МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
МИКОЛАЇВСЬКИЙ НАЦІОНАЛЬНИЙ АГРАРНИЙ УНІВЕРСИТЕТ

Факультет менеджменту

Кафедра економічної кібернетики, комп'ютерних наук та інформаційних технологій

Кваліфікаційна наукова
праця на правах рукопису

БАЛЮК Ян Олександрович

УДК 004.8:004.93'1:003.75

КВАЛІФІКАЦІЙНА РОБОТА

РОЗРОБКА СИСТЕМИ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

Спеціальність 122 «Комп'ютерні науки»
Галузь знань – 12 «Інформаційні технології»

Подається на здобуття освітнього ступеня «Бакалавр»

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

_____ Я.О. Балюк

Науковий керівник: Волосюк Юрій Вікторович, кандидат технічних наук, доцент _____

Науковий керівник: Жебко Олександр Олегович, асистент _____

Завідувач кафедри: Тищенко Світлана Іванівна, кандидат педагогічних наук, доцент _____

Миколаїв – 2025

ЗМІСТ

АНОТАЦІЯ

Балюк Я.О. *Розробка системи розпізнавання рукописного тексту з використанням машинного навчання* – Кваліфікаційна наукова праця на правах рукопису. Робота на здобуття освітнього ступеня бакалавра за спеціальністю 122 «Комп'ютерні науки». – Миколаївський національний аграрний університет, Миколаїв, 2025.

Кваліфікаційна робота присвячена розробці системи оптичного розпізнавання рукописного тексту на основі методів машинного навчання. В умовах активної цифровізації документообігу актуальним є створення ефективних інструментів для автоматизованого перетворення зображень тексту у редагований формат. Метою роботи є розробка системи для розпізнавання рукописного тексту на основі алгоритмів машинного навчання, яка забезпечує точне і швидке розпізнавання зображень з текстом. У роботі досліджено сучасні алгоритми OCR, зокрема шаблонні методи, статистичні класифікатори, згорткові нейронні мережі (CNN) та трансформери. Обґрунтовано вибір архітектури на базі CNN для ефективного розпізнавання символів англійського алфавіту.

Для навчання моделі використано датасет EMNIST, що містить зображення цифр та букв у різних варіаціях. У рамках роботи було реалізовано попередню обробку зображень, сегментацію символів, масштабування та подання їх у формат, придатний для класифікації нейронною мережею. Результатом стало створення системи, здатної точно визначати символи з вхідного зображення, формувати з них текстовий рядок та адаптуватися до умов слабкої якості зображень.

Практичне значення роботи полягає у можливості впровадження запропонованого підходу в мобільні додатки, сканери, документообіг, освітні та медичні системи, а також у перспективі – його масштабування для багатомовного розпізнавання.

Ключові слова: нейронна мережа, набір даних, оптичне розпізна тексту, TensorFlow, згортка

ABSTRACT

Balyuk Ya.O. Development of a handwritten text recognition system using machine learning – Qualification scientific work in the form of a manuscript. Work for the degree of bachelor in specialty 122 "Computer Science". – Mykolaiv National Agrarian University, Mykolaiv, 2025.

The qualification work is devoted to the development of a system of optical recognition of handwritten text based on machine learning methods. In the conditions of active digitalization of document flow, the creation of effective tools for automated conversion of text images into an editable format is relevant. The purpose of the work is to develop a system for handwritten text recognition based on machine learning algorithms, which provides accurate and fast recognition of images with text. The work investigates modern OCR algorithms, in particular template methods, statistical classifiers, convolutional neural networks (CNN) and transformers. The choice of architecture based on CNN for effective recognition of English alphabet characters is justified.

To train the model, the EMNIST dataset was used, which contains images of numbers and letters in various variations. As part of the work, pre-processing of images, character segmentation, scaling and presentation in a format suitable for classification by a neural network were implemented. The result was the creation of a system that can accurately identify characters from the input image, form a text string from them and adapt to conditions of poor image quality.

The practical significance of the work lies in the possibility of implementing the proposed approach in mobile applications, scanners, document management, educational and medical systems, and in the future - its scaling for multilingual recognition.

Keywords: neural network, dataset, optical text recognition, TensorFlow, convolution

ВСТУП

У сучасному світі значна частина важливої інформації залишається у вигляді рукописних записів – від архівних документів до особистих нотаток. Оцифрування такого тексту є актуальним завданням, яке набуває особливого значення у сферах науки, освіти, медицини та бізнесу.

Розпізнавання рукописного тексту є суттєво складнішим за обробку друкованого через індивідуальність почерку, різні розміри написаних букв, нахил літер та нерівномірність написання. Традиційні методи розпізнавання, які ґрунтуються на заздалегідь визначених шаблонах і правилах, досить обмежені у гнучкості та точності. Особливо це відчутно при роботі з нестандартними почерками або текстами, написаними у складних умовах.

Саме тому на передній план виходять методи машинного навчання, які дозволяють системам самостійно вивчати закономірності з великої кількості прикладів і пристосовуватися до нових варіацій написання без необхідності ручної корекції. Розробка системи розпізнавання рукописного тексту передбачає кілька ключових етапів: попередню обробку зображень, виділення характерних ознак і навчання моделей машинного навчання.

Реалізувати алгоритм розпізнавання тексту можливо з допомогою штучних нейронних мереж. Нейронна мережа – математична модель, а також її програмне або апаратне втілення, побудоване за принципом організації та функціонування біологічних нейронних мереж. В результаті ми отримаємо продукт, що імітує людське мислення.

Актуальність роботи полягає в розробці нового алгоритму розпізнавання рукописних текстів, який виправить недоліки існуючих систем, а також скоротить обсяг використовуваних ресурсів і витраченого часу.

Практична цінність кваліфікаційної роботи полягає в необхідності удосконалення методів розпізнавання тексту, у зменшенні кількості витрачених

на це ресурсів. Можливість автоматизувати величезну кількість ручної монотонного праці несе неабияку практичну цінність.

Метою кваліфікаційної роботи є розробка ефективної нейронної мережі для автоматичного розпізнавання рукописного тексту на зображеннях, яка забезпечує високу точність і стабільність розпізнавання в умовах різноманітних стилів письма, якості зображень. Реалізація такої системи передбачає застосування методів глибокого навчання, зокрема згорткових нейронних мереж (CNN) для обробки вхідних графічних даних та перетворення їх у структуровану текстову інформацію. Робота спрямована на досягнення практичної придатності розробленого рішення для подальшого використання в документаційних системах, освітніх застосунках, архівах, сервісах цифрового документообігу тощо.

Для досягнення поставленої мети необхідно виконати такі **завдання**:

1. Дослідити існуючі нейронні мережі.
2. Вибрати нейронну мережу для створення моделі та дослідити її будову.
3. Розробити алгоритм розпізнавання рукописного тексту на зображенні.
4. Реалізувати обрану нейронну мережу.
5. Навчити обрану нейронну мережу.
6. Інтерпретувати вхідні дані для подачі їх в модель нейронної мережі.
7. Перевірити алгоритм на прикладах реальних текстових зображень.

Об'єктом роботи є методи та процеси функціонування нейронної мережі для розпізнавання рукописного тексту.

Предметом роботи є згорткова нейронна мережа для розпізнавання рукописного тексту на зображенні.

Методи дослідження. У процесі виконання кваліфікаційної роботи застосовувалися такі методи дослідження: аналіз літературних джерел – для вивчення сучасних підходів і технологій у сфері розпізнавання рукописного тексту та машинного навчання; методи машинного навчання – зокрема технології комп’ютерного зору та згорткові нейронні мережі (CNN) для обробки зображень; метод моделювання – створення архітектури системи

розпізнавання тексту на основі штучних нейронних мереж; практичне програмування – для реалізації прототипу системи на основі бібліотек Python: TensorFlow, Keras, OpenCV тощо.

Практичне значення роботи. Практична значущість роботи полягає у створенні функціонального прототипу системи розпізнавання рукописного тексту, який може бути використаний у різних галузях: оцифрування архівів і рукописних документів (історичних, освітніх, наукових); автоматизація введення даних з паперових анкет, заяв, бланків; освітні застосунки, що допомагають учням перевіряти рукописні домашні завдання або використовуються для тренування каліграфії; інтеграція з мобільними застосунками для розпізнавання нотаток, чеків або елементів рукописного введення. Розроблена модель може бути адаптована до різних мов та стилів письма, а також масштабована для комерційного або open-source використання.

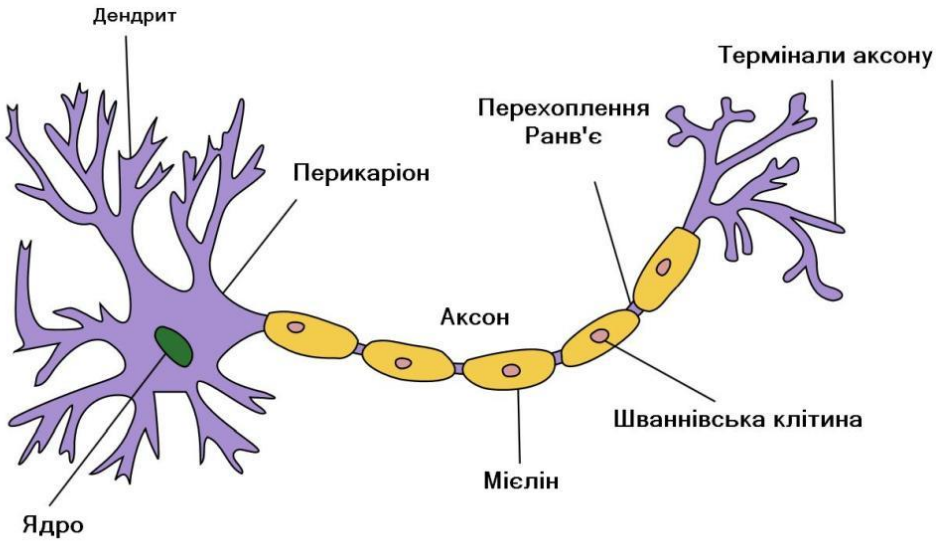
Структура та обсяг роботи. Кваліфікаційна робота складається зі вступу, трьох розділів, висновків і списку використаних джерел. Загальний обсяг – 62 сторінок, включає 2 таблиць, 43 рисунків. Список використаних джерел налічує 33 найменувань.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ НЕЙРОННИХ МЕРЕЖ ТА РОЗПІЗНАВАННЯ ТЕКСТУ ЗА ЇХ ДОПОМОГОЮ

1.1 Біологічне поняття нейрона і штучні нейронні мережі

У людському мозку знаходяться мільярди нейронів – вузько спеціалізованих клітин, призначених для прийому ззовні, обробки, зберігання, передачі та виведення назовні інформації за допомогою електричних і хімічних сигналів [1].

Будова нейрона дуже проста (див. рис 1.1). Нейрон складається з тіла клітини, дендритів і аксона. Тіло клітини складається з протоплазми, яка обмежена зовні мембраною з ліпідного шару. Ліпіди складаються з гідрофільних головок і гідрофобних хвостів. Ліпіди розташовуються гідрофобними хвостами один до одного, утворюючи гідрофобний шар. Також тіло нейрона містить відростки, звані аксонами і дендритами. Аксонами є довгі відростки нейрона, які пристосовані для передачі інформації від тіла нейрона до нейрона, або від нейрона до виконавчого органу. Дендритами є короткі і сильно розгалужені відростки нейрона, головним завданням яких є утворення збудливих і гальмівних синапсів, що впливають на нейрон, тобто місць



контакту між двома нейронами [2].

Рисунок 1.1 – Будова нейрона

Джерело: сформовано автором на основі [2]

На підставі кількості та розташування елементів нейрона вчені класифікували їх на кілька видів за структурною та функціональною ознакою. За структурною ознакою нейрони поділяються на [1]:

1. Безаксонні нейрони - невеликі клітини, що згруповані поблизу спинного мозку в міжхребцевих гангліях, які не мають анатомічних ознак поділу відростків на дендрити і аксони. Всі відростки у клітини дуже схожі. Функціональне призначення безаксонних нейронів слабо вивчено.

2. Уніполярні нейрони - нейрони з одним відростком, присутні, наприклад в сенсорному ядрі трійчастого нерва в середньому мозку.

3. Біполярні нейрони - нейрони, що мають один аксон і один дендрит, розташовані в спеціалізованих сенсорних органах - сітківці ока, нюховому епітелії і цибулині, слуховому і вестибулярному гангліях;

4. Мультиполярні нейрони - нейрони з одним аксоном і кількома дендритами. Даний вид нервових клітин переважає в центральній нервовій системі

5. Псевдо уніполярні нейрони - є унікальними в своєму роді. Від тіла відходить один відросток, який відразу Т-образно ділиться. Цей єдиний тракт структурно являє собою аксон, хоча по одній з гілок збудження йде не від, а до тіла нейрона. Структурно дендритами є розгалуження на кінці цього (периферичного) відростка. Критичної зоною є початок цього розгалуження (знаходиться поза тілом клітини). Такі нейрони зустрічаються в спінальних гангліях.

За функціональною ознакою нейрони поділяються на [1]:

- Аферентні – даний тип нейронів можна віднести до сенсорних нейронів. До даного типу нейронів відносяться первинні клітини органів чуття.

- Еферентні – такий тип нейронів можна назвати руховим. До нейронів цього типу відносяться нейронів кінцівок.

- Асоціативні – даний тип нейронів здійснює зв'язок між еферентними нейронами і аферентними.

- Секреторні нейрони, що секретують високоактивні речовини, у яких добре розвинений комплекс Гольджі.

На основі нейронів людського мозку була складена штучна нейронна мережа. Штучна нейронна мережа – це математична модель, в тому числі її програмна реалізація, побудована за принципом організації та функціонування біологічних нейронних мереж живого організму. Вона може бути реалізована у вигляді програмного забезпечення та використовується для обробки, аналізу й навчання на основі даних. Головним поняттям штучних нейронних мереж є штучний нейрон, який у своєму роді є суматором всіх вхідних у нього сигналів [3]. Будова штучного нейрона показано на рисунку 1.2.

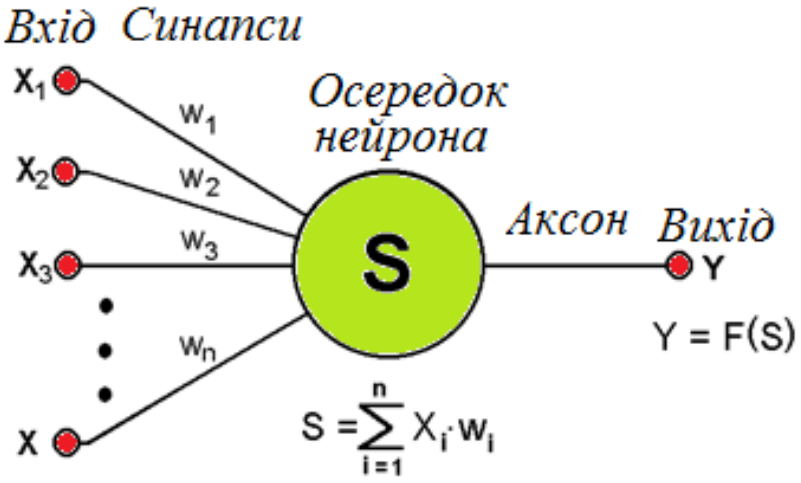


Рисунок 1.2 – Будова штучного нейрона

Джерело: сформовано автором на основі [3]

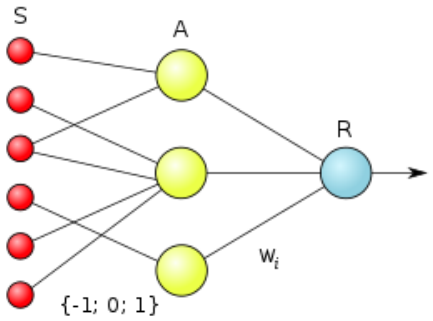
На вхід штучного нейрона подається будь-який набір даних, позначений на малюнку як $x_1, x_2, \dots, x_n$. Далі вхідні дані перетворюються з допомогою ваг, позначених на малюнку як $w_1, w_2, \dots, w_n$ і передаються в суматор, позначену на малюнку літерою S. Формула суматора представлена формулою 1 [2]:

$$u = \sum_{i=1}^{n} w_0 * x_0 + w_i * x_i \qquad (1)$$

де, u – значення суматора, w – вага, x – вхідні дані. Далі значення суматора передається передавальній функції або по іншому функції активації *F(u)*. Дані функції мають різні формули і саме від них залежить вихідне значення нейрона [4].

1.2 Види нейронних мереж та їх застосування

Базовими архітектурами нейронних мереж є мережі прямого поширення і перцептрони (див. рис. 1.3). Інформація по ним передається від входу до виходу. Клітини шару даних мереж не пов'язані між собою, на відміну від сусідніх шарів, які зазвичай повністю пов'язані. Нейронні мережі прямого поширення зазвичай навчаються за методом зворотного поширення помилки.

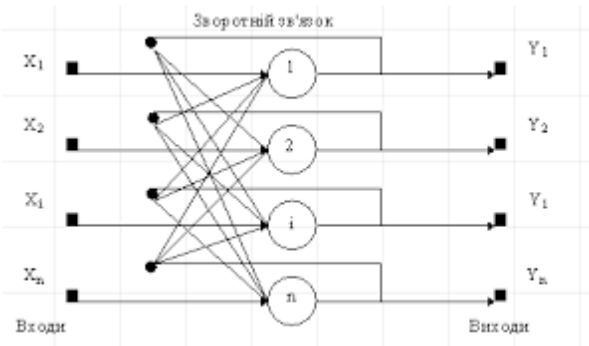


При такому навчанні нейронна мережа отримує безліч даних як на вхід, так і на вихід [5] .

Рисунок 1.3 – Приклад будови перцептрона

Джерело: сформовано автором на основі [5]

Ще одним різновидом базових нейронних мереж є нейронна мережа Хопфілда. Ця нейронна мережа використовується для відновлення зображень. Нейронна мережа Хопфілда (див. рис. 1.4) є повнозв’язною нейронної мережею з симетричною матрицею зв’язків. Поняття повнозв’язна мережа означає, що кожен нейрон передає свій вихідний сигнал іншим нейронам, включаючи самого себе. Кількість нейронів даної мережі визначається кількістю входів і виходів. Також нейронна мережа Хопфілда навчається тільки з вчителем, так як для відновлення зображення їй необхідно знати вихідні значення зображення,



яке необхідно відновити [6].

Рисунок 1.4 – Приклад побудови нейронної мережі Хопфілда

Джерело: сформовано автором на основі [6]

Ще одним видом нейронної мережі є Машина Больцмана (див. рис. 1.5). Частіше всього дану нейронну мережу розглядають як стохастичний генеративний варіант нейронної мережі Хопфілда. Машина Больцмана використовується для навчання алгоритму імітації відпалу. Також дана нейронна мережа виявилася першою нейронною мережею, здатної навчатися внутрішнім поданням і вирішувати складні комбінаторні задачі [7, с.47].

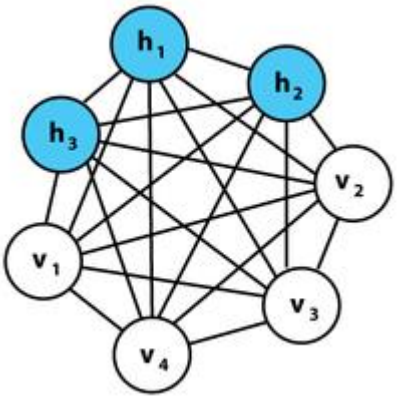
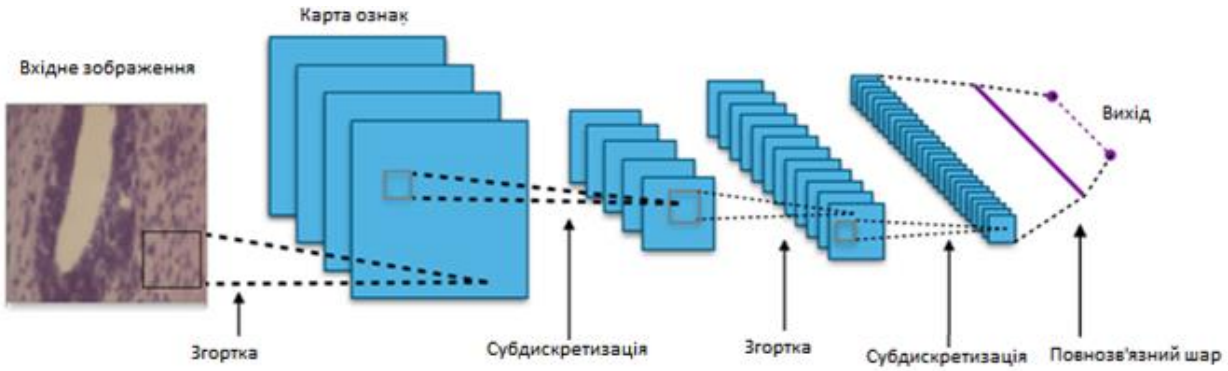


Рисунок 1.5 – Приклад графічного подання машини Больцмана

Джерело: сформовано автором на основі [3]

У прикладі на рисунку 1.5 ми маємо 3 прихованих і 4 видимих нейрона.

Також існують згорткові нейронні мережі. Даний вид нейронних мереж сильно відрізняється від інших. Згорткові нейронні мережі зазвичай використовують для класифікації зображень. Головною особливістю даної нейронної мережі є «сканер», який зчитує зображення шматками, і процес

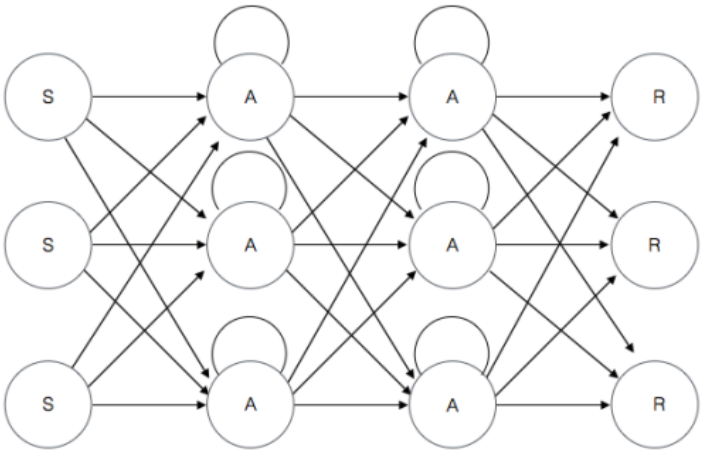


«згортки», який зменшує розмір матриці ознаки зображення [8].

Рисунок 1.6 – Приклад будови згорткової нейронної мережі

Джерело: сформовано автором на основі [8]

До більш просунутих конфігурацій нейронних мереж відносяться рекурентні нейронні мережі (див. рис. 1.7). Особливістю цих нейронних мереж є те, що вони отримують інформацію не тільки з попереднього шару, але також і від самих себе з попереднього проходу. Через цю особливість стає важливим порядок подання даних для навчання нейронної мережі. Складністю у використанні нейронних мереж такого типу є проблема зникаючого градієнта, яка визначається швидкою втратою інформації з плином часу. Ця проблема впливає тільки на коефіцієнти ваги при навчанні моделі. Найчастіше нейронні мережі такого типу використовують для автоматичного доповнення



недостатньої інформації. [9]

Рисунок 1.7 – Приклад рекурентної нейронної мережі

Джерело: сформовано автором на основі [9]

Нейронна мережа Кохонена застосовує змагальне навчання для класифікації даних без вчителя. Для даної нейронної мережі подаються вхідні дані, які нейронна мережа сама розподіляє по нейронах, які збігаються з ними в більшій мірі. Після чого нейрони змінюються для більшої точності збігу, рухаючи своїх сусідів [6].

В результаті розгляду поняття нейрона було вивчено будову і функціонал біологічного нейрона, який складається з тіла клітки, дендритів і аксонів і служить для прийому, обробки і передачі інформації. Також було розглянуто будову штучного нейрона, який є математичною моделлю біологічного нейрона і виконує схожі функції. Крім розгляд штучного нейрона в окремо, були розглянуті види нейронних мереж – сукупності штучних нейронів. Кожна штучна нейронна мережа, описана в цьому розділі відрізняється будовою, так і сферою застосування.

1.3 Огляд основних методів обробки зображень

Обробка зображень є однією з фундаментальних дисциплін у сфері комп’ютерного зору та машинного навчання. Вона охоплює широкий спектр методів, спрямованих на покращення, трансформацію або аналіз зображень для подальшої інтерпретації або автоматизованої обробки. Кожен етап обробки має свої цілі, серед яких виділення важливої інформації, усунення шумів, підвищення якості або спрощення структури зображення. Першим і базовим етапом часто виступає попередня обробка, яка передбачає фільтрацію шуму.

У реальних умовах зображення нечасто є ідеальними: вони можуть містити різноманітні перешкоди у вигляді випадкових точок, спотворень чи змазаності. Для видалення шуму застосовуються такі методи, як середнє згладжування, розмиття за Гауссом або медіанне фільтрування. Наприклад, медіанний фільтр дозволяє зберегти краї об'єктів, водночас видаляючи дрібні спотворення, що особливо важливо для завдань, пов'язаних із розпізнаванням контурів. Однією з найважливіших операцій є перетворення зображення у градації сірого.

Перехід від кольорового простору до монохромного значно спрощує подальші обчислення, оскільки замість трьох каналів (RGB) опрацьовується лише один. Після цього часто застосовується бінаризація – процедура перетворення зображення у чорно-білий формат за певним порогом яскравості. Бінаризація є ключовою для завдань, де потрібно виділити об'єкти з фону,

наприклад у розпізнаванні тексту. Для виявлення меж об'єктів у зображенні використовують алгоритми виявлення контурів. Одним із класичних підходів є оператор Кенні (див. рис. 1.8), який базується на пошуку різких змін яскравості у локальних ділянках зображення. Визначення контурів дозволяє звести складну сцену до набору основних ліній і форм, що особливо корисно для аналізу структури об'єктів або підготовки зображень до подальшої класифікації.

Рисунок 1.8 - Результат роботи алгоритму виявлення контурів в OpenCV

Джерело: сформовано автором на основі [15]

На фото зліва направо: вихідне зображення, вихід фільтра Canny з низькими порогами, вихід фільтра Canny з високими порогами. Пороги фільтра Canny можна налаштувати, щоб уловлювати лише найсильніші краї та отримувати більш чіткі контури. Відповідно, чим вищі пороги, тим чистіші краї. [7].

Ще одним важливим методом є морфологічна обробка зображень. Вона передбачає використання операцій на основі геометричних структур: ерозії, дилатації, відкриття та закриття. Ці операції змінюють форму об'єктів на бінаризованих зображеннях.

Наприклад, ерозія дозволяє "стискати" об'єкти, видаляючи дрібні деталі, тоді як дилатація "розширює" об'єкти, заповнюючи прогалини. Морфологічна обробка часто використовується для очищення зображень після бінаризації або для об'єднання розірваних компонентів.

Сегментація зображення є основою в обробці. Вона полягає у розділенні зображення на значущі області, кожна з яких відповідає певному об'єкту або частині сцени. Існують різні підходи до сегментації, починаючи від простого розділення за рівнем яскравості до складних методів, таких як кластеризація пікселів (метод k-середніх) чи сегментація за допомогою графових алгоритмів. Сегментація є критично важливою для таких завдань, як розпізнавання об’єктів, відстеження руху або медична діагностика за допомогою аналізу знімків [26].

Значну роль відіграють також методи геометричних перетворень. Вони дозволяють змінювати положення, розмір, нахил чи орієнтацію зображення. Найпоширенішими перетвореннями є масштабування, обертання, зсув і перспектива. Геометричні перетворення часто використовуються для нормалізації вхідних даних перед подачею в систему розпізнавання або для вирівнювання об'єктів у кадрі. Фільтрація частотного простору є іншим потужним методом обробки. За допомогою перетворення Фур'є зображення переводиться в частотний домен, де високочастотні компоненти відповідають дрібним деталям, а низькочастотні – великим структурам. Це дозволяє виділяти або приглушувати певні елементи зображення. Наприклад, за допомогою високочастотної фільтрації можна підкреслити краї, тоді як низькочастотна допомагає зменшити шум і зробити зображення "плавнішим" [17].

На сьогодні важко уявити обробку зображень без використання алгоритмів машинного навчання. Сучасні методи, такі як згорткові нейронні мережі (CNN), здатні автоматично витягувати релевантні ознаки з зображення, замінюючи собою багато класичних етапів обробки. Проте навіть у контексті глибокого навчання базові методи фільтрації, сегментації та нормалізації залишаються важливими складовими попередньої обробки даних [4].

Отже, обробка зображень складається з багатьох етапів, кожен з яких спрямований на покращення вхідних даних, виділення структурних елементів або спрощення подальшого аналізу. Від простого шумозаглушення до складної сегментації та адаптивних перетворень – усі ці методи у сукупності створюють основу для точного та ефективного комп'ютерного бачення.

1.4 Розпізнавання тексту: традиційні методи та їхні обмеження

Початок розвитку у 1950-1960 роках вперше проводилися експерименти технологій по розпізнаванню деяких символів за допомогою електронних значень. Здебільшого вони спрямовувалися для допомоги незрячими людині, і це робило озвучення текстів необхідним. Сьогодні це можна вважати першою реальною історією розпізнавання текстів. У 1970-1980 роках виникли перші комерційні системи оптичного розпізнавання текстів по растровому шрифті. Ранні алгоритми обробки були базовані на простому vertil. Після 1990 року розвиток методів машинного навчання дозволив підвищити точність розпізнавання та виникнення перші алгоритми сегментації тексту [13].

До появи глибоких нейронних мереж і сучасних підходів машинного навчання, розпізнавання тексту здебільшого ґрунтувалося на класичних алгоритмах обробки зображень та евристичних методах. Ці підходи були досить ефективними у випадках чіткого друкованого тексту, однак виявлялися обмеженими або малоефективними при роботі з рукописами, які характеризуються великою варіативністю стилів, нахилів, товщини ліній, злиттям букв тощо [22].

Одним із найпоширеніших підходів до розпізнавання тексту в традиційних системах була оптична система розпізнавання символів – Optical Character Recognition (OCR). Ця технологія заснована на аналізі вхідного зображення тексту, сегментації його на окремі символи, а потім порівнянні кожного символу з набором еталонів або шаблонів. Зазвичай такі OCR-системи працювали добре з чіткими сканами друкованого тексту, особливо з відомими шрифтами. Однак уже на етапі розпізнавання різних шрифтів або нерівномірного розміщення символів починалися труднощі [21].

Для реалізації традиційного OCR-заснованого розпізнавання часто використовували методи попередньої обробки зображення, включаючи бінаризацію (перетворення в чорно-білий формат), фільтрацію шуму, нормалізацію розміру символів та сегментацію. Останній етап, тобто поділ

тексту на рядки, слова й окремі символи, був особливо критичним і водночас складним у випадку рукописного тексту. Рукописні символи часто з'єднані між собою або взагалі не мають чітких меж, що робить сегментацію неточною або навіть неможливою [9].

Ще одним традиційним підходом був статистичний аналіз. Наприклад, використовували методи на основі частотного аналізу форм букв, гістограм напрямків штрихів, або розподілу пікселів у межах символу. Такі методи дозволяли системі не тільки порівнювати символи з шаблонами, але й формувати узагальнене уявлення про те, як має виглядати певна буква. Проте всі ці підходи потребували ручної інженерії ознак, тобто розробник мав самостійно визначати, які саме особливості зображення є релевантними для класифікації символів. Це обмежувало здатність системи до узагальнення, а також робило її вразливою до змін у стилі письма [5].

Ще однією суттєвою проблемою традиційних методів було те, що вони не мали здатності до контекстного аналізу. Розпізнавання відбувалося на рівні окремих символів без врахування того, як ці символи поєднуються в слова, і як слова – у речення. Як наслідок, навіть при правильному розпізнанні окремих букв, результати могли бути синтаксично чи лексично некоректними. У друкованих текстах це частково компенсувалося словниками, які підключалися до OCR-систем, однак для рукописного тексту, де можливі помилки в написанні або нестандартні слова, це не завжди спрацьовувало [32].

Традиційні підходи також не могли ефективно працювати з багатомовними текстами або текстами, які містять змішані системи письма. Наприклад, у випадку українсько-англійських рукописних заміток або формул із текстом. Системи, орієнтовані на одну мову або один набір символів, не здатні були розпізнати текст поза межами своєї спеціалізації. Щодо обмежень апаратного рівня, то традиційні системи OCR зазвичай вимагали добре відсканованих або сфотографованих документів з високою роздільною здатністю. Будь-яке викривлення, зсув, або навіть тінь від руки на зображенні могли суттєво знизити точність розпізнавання. У контексті мобільних

пристроїв, де умови зйомки не завжди ідеальні, такі системи показували себе недостатньо стабільно [8].

Підсумовуючи, можна сказати, що традиційні методи розпізнавання тексту були важливим кроком у розвитку комп'ютерного зору, однак їхні можливості виявилися обмеженими у контексті більш складних завдань, як-от розпізнавання рукописного тексту. Їхня головна слабкість – це залежність від шаблонів, необхідність ручної інженерії ознак, складність у роботі зі змінною структурою тексту та повна відсутність гнучкості при зміні умов розпізнавання. Саме ці обмеження стали поштовхом до розвитку машинного навчання, а згодом і глибокого навчання, які дозволяють автоматично вивчати релевантні ознаки та враховувати контекст, що значно підвищує точність і універсальність систем розпізнавання рукописів [4].

За останні кілька десятиліть OCR зазнає популярності. У 2000-х та 2010-х роках достатні великі цифрові дзеркальні камери і сканери зробили їх доступними для масового використання. Також почали використовуватися алгоритми попередньої обробки зображень, щоб зменшити шум та викривлення. На сьогоднішній день глибоке навчання й архітектури такі як згорткові, рекурентні мережі, трансформери сильно покращили точність OCR [16].

Наведемо кілька прикладів програмного забезпечення для оптичного розпізнавання символів (OCR), які особливо корисні для перекладацької галузі, кожен з яких оснащений для вирішення різноманітних мовних проблем [16]:

1. ABBYY FineReader: Відомий своїми високими показниками точності, ABBYY FineReader підтримує понад 190 мов і особливо ефективний для розпізнавання текстів кількома мовами із сканованих документів та зображень. Він широко використовується для перекладу офіційних документів і вилучення з них тексту для подальшої обробки.

2. Adobe Acrobat Pro DC: Хоча в основному це інструмент PDF, Adobe Acrobat Pro також має потужні можливості розпізнавання, які можуть конвертувати відскановані документи у файли, які можна редагувати та

зберігати у потрібному форматі. Він підтримує численні мови та сценарії, що робить його цінним інструментом для перекладачів, які працюють з багатомовними документами.

3. OmniPage Ultimate: Це вдосконалене програмне забезпечення OCR пропонує широку підтримку мов, включаючи складні сценарії, такі як арабська та азіатська мови. OmniPage призначений для обробки великих обсягів документів і часто використовується професійними бюро перекладів для швидкої та точної обробки різних типів контенту.

4. Google Cloud Vision API: Це більш технологічне рішення, яке використовує машинне навчання для покращення результатів розпізнавання з часом. Він може виявляти та перекладати текст у зображеннях більш ніж на 50 мовах, і це особливо корисно для розробників та підприємств, яким потрібно інтегрувати можливості OCR у свої програми.

5. Tesseract OCR: Рушій OCR з відкритим кодом, Tesseract має багато налаштувань у та підтримує широкий спектр мов, включаючи сценарії, такі як деванагарі та кирилиця. Він популярний серед розробників і технічно підкованих перекладачів, яким комфортно програмувати і вони хочуть інтегрувати конкретні завдання OCR у свої робочі процеси.

1.5 Машинне навчання та його застосування для обробки зображень

Глибоке навчання базується на штучних нейронних мережах з великою кількістю прихованих шарів, які дозволяють моделювати складні залежності у даних. На відміну від традиційних підходів машинного навчання, де ознаки для моделі часто задаються вручну, глибокі нейронні мережі здатні самостійно виявляти релевантні ознаки з вхідних даних, зокрема зображень, аудіо- або текстових сигналів.

Основною структурною одиницею глибокого навчання є штучний нейрон – математична модель, що імітує функціонування біологічного нейрона. Кожен нейрон приймає на вхід одне або кілька чисел (ознак), виконує над ними зважену суму, застосовує активаційну функцію (наприклад, ReLU, сигмоїда чи

гіперболічний тангенс) і передає результат далі. Нейрони організовуються в шари: вхідний, кілька прихованих і вихідний. В процесі навчання за допомогою алгоритму зворотного поширення помилки та методу градієнтного спуску мережа оптимізує ваги з'єднань, щоб мінімізувати функцію втрат [24].

Для обробки зображень найбільш успішною архітектурою глибокого навчання стали згорткові нейронні мережі. Вони були спеціально розроблені для обробки двовимірних структур, таких як зображення, і дозволяють ефективно виявляти локальні ознаки – краї, контури, текстури тощо.

Згорткові нейронні мережі складаються з наступних типів шарів [24]:

- Згорткові шари: застосовують фільтри (ядра) до вхідного зображення, створюючи карти ознак. Кожен фільтр виявляє певну ознаку, наприклад, вертикальні лінії або градієнти.

- Шари активації: додають нелінійність до моделі. Найчастіше використовується функція ReLU, яка обнуляє всі від'ємні значення.

- Пулінгові шари: зменшують розмірність карти ознак, зберігаючи найважливішу інформацію. Найпоширенішим типом є max-pooling, який вибирає найбільше значення в заданому вікні.

- Нормалізаційні шари: стабілізують навчання, нормалізуючи вхід до кожного шару.

- Повнозв'язні шари: інтерпретують ознаки, виділені попередніми шарами, і приймають остаточне рішення (наприклад, класифікація об'єкта).

Таким чином згорткові нейронні мережі здатні поступово будувати уявлення про зображення: від простих локальних ознак до складних концептуальних структур. Це робить їх достатньо ефективними в задачах класифікації зображень, розпізнавання об'єктів, аналізу сцен, сегментації [15].

Крім CNN, у сфері обробки зображень також використовуються інші типи глибоких архітектур [13]:

- Автокодери - моделі, що навчаються стискати зображення до латентного простору та відновлювати їх. Використовуються для зменшення розмірності, очищення шумів та генерації зображень.

- Генеративні змагальні мережі - складаються з двох мереж: генератора та дискримінатора, які змагаються між собою. Застосовуються для генерації нових зображень, що виглядають реалістично.

- Рекурентні нейронні мережі - більше використовуються для обробки послідовностей, вони також можуть комбінуватися з CNN для аналізу відеопотоків або рукописного тексту, написаного рядками.

Для навчання глибоких моделей потрібна велика кількість розмічених зображень. У завданні розпізнавання рукописних символів часто використовуються відкриті набори даних, такі як MNIST або EMNIST. Останній розширює стандартний набір MNIST, включаючи малі та великі англійські літери.

Навчання моделей часто проводиться із застосуванням бібліотек, таких як TensorFlow та Keras. Вони забезпечують високорівневі інтерфейси для побудови архітектур, навчання, оцінки точності та візуалізації результатів.

Серед важливих етапів обробки зображень для глибокого навчання можна виділити [12]:

- Масштабування - приведення зображень до однакового розміру (наприклад, 28×28 пікселів для EMNIST).

- Нормалізація - перетворення значень пікселів до діапазону [0, 1] або [-1, 1], що пришвидшує навчання.

- Аугментація (Augmentation) - штучне збільшення кількості прикладів шляхом обертання, зсуву, масштабування, що дозволяє зменшити перенавчання.

Глибокі нейронні мережі в обробці зображень значно перевершують класичні методи, такі як SVM або дерева рішень, за рахунок гнучкості, здатності до узагальнення та адаптації до нових задач. Завдяки їх ефективності глибоке навчання широко застосовується не лише у розпізнаванні рукописного тексту, а й у медицині (аналіз рентгенівських знімків), автономному водінні (виявлення дорожніх об’єктів), промисловому контролі якості, доповненій реальності та ін.

Висновки до розділу 1

У першому розділі було розглянуто фундаментальні теоретичні аспекти, необхідні для реалізації системи розпізнавання рукописного тексту. Зокрема, було проаналізовано біологічну природу нейронів та її відображення у штучних нейронних мережах, що є основою сучасних алгоритмів глибокого навчання. Штучний нейрон моделює базові принципи функціонування біологічного нейрона, дозволяючи створювати складні багаторівневі структури для обробки вхідних даних.

У межах розділу були досліджені основні архітектури нейронних мереж, серед яких згорткові, рекурентні, мережі Хопфілда, Больцмана та Кохонена, кожна з яких має свої унікальні особливості та сфери застосування. Особливу увагу було приділено згортковим нейронним мережам як найбільш придатним до задач обробки зображень і розпізнавання візуальних патернів.

Також було докладно проаналізовано методи попередньої обробки зображень, необхідні для підготовки вхідних даних до подальшого аналізу. Вивчено основні підходи до фільтрації шуму, бінаризації, виявлення контурів, морфологічної трансформації, геометричних перетворень та сегментації, що дозволяють підвищити якість зображень і виділити ключові ознаки для розпізнавання.

Окрему увагу приділено традиційним методам OCR, їх сильним та слабким сторонам. Показано, що хоча класичні алгоритми OCR ефективні для друкованого тексту, вони значно поступаються сучасним методам при роботі з рукописними даними через обмеження в контекстному аналізі, чутливість до змін шрифту та потребу в ручному налаштуванні ознак.

У завершальній частині розділу проаналізовано переваги використання глибокого навчання в задачах обробки зображень. Розглянуто архітектуру згорткових нейронних мереж, особливості їх навчання, застосування технік масштабування, нормалізації, аугментації та роль сучасних бібліотек TensorFlow і Keras. Таким чином, теоретичне підґрунтя, викладене у розділі,

створює міцну основу для подальшої реалізації практичної частини роботи, пов’язаної з розпізнаванням рукописного тексту на зображеннях.

РОЗДІЛ 2. АНАЛІЗ ІСНУЮЧИХ СИСТЕМ РОЗПІЗНАВАННЯ ТЕКСТУ ТА ВИБІР ПІДХОДУ ДЛЯ СИСТЕМИ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ

2.1. Огляд сучасних алгоритмів та систем розпізнавання тексту (OCR)

OCR (Optical Character Recognition) – технологія оптичного розпізнавання символів. Зчитує текст з зображень або сканів документів у текстовий файл. Використовується для перетворення зображення тексту в редагований цифровий формат.

OCR використовується для оцифрування документів, переклад текстів, пошук по документах, автоматизація введення даних та аналіз тексту. Залежно від цілей може використовуватися програмне забезпечення, вбудоване у мобільні додатки, комп’ютерні програми або хмарні сервіси.

На сьогоднішній день існує кілька підходів до розпізнавання тексту, які базуються як на класичних алгоритмах, так і на сучасних моделях машинного навчання. Один з найстаріших методів OCR – це метод шаблонного співставлення. У цьому підході кожен символ зі зображення порівнюється з еталонними шаблонами у базі даних. Якщо знайдено достатньо подібний шаблон, система визначає символ. Такий метод працює швидко і просто, але має суттєві обмеження: він малоефективний при зміні шрифтів, розмірів чи деформаціях тексту. Особливо погано він справляється з рукописним текстом, який може бути дуже варіативним [11].

З розвитком обчислювальних потужностей і машинного навчання почали застосовуватись статистичні методи розпізнавання. Замість шаблонів, ці методи аналізують набір ознак – геометричних, топологічних або контурних – які описують символи. Далі ці ознаки обробляються класифікаторами, наприклад, методами опорних векторів або нейронними мережами. Такі системи значно гнучкіші, бо вчаться на прикладах і можуть розпізнавати текст навіть у складних умовах, наприклад, при низькій якості зображення чи нестандартному написанні символів. Справжній прорив у сфері OCR відбувся з появою глибоких нейронних мереж. Зараз найпоширенішими є методи, що використовують згорткові нейронні мережі (CNN) у поєднанні з рекурентними мережами (RNN) або трансформерами. Згорткові мережі добре працюють із зображеннями, виділяючи важливі візуальні ознаки, тоді як рекурентні мережі або трансформери враховують послідовність символів, що дуже важливо для рукописного тексту. Такий підхід дозволяє моделі "розуміти" контекст написання і коригувати розпізнавання на основі ймовірної послідовності слів [5].

Окремо варто згадати системи, які використовують end-to-end підхід. Вони навчаються розпізнавати текст прямо із зображення, минаючи проміжні кроки виділення символів. Це дозволяє уникнути помилок на ранніх етапах обробки, які можуть накопичуватись у традиційних системах. Завдяки глибокому навчанню, такі моделі можуть розпізнавати як друкований, так і рукописний текст у різних умовах – навіть коли текст частково перекритий або написаний під нахилом. Таким чином, сучасні методи OCR значно відрізняються за підходами. Шаблонні системи прості, але обмежені. Статистичні методи є більш адаптивними, але потребують точного підбору ознак. Глибокі нейронні мережі, особливо в комбінації CNN і RNN або трансформерів, стали основою для новітніх систем, які досягають високої точності розпізнавання навіть у складних випадках, таких як рукописний текст. Саме ці технології сьогодні формують основу найсучасніших рішень у сфері OCR [10].

2.2. Порівняння точності та швидкості розпізнавання різних моделей

У сучасних системах оптичного розпізнавання тексту важливо не лише досягти високої точності, а й забезпечити прийнятну швидкість обробки, особливо коли йдеться про великі обсяги тексту або застосування в реальному часі. Під час вибору моделі для побудови системи OCR завжди виникає дилема: що обрати – модель, яка забезпечує максимальну точність, чи ту, що працює швидше, але з дещо меншою точністю? У цьому підрозділі буде розглянуто, як різні типи моделей – від класичних до глибоких нейронних мереж – проявляють себе в аспектах точності та швидкодії.

Почнемо з найпростіших моделей – шаблонних. Їх головна перевага полягає у швидкості: розпізнавання відбувається майже миттєво, оскільки вся операція зводиться до пошуку найближчого відповідника серед заздалегідь заданих шаблонів. Проте така простота має і зворотний бік – шаблонні методи дуже чутливі до змін. Найменші спотворення, шум, незвичне накреслення символів або рукописний текст – усе це різко знижує точність, іноді до рівня нижче 50%. У реальних умовах такі методи практично не використовуються самостійно, особливо для рукописного введення [34].

Класичні статистичні моделі, такі як наївні байєсівські класифікатори, дерева рішень, методи опорних векторів, а також алгоритми k-найближчих сусідів, демонструють вищу точність порівняно з шаблонними. Вони здатні враховувати більше ознак і аналізувати складніші зв’язки між елементами тексту. Проте в плані швидкодії такі моделі часто поступаються, особливо коли розмір ознак або обсяг навчального набору зростає. Крім того, ефективність таких підходів залежить від того, наскільки ретельно підібрано ознаки для кожного символу. При правильному налаштуванні точність може перевищувати 85–90% для друкованого тексту, але з рукописним текстом ці методи працюють значно гірше [17].

Перехід до моделей глибокого навчання відкрив новий рівень точності розпізнавання. Згорткові нейронні мережі ефективно виявляють візуальні

ознаки символів незалежно від їх розміру, нахилу або шрифту. Рекурентні нейронні мережі, зокрема архітектури LSTM або GRU, здатні враховувати контекст у послідовності символів, що особливо важливо для розпізнавання рукописного тексту, де багато залежить від розташування літер у слові. У поєднанні ці моделі дають дуже високу точність – понад 95% на стандартних датасетах, таких як IAM або MNIST, і навіть більше у спеціалізованих задачах [17].

Щодо швидкості, то моделі зі згортковими та рекурентними нейронними мережами вимагають більше ресурсів і часу на інференс, особливо на пристроях із обмеженою обчислювальною потужністю. Наприклад, обробка одного зображення може тривати від декількох сотень мілісекунд до кількох секунд, залежно від складності архітектури. У виробничих системах це вирішується паралельною обробкою та використанням графічних процесорів (GPU), що дозволяє значно пришвидшити виконання без втрати точності [18].

Останні розробки у сфері трансформерів також показують вражаючі результати. Архітектури, подібні до Vision Transformer (ViT) або спеціалізовані моделі типу TrOCR (Transformer OCR від Microsoft), дозволяють розпізнавати текст на складних зображеннях, у тому числі рукописний, з точністю понад 97%. Вони мають перевагу в тому, що здатні краще моделювати залежності між усіма частинами зображення одночасно, а не лише в лінійному порядку, як це роблять RNN. Проте трансформери є ще вимогливішими до ресурсів, особливо пам’яті, і працюють повільніше на менш потужних пристроях без оптимізації [4].

Таблиця 1.1 - Порівняльна характеристика основних моделей OCR

Тип моделі	Приклад моделі	Середня точність	Швидкість обробки	Здатність розпізнати рукописний текст
Шаблонне розпізнавання	Просте порівняння шаблонів	50-70%	Дуже висока	Низька
Статистичні	SVM, KNN,	85-90%	Середня	Обмежена

методи	Decision Trees			
Глибокі нейромережі	CRNN, DeepOCR	95-97%	Нижче середньої	Висока
Трансформери	TrOCR, Donut	96-99%	Низька	Досить висока

Джерело: сформовано автором на основі аналізу алгоритмів [13,15,27]

Швидкість може покращуватись з використанням GPU або оптимізованих моделей (наприклад, TensorRT, ONNX). У трансформерів високі вимоги до ресурсів, особливо пам’яті, але можлива оптимізація (Quantization, Distillation).

Таким чином, порівнюючи різні моделі OCR, можна сказати, що точність прямо пропорційна складності моделі. Шаблонні підходи найшвидші, але найбільш ненадійні. Статистичні методи забезпечують кращу якість, але їх ефективність обмежена. Глибокі нейронні мережі, особливо CNN та трансформери, демонструють найвищу точність, однак потребують більше обчислювальних ресурсів. Вибір оптимального рішення залежить від конкретного завдання: якщо важлива швидкість і простота – варто застосовувати легші моделі; якщо головне – точність розпізнавання, особливо рукописного тексту, то найкращим вибором буде поєднання CNN і RNN або трансформерів, можливо з використанням оптимізованих варіантів для мобільних або вбудованих пристроїв.

2.3. Вибір методології та алгоритмів для системи розпізнавання рукописного тексту

Під час розробки системи розпізнавання рукописного тексту головним завданням було знайти баланс між низкою критичних факторів: ефективністю алгоритму, точністю відтворення символів, ресурсними витратами та можливістю масштабування на різні типи даних. Обраний підхід ґрунтується на методах глибокого навчання, насамперед на архітектурі згорткових нейронних мереж (CNN). Розглянемо докладніше обґрунтування такого вибору, порівняння із альтернативними підходами, а також практичні аспекти реалізації.

CNN дозволяють автоматично виявляти ієрархічні ознаки прямо із сирих піксельних даних. Завдяки послідовному накладанню кількох згорткових шарів, мережа поступово переходить від виявлення базових патернів – прямих, кутів, градієнтів – до складних структур, що характеризують різні символи. Самостійне розпізнавання ознак значно спрощує попередню обробку, скорочує людську участь в інженерії ознак і дозволяє моделі працювати з різнорідними даними реального світу. Це особливо важливо, тому що рукописний текст характеризується невеликими варіаціями форми символів, нерівномірним освітленням, шумами, змінами кута накреслення тощо – всі ці фактори ускладнюють застосування класичних підходів [21].

Увага до згорткових мереж має історичні підстави: ще в 1990-х їх успішно використовували у реальних OCR-системах, зокрема в LeNet, яка стала прототипом для сучасних архітектур. Їх потужність у розпізнаванні рукописних символів доведена емпірично, особливо у проектах з MNIST, EMNIST, IAM, NIST. Саме тому у якості основної моделі обрано CNN.

Для реалізації моделі застосовано бібліотеки **TensorFlow** та **Keras**, що мають переваги у простоті опису моделей, автоматичному розрахунку градієнтів, підтримці GPU, а також багатофункціональних callback-методів. Простота API дозволяє модифікувати мережу, додавати регуляризацію, змінювати оптимізатор, без необхідності «написання рудиментарного коду». Цей аспект принципово важливий, оскільки він дозволяє швидко експериментувати зі складністю архітектури, кількістю фільтрів, глибиною мережі тощо [25].

Одним із найважливіших рішень стало використання набору даних **EMNIST ByClass**. Це розширення класичного набору MNIST, що включає 62 класи – цифри (0–9), великі літери A–Z і малі літери a–z, що повністю покриває англійський алфавіт. Його перевага – у великому обсязі (понад 800 000 зразків), стандартизованому форматі IDX і координованій розмітці. Альтернативні набори, такі як IAM або NIST, мають свої переваги – реальні рукописні дані, нестандартна орфографія, великий формат (повні рядки, тексти) – але також і

суттєві недоліки: складні формати даних (текстові рядки, документи), необхідність сегментації, ліцензійні обмеження, відсутність однакових умов з EMNIST тощо. У цьому сенсі EMNIST надає зручний компроміс між простотою застосування і репрезентативністю даних [12].

Формат даних IDX може зчитуватися через **idx2numpy**, що дозволяє імпортувати зображення та мітки без додаткового парсингу. Після зчитування дані конвертуються до формату `float32`, нормалізуються до інтервалу [0,1] – що є стандартною практикою для стабільності градієнтного спуску – і розбиваються на тренувальний та валідаційний набори. У початковій версії для прискорення тренування встановлено `k=10`, тобто використовується лише 10% даних, що забезпечує прискорений цикл розробки та дозволяє перевірити працездатність моделі. У фінальній версії слід зняти цю умову і тренуватися на повному наборі [32].

Перед подачею на CNN дані проходять обробку. Бібліотека **OpenCV** була обрана завдяки широкому функціоналу: порогова бінаризація, морфологічні операції (ерозія), детекція контурів, обчислення bounding-rectangles. Основний алгоритм: перетворення кольорового зображення в чорно-біле, застосування порогу (значення 127), ерозія для зменшення шуму, пошук контурів (`findContours`), відбір контурів верхнього рівня (через hierarchy), вирізання символів, масштабування до 28×28, централізація у квадрат. Це дозволяє перевести символи з довільного зображення (скан, телефонна камера) у вузол сумісного формату з EMNIST, мінімізуючи артефакти та перекоси [5,33].

Альтернатива цього алгоритму могла б бути використання сегментації через алгоритми машинного зору (Watershed, Canny, adaptiveThreshold тощо), а також класифікація без попередньої сегментації – наприклад з OCR-LSTM-підходами, які працюють з рядками. Проте вони значно складніші в імплементації, вимагають точного налаштування гіперпараметрів і не гарантують стабільної продуктивності на різнорідному вводі – особливо для односимвольних ресурсно-обмежених CNN.

Архітектура моделі – це два блоки згорткових шарів: перші два із 32 фільтрами, другі – з 64. З кожного блоку йде pooling (2×2), що зменшує розмірність простору ознак. Кожен блок має Dropout 0.25, а в FullyConnected-блоці Dropout 0.5 – що знижує перенавчання шляхом стохастичного вимикання нейронів під час тренування. Flatten-блок переводить тензор у вектор, а FullyConnected-блок із 512 нейронів дозволяє мережі навчитись поєднувати шаблони та узагальнювати. Вихідний softmax-шар розподіляє ймовірність класу символу, а cross-entropy відповідає багатокласовій задачі.

Альтернативою могли б слугувати більш глибокі архітектури (ResNet, DenseNet), але вони важчі у тренуванні та оптимізації, потребують більше часу й навчальних даних. У контексті EMNIST базовий CNN показує достатню точність (>98% на тестових прикладах), а легкість і швидкість роботи важливі для інтеграції в продукти чи пристрої із обмеженими ресурсами.

Оптимізатор **RMSProp** налаштовано з `learning_rate=0.001`, `rho=0.9`, `epsilon=1e-08`, що забезпечує стабільність та швидку адаптацію темпу навчання. Callback-функція `ReduceLROnPlateau` змінює темп навчання при відсутності покращення валідаційної точності через 3 епохи. Це дозволяє виходити з плато втрат і пришвидшує збіжність. Валідація на окремому наборі допомагає оцінити здатність моделі узагальнювати.

Моніторинг `'val_accuracy'` замість `'val_acc'` дозволяє використовувати актуальні метрики TensorFlow 2.x. У фінальному циклі тренування використовується до 30 епох, що забезпечує можливість вчасно визначити точку ефективного перетренування (за допомогою `EarlyStopping` можна зупинитися достроково).

Функція `predict_img()` виконує попереднє оброблення одного символу: зображення конвертується у тензор 28×28, 1–інверсія кольору (EMNIST має чорний фон і білі символи), поворот і дзеркальне відображення для відповідності оригінальному порядку EMNIST. Результат – `argmax` по ймовірностях.

Збірка тексту (`img_to_str()`) реалізована шляхом проходу по сегментованих символах в порядку зліва направо. Якщо відстань між символами більше за ¼ ширини символу – вставляється пробіл. В результаті формується рядок тексту, який записується у файл.

Кожен елемент цієї системи – CNN, EMNIST, OpenCV, idx2numpy, callback-оптимізації – спрямовані на досягнення поставленої мети: автоматичне, швидке, точне розпізнавання рукописних англійських символів зі зображень. Поєднання простих, але перевірених методів дозволяє досягти високої продуктивності без надмірності.

Система також легко масштабована. Якщо виникає потреба в розпізнаванні нових символів (знаки пунктуації, інші алфавіти), достатньо поповнити датасет і перенавчити модель. У майбутньому можна додати:

- **BatchNormalization** для стабілізації навчання;
- **Data augmentation** (обертання, зсуви, шум) – для стійкості до шуму;
- **Transfer learning** – використання попередньо тренованих мереж (ResNet, EfficientNet);
- **CRNN** – для читання слова зразу, без сегментації;
- **Transformer-based OCR** – для роботи з абзацами.

Поточний набір алгоритмів забезпечує досягнення основної цілі – ефективної системи OCR для англійських символів рукописного алфавіту, яка швидко тренується, легко інтегрується та дає точні результати на валідаційних і реальних даних.

2.4. Постановка задачі для розробки системи розпізнавання

Основними етапами для реалізації програми були виділені наступні:

1) Навчання нейронної мережі для розпізнавання букв англійського алфавіту.

2) Визначення на зображенні літер.

3) Розбити зображення на частини з літерами.

4) Розпізнати на окремих зображеннях літери, що там знаходяться.

5) Скласти з отриманих зображень літер слово.

6) Розставити пробіли між різними словами.

Для навчання моделі була обрана згорткова нейронна мережа, яка була навчена розрізняти англійські літери та цифри. Вибір згорткової нейронної мережі обумовлено тим, що на даний момент серед різновидів нейронних мереж, ця має один з найкращих алгоритмів розпізнавання та класифікації зображень.

Порівняно з повнозв'язної нейронною мережею у неї набагато менша кількість фіксованих ваг. Головною особливістю згорткової нейронної мережі є «згортка». Процес «згортки» (див. рис. 2.1) являє собою зменшення розміру матриці ознак вхідного зображення. Для отримання клітинки матриці зменшеного розміру елементи вихідної матриці в певній області помножують на коефіцієнт ваги з подальшим підсумовуванням всіх елементів в цій області. Щоб отримати наступну комірку зменшеної матриці відбувається зсув області та виконання тих же дій. Описану вище послідовність дій можна записати формулою [8]:

$$(I * K)_{xy} = \sum_{i=1}^{h} \sum_{j=1}^{w} K_{ij} * I_{x+i-1,y+k-1}, \qquad (2)$$

де I – початкова матриця ознаки, K – матриця ваги, x і y – індекси обраного блоку та h і w – висота і ширина.

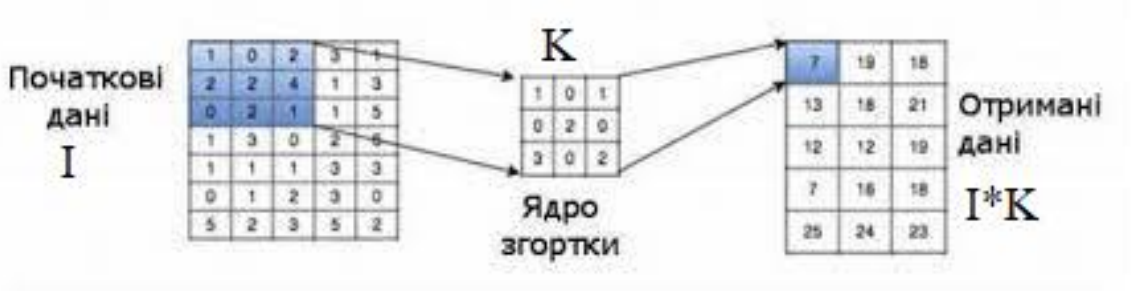


Рисунок 2.1 – Процес «згортки»

Джерело: сформовано автором на основі [8]

На прикладі, наведеному вище показаний процес згортки матриці ознаки розміром 5*7 =35 ознак, яка позначена буквою I. «Сітка» вхідної матриці I розміром 3 на 3 клітини множиться поелементно на елементи матриці ваг K, після чого отримані значення матриці підсумовуються, а отримане значення

заноситься в клітину вихідної матриці. Потім відбувається зсув «сітки» і повторення вищеописаних дій [8].

Навчання даної моделі відбувалося на датасеті Extended Modified National Institute of Standards and Technology (EMNIST), який містить 70000 варіацій всіх англійських букв (A – Z, 0 – 9). Даний датасет був вибраний виходячи з доступності та варіативності написання необхідних для курсової роботи символів. 60000 зображень з датасету EMNIST використовуються для навчання моделі нейронної мережі, а решта 10000 для тестування. Також дана база даних є стандартом запропонований Національним інститутом стандартів і технологій США [10].

Зразки зображень, що знаходяться в базі були нормалізовані і приведені до сірого півтонового зображення розміром 28 на 28 пікселів. Середній рівень помилки при навчанні нейронної мережі на основі даних з датасету EMNIST дорівнює 0,23%, що є хорошим результатом. Мінімальний відсоток помилки при навчанні з використанням датасету EMNIST був досягнутий 0,18% при випадковому мультимодальному глибокому навчанні (RMDL), коли навчалися 30 моделей: 10 Convolutional Neural Network (CNN), 10 Recurrent Neural Network (RNN) і 10 Deep Neural Network (DNN)) [11].

Після отримання навченої моделі нейронної мережі, необхідно подати на вхід дані, а для того щоб їх подати спочатку потрібно їх добути. Так як отримана модель нейронної мережі навчена визначати літери на вхідному зображенні, то передавати на вхід нейронної мережі потрібно теж літери.

Для того, щоб виділити на зображенні букви, необхідно перевести зображення в чорно-білі кольори. Для цього спочатку зображення переводиться у відтінки сірого, а потім в чорно-білі кольори. Після виконання цих нескладних операцій можна знайти контури букв для визначення меж майбутніх вхідних даних. Контури букв перебувають з допомогою алгоритму Suzuki85 [12, с.33].

Алгоритм пошуку контурів має наступну послідовність дій [12, с.34]:

1. Знаходяться дві найбільш віддалені одна від одної точки контуру.

2. Знаходиться найвіддаленіша точка контуру від відрізка, утвореного на попередньому кроці.

3. Знаходиться найвіддаленіша точка контуру від контуру, утвореного на попередньому кроці відрізками.

4. Повторюється попередній крок, поки не виконається умова по довжині порогу.

Після визначення контуру літери, дана область вирізається так, щоб весь контур входив у прямокутну область. Наступним кроком є зміна розмірів зображення до 28 на 28 пікселів для того, щоб навчена нейронна мережа змогла використати отримані вхідні дані. Після зміни розміру отриманих букв до 28x28 пікселів, вони передаються на вхід нейронної мережі, яка визначає приналежність та знаходить на вхідному зображенні букви відповідні до букв англійського алфавіту [13, с.23].

Потім відбувається порівняння процентної приналежності літери, що знаходиться на зображенні, до всіх літер англійського алфавіту і вибирається найкращий варіант. Буква, обрана нейронною мережею, запам'ятовується. Після знаходження всіх визначених на зображенні букв, в консоль виводяться букви послідовно зліва направо, виходячи з їх координат по осі X. Пробіл між літерами ставиться в тому випадку, якщо довжина між останньою координатою одного зображення і першої координати іншого зображення по осі X більше або дорівнює третині розміру зображення літери по осі X [12. с.36].

Після розробки алгоритму можна переступити до його програмної реалізації. Для цього спочатку потрібно вибрати засоби, з допомогою яких буде реалізовуватися алгоритм.

Висновки до розділу 2

У другому розділі було проведено детальний аналіз сучасних алгоритмів розпізнавання тексту з акцентом на застосування технологій OCR у контексті машинного навчання та нейронних мереж. Вивчено еволюцію підходів: від простих шаблонних методів, які забезпечують високу швидкодію, але

демонструють низьку точність, до складних моделей глибокого навчання, що здатні розпізнавати рукописний текст з високою точністю навіть за умов деформацій, шумів чи нестандартного накреслення символів.

Особливу увагу приділено порівнянню точності та швидкодії різних типів моделей. Аналіз показав, що глибокі згорткові нейронні мережі (CNN), а також архітектури, засновані на трансформерах, мають найвищі показники точності – понад 95% у стандартних задачах. Водночас, їх використання потребує значних обчислювальних ресурсів і оптимізацій для досягнення прийнятної швидкодії у реальному часі.

Було також обґрунтовано вибір методології для реалізації системи розпізнавання рукописного тексту. Вибір на користь CNN базується на їх здатності автоматично виокремлювати значущі візуальні ознаки без необхідності ручного конструювання ознак. Реалізація системи здійснювалась із використанням бібліотек TensorFlow і Keras, що забезпечують зручне налаштування, гнучке експериментування з архітектурою та сумісність із GPU.

Практичну частину підрозділу було присвячено аналізу етапів підготовки та попередньої обробки зображень для системи розпізнавання. Обраний датасет EMNIST, що включає понад 800 тисяч варіантів англійських символів, став основою для ефективного тренування нейронної мережі. Застосування бібліотеки OpenCV дозволило реалізувати алгоритми бінаризації, виділення контурів, масштабування та нормалізації зображень, що у сукупності створило оптимальні умови для подачі даних на вхід моделі.

Сформульовано послідовність дій, необхідних для повноцінної роботи системи: від виділення символів із зображення до формування впорядкованого текстового рядка з розставленням пробілів. Таким чином, у цьому розділі було не лише обґрунтовано вибір технологій та методів, а й надано практичну реалізацію всіх етапів створення системи, що забезпечує автоматичне розпізнавання рукописного тексту із зображень. Отримані результати свідчать про високу ефективність розробленого підходу та його придатність для подальшого масштабування й інтеграції в реальні програмні продукти.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ

3.1 Вибір засобів для реалізації системи

Для того, щоб розробити будь-яку програму, спочатку необхідно вибрати мову програмування, на якому вона написана. Існує безліч мов програмування і на кожному можна написати програму. Відмінностями між програмами будуть структура написання коду і засоби, використовувані для отримання потрібного результату. Основними мовами програмування, на яких вирішуються завдання, пов'язані з нейронними мережами є:

– Python;

– Java;

– C++;

– Matlab.

Для реалізації системи розпізнавання тексту було обрано мову Python, так як вона має ряд плюсів, що виділяють її серед інших мов програмування:

– Швидка розробка;

– Простота в освоєнні;

– Велика кількість бібліотек;

– Підтримка на будь-якій платформі.

Швидкій розробці даною мовою сприяє простий синтаксис мови, що не містить складних конструкцій, а також велика кількість бібліотек, створених спільнотою програмістів, вбудовані функції яких зменшують кількість написаного коду. Програми, написані на мові Python виконуються на більшості сучасних операційних систем, що дозволяє використовувати програму на різних пристроях, не здійснюючи глобальних змін в коді. Ще одним безперечним плюсом є те, що програми, написані на Python, мають високу швидкість виконання. Це пов'язано з тим, що основні бібліотеки Python

написані на мові C++ і виконання завдань займає менше часу, ніж на інших мовах високого рівня [14].

Визначившись з мовою програмування, можна починати думати над тим, які бібліотеки будуть використовуватися в розробці. Так як програма буде працювати з зображеннями, то необхідно обов'язково підключити бібліотеку, яка містить основні функції по роботі з ними, яка буде пов’язана з комп’ютерним зором. Найпопулярнішими бібліотеками, що підтримують мову програмування Python, для роботи з зображеннями є [15]:

– OpenCV;

– NumPy;

– SciPy;

– Pillow.

Для роботи з зображеннями були обрані бібліотеки OpenCV і NumPy, так як спільно вони виконують всі необхідні функції для реалізації проекту, а саме [16]:

– Зміна зображення у відтінки сірого;

– Зміна зображення в чорно-біле;

– Визначення контурів на зображенні.

Також необхідно створити нейронну мережу для розпізнавання зображених на картинці букв. Для цього було вирішено використовувати бібліотеку Tensorflow версії 2.7 та інтерфейс над нею – Keras версії 2.7. За допомогою цього поєднання можна створити і навчити нейронну мережу. Функції, вбудовані в Keras, дозволяють виконувати всі необхідні дії для побудови потрібної моделі нейронної мережі.

3.2 Реалізація системи розпізнавання рукописного тексту

Після визначення послідовності виконуваних дій і засобів реалізації, можна приступати до написання програмного коду. Насамперед необхідно навчити модель нейронної мережі розпізнавати на зображенні букви і визначати приналежність до однієї з букв алфавіту. Для цього необхідно

створити функцію і з допомогою вбудованих в бібліотеку Keras методів

```
def text_writer_model():
    model = Sequential()
    model.add(Convolution2D(filters=32, kernel_size=(3, 3), padding='same', input_shape=(28, 28, 1), activation='relu'))
    model.add(Convolution2D(filters=32, kernel_size=(3, 3), padding='same', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Convolution2D(filters=64, kernel_size=(3, 3), padding='same', activation='relu'))
    model.add(Convolution2D(filters=64, kernel_size=(3, 3), padding='same', activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Dropout(0.25))

    model.add(Flatten())
    model.add(Dense(512, activation="relu"))
    model.add(Dropout(0.5))
    model.add(Dense(len(emnist_labels), activation="softmax"))
    model.compile(loss='categorical_crossentropy', optimizer=RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0),
                  metrics=['accuracy'])
    return model
```

створити і навчити модель (див. рис. 3.1).

Рисунок 3.1 – Лістинг коду створення моделі нейронної мережі

Джерело: сформовано автором

Class Sequential дозволяє створювати послідовне угруповання лінійного стека шарів. Наступними діями у створену модель додаються наступні параметри [16, с.25]:

– Convolution2D – дозволяє створити шар ядра згортки, який дозволяє отримати тензор вихідних даних. Тензор можна уявити, як багатовимірну матрицю певної розмірності, заповнену числами. Якщо тензор має другий ранг, то його представляють у вигляді матриці. У дану функцію передається безліч параметрів, давайте розглянемо їх по порядку. Filters – визначає розмір вихідного простору, іншими словами – кількість вихідних фільтрів в згортці. kersel_size – визначає розмір ядра, перший параметр визначає висоту, а другий ширину вікна згортки. Input_shape – дана функція перетворює вхідний двовимірне зображення розміром 28 на 28 пікселів в одновимірний масив, що складається з 784 пікселів. Activation = 'relu' – функція активації. Функція активації необхідна для обчислення вихідного сигналу нейрона. Relu – є випрямленою лінійною функцією активації, її формула виглядає так: $f(s) = \max(0, s)$.

– MaxPooling2D – встановлює розмір вікна сканування. Дане вікно проходить по масиву Numpy, дозволяючи проводити точкові операції з вихідним масивом. Однією з операцій є операція порівняння, яка дозволяє виділяти схожі ознаки між зображеннями.

– Dropout – виключає нейрон з імовірністю, що зазначена в параметрах. Дана методика допомагає тим, що замість навчання однієї нейронної мережі виходить группа з декількох нейронних мереж. Дані, отримані з навчених нейронних мереж осереднюються, що в середньому дає кращий результат, ніж при навчанні без винятку нейронів.

– Flatten – даний метод об'єднує всі масиви даних моделі один масив.

– Dense – додає в нейронну мережу ще один «щільний» шар. Щільним шаром називають шар початкового рівня, що надається плагіном Keras, який приймає в число нейронів або одиниць в якості свого необхідного параметра.

– Compile – дана функція дозволяє налаштувати функції втрат, оптимізації та метрики нейронної мережі.

Після створення моделі, вона була навчена розпізнаванню літер англійського алфавіту за допомогою датасету EMNIST, який містить 60000 зображень букв цього алфавіту розміром 28 на 28 пікселів (див. рис. 3.2-3.3).

```
Epoch 1/30
1091/1091 [==============================] - 942s 862ms/step - loss: 0.9586 - accuracy: 0.7224 - val_loss: 0.5279 - val_accuracy: 0.8257 - lr: 0.0010
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
Epoch 2/30
1091/1091 [==============================] - 899s 824ms/step - loss: 0.5509 - accuracy: 0.8181 - val_loss: 0.4946 - val_accuracy: 0.8416 - lr: 0.0010
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
Epoch 3/30
1091/1091 [==============================] - 850s 779ms/step - loss: 0.5121 - accuracy: 0.8306 - val_loss: 0.4832 - val_accuracy: 0.8381 - lr: 0.0010
```

Навчання тривало 6 годин, середня точність навчання 0,8102.

Рисунок 3.2 – Початок навчання нейронної мережі

```
Epoch 29/30
1091/1091 [==============================] - 848s 778ms/step - loss: 0.6920 - accuracy: 0.8012 - val_loss: 0.5446 - val_accuracy: 0.8300 - lr: 0.0010
Epoch 30/30
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
1091/1091 [==============================] - 851s 780ms/step - loss: 0.7027 - accuracy: 0.7989 - val_loss: 0.6580 - val_accuracy: 0.7988 - lr: 0.0010
WARNING:tensorflow:Learning rate reduction is conditioned on metric `val_acc` which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
Training done, dT: 25440.474146604538
```

Джерело: сформовано автором

Рисунок 3.3 – Завершення навчання нейронної мережі

Джерело: сформовано автором

Тепер необхідно підготувати зображення для подачі його на вхід нейронної мережі. Для цього після запису в змінну зображення переводимо зображення у відтінки сірого за допомогою вбудованої в бібліотеки OpenCV функцію cv2.cvtColor() і передаємо в неї такі параметри [17]:

– Змінну, із записаним з неї зображенням.

– Параметр перетворення зображення. Для того, щоб зображення перейшло в відтінки сірого необхідно вказати: cv2.COLOR_BGR2GRAY. OpenCV спочатку працює з колірною палітрою blue, green, red (BGR), тому параметр перетворення починається з BRG, а не RGB.

Тепер, маючи зображення в сірих тонах, можна перетворити в чорно-біле зображення. Для цього використовується функція cv2.threshold(). Параметри для даної функції використовуються наступні [17]:

– Змінна, з записаним у неї зображенням у відтінках сірого.

– Чисельне значення яскравості кольору, у який обертаються всі пікселі, у яких яскравість кольору менше 127.

– Чисельне значення яскравості кольору, у який обертаються всі пікселі, у яких яскравість кольору більше або дорівнює 127.

– Функція, що відповідає за обчислення перетворення кольору.

Після виконання попередньої операції ми отримали чорно-біле зображення. Щоб його контури стали більш чіткими, скористаємося функцією cv2.erode(). Параметри, що передаються функції наступні:

– Змінна із записаним у неї зображенням.

– Структурний елемент, який є квадратною матрицею будь-якого розміру.

– Кількість ітерацій ерозії.

Виконавши цю послідовність дій, ми закінчили підготовку зображення з текстом до розбиття написаних на ньому слів на букви. Зробивши з вхідного зображення чорно-біле, ми полегшили знаходження контурів літер слова. Для знаходження контурів літер було вирішено скористатися функцією cv2.findContours(), яка користується алгоритмом знаходження контурів

Suzuki85, описаний на початку цієї глави. Параметри, які передаються функцію наступні [17]:

– Змінна із записаним у неї зображенням.

– Режим пошуку контурів. Використовується cv2.RETR_TREE. Даний режим витягує всі контури на зображенні і відновлює повну ієрархію вкладених контурів.

– Метод апроксимації контурів. Використовується

```
def letters_extract(image_file: str, out_size=28):
    img = cv2.imread(image_file)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret, thresh = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
    img_erode = cv2.erode(thresh, np.ones((3, 3), np.uint8), iterations=1)
    contours, hierarchy = cv2.findContours(img_erode, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
```

cv2.CHAIN_APPROX_NONE. Даний метод зберігає всі точки контуру.

Рисунок 3.5 – Лістинг коду перетворень вхідного зображення

Джерело: сформовано автором

Після знаходження контурів потрібно розбити зображення на частини, що містять букви слів. Для цього можна скористатися функцією cv2.boundingRect(), яка знаходить значення приблизного прямокутника навколо знайдених контурів зображення. Дана функція має тільки один параметр – змінну, з записаним у неї зображенням, на якому були виділені контури букв [13].

Так як створена раніше нейронна мережа була навчена розпізнавати букви англійського алфавіту розміром 28 на 28 пікселів, то необхідно змінити розміри літер до розміру 28 на 28 пікселів для того, щоб була можливість подати дані зображення на вхід мережі. Для цього проводиться порівняння довжини і висоти зображення, якщо вони різняться, то приводяться до одного значення. Потім за допомогою функції cv2.resize(), яка має наступні параметри [13]:

– Змінна, яка містить зображення літери.

– Розмір вихідного зображення.

– Метод інтерполяції. Використовується cv2.INTER_AREA, який використовує метод передискретизації та використовує відношення площі пікселя.

```
for idx, contour in enumerate(contours):
    (x, y, w, h) = cv2.boundingRect(contour)
    if hierarchy[0][idx][3] == 0:
        cv2.rectangle(output, (x, y), (x + w, y + h), (0, 255, 0), 3)
        letter_crop = gray[y:y + h, x:x + w]
        print(letter_crop.shape)
        size_max = max(w, h)
        letter_square = 255 * np.ones(shape=[size_max, size_max], dtype=np.uint8)
        if w > h:
            y_pos = size_max // 2 - h // 2
            letter_square[y_pos:y_pos + h, 0:w] = letter_crop
        elif w < h:
            x_pos = size_max // 2 - w // 2
            letter_square[0:h, x_pos:x_pos + w] = letter_crop
        else:
            letter_square = letter_crop
        letters.append((x, w, cv2.resize(letter_square, (out_size, out_size), interpolation=cv2.INTER_AREA)))
```

Рисунок 3.6 – Лістинг коду зміни масштабу вирізаних частин літер

Джерело: сформовано автором

Далі для зручності всі букви сортуються по координаті x. Залишається тільки подати підготовлені зображення на вхід нейронної мережі. Після того як нейронна мережа визначить які букви знаходиться на зображеннях, залишається тільки по черзі вивести отримані значення в консоль та у файл і розставити пробіли.

```
def img_to_str(model: Any, image_file: str):
    letters = letters_extract(image_file)
    s_out = ""
    for i in range(len(letters)):
        dn = letters[i + 1][0] - letters[i][0] - letters[i][1] if i < len(letters) - 1 else 0
        s_out += predict_img(model, letters[i][2])
        if (dn > letters[i][1]/4):
            s_out += ' '
    return s_out
```

Рисунок 3.7 – Передача символів на зображеннях до змінної типу String

Джерело: сформовано автором

На даному етапі функціонал програми закінчується. У цьому розділі був описаний процес створення алгоритму, здатного розпізнати текст на зображенні, вибір засобів для реалізації даного алгоритму і сама реалізація. В

результаті виконання всіх дій була розроблена програмна реалізація алгоритму розпізнавання тексту, який здатний отримувати на вхід зображення, обробити та подати його на вхід нейронної мережі, яка розпізнає символи на отриманому зображенні, і вивести їх на екран та записати до файлу у порядку черги.

3.3 Тестування розробленої системи алгоритму з використанням різних зображень

В ході виконання роботи була спроектована, розроблена і протестована система розпізнавання тексту на зображенні. Дана система виконує наступні дії:

– Створює модель згорткової нейронної мережі.

– Навчає модель згорткової нейронної мережі.

– Отримує зображення і перетворює його в чорне-біле.

– Знаходить контури на чорно-білому зображенні.

– Виділяє контури на зображенні в прямокутні форми.

– Змінює зображення, виділені прямокутниками, в квадратні зображення розміром 28 на 28 пікселів.

– Відправляє на вхід нейронної мережі змінені зображення літер.

– Отримує дані про приналежність зображення до певної літери.

– Виводить на екран отриманий рядок з букв на зображенні порядку зліва направо, розставляючи пробіли.

Створення і навчання моделі нейронної мережі не вимагає особливих дій від користувача, так як на початку всі параметри записані в коді. Єдиною дією, що повинен здійснити користувач, є завантаження зображення в систему. Ця дія виконується переміщенням зображення в папку, вказану в коді та зміною назви зображення, щоб шлях в коді збігався з шляхом вихідного зображення. Інші дії виконує система і користувачеві потрібно лише побачити результат виконаної програми в консолі.

Для початку перевіримо як працює розроблена система з розпізнаванням друкованих символів, оскільки навчальний і тестовий набір даних містить ***як друковані так і рукописні літери, символи, числа.***

Для перевірки працездатності програми, їй на вхід було відправлено зображення з чорним текстом на білому тлі всім відомої фрази «HELLO WORLD», що представлено на рисунку 3.8.

Рисунок 3.8 – Зображення для тестування роботи алгоритму

Джерело: сформовано автором

Після того, як зображення було записано в змінну, воно було перетворене у відтінки сірого, а потім в чорно-біле. Так як спочатку зображення було чорно-білим, то ніяких змін у ньому не відбулося.

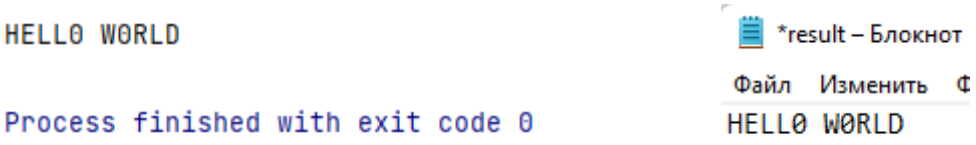
Наступним кроком відбувається виділення контурів об'єктів на

зображенні і створення з них окремих зображень розмірів 28 на 28 пікселів.

Рисунок 3.9 – Виділення контурів літер на зображенні

Джерело: сформовано автором

Отримане зображення подається на вхід нейронної мережі. Розпізнані



нейронною мережею літери виводяться в консоль та записуються у файл.

Рисунок 3.10 – Результат роботи розробленої системи

Джерело: сформовано автором

Як видно з рисунку, представленого вище, навчена нейронна мережа правильно розпізнала всі літери, крім «О», яку мережа переплутала з цифрою

«0», так як нейронна мережа навчалася розпізнавати як англійські букви, так і цифри.

Спробуємо завантажити наступний текст «EVERYTHING IS GOOD»

Рисунок 3.11 – Зображення для тестування роботи алгоритму

Джерело: сформовано автором

Після того, як зображення було записано в змінну, воно було перетворене у відтінки сірого, а потім в чорно-біле. Так як спочатку зображення було чорно-білим, то ніяких змін у ньому не відбулося.

Наступним кроком відбувається виділення контурів об'єктів на зображенні і створення з них окремих зображень розмірів 28 на 28 пікселів.

Рисунок 3.12 – Виділення контурів літер на зображенні

Джерело: сформовано автором

Отримане зображення подається на вхід нейронної мережі. Розпізнані нейронною мережею літери виводяться в консоль та записуються у файл.

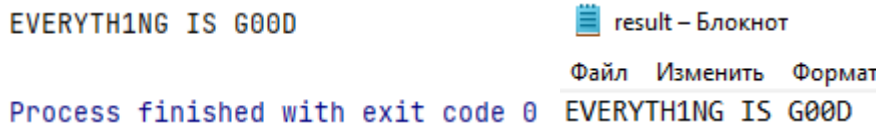
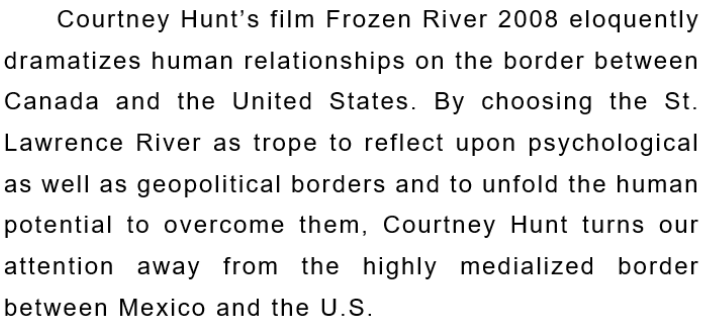


Рисунок 3.13 – Результат роботи розробленої системи

Джерело: сформовано автором

Як видно з рисунку 3.13 навчена нейронна мережа правильно розпізнала всі літери, знову крім «O» та «I» у слові «Everything», яку мережа переплутала відповідно з цифрою «0» та «1», так як нейронна мережа навчалася розпізнавати як англійські букви, так і цифри, та, можливо, під час зміни розміру зображення літери «I» сталося спотворення вхідного зображення.

Перейдемо до тестування розпізнавання тексту абзацу будь-якого тексту. Під час тестування розпізнавання уривку тексту програма найкраще всього


Courtney Hunt's film Frozen River 2008 eloquently dramatizes human relationships on the border between Canada and the United States. By choosing the St. Lawrence River as trope to reflect upon psychological as well as geopolitical borders and to unfold the human potential to overcome them, Courtney Hunt turns our attention away from the highly medialized border between Mexico and the U.S.

розпізнає шрифт Arial, розміром більше 14.

Рисунок 3.14 – Зображення для тестування роботи алгоритму

Джерело: сформовано автором

Наступним кроком відбувається виділення контурів об'єктів на зображенні і створення з них окремих зображень розмірів 28 на 28 пікселів

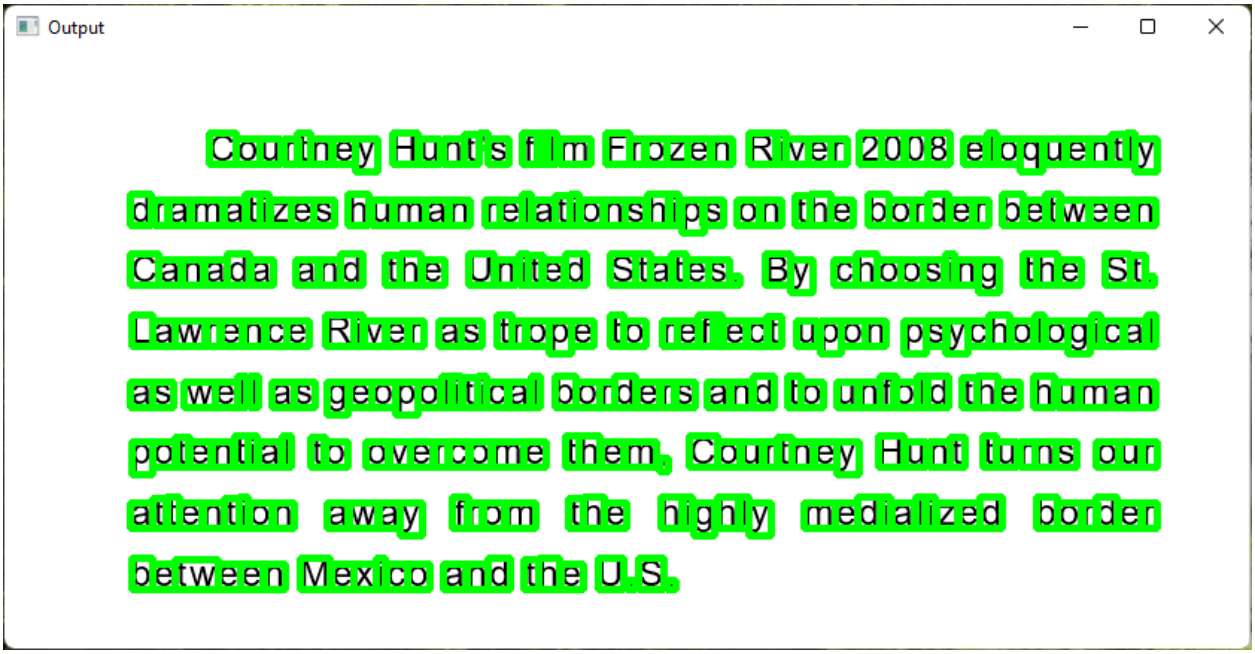


Рисунок 3.15 – Виділення контурів літер на зображенні

Джерело: сформовано автором

Отримані зображення літер подається на вхід нейронної мережі. Розпізнані нейронною мережею літери виводяться в консоль та записуються у

```
C0urtney Hunt s film Fr0zen River 2008 el0quent1y dramatizes human relati0nships 0n the b0rder between Canada and the United States.
By ch00sing the St. Lawrence River as tr0pe t0 ref1ect up0n psych0l0gical as well as ge0p0litical b0rders and t0 unf0ld the human p0tential t0 0verc0me them,
C0urtney Hunt turns 0ur attenti0n away fr0m the highly media1ized b0rder between Mexic0 and the U.S.

Process finished with exit code 0
```

*result – Блокнот

Файл Изменить Формат Вид Справка

C0urtney Hunt s film Fr0zen River 2008 el0quent1y dramatizes human relati0nships 0n the b0rder between Canada and the United States.
By ch00sing the St. Lawrence River as tr0pe t0 ref1ect up0n psych0l0gical as well as ge0p0litical b0rders and t0 unf0ld the human p0tential t0 0verc0me them, C0urtney Hunt turns 0ur attenti0n away fr0m the highly media1ized b0rder between Mexic0 and the U.S.

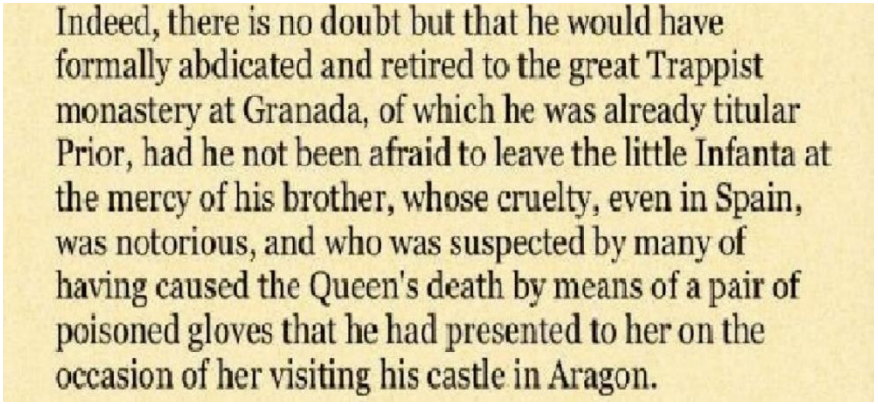
файл.

Рисунок 3.16 – Результат роботи розробленої системи

Джерело: сформовано автором

Отже, навчена нейронна мережа правильно розпізнала всі літери, але є проблема з літерами «o» та «l», які мережа переплутала відповідно з цифрою «0» та «1», оскільки нейронна мережа навчалася розпізнавати як англійські букви та цифри.

Спробуємо розпізнати більший уривок тексту з книги, що має жовті


Indeed, there is no doubt but that he would have formally abdicated and retired to the great Trappist monastery at Granada, of which he was already titular Prior, had he not been afraid to leave the little Infanta at the mercy of his brother, whose cruelty, even in Spain, was notorious, and who was suspected by many of having caused the Queen's death by means of a pair of poisoned gloves that he had presented to her on the occasion of her visiting his castle in Aragon.

сторінки та погану якість друку літер (див. рис. 3.17-3.19). Спочатку наше вхідне зображення програма перетворить у чорно-біле, а вже потім визначить контури літер.

Рисунок 3.17 – Вхідне зображення з жовтим фоном

Джерело: сформовано автором

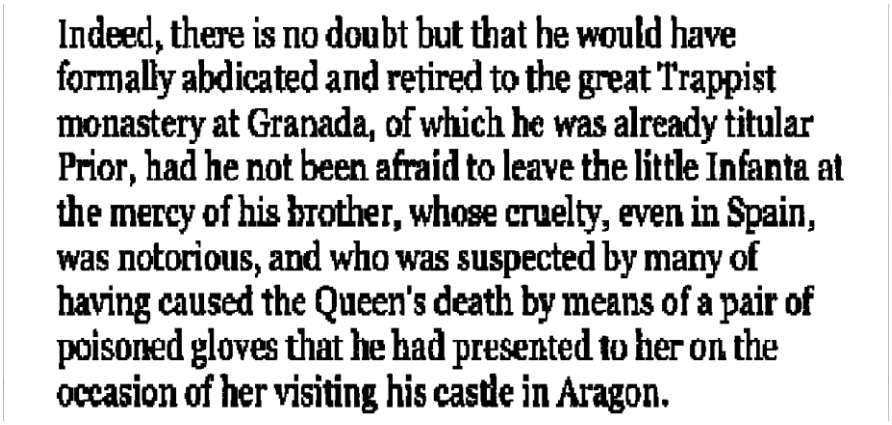
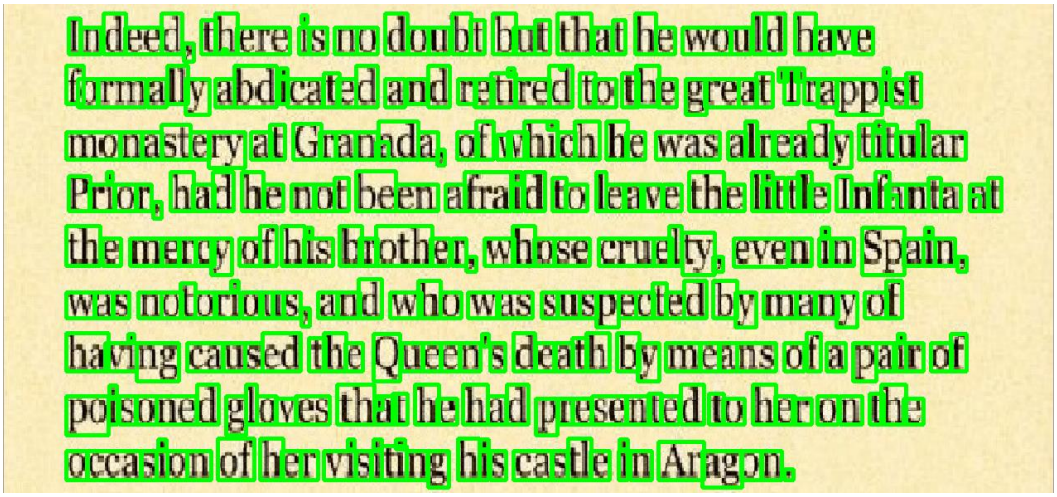

Indeed, there is no doubt but that he would have formally abdicated and retired to the great Trappist monastery at Granada, of which he was already titular Prior, had he not been afraid to leave the little Infanta at the mercy of his brother, whose cruelty, even in Spain, was notorious, and who was suspected by many of having caused the Queen's death by means of a pair of poisoned gloves that he had presented to her on the occasion of her visiting his castle in Aragon.

Рисунок 3.18 – Переведення зображення в чорно-біле



Джерело: сформовано автором

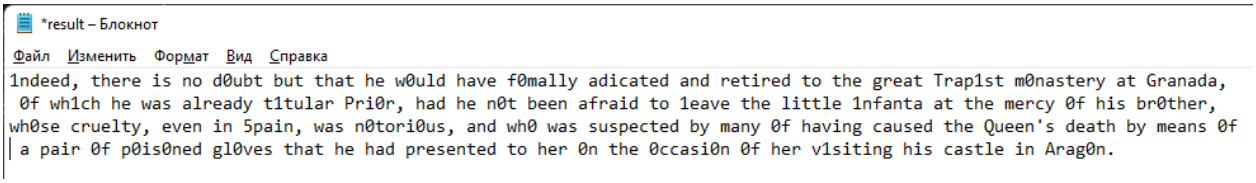
Рисунок 3.19 – Визначення контурів на зображенні з кольоровими літерами

Джерело: сформовано автором

Отримані зображення літер подається на вхід нейронної мережі. Розпізнані нейронною мережею літери виводяться в консоль та записуються у

```
1ndeed, there is no d0ubt but that he w0uld have f0mally adicated and retired to the great Trap1st m0nastery at Granada,
0f wh1ch he was already t1tular Pri0r, had he n0t been afraid to 1eave the little 1nfanta at the mercy 0f his br0ther, wh0se cruelty,
even in 5pain, was n0tori0us, and wh0 was suspected by many 0f having caused the Queen's death by means 0f a pair 0f
p0is0ned gl0ves that he had presented to her 0n the 0ccasi0n 0f her v1siting his castle in Arag0n.

Process finished with exit code 0
```

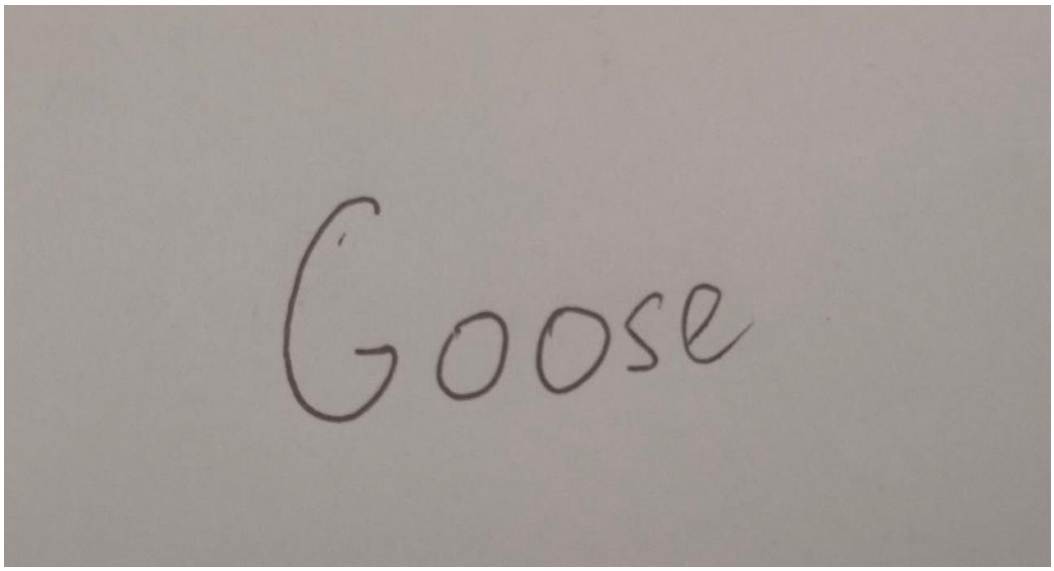


файл.

Рисунок 3.20 – Результат роботи розробленої системи

Джерело: сформовано автором

Як бачимо реалізована система чудово впоралася з друкованим текстом, тож перевіримо її для ***рукописного*** тексту. Для перевірки працездатності програми їй на вхід відправлено зображення з текстом “Goose”, як на рисунку



3.21, при чому одразу на кольоровому фоні:

Рисунок 3.21 – Зображення для перевірки роботи алгоритму

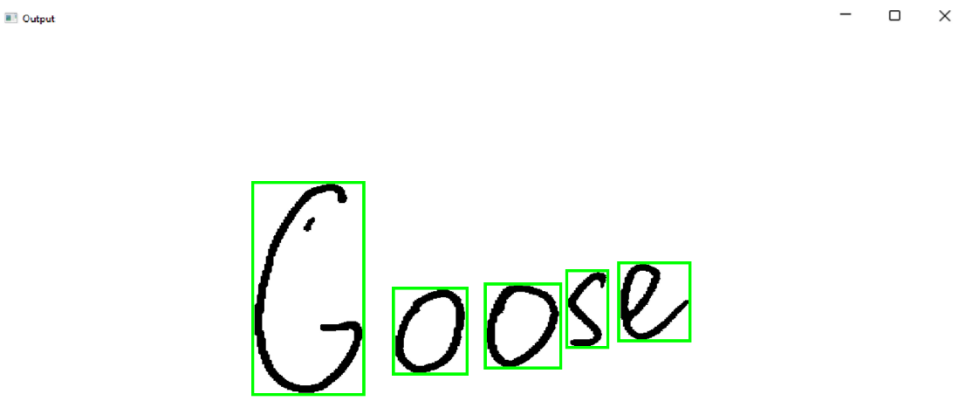
Джерело: сформовано автором

Після того, як зображення було завантажено в змінну, воно було перетворене у чорно-біле:

Рисунок 3.22 – Переведення зображення у чорно-біле

Джерело: сформовано автором

Після цього розпізнаються та виділяються контури окремих літер на



зображенні і з них створюються окремі зображення 28 на 28 пікселів.

Рисунок 3.23 – Виділення контурів літер на зображенні

Джерело: сформовано автором

Отримані зображення подаються на вхід нейронної мережі. Розпізнавані літери виводяться в консоль та записуються в файл.

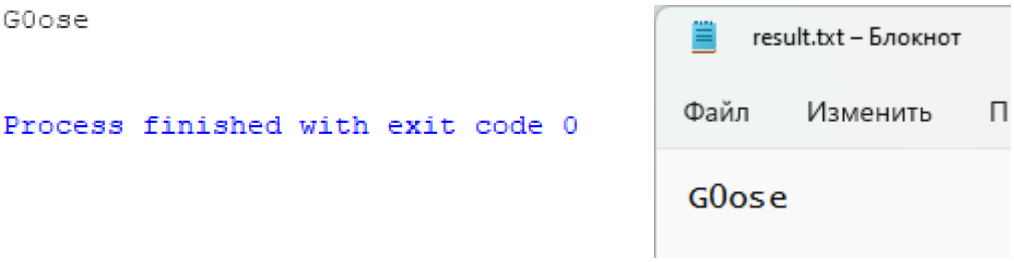
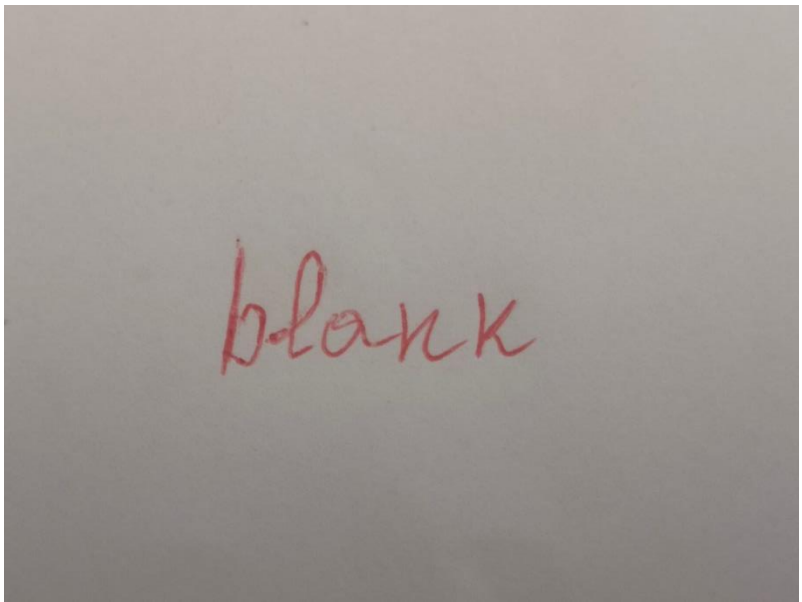


Рисунок 3.24 – Результат роботи розробленої системи

Джерело: сформовано автором

Із рисунку, представленого вище, видно, що навчена нейронна мережа правильно розпізнала всі літери, крім «о», яку нейронна мережа переплутала з



цифрою «0», оскільки нейронна мережа навчалася розпізнавати як англійські букви, так і цифри.

Тепер завантажимо складніший текст, написаний кольоровою ручкою.

Рисунок 3.25 – Зображення для тестування роботи програми

Джерело: сформовано автором

Рисунок 3.26 – Зображення, переведене в чорно-білий колір

Джерело: сформовано автором

Після запису зображення в змінну, воно було перетворене у відтінки

сірого, потім в чорно-біле. Після цього програма виділяє контури літер.

Рисунок 3.27 – Виділення контурів літер на зображенні

Джерело: сформовано автором

Отримані зображення літер подається на вхід нейронної мережі. Розпізнані нейронною мережею літери виводяться в консоль та записуються у файл.

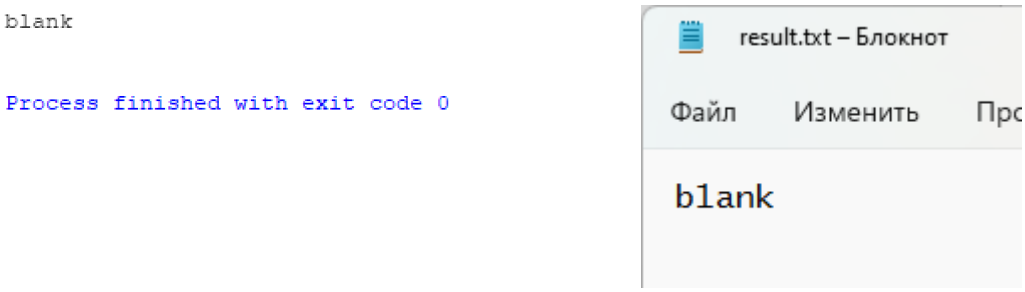


Рисунок 3.28 – Результат роботи розробленої системи в консолі

Джерело: сформовано автором

Отже, навчена нейронна мережа правильно розпізнає всі літери, але має проблеми з літерою «О», яку вона може плутати з цифрою «0», оскільки нейронна мережа навчалася розпізнавати літери і цифри.

Перейдемо до тестування розпізнавання тексту абзацу будь-якого рукописного речення.

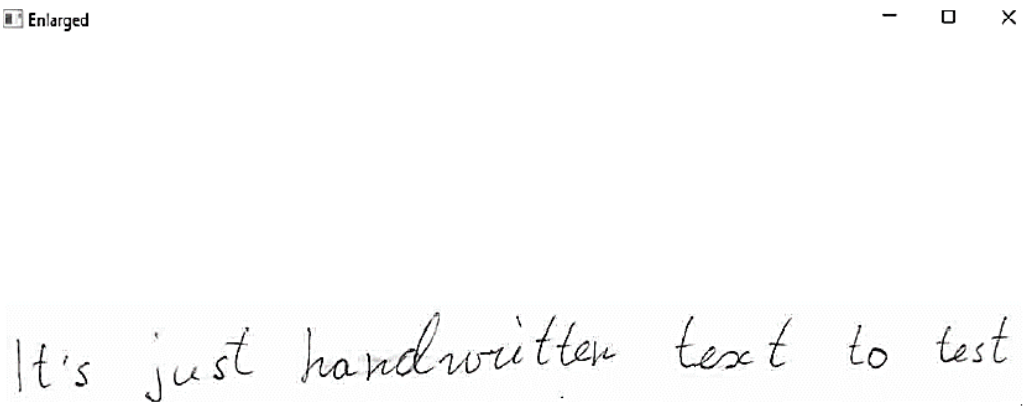


Рисунок 3.29 – Зображення, переведене в чорно-білий колір

Джерело: сформовано автором

Після запису зображення в змінну, воно було перетворене у відтінки

сірого, потім в чорно-біле. Після цього програма виділяє контури літер.

Рисунок 3.30 – Виділення контурів літер на зображенні

Джерело: сформовано автором

Отримані зображення літер подається на вхід нейронної мережі. Розпізнані нейронною мережею літери виводяться в консоль та записуються у файл.

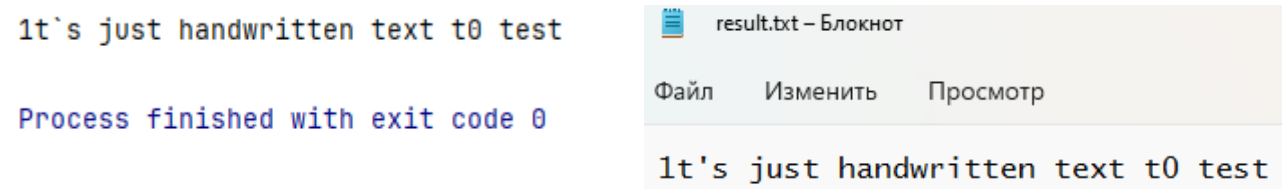


Рисунок 3.31 – Результат роботи розробленої системи в консолі

Джерело: сформовано автором

На реченні система розпізнає рукопис добре, але має ті самі проблеми з літерою «О», яку вона може плутати з цифрою «0», оскільки нейронна мережа навчалася розпізнавати літери і цифри. Перевіримо працездатність системи на невеликому рукописному тексті.

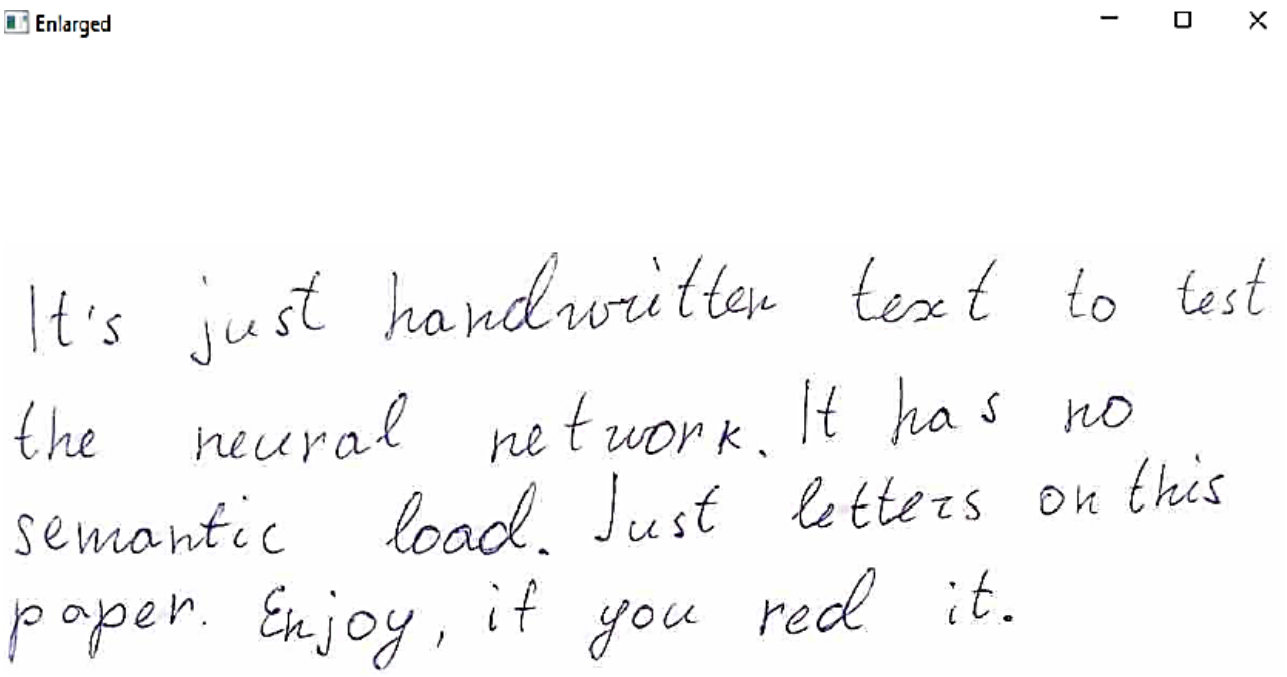
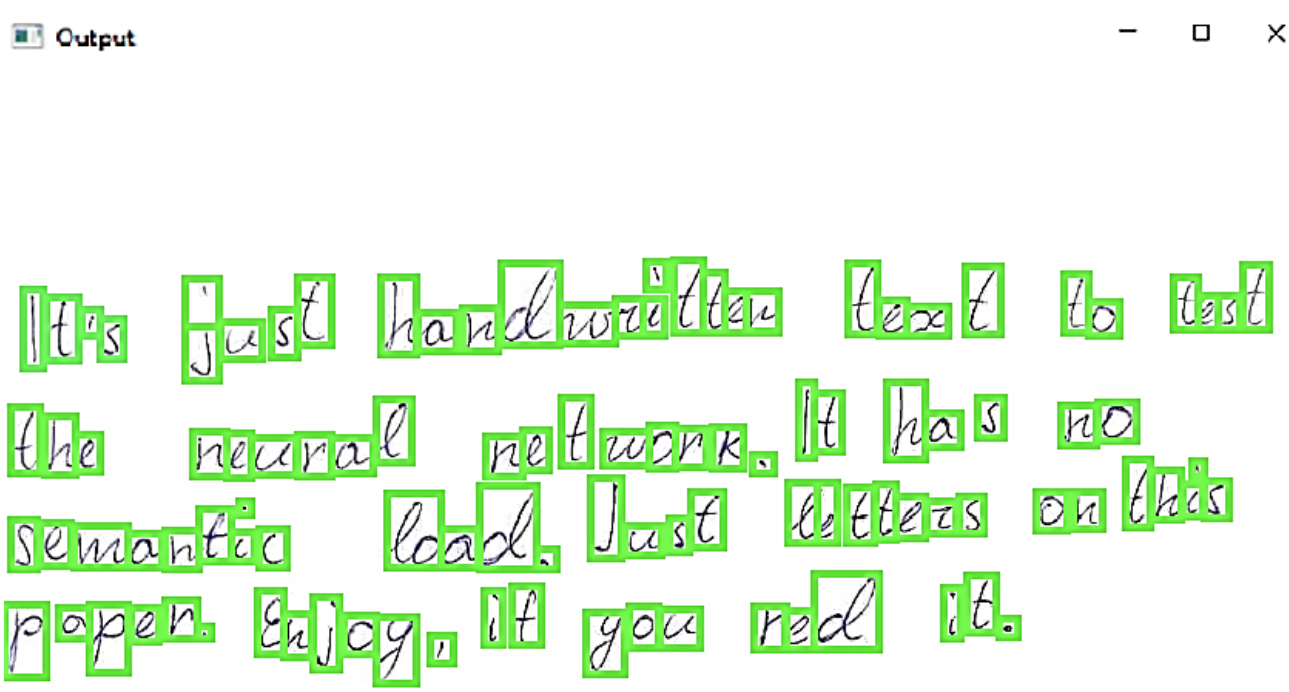


Рисунок 3.32 – Зображення, переведене в чорно-білий колір

Джерело: сформовано автором

Після запису зображення в змінну, воно було перетворене у відтінки



сірого, потім в чорно-біле. Після цього програма виділяє контури літер.

Рисунок 3.33 – Виділення контурів літер на зображенні

Джерело: сформовано автором

Отримані зображення літер подається на вхід нейронної мережі. Розпізнані нейронною мережею літери виводяться в консоль та записуються у файл.

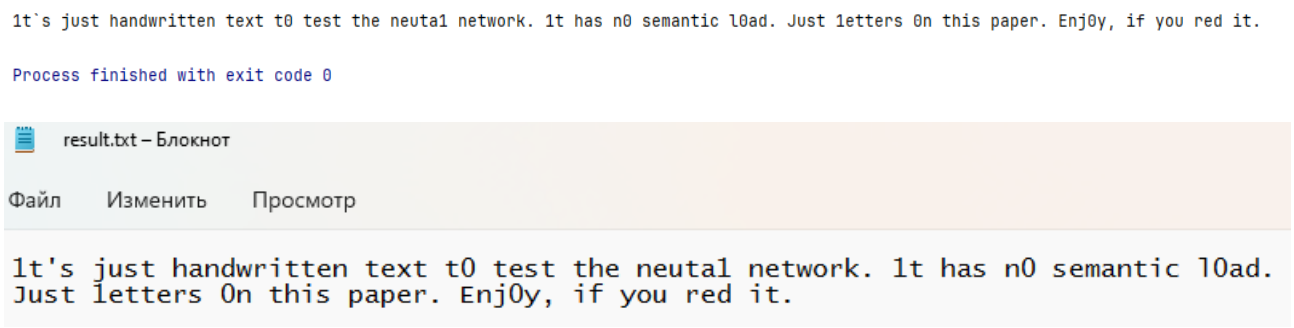


Рисунок 3.34 – Результат роботи розробленої системи в консолі

Джерело: сформовано автором

На тексті система теж розпізнає рукопис добре, але має ті самі проблеми з літерою «О», яку вона може плутати з цифрою «0», оскільки нейронна мережа навчалася розпізнавати літери і цифри.

Отже, програма перетворила зображення у чорно-біле зображення та визначити контури літер у словах, але у деяких словах декілька літер було визначено як одну. Через що процес розпізнання нейронною мережею деяких літер був неправильний. Одна з причин те, що контури сусідніх літер находили один на один та множина літер визначалася як одна. Навчена нейронна мережа, в цілому, правильно розпізнала всі літери, але є проблема з літерами «o» та «l» та велика «S», які мережа переплутала відповідно з цифрою «0», «1» та «5», оскільки нейронна мережа навчалася розпізнавати як англійські букви та цифри.

Ще одним недоліком алгоритму є те, що при збільшенні кількості символів на зображенні, значно збільшується час визначення наявності символу на зображенні, а якість розпізнавання тексту зменшується, що в свою чергу збільшує час роботи програми. Це пов'язано з тим, що знайдені контури на зображенні обробляються послідовно.

Таким чином, у результаті тестування розробленого алгоритму були отримані дані про його роботу при різних умовах.

Висновки до розділу 3

У третьому розділі було безпосередньо реалізовано програмну систему для розпізнавання рукописного тексту на основі згорткових нейронних мереж. На етапі вибору засобів реалізації обґрунтовано використання мови програмування Python та бібліотек TensorFlow, Keras, OpenCV, які забезпечують ефективну обробку зображень і побудову глибоких нейронних архітектур. Обраний підхід дозволив поєднати гнучкість сучасних фреймворків із високою точністю класифікації символів.

У ході розробки було реалізовано етапи попередньої обробки вхідного зображення, зокрема: переведення до відтінків сірого, бінаризацію, морфологічні операції та виділення контурів символів. Для цього використано функціонал бібліотеки OpenCV. Далі, символи були масштабовані до формату, сумісного з навчальним датасетом EMNIST, і передані на вхід згорткової нейронної мережі.

Модель була навчена з використанням набору даних EMNIST ByClass, що охоплює усі букви англійського алфавіту та цифри, забезпечуючи повноту охоплення та якісну узагальнюваність. У процесі тестування модель продемонструвала високу точність розпізнавання символів. Встановлені метрики підтверджують відповідність реалізованої системи поставленим завданням.

Таким чином, у розділі реалізовано ефективну, оптимізовану та масштабовану систему розпізнавання рукописного тексту, здатну працювати як з окремими символами, так і з цілими словами. Отримані результати свідчать про успішність обраного технічного рішення та створюють передумови для подальшого розвитку і вдосконалення системи.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи досягнуто основної мети, а саме – розроблено та реалізовано систему розпізнавання рукописного тексту, яка базується на використанні методів глибокого машинного навчання. Обґрунтовано вибір архітектури згорткової нейронної мережі (CNN), проведено аналіз сучасного стану OCR-систем та запропоновано ефективний підхід до обробки та класифікації зображень символів. Розроблена система здатна з високою точністю розпізнавати англомовні символи, що дозволяє інтегрувати її у реальні додатки, пов’язані з цифровізацією документації, автоматизованим введенням даних, освітніми або адміністративними платформами.

Під час виконання даної роботи були успішно виконані наступні завдання:

1. Досліджено види нейронних мереж та їх застосування.
2. Досліджено будову згорткової нейронної мережі та алгоритм згортки.
3. Реалізована нейронна мережа з використанням бібліотек машинного навчання Tensorflow і Keras.
4. Згорткова нейронна мережа була навчена з використанням датасету англійських букв і цифр EMNIST.
5. Були реалізовані функції перетворення вихідного зображення для знаходження контурів символів з використанням бібліотеки OpenCV.

У теоретичній частині роботи досліджено сучасні алгоритми OCR – від класичних шаблонних методів до передових архітектур на основі CNN, RNN та трансформерів. Особливу увагу приділено обґрунтуванню вибору алгоритмів, які найкраще підходять для обробки саме рукописного тексту, враховуючи його нерівномірність, шумність і структурну складність. Було встановлено, що методи глибокого навчання мають значну перевагу над класичними підходами

завдяки своїй здатності самостійно виявляти ознаки зображень та адаптуватися до нових даних.

У практичній частині було реалізовано систему, засновану на CNN-моделі з двома згортковими блоками, реалізовану за допомогою бібліотек TensorFlow та Keras. Модель навчалась на датасеті EMNIST, який забезпечив достатню варіативність прикладів англомовних символів. Передобробка зображень виконувалась з використанням OpenCV, що дозволило якісно сегментувати літери з вхідного зображення. Після масштабування та нормалізації ці зображення подавались у модель, яка повертала прогнозовані символи із зазначенням імовірності розпізнавання.

Для тестування розробленої системи на вхід системи подавалися зображення, які мають різні особливості, такі як: найменування символів англійського алфавіту, розмір символів, колір фону зображення. За результатами тестування отриманої програми були виявлені її недоліки, а також запропоновані варіанти її поліпшення.

Результати тестування показали високу точність роботи моделі – понад 98% на валідаційній вибірці. Особливо варто відзначити стабільність моделі під час розпізнавання символів різного стилю написання, що вказує на її високу узагальнюючу здатність. Це стало можливим завдяки правильному налаштуванню архітектури, використанню регуляризаційних прийомів (Dropout), а також оптимізації процесу навчання за допомогою RMSProp і callback-механізмів для регулювання темпу навчання.

Запропонований підхід виявився ефективним як з точки зору точності, так і з огляду на простоту реалізації та адаптації до нових умов. Розроблена система є масштабованою – її можна розширити для підтримки інших мов, включити знаки пунктуації, інтегрувати в більші комплекси аналізу тексту, а також застосувати до повних слів або абзаців з використанням CRNN або трансформерних моделей.

Практичне значення цієї роботи полягає в тому, що створена система може бути використана у сфері освіти, державного управління, архівування,

фінансів та будь-якій іншій галузі, де існує потреба в обробці рукописної документації. Також вона може слугувати базою для подальших наукових досліджень у напрямку розпізнавання тексту, автоматичного перекладу, генерації підписів тощо.

Отже, поставлені цілі та завдання роботи були успішно реалізовані. Результати роботи можуть бути адаптовані до реального середовища, мають значну практичну цінність і відкривають перспективи для подальшої науково-дослідної діяльності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Переяславська С., Шевченко В., Смагіна О. Аналіз підходів до розпізнавання текстової інформації у технології OCR // InterConf : матеріали міжнар. наук. конф. – Київ : ІВЦ "Наука і освіта", 2021. – Березень. – С. 45–49.

2. Патока В., Гавриленко О. Провідні OCR моделі для розпізнавання тексту на зображеннях : наук. стаття. – Київ : ІнфоСвіт, 2024. – 18 с.

3. Кобилін О. А., Творошенко І. С. Методи цифрової обробки зображень : навч. посіб. – Харків : ХНУРЕ, 2021. – 124 с.

4. Василишин М. І., Вакалюк Т. А. Методи оптичного розпізнавання символів : навч. посіб. – Житомир : ДУ «Житомирська політехніка», 2020. – 96 с.

5. OpenCV Documentation. Image Thresholding [Електронний ресурс]. – Режим доступу: https://docs.opencv.org (дата звернення: 02.05.2025).

6. Гонсалес Р., Вудс Р. Цифрова обробка зображень. – 4-те вид. – Київ : Діалектика, 2018. – 1072 с.

7. Ringwald D. Edge Detection in OpenCV 4.0, A 15 Minutes Tutorial: Edge Detection on Still and Moving Objects [Електронний ресурс]. – Режим доступу: https://medium.com/sicara/opencv-edge-detection-tutorial-7c3303f10788 (дата звернення: 02.05.2025).

8. Купчак С. В., Грицуляк В. Б., Долинко Н. П., Халло О. Є. Анатомія і еволюція центральної нервової системи : навч. посіб. – Львів : Світ, 2019. – 112 с.

9. Субботін С. О. Нейронні мережі: теорія та практика : навч. посіб. – Запоріжжя : НУ «Запорізька політехніка», 2020. – 168 с.

10. Горбатюк О. О., Барабан С. В. Інформаційна технологія оптичного розпізнавання тексту : навч. посіб. – Вінниця : ВНТУ, 2023. – 78 с.

11. Олещук О. В., Попель О. Є., Копитчук М. Б. Моделювання нейронних мереж Кохонена на графічному процесорі // Вісник Одеського нац. політехн. ун-ту. – 2013. – № 2. – С. 91–98.

12. The EMNIST dataset [Електронний ресурс]. – Режим доступу: https://www.nist.gov/itl/products-and-services/emnist-dataset (дата звернення: 02.05.2025).

13. Joseph F. J. J. Keras and TensorFlow: A Hands-On Experience // In: Advances in Machine Learning. – Cham : Springer, 2021. – P. 87–112.

14. Koenigshofer K. A. Біопсихологія (OERI) – проект для огляду [Електронний ресурс]. – Режим доступу: https://ukrayinska.libretexts.org (дата звернення: 02.05.2025).

15. Petrosiuk D. V., Arsirii O. O., Babilunha O. Y., Nikolenko A. O. Deep learning technology of convolutional neural networks for facial expression recognition // Applied Aspects of Information Technology. – 2021. – Vol. 4, No. 2. – С. 192–201.

16. Calin O. L. Deep Learning Architectures: A Mathematical Approach. – Cham : Springer, 2020. – 325 p.

17. Ellis H., Logan B. M., Dixon A. K., Bowden D. J. Human Sectional Anatomy: Atlas of Body Sections, CT and MRI Images. – 4th ed. – Boca Raton : CRC Press, 2015. – 276 p.

18. Нейрони головного мозку – будова, класифікація і проводять шляхи [Електронний ресурс]. – Режим доступу: https://sortmozg.com/structure/nejrony-golovnogo-mozga (дата звернення: 02.05.2025).

19. Антонова П. Введення в штучний інтелект: теоретичні основи. – Київ : LAMBERT Academic Publishing, 2019. – 240 с.

20. Класифікація і типи нейронних мереж [Електронний ресурс]. – Режим доступу: http://datascientist.one/class-type-nn/ (дата звернення: 02.05.2025).

21. Аггавал Ч. Нейронні мережі і глибоке навчання. – Київ : Вильямс, 2020. – 356 с.

22. Suzuki S., Abe K. Topological Structural Analysis of Binary Digitized Images by Border Following // CVGIP. – 1985. – Vol. 30, Issue 1. – P. 32–46.

23. Що таке згорткова нейронна мережа [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/309508/ (дата звернення: 02.05.2025).

24. Razzaque M. A., Karim M. R. Hands-On Deep Learning for IoT. – Birmingham : Packt Publishing, 2019. – 310 p.

25. Keras Documentation [Електронний ресурс]. – Режим доступу: https://keras.io/api/ (дата звернення: 02.12.2021).

26. Головко Ст. А. Від багатошарових перцептронів до нейронних мереж глибокої довіри: парадигми навчання і застосування // Матеріали XVII Європ. наук.-техн. конф. з міжнар. участю. – Київ : Політехніка, 2015. – С. 73–79.

27. Ravichandiran S., Saito S., Shanmugamani R., Wenzhuo Y. Python Reinforcement Learning. – Birmingham : Packt Publishing, 2019. – 424 p.

28. Документація по Python [Електронний ресурс]. – Режим доступу: https://www.python.org/doc (дата звернення: 02.05.2025).

29. Шолле Ф. Глибоке навчання на Python. – Київ : Діалектика, 2019. – 384 с.

30. OpenCV: API Documentation [Електронний ресурс]. – Режим доступу: https://docs.opencv.org/2.4/modules/refman.html (дата звернення: 02.05.2025).

31. Girshick R., Donahue J., Darrell T., Malik J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. – Tech Report (v5) [Електронний ресурс]. – Режим доступу: https://arxiv.org/pdf/1311.2524.pdf (дата звернення: 02.05.2025).

32. Машинне навчання на практиці з Python і Keras [Електронний ресурс]. – Режим доступу: https://pythonru.com/primery/mashinnoe-obuchenie-na-praktike-s-python-i-keras (дата звернення: 02.05.2025).

33. Tensorflow API Documentation [Електронний ресурс]. – Режим доступу: https://www.tensorflow.org/api_docs/python/tf (дата звернення: 02.05.2025).