

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка
до дипломного проєкту (роботи)
бакалавра
(ступінь вищої освіти (освітній ступінь))

на тему: «РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ДЛЯ РОЗПІЗНАВАННЯ
ДОКУМЕНТІВ З ДОПОМОГОЮ БІБЛІОТЕКИ EASYOCR»

Виконав: студент 4 курсу, групи КНТ-511
Спеціальності 123 Комп'ютерна інженерія
Освітня програма (спеціалізація)
Комп'ютерна інженерія
(код і назва спеціальності)

ЛЯХОВСЬКИЙ М.В.
(прізвище та ініціали)
Керівник СКРУПСЬКИЙ С.Ю.
(прізвище та ініціали)
Рецензент ЄФИМЕНКО М.В.
(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій

Кафедра «Комп'ютерні системи та мережі»

Ступінь вищої освіти (освітній ступінь) бакалаврський

Спеціальність	123	Комп'ютерна інженерія
---------------	-----	-----------------------

(код і назва)

Освітня програма (спеціалізація) Комп'ютерна інженерія

(назва)

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“ ” 2025 року

ЗАВДАННЯ НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

ЛЯХОВСЬКИЙ Микита Віталійович

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) «Розробка комп'ютерної системи для розпізнавання документів з допомогою бібліотеки EasyOCR»

керівник проекту (роботи) СКРУПСЬКИЙ С.Ю., к. т. н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “08” квітня 2025 року № 151

2. Строк подання студентом проекту (роботи) 01.06.2025 року

3. Вихідні дані до проекту (роботи) нейромережі, мова програмування python, бібліотека EasyOCR

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) _____

1) Технології та програмні засоби для автоматичного розпізнавання тексту та внесення даних

2) Інструменти для реалізації комп'ютерної системи

3 Моделювання і розробка програмного забезпечення системи

4) Реалізація системи

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Слайди презентації

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	СКРУПСЬКИЙ С.Ю. , к.т.н., доцент		
Нормоконтроль	ПОЛЬСЬКА О.В., ст.викладач		

7. Дата видачі завдання 15.04.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Огляд літератури	20.04.2025 р.	
2	Аналіз технічного завдання	25.04.2025 р.	
3	Огляд існуючих рішень	30.04.2025р.	
4	Проектування системи	10.05.2025 р.	
5	Розробка системи розпізнавання	15.05.2025 р.	
6	Тестування системи	20.05.2025 р.	
7	Оформлення ПЗ	26.05.2025 р.	

Студент _____ **Микита ЛЯХОВСЬКИЙ**
(підпис) (ініціали та прізвище)

Керівник проекту (роботи) _____ **Степан СКРУПСЬКИЙ**
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:

ПЗ: 61 с., 24 рис., 1 табл., 4 лістинг., 14 джерел.

PYTHON, НЕЙРОННА МЕРЕЖ, РОЗПІЗНАВАННЯ ТЕКСТУ, ШТУЧНИЙ ІНТЕЛЕКТ

Об'єктом дослідження є процеси розпізнавання документів та введення даних, а предметом дослідження – технології та програмні засоби, що забезпечують процеси розпізнавання тексту та введення даних. Метою роботи є розробка нейромережевої системи, яка підвищує ефективність процесів розпізнавання документів та автоматичного введення даних.

Пояснювальна записка дипломної роботи включає вступ, чотири основних розділи, висновки та додатки.

У першому розділі здійснено аналіз існуючих технологій та програмних засобів, що використовуються для розпізнавання тексту та автоматичного введення даних. Другий розділ присвячений розгляду моделей, методів та інформаційних технологій, які можуть бути використані для вирішення поставленої задачі. У третьому розділі проведено моделювання та проектування нейромережевої системи, яка сприяє підвищенню ефективності процесів розпізнавання документів та введення даних. Четвертий розділ містить програмну реалізацію запропонованої системи та розробку необхідної документації.

У результаті виконаної роботи зроблено висновки щодо доцільності та ефективності застосування розробленої нейромережевої системи для розпізнавання документів та автоматичного введення даних.

ABSTRACT

Explanatory note to the master's work: 61 p., 24 figures, 1 tabl., 4 listing, 14 sources.

PYTHON, NEURAL NETWORK, TEXT RECOGNITION, PIECE INTELLIGENCE

The object of research is the processes of document recognition and data entry, and the subject of research is technologies and software tools that support the processes of text recognition and data entry. The aim of the work is to develop a neural network system that increases the efficiency of document recognition and automatic data entry processes.

The thesis explanatory note includes an introduction, four main chapters, conclusions and appendices.

The first chapter analyses existing technologies and software tools used for text recognition and automatic data entry. The second section is devoted to the consideration of models, methods and information technologies that can be used to solve the task. The third section describes the modelling and design of a neural network system that helps to improve the efficiency of document recognition and data entry processes. The fourth section contains the software implementation of the proposed system and the development of the necessary documentation.

As a result of the work performed, conclusions have been drawn regarding the feasibility and effectiveness of using the developed neural network system for document recognition and automatic data entry.

ЗМІСТ

Вступ.....	8
1 Технології та програмні засоби для автоматичного розпізнавання тексту та внесення даних.....	10
1.1 Аналіз якості роботи.....	10
1.2 Огляд аналогів для вирішення задач.....	12
1.2.1 Програмне забезпечення Azoft.....	12
1.2.2 Програмне забезпечення компанії EasyOCR2	16
1.3 Постановка завдання.....	18
1.4 Висновки до розділу.....	19
2 Інструменти для реалізації комп'ютерної системи	19
2.1 Варіанти вирішення задачі	19
2.2 Засоби розробки системи	24
2.3 Висновки до розділу.....	31
3 Моделювання і розробка програмного забезпечення системи	32
3.1 Опис та використання бібліотеки Tesseract.....	32
3.2 Порівняння пакетів програм для реалізації системи.....	37
3.3 Бібліотека EasyOCR.....	41
3.3 Висновки до розділу.....	50
4 Реалізація системи	51
4.1 Програмне забезпечення EasyOCR.....	51
4.2 Програмна реалізація системи.....	55
4.3 Висновки до розділу.....	58
Висновки.....	59
Перелік джерел посилання	60

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ASCII	American Standard Code for Information Interchange, Американський стандартний код для інформаційного обміну
CNN	Convolutional Neural Networks , Згорткові нейронні мережі
CLSTM	Long Short-Term Memory, Реалізація моделі рекурентної нейронної мережі LSTM на C++
GPU	Graphics Processing Unit , Графічний процесор
FPGA	Field-Programmable Gate Array, Програмована користувачем вентильна матриця
LSTM	Long Short-Term Memory , Мова специфікації графів змінних
OCR	Optical Character Recognition , Оптичне розпізнавання символів
OpenCV	Open Source Computer Vision Library, Бібліотека комп'ютерного зору з відкритим кодом
RNN	Recurrent Neural Networks, Рекурентні нейронні мережі
VGSL	Variable Graph Specification Language, Мова специфікації графів змінних

ВСТУП

Починаючи з XX століття, сучасні технології все більше інтегруються у повсякденне життя. Автоматизація охоплює різні сфери та галузі виробництва, спрощуючи та прискорюючи багато процесів.

Однак, оскільки такі напрями, як штучний інтелект, автоматизація та робототехніка, ще відносно молоді, вони постійно розвиваються та вдосконалюються.

Однією з актуальних сфер для застосування сучасних технологій є державні соціальні установи, зокрема паспортні столи, опікунські ради, пенсійні фонди, центри зайнятості, податкові інспекції, відділи поліції, соціальні фонди, служби оплати комунальних послуг тощо. На даний момент у цих установах існує значний недолік – тривалі та затягнуті процеси, зумовлені необхідністю багаторазового переписування імен та фрагментів тексту вручну.

Впровадження у держустановах обладнання для автоматичного зчитування тексту з паперових або фізичних документів та його перенесення у цифровий формат значно прискорило б рутинні операції. Це дозволило б спростити роботу завдяки простому використанню команд копіювання та вставлення тексту (Ctrl+C та Ctrl+V).

З огляду на актуальність цього питання, об'єктом дослідження є процеси розпізнавання документів та введення даних. Відповідно, предметом проектування виступають технології та програмне забезпечення для розпізнавання тексту та автоматизованого введення даних з використанням OCR (оптичного розпізнавання символів) [1].

Мета проєкту – створення нейромережевої системи для підвищення ефективності розпізнавання документів та автоматичного введення даних.

Для досягнення поставленої мети буде використано:

- знання про технології OCR;
- бібліотеки, що працюють з нейронними мережами для пошуку та

розпізнавання символів на зображеннях;

- знання мови програмування Python;
- допоміжні навички роботи з Web-бібліотеками, такими як Django тощо.

На основі нейромережевої системи може бути можливість використання розпізнавання документів та введення даних.

1 ТЕХНОЛОГІЇ ТА ПРОГРАМНІ ЗАСОБИ ДЛЯ АВТОМАТИЧНОГО РОЗПІЗНАВАННЯ ТЕКСТУ ТА ВНЕСЕННЯ ДАНИХ

1.1 Аналіз якості роботи

З моменту створення першої ЕОМ і до сучасних суперкомп'ютерів головною метою розвитку обчислювальної техніки було й залишається спрощення та оптимізація різних сфер людської діяльності. Якщо на початкових етапах комп'ютери використовувалися переважно для виконання базових обчислень, подібно до калькуляторів, то сьогодні вони охоплюють практично всі сфери життя: політику, освіту, оборону, фінанси, розваги та багато інших галузей. Збільшення рівня автоматизації зумовлене не лише зручністю, а й постійним зростанням обчислювальної потужності, яка розвивається відповідно до закону Мура, згідно з яким кожні 1,5 роки кількість транзисторів на кристалі збільшується на 30%.

З підвищенням обчислювальних можливостей комп'ютерів зростає і складність завдань, які вони можуть виконувати. Обчислення, що можуть зайняти у людини години чи навіть дні, процесор здатний виконати за лічені секунди. Завдяки цьому комп'ютери дедалі частіше застосовуються навіть у таких критично важливих сферах, як медицина. Наприклад, від системи CROE, яка автоматизує створення медичних карток, до DaVinci від Intuitive – роботизованої системи для дистанційного проведення операцій.

Попри значний прогрес у сфері технологій за останні 20 років – розвиток інтернету, веб-технологій, штучного інтелекту, робототехніки та нейронних мереж – соціальна сфера, зокрема в Україні, суттєво відстає. У більшості державних установ, таких як страхові компанії, пенсійні фонди, податкові інспекції, а також правоохоронні та медичні установи (поліклініки, поліція, швидка допомога), все ще широко використовуються паперові документи. Незважаючи на паралельне впровадження комп'ютерних систем, процес перенесення інформації з друкованих документів (паспортних даних, свідоцтв,

договорів) в електронний вигляд є тривалим і неефективним.

Перехід до електронних документів, цифрових підписів та централізованих систем документообігу має значний потенціал, зокрема для бізнесу, оскільки це дозволить мінімізувати ризик втрати важливих документів. Крім того, поступова відмова від паперових носіїв сприятиме збереженню природних ресурсів, що позитивно вплине не лише на екологію, а й на економіку та фінансову сферу.

Однак поки що важливо не лише орієнтуватися на майбутнє повної цифровізації, а й забезпечити швидку та ефективну обробку документів у сучасних реаліях. Наприклад, значна частина часу у державних установах витрачається на ручне переписування паспортних даних та інших особистих відомостей. Використання технології оптичного розпізнавання символів (OCR) дозволило б скоротити цей процес у десять разів [2].

Електронні черги, які нині запроваджуються у багатьох державних установах, поки що не можуть повністю вирішити проблему затримок через велику кількість громадян, особливо літнього віку, які не користуються такими послугами. Тому основним варіантом оптимізації роботи соціальних служб є автоматизація введення та обробки даних, що значно полегшить завдання працівників та прискорить обслуговування громадян.

Особливо це актуально для паспортних столів, які виконують такі функції, як:

- набуття громадянства України;
- видача паспорта громадянина України;
- оформлення документів для виїзду за кордон;
- реєстрація або зняття з реєстрації місця проживання;
- документування іноземців;
- надання статусу біженця або додаткового захисту.

Без автоматизації обробки особистих даних працівники змушені виконувати значний обсяг рутинної роботи вручну, що призводить до перевантаження та негативного ставлення громадян до роботи установ. Відсутність системи автоматизованого розпізнавання тексту призводить до того,

що навіть найпростіші завдання займають багато часу та ресурсів[3].

Впровадження OCR-систем дозволить значно прискорити процеси опрацювання документів, зменшити ризик втрати даних, а також знизити ймовірність помилок, пов'язаних із людським фактором.

Основні причини переходу на автоматизовану систему введення даних:

- екологічний аспект – зменшення вирубки лісів;
- скорочення черг у державних установах;
- зниження ризику помилок під час обробки інформації завдяки усуненню людського фактору.

Таким чином, розробка та впровадження системи автоматичного розпізнавання документів стане значним кроком до оптимізації роботи державних установ і сприятиме загальному переходу до цифрового документообігу.

1.2 Огляд аналогів для вирішення задач

Як уже згадувалося, існує низка програмних рішень, що дозволяють безпосередньо отримувати текст із фотографій або камери. Вони працюють на основі технології оптичного розпізнавання символів. Можна виділити кілька компаній та сервісів, які пропонують такі послуги для користувачів. Ось один із них.

1.2.1 Огляд програмного забезпечення Azoft

Компанія Azoft демонструє принцип роботи своєї системи розпізнавання документів, пояснюючи загальні механізми функціонування їхнього програмного забезпечення.[4] У процесі розробки вони виокремили чотири ключові етапи роботи програми:

- локалізація;
- фільтрування;
- вилучення рядків;

- розпізнавання тексту на рівні символів або рядків.

Локалізація передбачає виявлення документа на зображенні. Для цього компанія випробувала три основні підходи:

- використання OpenCV у поєднанні з навченим класифікатором Хаара;
- застосування повнозгорткових нейронних мереж;
- аналітичний підхід, заснований на пошуку зв'язкових компонентів.

У першому методі документ визначається у вигляді прямокутної області, яка поступово обрізається до тих пір, поки в межах рамки не залишиться лише текст [4]. Якщо документ у відеопотоці не виявлено, алгоритм продовжує його пошук.

У другому підході використовуються повнозгорткові нейронні мережі. Вхідні дані для мережі - кольорове зображення паспорта. На виході формується два канали: перший визначає центр першої сторінки документа, а другий - центр другої сторінки. При цьому мережа навчається передбачати маску з гаусовими піками, що відповідають центрам сторінок паспорта (рисунок 1.1).

Третій підхід було застосовано для розпізнавання паспорта. Компанія використала шаблон, який передбачав наявність двох спеціальних областей із серією та номером документа на розвороті паспорта. Якщо у кадрі виявлявся такий шаблон, система визначала, що перед нею знаходиться паспорт.

Оскільки фонові елементи документів, такі як ламіноване покриття, відблиски світла та водяні знаки, створюють перешкоди для коректного розпізнавання символів, їх необхідно усунути. Для цього було застосовано такі методи:

- алгоритм фільтрації шуму - `'fastNlMeansDenoisingColored'`, який використовує вікно відповідного розміру для згладжування ліній;
- білатеральний фільтр, що дозволяє отримати більш однорідний фон документа;
- метод адаптивної бінаризації, який допомагає покращити контрастність та виокремити текст.

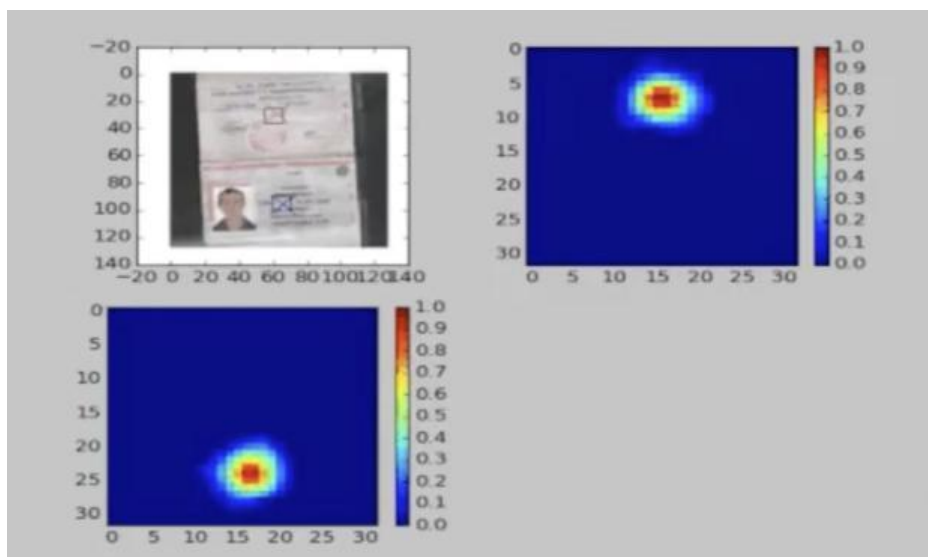


Рисунок 1.1 – Розподілення гаусові пики

Для розпізнавання водійських прав застосовувався OpenCV – система детектувала контури літер, після чого текст розбивався на окремі символи. Ці символи передавалися на вхід нейронної мережі для подальшого розпізнавання (рисунок 1.2).

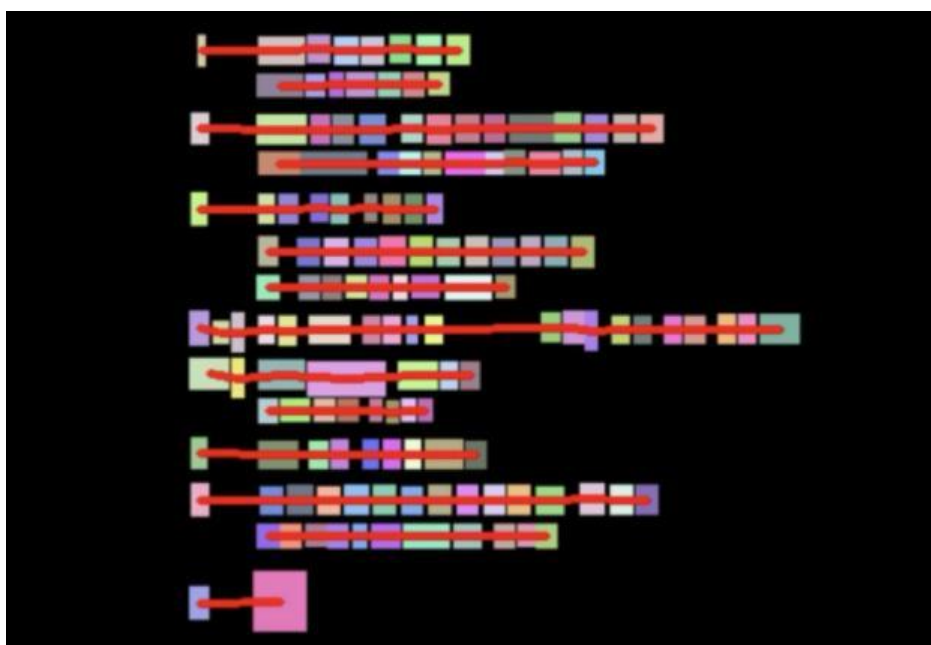


Рисунок 1.2 – Зображення як система зчитує рядки

Отримання рядків паспорта. Для виокремлення рядків тексту у паспорті використовується нейронна мережа, яка дозволяє точно ідентифікувати

розташування тексту.

Розпізнавання тексту. Для розпізнавання тексту застосовуються нейронні мережі. У випадку з водійськими правами текст розпізнається по літерах, використовуючи згорткову нейронну мережу, яка була навчена на великому датасеті зображень символів. Для її створення та навчання використовувався фреймворк Torch.

Текст у паспорті також розпізнається повнозгортковою нейронною мережею. При цьому реєстр символів не враховується. Для ефективного навчання мережі було створено спеціальний скрипт – генератор навчальної вибірки, який дозволяє підготувати дані для тренування моделі.

На малюнку 1.3 представлено послідовність дій для передачі даних.

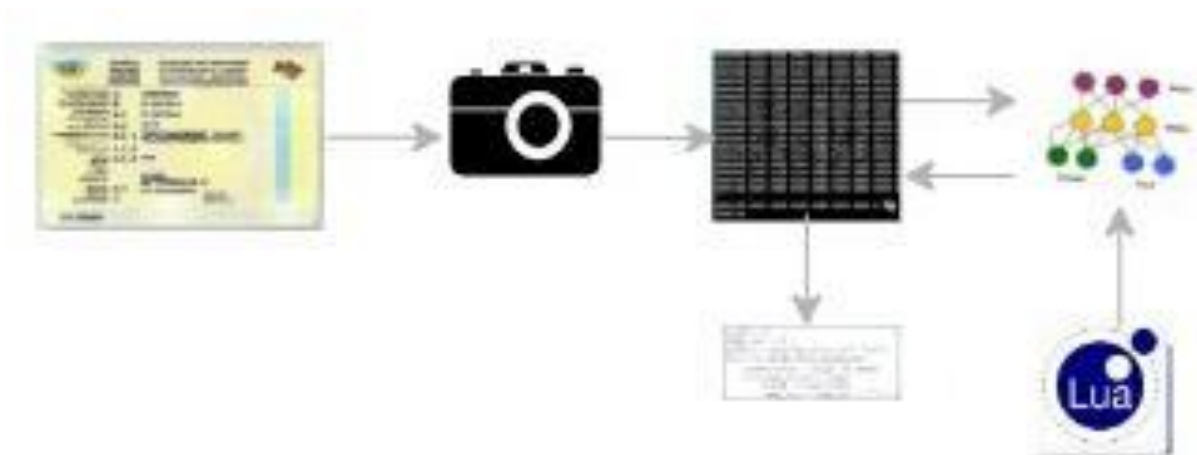


Рисунок 1.3 – Схема передачі даних для обробки

Водійське посвідчення. Документ, що підтверджує право особи керувати транспортними засобами. Для розпізнавання даних з водійських посвідчень використовуються технології комп'ютерного зору та нейронні мережі.

Камера. Пристрій для зйомки зображень або відео. У цьому контексті камера використовується для захоплення зображення водійського посвідчення чи інших документів, які потім обробляються системою розпізнавання.

Виконуваний бінарний файл. Файл, що містить скомпільований програмний код, який може бути безпосередньо виконаний на комп'ютері або

іншому пристрої. У цьому випадку бінарний файл є результатом компіляції програмного коду, наприклад, на мові C++.

Вихідні коди C++, у тому числі зібраний виконуваний файл. Код програми, написаний мовою програмування C++, що реалізує алгоритми для розпізнавання тексту або інші функціональні можливості. Зібраний виконуваний файл є результатом компіляції цього коду, який може бути запущений на пристрої.

Lua скрипт, який здійснює роботу з нейромережею. Скрипт, написаний мовою програмування Lua, який взаємодіє з нейронною мережею, керує її навчанням та виконанням. У скрипті описана архітектура мережі, яка визначає, як обробляються вхідні дані та як отримуються результати розпізнавання.

Бінарний файл з вагами мережі, що завантажується Lua скриптом. Файл, що містить ваги нейронної мережі після її навчання. Ці ваги є результатом навчання мережі на великому наборі даних, і Lua скрипт використовує цей файл для ініціалізації мережі перед її виконанням.

1.2.2 Програмне забезпечення компанії EasyOCR2

Euresys - це провідна інноваційна високотехнологічна компанія, що спеціалізується на розробці та постачанні компонентів для збору зображень та відео, пристроїв захоплення кадрів, IP-ядер FPGA та програмного забезпечення для обробки зображень. Компанія активно працює у сферах комп'ютерного зору, машинного зору, промислової автоматизації та медичної візуалізації [5].

Завдяки багаторічному досвіду у галузі збору зображень, компанія Euresys займається отриманням як аналогових, так і цифрових відео, програмуванням FPGA, розробкою високочастотної електроніки, стисненням відео та управлінням камерами. Після придбання компанії Sensor to Image, Euresys набув 30-річного досвіду в обробці зображень, що охоплює такі технології, як GigE Vision, USB3 Vision, CoaXPress, Camera Link і GenICam.

У сфері аналізу зображень Euresys спеціалізується на таких завданнях, як виявлення плям, субпіксельне вимірювання, зіставлення образів, аналіз кольору, оптичне розпізнавання символів, зчитування та перевірка штрих-кодів, 3D-перевірка та класифікація з використанням глибокого навчання (1.4).

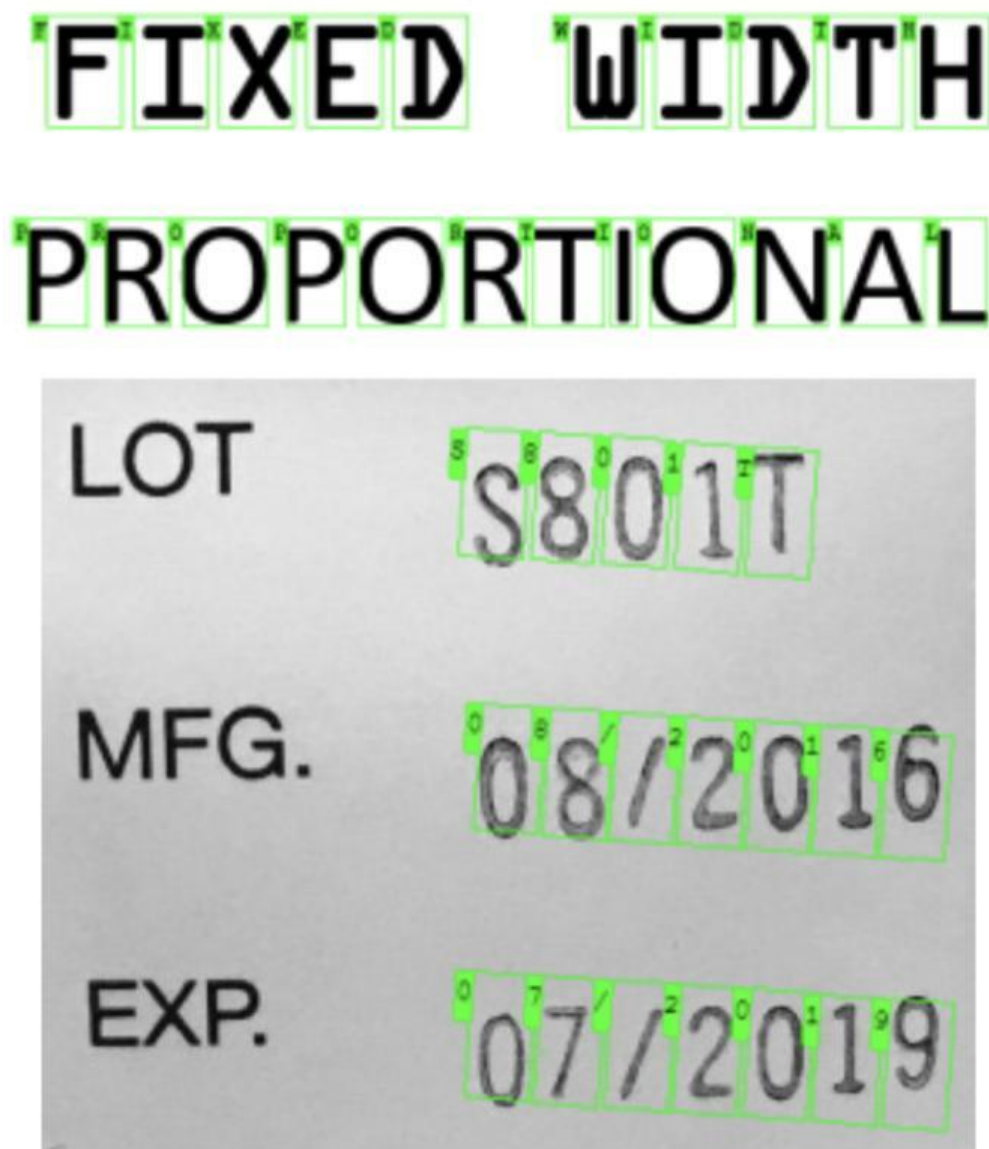


Рисунок 1.4 – Приклад роботи ексклюзивного продукту компанії Euresys

EasyOCR2 - це технологія, яка здійснює процес розпізнавання символів на зображеннях, працюючи наступним чином:

- сегментація зображення - EasyOCR2 виявляє краплі, що складають частини символів, на зображенні;
- фільтрація крапель - краплі, які є занадто великими або занадто малими для того, щоб вважатися частинами символів, відфільтровуються;
- підгонка блоків символів - EasyOCR2 підганяє блоки символів до виявлених великих двійкових об'єктів згідно з заданою топологією та методом виявлення (топологія визначає структуру тексту на зображенні, наприклад,

кількість рядків, кількість слів у кожному рядку та кількість символів у словах);

- витягування пікселів - EasyOCR2 витягує пікселі всередині кожного символного поля на зображенні, що дозволяє отримати чисті зображення символів, які можна використовувати для подальшого навчання або розпізнавання.

Цей процес дозволяє ефективно і точно здійснювати розпізнавання тексту на зображеннях, що є важливим для різних застосувань, таких як оптичне розпізнавання символів (OCR).

1.3 Постановка завдання

Беручи до уваги актуальність цієї проблеми, можна стверджувати, що основним об'єктом дослідження є нейронні мережі, які є найпопулярнішим напрямом у сфері розвитку та дослідження штучного інтелекту. У свою чергу, предметом дослідження є розробка та налаштування нейронної системи для розпізнавання тексту за допомогою технології OCR.

Мета: здійснення розпізнавання тексту з документів, з подальшою можливістю використання розпізнаного тексту.

Для досягнення цієї мети будуть використані:

- знання про хоча б одну систему OCR;
- бібліотеки, що використовують нейронні мережі для пошуку та розпізнавання символів на зображеннях;
- орієнтація на програмування мовою Python.

Для виконання завдання необхідно визначити вимоги до проекту, а саме:

- призначення - програма, яка буде використовуватися для зчитування тексту з фотографій або інших зображень, що містять текст;
- кордон проекту - зчитування тексту з фотографії та його виведення у вигляді рядкових змінних;

- класи користувачів - користувачами можуть бути як фізичні, так і юридичні особи;
- основний користувач - працівники соціальної сфери;
- вимоги до даних - зображення повинні бути в одному з форматів (png, jpg). Якість фотографії має бути доброю та без розмиття, оскільки зниження якості може призвести до помилок у розпізнаванні. Використання сканованих документів також полегшить процес розпізнавання.

1.4 Висновки до розділу

З урахуванням всього, що було розглянуто в першому розділі, можна зробити висновок, що порушена проблема є надзвичайно актуальною як сьогодні, так і в майбутньому, коли почнеться епоха комп'ютеризації всіх аспектів людської діяльності. Немає сумнівів, що подібні технології будуть постійно удосконалюватися, а штучний інтелект ставати дедалі більш досконалим, наближаючись до функціонування та інтелекту людини.

Мета даного проєкту полягає в тому, щоб полегшити роботу соціальних служб, які щодня обслуговують велику кількість людей, зокрема людей похилого віку. Це, у свою чергу, значно спростить роботу працівників соціальних служб та покращить ефективність надання послуг.

2 ІНСТРУМЕНТИ ДЛЯ РЕАЛІЗАЦІЇ КОМП'ЮТЕРНОЇ СИСТЕМИ

2.1 Варіанти вирішення задачі

OCR (Optical Character Recognition) - це технологія, що використовується для визначення та перетворення друкованого або рукописного тексту з

відсканованих документів у цифровий формат, який легко розпізнається комп'ютерними програмами. Основний процес розпізнавання передбачає аналіз текстових символів і трансформацію їх у код, який може бути використаний для подальшої обробки. У деяких випадках OCR також називають системою розпізнавання тексту [6].

У 1974 році Рей Курцвейл заснував компанію Kurzweil Computer Products, Inc., яка розробила систему багатошрифтового оптичного розпізнавання символів, здатну ідентифікувати текст, надрукований у будь-якому шрифті. Він вважав, що найкращим застосуванням цієї технології стане пристрій для машинного навчання для людей із вадами зору. Тому Курцвейл створив машину, яка могла зчитувати текст і синтезувати мовлення. У 1980 році він продав свою компанію корпорації Xerox, яка прагнула розвивати технологію перетворення друкованого тексту в цифровий формат.

На початку 1990-х років OCR набуло популярності завдяки масовому оцифруванню історичних газет. Відтоді технологія значно вдосконалилася, досягнувши майже ідеальної точності розпізнавання. Для автоматизації обробки документів застосовуються передові методи. До появи OCR єдиним способом переведення паперових матеріалів у цифровий формат було ручне введення тексту, що займало багато часу й часто супроводжувалося помилками. Сьогодні OCR-сервіси широко доступні, наприклад, Google Cloud Vision OCR дозволяє сканувати та зберігати документи прямо зі смартфона.

Технологія OCR поєднує апаратне й програмне забезпечення для конвертації фізичних документів у машиночитаний текст. Оптичні сканери або спеціалізовані пристрої здійснюють зчитування, а програмне забезпечення аналізує й обробляє текст. Використання штучного інтелекту дозволяє вдосконалювати процес розпізнавання, зокрема ідентифікувати мову та стилі рукописного введення.

OCR часто застосовується для конвертації юридичних і архівних документів у PDF-файли, що згодом можуть редагуватися та форматовуватися у стандартних текстових редакторах.

Першим етапом оптичного розпізнавання тексту є сканування документа для обробки його фізичної форми. Після цього OCR-програма створює двоколірне або чорно-біле зображення документа. Аналізуючи отримане растрове зображення, система визначає світлі та темні області: темні інтерпретуються як символи, що підлягають розпізнаванню, а світлі - як фон. Далі темні області аналізуються для виявлення літер або цифр.

Різні OCR-програми можуть використовувати різні методи розпізнавання, але загальний процес включає ідентифікацію окремих символів, слів або блоків тексту. Основні алгоритми обробки базуються на розпізнаванні особливостей шрифтів і текстових форматів.

Технологія OCR застосовує правила виявлення ознак, характерних для конкретних літер і цифр. Програмне забезпечення аналізує, як формується символ - наприклад, велика літера "А" може бути представлена як дві діагональні лінії з горизонтальною лінією посередині. Після визначення символу він кодується у формат ASCII, що робить його придатним для подальшого використання в комп'ютерних системах. Перед остаточним збереженням перевіряється коректність розпізнаного тексту, особливо у випадках складних макетів.

Основні застосування OCR:

- сканування друкованих документів для подальшого редагування у текстових редакторах.
- індексування текстів для пошукових систем;
- автоматизована обробка та введення даних;
- перетворення тексту в аудіоформат для людей із порушенням зору;
- оцифрування та архівування історичних газет і журналів із можливістю пошуку;
- збереження юридичних документів у електронних базах;
- розпізнавання номерних знаків транспортних засобів за допомогою камер контролю швидкості;
- автоматичне сортування поштових відправлень;
- переклад тексту на зображенні іншою мовою;

- пошук інформації у сканованих книгах.

До появи OCR оцифрування паперових документів вимагало ручного введення тексту, що було тривалим і супроводжувалося помилками. Завдяки цій технології процес став швидшим і точнішим, мінімізуючи людський фактор. Крім того, OCR дозволяє виконувати дії, неможливі для фізичних копій, такі як стиснення у ZIP-файл, пошук за ключовими словами, публікація документів онлайн або їх прикріплення до електронних листів.

Метрика - це певна функція, що визначає відстань між об'єктами в просторі. Якщо два об'єкти розташовані близько один до одного (наприклад, дві літери "А" у різних шрифтах), їхні метрики будуть або збігатися, або бути дуже схожими. У цьому методі використовується метрика Хеммінга, яка визначає рівень відмінностей між об'єктами.

Ця метрика часто застосовується в кодуванні та передачі даних. Наприклад, якщо передана бітова послідовність (1001001) порівнюється з очікуваною (1000101), то різниця у двох позиціях дає значення метрики Хеммінга, рівне 2. Чим більше значення, тим більша відмінність між об'єктами.

Щоб розпізнати літеру, її метрика порівнюється з метриками всіх доступних шаблонів, а найближчий до 0 результат вказує на правильний варіант. Проте лише одного такого підходу недостатньо, оскільки багато символів схожі між собою (наприклад, "i" та "j"), що спричиняє помилки у розпізнаванні.

Для покращення точності були розроблені додаткові метрики, такі як:

- відображення по горизонталі та вертикалі;
- розподіл ваги по горизонталі та вертикалі.

Аналіз показав, що символи "H", "I", "i", "O", "o", "X", "x", "l" мають суперсиметрію (збігаються зі своїми відображеннями, а значущі пікселі рівномірно розподілені). Тому їх винесли в окремий клас, що дозволило скоротити кількість необхідних обчислень у 6 разів. Аналогічно були згруповані й інші літери, що в середньому зменшило список варіантів у 3 рази.

Деякі символи, як-от "J", є унікальними у своїх класах, що робить їх однозначно розпізнаваними. Після класифікації кожного символу метрика

Хеммінга використовується на більш локальному рівні, що значно покращує точність.

Розпізнавання було налаштоване для шрифту «Consolas». При використанні цього шрифту точність досягає 99%, але при зміні шрифту падає до 70%.

Нейронні мережі є складною темою як у біологічному, так і в математичному сенсі. У спрощеному вигляді вони імітують принцип роботи людського мозку. Одним із прикладів такої моделі є одношарова мережа Кохонена.

Принцип роботи нейромережі такий: при подачі нового зображення на вхідний шар нейронів один з них активується найбільше. Оскільки кожен нейрон відповідає певному символу, реакція нейрона визначає результат розпізнавання.

Кожен нейрон має багато входів, які відповідають пікселям зображення. Наприклад, якщо розмір символу 16×16 , мережа має 256 входів.

Вхідні дані мають власні коефіцієнти ваги, які визначають вплив кожного пікселя. Після обробки всієї інформації нейрон з найбільшим зарядом активується і визначає символ.

Перед використанням мережу потрібно навчити. Це робить спеціальний навчальний модуль, який:

- передає навчальне зображення в мережу;
- аналізує чорні пікселі та налаштовує вагові коефіцієнти, мінімізуючи помилки;
- використовує градієнтний метод для коригування зв'язків між нейронами.

Цей метод дозволяє розпізнавати текст з високою точністю, незалежно від шрифту, оскільки нейромережа навчається на великому обсязі даних і може адаптуватися до різних стилів написання символів.

2.2 Засоби розробки системи

Розпізнавання тексту можна реалізувати за допомогою різних технологій. Зважаючи на те, що завдання пов'язане зі штучним інтелектом і нейронними мережами, доцільно використовувати мову програмування Python.

Серед найбільш популярних технологій OCR виділяють:

- Tesseract OCR;
- EasyOCR.

Tesseract - це механізм оптичного розпізнавання тексту з відкритим вихідним кодом, доступний за ліцензією Apache 2.0. Він може працювати як самостійно, так і через API, що дозволяє розробникам інтегрувати його у свої застосунки для вилучення друкованого тексту із зображень (рис.2.1).

Основні особливості Tesseract OCR [7]:

- підтримує багато мов;
- не має власного графічного інтерфейсу, але підтримує сторонні GUI
- сумісний з різними мовами програмування та фреймворками;
- може працювати як із цілими документами, так і з окремими текстовими рядками.

Tesseract можна використовувати для розпізнавання тексту у великих документах разом із вбудованим аналізом макета або в поєднанні з зовнішнім детектором тексту, що дозволяє обробляти окремі рядки тексту.

Версія Tesseract 4.00 включає вдосконалену підсистему нейронної мережі для розпізнавання текстових рядків. Вона була створена на основі LSTM, що походить від Python-бібліотеки OCRopus, але була переписана на C++ спеціально для Tesseract.

Хоча система нейронних мереж у Tesseract була розроблена ще до появи TensorFlow, вона сумісна з ним завдяки VGSL. VGSL дозволяє описувати архітектуру нейромереж, що робить її придатною як для Tesseract, так і для TensorFlow.

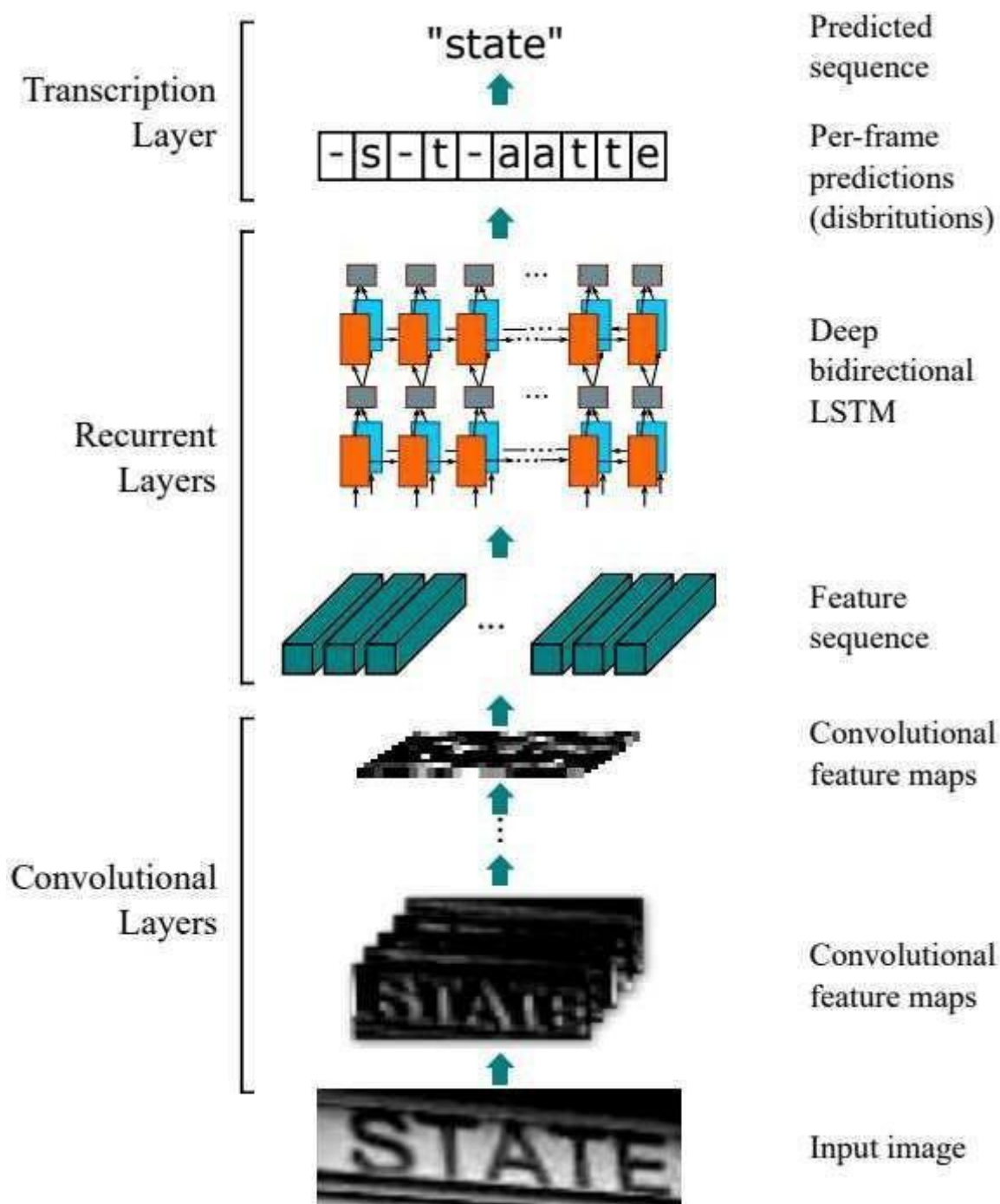


Рисунок 2.1 – Структура процесу Tesseract OCR

Залежно від типу розпізнаваного тексту застосовуються різні архітектури неймереж [8]:

- CNN - використовуються для розпізнавання окремих символів;
- RNN - застосовуються для обробки послідовностей символів;
- LSTM - одна з найефективніших форм RNN, що використовується для розпізнавання тексту довільної довжини.

Хоча LSTM добре підходять для аналізу послідовностей, їх продуктивність знижується при збільшенні кількості станів. Дослідження показують, що ефективніше навчати LSTM на довгих послідовностях, а не на коротких із багатьма класами.

Tesseract базується на моделі OCRopus, але її було оптимізовано на C++ у вигляді бібліотеки CLSTM (рис.2.2).

CLSTM - це реалізація LSTM на C++, яка використовує бібліотеку Eigen для швидких чисельних обчислень. Це дозволяє значно підвищити швидкість і точність розпізнавання тексту.

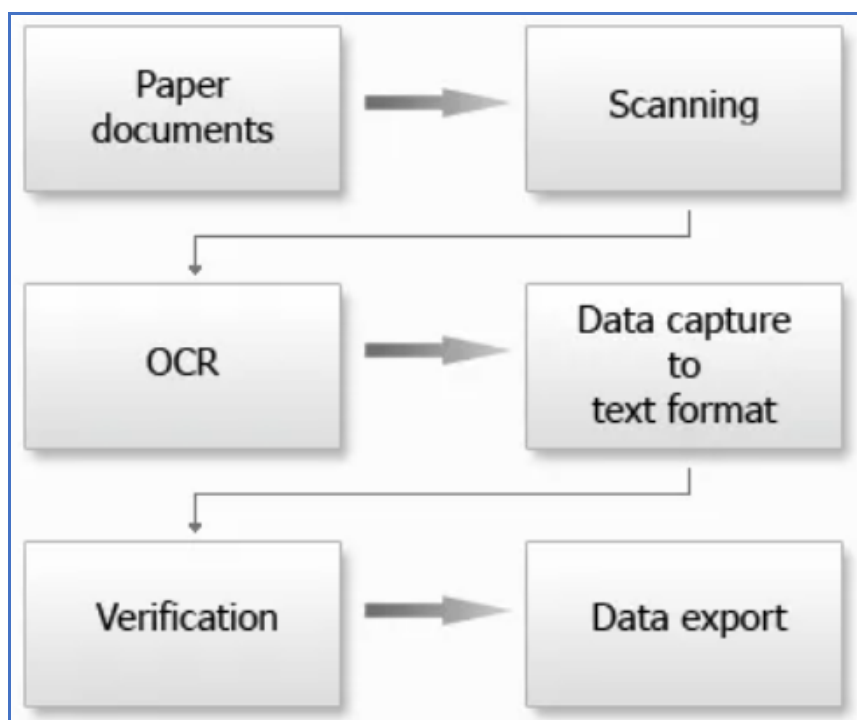


Рисунок 2.2 – Робота Tesseract з текстом

У версіях Tesseract 3.x використовувався багатоетапний підхід до розпізнавання тексту, що включав три основні кроки [9]:

- пошук слів;
- пошук ліній;
- класифікація символів.

Процес розпізнавання в Tesseract 3.x складається із наступних дій:

- пошук слів здійснювався шляхом організації текстових рядків у "краплі"

(droplets), що дозволяло аналізувати текст у межах рядків та областей;

- аналіз проводився для виявлення фіксованого або пропорційного інтервалу між символами;

- рядки тексту поділялися на слова за міжсимвольними проміжками.

Двопрохідний процес розпізнавання:

а) перший прохід:

- 1) кожне слово розпізнавалося окремо;

- 2) слова, які були ідентифіковані з високою ймовірністю, використовувалися для навчання адаптивного класифікатора;

б) другий прохід:

- 1) адаптивний класифікатор застосовував отримані знання для більш точного розпізнавання тексту в нижній частині сторінки.

Модернізація Tesseract було спрямоване на перехід на LSTM:

- очищення коду;

- додавання нової LSTM - моделі для точнішого розпізнавання;

- вхідне зображення аналізується рядково;

- кожен рядок подається у модель LSTM у вигляді прямокутних блоків;

- на виході модель LSTM видає розпізнаний текст.

Завдяки використанню LSTM значно покращилася точність розпізнавання та швидкість обробки тексту, особливо для складних шрифтів і нерівномірно розташованих символів.

Пакет EasyOCR розроблений та підтримується компанією Jaided AI, яка спеціалізується на технологіях оптичного розпізнавання символів (OCR).

Цей інструмент реалізований мовою Python із використанням бібліотеки PyTorch. Якщо ваш комп'ютер оснащений графічним процесором із підтримкою CUDA, то бібліотека глибокого навчання PyTorch може значно прискорити як виявлення тексту, так і його розпізнавання.

На даний час EasyOCR підтримує 58 мов, серед яких англійська, німецька, хінді, російська та багато інших. Розробники планують розширити список доступних мов у майбутніх оновленнях. Актуальний перелік підтримуваних мов

можна знайти на офіційній сторінці EasyOCR (рис.2.3).

EasyOCR здатний поки що розпізнавати лише друкований текст. Проте розробники планували випустити модель для рукописного тексту декілька років тому.

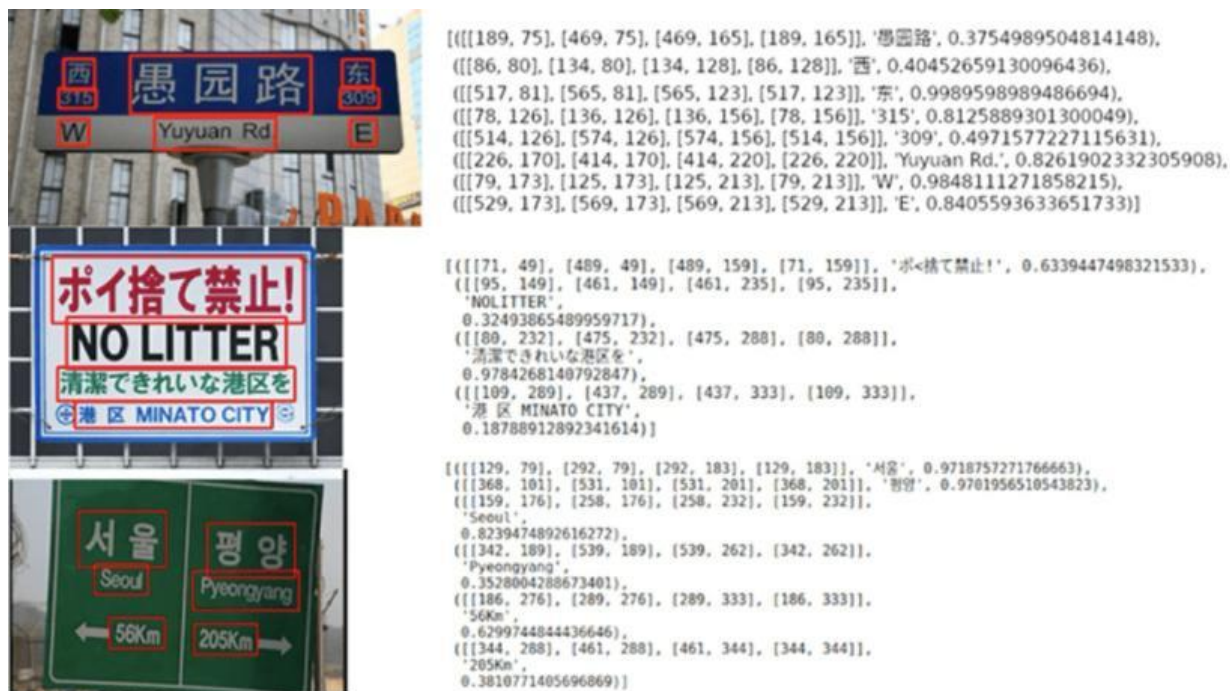


Рисунок 2.3 – Можливості OCR технологій

Окрім EasyOCR, існує безліч інших OCR-бібліотек та технологій, здатних зчитувати текст із зображень. Також доступні онлайн-сервіси для розпізнавання тексту, як платні, так і безкоштовні. Однак якість безкоштовних сервісів часто залишає бажати кращого, що робить їх непридатними для реалізації, описаної в цій бакалаврській роботі.

Комп'ютерний зір - це процес, за допомогою якого комп'ютери здатні інтерпретувати зображення та відео, зберігати їх та маніпулювати ними, а також витягувати з них корисні дані. Ця технологія є ключовою для розвитку штучного інтелекту. Комп'ютерний зір відіграє важливу роль у таких сферах, як безпілотні автомобілі, робототехніка та програми для обробки фотографій.

OpenCV - це потужна бібліотека з відкритим вихідним кодом, яка забезпечує можливості для комп'ютерного зору, машинного навчання та обробки

зображень. Вона відіграє ключову роль у реальному часі, що є надзвичайно важливим для сучасних систем [10]. Завдяки OpenCV можна виконувати обробку зображень і відео для ідентифікації об'єктів, осіб чи навіть почерку. Відсутність цієї бібліотеки ускладнила б вирішення багатьох завдань, зокрема первинного завдання - дозволити комп'ютеру розпізнавати зображення, зокрема для камер, які можуть зчитувати документи без необхідності їх фотографування, просто підносячи їх до об'єктива.

OpenCV інтегрується з різними бібліотеками, такими як NumPy, що дозволяє Python обробляти структури OpenCV для подальшого аналізу. Для виявлення шаблонів зображень і різноманітних функцій використовуються векторні простори та математичні операції з ними.

Перша версія OpenCV була випущена під номером 1.0. Бібліотека поширюється за ліцензією BSD, що робить її безкоштовною для академічного та комерційного використання. OpenCV має інтерфейси для C++, C, Python та Java, а також підтримує операційні системи Windows, Linux, Mac OS, iOS та Android. Під час розробки основну увагу було приділено досягненню високої обчислювальної ефективності для застосувань у реальному часі, а всі компоненти написані на оптимізованих мовах C/C++ для максимальної продуктивності на багатоядерних процесорах.

З наведеного нижче зображення (рис.2.4) можна отримати низку фрагментів, що містять різну інформацію, присутню на вихідному зображенні. Як у цьому прикладі, ми можемо виявити на зображенні дві особи, а також людину (я), що носить браслет або годинник. За допомогою OpenCV можна отримати цю інформацію та інші деталі з вихідного зображення.

Обробка зображень є методом виконання різноманітних операцій із зображеннями з метою покращення їх якості або вилучення корисної інформації. Якщо говорити про основне визначення обробки зображень, то це «аналіз та обробка оцифрованого зображення, зокрема для покращення його якості».

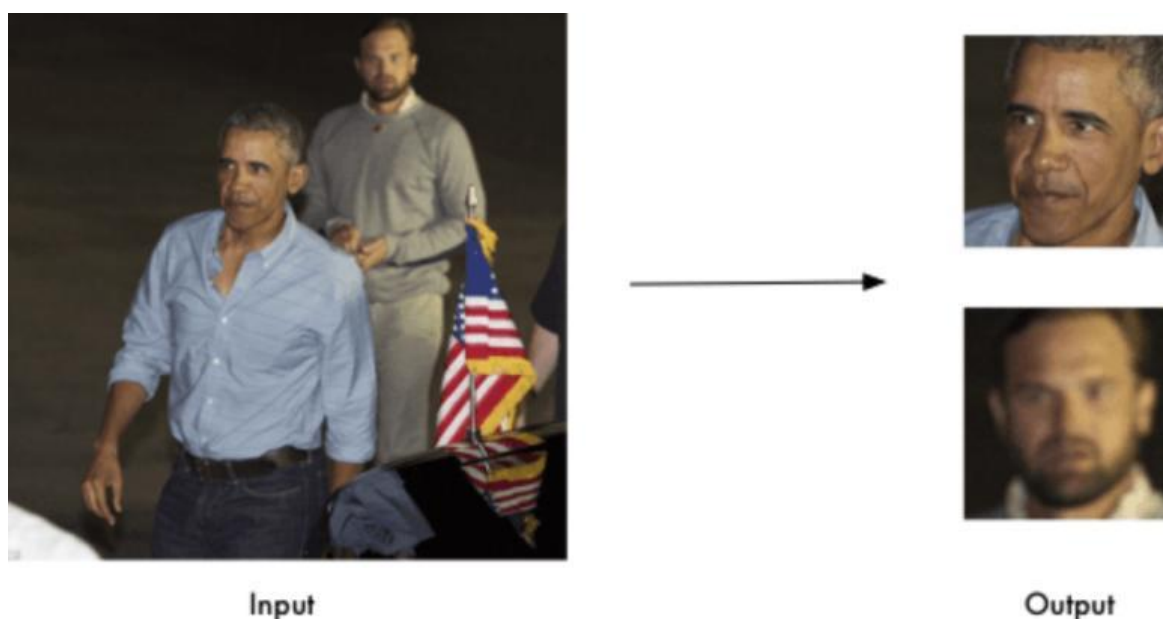


Рисунок 2.4 – Фрагментація інформації за допомогою OpenCV

Програми OpenCV охоплюють багато різноманітних завдань, ось деякі з них:

- розпізнавання осіб;
- автоматизований контроль і спостереження;
- підрахунок людей (наприклад, для визначення прохідності в торгових центрах);
- підрахунок транспортних засобів на автомагістралях і визначення їх швидкості;
- інтерактивні арт-інсталяції;
- виявлення аномалій або дефектів у виробничих процесах (наприклад, для визначення бракованих виробів);
- зшивання зображень для створення оглядів вулиць;
- пошук і вилучення відео та зображень;
- роботизована навігація та управління безпілотними автомобілями;
- розпізнавання об'єктів;
- аналіз медичних зображень;
- створення 3D-структур з руху для кінематографії;
- розпізнавання реклами на телевізійних каналах;

- основні функції OpenCV;
- обробка і відображення зображень/відео (ядро, imgproc, highgui);
- виявлення об'єктів та функцій (objdetect, features2d, nonfree);
- монокулярний і стерео-комп'ютерний зір на основі геометрії (calib3d, stitching, відеостабілізація);
- комп'ютерна фотографія (обробка фотографій і відео, супердозвіл);
- машинне навчання і кластеризація (ml, flann);
- прискорення через CUDA (за допомогою графічного процесора).

Зображення можна описати як двовимірну функцію $f(x, y)$, де x та y - це просторові (площинні) координати, а амплітуда в точці (x, y) називається інтенсивністю або рівнем сірого для цієї пари координат. Іншими словами, зображення є двовимірною матрицею (або тривимірною у випадку кольорових зображень), що визначається математичною функцією $f(x, y)$. Ця функція надає значення пікселя в кожній точці зображення, що описує яскравість і колір пікселя.

Обробка зображень полягає в обробці сигналів, де вхідними даними є зображення, а результатом є змінене зображення або характеристика, відповідно до поставлених вимог. Обробка зображень зазвичай включає три основні етапи:

- імпорт зображення - процес завантаження зображення для подальшої роботи;
- аналіз та обробка зображення - виконання операцій на зображенні для виявлення або покращення потрібних характеристик;
- висновок - результат, який може бути представленим у вигляді зміненого зображення або звіту, заснованого на аналізі зображення.

2.3 Висновки до розділу

З цього розділу можна зробити висновок, що штучний інтелект і нейронні мережі незабаром знайдуть своє застосування в багатьох сферах, починаючи від

автономних транспортних засобів і закінчуючи побутовими технологіями, такими як розпізнавання музики чи голосів. Описані технології, зокрема EasyOCR і OpenCV, відкривають великі можливості для широкого кола користувачів. Завдяки цим інструментам, якщо поставити мету, можна розробити багато інноваційних рішень, які тільки починають входити в наше повсякденне життя.

До таких можливостей можна віднести:

- автоматизовані боти для надсилання повідомлень у месенджерах;
- голосові команди для управління комп'ютером, смартфоном або навіть для RGB-світильників та інших побутових пристроїв.

Цей список можна продовжувати без кінця. І це не є чимось віддаленим чи фантастичним. Це все досяжно для кожного, хто захоплюється програмуванням та вивченням нейронних мереж.

Технології комп'ютерного зору та розпізнавання символів, об'єктів і так далі - це новий етап у розвитку комп'ютеризації, який вимагає значних удосконалень і вдосконалення. Проте, ми також маємо на увазі й потенційні ризики. Ці питання піднімають також письменники-фантасти, чия уява часто передбачає технології, які можуть змінити наше майбутнє. Однак важливо залишатися пильними та обережними щодо автоматизації і розвитку штучного інтелекту, адже якщо не контролювати цей процес, можна опинитися на межі нових викликів, що можуть поставити перед нами серйозні загрози.

3 МОДЕЛЮВАННЯ І РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ

3.1 Опис та використання бібліотеки Tesseract

Для виконання цієї роботи був обраний пакет EasyOCR. Хоча всі пакети та бібліотеки, що забезпечують функціонал розпізнавання тексту, мають свої переваги та недоліки, цей пакет виявився найбільш підходящим для реалізації

завдання.

Спочатку був зроблений вибір на користь Tesseract, однак його основним недоліком є відсутність підтримки для операційної системи MacOS, що є актуальним для поточного робочого ноутбука. Відсутність сумісності з MacOS пов'язана з тим, що для використання Tesseract необхідно додавати його до змінної шляху PATH, що є простим для ОС Windows, де система більш гнучка. Водночас, безпека в MacOS обмежує можливість завантажувати open-source ресурси з Інтернету в змінну PATH. Через це при спробі імпортувати бібліотеку Tesseract у проект виникає помилка, що повідомляє про відсутність Tesseract у змінній PATH. Рисунок подають одразу після тексту, де вперше посилаються на нього, або якнайближче до нього на наступній сторінці, а за потреби – в додатках. Назва рисунка має відображати його зміст, бути конкретною та стислою (рис.3.1).

```
File "/home/appuser/venv/lib/python3.8/site-packages/streamlit/script_runner.py", line
    exec(code, module.__dict__)
File "/app/ocr_python/main.py", line 75, in <module>
    ocr.inicial()
File "/app/ocr_python/main.py", line 28, in inicial
    self.texto = self.extrair_texto(img)
File "/app/ocr_python/main.py", line 38, in extrair_texto
    texto = pytesseract.image_to_string(img, lang="por")
File "/home/appuser/venv/lib/python3.8/site-packages/pytesseract/pytesseract.py", line
    return {
File "/home/appuser/venv/lib/python3.8/site-packages/pytesseract/pytesseract.py", line
    Output.STRING: lambda: run_and_get_output(*args),
File "/home/appuser/venv/lib/python3.8/site-packages/pytesseract/pytesseract.py", line
    run_tesseract(**kwargs)
File "/home/appuser/venv/lib/python3.8/site-packages/pytesseract/pytesseract.py", line
    raise TesseractNotFoundError()
```

Рисунок 3.1 – Результати частини коду

Причина виникнення труднощів при встановленні пакету Tesseract на операційній системі MacOS може полягати в тому, що на момент звернення до головного сайту Tesseract посилання на середовище розробників та GitHub було

переміщене або втрачено за ініціативою розробника. Це підтверджується тим, що в інструкціях із встановлення Tesseract для MacOS містяться застарілі та некоректні команди, що ускладнює процес завантаження та встановлення бібліотеки.

З метою збереження продуктивності робочого ноутбука було ухвалене рішення використовувати бібліотеку EasyOCR для реалізації завдання в рамках БКР. Хоча існують варіанти для вирішення цієї проблеми, наприклад, завантаження бібліотеки Tesseract через розпакування .exe файлу, цей метод не підходить для MacOS, оскільки ця операційна система не підтримує файли з розширенням .exe, в той час як .pkg файл для MacOS просто відсутній.

Це обмеження, зрештою, виявилось навіть на краще, оскільки для розпізнавання тексту з фотографій та відеокамер використання пакету Tesseract виявляється не настільки вигідним.

Отже, основні недоліки Tesseract можна підсумувати наступним чином:

- з березня 2024 року посилання на пакет на GitHub було перенесено або видалено;
- в результаті, бібліотека не може бути використана на MacOS;
- якість роботи пакету, що буде досліджено в наступному пункті, також є важливою для порівняння.

Далі буде проведено порівняльний аналіз кількох бібліотек, зокрема Tesseract та EasyOCR, на практиці для визначення їх якості роботи.

3.2 Вимоги до системи для використання бібліотеки EasyOCR

Бібліотека EasyOCR є сучасним інструментом для оптичного розпізнавання тексту, що підтримує понад 80 мов, включаючи українську. Вона базується на сучасних глибоких нейронних мережах, зокрема на поєднанні PyTorch, CRAFT (Character-Region Awareness for Text Detection) і CRNN

(Convolutional Recurrent Neural Network). Враховуючи цю технологічну базу, система, на якій планується використання EasyOCR, повинна відповідати ряду апаратних і програмних вимог для забезпечення стабільної, продуктивної роботи.

3.2.1 Програмні вимоги

Операційна система:

- Windows 10 / 11 (64-bit);
- Linux (Ubuntu 18.04 або новіше);
- macOS 10.15+;
- підтримка Python у середовищі WSL або Conda у разі Windows.

Мова програмування - Python 3.6–3.10. Рекомендовано Python 3.8 для максимальної сумісності з бібліотеками.

Необхідні бібліотеки та залежності. Для встановлення EasyOCR необхідно мати встановлені наступні компоненти:

- torch (PyTorch);
- torchvision;
- easyocr;
- opencv-python (для роботи із зображеннями);
- Pillow, numpy, scipy - для обробки зображень і матричних операцій.

Приклад команди встановлення:

```
pip install easyocr
```

Сумісність з середовищами розробки:

- Jupyter Notebook;
- Visual Studio Code;
- PyCharm;
- Google Colab (для тестування без потреби локального GPU).

3.2.2 Апаратні вимоги

EasyOCR може працювати як на CPU, так і на GPU. Проте для обробки великої кількості зображень або роботи в реальному часі бажано використовувати GPU.

Мінімальні системні вимоги (для CPU-режиму):

- процесор - Intel Core i3 або AMD Ryzen 3;
- оперативна пам'ять (RAM) - від 4 ГБ;
- місце на диску - від 1 ГБ для бібліотек і моделей;
- графіка - внтегрована графіка або базова дискретна карта (не обов'язково).

Рекомендовані вимоги (для GPU-режиму):

- процесор - Intel Core i5 / i7 або AMD Ryzen 5 / 7;
- RAM - 8 ГБ і більше%;
- GPU - NVIDIA з підтримкою CUDA (GTX 1050 або новіше);
- CUDA Toolkit - відповідна версія до встановленої PyTorch (зазвичай CUDA 10.2 або 11.x);
- місце на диску - 2–3 ГБ (включаючи кеш моделей).

GPU-прискорення суттєво знижує час розпізнавання зображень, особливо при пакетній обробці великої кількості файлів. Наприклад, обробка 100 зображень з текстом на CPU може зайняти до 5–10 хвилин, тоді як на GPU - лише 1–2 хвилини.

3.2.3 Додаткові вимоги

Інтернет-з'єднання: потрібне лише при першому запуску для завантаження попередньо навчених моделей. Після цього EasyOCR може працювати повністю офлайн.

Безпека та доступи: у випадках інтеграції в більші системи або сервіси (наприклад, веб-інтерфейс, API), слід враховувати безпеку доступу до бібліотеки, щоб уникнути виконання небажаних команд або неконтрольованої обробки файлів.

EasyOCR - це зручна, потужна та відносно легка у використанні бібліотека для розпізнавання тексту з зображень. Вона має невисокі вимоги до системи в базовому режимі, але надає розширені можливості за наявності GPU-прискорення. Її гнучкість і універсальність роблять її ідеальним вибором для desktop-додатків, автоматизованих систем аналізу документів, веб-інтерфейсів та мобільних сервісів.

Якщо потрібно - можу додати ще порівняння CPU vs GPU-тестів на прикладі EasyOCR.

3.3 Порівняння пакетів програм для реалізації системи

На ринку доступні такі платні сервіси для оптичного розпізнавання символів та об'єктів, як Amazon Textract [11], когнітивні сервіси Microsoft, хмарний зір Google тощо. Проте, в цьому розділі зосереджено увагу на трьох гарних безкоштовних OCR-бібліотеках з відкритим вихідним кодом: tesseract, EasyOCR та Keras-OCR. Ці бібліотеки продемонстрували високі результати, порівнянні з іншими платними API-сервісами. У розглянуто, як налаштувати ці бібліотеки, використовувати їх та як вони працюють в різних сценаріях застосування.

Бібліотека Keras-OCR підтримує використання URL-посилань як типу вихідних даних у своїй функції `'Pipeline()'`, що значно спрощує процес зчитування. У програмі використовується папка з фотографіями, розміщеними на Github, у вигляді URL-посилань, наданих одним закордонним програмістом, що служить тестом для перевірки ефективності бібліотеки: чим більше зображень бібліотека здатна коректно розпізнати, тим вища її якість. Подібну можливість також надають пакети EasyOCR та Tesseract [12].

За замовчуванням, результати зчитування виводяться у вигляді блоку зображень, накладених на вихідне зображення. Однак, за допомогою наступного фрагменту коду, можна вивести результат у текстовий файл, що дозволяє працювати з отриманими даними після перевірки, що нейронна мережа коректно розпізнала і обробила вхідні дані (лістинг 3.1).

Цей код записує результат розпізнавання у текстовий файл `'my_file.txt'` (рис.3.2), що дає змогу легко працювати з отриманими даними для подальшої обробки.

Лістинг 3.1 – Код для виведення результату

```

```python
x_max = 0
temp_str = ""
myfile = open("my_file.txt", "a+")
for i in prediction_groups[0]:
 x_max_local = i[1][: 0].max()
 if x_max_local > x_max:
 x_max = x_max_local
 temp_str = temp_str + " " + i[0].ljust(15)
 else:
 x_max = 0
 temp_str = temp_str + "\n"
myfile.write(temp_str)
print(temp_str)
temp_str = ""

myfile.close()
```

```



Рисунок 3.2 – Номерні знаки

Спершу розглянемо, як бібліотеки Tesseract та EasyOCR обробляють зображення. На малюнку нижче представлена фотографія (рис.3.3), яку буде зчитувати обидві бібліотеки для порівняння їхньої точності та якості розпізнавання тексту.

Код програми для використання Tesseract OCR з OpenCV у Python представлено у наступному лістингу 3.2.



Рисунок 3.3 – Приклад для розпізнавання тексту

Лістинг 3.2 - Код програми для використання Tesseract OCR з OpenCV

```
```python
import pytesseract
import cv2

Завантаження зображення
img1 = cv2.imread('8.jpg')

Відображення зображення у вікні
cv2.imshow('sample', img1)
cv2.waitKey(0)
cv2.destroyAllWindows()

Розпізнавання тексту
text1 = pytesseract.image_to_string(img1)

Вивід результату
print(text1)
```
```

Цей код дозволяє завантажити зображення, відобразити його у вікні та виконати OCR-розпізнавання тексту за допомогою `Tesseract OCR` (рис.3.4).

eS
 \ - =
 = HR260K8337/m F

Рисунок 3.4 – Скріншот результату роботи програми на основі пакету Tesseract

Безперечно, однією з головних переваг цього пакета є його простота та компактність: функціональність реалізується за допомогою всього кількох рядків коду. Однак при аналізі результатів роботи бібліотеки у порівнянні з вихідним зображенням можна помітити, що точність розпізнавання не завжди є ідеальною (лістинг 3.3, рис.3.5).

Лістинг 3.3 - Покращення продуктивності розпізнавання для конкретних сценаріїв використання

```
```python
import easyocr
import cv2
from PIL import Image, ImageDraw

Ініціалізація моделі для російської та англійської мов
reader = easyocr.Reader(["ru", "en"])

Відкриття зображення
img = Image.open("8.jpg")
img.show() # Відобразити зображення

Виконання розпізнавання тексту
bound = reader.readtext("8.jpg")

Вивід результатів
print(bound)
```

```python
draw = ImageDraw.Draw(img)
for bbox, text, prob in bound:
 (top_left, top_right, bottom_right, bottom_left) = bbox
 draw.polygon([top_left, top_right, bottom_right,
bottom_left], outline="red")

img.show()
```
```

Як випливає з назви, EasyOCR - це проста та легка у використанні бібліотека в порівнянні з іншими. Вона підтримує велику кількість мов, що робить її універсальним інструментом для розпізнавання тексту. Додатково, регулюючи певні гіперпараметри, можна покращити продуктивність та точність розпізнавання для конкретних сценаріїв використання.

```
[([[72, 92], [214, 92], [214, 124], [72, 124]],
  'HRZ6DK8337',
  0.4664633224303828)]
```

Рисунок 3.5 – скріншот результату роботи програми на основі пакету EasyOCR

EasyOCR дійсно є зручною бібліотекою, яка підтримує різні мови та дає гнучкість у налаштуванні розпізнавання.

Прогнозування результатів OCR залежить не лише від обраної моделі, а й від ряду інших факторів, таких як чіткість зображення, відтінки сірого, налаштування гіперпараметрів, вагові коефіцієнти тощо:

- Tesseract - найкраще працює із зображеннями високої роздільної здатності. Додаткове застосування морфологічних операцій (наприклад, дилатації, ерозії, бінаризації OTSU) може значно покращити точність розпізнавання;

- EasyOCR є полегшеною моделлю, що демонструє відмінну продуктивність при обробці структурованих документів, таких як PDF-файли, квитанції, рахунки тощо;

- Keras-OCR - найбільш ефективний при розпізнаванні тексту, вбудованого в зображення. Він добре справляється із завданням, якщо шрифти та кольори тексту неорганізовані та варіативні.

Незважаючи на відсутність жорстких правил при виборі OCR-інструмента, ці характеристики можуть допомогти у виборі найоптимальнішого рішення залежно від конкретного випадку.

3.4 Біблаотека EasyOCR

Бібліотека EasyOCR ефективно розпізнає короткі тексти, такі як серійні номери, номери деталей, дати тощо.

Вона використовує попередньо визначені файли шрифтів (OCR-A, OCR-B

та інші) у поєднанні з алгоритмом зіставлення шаблонів, що дозволяє розпізнавати навіть погано надруковані, зламані або з'єднані символи незалежно від їх розміру.

Процес розпізнавання символів проходить чотири основні етапи (рис.3.6):

- попередня обробка зображення - бінаризація, нормалізація контрасту;
- сегментація тексту - розділення символів і рядків;
- класифікація символів - розпізнавання за шаблонами або нейронною мережею;
- постобробка - усунення шумів, виправлення помилок.

Ця багаторівнева система дозволяє отримати високоточні результати навіть у складних випадках.



Рисунок 3.6 – Чотири етапи на шляху до потрібного результату

На схемі робочого процесу EasyOCR (рис.3.7) ілюструються основні етапи розпізнавання тексту, які виконує бібліотека [13]:

а) обробка вхідного зображення:

- 1) завантаження зображення;
- 2) попередня обробка (зміна розміру, нормалізація, підвищення контрасту);

б) сегментація тексту:

- 1) виявлення текстових областей;
- 2) визначення меж слів або символів;

в) розпізнавання символів:

- 1) класифікація кожного символу за допомогою нейронної мережі;
- 2) аналіз контексту для підвищення точності;

г) постобробка та вивід результату:

- 1) корекція помилок;
- 2) форматування тексту;
- 3) збереження або передача результатів.

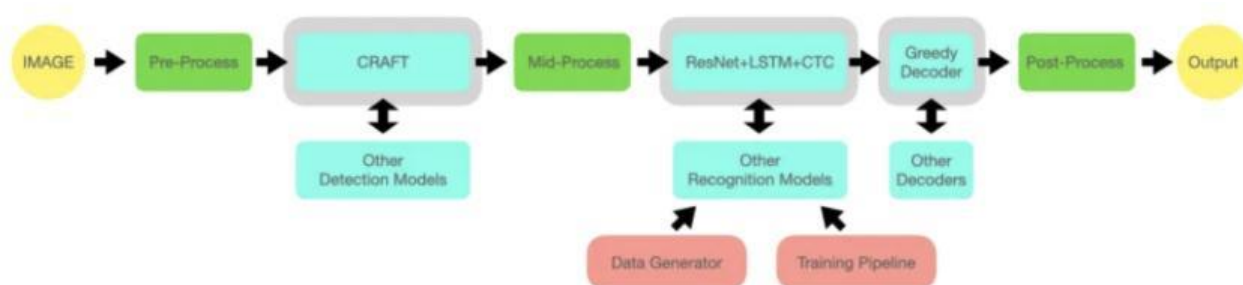


Рисунок 3.7 – Робочий процес бібліотеки EasyOCR

Ця покрокова обробка дозволяє ефективно розпізнавати текст навіть у складних умовах, таких як погана якість друку, нестандартні шрифти або зашумлені зображення.

Нижче представлена спроба детально описати методи, функції та процеси, які виконує бібліотека, а також пояснити принцип її роботи, спираючись на рисунок 3.7.

3.4.1 Процес навчання

За необхідності можна навчити бібліотеку розпізнавати символи для створення файлу шрифту.

EasyOCR аналізує символи по одному, формуючи базу даних, яка називається шрифтом. Кожен символ отримує числовий код (зазвичай ASCII) і відноситься до певного класу символів, який використовується під час розпізнавання.

Файл шрифту формується в кілька етапів:

- NewFont - очищає поточний шрифт;
- LearnPattern або LearnPatterns - додає шаблони символів із вхідного зображення до шрифту;
- FindAllChars - визначає індекси символів і впорядковує шаблони;
- збереження шаблонів - кожен символ представлений у вигляді

невеликого масиву пікселів (за замовчуванням 5×9 пікселів), але цей розмір можна змінити за допомогою параметрів `PatternWidth` і `PatternHeight` перед навчанням;

- `RemovePattern` - видаляє небажані шаблони (за потреби);
- `Save` - зберігає шрифт у файл разом із такими параметрами, як `NoiseArea`, `MaxCharWidth`, `MaxCharHeight`, `MinCharWidth`, `MinCharHeight`, `CharSpacing`, `TextColor`.

Таким чином, бібліотека створює унікальний набір шрифтів, що використовується для точного розпізнавання тексту.

3.4.2 Сегментація

EasyOCR аналізує окремі елементи зображення (так звані "краплі"), щоб визначити символи та їхні обмежувальні рамки. Для цього використовуються два режими сегментації:

- режим збереження об'єктів – кожна окрема крапля відповідає одному символу;
- режим «переклеювання» об'єктів – кілька крапель об'єднуються в символи стандартного розміру. Це особливо корисно, якщо символи розірвані або складаються з кількох частин.

Якщо двійковий об'єкт занадто великий, щоб вважатися одним символом, його можна автоматично розділити за допомогою функції `CutLargeChars` (рис.3.8). На представленому рисунку можна бачити, що фільтри видаляють дуже великі та дуже маленькі небажані функції.

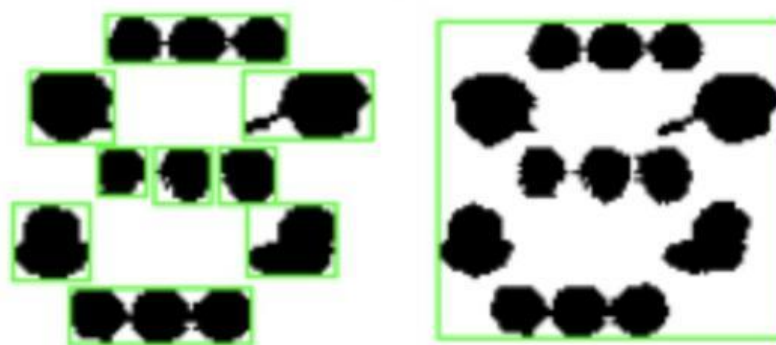


Рисунок 3.8 – Розділення за допомогою функції `CutLargeChars`

EasyOCR обробляє зображення символу, приводячи його до розміру, що відповідає обмежувальній рамці, витягує необхідні ознаки та зберігає їх у файлі шрифту. Шаблони в шрифті зберігаються у вигляді масивів пікселів, чий розмір визначається параметрами `PatternWidth` та `PatternHeight` (за замовчуванням 5 пікселів у ширину та 9 пікселів у висоту).

Параметри сегментації повинні бути однаковими як під час навчання, так і під час розпізнавання. Якісна сегментація значно підвищує точність розпізнавання.

Параметр `Threshold` використовується для відділення тексту від фону. Якщо значення порогу надто велике, чорні символи на білому фоні можуть стати надто густими, що спричинить їх злиття, а якщо значення порогу занадто мале - деталі можуть зникнути.

Якщо освітлення змінюється, найкращим варіантом буде автоматичне встановлення порогу (рис.3.9). Занадто високе граничне значення (ліворуч); регулювання порога (посередині); занадто низьке граничне значення (праворуч).

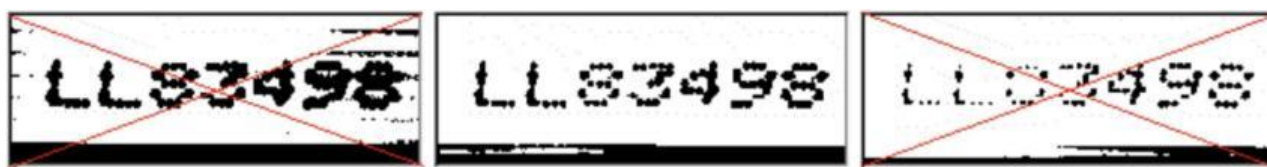


Рисунок 3.9 – Автоматичне встановлення порогу

`NoiseArea`. Це параметр для визначення області великих двійкових об'єктів. Якщо об'єкт має розмір менший за це значення, він відкидається. Це допомагає зберігати дрібні деталі символів, наприклад, крапки над літерами, як у «і».

`MaxCharWidth`, `MaxCharHeight`. Ці параметри задають максимальний розмір символу. Якщо об'єкт перевищує ці розміри, він буде відкинутий або поділений на кілька частин за допомогою вертикальних ліній розрізу. Якщо кілька великих двійкових об'єктів поміщуються в прямокутник з такими розмірами, вони об'єднуються.

`MinCharWidth`, `MinCharHeight`. Це мінімальні розміри для символу. Якщо

об'єкт або група об'єктів мають розміри менші за ці параметри, вони будуть відкинуті.

CharSpacing. Цей параметр визначає мінімальну ширину проміжку між сусідніми символами. Якщо проміжок більший за **MaxCharWidth**, він ігнорується. Якщо проміжок між символами перевищує це значення, вони вважаються різними символами, що запобігає неправильному об'єднанню тонких символів.

RemoveBorder. Це параметр, який не дозволяє використовувати об'єкти, що розташовані біля країв зображення або області інтересу для розпізнавання символів. За замовчуванням такі об'єкти відкидаються.

3.4.3 Процес розпізнавання символів

Символи порівнюються з набором шаблонів, що зберігаються в шрифтах. Кожен символ знаходить найкраще співвідношення з шаблоном, і після цього його розмір нормалізується для порівняння. Після нормалізації символ порівнюється з усіма нормалізованими шаблонами з бази даних шрифтів і повертаються найкращі збіги [14]. А саме:

- **Load** - завантажує раніше збережений шрифт із файлу на диску;
- **BuildObjects** - на цьому етапі зображення розбивається на об'єкти або краплі (з'єднані компоненти), що допомагають ідентифікувати символи. Якщо точне розташування символів відоме, цей крок можна пропустити, а місця символів можна вказати через **AddChar** і **EmptyChars**;
- **FindAllChars** - вибирає об'єкти, які вважаються символами, і сортує їх у порядку зверху вниз, зліва направо;
- **ReadText** - виконує зіставлення символів і фільтрує їх, враховуючи надані маркери або фільтри наборів символів.

Розпізнавання символів. Кожен символ порівнюється з набором шаблонів, і при знаходженні найкращого збігу, його розмір нормалізується, щоб він вмістився в заданий прямокутник. Далі символ порівнюється з кожним шаблоном з бази даних шрифтів.

Фільтрація набору символів може підвищити точність розпізнавання і швидкість обробки, обмежуючи діапазон порівнюваних символів. Наприклад,

якщо маркування завжди складається з двох великих літер і п'яти цифр, де остання завжди парна, можна призначити кожному символу клас (максимум 32 класи) і фільтрувати символи лише за заданими класами, наприклад, для двох великих літер і чотирьох цифр.

Параметри для розпізнавання [14]:

- `MaxCharWidth`, `MaxCharHeight` - визначають максимальний розмір символу (якщо двійковий об'єкт не поміщається в прямокутник цих розмірів, він вважається занадто великим і відкидається, якщо прямокутник містить кілька об'єктів, вони об'єднуються в один символ);

- `MinCharWidth`, `MinCharHeight` - визначають мінімальний розмір символу (якщо об'єкт або група об'єктів менше цих розмірів, вони вважаються занадто маленькими і відкидаються);

- `RemoveNarrowOrFlat` - маленькі символи відкидаються, якщо вони вузькі або плоскі;

- `CharSpacing` - визначає мінімальну ширину проміжку між символами (якщо проміжок між символами більший за це значення, вони вважаються різними символами);

- `CutLargeChars` - дозволяє розділяти великі двійкові об'єкти на кілька частин, якщо вони перевищують розмір `MaxCharWidth`, простір між частинами визначається параметром `RelativeSpacing`;

- `RelativeSpacing` - дозволяє вказати, скільки пробілів має бути вставлено між розділеними частинами великих двійкових об'єктів при використанні `CutLargeChars`.

Ці параметри дозволяють точніше налаштувати процес розпізнавання і адаптувати його до конкретних умов та вимог (рис.3.10).

Параметри розпізнавання можна налаштувати для досягнення оптимальної точності та продуктивності (табл..3.1):

а) `CompareAspectRatio`:

- 1) увімкнення цього параметра дозволяє EasyOCR звертати увагу на співвідношення сторін символів. Це особливо корисно для покращення

розпізнавання, коли символи виглядають однаково після нормалізації розміру, оскільки цей режим підсилює відмінності між вузькими та широкими символами, що дозволяє точніше визначити їхні форми;

2) фільтрування символів (у методі ReadText):

- використовується, якщо структура маркування є фіксованою. Це дозволяє значно підвищити точність розпізнавання, обмежуючи діапазон символів, які можуть бути розпізнані;

3) розподіл великих об'єктів:

- якщо двійковий об'єкт перевищує максимальний розмір, заданий параметром MaxCharWidth, він може бути поділений на кілька частин за допомогою вертикальних ліній розрізу. Це дозволяє обробляти символи, які є надто великими для одного обмежувального прямокутника, шляхом їх коректного розділення на частини;

б) ESegmentationMode (режим сегментації) - визначає спосіб ізоляції символів у зображенні:

1) збереження режиму об'єктів:

- кожен символ вважається окремим об'єктом (краплею). Не робиться спроб об'єднати великі двійкові об'єкти, що означає, що пошкоджені символи не будуть оброблені, а дрібні елементи, такі як акценти та точки, можуть бути відкинуті, якщо їхній розмір є меншим за мінімально допустимий розмір символу;

2) режим повторної вставки об'єктів:

- кілька крапель можуть бути об'єднані в один символ, якщо їхній розмір не перевищує максимальні розміри символів і вони не розділені вертикальним проміжком. Це дозволяє зберегти навіть маленькі елементи, такі як акценти або точки над літерами, що особливо корисно для мов із численними діакритичними знаками.



Рисунок 3.10 – Неправильні налаштування розпізнавання

Таблиця 3.1 - Порівняльна таблиця OCR-бібліотек

| Критерій | EasyOCR | Tesseract OCR | PaddleOCR | Google Cloud OCR | Amazon Textract | Azure OCR | Keras-OCR |
|---------------------------------|---------------------------------|----------------------|---------------------|------------------|-----------------|--------------|-------------------|
| Підтримка багатьох мов | 80+ мов (включно з українською) | 100+ мов | 80+ мов | 50+ мов | Багато мов | Багато мов | Обмежена |
| Підтримка рукописного тексту | Частково підтримується | ✗ | ✓ | ✓ | ✓ | ✓ | Частково |
| Працює офлайн | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ |
| Простота використання | Легка інтеграція | Складна конфігурація | Потрібні залежності | Простий API | Простий API | Простий API | Базовий інтерфейс |
| Безкоштовність | Повністю безкоштовна | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ |
| Якість розпізнавання тексту | Висока | Середня | Висока | Висока | Висока | Висока | Середня |
| Підтримка нестандартних шрифтів | ✓ | Низька | ✓ | ✓ | ✓ | ✓ | Часткова |
| Швидкість роботи | Висока | Помірна | Залежить від моделі | Висока | Висока | Висока | Середня |
| Гнучкість налаштувань | Можна кастомізувати | ✓ | ✓ | ✗ | ✗ | ✗ | ✓ |
| Ліцензія | MIT | Apache 2.0 | Apache 2.0 | Пропрієтарна | Пропрієтарна | Пропрієтарна | MIT |

EasyOCR - оптимальний вибір для більшості користувачів, які шукають:

- безкоштовне рішення;
- з простою інтеграцією;
- підтримкою української мови;
- можливістю роботи офлайн;
- та гнучкістю для налаштувань.

Вона поєднує високу точність, швидкість і доступність, не вимагаючи складної конфігурації чи хмарної інфраструктури. Це робить її особливо зручною для дослідницьких, освітніх та комерційних проєктів, де потрібне швидке та ефективне розпізнавання тексту.

Якщо потрібно, можу також скласти графічну діаграму для візуального порівняння.

Ці параметри дозволяють адаптувати розпізнавання під конкретні умови та типи зображень, оптимізуючи точність та зменшуючи кількість помилок при обробці різноманітних текстів.

3.5 Висновки до розділу

У цьому розділі було детально досліджено бібліотеку EasyOCR, зокрема її основний функціонал, принципи роботи та специфіку застосування. З результатів дослідження, представлених у пункті 3.3, можна зробити висновок, що алгоритм, який використовується у даній бібліотеці, є типовим для більшості сучасних інструментів, що реалізують функцію оптичного розпізнавання символів. Однак, кожен з таких інструментів має свої специфічні риси, що залежить від технічної реалізації, набору можливостей і здатності вирішувати конкретні задачі з точністю та швидкістю.

З'ясувалося, що не всі доступні бібліотеки OCR здатні ефективно виконувати завдання, що ставляться перед ними. У цьому контексті виникла

необхідність у порівняльному аналізі кількох популярних рішень, щоб вибрати найбільш відповідне для вирішення проблеми, пов'язаної з розпізнаванням тексту на зображеннях. У процесі цього аналізу були оцінені різні фактори, такі як точність розпізнавання, швидкість обробки, можливості налаштування та обробка різних типів зображень.

Таким чином, проведене дослідження допомогло не лише зрозуміти особливості роботи EasyOCR, але й отримати необхідні знання для оптимального вибору бібліотеки, яка дозволить максимально ефективно вирішити задачу оптичного розпізнавання тексту у конкретних умовах.

4 РЕАЛІЗАЦІЯ СИСТЕМИ

4.1 Програмне забезпечення EasyOCR

Щоб використовувати бібліотеку, її спочатку необхідно імпортувати за допомогою команди:

```
```python
import easyocr
```
```

Потім слід вказати EasyOCR, якою мовою потрібно здійснювати розпізнавання тексту (рис.4.1).

EasyOCR підтримує розпізнавання кількох мов одночасно, проте вони повинні бути сумісними між собою. Англійська мова є універсальною та поєднується з усіма іншими. Також добре сумісні мови, що використовують спільний набір символів, наприклад, різні варіанти латинського алфавіту. Розділ повинен містити рекомендації або власне бачення щодо архітектурних рішень розроблюваного ПЗ, виходячи з результатів моделювання ПЗ та проведеного аналізу побудови сучасних типових архітектур. Процес проектування має завершуватися побудовою діаграми розміщення (Deployment Diagram), або інших рисунків, що визначенні у завданні до проєкту.



Рисунок 4.1 – Фотографія, яку буде протестовано

Після імпортування EasyOCR перевіряє, чи наявні необхідні файли моделі у користувача. Якщо їх немає, бібліотека автоматично завантажує потрібні моделі. Після цього вони завантажуються в оперативну пам'ять, що може зайняти кілька секунд залежно від продуктивності обладнання. Після початкового завантаження можна обробляти будь-яку кількість зображень без повторного виконання цього кроку.

Окрім списку мовних кодів, EasyOCR дозволяє передавати додаткові параметри.

- `gpu` – за замовчуванням встановлено `'True'`, що означає використання графічного процесора (GPU) для обчислень, якщо це можливо. Якщо потрібно вимкнути GPU, достатньо передати `'gpu=False'`;

- вибір конкретного GPU – якщо у системі кілька графічних процесорів, можна вибрати потрібний, наприклад, `'gpu=cuda:0'`;

- `model_storage_directory` – дозволяє задати шлях до каталогу, де EasyOCR зберігає моделі. Якщо цей параметр не вказано, використовується стандартний каталог `~/EasyOCR/model`.

Щоб витягнути текст із зображення, необхідно передати шлях до файлу у функцію `readtext` наступним чином:

```
```python
result = reader.readtext('chinese_tra.jpg')
```
```

Результат роботи (рис.4.2) має формат списку, де кожен елемент є підсписком, що містить:

- координати прямокутної області тексту `[[x1, y1], [x2, y2], ...]`;
- розпізнаний текст;

- рівень достовірності розпізнавання.

```
Text: IND, Probability: 0.9986064134764059
Text: cG 07 0 5599, Probability: 0.5573922112991998
```

Рисунок 4.2 – Результат зчитування

Результат упорядкований зверху донизу. Як видно, це може не відповідати природному людському читанню, але є можливість автоматично комбінувати ці слова в кінці цього уроку.

Замість шляху до файлу `chinese_tra.jpg` передано зображення у вигляді масиву `pumpy` (з `opencv`):

```
img = cv2.imread('chinese_tra.jpg') result = reader.readtext(img),
```

або URL-адреси зображення:

```
result = reader.readtext('https://www.somewebsite.com/chinese_tra.jpg').
```

Також нижче є приклад із використанням хінді (рис.4.3). У цій фотографії, як описувалося раніше, дані можна виводити у форму шарів зображень поверх вихідного, що може бути дуже зручно для відео реєстраторів:



Рисунок 4.3 – Демонстрація здатності пакету до розпізнавання

EasyOCR підтримує широкий спектр мов, включаючи як популярні, так і менш поширені, зокрема азійські та східні мови. Навчання моделі здійснюється на основі мовних баз даних, які зберігаються у форматі `.pth`. Кожен із цих файлів має значний розмір (понад 200 МБ), що свідчить про високу якість та точність попереднього навчання бібліотеки.

Коли створюється об'єкт класу `Reader`, відбувається ініціалізація нейронної мережі, і вона навчається відповідно до вибраної користувачем мовної моделі. Параметр, що передається під час оголошення цього об'єкта, визначає, яку саме мовну базу буде використовувати бібліотека для розпізнавання тексту.

Нижче на рисунку 4.4 наведено список деяких мовних моделей, які підтримує EasyOCR, серед них є українська:

Supported Languages

| Language | Code Name |
|-------------|-----------|
| Abaza | abq |
| Adyghe | ady |
| Afrikaans | af |
| Angika | ang |
| Arabic | ar |
| Assamese | as |
| Avar | ava |
| Azerbaijani | az |

Рисунок 4.4 – Мови, які підтримує EasyOCR

Під час створення об'єкта класу `Reader` відбувається повна ініціалізація нейронної мережі, яка лежить в основі процесу оптичного або мовного розпізнавання. Цей об'єкт є основним інтерфейсом для взаємодії з бібліотекою, яка реалізує механізм розпізнавання мовлення або тексту, зокрема в таких бібліотеках як `easyOCR`, `whisper`, `speechRecognition` та подібних.

У процесі ініціалізації відбувається завантаження необхідної мовної моделі, яка забезпечує розпізнавання мовного сигналу або візуального образу тексту, залежно від типу вхідних даних. Вибір конкретної моделі здійснюється на основі параметра, що передається в конструкторі при створенні об'єкта.

4.2 Програмна реалізація системи

Код у EasyOCR є досить компактним і не займає багато місця, що робить його зручним у використанні. Завдяки цьому бібліотека легко інтегрується в різні проєкти без зайвого навантаження на систему.

Представлена програмна реалізація ефективно вирішує завдання в межах роботи та забезпечує роботу нейромережевої системи для автоматичного розпізнавання документів і введення даних.

Нижче представлено середовище розробки (рис.4.5), у якому можна переглянути робочий процес та структуру реалізації проєкту:

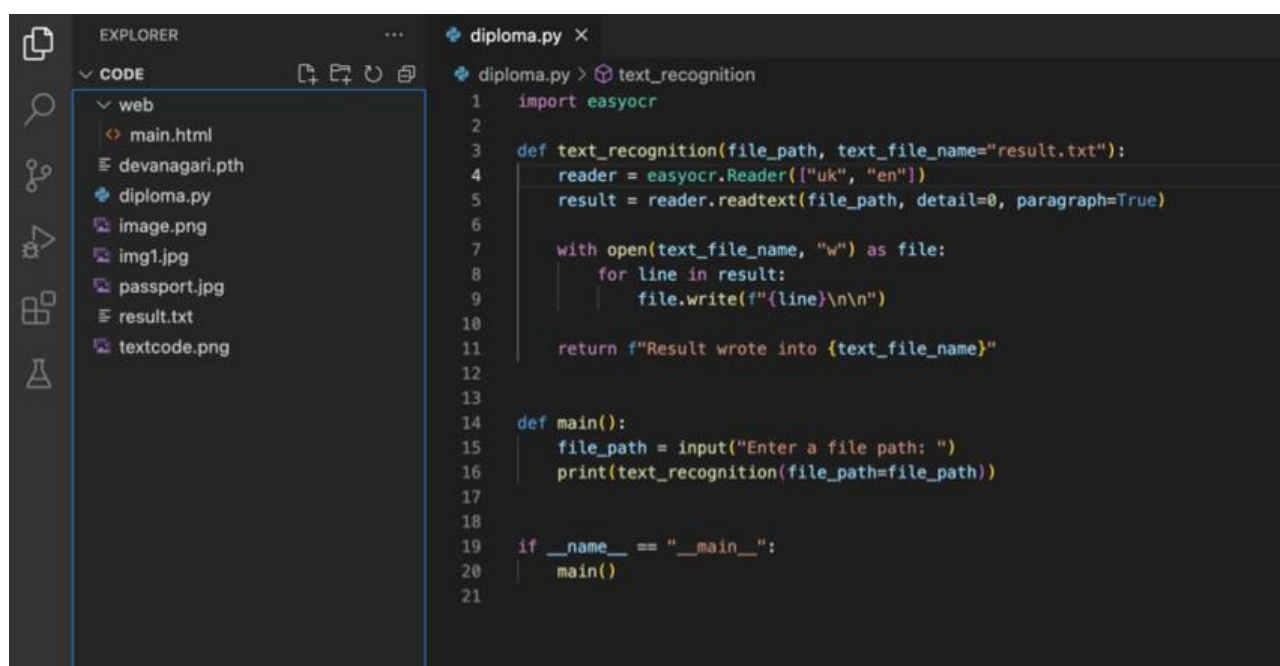


Рисунок 4.5 – Робоче середовище

Програма містить лише одну функцію під назвою `text\_recognition`, яка відповідає за розпізнавання тексту на зображеннях.

Фрагмент коду (лістинг 4.1) нижче демонструє збереження отриманого результату у текстовий файл:

Лістинг 4.1 – Доступ до файлу

```
```python
with open(text_file_name, "w") as file:
 for line in result:
 file.write(f"{line}\n\n")
```
```

У цьому блоці коду використовується цикл для запису зчитаної інформації зі змінної `result` у текстовий файл `result.txt`. Цей файл можна використовувати для подальшої обробки залежно від потреб користувача.

Нижче наведено приклад зображення паспорта автора роботи (рис.4.6), який програма повинна обробити та витягти з нього текстову інформацію (рис.4.7).



Рисунок 4.6 – Зображення паспорта у кореневій папці програми

Як видно на рисунку 4.7, користувачеві необхідно ввести назву файлу зображення, після чого програма розпочне обробку.

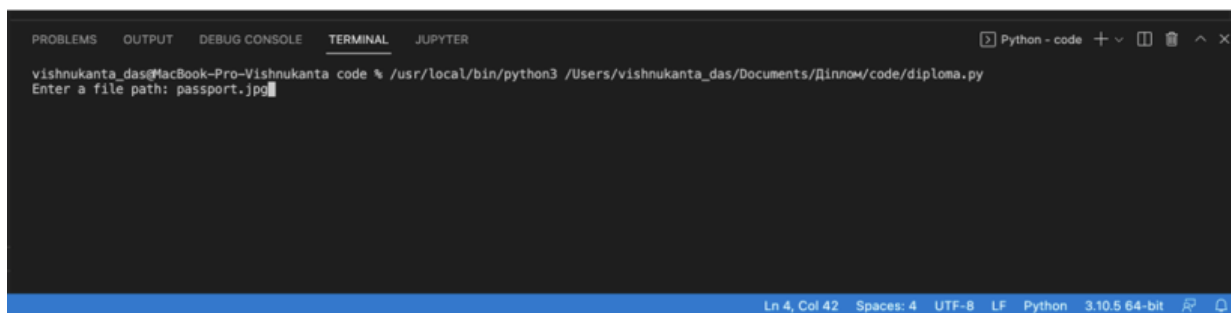


Рисунок 4.7 – Зображення паспорту у кореневій папці програми

Після завершення виконання програми у консолі з'явиться повідомлення:

```

...
CUDA not available - defaulting to CPU. Note: This module is
much faster with a GPU
Result wrote into result.txt
...

```

Це означає, що якщо графічний процесор (GPU) недоступний, програма автоматично переключається на використання центрального процесора (CPU), хоча обробка на GPU значно швидша.

Далі можна відкрити файл `result.txt` (рис.4.8) і оцінити результати розпізнавання тексту, перевіривши, наскільки коректно програма зчитала інформацію із зображення.

Рисунок 4.8 – Фрагмент файлу `result.txt`

Оцінюючи роботу програми, можна відзначити, що, незважаючи на неідеальну якість вхідного зображення, зробленого звичайним смартфоном із середньою якістю камери, отриманий результат є досить точним і наближеним до ідеального.

Однак під час аналізу розпізнаного тексту були виявлені деякі недоліки:

- некоректне визначення регістру символів у розпізнаному тексті;

- неправильне трактування символу "/" (слеш), який програма іноді сприймає як англійську літеру "l".

Попри ці дрібні похибки, результати розпізнавання залишаються якісними, і програму можна вважати ефективною для роботи з документами. За необхідності точність можна покращити шляхом додаткового навчання моделі або застосування методів постобробки тексту.

4.3 Висновки до розділу

Підсумовуючи, можна зробити висновок, що бібліотека EasyOCR є досить простою у використанні та реалізації. Лише кількома рядками коду можна налаштувати нейронну мережу для розпізнавання тексту, просто підключивши відповідну базу даних мов. У даному випадку використовувалися українська та англійська мови, що особливо актуально при роботі з паспортами та іншими офіційними документами.

Окрім друкованих текстів, ця бібліотека здатна розпізнавати й рукописні записи, хоча точність такого розпізнавання значною мірою залежить від розбірливості почерку.

Тестування показало, що навіть за умови низької якості вхідного зображення нейронна мережа успішно впоралася із завданням. Виявлені похибки були незначними: некоректне розпізнавання регістру символів і помилкове трактування знака «/» як літери «l».

Основним напрямком для подальшого розвитку є створення зручного десктопного програмного забезпечення. Це дозволить значно покращити користувацький досвід і знизити можливі технічні труднощі під час експлуатації продукту.

ВИСНОВКИ

У процесі виконання роботи бакалавра була реалізована система для оптичного розпізнавання даних. Проведено порівняльний аналіз безкоштовних пакетів з відкритим вихідним кодом, в результаті чого було підтверджено, що бібліотека EasyOCR відповідає вимогам потенційних користувачів, для яких розроблений продукт. Незважаючи на низьку якість фотографії, програма успішно здійснила зчитування тексту з фото паспорта.

У першому розділі було підкреслено важливість проблематики, яка обговорюється, оскільки вона є надзвичайно актуальною в умовах ери комп'ютеризації всіх аспектів життя та діяльності. Безсумнівно, подібні продукти будуть продовжувати розвиватися, а штучний інтелект ставати дедалі більш досконалим, поступово наближаючись до рівня людської розумової діяльності.

Другий розділ був присвячений опису нейронних мереж та штучного інтелекту, зокрема їхнього потенціалу для впливу на різні сфери життя. В цьому контексті було акцентовано увагу на технологіях, таких як EasyOCR та OpenCV, які доступні кожному користувачеві. Завдяки цим технологіям можна реалізувати багато інноваційних ідей, які ще знаходяться на етапі впровадження в повсякденне життя.

У третьому розділі було з'ясовано, що не всі відкриті бібліотеки здатні ефективно виконати поставлені завдання. Тому було проведено детальний порівняльний аналіз, щоб вибрати найбільш підходящу бібліотеку для вирішення проблеми роботи бакалавра.

Четвертий розділ був присвячений розгляду принципів роботи бібліотеки EasyOCR та демонстрації її використання для виконання завдання БКР. Основним напрямом для подальшої роботи є адаптація інтерфейсу майбутнього десктопного додатку відповідно до потреб користувачів та замовників.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. OCR-технології для розпізнавання паперових документів. URL: <https://sites.google.com/view/informaticdnzhvpku> (дата звернення 21.03.2025).
2. Роль OCR у оцифровці документів. URL: <https://uk.shaip.com/blog/ocr-in-document-digitization/#:~:text=> (дата звернення 18.03.2025).
3. OCR (оптичне розпізнавання символів) – визначення, переваги, проблеми та випадки використання. URL: <https://uk.shaip.com/blog/ocr-definition-benefits-challenges-and-use-cases> (дата звернення 24.03.2025).
4. Підручник з OpenCV: посібник із вивчення OpenCV у Python. URL: <http://platoaistream.com/uk/дані-платона/підручник-з-opencv-посібник-із-вивчення-opencv-у-python/> (дата звернення 11.03.2025).
5. EasyOCR2 Industrial optical character recognition library. URL: <http://www.euresys.com/en/products/software/easyocr2/> (Last accessed: 01.04.2025).
6. Оптичне розпізнавання символів. URL: http://uk.wikipedia.org/wiki/Оптичне_розпізнавання_символів (дата звернення 27.03.2025).
7. Оптичне програмне забезпечення для розпізнавання символів. URL: <http://www.machinetranslation.com/uk/blog/ocr-technology> (дата звернення 31.05.2025).
8. Michael Nielsen, Neural Networks and Deep Learning. URL: <http://neuralnetworksanddeeplearning.com/index.html> (Last accessed: 07.04.2025)/
9. Tesseract wrappers URL: <http://tesseract-ocr.github.io/tessdoc/AddOns.html#tesseract-wrappers> (Last accessed: 24.03.2025).
10. Вступ до OpenCV. Комп'ютерний зір. URL: <http://itmaster.biz.ua/programming/vision/opencv.html> (дата звернення 04.04.2025).
11. Інструменти розпізнавання тексту та ефективність бізнесу. URL: <http://www.ranktracker.com/uk/blog/7-best-ocr-tools-and-applications-to-boost-business-efficiency/> (дата звернення 28.03.2025).
12. Keras - відкрита нейромережна бібліотека, написана мовою Python.

URL: <http://www.uk.wikipedia.org/wiki/Keras> (дата звернення 19.04.2025).

13. Optical Character Recognition using EasyOCR in Python. URL: <https://github.com/JaidedAI/EasyOCR> (Last accessed: 24.03/2025).

14. EasyOCR - Reading Texts. URL: http://documentation.euresys.com/Products/Open_eVision/Open_eVision_2_5/en-us/Content/03_Using_Open_eVision/D1_EasyOCR_-_Reading_Texts/EasyOCR_-_Reading_Texts.htm (Last accessed: 24.04.2025).

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»
Кафедра комп'ютерних систем та мереж

ДИПЛОМНИЙ ПРОЄКТ БАКАЛАВРА

НА ТЕМУ: «Розробка комп'ютерної системи для розпізнавання документів з допомогою бібліотеки EasyOCR»

Ст. групи КНТ - 511
Ляховський М.В.

1

Мета проєкту – створення нейромережевої системи для підвищення ефективності розпізнавання документів та автоматичного введення даних.

Для досягнення поставленої мети буде використано:

- знання про технології OCR;
- бібліотеки, що працюють з нейронними мережами для пошуку та розпізнавання символів на зображеннях;
- знання мови програмування Python;
- допоміжні навички роботи з Web-бібліотеками, такими як Django тощо.

2

Для досягнення цієї мети будуть використані:

- знання про хоча б одну систему OCR;
- бібліотеки, що використовують нейронні мережі для пошуку та розпізнавання символів на зображеннях;
- орієнтація на програмування мовою Python.

Для виконання завдання необхідно визначити вимоги до проекту, а саме:

- призначення - програма, яка буде використовуватися для зчитування тексту з фотографій або інших зображень, що містять текст;
- кордон проєкту - зчитування тексту з фотографії та його виведення у вигляді рядкових змінних;
- класи користувачів - користувачами можуть бути як фізичні, так і юридичні особи;
- основний користувач - працівники соціальної сфери;
- вимоги до даних - зображення повинні бути в одному з форматів (png, jpg).

3

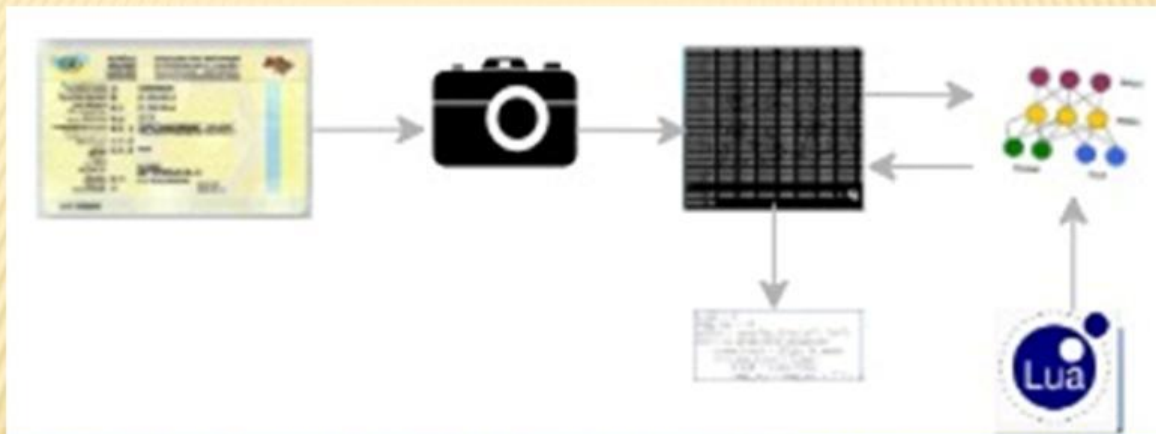


Схема передачі даних для обробки

4

Операційна система:

- Windows 10 / 11 (64-bit);
- Linux (Ubuntu 18.04 або новіше);
- macOS 10.15+;
- підтримка Python у середовищі WSL або Conda у разі Windows.

Мова програмування - Python 3.8

Для встановлення EasyOCR необхідні компоненти:

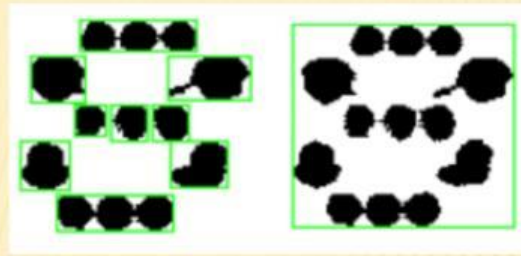
- torch (PyTorch);
- torchvision;
- easyocr;
- opencv-python (для роботи із зображеннями);
- Pillow, numpy, scipy - для обробки зображень і матричних операцій.

5

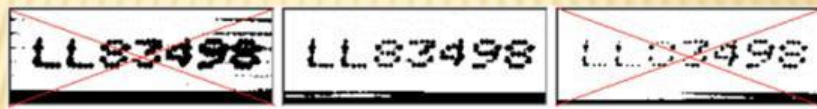
Мінімальні системні вимоги (для CPU-режиму):

- процесор - Intel Core i3 або AMD Ryzen 3;
- оперативна пам'ять (RAM) - від 4 ГБ;
- місце на диску - від 1 ГБ для бібліотек і моделей;
- графіка - інтегрована графіка або базова дискретна карта .

6

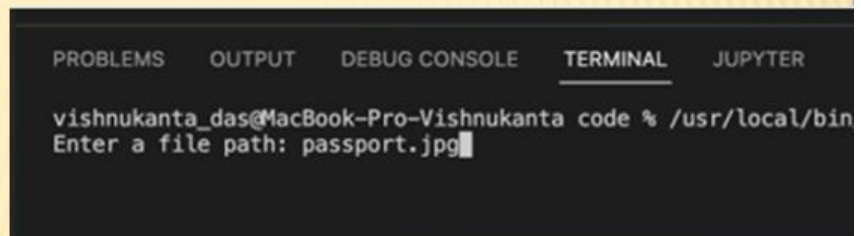


Розділення великого об'єкту

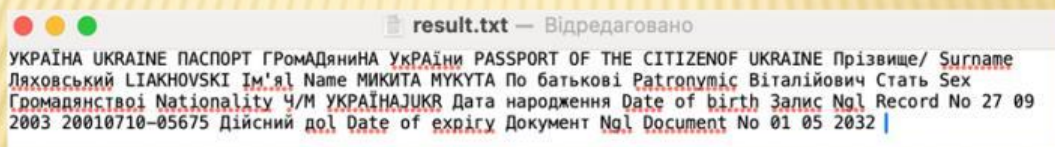


Встановлення порогу

7



Відображення паспорту та назва зображення у кореневій папці програми



Текстовий файл result.txt

8

ВИСНОВКИ

У процесі виконання роботи бакалавра була реалізована система для оптичного розпізнавання даних. Проведено порівняльний аналіз безкоштовних пакетів з відкритим вихідним кодом, в результаті чого було підтверджено, що бібліотека EasyOCR відповідає вимогам потенційних користувачів, для яких розроблений продукт. Незважаючи на низьку якість фотографії, програма успішно здійснила зчитування тексту з фото паспорта.

9

ДЯКУЮ ЗА УВАГУ !

10