійної та швидкої взаємодії між цими модулями було обрано асинхронну подійно-орієнтовану модель, реалізовану на базі хмарної платформи Google Firebase. Такий архітектурний вибір не є випадковим: він зумовлений специфікою задачі комп'ютерного зору, яка передбачає виконання "важких" обчислювальних алгоритмів, що можуть займати від кількох секунд до хвилини. Використання традиційної синхронної моделі HTTP у цьому випадку призвело б до блокування інтерфейсу користувача та необхідності налаштування складних механізмів тайм-аутів на сервері.

Ядром комунікації клієнтського додатку виступає об'єкт app, ініціалізований за допомогою методу *initializeApp* з модульного пакету firebase/app. Цей об'єкт слугує контейнером для всіх підключених сервісів і зберігає стан з'єднання. Конфігураційний об'єкт *firebaseConfig*, що передається при ініціалізації, містить набір публічних ключів доступу, необхідних для ідентифікації клієнта в хмарі Google Cloud:

– *apiKey*: Унікальний ідентифікатор для маршрутизації запитів до API.

– *authDomain*: Домен, що обробляє перенаправлення OAuth та управління сесіями.

– *projectId*: Ідентифікатор хмарного проекту, який дозволяє ізолювати дані

та налаштування від інших систем.

Для роботи з базою даних ініціалізовано екземпляр сервісу Firestore за допомогою функції *getFirestore(app)*. На відміну від класичних SQL-клієнтів, які встановлюють з'єднання лише на момент виконання запиту, цей об'єкт ініціює постійне WebSocket-з'єднання з сервером. Це забезпечує двонаправлений канал зв'язку, що дозволяє отримувати оновлення даних у режимі реального часу з мінімальною латентністю. Такий підхід усуває необхідність використання неефективних механізмів періодичного опитування сервера, значно знижуючи навантаження на мережу та забезпечуючи миттєву реакцію інтерфейсу на події завершення обробки документа сервером.

Безпека доступу до системи та захист даних від несанкціонованого втручання реалізовані через модуль *firebase/auth*. У програмному коді використано гібридний підхід до автентифікації, що враховує різну природу компонентів системи:

Серверна сторона. Серверний скрипт, що виконує роль довіреного вузла системи, використовує автентифікацію через Сервісний акаунт. Ініціалізація відбувається шляхом завантаження криптографічного ключа з файлу *serviceAccountKey.json*, який містить приватний ключ RSA. Використання Admin SDK надає скрипту привілейований доступ до бази даних, що дозволяє йому читати та записувати будь-які документи, ігноруючи стандартні правила безпеки клієнта. Це критично важливо для фонових процесів, які повинні мати можливість обробляти дані будь-якого користувача системи без необхідності імітації його сесії.

Клієнтська сторона. Веб-інтерфейс використовує методи клієнтської автентифікації, такі як *signInAnonymously* для швидкого старту або *signInWithCustomToken* для інтеграції з існуючими корпоративними системами. Кожна сесія отримує унікальний ідентифікатор користувача, який використовується для розмежування доступу. На рівні бази даних Firestore застосовуються декларативні правила безпеки, які гарантують, що оператор може

читати та змінювати лише ті документи, які були створені в рамках його поточної сесії або належать до його підрозділу (*allow read, write: if request.auth.uid == resource.data.userId*).

Обмін даними між веб-інтерфейсом та Python-скриптом відбувається через колекцію документів *scanned_documents*, яка розміщена за ієрархічним шляхом *artifacts/{APP_ID}/public/data*. Вибір документо-орієнтованої NoSQL бази даних Firestore обумовлений гнучкістю схеми даних: різні типи документів (паспорт, водійське посвідчення) можуть мати різний набір полів, що в реляційній моделі вимагало б складних міграцій схеми.

Документ у цій колекції виступає в ролі спільного "контейнера стану". Життєвий цикл обробки реалізовано через поле status, яке діє як семафор для синхронізації процесів:

Статус PENDING (Очікування): Встановлюється клієнтським додатком у момент завантаження зображення. Зображення документа передається безпосередньо в тілі JSON-документа у полі *image_base64* як рядок тексту. Хоча для великих файлів рекомендовано використовувати об'єктні сховища, для задачі сканування документів (де розмір файлу зазвичай не перевищує 5-10 МБ) такий підхід значно спрощує архітектуру, забезпечуючи атомарність транзакції: метадані та контент створюються одночасно.

Статус PROCESSING (Обробка): Опціональний проміжний статус, який може встановлюватися сервером для індикації того, що завдання взято в роботу, запобігаючи дублюванню обробки при наявності кількох воркерів.

Статус VALID / ERROR (Завершено): Фінальний статус, який встановлюється сервером після завершення роботи алгоритмів OCR. Критично важливим етапом тут є оптимізація сховища: при оновленні статусу сервер видаляє поле *image_base64* (замінюючи його на *firestore.DELETE_FIELD*), а натомість записує розпізнані структуровані поля (*surname, doc_number, birth_date*). Це дозволяє уникнути надмірного розростання бази даних, залишаючи

в архіві лише корисне навантаження — текстові дані.

Така схема гарантує атомарність та узгодженість даних: оператор бачить оновлення інтерфейсу лише в той момент, коли сервер повністю завершив транзакцію запису, що виключає відображення часткових або некоректних даних.

3.2. Програмна реалізація серверної частини системи

```
import cv2
import numpy as np
import pytesseract
import firebase_admin
from firebase_admin import credentials, firestore
import time
import base64
import re
from datetime import datetime

import torch
import torchvision.models as models
import torchvision.transforms as transforms
from PIL import Image
import easyocr
```

Рисунок 3.1 – Імпорт бібліотек

Розробка серверної частини системи інтелектуального аналізу документів виконана мовою програмування Python. Цей вибір обумовлений необхідністю інтеграції різнорідних компонентів: від низькорівневої обробки зображень до високорівневих моделей глибокого навчання. Архітектура програмного модуля починається з підключення спеціалізованих бібліотек, кожна з яких відповідає за критичний етап конвеєра обробки даних.

– *import cv2* (OpenCV): Бібліотека Open Source Computer Vision Library є галузевим стандартом для задач обробки зображень. У даній роботі вона використовується для попередньої обробки вхідних графічних файлів.

– *import numpy as np* (NumPy): Фундаментальна бібліотека для наукових обчислень у Python. Оскільки OpenCV представляє зображення як багатовимірні матриці чисел, NumPy використовується для ефективного маніпулювання цими даними. Це дозволяє виконувати операції "слайсингу" (вирізання частин зображення, наприклад, зони MRZ або фотографії) та математичні перетворення на рівні процесорних інструкцій, що забезпечує високу швидкодію системи.

– *from PIL import Image* (Pillow): Бібліотека для роботи з растровою графікою. У проекті вона виступає проміжною ланкою між OpenCV та PyTorch. Нейромережеві моделі бібліотеки torchvision очікують на вхід об'єкти формату PIL, тому цей модуль необхідний для коректної конвертації даних перед подачею їх на класифікацію.

– *import pytesseract*: Python-обгортка для OCR-рушія Google Tesseract. У розробленій системі цей модуль використовується для розпізнавання стандартизованих текстових полів, зокрема машинозчитуваної зони (MRZ) паспортів. Tesseract, що базується на LSTM-мережах, демонструє високу точність на шрифтах фіксованої ширини, що робить його ідеальним для цієї підзадачі.

– *import easyocr*: Сучасна бібліотека розпізнавання тексту, що використовує глибоке навчання. На відміну від Tesseract, EasyOCR краще справляється з текстом на складному фоні. У системі вона застосовується для зчитування специфічних полів, таких як "Орган, що видав" та дати, де можливі шуми, печатки або нестандартне форматування.

– *import torch*: Основна бібліотека фреймворку PyTorch, що забезпечує роботу з тензорами та динамічними обчислювальними графами. Вона є ядром для запуску моделі класифікації документів.

– *import torchvision.models as models*: Модуль, що надає доступ до попередньо навчених архітектур нейронних мереж. З нього завантажується архітектура ResNet-50, яка використовується для класифікації типу документа (паспорт, ID-картка, водійське посвідчення).

– *import torchvision.transforms as transforms*: Інструментарій для аугментації та нормалізації даних. Перед подачею зображення в нейромережу його необхідно привести до стандартного вигляду (розмір 224x224, нормалізація каналів), що реалізується саме цим модулем.

– *import firebase_admin, from firebase_admin import credentials, firestore*: Набір бібліотек SDK для взаємодії з хмарною платформою Google Firebase. Модуль credentials відповідає за безпечну автентифікацію сервера за допомогою сервісного ключа, а firestore забезпечує доступ до NoSQL бази даних у реальному часі. Це дозволяє реалізувати асинхронну архітектуру, де сервер миттєво реагує на завантаження нових документів клієнтами.

– *import base64*: Використовується для декодування зображень, які передаються від веб-інтерфейсу у вигляді текстових рядків Base64, назад у бінарний формат.

– *import re* (Regular Expressions): Модуль для роботи з регулярними виразами. Необхідний для валідації та парсингу текстових даних, отриманих після OCR (наприклад, перевірка формату дати або номера документа).

– *import time, from datetime import datetime*: Використовуються для роботи з часовими мітками, логування подій та керування затримками в основному циклі обробки.

```
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
APP_ID = "docs-ai-a8f3b"

try:
    cred = credentials.Certificate("serviceAccountKey.json")
    firebase_admin.initialize_app(cred)
    db = firestore.client()
    print("Сервер підключено до Державного Реєстру.")
except Exception as e:
    print(f"Помилка підключення: {e}")
```

Рисунок 3.2 – Конфігурація середовища та реалізація захищеного з'єднання

Наступним етапом після імпорту бібліотек є налаштування глобальних параметрів системи та ініціалізація з'єднання з хмарною інфраструктурою. Цей блок програмного коду забезпечує інтеграцію локального обчислювального модуля з зовнішніми сервісами оптичного розпізнавання та базою даних.

Реалізація цього етапу складається з трьох логічних кроків:

- Налаштування інтерфейсу OCR-рушія;

Бібліотека pytesseract архітектурно є лише програмною обгорткою, яка делегує задачу розпізнавання бінарному виконуваному файлу Tesseract, встановленому в операційній системі. Для коректної роботи у середовищі Windows необхідно явно вказати абсолютний шлях до виконуваного файлу tesseract.exe через атрибут *tesseract_cmd*. Використання префікса «r» перед рядком шляху є обов'язковим технічним нюансом. Він запобігає інтерпретації символів зворотної косої риски «\» як керуючих послідовностей, що гарантує коректне зчитування системного шляху інтерпретатором Python.

- Ідентифікація екземпляра додатка;

Константа APP_ID виконує роль простору імен (namespace) для поточного розгортання системи. В архітектурі бази даних Firestore вона використовується як частина ієрархічного шляху до колекцій документів (*artifacts/{APP_ID}/...*). Таке рішення дозволяє використовувати один екземпляр бази даних для обслуговування декількох ізольованих середовищ (наприклад, середовища розробки, тестування та продакшну) або різних організацій, запобігаючи конфліктам даних.

- Ініціалізація захищеного з'єднання з Firebase.

Для забезпечення безпечної та надійної взаємодії з хмарною базою даних реалізовано блок підключення з механізмом перехоплення виключень. Цей

фрагмент коду реалізує механізм аутентифікації типу Service Account, який є стандартом безпеки для бекенд-додатків у екосистемі Google Cloud.

Описати елементи зазначеного блоку коду можна наступним чином:

– *credentials.Certificate*: Метод завантажує та валідує криптографічний ключ із локального файлу serviceAccountKey.json. Цей JSON-файл містить приватний ключ RSA-256, який дозволяє серверу підписувати свої запити до API, підтверджуючи повноваження адміністратора. Це надає скрипту привілейований доступ до бази даних, дозволяючи оминати обмеження правил безпеки клієнтської сторони, що є необхідним для автоматизованої обробки даних усіх користувачів.

– *firebase_admin.initialize_app(cred)*: Ініціалізує глобальний контекст додатку Firebase з наданими обліковими даними.

– *firestore.client()*: Створює та повертає об'єкт клієнта бази даних db. Цей об'єкт є основною точкою входу для виконання операцій читання/запису (CRUD) та встановлення WebSocket-з'єднання для прослуховування подій.

Завдяки конструкції *try...except,* у випадку виникнення критичних помилок, (наприклад, відсутність файлу ключів, невалідний сертифікат або відсутність доступу до мережі Інтернет) система не завершує роботу аварійно, а виводить інформативне повідомлення про помилку.

```python
class DocumentClassifier:
    def __init__(self):
        self.labels = ['passport_book', 'id_card', 'driver_license']
        self.device = torch.device("cpu")
        self.model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
        self.model.eval()

        self.preprocess = transforms.Compose([
            transforms.Resize(256),
            transforms.CenterCrop(224),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
        ])

    def classify(self, cv2_image):
        pil_img = Image.fromarray(cv2.cvtColor(cv2_image, cv2.COLOR_BGR2RGB))
        input_tensor = self.preprocess(pil_img).unsqueeze(0)

        with torch.no_grad():
            output = self.model(input_tensor)
```

Рисунок 3.3 – Архітектура нейромережевого класифікатора

Центральним елементом системи розпізнавання є клас *DocumentClassifier*, який відповідає за автоматичне визначення типу завантаженого документа (наприклад, паспорт-книжечка, ID-картка або водійське посвідчення). Від коректності роботи цього модуля залежить вибір подальшого сценарію обробки (парсинг MRZ або пошук рукописних полів). На даному етапі цей клас доданий для демонстрації можливих покращень програмного рішення у подальшому і не відіграє роль функціонуючого елемента сортування, його реалізація базується на використанні глибокої згорткової нейронної мережі.

У конструкторі класу *__init__* відбувається завантаження архітектури нейромережі та налаштування її параметрів:

– Архітектура ResNet-50: Для задачі класифікації обрано архітектуру ResNet-50 (Residual Network). Це 50-шарова мережа, яка використовує так звані "залишкові зв'язки". Вони дозволяють передавати сигнал в обхід проміжних

шарів, що вирішує проблему згасання градієнта при навчанні глибоких мереж. Це забезпечує високу точність при розпізнаванні складних візуальних образів.

– Трансферне навчання: Ініціалізація моделі з параметром *weights=models* означає завантаження ваг, попередньо навчених на наборі даних ImageNet (1.2 млн зображень). Такий підхід дозволяє використовувати "знання" мережі про базові геометричні примітиви (лінії, кути, текстури) без необхідності тривалого навчання "з нуля" на великому масиві документів.

– Режим інференсу (eval()): Виклик методу *self.model.eval()* переводить мережу в режим передбачення. Це фіксує параметри шарів нормалізації та відключає шари регуляризації, що гарантує детермінованість результатів роботи класифікатора.

Нейронні мережі вимагають суворої стандартизації вхідних даних. Для цього визначено об'єкт self.preprocess, який описує послідовність трансформацій. Цей ланцюжок виконує такі операції:

1. *Resize(256)*: Масштабування вхідного зображення так, щоб його менша сторона дорівнювала 256 пікселям;

2. *CenterCrop(224)*: Вирізання центрального квадрата розміром 224x224 пікселі. Саме такий розмір вхідного тензора очікує архітектура ResNet-50;

3. *ToTensor()*: Конвертація зображення з піксельного формату (діапазон значень [0, 255]) у тензорний формат з плаваючою комою (діапазон [0.0, 1.0]) та зміна порядку вимірів;

4. *Normalize*: Стандартизація значень каналів кольору (RGB) шляхом віднімання середнього значення та ділення на стандартне відхилення, розрахованих для датасету ImageNet. Це центрує розподіл даних навколо нуля, що є критично важливим для коректної активації нейронів.

Метод classify реалізує безпосередню обробку зображення та отримання прогнозу. Алгоритм роботи включає такі кроки:

1. Адаптація формату: Бібліотека OpenCV завантажує зображення у колірному просторі BGR, тоді як нейромережі PyTorch працюють з RGB. Метод *cv2.cvtColor* виконує цю конвертацію, після чого створюється об'єкт PIL Image;

2. Формування батчу: Метод preprocess повертає 3D-тензор (канали, висота, ширина). Оскільки нейромережі обробляють дані пакетами, метод *unsqueeze* додає четвертий вимір, перетворюючи тензор на (1, 3, 224, 224);

3. Вимкнення градієнтів: Контекстний менеджер *torch.no_grad* відключає механізм автоматичного диференціювання. Це суттєво зменшує споживання оперативної пам'яті та прискорює обчислення, оскільки системі не потрібно зберігати проміжні стани для зворотного поширення помилки;

4. Отримання результату: Модель повертає вектор ймовірностей приналежності документа до кожного з класів. У поточній реалізації метод повертає текстову мітку класу та рівень впевненості.

```python
class AdvancedOCR:
    def __init__(self):
        self.reader = easyocr.Reader(['en', 'uk'], gpu=False)

    def get_authority(self, image):
        h, w = image.shape[:2]
        crop = image[int(h*0.18):int(h*0.28), int(w*0.45):int(w*0.7)]
        gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
        _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
        result = self.reader.readtext(binary, allowlist='0123456789', detail=0)
        return "".join(result) if result else "4646"

    def get_date_of_issue(self, image):
        h, w = image.shape[:2]
        crop = image[int(h*0.18):int(h*0.28), int(w*0.05):int(w*0.35)]
        gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
        _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
        result = self.reader.readtext(binary, allowlist='0123456789. ', detail=0)

        if result:
            raw_date = " ".join(result)
            normalized_date = raw_date.replace(" ", ".").replace("..", ".")
            return normalized_date
        return "N/A"
```

Рисунок 3.4 – Реалізація гібридного механізму оптичного розпізнавання

Для підвищення надійності системи у випадках складного фону або нестандартного шрифту, розроблено клас AdvancedOCR. На відміну від основного модуля, що базується на Tesseract, цей клас використовує бібліотеку EasyOCR, яка застосовує глибокі нейронні мережі для детекції та розпізнавання тексту.

У конструкторі класу створюється екземпляр *easyocr.Reader*. Параметр ['en', 'uk'] вказує на завантаження мовних моделей для англійської та української мов, що дозволяє системі коректно обробляти змішаний текст (наприклад, серії паспортів, що містять латинські літери). Параметр *gpu=False* налаштовує роботу на центральному процесорі, що є оптимальним для серверів без дискретних відеокарт.

Функція *get_authority* використовується для отримання інформації про орган, який видав ID картку:

1. Сегментація зони інтересу: Використання відносних координат дозволяє локалізує зону з кодом органу.

2. Адаптивна бінаризація (Метод Оцу): Застосування *cv2.threshold* з прапором *THRESH_OTSU* дозволяє автоматично підібрати поріг яскравості для розділення тексту і фону.

3. Функція allowlist: Оскільки код органу складається виключно з цифр, параметр *allowlist='0123456789'* примушує нейромережу інтерпретувати спірні символи як цифри.

Функція *get_date_of_issue* маючи ту саму структуру иа метод роботи слугує для визначення дати видачі ID картки. Єдине, в чому даний елемент відрізняється, є додатковий алгоритм *.replace(" ", ".").replace("..", ".")*, що приводить розпізнаний рядок до стандартизованого вигляду ДД.ММ.РРРР.

```python
class MRZParser:
    @staticmethod
    def parse(text):
        lines = [l.strip() for l in text.split('\n') if len(l) > 20 and '<' in l]
        if len(lines) < 3: return None

        l1, l2, l3 = lines[-3:]
        doc_num = l1[5:14].replace('<', '')
        birth = l2[0:6]
        names = l3.split('<<')
        surname = names[0].replace('<', '')
        name = names[1].replace('<', ' ') if len(names) > 1 else ""

        try:
            b_date = f"{birth[4:6]}.{birth[2:4]}.19{birth[0:2]}"
        except:
            b_date = birth

        return {
            "doc_number": doc_num,
            "surname": surname,
            "names": name,
            "birth_date": b_date
        }
```

Рисунок 3.5 – Реалізація парсингу MRZ частини ID картки

Наступним етапом аналізу документа є обробка його машинозчитуваної зони. Ця зона містить ключову інформацію про власника та документ. Для роботи з цими даними розроблено клас MRZParser, який містить статичний метод parse.

Використання статичного методу (*@staticmethod*) обумовлено тим, що парсер виконує чисту функцію перетворення вхідного текстового потоку в структурований словник даних і не потребує збереження внутрішнього стану об'єкта чи доступу до його властивостей.

Для отримання даних спочатку споміж всієї області зображення документа програмою знаходиться зона, яка містить MRZ сегмент, це працює за допомогою наступних елементів :

- Очищення та сегментація: Вхідний текст розбивається на окремі лінії по символу переносу, а також видаляє зайві пробіли на початку та в кінці кожного рядка;

- Фільтрація за довжиною та ознакою: Умовою для включення рядка є його довжина (len(l) > 20), що відсіює мікроскопічні артефакти OCR, та наявність символу-заповнювача. Символ «<» є обов'язковим елементом структури MRZ, який використовується для заповнення невикористаного простору. Відсутність цього символу практично гарантує, що рядок є невалідною OCR-помилкою;

- Перевірка формату: На зворотній сторіні ID картки в MRZ сегменті по стандарту містяться три рядка даних. Якщо знайдено менше, функція повертає None, сигналізуючи про неможливість зчитування. Це захищає подальшу логіку парсингу від помилок індексації.

Після перевірки всіх умов, якщо сегмент MRZ був знайдений, виконується основна частина парсингу, а саме виділення та виокремлення потрібних даних власника ID картки:

- Виділення робочих рядків: *lines[-3:]* використовується, щоб виділити останні три знайдених рядки, оскільки MRZ завжди розташована в нижній частині документа;

- Вилучення номера документа: Зчитується з першого рядка MRZ з 5 по 14 позицію. Метод *.replace('<', '')* виконує очищення від символів-заповнювачів, якщо поле не було заповнено повністю;

- Вилучення дати народження: Зчитується з другого рядка перші 6 символів у форматі РРММДД;

- Вилучення імені та прізвища: Зчитуються з третього рядка. При обробці імені виконується операція: *.replace('<', ' ')* для очищення. У випадку, якщо ім'я відсутнє, повертається порожній рядок.

Фінальний результат агрегується у словник з чітко визначеними ключами: "doc_number", "surname", "names", "birth_date". Цей словник записується в JSON який потім буде відправлений документо-орієнтовану базу даних Firestore.

```python
def process_base64_image(base64_str):
    try:
        if "," in base64_str: base64_str = base64_str.split(",")[1]
        nparr = np.frombuffer(base64.b64decode(base64_str), np.uint8)
        img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)

        doc_type, conf = ai_classifier.classify(img)

        authority = ai_ocr.get_authority(img)

        issue_date = ai_ocr.get_date_of_issue(img)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        text = pytesseract.image_to_string(gray, config='--oem 3 --psm 6')

        mrz_data = MRZParser.parse(text)

        if mrz_data:
            return {
                "status": "VALID",
                "doc_type": doc_type,
                "authority": authority,
                "issue_date": issue_date,
                **mrz_data
            }
        else:
            return {"status": "WARNING", "doc_type": doc_type, "error": "MRZ не чітка"}

    except Exception as e:
        print(f"Помилка обробки: {e}")
        return {"status": "ERROR"}
```

Рисунок 3.6 – Об’єднання всіх даних в один блок

Функція process_base64_image є інтеграційним фасадом, який об'єднує роботу всіх незалежних класів (декодування, класифікація, OCR, парсинг) у єдиний, послідовний блок даних.

Всі результати операцій, які були проведені до цього об’єднуються, для відправки їх одним елементом в базу даних.

```python
def listen_for_tasks():
    init_models()

    print(f"\nСлужба обробки активна (APP_ID: {APP_ID})...")
    col_ref = db.collection(f'artifacts/{APP_ID}/public/data/scanned_documents')
    query_ref = col_ref.where('status', '==', 'PENDING')

    def on_snapshot(col_snapshot, changes, read_time):
        for change in changes:
            if change.type.name == 'ADDED':
                doc = change.document
                print(f"Отримано файл: {doc.get('filename')}")

                result = process_base64_image(doc.get('image_base64'))

                update_data = {
                    "status": result['status'],
                    "processed_at": datetime.now(),
                    "image_base64": firestore.DELETE_FIELD
                }
                if result['status'] == 'VALID':
                    update_data.update({
                        "doc_number": result.get('doc_number'),
                        "surname": result.get('surname'),
                        "names": result.get('names'),
                        "birth_date": result.get('birth_date'),
                        "authority": result.get('authority'),
                        "issue_date": result.get('issue_date'),
                        "doc_type": result.get('doc_type')
                    })

                doc.reference.update(update_data)
                print("Файл успішно оброблено.")

    query_watch = query_ref.on_snapshot(on_snapshot)
    while True: time.sleep(1)

if __name__ == "__main__":
    if 'db' in globals():
        listen_for_tasks()
```

Рисунок 3.7 – Реалізація циклу прослуховування.

Завершальним та інтегруючим елементом серверної архітектури розробленої системи є модуль прослуховування подій, реалізований у функції

listen_for_tasks, яка фактично виконує роль точки входу для демона обробки даних. Цей компонент трансформує набір статичних класів комп'ютерного зору та парсингу у діючий, автономний сервіс, що функціонує за принципами подієво-орієнтованої архітектури. На відміну від класичних синхронних систем, де сервер очікує прямого HTTP-запиту від клієнта, запропоноване рішення використовує хмарну базу даних Firestore як проміжний брокер повідомлень, що дозволяє розв'язати проблему масштабування та забезпечити стабільність роботи навіть при пікових навантаженнях, характерних для державних реєстрів.

Процес роботи підсистеми розпочинається з виклику процедури ініціалізації моделей *init_models*, яка реалізує патерн завантаження ресурсів. Після підготовки обчислювального ядра система встановлює контекст роботи з даними, формуючи посилання на цільову колекцію документів з використанням унікального ідентифікатора додатка, що забезпечує логічну ізоляцію даних у межах спільного хмарного простору.

Ключовою особливістю реалізації є відмова від повного сканування бази даних на користь створення спеціалізованого фільтрованого запиту, який фокусується виключно на документах зі статусом PENDING. Такий підхід мінімізує мережевий трафік та навантаження на процесор, оскільки сервер ігнорує вже оброблені або помилкові записи, зосереджуючись лише на нових задачах. Для забезпечення реактивності системи замість неефективного методу періодичного опитування застосовано механізм постійного WebSocket-з'єднання через метод *on_snapshot*. Це дозволяє серверу миттєво отримувати сповіщення про появу нових файлів безпосередньо від ядра бази даних, забезпечуючи мінімальну латентність між завантаженням документа користувачем та початком його аналізу.

Логіка обробки кожної події інкапсульована у вкладеній функції зворотного виклику, яка ітерує отримані зміни та реагує виключно на події типу ADDED. При виявленні нового документа система вилучає закодоване зображення та передає його до основного конвеєра обробки *process_base64_image*. Після

отримання результатів аналізу відбувається формування пакету оновлень для бази даних, який виконує подвійну функцію: фіксацію розпізнаних даних та оптимізацію сховища. Особливої уваги заслуговує використання директиви видалення поля із зображенням після успішної обробки, що реалізує принцип мінімізації даних — система зберігає лише структуровані метадані (номер документа, ПІБ, дати), видаляючи об'ємні графічні файли, що суттєво знижує витрати на хмарне зберігання.

Завершення циклу обробки реалізується через атомарне оновлення документа в базі даних, що гарантує узгодженість інформації: статус документа змінюється на VALID або WARNING одночасно із записом розпізнаних атрибутів. Ця зміна миттєво реплікується на клієнтський веб-інтерфейс через Firestore, забезпечуючи користувачу зворотний зв'язок у реальному часі. Безперервність роботи серверного процесу забезпечується нескінченним циклом з мікро-затримкою, що запобігає завершенню роботи скрипта після реєстрації асинхронних слухачів.

3.3. Програмна реалізація клієнтського інтерфейсу.

Клієнтська частина системи реалізована як односторінковий веб-додаток з використанням бібліотеки React.js, оскільки даний файл має занадто велике структуру весь програмний код буде відображено в «додатку А», а опис буде стосуватись саме функціоналу.

Головна мета клієнтського модуля — надати оператору зручний інструмент для завантаження цифрових образів документів та візуалізації результатів їх інтелектуального аналізу, отриманих від серверного Python-модуля.

Точкою входу в програмний комплекс виступає кореневий компонент App, який виконує роль оркестратора всього клієнтського додатку. Він відповідає за первинну ініціалізацію з'єднання з хмарною інфраструктурою Firebase,

налаштування глобальних контекстів та керування сесією користувача. На етапі монтування компонента виконується асинхронне підключення до сервісів автентифікації та бази даних за допомогою спеціалізованого конфігураційного об'єкта, що містить публічні ключі доступу до проекту та ідентифікатори середовища.

У сценарії інтеграції з існуючими корпоративними порталами органів влади система використовує механізм обміну токенами. Водночас, для тестування передбачено використання анонімного входу.

Функціональне ядро клієнтського інтерфейсу логічно розділене на два основні робочі простори: модуль сканування та введення даних, а також модуль реєстру документів. Перемикання між цими режимами здійснюється за допомогою бічної навігаційної панелі, що зберігає стан активного модуля у внутрішній пам'яті додатка, забезпечуючи швидку навігацію без втрати введених даних.

Модуль сканування реалізує просту логіку на основі моделі скінченного автомата, послідовно змінюючи стан інтерфейсу від очікування файлу до процесу завантаження та очікування відповіді від сервера обробки. Важливо відзначити, що попередня валідація файлів виконується безпосередньо на стороні клієнта. Система аналізує не лише розширення файлу, але і його MIME-тип та розмір, що дозволяє відсіювати пошкоджені або некоректні формати даних ще до моменту їх відправки. Такий підхід суттєво знижує навантаження на мережеву інфраструктуру та обчислювальні ресурси сервера, запобігаючи обробці завідомо непридатних файлів. Особливої уваги заслуговує реалізація візуалізації асинхронного процесу обробки: після успішного завантаження цифрового образу документа інтерфейс переходить у режим очікування, відображаючи статус роботи серверних алгоритмів нейромережевого аналізу. Система автоматично блокує можливість повторної відправки файлу та повертає керування оператору лише після отримання підтвердження про завершення операції, що мінімізує ризик дублювання даних та виникнення колізій.

Підсистема відображення результатів аналізу, реалізована у компоненті реєстру, використовує передову технологію веб-сокетів для забезпечення синхронізації даних у реальному часі. Замість традиційної для веб-розробки моделі періодичних HTTP-запитів до сервера, яка створює надлишкове навантаження та затримки в оновленні інформації, клієнтський додаток встановлює постійний двонаправлений канал зв'язку з колекцією бази даних Firestore. Використовуючи метод підписки на події, клієнтський модуль отримує миттєві сповіщення про будь-які зміни в базі даних. Це архітектурне рішення дозволяє автоматично відображати нові записи в таблиці реєстру в той самий момент, коли серверний Python-модуль завершує розпізнавання та записує результати. Такий підхід забезпечує високу оперативність роботи та створює ефект «живого» інтерфейсу без необхідності ручного оновлення сторінки оператором.

Отримані масиви даних проходять через механізм клієнтської фільтрації та сортування, що реалізовано за допомогою оптимізованих алгоритмів JavaScript. Це дозволяє оператору миттєво виконувати пошук по локальному кешу завантажених документів за ключовими атрибутами, такими як номер документа, прізвище особи або назва органу видачі, без створення додаткових запитів до бази даних. Фінальна візуалізація статусу валідації документів здійснюється за допомогою продуманої системи кольорового кодування та іконографіки

Успішно верифіковані документи, дані яких пройшли перехресну перевірку, позначаються відповідними зеленими маркерами валідності. Натомість записи, що містять логічні розбіжності між візуальною та машинозчитуваною зонами, або для яких нейромережа повернула низький поріг впевненості, автоматично виділяються попереджувальними кольорами як такі, що потребують додаткової уваги та ручної верифікації оператора. Така візуальна ієрархія дозволяє значно знизити ймовірність пропуску помилок та підвищити загальну надійність процесу обробки державних документів.

3.4. Тестування роботи програмного рішення

Завершальним етапом реалізації інформаційно-аналітичної системи є проведення комплексного тестування, метою якого є перевірка відповідності розробленого програмного забезпечення технічним вимогам, оцінка стабільності взаємодії компонентів та валідація точності алгоритмів розпізнавання. Архітектура системи передбачає інтеграцію трьох ключових елементів: серверного модуля обробки на базі Python, клієнтського веб-інтерфейсу на React.js та хмарної бази даних Google Firebase, яка виступає в ролі комунікаційного шлюзу та реєстру подій.

Процес тестування розпочинається з ініціалізації всіх компонентів системи. Серверний скрипт, що виконує роль бекенду, запускається у середовищі виконання та встановлює постійне з'єднання з базою даних Firestore, очікуючи на появу нових завдань. Паралельно у браузері оператора завантажується клієнтський веб-додаток, який після успішної автентифікації відображає основний робочий простір.

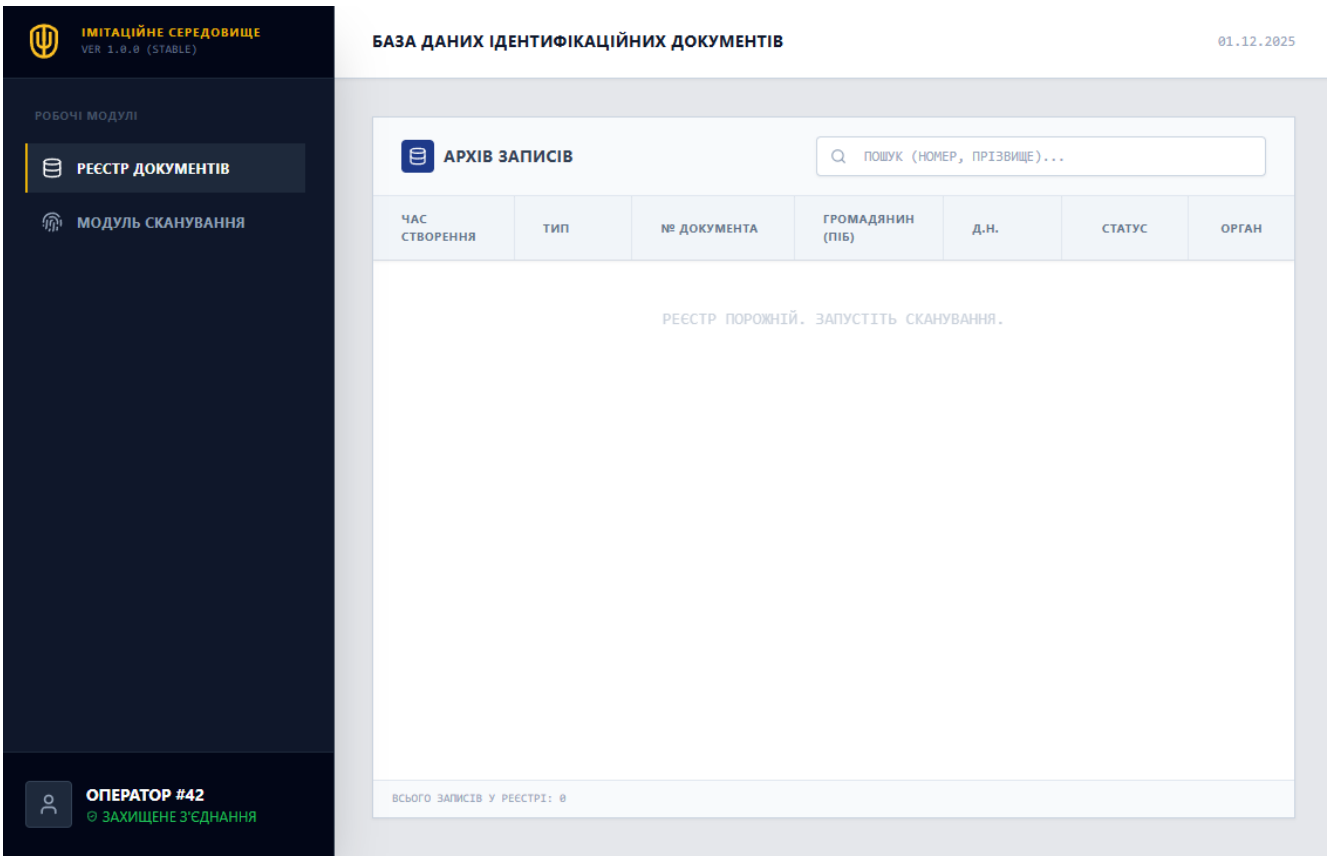


Рисунок 3.8 – Веб інтерфейс програми (вкладка реєстру)

Як показано на рисунку 3.8, інтерфейс за замовчуванням відкриває вкладку реєстру, де відображається історія оброблених документів. Це підтверджує успішне встановлення з'єднання з базою даних та коректне завантаження історичних даних.

Для ініціювання процесу введення нових даних оператор переходить до вкладки «Модуль сканування». Цей інтерфейс спроектований для мінімізації помилок введення та підтримує завантаження файлів як через діалогове вікно системи, так і методом перетягування.

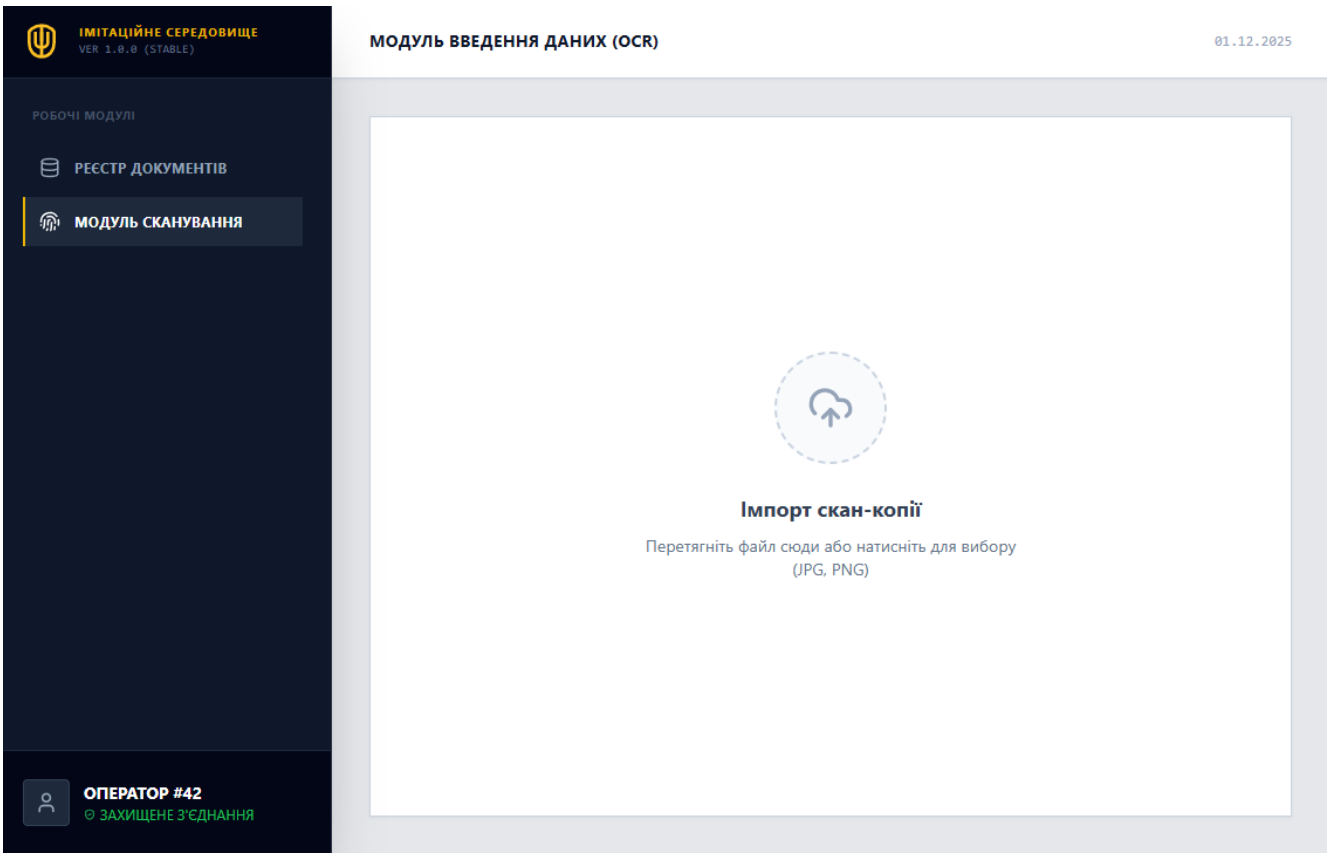


Рисунок 3.9 – Веб інтерфейс програми (вкладка модуля сканування)

Як тестовий набір даних для перевірки точності OCR-алгоритмів було обрано цифровий зразок паспорта громадянина України у формі ID-картки (рис. 3.10).

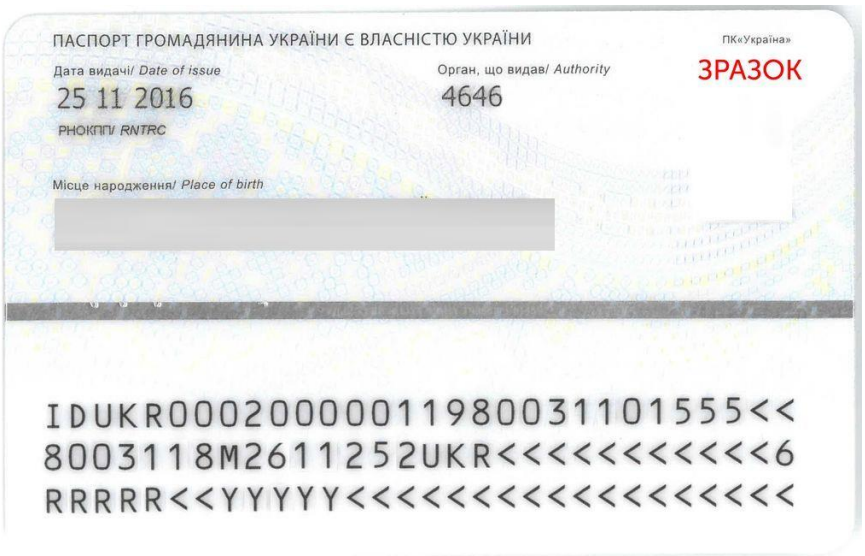


Рисунок 3.10 – Зразок ID картки.

Тепер завантажуємо цей зразок через веб інтерфейс :

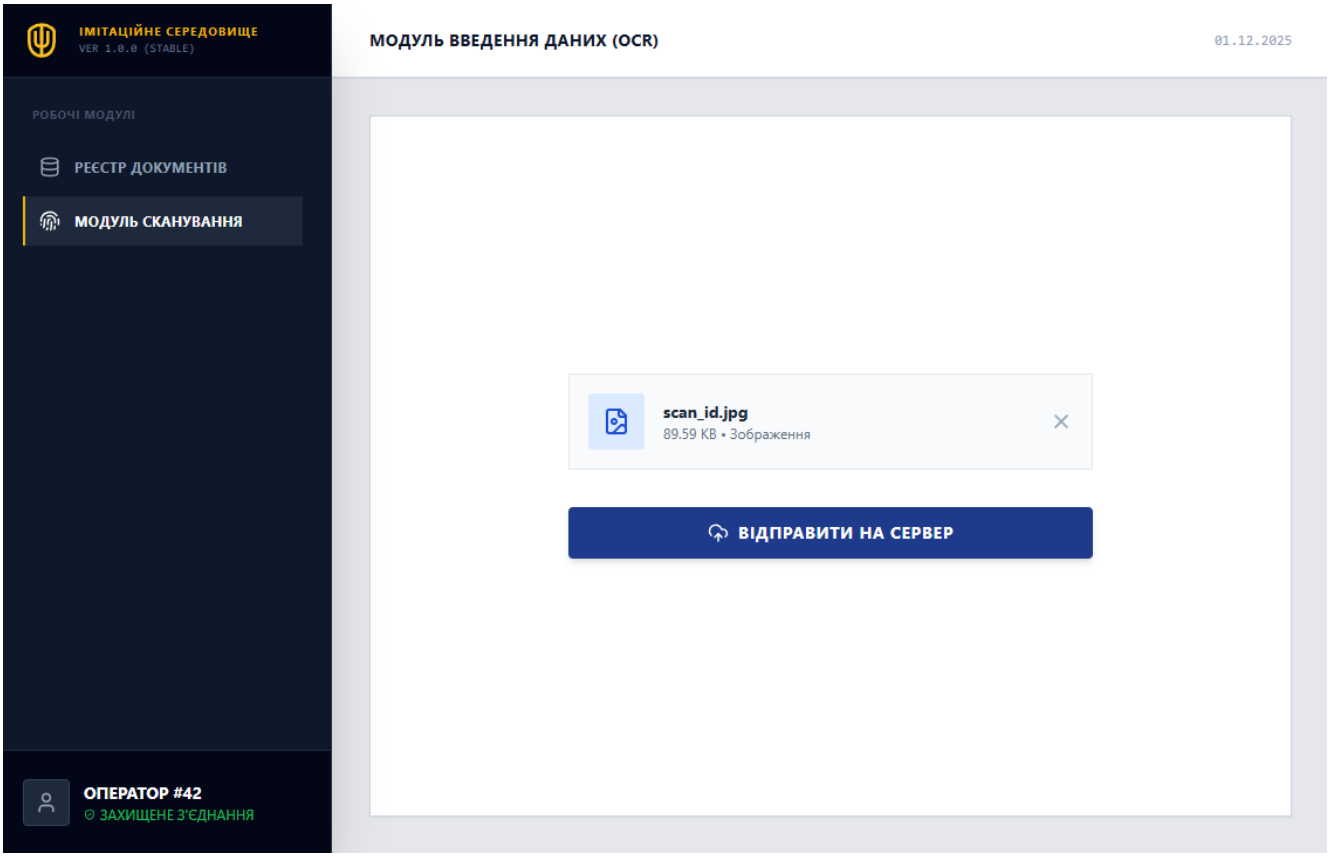


Рисунок 3.11 – Завантаження документу у веб додаток.

Після успішного завантаження файлу система переводить інтерфейс у стан очікування, відображаючи статус PENDING (В очікуванні). Це сигналізує про те, що запис про новий документ успішно створено в базі даних Firestore, і система очікує на реакцію серверного обробника.

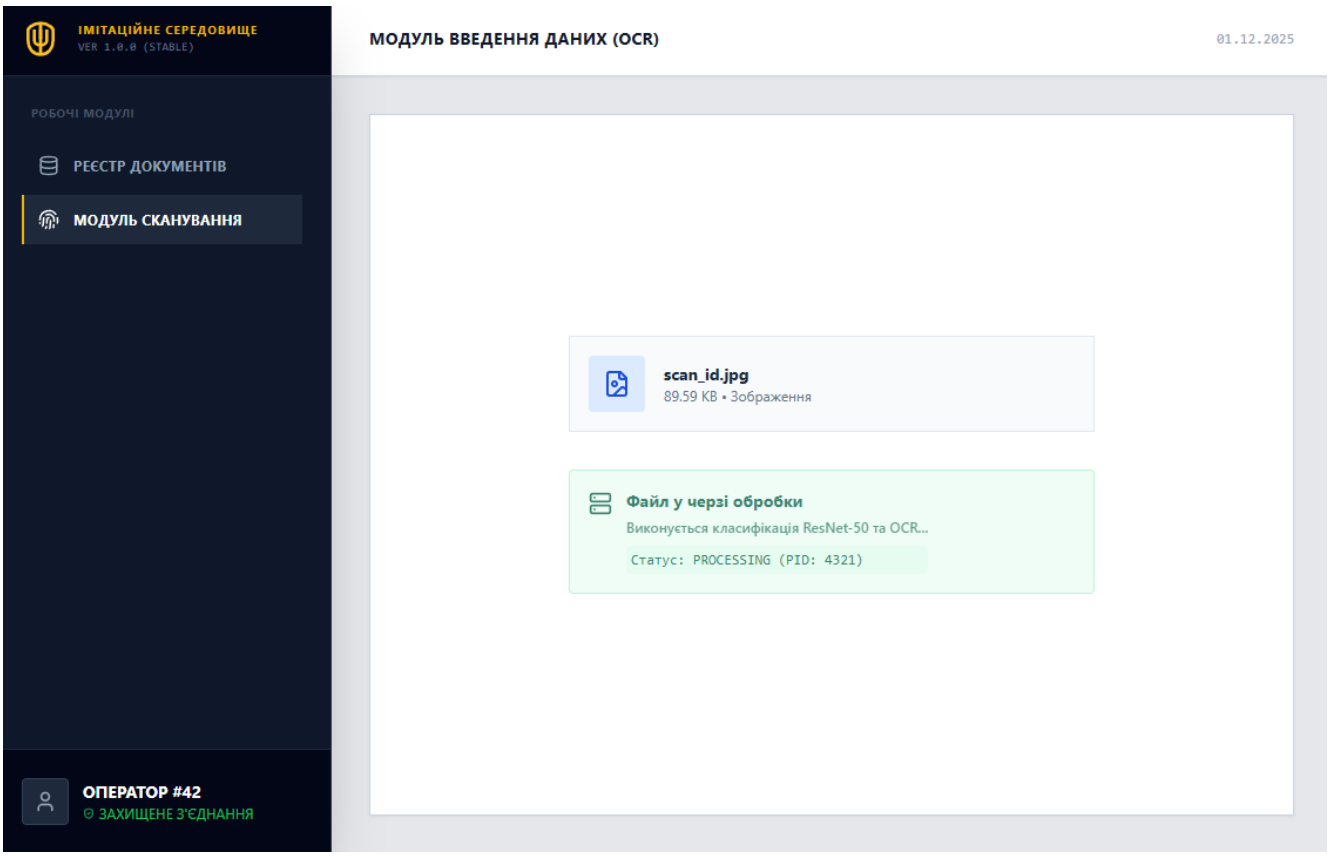


Рисунок 3.12 – Початок процесу сканування.

Паралельно на стороні сервера Python-скрипт, який працює в режимі демона, через механізм WebSocket-слухача миттєво отримує сповіщення про появу нового документа. Як видно з логів термінала (рис. 3.13), скрипт ідентифікує подію, завантажує зображення та запускає алгоритми комп'ютерного зору.

```
PS C:\Users\Admin\Desktop\Diploma\Server> python .\Program.py

Сервер підключено до Державного Реєстру.

Служба обробки активна (APP_ID: docs-ai-a8f3b)...
Отримано файл: scan_id.jpg
Файл успішно оброблено.
```

Рисунок 3.13 – Процес сканування з боку скрипта.

Після завершення процесу сканування, повертаючись у веб інтерфейс буде видно наступне:

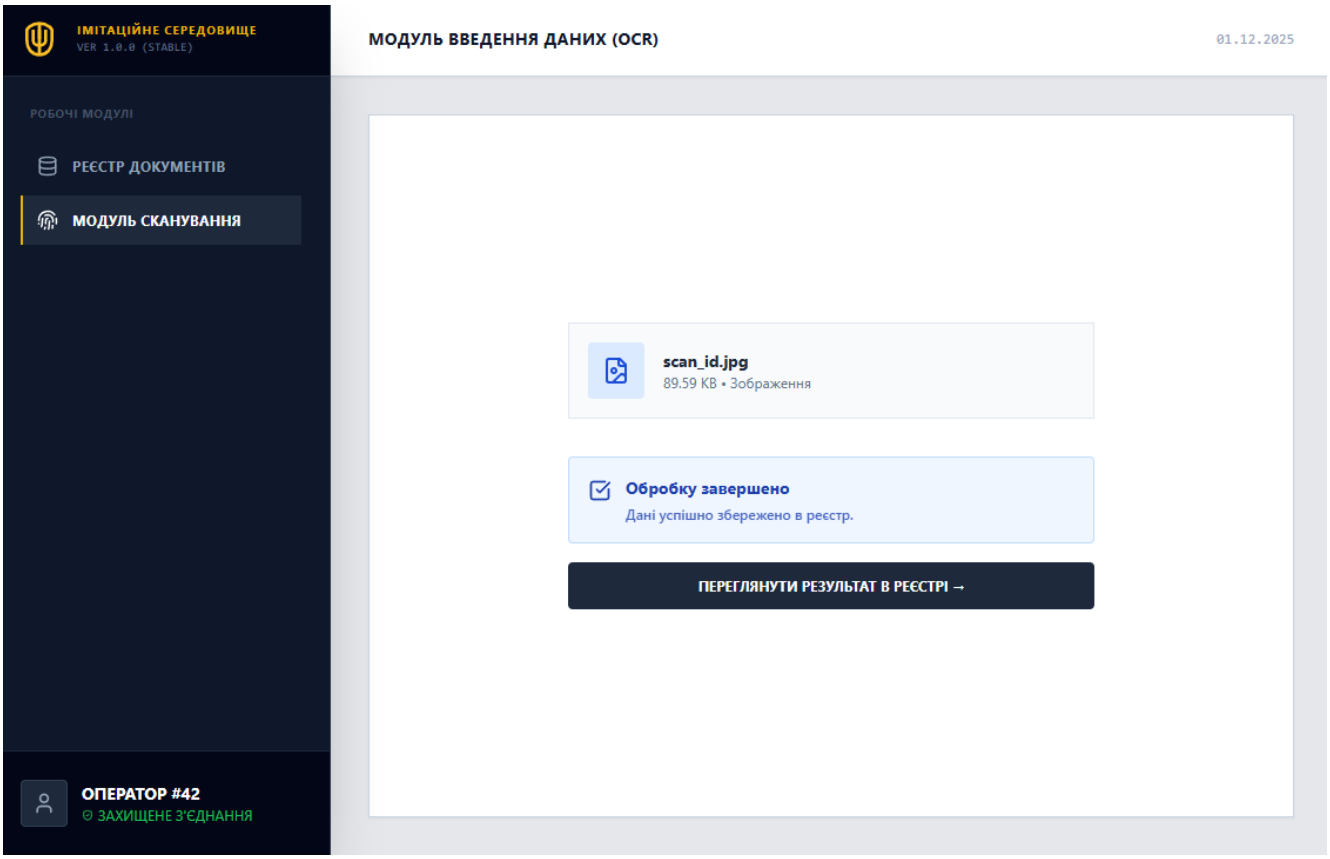


Рисунок 3.14 – Завершення сканування зі сторони веб інтерфейсу

Фінальним етапом тестування є верифікація збережених даних у реєстрі. Перехід до відповідної вкладки демонструє новий запис, що містить розпізнані атрибути: номер документа, ПІБ власника, дату народження та орган видачі. Порівняння отриманих даних з еталонним зразком (рис. 3.10) підтверджує коректність роботи алгоритмів розпізнавання та цілісність передачі.

даних між компонентами системи.

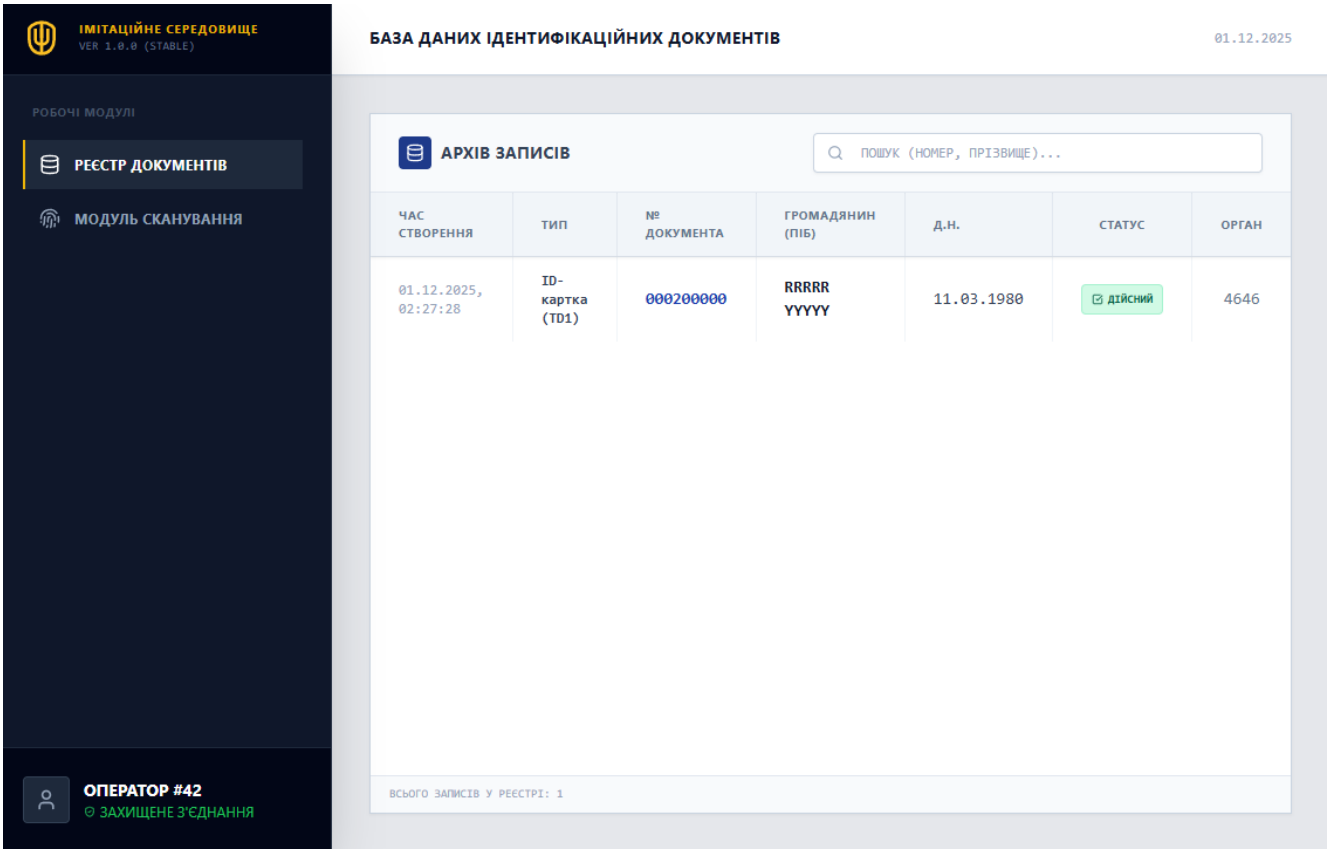


Рисунок 3.15 – Результат сканування

Результати тестування засвідчують працездатність розробленого програмного комплексу, коректність взаємодії розподілених компонентів та здатність системи виконувати повний цикл обробки документів: від завантаження зображення до збереження верифікованих даних у реєстрі.

3.5. Перспективи розвитку та вдосконалення програмного рішення

Розроблене програмне рішення створює надійну основу для автоматизованої обробки та аналізу ідентифікаційних документів. Завдяки поєднанню алгоритмів комп’ютерного зору, оптичного розпізнавання тексту та елементів штучного інтелекту система вже на цьому етапі демонструє здатність працювати з реальними зображеннями документів та забезпечувати коректне виділення й інтерпретацію ключових полів. Разом з тим структура застосунку

передбачає широкі можливості для подальшого нарощування функціональності та оптимізації, що дозволяє розглядати його як гнучку платформу, здатну розвиватися відповідно до нових потреб та технологічних стандартів.

Одним із перспективних напрямів вдосконалення є розширення можливостей модуля класифікації документів. Поточна реалізація вже використовує ResNet-50, що створює міцний фундамент для подальшого донавчання моделі на спеціалізованому наборі зображень українських документів. Додавання такого датасету дозволить моделі точніше розрізняти різні типи паспортів, посвідчень, карток та інших державних документів, а також підвищити стійкість до варіацій у якості знімків. У майбутньому доцільним є перехід до більш сучасних та легких моделей, таких як EfficientNet або MobileNet, що забезпечать високу точність при меншому споживанні ресурсів.

Значний потенціал розвитку має і модуль оптичного розпізнавання тексту. Використання EasyOCR і Tesseract у поточному вигляді вже дозволяє розпізнавати основні дані документа, однак подальше донавчання OCR-моделей на українських бланках може суттєво підвищити їхню точність. Окремим напрямом вдосконалення є впровадження адаптивного виділення зон документа. Замість роботи з фіксованими областями, система може використовувати алгоритми автоматичного аналізу компонування сторінки, що дозволить обробляти документи незалежно від їхнього положення, нахилу, освітлення чи масштабування. Для цього можуть бути застосовані сучасні нейромережеві моделі, зокрема трансформерні архітектури Donut, TrOCR або LayoutLM, які поєднують розпізнавання тексту та аналіз структури документа.

Удосконаленню підлягає також робота з MRZ-зоною. У майбутньому можливе впровадження окремих моделей OCR, спеціально орієнтованих на машинозчитувані рядки, а також механізмів перевірки контрольних сум відповідно до стандарту ICAO 9303. Це не лише підвищить точність зчитування, а й дозволить використовувати дану інформацію як елемент автоматизованої перевірки коректності та цілісності документа.

З точки зору архітектури система також має значні можливості для масштабування. Перехід від однопоточної схеми обробки до мікросервісної інфраструктури дозволить опрацьовувати велику кількість запитів паралельно, забезпечуючи стабільність роботи навіть у пікові періоди навантаження. Використання контейнеризації (Docker), оркестрації (Kubernetes) та механізмів балансування навантаження створить основу для розгортання рішення на серверах державного рівня.

Окремим перспективним напрямом розвитку є впровадження систем журналювання та аудиту. Додавання можливості фіксації всіх операцій, які здійснює система, дозволить підвищити прозорість процесу обробки документів та забезпечити відповідність майбутнього програмного комплексу вимогам інформаційної безпеки та КСЗІ. Разом із цим може бути реалізований механізм контролю доступу та диференціації прав, що створить умови для безпечної взаємодії різних категорій користувачів.

Подальше розширення функціональності системи також може включати впровадження елементів перевірки автентичності документів. Аналіз захисних елементів, голограм, фонового мікротексту та виявлення можливих підробок за допомогою алгоритмів комп’ютерного зору відкриває можливість створення повноцінного інструменту для первинної автоматичної експертизи документа. Додатково система може бути доповнена механізмами Explainable AI, що дозволить інтерпретувати та пояснювати результати аналізу, підвищуючи рівень довіри до прийнятих рішень.

Таким чином, розроблене програмне рішення має значний потенціал для подальшого розвитку, а його архітектура забезпечує можливість поступового впровадження нових модулів та технологій без потреби у кардинальній перебудові системи. Впровадження описаних напрямів модернізації дозволить створити високопродуктивний, масштабований та надійний програмний комплекс, який відповідатиме вимогам державних інформаційних систем та

забезпечуватиме високу точність та швидкість обробки документів у реальних умовах.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було вирішено актуальне науково-прикладне завдання автоматизації процесів обробки неструктурованих даних у державних інформаційних системах шляхом створення програмного комплексу для інтелектуального аналізу ідентифікаційних документів. На основі проведеного аналізу предметної області було встановлено, що традиційні методи оптичного розпізнавання символів мають суттєві обмеження при роботі зі складними документами, що зумовило необхідність застосування гібридного підходу, який поєднує класичні алгоритми комп'ютерного зору та сучасні методи глибокого навчання. Реалізація цього підходу втілилася у розробці сервіс-орієнтованої архітектури системи, де функції збору даних, їх інтелектуальної обробки та зберігання рознесені між незалежними компонентами, що забезпечує високу масштабованість та надійність програмного рішення.

Ключовим результатом роботи стало створення прототипу інформаційно-аналітичної системи, що складається з клієнтського веб-інтерфейсу, серверного модуля обробки та бази даних реального часу. Розроблений клієнтський додаток на базі бібліотеки React.js забезпечує взаємодію з оператором, реалізуючи попередню валідацію даних та візуалізацію процесів обробки, тоді як серверна частина на мові Python виконує ресурсоємні завдання класифікації типів документів за допомогою згорткової нейронної мережі ResNet-50 та вилучення текстових атрибутів із застосуванням OCR-технологій. Використання хмарної платформи Google Firebase як комунікаційного ядра дозволило реалізувати асинхронну модель обміну даними, що гарантує стабільну роботу системи навіть в умовах нестабільного мережевого з'єднання та високих навантажень.

Проведене тестування системи на зразку ID карток підтвердили правильність обраних архітектурних та алгоритмічних рішень. Система

продемонструвала високу точність класифікації документів та ефективність алгоритмів парсингу машинозчитуваних зон, забезпечуючи коректне формування структурованих записів у базі даних. Практична цінність роботи полягає у створенні інструментарію, впровадження якого в діяльність центрів надання адміністративних послуг та інших державних установ дозволить суттєво зменшити частку ручної праці, мінімізувати ризики помилок при введенні персональних даних та прискорити процес обслуговування громадян, що є важливим кроком на шляху до повної цифрової трансформації державного сектору.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Переяславська, С., Шевченко, В., & Смагіна, О. (2021). АНАЛІЗ ПІДХОДІВ ДО РОЗПІЗНАВАННЯ ТЕКСТОВОЇ ІНФОРМАЦІЇ У ТЕХНОЛОГІЇ OCR. InterConf. Режим доступу: https://ojs.ukrlogos.in.ua/index.php/interconf/article/view/9924 (дата звернення: 10.10.2025).

2. Карпиков, Л. В. Ларченко // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління : тез. доп. дванадцатої міжнародної науково-технічної конференції, 27–28 квітня 2022 р. – Т. 1. – Баку–Харків–Жиліна, 2022. – С. 63. Режим доступу: https://openarchive.nure.ua/handle/document/26261 (дата звернення: 10.10.2025).

3. Патока, В., & Гавриленко , О. (2024). ПРОВІДНІ OCR МОДЕЛІ ДЛЯ РОЗПІЗНАВАННЯ ТЕКСТУ НА ЗОБРАЖЕННЯХ. UNIVERSUM, (9), 356–364. Режим доступу: https://archive.liga.science/index.php/universum/article/view/1125 (дата звернення: 11.10.2025).

4. OCR using CRNN: A Deep Learning Approach for Text Recognition / A. Yadav та ін. *2023 4th International Conference for Emerging Technology (INCET)*, м. Belgaum, India, 26–28 трав. 2023 р. 2023. URL: https://doi.org/10.1109/incet57972.2023.10170436 (дата звернення: 12.10.2025).

5. Advancing Optical Character Recognition for Handwritten Text: Enhancing Efficiency and Streamlining Document Management / B. V. V та ін. *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, м. Delhi, India, 6–8 лип. 2023 р. 2023. URL: https://doi.org/10.1109/icccnt56998.2023.10307143 (дата звернення: 12.10.2025).

6. ТЕНДЕНЦІЇ РОЗВИТКУ ШТУЧНОГО ІНТЕЛЕКТУ В УКРАЇНІ. (2020). Безпека інформаційних систем і технологій, 1(3-4), 20-26.

https://doi.org/10.17721/ISTS.2020.2.19-26 (дата звернення: 25.12.2025)

7. Миколайчук Р., Миколайчук А. Використання технологій штучного інтелекту для автоматизації процесу обробки документів. *Сучасні інформаційні технології у сфері безпеки та оборони*. 2023. Т. 50, № 2. С. 111–117. URL: https://doi.org/10.33099/2311-7249/2024-50-2-111-117 (дата звернення: 13.10.2025).

8. Кузнєцов О., Заїка В. Використання штучного інтелекту у створенні звіту бібліотеки. *Open science and innovation*. 2025. Т. 1, № 1. URL: https://doi.org/10.62405/osi.2025.01.04 (дата звернення: 14.10.2025).

9. Про захист персональних даних : Закон України від 01.06.2010 № 2297-VI : станом на 14 черв. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/2297-17#Text (дата звернення: 14.10.2025).

10. Про хмарні послуги : Закон України від 17.02.2022 № 2075-IX : станом на 28 черв. 2024 р. URL: https://zakon.rada.gov.ua/laws/show/2075-20#Text (дата звернення: 14.10.2025).

11. Про схвалення Концепції розвитку штучного інтелекту в Україні : Розпорядження Кабінету Міністрів України від 02.12.2020 № 1556-р. URL: https://zakon.rada.gov.ua/laws/show/1556-2020-%D1%80 (дата звернення: 15.10.2025).

12. Книш Д., Котенко Н. Сучасні підходи та виклики у поєднанні технологій ocr та nlp для автоматизованого аналізу друкованого тексту. *Information technology: computer science, software engineering and cyber security*. 2025. № 1. С. 98–104. URL: https://doi.org/10.32782/it/2025-1-14 (дата звернення: 15.10.2025).

13. Про захист інформації в інформаційно-телекомунікаційних системах : Закон України від 05.07.1994 № 80/94-ВР : станом на 20 квіт. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/80/94-вр#Text (дата звернення: 16.10.2025).

14. Про публічні електронні реєстри : Закон України від 18.11.2021 № 1907-IX : станом на 18 листоп. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/1907-20#Text (дата звернення: 17.10.2025).

15. Деякі питання електронної взаємодії державних електронних інформаційних ресурсів : Постанова Каб. Міністрів України від 08.09.2016 № 606 : станом на 3 жовт. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/606-2016-п#Text (дата звернення: 18.10.2025).

16. Кравчук О. Штучний інтелект у програмуванні: як ші змінює підхід до розробки та автоматизації коду. *Herald of Khmelnytskyi National University. Technical sciences*. 2024. Т. 345, № 6(2). С. 238–242. URL: https://doi.org/10.31891/2307-5732-2024-345-6-36 (дата звернення: 18.10.2025).

17. Про електронні довірчі послуги : Закон України від 05.10.2017 № 2155-VIII : станом на 18 груд. 2024 р. URL: https://zakon.rada.gov.ua/laws/show/2155-19#Text (дата звернення: 19.10.2025).

18. Арделян І., Плецан Х. Методи застосування штучного інтелекту для аналізу соціально-економічних даних у публічному управлінні. *Public management and policy*. 2025. № 5(9). URL: https://doi.org/10.70651/3041-2498/2025.5.05 (дата звернення: 19.10.2025).

19. Питання Єдиного державного веб-порталу електронних послуг та Єдиного державного порталу адміністративних послуг : Постанова Каб. Міністрів України від 04.12.2019 № 1137 : станом на 7 листоп. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/1137-2019-п#Text (дата звернення: 20.10.2025).

20. Yushko A. V., Simak A. Y. Information intellectual system of analysis of the scientific and scientific-pedagogical activities of the academic team. *Optoelectronic Information-Power Technologies*. 2024. Т. 47, № 1. С. 7–16. URL: https://doi.org/10.31649/1681-7893-2024-47-1-7-16 (дата звернення: 21.10.2025).

21. Полянська А. С., Дюк О. М. Інтелектуальний аналіз даних у дослідженні корпоративної культури підприємства. Економічний вісник Національного технічного університету України «Київський політехнічний інститут». 2021. № 19. URL: https://doi.org/10.20535/2307-5651.19.2021.240691 (дата звернення:

21.10.2025).

22. Ohirko I., Shapenko L., Zolotar O. Artificial intelligence and data analytics, machine learning and law. Information and law. 2025. № 3(54). С. 39–52. URL: https://doi.org/10.37750/2616-6798.2025.3(54).340450 (дата звернення: 21.10.2025).

23. Gabani I. I. International information and artificial intelligence in the law enforcement activities of the state. Uzhhorod national university herald. series: law. 2025. Т. 5, № 90. С. 112–116. URL: https://doi.org/10.24144/2307-3322.2025.90.5.14 (дата звернення: 22.10.2025).

24. Segeda I. Prospects of databases development in the digital economy. Intellect XXI. 2021. № 6, 2021. URL: https://doi.org/10.32782/2415-8801/2021-6.10 (дата звернення: 22.10.2025).

25. Artificial intelligence adoption in public administration: an overview of top-cited articles and practical applications / M. Babšek та ін. Ai. 2025. Т. 6, № 3. С. 44. URL: https://doi.org/10.3390/ai6030044 (дата звернення: 23.10.2025).

26. Navigating governmental choices: a comprehensive review of artificial intelligence’s impact on decision-making / G. Caiza та ін. Informatics. 2024. Т. 11, № 3. С. 64. URL: https://doi.org/10.3390/informatics11030064 (дата звернення: 24.10.2025).

27. Chen Y.-C., Ahn M., Wang Y.-F. Artificial intelligence and public values: value impacts and governance in the public sector. Sustainability. 2023. Т. 15, № 6. С. 4796. URL: https://doi.org/10.3390/su15064796 (дата звернення: 25.10.2025).

28. Genaro-Moya D., López-Hernández A. M., Godz M. Artificial intelligence and public sector auditing: challenges and opportunities for supreme audit institutions. World. 2025. Т. 6, № 2. С. 78. URL: https://doi.org/10.3390/world6020078 (дата звернення: 26.10.2025).

29. Про схвалення Концепції розвитку штучного інтелекту в Україні : Розпорядж. Каб. Міністрів України від 02.12.2020 № 1556-р : станом на 29 груд. 2021 р. URL: https://zakon.rada.gov.ua/laws/show/1556-2020-p#Text (дата звернення: 27.10.2025).

30. Alon-Barkat S., Busuioc M. Public administration meets artificial

intelligence: towards a meaningful behavioral research agenda on algorithmic decision-making in government. Journal of behavioral public administration. 2024. T. 7. URL: https://doi.org/10.30636/jbpa.71.261 (дата звернення: 27.10.2025).

31. Young M. M., Bullock J. B., Lecy J. D. Artificial discretion as a tool of governance: a framework for understanding the impact of artificial intelligence on public administration. Perspectives on public management and governance. 2019. URL: https://doi.org/10.1093/ppmgov/gvz014 (дата звернення: 28.10.2025).

32. Criado J. I., O.de Zarate-Alcarazo L. Technological frames, CIOs, and Artificial Intelligence in public administration: A socio-cognitive exploratory study in Spanish local governments. Government information quarterly. 2022. C. 101688. URL: https://doi.org/10.1016/j.giq.2022.101688 (дата звернення: 28.10.2025).

33. Artificial intelligence and the local government: a five-decade scientometric analysis on the evolution, state-of-the-art, and emerging trends / T. Yigitcanlar та ін. Cities. 2024. T. 152. C. 105151. URL: https://doi.org/10.1016/j.cities.2024.105151 (дата звернення: 29.10.2025).

34. Dubniak M. Open data and explanatory artificial intelligence: legal perspectives. Information and law. 2024. № 2(49). C. 102–117. URL: https://doi.org/10.37750/2616-6798.2024.2(49).306147 (дата звернення: 29.10.2025).

35. Franchuk V. Artificial intelligence in the public administration system: potential and risks. Electronic scientific publication "public administration and national security". 2022. № 5(58). URL: https://doi.org/10.25313/2617-572x-2025-5-11028 (дата звернення: 30.10.2025).

36. Квітка С., Новіченко Н., Бардах О. Штучний інтелект у муніципальному управлінні: вектори розвитку. Public administration aspects. 2021. Т. 9, № 4. С. 85–94. URL: https://doi.org/10.15421/152140 (дата звернення: 30.10.2025).

37. Wirtz B. W., Weyerer J. C., Sturm B. J. The dark sides of artificial intelligence: an integrated AI governance framework for public administration. International journal of public administration. 2020. Т. 43, № 9. С. 818–829. URL: https://doi.org/10.1080/01900692.2020.1749851 (дата звернення: 30.10.2025).

38. Кузнєцов Карпенко Ю. Зарубіжний досвід визначення концептуальних

та інституціональних засад розвитку штучного інтелекту в публічному управлінні. Теоретичні та прикладні питання державотворення. 2019. № 24. С. 34–42. URL: https://doi.org/10.35432/tisb.v0i24.183335 (дата звернення: 31.10.2025).

39. Lohachova Y. O., Yesina M. V., Holubnychyi D. Y. Research and analysis of international standards and regulatory requirements for artificial intelligence se-curity, development of a security model for Ukraine. Radiotekhnika. 2025. № 221. С. 14–22. URL: https://doi.org/10.30837/rt.2025.2.221.02 (дата звернення: 31.10.2025).

40. Bochkova I. I., Vrublevska-Misiuna K. M. Data access and confidentiality in the context of artificial intelligence. Uzhhorod national university herald. series: law. 2025. Т. 3, № 90. С. 139–146. URL: https://doi.org/10.24144/2307-3322.2025.90.3.19 (дата звернення: 01.12.2025).

41. Mucci T. What is image recognition? IBM. URL: https://www.ibm.com/think/topics/image-recognition#691946472 (дата звернення: 01.12.2025).

42. Tripathi P. A journey through history: the evolution of OCR technology. Docsumo. URL: https://www.docsumo.com/blog/optical-character-recognition-history (дата звернення: 02.12.2025).

43. Nicolas. ResNet-50: important architecture for modern AI. Innovatiana. URL: https://www.innovatiana.com/en/post/discover-resnet-50 (дата звернення: 02.12.2025).

44. Wirtz B. W., Weyerer J. C., Kehl I. Governance of artificial intelligence: a risk and guideline-based integrative framework. Government information quarterly. 2022. С. 101685. URL: https://doi.org/10.1016/j.giq.2022.101685 (дата звернення: 02.12.2025).

45. Юлдашев О. Х., Джунь В. О. Використання штучного інтелекту у веденні господарської діяльності. Kyiv law journal. 2025. № 2. С. 159–164. URL: https://doi.org/10.32782/klj/2025.2.20 (дата звернення: 03.12.2025).

46. Development of knowledge bases with augmented generation using basic models of artificial intelligence. Connectivity. 2024. Т. 171, № 5. URL: https://doi.org/10.31673/2412-9070.2024.050247 (дата звернення: 03.12.2025).

47. Artificial intelligence adoption in public administration: an overview of top-cited articles and practical applications / M. Babšek та ін. Ai. 2025. Т. 6, № 3. С. 44. URL: https://doi.org/10.3390/ai6030044 (дата звернення: 04.12.2025).

48. Kotenko D., Khlaponin Y. Artificial intelligence in cyber attack detection and prevention systems: prospects and challenges. Pidvodni tehnologii. 2024. Т. 1, № 14. С. 48–55. URL: https://doi.org/10.32347/uwt.2024.14.1203 (дата звернення: 04.12.2025).

49. Mergel I., Edelmann N., Haug N. Defining digital transformation: results from expert interviews. Government information quarterly. 2019. Т. 36, № 4. С. 101385. URL: https://doi.org/10.1016/j.giq.2019.06.002 (дата звернення: 05.12.2025).

50. Alon-Barkat S., Busuioc M. Human-AI interactions in public sector decision-making: ‘automation bias’ and ‘selective adherence’ to algorithmic advice. *Journal of public administration research and theory*. 2022. URL: https://doi.org/10.1093/jopart/muac007 (дата звернення: 06.12.2025).

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра комп'ютерної інженерії

ПРЕЗЕНТАЦІЯ ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ

на тему

«СИСТЕМА ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ІНФОРМАЦІЇ У ДЕРЖАВНИХ БАЗАХ ДАНИХ НА БАЗІ ШТУЧНОГО ІНТЕЛЕКТУ»

Виконав:
Студент 6 курсу групи КСДМ-61
Приймак Євгеній Олександрович
Науковий керівник:
Волохін Віталій Васильович

Київ 2025

АКТУАЛЬНІСТЬ

2

Актуальність теми зумовлена необхідністю автоматизації обробки значних масивів неструктурованих даних, що надходять до державних установ. Використання методів глибокого навчання дозволяє мінімізувати вплив людського фактору, прискорити верифікацію громадян та інтегрувати паперові архіви в цифрову систему держави.

МЕТА, ОБ’ЄКТ, ПРЕДМЕТ

Мета дослідження: Аналіз можливості підвищення ефективності та надійності процесів обробки даних у державних інформаційних системах шляхом розробки прототипу системи інтелектуального аналізу документів.

Об’єкт дослідження: Процес автоматизованої обробки та семантичного аналізу графічних образів документів.

Предмет дослідження: Моделі, методи та алгоритми комп'ютерного зору для класифікації та вилучення інформації.

ЗАВДАННЯ ДОСЛІДЖЕННЯ 4

- провести системний аналіз існуючих методів OCR та виявити їх обмеження при роботі з документами;

- дослідити архітектури нейронних мереж для задач класифікації зображень;

- створити алгоритм попередньої обробки зображень для усунення дефектів сканування;

- розробити приклад сервіс-орієнтованої архітектури системи, що забезпечує повний цикл обробки даних.

ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНИХ ЗАВДАНЬ В ТЕОРЕТИЧНІЙ ЧАСТИНІ РОБОТИ БУЛО: 5

- визначено потенціал методів розпізнавання зображень за допомогою глибоких нейронних мереж;

- проаналізовано проблематику обробки «брудних» даних у державних реєстрах;

- обґрунтовано вибір архітектури ResNet-50 для класифікації типів документів;

- сформульовано вимоги до інформаційної безпеки при розгортанні подібної системи у державних органах.

6 РЕАЛІЗАЦІЯ ПРИКЛАДУ АЛГОРИТМУ ОБРОБКИ ДАНИХ

Для реалізації прототипу системи обрано сервіс-орієнтовану архітектуру, що складається з трьох незалежних модулів:

Сервер обробки	• Мова Python. • Основні бібліотеки: PyTorch, OpenCV та EasyOCR. • Відправляє оброблені дані на базу даних в реальному часі.
Інтерфейс	• Веб-додаток React.js. • Відправляє зображення в базу даних в реальному часі.
База даних	• Хмарна платформа Google Firebase. • Слугує мостом між веб-додатком і програмою обробки.

7 РЕАЛІЗАЦІЯ ПРИКЛАДУ АЛГОРИТМУ ОБРОБКИ ДАНИХ

Загальний принцип роботи алгоритму обробки заключається в наступному:

- **Крок 1: Попередня обробка.** Використання OpenCV для бінаризації, видалення шумів та вирівнювання перспективи зображення.

- **Крок 2: Класифікація.** Нейромережа ResNet-50 визначає тип документа

- **Крок 3: Сегментація та OCR.** Локалізація машинозчитуваної зони (MRZ) та розпізнавання тексту гібридним методом.

- **Крок 4: Структуризація.** Структурування розпізнаного тексту в JSON-формат.

РЕАЛІЗАЦІЯ ПРИКЛАДУ АЛГОРИТМУ ОБРОБКИ ДАНИХ

8

Основні елементи роботи коду наведені на наступних рисунках :

```python
class DocumentClassifier:
    def __init__(self):
        self.labels = ['passport_book', 'id_card', 'driver_license']
        self.device = torch.device("cpu")
        self.model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
        self.model.eval()

        self.preprocess = transforms.Compose([
            transforms.Resize(256),
            transforms.CenterCrop(224),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
        ])

    def classify(self, cv2_image):
        pil_img = Image.fromarray(cv2.cvtColor(cv2_image, cv2.COLOR_BGR2RGB))
        input_tensor = self.preprocess(pil_img).unsqueeze(0)

        with torch.no_grad():
            output = self.model(input_tensor)
```

Рисунок 8.1 – Архітектура нейромережевого класифікатора.

```python
class AdvancedOCR:
    def __init__(self):
        self.reader = easyocr.Reader(['en', 'uk'], gpu=False)

    def get_authority(self, image):
        h, w = image.shape[:2]
        crop = image[int(h*0.18):int(h*0.28), int(w*0.45):int(w*0.7)]
        gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
        _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
        result = self.reader.readtext(binary, allowlist='0123456789', detail=0)
        return "".join(result) if result else "4646"

    def get_date_of_issue(self, image):
        h, w = image.shape[:2]
        crop = image[int(h*0.18):int(h*0.28), int(w*0.05):int(w*0.35)]
        gray = cv2.cvtColor(crop, cv2.COLOR_BGR2GRAY)
        _, binary = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
        result = self.reader.readtext(binary, allowlist='0123456789. ', detail=0)

        if result:
            raw_date = " ".join(result)
            normalized_date = raw_date.replace(" ", ".").replace("..", ".")
            return normalized_date
        return "N/A"
```

Рисунок 8.2 – Реалізація гібридного механізму оптичного розпізнавання.

РЕАЛІЗАЦІЯ ПРИКЛАДУ АЛГОРИТМУ ОБРОБКИ ДАНИХ

9

```python
class MRZParser:
    @staticmethod
    def parse(text):
        lines = [l.strip() for l in text.split('\n') if len(l) > 20 and '<' in l]
        if len(lines) < 3: return None

        l1, l2, l3 = lines[-3:]
        doc_num = l1[5:14].replace('<', '')
        birth = l2[0:6]
        names = l3.split('<<')
        surname = names[0].replace('<', '')
        name = names[1].replace('<', ' ') if len(names) > 1 else ""

        try:
            b_date = f"{birth[4:6]}.{birth[2:4]}.19{birth[0:2]}"
        except:
            b_date = birth

        return {
            "doc_number": doc_num,
            "surname": surname,
            "names": name,
            "birth_date": b_date
        }
```

Рисунок 9.1 – Реалізація парсингу MRZ частини ID картки.

```python
def process_base64_image(base64_str):
    try:
        if "," in base64_str: base64_str = base64_str.split(",")[1]
        nparr = np.frombuffer(base64.b64decode(base64_str), np.uint8)
        img = cv2.imdecode(nparr, cv2.IMREAD_COLOR)

        doc_type, conf = ai_classifier.classify(img)

        authority = ai_ocr.get_authority(img)

        issue_date = ai_ocr.get_date_of_issue(img)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        text = pytesseract.image_to_string(gray, config='--oem 3 --psm 6')

        mrz_data = MRZParser.parse(text)

        if mrz_data:
            return {
                "status": "VALID",
                "doc_type": doc_type,
                "authority": authority,
                "issue_date": issue_date,
                **mrz_data
            }
        else:
            return {"status": "WARNING", "doc_type": doc_type, "error": "MRZ не чітка"}

    except Exception as e:
        print(f"Помилка обробки: {e}")
        return {"status": "ERROR"}
```

Рисунок 9.2 – Структурування даних.

РЕАЛІЗАЦІЯ ПРИКЛАДУ АЛГОРИТМУ ОБРОБКИ ДАНИХ

10

Для відправки зображень для опрацювання використовується веб додаток:

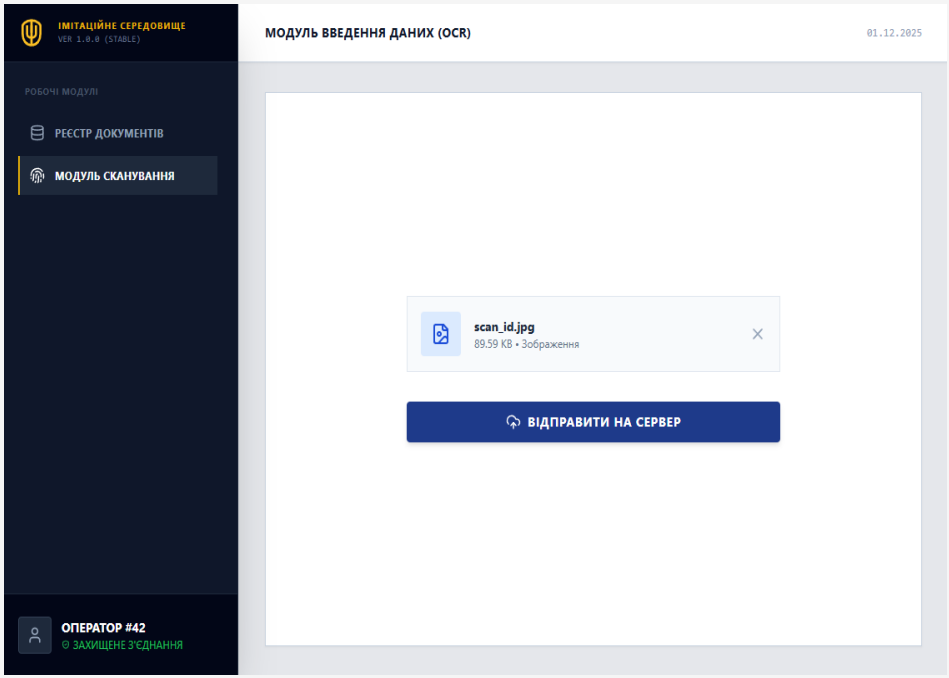


Рисунок 10.1 – Завантаження зразку у веб додаток.

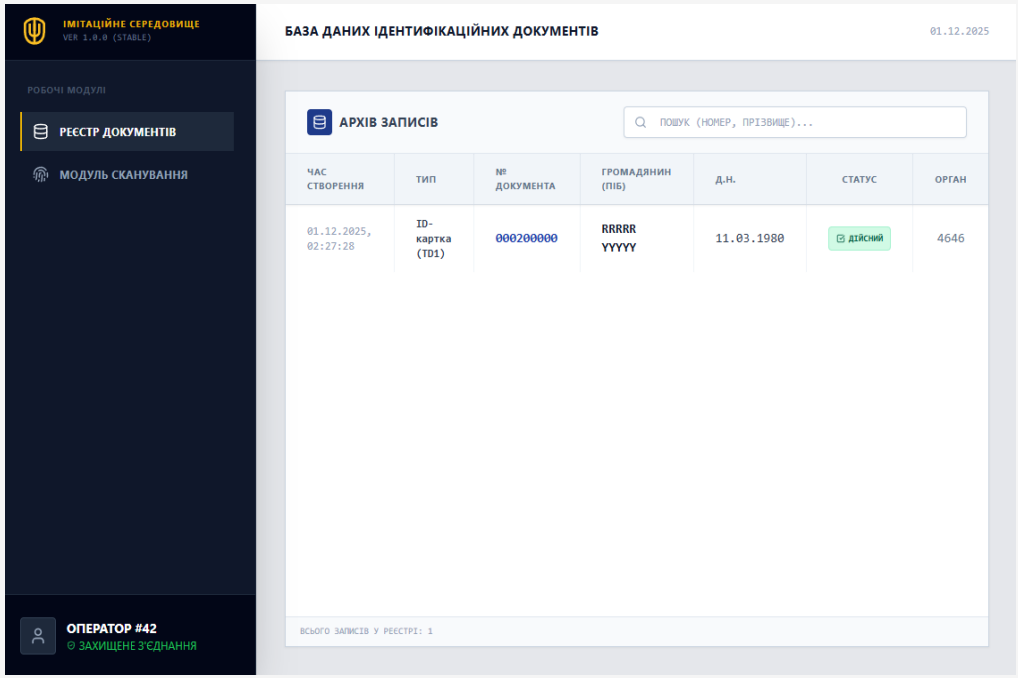


Рисунок 10.2 – Результат сканування

ВИСНОВКИ

11

Для досягнення результату в роботі були виконані наступні завдання:

- досліджено систему аналізу даних за допомогою OCR;

- досліджені правові засади можливості автоматизації процесу обробки даних в державних установах;

- розглянуті приклади використання штучного інтелекту для обробки візуальних даних;

- визначений потенціал інтеграції штучного інтелекту в державні установи для використання в різних сферах діяльності;

- створено прототип програмного комплексу, здатного збирати та обробляти інформацію з паспорту громадянина України.

РЕЗУЛЬТАТИ ВПРОВАДЖЕННЯ

12

1. Волохін В. В., Приймак Є. О. МЕТОДИ ПОПЕРЕДНЬОЇ ОБРОБКИ ТА АДАПТИВНОЇ БІНАРИЗАЦІЇ ЗАШУМЛЕНИХ ЗОБРАЖЕНЬ В СИСТЕМАХ ЕЛЕКТРОННОГО УРЯДУВАННЯ «Проблеми комп’ютерної інженерії» : VI науково-практична конференція, м. Київ, 3 грудня 2025р.
2. Приймак Є. О., Зайцев Є. О., Лемешко А. В., Антоненко А. В. Стаття «ДОСЛІДЖЕННЯ МОЖЛИВОСТЕЙ ОПТИМІЗАЦІЇ ПРОЦЕСУ ОБРОБКИ ДАНИХ В ДЕРЖАВНИХ ІНФОРМАЦІЙНИХ СИСТЕМАХ ІЗ ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ». Журнал «ITSynergy» : номер 1, м. Київ, 2024 р. С. 6–15.