Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Комп'ютерна система розпізнавання рукописних цифр на основі нейронної мережі»

08-54.БКР.011.00.000 ПЗ

Виконав: студент 2 курсу, групи 1КІ-23мс
спеціальності 123 — «Комп'ютерна інженерія»
освітня програма — «Комп'ютерна інженерія»

_____ Курочкін Д. О.

Керівник: к.т.н., доц. каф. ОТ

_____ Колесник І. С.

Рецензент: зав. каф. МБІС

_____ Карпінець В. В.

Допущено до захисту:
завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.

«19» 06 2025 р.

Вінниця ВНТУ 2025 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень бакалавр
Галузь знань — 12 Інформаційні технології
Спеціальність — 123 Комп'ютерна інженерія
Освітня програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. _____О. Д. Азаров
« 21 » березня 2025 р.

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Курочкіну Дмитру Олександровичу

1 Тема роботи «Комп'ютерна система розпізнавання рукописних цифр на основі нейронної мережі», керівник роботи Колесник Ірина Сергіївна, к.т.н., доцент, затверджені наказом вищого навчального закладу від 20 березня 2025р. № 97.

2 Строк подання студентом роботи 13.05.2025 р.

3 Вихідні дані до роботи: роздільна здатність зображення тексту не менше 300 dpi, модель кольорів для представлення зображення — RGB, кількість градацій яскравості зображення — 256.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ; огляд і аналіз існуючих методів розпізнавання рукописних символів; розробка послідовності розпізнавання рукописних цифр; розробка програми розпізнавання рукописних цифр; висновки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): структурна схема програми розпізнавання рукописних цифр, блок-схема послідовності розпізнавання рукописних цифр.

6 Консультанти розділів роботи представлені в таблиці 1.

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-3	к.т.н., доц. каф. ОТ Колесник І.С.	21.03.2025 р.	13.05.2025
Нормоконтроль	асистент каф. ОТ Швець С. І.	13.05.2025 р.	30.05.2025

7 Дата видачі завдання 21.03.2025 р.

8 Календарний план наведено в таблиці 2.

Таблиця 2—Календарний план

№	Назва етапів виконання бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Постановка задачі роботи. Вступ	21.03-23.03.25	
2	Аналіз методів та способів виділення та розпізнавання рукописних символів	24.03-30.03.25	
3	Розробка способу та послідовності розпізнавання рукописних цифр	01.04-21.04.25	
4	Розробка програми розпізнавання рукописних цифр	22.04-05.05.25	
5	Тестування програми розпізнавання рукописних цифр, результати	06.05-10.05.25	
6	Оформлення пояснювальної записки та ілюстративного матеріалу	11.05-12.05.25	
7	Перевірка якості виконання бакалаврської роботи та усунення недоліків	13.05.25	

Студент Дмитро КУРОЧКІН

Керівник роботи к.т.н., доц. Ірина КОЛЕ

АНОТАЦІЯ

УДК 004.932

Курочкін Д. О. Комп’ютерна система розпізнавання рукописних цифр на основі нейронної мережі. Бакалаврська кваліфікаційна робота зі спеціальності 123 — комп’ютерна інженерія, освітня програма — комп’ютерна інженерія. Вінниця: ВНТУ, 2025, 86 с.

На укр. мові. Бібліогр.: 29 назв, рис.11, табл. 1.

В бакалаврській кваліфікаційній роботі виконано огляд існуючих систем, методів та підходів до оптичного розпізнавання рукописних текстових документів, проведено порівняльний аналіз відомих засобів розпізнавання, запропонована послідовність розпізнавання рукописних цифр текстових документів, розроблена блок-схема алгоритму виділення рукописних символів та їх розпізнавання, сформована структура програмного продукту та розроблена програма по розпізнаванню рукописних цифр. Також було проведено експериментальну перевірку запропонованого підходу по розпізнаванню рукописних цифр

Ключові слова: формування ознак символів, рукописні цифри, згорткова нейрона мережа.

ANNOTATION

Kurochkin D. O. A computer system for recognizing handwritten digits based on a neural network. Bachelor's thesis in specialty 123 — computer engineering, educational program — computer engineering. Vinnytsia: VNTU, 2025, 86 p.

In Ukrainian. Bibliogr.: 29 titles, fig. 11, table. 1.

In the bachelor's thesis, the author reviewed existing systems, methods, and approaches to optical recognition of handwritten text documents, conducted a comparative analysis of known recognition tools, proposed a sequence for recognizing handwritten digits in text documents, developed a flowchart for the algorithm for extracting handwritten characters and recognizing them, formed the structure of a software product, and developed a program for recognizing handwritten digits. The proposed approach to handwritten digit recognition was also experimentally tested

Keywords: character feature formation, handwritten digits, convolutional neural network.

ЗМІСТ

ВСТУП.....3
1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ОПТИЧНОГО РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР.....6
1.1 Розпізнавання символів тексту як задача розпізнавання образів.....6
1.2 Структура систем оптичного розпізнавання тексту.....10
1.3 Аналіз методів розпізнавання символів.....16
1.4 Використання штучної нейронної мережі.....20
1.5 Підходи та засоби розпізнавання рукописних символів.....22
2 РОЗРОБКА ПОСЛІДОВНОСТІ ТА ЗАСОБІВ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР.....29
2.1 Послідовність розпізнавання рукописного тексту.....29
2.2 Розробка способу розпізнавання рукописних цифр.....33
2.3 Нейронні мережі для розпізнавання символів.....38
3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР.....48
3.1 Вибір інструментальних засобів програмування.....48
3.2 Розробка архітектури програми розпізнавання.....50
3.3 Розробка програми розпізнавання рукописних цифр.....52
3.4 Склад комп'ютерної системи розпізнавання рукописних цифр.....58
3.5 Перевірка якості роботи програми розпізнавання.....60
3.6 Верифікація програми розпізнавання рукописних цифр.....62
ВИСНОВКИ.....64
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ.....66
ДОДАТОК А Технічне завдання.....69
ДОДАТОК Б Протокол перевірки навчальної (кваліфікаційної) роботи.....73
ДОДАТОК В Графічна частина.....74
ДОДАТОК Г Ілюстративна частина.....76
ДОДАТОК Д Лістинг програми.....78

ВСТУП

Методи автоматичного розпізнавання образів та їх практична реалізація у системах оптичного читання текстів (OCR або OpticalCharacterRecognition) є проявом однієї із найбільш чудових технологій штучного інтелекту. Під системою OCR-системою сприймається автоматичне розпізнавання за допомогою спеціальних програм зображень символів друкованого або ж рукописного тексту й переведення їх у такий формат, що придатний для подальшого оброблення редакторами текстів, текстовими процесорами та інше. Під OCR-системою іноді розуміють пристрої автоматичного читання тексту або ж оптичного розпізнавання символів. У промисловому використанні пристроїв OCR передбачається оброблення документів хорошої й середньої якості, до яких можуть бути віднесені бланки статистичного обліку, бланки різних податкових декларацій, різноманітного виду банківські рахунки та інше [1-3].

Завдання переведення інформації із друкованих паперових на електронні носії є дуже актуальним не тільки у рамках потреб, що виникають в різних системах електронного документообігу. Сучасні інформаційні технології суттєво спрощують доступ до різноманітних інформаційних ресурсів за виконання такої умови, що вони будуть переведені в більш зручний для оброблення електронний вигляд. Найбільш швидким й простим є здійснення операції сканування текстових документів із використанням сканерів. У підсумку отримуємо цифрове зображення початкового документа в вигляді нового графічного файлу. Але значно кращим, в порівнянні з його графічним представленням, є текстове подання цієї інформації. Цей варіант дозволяє досить суттєво зменшити витрати на обсяги зберігання й передачі інформації, та створює додаткові можливості для реалізації усіх можливих сценаріїв використання й аналізу електронних документів. Тому із практичної точки зору значний інтерес представляє саме переведення наявних паперових носіїв у текстові електронні документи.

Дуже складним стає це завдання переведення документів при виконанні операцій автоматизованого розпізнавання рукописних текстів, яке в даний час поки ще так і не вирішено в повному обсязі та вважається одним із найбільш актуальних й складних завдань розпізнавання даних [4].

Значний обсяг даних, що використовується для розпізнавання рукописних символів, вимагає формування нових й подальшого вдосконалення вже існуючих підходів до оброблення сформованих цифрових зображень рукописних символів текстових документів із метою виділення й розпізнавання рукописних символів. Значна сфера використання методів оброблення зображень рукописних текстових документів для розпізнавання у них рукописних символів вказує на **актуальність** цієї проблеми й вимагає подальших досліджень та вдосконалення.

Метою дослідження бакалаврської роботи є вдосконалення методів виділення та розпізнавання рукописних цифр у текстових документах.

Задачі дослідження бакалаврської роботи:

— здійснити аналіз методів та способів побудови систем розпізнавання рукописних цифр;

— запропонувати кращий підхід для розпізнавання рукописних цифр;

— створити послідовність та розробити програму для обробки зображення для виділення та розпізнавання рукописних цифр.

Об’єкт дослідження бакалаврської роботи є процес отримання даних про рукописний текстовий документ шляхом знаходження у виділеному фрагменті зображення рукописних цифр та їх подальшого розпізнавання.

Предметом дослідження бакалаврської роботи є методи оброблення зображення текстового документу для виділення та розпізнавання рукописних цифр.

Методи дослідження бакалаврської роботи: використовувались методи теорії множин для формування множини ознак для розпізнавання, методи диференційного числення для виділенню граничних контурів рукописних символів, методи математичної статистики для виконання аналізу отриманих

результатів розпізнавання. У роботі використано принципи об’єктно-орієнтованого програмування для реалізації запропонованого підходу.

Новизна отриманих результатів бакалаврської роботи полягає у тому, що удосконалено метод оброблення отриманого зображення текстового документу для виділення та розпізнавання рукописних цифр тексту, у якому на першому етапі здійснюється попередня обробка текстового документу та виділяються рукописні цифри, а на другому етапі використовується нейронна мережа для розпізнавання рукописних цифр.

Практичне значення одержаних результатів бакалаврської роботи:

— створено послідовність оброблення цифрових зображень для виділення та розпізнавання рукописних цифр текстових документів;

— розроблено програму оброблення зображення для виділення рукописних цифр та їх розпізнавання у рукописних текстах.

Апробація результатів бакалаврської роботи: підготовлено доповідь на Міжнародну науково-практичну інтернет-конференцію «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)».

За результатами бакалаврської роботи подано до **опублікування** 1 наукову працю [5].

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ОПТИЧНОГО РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР

Даний розділ бакалаврської роботи присвячений аналізу методів виділення та розпізнавання рукописних цифр та обговоренню інструментів, які використовуються для виконання операцій розпізнавання рукописного тексту в цифрових зображеннях.

1.1 Розпізнавання символів тексту як задача розпізнавання образів

Оптичне розпізнавання символів (OCR) — це електронний перетворювач зображень рукописного або друкованого тексту на послідовність кодів, що використовуються для їх відображення в текстовому процесорі. Проблема ефективного розпізнавання тексту відіграє важливу роль у сфері інформатизації різних видів людської діяльності. Представлення інформації в текстовому вигляді, порівняно з графічним представленням, дозволяє значно зменшити витрати на зберігання та передачу даних, а також реалізувати всі методи використання та аналізу електронних документів. Тому з практичної точки зору найбільший інтерес представляє перетворення даних з паперових носіїв у текстові електронні документи [1].

Розпізнавання тексту є одним з аспектів розпізнавання образів. У цьому випадку розпізнавання стосується ознак зображення об'єкта (його форми), які пов'язані з самим об'єктом. Приклади розпізнавання тексту:

Перетворення текстових зображень у цифровий вигляд (скановані книги, статті, журнали) для ефективнішої роботи з цифровими аналогами:

— обробка анкет;

— обробка даних систем тестування знань;

— розпізнавання номерного знаку транспортного засобу.

Оптичне розпізнавання символів — це процес, який реалізує переведення зображення машинописного, друкованого або ж рукописного тексту в набір текстових даних, що представлені у електронному вигляді [1].

Виконання цього процесу розпізнавання символів залежить від початкових вхідних даних, тобто зображення, що аналізується. Стан зображення текстового документа характеризується наявністю або відсутністю різних дефектів та відхилень під час друку та оцифрування документа, що призводить до подальших труднощів у виконанні завдання розпізнавання.

Процес оптичного розпізнавання символів складається з таких етапів: сприйняття, попередня обробка, сегментація та розпізнавання. На кожному етапі проблема вирішується шляхом виявлення наявності певних спотворень та їх зменшення. Аналіз ефективності методів здійснюється за таким принципом: початковий набір даних подається на вхід аналізованого методу, після перетворення даних результат оцінюється на основі деяких заздалегідь вибраних критеріїв та оцінюється продуктивність методу.

Більшість сучасних систем оптичного розпізнавання символів (OCR) аналізують документ за одним із наступних принципів: зверху вниз або знизу вгору. Наприклад, аналіз та вибір об'єктів у документі відбувається зверху вниз, тоді як створення електронної версії відбувається знизу вгору.

Одним з перших і найпоширеніших методів є розпізнавання образів. За допомогою цієї групи методів відскановане зображення швидко перетворюється на зображення (тобто зображення точок), яке потім порівнюється з шаблонами, сформованими в базі даних. Критерієм вибору шаблону є найменша кількість точок, що відрізняються від досліджуваної форми. Шаблон для кожного класу отримується шляхом усереднення зображень символів навчальної вибірки. Перевагою шаблонних методів є їх дуже висока точність розпізнавання символів з різними дефектами (від прогалин у намальованих символах до різних видів склеювання). Основним недоліком цього розпізнавання є те, що воно залежить від шрифту, знайденого на аналізованому зображенні.

Шрифт потрібно знати заздалегідь, інакше не вдасться правильно розпізнати досліджуваний символ за отриманим зображенням. Враховуючи існуюче розмаїття та багатство друкованої продукції, охопити всі відомі

шрифти та варіації під час навчання практично неможливо, тому цей фактор суттєво обмежує універсальність методів розпізнавання символів на основі шаблонів.

Іншою групою методів є методи структурного розпізнавання. Ця група методів використовує інформацію не про точкову карту об'єкта, а про його топологічне представлення: враховується взаємне розташування окремих частин аналізованого об'єкта [3, 4]. До основних переваг методів розпізнавання структурних ознак належить їхня незмінність до типів та розмірів шрифтів [4]. Основною проблемою топологічних методів є виявлення вибраних ознак з деякими дефектами (наприклад, перетини ліній або злиття суміжних ліній), а також низька продуктивність під час роботи [3].

Інша група методів належить до методів ознак. Ця група методів розпізнавання символів найчастіше зустрічається в обробці текстових документів. Їхній аналіз базується на тому факті, що певний N-вимірний вектор ознак пов'язаний із зображенням цілі. Процес розпізнавання полягає в послідовному порівнянні отриманого вектора з набором векторів такої ж розмірності у створеній довідковій базі даних. Якість процесу розпізнавання символів залежить від згенерованих ознак та кількості вибраних ознак. Вектор генерується під час аналізу підготовленого зображення, подібного до цього, шляхом обробки символів у стандартній навчальній вибірці для кожного класу.

Основними перевагами цієї групи розпізнавання ознак є значна простота реалізації, достатня стійкість до змін форми символів, добра здатність до узагальнення та достатньо висока швидкість. Серед недоліків цих методів визначення ознак варто відзначити нестабільність можливих дефектів отриманого зображення, втрату частини інформації про об'єкт під час процесу вибору та вилучення ознак, а також відсутність чітких та єдиних правил процесу формування ознак.Крім того, методи розпізнавання ознак мають ще один серйозний недолік – під час виділення та формування ознак частина інформації про спостережувані символи безповоротно втрачається.

Процес вилучення ознак виконується незалежно від розташування символів, тому інформація про відносне розташування елементів заданого символу втрачається.

Поряд із цими методами існують також альтернативні методи, які не потребують попередньої сегментації, такі як ієрархічні приховані марковські моделі та згорткові нейронні мережі. Через появу нової хвилі популярності класифікаторів нейронних мереж, вони все частіше використовуються в дослідженнях розпізнавання тексту [5, 9]. Головною перевагою використання нейронних мережевих технологій є їхня здатність узагальнювати, використовувати контекстний аналіз та розпізнавати об'єкти на основі навколишніх об'єктів.

Існує кілька важливих питань, пов'язаних з розпізнаванням рукописних та друкованих символів. Найважливішими з них є:

— варіації у розмірах та масштабах, що використовуються, а також типи дизайну символів, різні шрифти;

— різні види креслярських символів;

— спотворення графічних елементів символів.

— спотворення цифрових зображень текстових символів може спричинити наступне:

— світлові ефекти (тіні, відблиски тощо) під час зйомки відеокамерою та різні оптичні спотворення, включаючи нерівномірне освітлення аналізованого документа;

— зміна нахилу письма символів;

— зміна символів або їх частин до очікуваного стану;

— спотворення зображення символу шляхом перетворення зображення в цифрову форму, відоме як «груба» дискретизація;

— Поява різних помилкових симптомів.

Зрештою, система оптичного розпізнавання тексту повинна цифровим чином представляти текстові поля, відображати окремі рядки всередині них,

потім відображати окремі символи, розпізнавати ці символи та водночас бути чутливою (толерантною) до макета, міжрядкового інтервалу та інших параметрів друку.

1.2 Структура систем оптичного розпізнавання тексту

Системи розпізнавання текстових документів, отриманих різними методами, мають певну послідовність операцій. Ця послідовність розпізнавання текстових символів в отриманому документі складається з наступних етапів.

Перший крок — пошук та виділення областей тексту в отриманому документі, видалення різних типів ліній та нетекстових елементів. Для вирішення цієї проблеми використовуються різні типи нейронних мереж, детектор Віоли-Джонса, методи класифікації на основі аналізу ділянок тканин, ознаки Тамура та різні детектори [5]. Однак сучасні системи оптичного розпізнавання символів (OCR) мають труднощі з виявленням та розрізненням тексту будь-якого типу, оскільки вони розроблені для роботи з текстом з однорідною структурою, що рідко зустрічається на зображеннях та відео реальних сцен.

Наступний крок – покращення якості вибраних фрагментів тексту та перетворення їх у двійковий формат. На цьому етапі виконується попереднє пом'якшення зображення, виявлення існуючих видів перешкод та їх видалення. Потім визначається структура вибраного фрагмента тексту та порядок читання.

Далі текстовий документ послідовно сегментується на рядки, потім на слова, і завершується виділенням окремих символів. Пошук рядків базується на визначенні періодичності та узгодженості текстових полів і базується на методі Хафа, методі зв'язних компонент та аналізі горизонтальних, вертикальних та діагональних гістограм.Тепер кожен символ має своє власне тлумачення, і найважливіший крок його розпізнавання виконано. Для військового розпізнавання впроваджено різні класифікатори, одним з яких є

нейронні мережі, і наразі нейронні мережі використовуються найчастіше. Заключним етапом у багатьох системах розпізнавання є перевірка слів за наявним словником. Перевірка словникового запасу виконується на основі традиційних або динамічно згенерованих мовних словників, L-грам, які реалізовані у вигляді списків, дерев та графів [9, 11].

Сучасні системи оптичного розпізнавання символів (OCR) складаються з таких основних блоків, які забезпечують апаратну або програмну реалізацію процесу розпізнавання [5]:

— блок обробки зображень;

— блок сегментації текстових елементів (локалізація та вибір);

— блок вибору ознак;

— блок розпізнавання символів.

Ці блоки системи розпізнавання відповідають послідовним крокам обробки та аналізу зображень, які виконуються послідовно.

Спочатку документ проходить попередню обробку. Коли зображення приймається для розпізнавання, можуть бути різні початкові умови: шум, неправильна яскравість отриманого зображення, розмиття тощо. Тому попередня обробка є важливим етапом процесу розпізнавання символів, що дозволяє згладжувати, нормалізувати, сегментувати та апроксимувати відрізки ліній.

Потім виконується пошук та виділення текстових полів, виділення рядків тексту, і, нарешті, рядки тексту поділяються на окремі слова, а потім на такі знайомі місця, кожне з яких характеризує окремий текстовий символ (сегментація тексту).

Після сегментації (а іноді до або під час сегментації) символи, представлені у вигляді двовимірної матриці пікселів, згладжуються, фільтруються для видалення шуму, масштабуються та фільтруються для виділення числових ознак, що використовуються для подальшого розпізнавання.

Розпізнавання символів відбувається в процесі порівняння вибраних ознак символів зі структурою ознак, сформованою та запам'ятовуваною під час навчання системи на еталонних наборах та реальних прикладах текстових символів.

На заключному етапі семантична або контекстуальна інформація може бути використана для вирішення невизначеності, що виникає під час розпізнавання окремих символів однакового розміру, а також для виправлення слів з орфографічними помилками та навіть словосполучень.

У цьому випадку згладжування належить до більшої групи обробки зображень. Зокрема, широко використовуються морфологічні оператори заповнення та проріджування. Пломбування усуває дрібні розриви та прогалини. Витончення — це процес зменшення товщини лінії, при цьому кожен крок лише «витончує» лінію до області розміром у кілька пікселів. Морфологічний метод виконання таких операцій, заснований на операторах розкриття та стиснення, широко використовується [12].

Також описано спеціальний алгоритм для двійкової фільтрації зображень текстових символів [11], який називається скороченою фільтрацією. «Фрагмент» тут стосується нерівномірності меж символу, яка, по-перше, перешкоджає правильному визначенню його розмірів, а по-друге, спотворює зображення символу та заважає його подальшому розпізнаванню за контурними ознаками.

Технологія OCR також використовує алгоритм бінарної фільтрації під назвою «скорочений запис». Його суть полягає в послідовному спотворенні крайніх елементів (наприклад, верхнього, нижнього, лівого та правого пікселів) у дефектах поблизу меж цілі, що перешкоджає її точній ідентифікації, розпізнаванню за розміром цілі та контурними ознаками.

Подібно до опису видалення ребер з перерізів, можна ввести видалення ребер з нулів. Водночас, замінюючи одиниці нулями та одиницями, «реберні нулі», що відповідають наведеним вище матрицям, також «розбиваються», тобто замінюються одиницями.

Комбінований алгоритм стирання країв працює одночасно як з нулями символів, так і з нулями фону. Після використання цього алгоритму ймовірність точного розпізнавання символів значно зростає. Для нерівномірно вирівняних зображень символів використовується геометрична регуляризація, щоб усунути скелети, спотворити окремі символи та налаштувати їхню довжину та ширину.

Геометрична нормалізація зображень документів передбачає використання алгоритмів, що усувають спотворення та деформації окремих символів, слів або рядків, а також процедур, що нормалізують символи по висоті та ширині після відповідної попередньої обробки.

Ви можете усунути невеликі розриви та прогалини, заповнивши їх. Витончення — це процес зменшення товщини лінії, при цьому кожен крок лише «витончує» лінію до області розміром у кілька пікселів.

Процедури сегментації поділяють зображення документа на окремі області. Як правило, першим кроком є видалення друкованого тексту з графіки та рукописних нотаток. Далі, більшість алгоритмів оптичного розпізнавання символів розбивають текст на символи та розпізнають їх окремо. Це просте рішення насправді ефективніше, якщо текстові символи не перекриваються один з одним. Змішування символів може бути спричинене типом шрифту, який використовується для набору тексту, низькою роздільною здатністю друкуючого пристрою або високим рівнем яскравості, вибраним для відновлення пошкоджених символів.

Рекомендується додатково розділяти текстові поля та рядки на слова, якщо об'єкт, що розпізнається, є словом. Подібний підхід, де розпізнається не окремий символ, а ціле слово, важко реалізувати через велику кількість елементів, які потрібно запам'ятати та додатково розпізнати, але може бути певною мірою корисним та ефективним, коли набір слів у кодовому словнику суттєво обмежений умовами задачі.

Одним із найскладніших завдань у розпізнаванні образів є вилучення ознак для кожного з них. Наразі існує багато різних систем знаків. Завдання

полягає у визначенні ознак, які ефективно відрізнятимуть один клас персонажів від інших у певному застосуванні. Таким чином, метод порівняння зображення зі стандартом вважається дуже ефективним у розпізнаванні символів. У цьому випадку визначається ступінь подібності між зображенням та кожним з існуючих стандартів. Класифікація текстового зображення символу здійснюється за допомогою методу найближчого сусіда. Однак цей метод кореляції має суттєвий недолік: будь-які спотворення в результуючому зображенні цілі можуть перешкодити правильному розпізнаванню цієї цілі, тому використовуються інші методи порівняння зображень.

Вилучення ознак вважається одним із найскладніших і найважливіших завдань у розпізнаванні образів. Для розпізнавання символів можна використовувати багато різних систем ознак. Завдання полягає у визначенні ознак, які ефективно відрізнятимуть один клас персонажів від інших у певному застосуванні.

1.3 Аналіз методів розпізнавання символів

На теперішній час існують такі методи розпізнавання текстових символів: ознаковий, шаблонний, структурний та використання нейронних мереж (рис. Г.1 Додатка Г).

Метод зіставлення зображення за шаблоном. Ця група методів базується на виконанні прямого порівняння отриманих тестових зображень та існуючих опорних символів. У цьому випадку обчислюється ступінь подібності між отриманим зображенням та кожним із стандартів. Класифікація тестового зображення цілі виконується за допомогою методу найближчого сусіда. Існують інші способи порівняння текстових зображень, такі як методи кореляції та послідовні фільтри зображень.

З практичної точки зору, ці методи легко впровадити, і багато комерційних систем оптичного розпізнавання символів (OCR) їх використовують. Однак, при «прямому» впровадженні методів кореляції

навіть невелика темна пляма на зовнішньому контурі цілі може суттєво вплинути на результат розпізнавання. Тому системи, що використовують зіставлення зі зразком, застосовують інші, спеціалізовані методи порівняння зображень для досягнення хорошої якості розпізнавання.

Одна з головних змін в алгоритмі зіставлення зі зразком полягає в тому, що він використовує представлення зразків як набір певних логічних правил.

У групі методів статистичних ознак вибір ознак базується на аналізі статистично різних розподілів. Найбільш відомі методи цієї групи використовують розрахунки моментів та розрахунки перетину.

Різні моменти порядку успішно використовуються в різних галузях машинного зору як дескриптори форми вибраних місць та об'єктів. У випадку розпізнавання текстових символів, як набір ознак використовуються значення моментів набору «чорних» точок відносно деякого вибраного центру. Найчастіше в цьому типі програм використовуються стандартні, центральні та нормалізовані моменти.

Порядок моменту для цифрового зображення, що зберігається у двовимірному масиві, є функцією координат кожної точки на зображенні, а саме:

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q f(x, y), \text{ де} \qquad (1.1)$$

де $p,q \in \{0,1,\ldots,\infty\}$;

M і N є розмірами цифрового зображення по горизонталі та вертикалі;

f(x,y) це яскравість пікселя в точці із координатами (x,y) у зображенні.

Нормалізовані центральні моменти отримують діленням центральних моментів на моменти нульового порядку.

Варто зазначити, що моменти, чутливі до часу, зазвичай забезпечують нижчий рівень безпеки. Центральні та нормалізовані моменти є кращими, оскільки вони менш схильні до змін зображення.

У методі перетину ознаки генеруються шляхом підрахунку кількості та способів перетину зображення символу з вибраними лініями, намальованими під певними кутами. Цей метод часто використовується в комерційних системах, оскільки він нечутливий до спотворення символів та незначних стилістичних змін, дуже швидкий і не вимагає високих обчислювальних витрат.

Зонний метод передбачає поділ області кадру, що містить ціль, на області, а потім використання щільності точок у різних областях як набору характерних ознак.

У методі матриці суміжності частоти спільної появи «чорних» та «білих» елементів у різних геометричних комбінаціях розглядаються як ознаки. Метод локусів ознак показує, скільки разів вертикальні та горизонтальні вектори перетинають відрізки ліній для кожної яскравої точки в області фону цілі. У цій групі є багато інших методів.

Серед сучасних технологій розпізнавання, що базуються на різних перетвореннях, виділяються методи, що використовують фур'є-дескриптори символів, а також дескриптори частоти краю.

Переваги методів, що використовують перетворення Фур'є-Меллена, полягають у тому, що вони інваріантні до збільшення, обертання та переміщення цілі. Основним недоліком цих методів є їхня чутливість до різких стрибків світла на межах, «О» від символу «Q», тощо. Одночасно, при фільтрації шуму на кордонах символу, ця властивість може виявитися корисною.

Структурні ознаки зазвичай використовуються для представлення загальної структури зображення. Вони описують геометричні та топологічні властивості символу. Найпростіший спосіб візуалізувати ідею розпізнавання структурованих текстових символів – це застосувати її до задачі автоматичного зчитування поштових індексів. У таких «ручкових» шрифтах положення кожного сегмента штриха відоме заздалегідь, і символ розрізняється наявністю або відсутністю цілого штриха. Аналогічна

проблема виникає, коли йдеться про моніторинг простих рідкокристалічних дисплеїв. У таких системах вибір структурних компонентів зводиться до аналізу елементів заздалегідь відомого шаблону (набору сегментів, що потребують визначення).

У системах структурного розпізнавання для складніших шрифтів штрихи часто використовуються для ідентифікації таких характерних ознак форми: кінцевих точок, точок перетину сегментів, замкнутих циклів та розташування в рамці, що оточує символ.

Головною перевагою методів структурного розпізнавання є їхня стійкість до змін під малим кутом, збільшення та повороту символу, а також до можливих спотворень та різних стилістичних змін і незначних спотворень шрифтів.

Існують алгоритми, пов'язані зі шрифтами та алгоритми, що незалежні від шрифтів. Ці методи базуються на концепції єдиного типу даних. Тобто, вона зосереджена на вже відомих символах, наприклад, використовується метод малювання [10]. Символьні або символьно-орієнтовані алгоритми використовують переважні символьні дані, які друкують певні літери, цифри або символи. Це означає, що система розпізнавання повинна мати повну копію тексту, надруковану шрифтом. Програма вимірює та аналізує різні характеристики власності та зберігає їх у базі даних еталонних характеристик. Після завершення цього процесу оптичне розпізнавання символів готове до розпізнавання цього символу. Цей процес можна назвати продовженням. Навчання повторюється для деяких символів залежно від ліміту використання.

До недоліків цього методу можна віднести наступні фактори. Алгоритм повинен заздалегідь знати шрифт, який він зчитуватиме, щоб розпізнати його, тобто він повинен зберігати різні характеристики цього шрифту в базі даних. Якість розпізнавання тексту, надрукованого випадковим символом, прямо пропорційна співвідношенню цих символів до символів у базі даних системи розпізнавання. Якщо в процесі навчання є

великий тиск, неможливо охопити всіх персонажів та їхні зміни. Наприклад, сучасні комп'ютеризовані системи друку документів використовують приблизно 200 різних символів, і це число щороку збільшується. Іншими словами, цей фактор обмежує універсальність таких алгоритмів.

Сповідь вимагає блоку конфігурації для певного персонажа. Зрозуміло, що цей пристрій або враховуватиме коефіцієнт помилок в інтегральній оцінці якості розпізнавання, або функцію налаштування шрифту залишить користувачеві [11].

Програма, заснована на алгоритмі символів, розпізнає символи користувача зі специфічними знаннями про групи та відмінності між ними, а також символи, що використовуються для друку документа користувача. Зверніть увагу, що якщо паперовий документ не був створений користувачем і введений анонімно, немає стандартного способу дізнатися, яким шрифтом було надруковано ваш документ. Потреба в спеціалізованих знаннях звужує коло потенційних користувачів і призводить їх до організацій з відповідним персоналом.

З іншого боку, характеристичний підхід має суттєві переваги, саме тому він активно використовується і буде використовуватися в майбутньому. Розробляючи апріорну інформацію про символи, можна створювати високоточні та надійні алгоритми розпізнавання. Загалом, коли створюється алгоритм розпізнавання символів, надійність та якість розпізнавання ознак є величиною, яку можна виразити інтуїтивно та з математичною точністю. Це значення визначається як відстань від опорного символу, наданого програмі під час навчання, до символу, який програма намагається розпізнати.

Інший клас алгоритмів є безособовими, тобто алгоритмами, які заздалегідь не знають символів, заданих на вхідних даних. Такі алгоритми вимірюють та аналізують різні властивості (характеристики), властиві літерам, які не змінюються між символами [12]. У деяких випадках цілеспрямований алгоритм може не мати кривої навчання. У цьому випадку властивості об'єкта вимірюються користувачем, кодуються та зберігаються в

базі даних. Однак на практиці цей варіант рідко вирішує проблему розпізнавання. Більш поширений спосіб створення бази даних – це програмування програми для виведення даних у режимі реального часу.

До недоліків цього підходу можна віднести наступні фактори. Фактично, досяжна якість розпізнавання тексту поступається алгоритмам шрифтів. Оскільки рівень узагальнення у вимірюванні властивостей шрифтів кращий, ніж у алгоритмів, специфічних для шрифтів. Фактично, це означає, що для того, щоб ці алгоритми працювали бездротово, різні припущення та нерівності у вимірюванні властивостей цілі можуть бути в 2-20 разів більшими, ніж самі цілі.

Якщо алгоритм є достатньо та фізично бездоганним без символів, його можна вважати великим успіхом, тобто надійність коефіцієнта розпізнавання природним чином випливає з базового алгоритму. Слід визнати, що в більшості випадків оцінки точності або відсутні, або створюються штучно. Штучне значення означає, що алгоритм не має значної узгодженості з ймовірністю знаходження правильної відповіді.

Переваги цього підходу тісно пов'язані з його недоліками. Головними перевагами є універсальність, адаптивність та простота використання програм.

Різноманітність. З одного боку, такий підхід означає, що його можна використовувати там, де є велика різноманітність символів, які можуть потрапити в систему як оригінали. З іншого боку, завдяки своїй узагальненості, такі алгоритми можуть екстраполювати знання, накопичені поза межами методу навчання, тобто вони завжди можуть розпізнавати символи, які знаходяться далеко від символів у методі навчання.

Процес навчання алгоритмів на основі шрифтів загалом простіший і простіший у тому сенсі, що вибірка не поділяється на різні класи (символи, крапки тощо). Однак, не обов'язково, щоб ці класи в об'єктній базі даних підтримували різні умови співіснування. Ще однією формою адаптації є створення майже автоматичних програм навчання.

Зручність використання програми. Якщо програма базується на алгоритмах, що залежать від шрифтів, користувачеві не потрібно нічого знати про сторінку, яку він хоче зберегти в пам'яті комп'ютера, і передавати ці знання програмі. Це також спрощує інтерфейс користувача програми через відсутність опцій та діалогових вікон, що використовуються для навчання та керування базою даних програми. У цьому випадку процес розпізнавання може бути представлений користувачеві як «чорна скринька» (користувач не має можливості контролювати або змінювати хід процесу розпізнавання). В результаті, кількість потенційних користувачів розширюється, зокрема й тих, хто має мінімальні знання комп'ютера.

1.4 Використання штучної нейронної мережі

Для розпізнавання символів часто використовуються штучні нейронні мережі. Алгоритми, розроблені для нейронних мереж розпізнавання символів, часто будуються так: зображення символу, яке є вхідним зображенням для читання, зменшується до певного стандартного розміру. Зазвичай зображення зменшується до 16x16 пікселів.

Кількість вихідних параметрів нейронної мережі така сама, як кількість розпізнаних символів. Результатом розпізнавання є символ, що відповідає найвищому значенню джерела нейронної мережі. Підвищення надійності таких алгоритмів зазвичай передбачає пошук більш інформативної функціональності введення або ускладнення структури нейронної мережі.

Від вибору структури та параметрів нейронної мережі значною мірою залежить надійність виявлення та потреба програми в обчислювальних ресурсах. Зображення зображень зменшуються до одного розміру (28x28 пікселів). Отримане зображення надсилається на вхід нейронної мережі, яка має три внутрішні шари і 10 вузлів верхнього шару. Нижні шари мережі не пов'язані між собою. Нижні вузли мають загальний набір ваг. На думку розробників, це повинно підвищити здатність нижніх шарів мережі

розрізняти основні функції зображення. Невикористані масштаби видаляються, щоб збільшити здатність мережі до узагальнення та зменшити кількість необхідних обчислень і пам’яті. Це зменшує кількість незалежних параметрів на чотири. Навчання нейронної мережі виконується на наборі з 7300 символів, тест з багатьох 2000 символів. Похибки виявлення становлять приблизно 1% для навчального набору і 5% для тесту [14].

Замість значень яскравості у вузлах нормованої сітки в якості вхідних параметрів нейронної мережі можна використовувати значення, що характеризують різницю яскравості. За допомогою цих вхідних параметрів ви можете краще визначити межі літер. Об’єкти, що сприймають, зменшуються до 16x16 пікселів. Потім вони додатково обробляються, щоб виділити ділянки з найбільшою різницею в яскравості. Одним із широко використовуваних методів для підвищення точності розпізнавання є одночасне використання кількох модулів розпізнавання та подальша інтеграція результатів (наприклад, за допомогою координації). Дуже важливо, щоб алгоритми, які використовуються цими модулями, були максимально незалежними. Цього можна досягти за допомогою модулів розпізнавання, які в основному використовують різні алгоритми розпізнавання, а також спеціального відбору навчальних даних.

Як правило, алгоритм розпізнавання заснований на розділенні сітки із зображенням букв основних ознак і на подальшому використанні штучної нейронної мережі для оцінки близькості вхідного зображення до символів заданого набору букв. Результатом є набір оцінок, які відображають ступінь близькості розпізнаного символу до символів заданого набору символів [16].

Розпізнаваний набір символів може містити літери та цифри. Вхідні дані для розпізнавання символів перетворюються у вимір. Особливістю реалізованого алгоритму є використання нейронної мережі з великою кількістю вхідних функцій. Оригінальне зображення виділяє основні особливості, що характеризують різницю яскравості у вузлах сітки. Нейронна мережа має внутрішній шар, який містить 100 вузлів і пов’язаний

універсально, тобто кожен внутрішній X_i вузол підключений до всіх вхідних вузлів, а кожен вузол вищого рівня підключений до всіх вузлів внутрішнього рівня. Щоб зменшити кількість обчислень розпізнавання, для кожного зображення розпізнаного символу використовуються не всі вхідні символи, а лише частина з них, тобто X_i вектор вхідних параметрів нейронної мережі сильно розбавлений [17].

Для роботи системи розпізнавання тексту необхідно вставити цифровий файл зі відсканованим текстом (текстовий елемент). Наступним етапом системи розпізнавання тексту є так звана «сегментація» та попередня обробка даних. На цьому етапі роботи відбувається вирівнювання елементів тексту. Текстовими елементами можуть бути фотографії з певними фрагментами тексту, частинами відсканованих сторінок і подібними параметрами цифрового макета. Після вирівнювання всі ці елементи мають бути звільнені від непотрібних частинок, тобто текст повинен складатися виключно із «знайомих» для системи символів, а решта – для того, щоб очистити поле зору системи або просто пропустити такі елементів. Потім фон та інші елементи ізолюють, щоб їх можна було прочитати як білий елемент, а потрібний текст вводиться в систему лише як чорний текст. Потім система розпізнавання розділяє стовпці та рядки, розбиває безперервний текст на слова та змінює розмір зображення для зручного порівняння зі згенерованими «еталонними» словами та символами. Завершальним етапом є додаткове використання словника.

1.5 Підходи та засоби розпізнавання рукописних символів

В останні роки розпізнавання рукописних символів стало однією з найперспективніших галузей досліджень в обробці зображень та розпізнаванні образів. Розпізнавання рукописного тексту поділяється на два типи: офлайн та онлайн. Під час розпізнавання в автономному режимі рукописні символи зазвичай скануються за допомогою оптичного сканера та доступні у вигляді готового зображення символу. Онлайн-методи

перевершують конкуруючі офлайн-методи через те, що онлайн-методи мають часові обмеження. Однак, такі програми, як поштові замовлення, обробка банківських чеків, читання документів та розпізнавання поштових адрес, вимагають автономних методів розпізнавання рукописного тексту. Таким чином, автономне розпізнавання рукописного тексту продовжує бути активною та перспективною галуззю досліджень і вимагає вивчення нових методів, які покращать точність розпізнавання символів.

Офлайн розпізнавання відбувається для тексту, написаного вже на папері. Текстовий документ, сторінка книги тощо, надсилаються як скан або фотографія. Онлайн є більш складним методом, оскільки можна відстежувати процес написання тексту онлайн та на його основі створювати алгоритм розпізнавання. Складність розпізнавання почерку полягає в різноманітності письма, форми та розміру літер у різних мовах. Крім того, текстовий папір може мати «шум»: дефекти паперу, сторонні плями, які ще більше ускладнюють весь процес.

Вилучення ознак є одним з ключових складових створення послідовності розпізнавання рукописного тексту. Ефективність системи розпізнавання значною мірою залежить від вибору ефективних ознак та вибору відповідного класифікатора. Етап вилучення ознак аналізує розділ (частину) тексту та вибирає набір ознак, які можна використовувати для унікальної ідентифікації цього розділу. Отримані ознаки потім використовуються як вхідні дані для класифікатора почерку. Метою вилучення ознак є ідентифікація закономірностей, використовуючи мінімальну кількість ознак, які ефективні для розпізнавання класів закономірностей. Методи вилучення ознак можна умовно розділити на три основні категорії: статистичні, структурні та глобальні трансформації.

Статистичний метод подає зображення документа за допомогою розподілу точок. Зображення символу розділяється на блоки, які називаються зонами. Якщо горизонтальну площину розділити на m ліній, а вертикальну площину на n ліній, то кількість утворених зон дорівнює mxn. Потім

обчислюється середня інтенсивність рівня сірого для заданої області зони. Тип символу однозначно ідентифікує кожен набір символів і тому використовується як ознака.

У структурному підході символами є лінії, петлі, перетини, кінцеві точки тощо, що стає набором топологічних примітивів. Ці ініціали потім формуються за допомогою властивостей з'єднань або графів. Ці графіки відображаються двома способами. Перший метод використовує координати зображення символу. Другий метод полягає в абстрактному зображенні вертикалей, що відповідають ударам та штрихам. Дерева використовуються для представлення символів на основі різних функцій, які мають ієрархічний зв'язок.

Методи глобального перетворення переводять зображення в компактну форму, що зменшує розмір об'єкта. Зміну форми зазвичай можна представити лінійною комбінацією чітко визначених синусів і косинусів. Перші N коефіцієнтів перетворення Фур'є використовуються як характеристичний вектор певного символу. Оскільки існує зворотне перетворення, зображення можна відновити.

Існує щонайменше два підходи до розпізнавання рукописного тексту, які дають надійні результати: використання прихованої моделі Маркова та штучної нейронної мережі (ШНМ). На практиці також використовується гібридний підхід, коли обидва методи поєднуються одночасно.

Це включає підготовку рукописного документа для аналізу, його перевірку та коректуру. Виконується проміжна бінаризація, в цьому випадку відбувається процес відділення тексту від фонового об'єкта. Результатом є чорний текст на білому фоні. Зменшення шуму: видаляє артефакти із зображення, не впливаючи на написаний текст.

Відбувається поділ тексту на рядки, слова та символи для легшого розпізнавання за допомогою штучних нейронних мереж. Чим менше рядків у тексті, що нагадують прямі лінії, тим гірше працює алгоритм сегментації ліній. Початкова сегментація слів відбувається, коли відстань між словами

перевищує розмір літер. Змінюючи кут нахилу, коригують форму літер. Кут скосу: кут між вертикаллю та літерою. Фаза корекції спрямована на зменшення цього кута.

Вхідні дані для класифікатора у вигляді ШНМ можуть бути у вигляді окремих слів та символів. ШНМ складається з шарів. Саме тут і відбувається вся магія та математика: спочатку використовуються згорткові та конволюційні операції, які є одним із типів LSTM, mdlstm, IDCN та рекурентних нейронних мереж (RNN). Суть обробки полягає в тому, що кожна частина зображення множиться на поелементну матрицю згортки (ядро), а результат підсумовується та записується в тому ж місці вихідного зображення (формується карта ознак). Процес стиснення може значно зменшити розмір зображення. Групування пояснюється наступним чином: якщо деякі об'єкти вже були ідентифіковані в попередньому процесі розв'язання, таке детальне зображення більше не потрібне для подальшої обробки та стискається з більш детальним зображенням — розмір створених карт об'єктів зменшується. Кінцевий результат також залежить від бази даних — набору шаблонів зображень для кожної літери, написаної різними шрифтами.

На теперішній час доступний широкий спектр програмних засобів, які можна використовувати для аналізу, створення, редагування та оброблення зображень із текстовими фрагментами. Розглянемо наявне програмне забезпечення, яке може розпізнавати й перетворювати документи із текстом із графічного формату подання у текстовий формат.

Найбільш поширеними на теперішній час є такі системи: ABBYY FineReader, CuneiForm, Tesseract OCR, Readiris, Acrobat, Scanitto Pro і RiDoc та інші [8-12].

ABBYY FineReader 12 — це професійне програмне забезпечення, яке може розпізнавати текст із графічних файлів і конвертувати його в різні редаговані формати. Програма визначає і переводить текст у новий формат. Це зменшує витрати на обробку, оскільки не потрібно перечитувати текст

вручну. Завдяки Abbyy FineReader можна конвертувати файли tiff, jpeg, pdf, doc, xlsx та txt. Результати тестування показують, що ця система дуже добре обробляє та перетворює документи з графічного формату на текстовий. Це програмне забезпечення показало себе як продукт високої якості. У AbbyFineReader можна використовувати такі системи: SimpleOCR; OmniPage; Reading і Writing

Переваги: висока якість розпізнавання тексту; широкий вибір форматів вхідних і вихідних документів; простий і зрозумілий інтерфейс користувача. Недоліки: програмна система є досить коштовною, а вихідний код програми недоступний.

Free Online Service OCR — безкоштовний онлайн-сервіс для розпізнавання тексту в різних форматах. Простий у використанні та не вимагає встановлення програми на комп'ютер. Цей продукт підтримує багато мов. Можна зберегти текст у будь-який із трьох форматів: звичайний текст, Microsoft Word і Microsoft Excel. У більшості випадків цих форматів достатньо для швидкого розпізнавання тексту.

Переваги: багато мов для розпізнавання, швидкий доступ до пристроїв із доступом до Інтернету та простота використання. Недоліки: цей продукт не містить повних функцій, на відміну від завантаженої програми оптичного розпізнавання символів.

CuneiForm — це програма, яка може розпізнавати текстові документи та перетворювати їх у редаговану форму. Цей продукт можна отримати безкоштовно для будь-якого користувача. Програми можна використовувати для отримання файлу у новому форматі зі зміненим текстом. Дана програма не підтримує деякі з найпоширеніших форматів файлів, таких як Microsoft Word (розширення .doc) і Acrobat Reader DC (розширення .pdf). Виходячи з вищезазначеного цей продукт буде менш зручним та ефективним у використанні, ніж аналог ABBYY FineReader 12.

Результати тестування показують, що OCR CuneiForm є низькоефективним програмним забезпеченням для оптичного розпізнавання

тексту. Програма може не розпізнати деякі об’єкти, а розпізнані слова можуть виявитись набором незрозумілих символів. Середня кількість розпізнаних слів трохи менше дев’яносто відсотків.

Переваги: ця програма безкоштовна, а вихідний код програми доступний. Недоліки: погана якість розпізнавання; аналіз виконується без налаштування активних полів для розпізнавання, тому програми розпізнають текст лише у тих місцях графічного файлу, які визначила програма.

У Readiris є велика перевага: ефективна система для читання рукописного тексту. Acrobat розроблено для інших цілей, але вбудована функція розпізнавання працює добре, навіть якщо вона менш практична, ніж інші рішення в нашому прикладі.

Scanitto Pro і RiDoc можуть швидко створити документ, зображення або аркуш паперу з символами. Утиліта не вимагає багато ресурсів і є ощадливою.

Програма OCRopus зможе точно визначати текст у графічному файлі та конвертувати його у звичайний текстовий формат для подальшого редагування. Програма може ідентифікувати не тільки друкований текст, але й рукописні матеріали. OCRopus доступний лише для GNU/Linux. В даний час OCRopus використовує лише командний рядок для прийому вказівок на вхідні зображення з текстом і виводу даних у форматах hOCR або TXT [17].

FreeOCR — це безкоштовна програма, яка може розпізнавати відсканований текст. Вона може працювати з файлами зображень, pdf-файлами та безпосередньо зі сканером. Основним плюсом програми є те, що вона повністю автоматизована і не потребує будь-яких налаштувань. Підтримуються такі формати, як PDF, JPEG, GIF, TIFF і BMP. Крім того, існує обмеження на десять зображень на годину. Система здатна розпізнавати більшість східноєвропейських мов, українську в тому числі.

Система розпізнавання тексту з відкритим кодом (OCR) Tesseract OCR доступна за ліцензією Apache 2.0. Систему можна використовувати безпосередньо для видалення друкованого тексту із зображень або через API

для програмістів. Хоча Tesseract не має вбудованих графічних інтерфейсів, деякі можна знайти на сторонніх сайтах розробників. За допомогою оболонки Tesseract сумісний з багатьма мовами програмування та фреймворками.

Системи розпізнавання тексту з відкритим кодом та вільним розповсюдженням Tesseract OCR, CuneiForm та Free Online Service OCR розпізнають тільки друкований текст. Інші типи систем, такі як ABBYY FineReader, Acrobat Reader, Readiris Pro є комерційними і вимагають значних коштів для свого використання. Тому постає питання розробки програмного додатку для розпізнавання рукописних цифр.

Більшість програмного забезпечення, широко доступного на ринку, не може повністю вирішити проблему розпізнавання рукописних цифр. Хоча рукописний текст можна розпізнати, розпізнавання супроводжується багатьма помилками та неточностями. Крім того, більшість існуючих програм є платними, що ускладнює їх використання.

У цьому розділі бакалаврської роботи був виконаний огляд існуючих систем, методів та підходів до оптичного розпізнавання рукописних текстових документів, проведено порівняльний аналіз відомих засобів розпізнавання та вказано на деякі недоліки й обґрунтовано необхідність розробки програмного продукту для розпізнавання рукописних цифр.

2 РОЗРОБКА ЗАСОБІВ РОЗПІЗНАВАННЯ РУКОПИСНИХ СИМВОЛІВ

У даному розділі бакалаврської роботи розроблена послідовність розпізнавання рукописних цифр у текстових документах та розглянута згорткова нейронна мережа для їх розпізнавання.

2.1 Послідовність розпізнавання рукописного тексту

Розпізнавання рукописного тексту відбувається в кілька етапів. На початковому етапі здійснюється попередня обробка отриманого текстового документа. Обробка зображень: на цьому етапі зображення обробляється для покращення його якості та придатності для сегментації.

Сегментація: На цьому етапі текст на зображенні виділяється та поділяється на частини. Зазвичай текст обробляється ієрархічно: спочатку вибираються окремі рядки, потім окремі слова, а потім символи або частини символів.

Вилучення ознак: На цьому етапі створюється опис ознак сегментів, визначених під час фази сегментації.

Класифікація: На цьому етапі, на основі опису ознаки, створеного на етапі вилучення ознак, система вирішує, до якої групи класифікувати елемент, вибраний на етапі сегментації, до певного класу.

Обробка результатів (переробка): На цьому етапі структура остаточного тексту базується на результатах класифікації вибраних частин тексту.

Обробка зображень. Під час попередньої обробки виконується покращення зображення, яке призводить його до форми, оптимальної для подальшої автоматичної обробки [5]. Додаткове покращення зображення для розпізнавання тексту включає видалення дефектів зображення та розмиття фону [6].

Видалення дефектів Дефекти видаляються за допомогою стандартних методів обробки зображень. Найпоширенішим фільтром для видалення шуму

є гауссівський фільтр для придушення високочастотного шуму та медіанний фільтр для видалення шуму [7].

Відокремлення тексту від фону є окремим випадком задачі представлення об'єкта на зображенні. Задачу можна сформулювати так: починаючи з текстового зображення A, нам потрібно побудувати бінарне зображення B такого ж розміру. з B (i, j) = 1, якщо піксель (i, j) належить до тексту на A, якщо ні, тоді B (i, j) = 0. Це перетворення зображення дає можливість подальшого використання аналізу з'єднаних компонентів, контурів, скелетів тощо.

Найпоширенішим методом видалення тексту з фону є бінарне порогове значення. Для зображення I (i, j) — це яскравість пікселя із координатами (i,j). Порогова бінаризація зображення, це таке покрокове перетворення f (i,j), що f(i,j) = 1, якщо I (i, j) ≥ d, f (i, j) = 0, якщо I (i, j) <d, де d називається порогом бінаризації.

Зазвичай на гістограмі яскравості текстового зображення спостерігаються два піки: високий пік у світлій області пікселів (на фоні, тобто папері) та нижчий пік у темній області пікселів (у тексті). Отже, виникає проблема знаходження оптимального значення між двома піками гістограми, наприклад, піксель зі значенням вище цього значення (фон) як чорний, а нижче (текст) як білий (таке «перетворення» кольору для спрощення використання багатьох алгоритмів у майбутньому). Існують добре досліджені методи вирішення цієї проблеми, такі як метод Оцу [8] та його варіанти. Результатом обробки зображення є двійкове зображення білих чисел на чорному фоні, яке відповідає вихідному зображенню.

Сегментація. На етапі сегментації з зображення, отриманого під час попередньої обробки, вибираються окремі числа. Виходом кроку сегментації має бути послідовність ідентичних зображень, які подаються на вхід процедури вилучення ознак. Ця стратегія, яка називається стратегією неявної сегментації, вирішує проблему розпізнавання рукописної послідовності цифр краще, ніж явна сегментація, де вилучення цифр, вилучення ознак та

класифікація виконуються паралельно [9]. Позначте групи чисел. Після того, як текст буде відокремлено від фону під час попередньої обробки, числові групи можна витягти за допомогою аналізу конкатенованих компонентів. Якщо вважати білі пікселі кутовими точками графіка та з'єднати кутові точки, що відповідають сусіднім пікселям, ребрами, можна буде вибрати точки з'єднання, що відповідають одному або кільком номерам перетину. Існують стандартні та добре відомі алгоритми пошуку для ідентифікації пов'язаних компонентів, такі як пошук у глибину та пошук у ширину.

Після того, як текст буде відокремлено від фону під час попередньої обробки, числові групи можна витягти за допомогою аналізу конкатенованих компонентів. Якщо вважати білі пікселі кутовими точками графіка та з'єднати кутові точки, що відповідають сусіднім пікселям, ребрами, можна буде вибрати точки з'єднання, що відповідають одному або кільком номерам перетину. Існують стандартні та добре відомі алгоритми пошуку для ідентифікації пов'язаних компонентів, такі як пошук у глибину та пошук у ширину.

Розділення об'єднаних символів. Після того, як ви позначили компонент з'єднання на схемі зображення, вам потрібно визначити, чи складається цей компонент з одного чи кількох символів, і якщо їх більше, як їх розділити. Компонент зв'язності зазвичай сегментується на числа у два етапи: спочатку генеруються можливі типи сегментації компонента зв'язності, а потім вибирається найкращий з них. Побудова можливих методів сегментації зазвичай здійснюється за допомогою евристичних методів. Вибір найкращого методу сегментації може здійснюватися як підсистемою сегментації (у цьому випадку для вилучення ознак та вхідних даних для підсистеми класифікації використовується лише один набір вхідних даних), так і підсистемою класифікації (у цьому випадку кілька вхідних даних). Дані надаються системою вилучення ознак та підсистемою класифікації, а рішення про те, яка сегментація була виконана правильно, ґрунтується на результатах розпізнавання.

Для побудови прозорої сегментації часто використовуються два типи даних: контурні та скелетні. Межа об'єкта - це множина всіх пікселів, що належать об'єкту та мають сусідні пікселі, які не належать цьому об'єкту. Точки в правому та лівому кутах поділяють його на верхній та нижній шляхи. Опукла область контуру називається долиною, а увігнута область – пагорбом.

Методи сегментації надаються на основі аналізу попередньо визначених меж компонентів зв'язності [14]. У цих методах, виходячи з деяких припущень щодо контурів сполучних компонентів, вибираються потенційні точки перетину, через які проводяться лінії перетину відповідно до певних умов. Точки перетину створюються там, де відстань між верхнім і нижнім контурами перевищує порогове значення, а найкраще перетин вибирається як найкоротший шлях у графі, перпендикулярний до можливих перетинів. У [15] можливі точки перетину евристично вибрані як точки, де профіль стає максимально повернутим за годинниковою стрілкою при русі проти годинникової стрілки.

Вилучення ознак. Нехай X — множина розглянутих об'єктів, Y — скінченна множина. Існує функція y: X → Y, значення якої відоме тільки на кінцевій вибірці $X = \{(x_1, y_1). ... ..., (x_m, y_m)\}$. Необхідно побудувати функцію K: X → Y для функції y. Функцію K називають класифікатором.

Загалом, систему розпізнавання рукописного тексту можна представити як три підсистеми: підсистему вилучення ознак, підсистему розпізнавання та підсистему прийняття рішень, пов'язану з цим об'єктом.

Вилучення ознак — це процес перетворення вхідних об'єктів в однорідну, компактну та зручну форму шляхом видалення більшої частини інформації з об'єкта, яка мало впливає на класифікацію. Метод вилучення ознак залежить від типу об'єктів та їхнього вихідного коду і вибирається вручну.

Підсистема розпізнавання — це алгоритм, який розділяє простір ознак на частини, що відповідають заданим класам.

Неможливо розробити, протестувати та налагодити систему введення тексту без створення бази даних символів. База даних шрифтів є одним з ключових компонентів системи розпізнавання рукописного тексту. Відображення символів у цій базі даних залежить від того, як розпізнається зображення рукописного тексту. Інтерпретація розрахованих ймовірностей відбувається через процес прийняття рішень, окремий від розпізнавання.

Рукопис обробляється у два етапи. На першому кроці вихідне зображення зводиться до універсальної векторної форми. Для цього необхідно реалізувати такі зміни, які призведуть до незмінного представлення даних:

— перетворити зображення у градації сірого;

— зменшити видимість чорно-білого зображення;

— аналізувати та видаляти шум з отриманих текстових зображень;

— виконувати скелетонізацію чорно-білих зображень;

— виконувати векторизацію зображень.

На цьому етапі проблема сегментації рукописного тексту у вступі базується на структурному підході, який передбачає опис складних об'єктів за допомогою простіших сутностей [3].

Рядки виступають у ролі робочого простору, який потім потрібно розділити на слова. Через високий рівень узгодженості почеркових моделей більшості людей, поділ слів на літери є виснажливим завданням.

Другий крок — класифікація зображень. Завдання класифікації полягає у визначенні того, чи належить вхідне зображення до одного з попередньо визначених класів, або у порівнянні векторного представлення (інтерпретації) вхідного рукописного символу з векторним представленням опорного символу в базі даних.

2.2 Розробка послідовності розпізнавання рукописних цифр

Послідовність розпізнавання рукописних чисел схожа на послідовність розпізнавання друкованих символів у багатьох кроках. Ця послідовність базується на серії дій для виділення рядків тексту, пошуку груп чисел у числовому форматі та розділення цих груп на окремі пробіли з числовим форматом.

Послідовність отримання зображень для розпізнавання рукописних цифр складається з кількох основних етапів, а саме: вибір області фрагмента тексту на отриманому зображенні, попередня обробка вибраних фрагментів зображення, поділ фрагментів на лінії, вилучення цифр, чисел, ознак, класифікація та розпізнавання рукописних символів, а також остаточна обробка (повторна обробка). Послідовність списку для обробки зображення з фрагментом тексту показано на рисунку 2.1. Ми використовуватимемо ці кроки як основу для створення технологічної послідовності обробки зображень для маркування та розпізнавання рукописних цифр.

Інструмент розпізнавання рукописних цифр приймає зображення текстового документа як початкове вхідне зображення. Зображення має бути у форматі BMP, JPEG, PNG або іншому. Таке зображення можна отримати за допомогою сканера, цифрової камери або будь-якого пристрою для створення та введення цифрових зображень.

Наступний крок — працювати на випередження. Він складається з кількох операцій, що виконуються над отриманим вхідним зображенням. Це значно покращить та сегментує зображення. Послідовність операцій, що виконуються на зображенні, показано на рисунку 2.1.

Процедура бінаризації використовує методи глобальних обмежень для бінаризації зображення у градаціях сірого. Розширення країв двовимірного зображення досягається за допомогою операції відбілювання. Збільшення зображень та заповнення можливих прогалин виконується на останніх двох етапах для формування попередньо оброблених зображень, придатних для остаточного етапу сегментації.

Для використання на етапі опису символічного аналізу необхідно пройти кілька етапів попередньої обробки. Метою попередньої обробки є отримання даних, придатних для системи оптичного розпізнавання символів. Основна функція попередньої обробки полягає у фільтрації шуму, покращенні контрастності та усуненні зв'язування результуючого зображення.

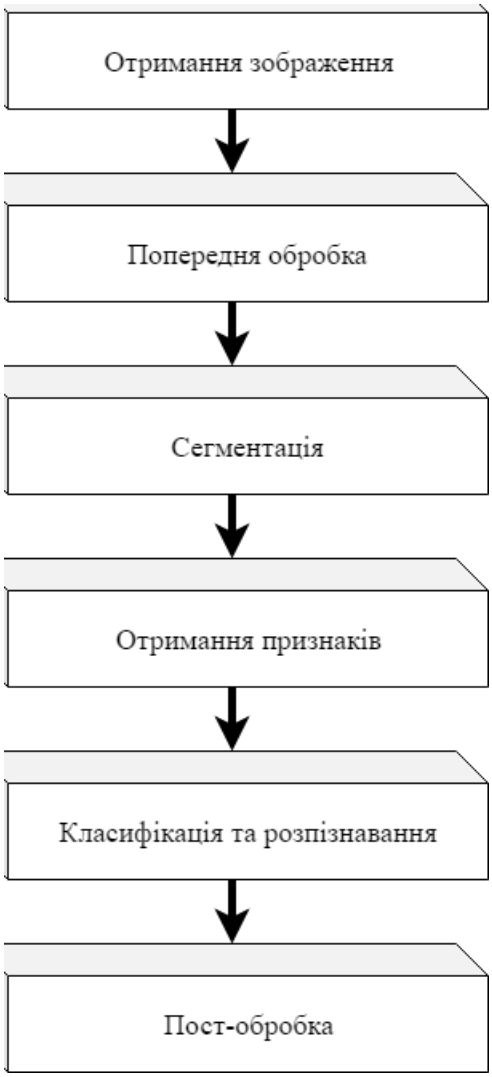

Рисунок 2.1 — Загальна схема розв’язання задачі розпізнавання

Під час сканування документів на відсканованих зображеннях може бути додатковий шум, і ці зображення низької якості вплинуть на наступний етап обробки документів. Тому для покращення якості зображення перед переходом до наступного етапу обробки документа необхідний етап

попередньої обробки. Шум спричинений помилковими окремими сегментами, причиною можуть бути довгі розриви рядків тощо, тому важливо усунути всі ці помилки, щоб отримати кращу інформацію. У зображеннях є багато видів звуків. Додатковий звук називається «звук солі та перцю», який складається з чорних і білих точок, розкиданих по всьому зображенню, що зустрічаються в більшості документів. Методи зменшення шуму можна розділити на дві основні групи: фільтрація та морфологічна обробка.

Мета фільтра полягає в усуненні шуму та зменшенні надлишкових точок, які зазвичай спричинені нерівними поверхнями запису та/або низькою частотою дискретизації пристрою збору даних. Морфологічні операції часто використовуються як інструменти обробки зображень для вилучення корисних ознак зображення для представлення та опису форми області. Процеси редагування можна успішно використовувати для видалення шуму із зображень документів, спричиненого поганою якістю паперу та чорнила, а також зовнішніми рухами рук. Бінарне представлення отриманих зображень символів є важливим кроком у розпізнаванні символів офлайн. Хороша двійкова система числення полегшує ділення та розпізнавання рукописних чисел.

Сегментація зображень розділяє послідовність символів на окремі підсторінки символів. У запропонованій системі попередньо оброблене вхідне зображення розділяється на окремі символи, і для присвоєння номера кожному символу використовується процес маркування. Ця позначка надає інформацію про кількість символів на зображенні. Ребра характеризують межі об'єктів і тому корисні для сегментації, реєстрації та ідентифікації об'єктів. Виявлення країв зображення значно зменшує обсяг даних та фільтрує непотрібну інформацію, зберігаючи важливі структурні особливості зображення.

Існує багато способів визначення країв. Однак більшість різних методів можна розділити на дві категорії: градієнтні методи та методи Лапласа.

Градієнтний метод ідентифікує ребра, знаходячи найвище та найнижче значення першої згенерованої форми. Метод Лапласа знаходить ребра, шукаючи перетини нуля у другій похідній.

Для використання на етапі опису символічного аналізу необхідно пройти кілька етапів попередньої обробки. Метою попередньої обробки є отримання даних, придатних для системи оптичного розпізнавання символів. Основна функція попередньої обробки полягає у фільтрації шуму, покращенні контрастності та усуненні зв'язування результуючого зображення.

Під час сканування документів на відсканованих зображеннях може бути додатковий шум, і ці зображення низької якості вплинуть на наступний етап обробки документів. Тому для покращення якості зображення перед переходом до наступного етапу обробки документа необхідний етап попередньої обробки. Шум спричинений помилковими окремими сегментами, причиною можуть бути довгі розриви рядків тощо, тому важливо усунути всі ці помилки, щоб отримати кращу інформацію. У зображеннях є багато видів звуків. Додатковий звук називається «звук солі та перцю», який складається з чорних і білих точок, розкиданих по всьому зображенню, що зустрічаються в більшості документів. Методи зменшення шуму можна розділити на дві основні групи: фільтрація та морфологічна обробка.

Мета фільтра полягає в усуненні шуму та зменшенні надлишкових точок, які зазвичай спричинені нерівними поверхнями запису та/або низькою частотою дискретизації пристрою збору даних. Морфологічні операції часто використовуються як інструменти обробки зображень для вилучення корисних ознак зображення для представлення та опису форми області. Процеси редагування можна успішно використовувати для видалення шуму із зображень документів, спричиненого поганою якістю паперу та чорнила, а також зовнішніми рухами рук. Бінарне представлення отриманих зображень символів є важливим кроком у розпізнаванні символів офлайн. Хороша

двійкова система числення полегшує ділення та розпізнавання рукописних чисел.

Сегментація зображень розділяє послідовність символів на окремі підсторінки символів. У запропонованій системі попередньо оброблене вхідне зображення розділяється на окремі символи, і для присвоєння номера кожному символу використовується процес маркування. Ця позначка надає інформацію про кількість символів на зображенні. Ребра характеризують межі об'єктів і тому корисні для сегментації, реєстрації та ідентифікації об'єктів. Виявлення країв зображення значно зменшує обсяг даних та фільтрує непотрібну інформацію, зберігаючи важливі структурні особливості зображення.

Існує багато способів визначення країв. Однак більшість різних методів можна розділити на дві категорії: градієнтні методи та методи Лапласа. Градієнтний метод ідентифікує ребра, знаходячи найвище та найнижче значення першої згенерованої форми. Метод Лапласа знаходить ребра, шукаючи перетини нуля у другій похідній.

Потім ми розділяємо сторінку, шукаючи області з білим і чорним текстом. Внутрішня сегментація — це процес, який намагається розділити рядок на менші частини з окремих символів. Хоча ці методи значно просунулися за останнє десятиліття і з'явилося багато різних методів, поділ рукописних шрифтів на символи залишається відкритою проблемою.

Крок класифікації, який базується на вилученні ознак з рукописних цифр, є частиною процесу прийняття рішень у згенерованій послідовності розпізнавання. Нейронна мережа використовується для класифікації та розпізнавання рукописних чисел. Відокремлене текстове зображення з останнього кроку подається на вхід нейронної мережі. Загальна кількість нейронів у вихідному шарі мережі становить 10, оскільки запропонований програмний засіб призначений для виявлення та розпізнавання текстових документів на основі представлення арабських цифр.

Фаза повторної обробки (reprocessing) є завершальним етапом запропонованої послідовності розпізнавання. Тут виявлені та розпізнані числа відображаються у структурованому тексті з використанням індексів розпізнаних чисел тестового шаблону та визначенням відповідного еквівалентного значення в таблиці кодів ASCII.

2.3 Нейронні мережі для розпізнавання символів

Запропоновано використовувати згорткову нейронну мережу для розпізнавання рукописних цифр у вибраному текстовому зображенні.

Згорткові нейронні мережі (ЗНМ) – це клас штучних нейронних мереж з глибоким прямим зв'язком, що використовуються для аналізу зображень. Ці мережі створюються за аналогією з біологічними процесами, є формами багатошарових сенсорів і потребують мінімального обслуговування. Згорткова мережа — це тип багатошарової нейронної мережі, яка називається «згорткова мережа» за назвою операції згортки. Нервова система складається з вхідного та вихідного шарів, а також кількох прихованих шарів.

Ідея згорткових нейронних мереж полягає в тому, щоб на виході були чергування згорткових шарів (C-шари), підшарів (S-шари) та повністю зв'язаних шарів (F-шари) [25].

Загальна структура ZNM складається з багатьох шарів. Вхідне зображення подається на перший шар, а потім сигнал проходить через послідовні згорткові шари, де він поступово розділяється та перетворюється. Карту ознак можна створити за допомогою змінних шарів. З кожним наступним рівнем розмір цієї карти зменшується, а кількість каналів збільшується. Після проходження через певний шар, карта об'єктів перетворюється на скалярну або векторну карту. ZNM створює сотні таких карт об'єктів. Після проходження через такі шари згортки, вхідні дані подаються на останні кілька шарів перцептрона (повністю зв'язаної нейронної мережі), до якої надходить щойно згенерована карта ознак.

Ключовими компонентами цього типу штучної нейронної мережі є згорткові шари, які застосовують операцію згортки до вхідного зображення та передають результат на наступний шар мережі для обробки. Обговорення моделює реакцію окремого нейрона на візуальну стимуляцію.

На рисунку 2.3 показана типова архітектура згорткової нейронної мережі.

Кожен шар стеку має такі основні характеристики:

— F представляє розмір рецептивного поля, який відповідає кількості нейронів у попередньому шарі, які потім будуть підключені до нейрона в цьому шарі мережі;

— K представляє глибину, тобто кількість нейронів у цьому шарі (характеризує глибину матриці на виході мережі);

— S представляє крок сітки, який контролює ступінь перекриття між полями прийому;

— P розмір нульового доповнення, який контролює розмір вихідної матриці мережі.

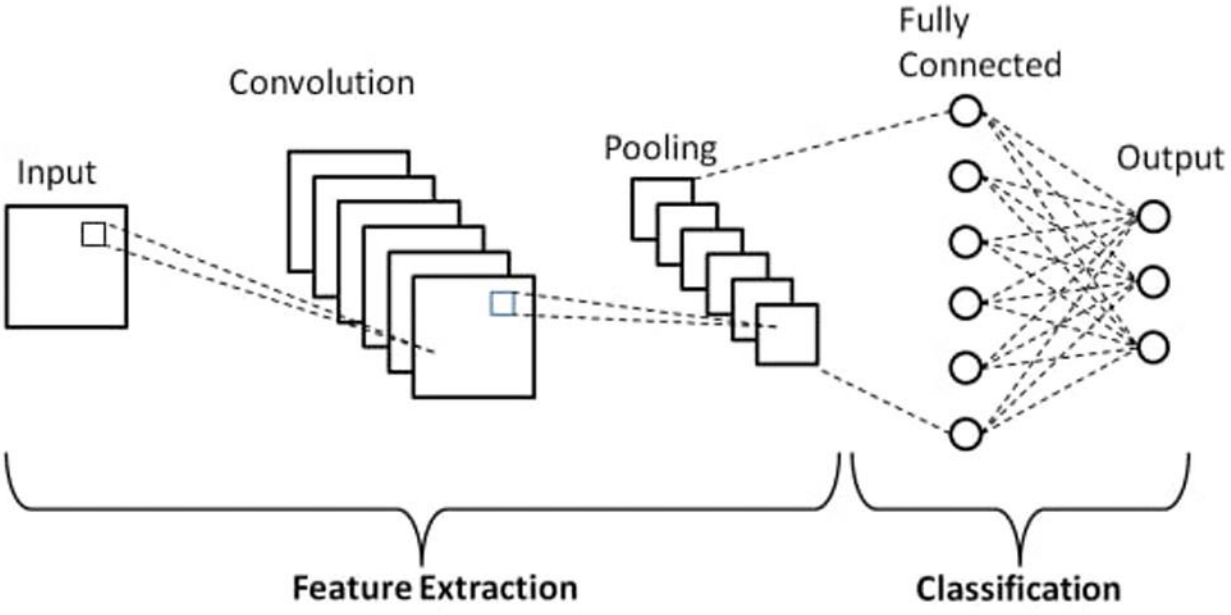

Рисунок 2.3 — Структура згорткової нейронної мережі

За потреби, вхідні дані на краю зображення заповнюються нулями. У мережі кожен нейрон згорткового шару представлений фільтром (ядром).

Цей фільтр являє собою матрицю розміром F1 × F2 × F3, параметри якої можна змінювати під час навчання мережі. За допомогою цього фільтра формується двовимірна карта активації мережі, яка є частиною вихідної матриці згорткового шару.

У у згорткових шарах використовуються спільні ваги, що означає, що для кожного рецептивного поля мережевого шару використовується один і той самий фільтр у вигляді банку ваг, що зменшує кількість необхідних шарів та покращує продуктивність мережі.

У згортковому шарі нейронної мережі виконується операція згортки над частинами зображення, що подаються на вхід цього шару, шляхом обчислення зваженої суми аналізованих частин зображення навколо заданого та визначеного згорткового ядра:

$$(f*g)[m,n] = \sum_{k,l}(f[m-k, n-1] * g[k.l]), \qquad (2.1)$$

де f — початкова матриця аналізованого зображення;

g — власне ядро згортки зображення;

k — порядковий номер рядка ядра згортки;

l — порядковий номер стовпця ядра згортки.

На рис. 2.4 показано схематичне зображення принципу роботи складного шару.

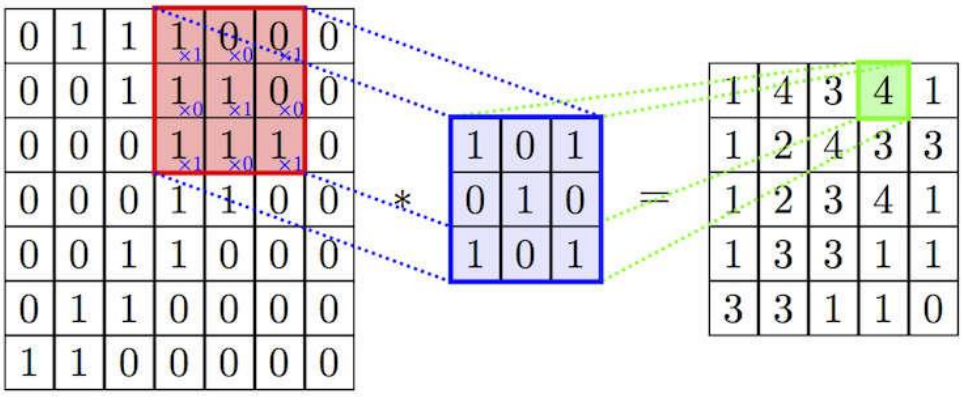

Рисунок 2.4 — Принцип роботи складного шару

Зокрема, вибрано такі гіперпараметри: глибина, заповнення та рівні дна.

Глибина — скільки ядер та коефіцієнтів обміну знаходиться в шарі; висота та ширина кожного ядра; крок (зсув) – наскільки зміщується ядро на кожному кроці під час обчислення наступного пікселя результуючого зображення. Зазвичай це 1, і чим вище значення, тим менший розмір оригінального зображення;

Заповнення: Згортка з ядром більшим за 1x1 зменшує розмір вихідного зображення. Оскільки бажано зберегти розмір оригінального зображення, малюнок доповнюється нулями по краях.

У згорткових нейронних мережах рівні реплікації зменшують розмір вхідної карти ознак (зазвичай у 2 рази) [20]. Це можна зробити кількома способами, в даному випадку ми розглядаємо метод вибору найвищого елемента (найбільше кліків) – вся карта поділяється на комірки 2x2, де вибирається найбільше значення. Тому зображення під час копіювання значно зменшується в розмірі, але вміст залишається незмінним або змінюється лише незначно.

Основа функціонування таких ЗНМ базується на безперервній зміні згорткових шарів та нижчих шарів (також званих басейновими шарами). Структура такої нервової системи є односторонньою, без зворотного зв'язку та базується на принципі використання кількох шарів. При навчанні такої нейронної мережі використовується метод множення помилок у шарах мережі.

Базова модель ZNM складається з трьох типів шарів: згорткового шару, нижнього шару та «послідовної» нейронної мережі у формі перцептивної мережі. Сенсор, будучи штучним нейроном, по суті є повноцінною нейронною мережею, в якій кожен нейрон системи пов'язаний з усіма нейронами попереднього шару, і ці зв'язки мають свої вагові коефіцієнти. У

такій нейронній мережі операція згортки виконується з матрицею малої вагової коефіцієнта, яка проходить через усі вхідні зображення, представлені цим нейронам, і формує сигнал активації, який активується після кожного зсуву для нейрона наступного шару з подібною позицією. Ця вагова матриця використовується як специфічне ядро згортки для різних нейронів у вхідному шарі. Його представлення — це графічне кодування певної характеристики зображення. Наступний шар, сформований шляхом розв'язання цієї вагової матриці, вказує на наявність певного об'єкта, ідентифікованого в оброблюваному шарі, та відображає його координати, таким чином формуючи карту ознак.

У ЗНМ набір ваг не один, а їх ціла гама, вони кодують елементи аналізованого зображення (наприклад лінії під різними кутами і дуги). Ці згорткові ядра є унікальними для цієї мережі, оскільки вони формуються за допомогою методу зворотного навчання, а не вставляються вручну оператором-людиною.

При пропусканні через нейронну мережу формується набір ознак з набором вагових коефіцієнтів, які роблять цю нейронну мережу багатоканальною. Згортковий шар застосовує операцію згортки до вхідних даних і передає результат наступному шару. Основою дросельної сітки є певний ваговий коефіцієнт. Результатом цього процесу є зображення у вигляді карти об'єктів.

Залежно від матриці, обраної для операції малювання, карта ознак відображає деякі ознаки обробленого вхідного зображення. Для вилучення найкращих ознак вхідного зображення використовується кілька різних згорткових ядер, а на виході згорткового шару мережі формується кілька карт з різними ознаками.

На малюнку зображено реакцію нейрона на зоровий подразник. Кожен згортковий нейрон обробляє дані лише для свого рецептивного поля. Параметри рівня складаються з навчальних фільтрів (або ядер), які мають невелику область прийняття, але охоплюють всю глибину вхідної структури.

Під час побудови кожен фільтр масштабується до ширини та висоти вхідного об'єму, і обчислюється скалярний добуток фільтра та вхідних даних, створюючи двовимірну карту збудження фільтра. Це дозволяє дізнатися, які фільтри були активовані, коли певний об'єкт був виявлений у певному просторі мережевого запису.

Залежно від матриці, обраної для операції малювання, карта ознак відображає деякі ознаки обробленого вхідного зображення. Для вилучення найкращих ознак вхідного зображення використовується кілька різних згорткових ядер, а на виході згорткового шару мережі формується кілька карт з різними ознаками.

Ступінь стиснення в мережі – це нелінійне стиснення (нелінійне перетворення) результуючої карти ознак, де певна група пікселів (зазвичай 2 × 2 пікселі) стискається в один піксель. Схематичне зображення процесу підвибірки показано на рисунку 2.5.

Зазвичай для виконання цього перетворення використовується функція пошуку верхнього рівня. Для заміни вибираються прямокутники або квадрати, що не перекриваються, і кожен прямокутник замінюється пікселем, вибраним як піксель з найбільшим значенням у сегменті аналізованого зображення.

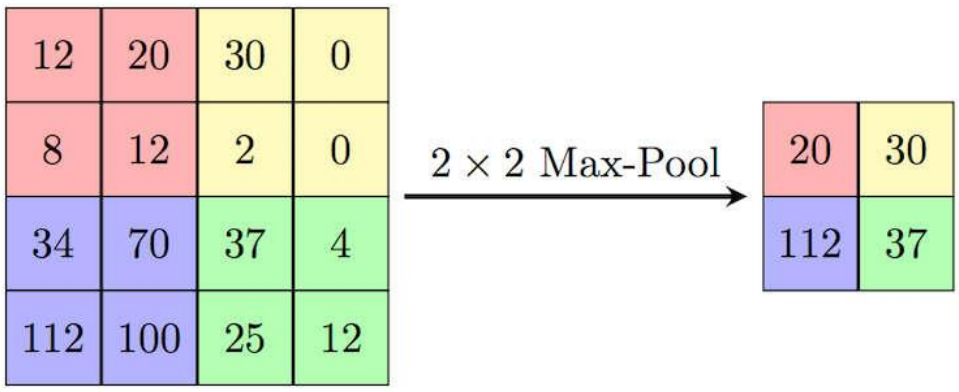

Рисунок 2.5 — Процес підвибірки

Окрім стиснення з функцією максимуму, використовуються й інші функції — середнє значення, L2-регуляризація, але, як показує практика, найкращою є комбінація з функцією максимуму.

Згортковий шар і низхідний шар міняються місцями, і низхідний шар розміщується після попереднього згорткового шару та перед наступним згортковим шаром. Після кількох проходів шумозаглушення та стиснення зображення сітка перетворюється з конкретної сітки пікселів високої роздільної здатності на більш абстрактні карти ознак, причому кожен наступний шар збільшує кількість каналів і зменшує розмір зображення.

В результаті ми отримуємо велику кількість вихідних каналів, що містять невелику кількість інформації або навіть один параметр. Унікальні числа, що розуміються як найабстрактніші поняття, спочатку виявляються на аналізованому зображенні.

Після серії ітерацій вхідного та вихідного шарів використовуються один або декілька повністю зв'язаних нейронних шарів. Нейрони в повністю зв'язаному шарі з'єднані з усіма нейронами попереднього шару, як і в традиційних нейронних мережах прямого зв'язку. Ця інформація потім об'єднується та подається в типову повністю зв'язану нейронну мережу, яка також може складатися з кількох шарів. Водночас, повністю зв'язані шари втрачають просторову структуру пікселя та мають менший розмір порівняно з кількістю пікселів у вихідному зображенні.

Повністю зв'язаний шар з'єднує кожен нейрон одного шару з кожним нейроном наступного шару, забезпечуючи мітку високого рівня. Такий шар є типом шару в типовій багатошаровій нейронній мережі. Повністю зв'язані шари з'єднують кожен нейрон одного шару з кожним нейроном наступного шару. Згортковий шар ZNM має на меті об'єднати та з'єднати вихід груп нейронів одного шару з нейронами наступного шару, локально або глобально.

Усічений лінійний одиничний шар (ReLU) використовує ненасичену передавальну функцію ReLU.

Основна відмінність між LNN та багатошаровою нейронною мережею полягає в наявності просторової організації (тривимірної здатності нейронів) та локальних зв'язків між нейронами сусідніх шарів у мережі для забезпечення певної області. Розташування поля дозволяє визначити глобальні параметри (ваги) незалежно від їхнього положення в полі та пов'язаному з ним шарі. Поєднання цих властивостей дозволяє CNM більш загально вирішувати проблеми зору. Розподіл ваги значно зменшить кількість вільних параметрів, які вивчає мережа, зменшить обсяг пам'яті, необхідний для роботи мережі, та дозволить розробляти більші та потужніші мережі. Множення типових помилок дозволяє нам навчати згорткову нейронну мережу, вибираючи ваги з фільтра. Навчання штучних нейронних мереж (ШНМ) виконується з використанням багатогранного підходу.

Розроблена структура програмного забезпечення складається з кількох модулів, які послідовно вибирають сторінки з отриманого цифрового зображення текстового документа, потім витягують фрагменти тексту та ідентифікують рукописні символи. Процес розпізнавання рукописних символів спирається на згорткову нейронну мережу. Спочатку ми розробляємо згорткову нейронну мережу для розпізнавання рукописних символів [6]. Розроблене програмне забезпечення також включає модуль для початкового навчання цієї згорткової нейронної мережі для виконання процесу розпізнавання рукописних цифр та модуль для виведення результатів розпізнавання рукописних цифр.

Розпізнавання рукописних цифр у запропонованому підході здійснюється із використанням згорткової нейронної мережі, яка згортає отримане зображення розмірності $m_0 \times n_0 \times 3$ до потрібної$m \times n \times c$, де c — кількість каналів, Це робиться за допомогою двох основних операцій – пулінгу (2-вимірний Max Pooling) та операцій згортки (2-вимірна згортка). Додаткові канали для обробки даних формуються шляхом застосування двовимірних згорткових фільтрів до вхідної матриці. Розмірність зображення поступово зменшується за допомогою операції об'єднання, яка вибирає

найбільше значення у вибраній області, зазвичай 2x2, 3x3 або 4x4, і записує результат у нову матрицю. Операції об'єднання вирішують дві основні проблеми: вони зменшують кількість ознак, відкидаючи ті, що несуть мало інформації, тим самим зменшуючи кількість математичних операцій, і вони збільшують щільність згенерованої матриці ознак, відкидаючи багато нульових значень, запобігаючи проблемі зникнення градієнта.

Кожен вихідний канал ZNM надає інформацію про певну область, дозволяючи нам перейти від матриці вихідного зображення, що описує значення кольорів, до результуючої матриці ознак. Ці ознаки приховані та несуть важливу інформацію для вибраної моделі згорткової нейронної мережі.

Для виконання процесу розпізнавання вхідне зображення розділяється на розміри $h_x \times w_x \times c$. Вибрані області проходять певний цикл перетворення для отримання матриці характерних ознак у розмірах $h \times w \times c$. Зрештою, всі перетворені області об'єднуються в матрицю розміром $n \times h \times w \times c$, де n – кількість областей вихідного зображення, які були вилучені. Згенерована матриця обробляється повнозв'язним щільним шаром, виходом якого є матриця розміром $n \times d$, де d представляє кількість нейронів. Матриця $n \times d$ видає наступні два вихідні значення — матрицю $n \times q$ з нормалізованою експоненціальною активацією softmax, де q представляє кількість можливих класів, та матрицю $n \times 4$, яка є матрицею квадратурних координат, що описують вибрані об'єкти. Розроблена нейронна мережа використовується для розпізнавання вибраних рукописних цифр.

У цьому розділі бакалаврської роботи розглянуто послідовність обробки текстового документа з рукописними цифрами, вимоги до згорткової нейронної мережі для виконання процесу розпізнавання рукописних цифр та послідовність розпізнавання рукописних цифр у текстових документах. Запропоновано розділити процес розпізнавання рукописних цифр на кілька кроків, де послідовно виконується попередня обробка отриманого документа, поділ вибраного масиву даних на рядки,

абзаци, числа та цифри, а також використання згорткової нейронної мережі для розпізнавання рукописних цифр.

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР

Процес виділення і розпізнавання рукописних цифр сканованих текстових документів виконується за послідовний ряд етапів, реалізація яких у вигляді програми розглянуті у даному розділі бакалаврської роботи.

3.1 Вибір засобів розробки програми розпізнавання

При виборі мови програмування і середовища розробки були враховані такі ключові вимоги: необхідність сумісності програми із більшістю операційних систем та доступність і безкоштовність (або низька вартість) середовища розробки із огляду на технічну та економічну доцільність роботи.

Цим критеріям відповідає мова програмування Python та інтегроване середовище розробки Python IDE. Середовище Python IDE є тим універсальним інструментом, який формує умови запуску програм створених на базі Python на різних пристроях із різними операційними системами.

Python — це універсальна мова програмування високого рівня, спрямована на підвищення продуктивності розробників та читабельності коду. Вона має лаконічний синтаксис ядра, а також пропонує велику стандартну бібліотеку з широким набором функцій.

Python підтримує кілька стилів програмування, включаючи структуроване, об'єктно-орієнтоване, функціональне, імперативне та аспектне програмування. Мова має динамічну типізацію, автоматичне керування пам'яттю, вбудовані винятки, багатопотоковість та зручні структури даних високого рівня. Код Python поділено на модулі, а модулі можна розділити на функції та класи, які далі можна розділити на пакети.

Вибір Python для цього програмного продукту зумовлений його простотою використання та оптимальною мовою для виконання складних математичних операцій. Python — це універсальна мова програмування

високого рівня, спрямована на підвищення продуктивності розробників, її легко писати та читати. Синтаксис ядра Python мінімалістичний. Водночас стандартна бібліотека містить багато корисних функцій. Python підтримує структуроване, об'єктно-орієнтоване, імперативне, аспектне та функціональне програмування. Основними особливостями архітектури є повнотекстовий перегляд, динамічний доступ, обробка винятків, автоматичне керування пам'яттю, високорівневі структури даних та підтримка багатопотоковості. Мова підтримує поділ програми на окремі модулі, які, в свою чергу, можна об'єднувати в пакети.

Інтерпретатор CPython — це найкраща реалізація Python, яка підтримує найактивніші платформи. Він вільно розповсюджується за ліцензією Python Software Foundation, що дозволяє використовувати його без обмежень у будь-якій програмі, включаючи користувацькі програми. Python працює майже на всіх відомих платформах, від портативних та мейнфрейм-платформ до КПК. Існують версії для Microsoft Windows, UNIX-подібних систем (включаючи FreeBSD та Linux), Windows Mobile, Symbian та Android.

Однак, на відміну від інших портованих систем, Python підтримує платформо-специфічні технології (такі як Microsoft COM/DCOM). Існує також Jython, спеціальна версія Python для віртуальної машини Java, яка дозволяє інтерпретатору працювати на будь-якій системі, що підтримує Java. Однак класи Java можна використовувати безпосередньо з Python і навіть писати на Python. Код компілюється та виконується на віртуальній машині Java (JVM), що дозволяє використовувати код Python з програм Java. Розроблений прототип системи розпізнавання тексту з розбитими символами має продемонструвати роботу алгоритму розпізнавання вмісту розбитих текстів. Програмне забезпечення реалізовано мовами програмування вищого рівня Java та Python.

Основні переваги мови програмування Python полягають у наступному. Добре розроблена мова програмування, яка включає сучасні тенденції програмування з нуля. Крім того, мова програмування Python розвивається

швидкими темпами, процес додавання нових конструкцій до мови йде повним ходом, і вона продовжує впроваджувати такі методи, як аспектне програмування, функціональне програмування тощо, залишаючись при цьому внутрішньо сумісною та узгодженою. Крім того, мова Python проста в програмуванні.

Python використовує простий та зрозумілий синтаксис (порівняно з C++, PHP) і дозволяє легко читати чужий код і вільно розуміти власний давно написаний код. Існує багато бібліотек з кодом для будь-якого випадку (наприклад, робота із зображеннями, електронними таблицями Excel або Twitter).

Python дуже портативний і може працювати на всіх поширених операційних системах і на багатьох архітектурах: Windows, MacOS, Linux та Arduino на мінікомп'ютерах. Система залежностей дуже добре продумана, а перенесення програм на іншу систему є простим і зрозумілим.

Python має сильні та однорідні абстракції для типів даних і значень. Крім того, мова має гнучкі симетричні структури абстракції для побудови типів і класів.

Крім того, система використовує бібліотеку OpenCV, яка включає алгоритми машинного навчання для попередньої обробки зображень, взятих з текстових документів. OpenCV — це бібліотека алгоритмів комп'ютерного зору з відкритим кодом, яка дозволяє легко працювати з обробкою зображень та загальноприйнятими числовими алгоритмами. Бібліотека Numpy розширює можливості Python, підтримуючи великі багатовимірні масиви та матриці, а також математичні функції високого рівня. Все це працює дуже швидко завдяки використанню реалізацій мов C та C++.

3.2 Розробка структури програми розпізнавання рукописних цифр

Структура програми розпізнавання рукописних цифр складається із трьох базових модулів й одного допоміжного модуля, якими є модуль попередньої обробки текстового зображень, модуль сегментації текстового

документа та модуля розпізнавання рукописних цифр на основі нейронної мережі. У складі програми є ще один допоміжний модуль для початкового налаштування нейронної мережі для здійснення процесу розпізнавання рукописних цифр (рис. В.1).

Модуль попередньої обробки текстового зображення виконує такі дії. Отримуємо зображення рукописного тексту, здійснюємо фільтрацію від можливих завад та морфологічних дефектів, переводимо зображення тексту з отриманого кольору у бітовий формат та почергово виділяємо області, що відповідають окремим сторінкам текстового документу. Надалі розділяємо виділену сторінку тексту на окремі рядки із використанням гістограм вертикального розподілу яскравості зображення текстового документу. Наступним кроком є розподіл рядків на окремі слова чи групи цифр із використанням гістограми розподілу яскравості виділеного рядка тексту по горизонталі. Надалі розподіляємо виділені групи цифр на окремі рукописні символи та визначаємо їх положення. Після розподілу груп цифр на окремі символи із цифр виконуємо їх нормалізацію до необхідного розміру для подальшого виконання процесу розпізнавання рукописних цифр на основі згорткової нейронної мережі. Процес розпізнавання рукописних цифр відбувається із використанням нейронної мережі глибокого навчання.

Для налаштування модуля підготовки нейронної мережі для подальшої роботи використовуємо базу даних еталонних символів MNIST, до складу якої входить набір із 10 базових символів цифр із значним розмаїттям різного їх написання, щоб настроїти згорткову нейронну мережу таким чином, щоб вона здійснювала процес розпізнавання рукописних цифр [23]. Приклад таких цифр приведено на рис. 3.1.

Навчальний набір MNIST складається із 20000 моделей із NIST Special Database3 та 20000 моделей із NIST Special Database1. Тестовий набір для перевірки якості навчання нейронної мережі складається із 4000 зображень із SD-3 та 4000 зразків із SD-1. Сформований набір навчальних і

тренувальних даних у розмірі 40000 екземплярів цифр містить приклади різних видів написання рукописних цифр.

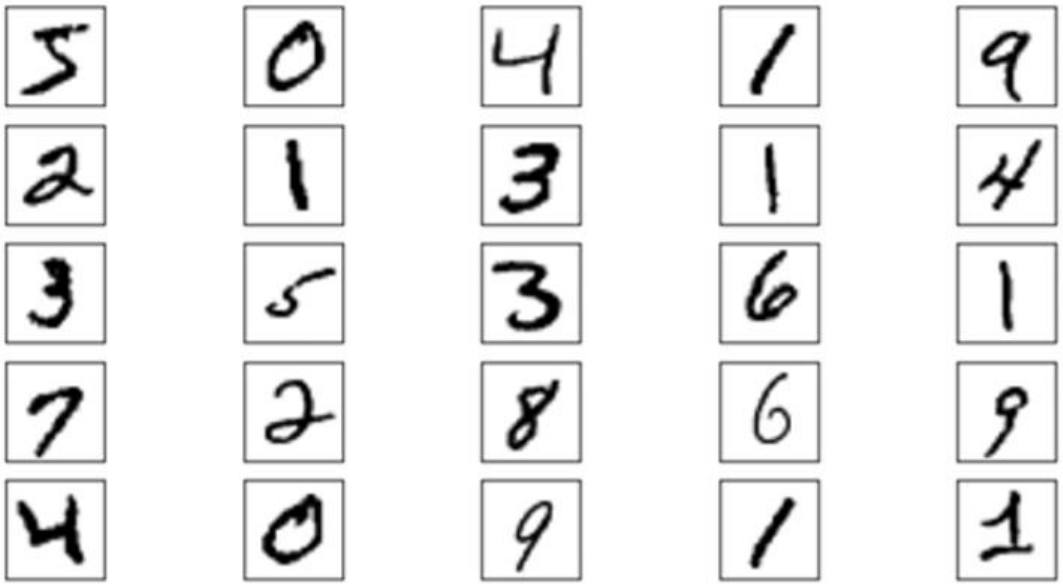

Рисунок 3.1 — Приклад набору рукописних цифр

Сформована послідовність розпізнавання рукописних цифр наведена у Додатку В на рис. В.2 до бакалаврської кваліфікаційної роботи.

3.3 Розробка програмного засобу розпізнавання рукописних цифр

Програма для розпізнавання рукописних чисел виконує низку дій, в результаті яких можна вибрати та розпізнати певний символ. Робота програми починається з отримання сканованого рукописного текстового документа. Наступними кроками є етапи поділу тексту на рядки, числа та відображення в них окремих цифр. Поділ отриманого тексту на окремі цифри є одним з важливих кроків у роботі програми для виконання операції розпізнавання рукописних символів. Для цього ми скористаємося можливостями відкритої бібліотеки OpenCV та виберемо одну з її функцій для відображення контурів cv2.findContours. Ця функція дозволить нам виконувати операції над відрізками, числами та відображення окремих цифр.

Далі, інструменти бібліотеки OpenCV були використані для відображення областей зображення, що містять розпізнані числа або літери.

Для відображення контурів чисел була використана функція cv2.findContours. Алгоритм попередньої обробки зображення та відображення області, що містить цифри числа, включає такі етапи роботи.

Фільтрація зображення для зниження рівня завад (використовуємо фільтр Гауса, це функція cv2.GaussianBlur).Наступним кроком є бінаризація зображення (використовуємо функцію cv2.threshold, її параметри було налаштовано для підготовки отриманого зображення для подальшого надійного виділення контурів символів). Надалі виконується морфологічна трансформація або дилатація із використанням функції cv2.dilate, яка використовується іздекількома ітераціями. Виділення контурів символів та їх сортування (виділення контурів символів здійснюється за допомогою функції cv2.findContours). Наступним є етап сегментації зображення, тобто поступове виділення областей розпізнавання текстового документу як набору прямокутників, що містять контури рукописних цифр (тут у тому числі використовувалась функція cv2.boundingRect).

Для прямого розпізнавання вибрані області інтересу були вирізані з оригінального зображення, нанесені на них у двійковому форматі, після чого зображення окремих символів (без розширення чи інших спотворень) були збільшені до розміру 28x28. Кожне значення пікселя зображень було в діапазоні від 0 до 255, тому нормалізація цих значень пікселів виконувалася шляхом ділення на 255, щоб усі значення в масиві, що описує зображення для розпізнавання, могли бути в діапазоні від 0 до 1.

Вихідне зображення має бути підготовлене до початку сегментації. У цій роботі реалізовано такі функції:

—Cv2.Imread(): метод завантажує зображення з заданого шляху та перетворює його в масив пікселів;

—Gray(): перетворює зображення у градації сірого, використовуючи лише одне значення яскравості замість трьох кольорових каналів;

—Contrast(): покращує контрастність зображення, нормалізуючи всі значення пікселів у діапазоні від 0 до 255;

—As_white_background(): Переконайтеся, що фон зображення білий. Якщо середнє значення пікселів менше 128, текст буде темним, а фон світлим.

Морфологічний аналіз застосовується за допомогою фільтра Канні в методі get_edges, який визначає контури на зображенні. Це дуже важливий крок у визначенні рядків тексту на зображенні, що дозволяє нам точніше розділити текст на різні сегменти для подальшого аналізу.

Одним з основних аспектів нашого підходу є використання геометричних та статистичних ознак у поєднанні з процедурою проекції. Як і в методах get_lines_positions та get_words_positions, ми аналізуємо гістограми зображення, щоб визначити горизонтальні та вертикальні лінії. Використання цих методів дозволяє нам враховувати розташування та розташування літер і цифр на зображенні, і таким чином виявляти геометричні характеристики тексту.

Метод get_letters_positions використовує геометричний аналіз для визначення розташування окремих чисел у межах вибраного числа. Він використовує функції бібліотеки OpenCV, такі як cv2.dilate та cv2.findContours, для виявлення та позначення унікальних чисел, які вказують на нашу залежність від геометричних аспектів тексту.

Щоб витягти текст зі сканованого зображення у форматі PNG, ми відкриваємо його за допомогою стандартних процедур, доступних у бібліотеці OpenCV (cv2.imread) [28]. Наступний крок — конвертувати його у градації сірого та виконати процес бінаризації зображення, використовуючи поточну кольорову модель RGB cv2.cvtColor, потім змінити його розмір за допомогою процедури (cv2.erode) та згенерувати рукописні цифрові контури за допомогою процедури cv2.threshold (лістинг 3.1).

Лістинг 3.1 — Формування контуру рукописних цифр

```
image_file = " text.png "
img = cv2.Imread(image_file)
```

```
gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY)
img_erode = cv2.erode(thresh, np.ones((3, 3), np.uint8), iterations=1)
```

На отриманій сторінці виконуємо серію операцій виділення, спочатку на рядках, потім на числах, а потім переходимо до виділених символів у вигляді рукописних чисел. Виконуємо процедуру формування ієрархії виділених контурів символів у порядку складності.

В результаті виконаних операцій отримуємо ієрархічне дерево контурів за допомогою параметра cv2.RETR_TREE. Спочатку формується контур загального зображення отриманого зображення текстової сторінки, потім формуються контури слів і завершуємо контурами окремих символів. Таким чином, виконуємо поділ текстової сторінки на окремі рівні символів. У нашому випадку процес розпізнавання має зберігатися на контурах окремих символів, тому ми зупинимося на цьому рівні текстового документа.

Наступним кроком обробки текстового документа буде збереження кожного з виділених рукописних чисел у масштабі 28x28 біт, оскільки база даних довідкових чисел MNIST [25] працює в такому масштабі, як і нейронна мережа для розпізнавання рукописних чисел. Крім того, ми розташовуємо координати вибраних рукописних символів вздовж осі X, щоб вибрані координати можна було використовувати пізніше для формування пробілів між цими символами.

Для цього слід змінити розмір отриманого фрагменту зображення символу до квадратного розміру (лістинг 3.2).

Лістинг 3.2 — Зміна розміру зображення символів

```
size_ max= max (w,h)
letter_ square= 255 *np.ones(shape=[size_ max, size_ max], dtype= np.uint8)
if w>h:
```

Ми форматуємо розмір зображення символу, якщо дивитися зверху вниз. Ми також форматуємо розмір зображення символу, якщо дивитися зліва направо у вибраному рядку. Ми завершуємо сортування масиву символів за координатою X. Після того, як ми завершимо сортування символів за координатою X, ми зберігаємо їх як функцію кортежу (x, w, літера), щоб після розпізнавання ми могли створювати пробіли для кожного символу та перевіряти правильність роботи.

Ми використовуємо нейронну мережу для розпізнавання вибраних символів. Ми також використовуватимемо базу даних символів MNIST, яка являє собою колекцію різних символів.

Тренування нейронної мережі FasterR-CNN для розпізнавання рукописних символів відбувається у два етапи. Першим виконується тренування базової моделі класифікатора, яким даному випадку був вибраний VGG-16. З числа проаналізованих варіантів кращу точність та надійність розпізнавання символів тексту забезпечила модель мережі із архітектурою типу VGG-16. Структура цієї нейронної мережі складається з послідовності згорткових та субзгорткових шарів, вихід яких подається як вхідне значення повністю зв'язного шару. У випадку використання цієї моделі як основи для FasterR-CNN, повністю зв'язані шари не використовуються, а вихідним значенням неповного VGG-16 є матриця ознак зображення, що використовується мережею ознак регіону (Region Proposal Network RPN) для розпізнавання об'єктів.

Для роботи нейронної мережі нам спочатку потрібно імпортувати всі модулі, які будуть потрібні для початкового навчання нашої моделі. Ми також будемо використовувати бібліотеку Keras, яка вже містить певну кількість готових модулів. Серед цих модулів є база даних MNIST. Таким чином, ми імпортуємо необхідні для роботи модулі та використовуємо їх у нашій роботі.

Перш ніж розпочати роботу, нам потрібно використовувати навчальний модуль нейронної мережі, створений перед початком роботи. Він працює

наступним чином. Вхідний шар мережі містить 784 нейрони, що відповідають зображенню розміром 28 на 28 пікселів, де кожне значення відповідає яскравості. На вході нейронна мережа оперує вибраною частиною зображення символу розміром 28x28 біт, а на виході є 10 відповідних виходів. Якщо символ розпізнано, в одному з виходів формується логічне значення, що вказує на факт знаходження та розпізнавання відповідного символу. Також створюємо модель згорткової нейронної мережі (лістинг 3.3).

Лістинг 3.3 — Опис моделі згорткової мережі

```
model=Sequential()
model.add(Convolution2D (filters=32, kernel_ size=(3, 3), padding='valid', input_ shape= (28, 28,1), activation='relu'))
model.add (Convolution2D (filters=10, kernel_ size=(3, 3), activation= 'relu'))
model.add (MaxPooling2D (pool_ size=(2, 2)))
model.add (Flatten())
model.add (Dropout(0.25))
model.add (Dense (512, activation='relu'))
model.add (Dropout (0.5))
```

Наступний етап — формування шарів моделі згорткової нейронної мережі. Така мережа складається з навчальних шарів та шарів передтестування. Така мережа найкраще працює з даними, представленими у вигляді матриці. Тому цей тип мережі використовується для обробки зображень. Інший шар використовується для виключення окремих нейронів під час навчання, щоб зменшити ймовірність перенавчання.

Тепер переходимо до етапу навчання мережі. Тут ми будемо використовувати базу даних MNIST. Для зчитування даних з цієї бази даних та її роботи ми скористаємося бібліотекою idx2numpy та підготуємо дані, необхідні для навчання та подальшого тестування нейронною мережею.

Завантажуємо створену модель для виконання операції розпізнавання та викликаємо функцію predict_class.

В результаті отримуємо матрицю, що описує розпізнані рукописні цифри. Перелік програм для розпізнавання рукописних цифр наведено в Додатку Д бакалаврської роботи.

3.4 Склад комп’ютерної системи розпізнавання рукописних цифр

При розпізнаванні образів у вигляді зображення останні подаються на матрицю рецепторів по аналогії з тим, як зображення потрапляє на сітківку ока. Отримане зображення далі надходить в комп'ютер для подальшої переробки і розпізнавання, тобто зарахування образу до того чи іншого класу символів. За аналогією будується модель з полем рецепторів, які представляють собою прямокутний масив, на якому можна зображувати різноманітні конфігурації текстових символів. Введення інформації в комп'ютер являє собою процес сканування зображення символів за допомогою периферійного пристрою - сканера, і збереження зображення у форматі графічного файлу. Іншим способом може служити створення зображення в будь-який графічній програмі. Збережене зображення являє собою графічний файл, тобто послідовність кодових знаків, які несуть в собі інформацію про структуру візуального зображення. Однак для використання цієї інформації для подальшого розпізнавання необхідно перетворити цей зашифрований код в більш доступне матричне подання даних. Отже мінімальний набір апаратних засобів для розпізнавання текстових документів повинен складатися із комп'ютера і сканера. Комп'ютер повинен мати у своєму складу такий мінімальний набір вузлів: материнську плату, жорсткий диск, оперативну пам'ять, монітор.

На сканер покладаються завдання переведення зображення текстового документу у послідовність цифрових символів. Основна його характеристика називається оптичною роздільною здатністю. Вона визначається кількістю світлочутливих елементів (фотодатчиків), які припадають на дюйм

горизонталі, яка сканується. Зазвичай її визначають за кількістю точок на дюйм - dpi (dots per inch). Нормальний рівень роздільної здатності не менше 600 dpi, збільшення його означає застосування більш дорогої оптики, дорогих світлочутливих елементів, збільшення часу сканування. Для обробки слайдів необхідна більш висока роздільна здатність 1200 dpi.

Існуючі на ринку програми розпізнавання текстів застосовують певний лексичний набір слів для звірки тексту з метою усунення можливих помилок, що дозволяє штучно підняти відсоток розпізнавання слів. Крім того, наявне програмне забезпечення для розпізнавання образів у вигляді набору слів і цифр, в основному, працюють під певні шрифти, що є певним недоліком і звужує коло завдань розпізнавання. Сучасні програми розпізнавання і дослідження текстових документів також використовують матриці з високою роздільною здатністю. Чим вище роздільна здатність сканера, тим точніше можна виявити закономірності і правильно налаштувати модель для розпізнавання тексту. Для отримання більш точних характеристик повинна бути введена досить велика база даних еталонних елементів, що складається не менше ніж 1000 різних видів одного елемента. Набір даних для еталонної бази для подальшого скрупульозного дослідження графічної структури зображення може тривати дуже значний час і вимагатиме значних ресурсів.

Для функціонування пропонується така комп’ютерна система. Комп’ютер для перевірки якості роботи створеної програми характеризується такими параметрами: процесор типу AMD Ryzen5 3600x із тактовою частотою 3,60 ГГц, жорсткий диск 2 ТВ, оперативна пам’ять комп’ютера складає 32GB DIMM DDR4, використовувалась відеокарта Nvidia GTX 1650 із об’ємом відеопам’яті 4GB. Для виведення результатів роботи по аналізу зображень використовувався пристрій Acer Nitro VG3 21.8.

Сканери характеризується рядом параметрів, і на першому місці є роздільна здатність, яка вказує максимальну кількість точок на кожен лінійний дюйм (dots per inch). Як правило, для сканерів вказуються два значення, наприклад 1200×1200 dpi. Перша цифра характеризує оптичну

(горизонтальну) роздільну здатність. Вона залежить від щільності світлочутливих датчиків, що припадають на кожен дюйм зображення по горизонталі. Друга цифра характеризує механічну (вертикальну) роздільну здатність. Вона вказує точність переміщення скануючої каретки або сканованого оригіналу. Для нас важливим є ще розмір поля сканування. Більшість сканерів працюють з форматом А4. Виходячи із цих характеристик, вибираємо сканер типу Canon CanoScan LIDE 300, у якого роздільна здатність 2400x 4800 dpi і працює з форматом паперу А4.

Перераховані технічні показники складу комп'ютера і сканера є достатніми для здійснення процесу тестування розробленого програмного комплексу й дозволять отримати результати контролю для здійснення аналізу працездатності та коректності функціонування створеного програмного продукту.

3.5 Перевірка якості програми розпізнавання рукописних цифр

Найбільш популярними та затребуваними є запропоновані системи OCR, але враховуючи активний розвиток галузі розпізнавання, таких систем набагато більше. Існують різні критерії оцінки якості систем оптичного розпізнавання тексту (OCR). Весь потік операцій, що входять до процесу оптичного розпізнавання, можна розділити на два етапи, які суттєво впливають на кінцевий етап: структурний аналіз (сегментація документа, класифікація регіонів) та розпізнавання тексту. Для кожного етапу існують свої методи оцінки якості :

— оцінка документа за результатами структурного аналізу документа;

— оцінка розпізнавання результатів розпізнавання за стандартами;

— оцінка розпізнавання результатів розпізнавання без стандартів [22].

Існує кілька важливих питань, пов'язаних з розпізнаванням рукописних та друкованих символів. Найважливішими з них є наступні:

— різноманітність форм та написання символів;

— спотворення символів;

— варіації розмірів та масштабів символів.

Для оцінки якості систем та алгоритмів розпізнавання тексту використовуються такі категорії, як точність та надійність. Точність системи розпізнавання тексту відображає її здатність правильно розпізнавати вибраний об'єкт. Однак на практиці дуже важко визначити справжнє значення ймовірності розпізнавання, а точність розпізнавання визначається апостеріорною ймовірністю правильного розпізнавання текстових символів для вхідного набору текстових даних. За такого підходу задачу тестового розпізнавання можна сформулювати як задачу максимізації апостеріорної ймовірності правильного розпізнавання символів для вибраного набору текстових даних.

Ще одним показником якості системи розпізнавання є надійність розпізнавання. Оцінка надійності системи розпізнавання відображає її здатність передбачати ступінь точності її результатів.

Для перевірки якості роботи створеного програмного продукту була підготовлена навчальна вибірка із 40000 різних варіантів написання рукописних цифр. Окремо було сформована вибірка із 10000 рукописних символів, до складу цієї вибірки були включені рукописні літери латинського й кириличного алфавітів. Всього для додаткової перевірки роботи для розпізнавання цифр та відкидання літер було використано у різному представлені 480 текстових символів.

У результаті проведених експериментів по розпізнаванню першої групи символів було розпізнано 99,3% рукописних цифр. У цій групі не було розпізнано всього 0,7% сформованих варіантів рукописних цифр. Деякі символи в цьому наборі були неправильно ідентифіковані через помилку у визначенні розміру рукописної цифри на основі розміру прямокутника, в якому мав бути розміщений підпис. Цю варіацію розпізнавання можна пояснити тим, що розмір символів був нормалізований, зменшений до стандартного розміру, що призводить до того, що деякі символи схожі один

на одного. Детальніше вивчення таких неправильних символів показує, що більшість нерозпізнаваних символів належать до форм, які значно упереджені до рукописних цифр. Це можна пояснити тим фактором, що у такому випадку рукописні цифри значно відрізняються по своїй формі представлення від базового подання символів, які були задіяні для виконання навчання згорткової нейронної мережі.

Експериментальні дослідження показують, що розроблений програмний продукт для розпізнавання рукописних цифр може бути використаний у різних системах для перетворення рукописних документів в електронну форму.

3.6 Верифікація програми розпізнавання рукописних цифр

Було проведено програмне тестування запропонованого підходу до оптичного розпізнавання рукописних цифр у текстових документах. Здатність виявляти рукописні цифри розробленим програмним забезпеченням були перевірені на різних документах з різним ступенем письма, перешкод та повороту символів.

Для верифікації розробленої програми було обрано високопродуктивну комп'ютерну систему, що складається з комп'ютера та сканера.

Обрані програмні засоби для тестування програми включають мову програмування Python, модулі TensorFlow, Keras, Pillow та NumPy. Було протестовано працездатність програмного забезпечення для бакалаврської роботи, розробленого для розпізнавання рукописних чисел та форматування тексту в документах різних розширень.

Для перевірки якості розпізнавання тестування проводиться у два етапи. На першому етапі перевіряється точність відображення груп чисел у числовому форматі, а також малюються обмежувальні рамки розпізнаних рукописних чисел за допомогою модуля ImageDraw бібліотеки Pillow та використовується вбудований метод зображення Pillow.Image.show(), який відкриває зображення у стандартному переглядачі зображень Windows.

Наступним протестованим модулем був структурний модуль, який використовує математичні операції для представлення окремих чисел у документі (рис. Г.2 та рис. Г.3). Вихідні зображення відображаються відповідно до алгоритму з попереднього раунду тестування.

Результати проведеного тестування відповідають заданим цілям. Отриманий програмний продукт може бути використаний як основа для формування повноцінного програмного застосунку для обробки документів із рукописними цифрами.

Даний розділ бакалаврської роботи присвячений розробленню та опису структури програми для розпізнавання рукописних цифр текстових документів, сформована програма розпізнавання рукописних цифр та виконано процес навчання загорткової нейронної мережі на підготовленому тестовому наборі рукописних цифр. Також було проведено тестування створеного програмного продукту для розпізнаних рукописних цифру складі повноцінного текстового документу.

ВИСНОВКИ

Існуючі на теперішній час системи розпізнавання текстових документів із високою якістю та достовірністю дозволяють переводити ці документи в електронну форму. Але якщо у таких документах з’являються рукописні символи, то значно зростає відсоток помилок розпізнавання. Указана проблема на теперішній час є у значній мірі актуальною. Дана бакалаврська робота присвячена вирішенню завдання створення програмних засобів розпізнавання текстів із рукописними цифрами у системах електронного документообігу.

У першому розділі бакалаврської роботи був виконаний огляд існуючих систем, методів та підходів до оптичного розпізнавання рукописних текстових документів, проведено порівняльний аналіз відомих засобів розпізнавання та вказано на деякі недоліки й обґрунтовано необхідність розробки програмного продукту для розпізнавання рукописних цифр. Проведений аналіз основних методів та засобів розпізнавання символів рукописних документів дозволив виявити проблеми, що є у цій області, та уточнити сферу застосування різних методів з урахуванням їх переваг та недоліків у застосуванні до вирішення завдання по розпізнаванню рукописних цифр.

У другому розділі бакалаврської роботи запропонована послідовність розпізнавання рукописних цифр документів, розроблена блок-схема алгоритму виділення рукописних символів та їх розпізнавання. Запропоновано процес розпізнавання рукописних цифр розділити на два етапи: на першому етапі здійснюється попередня обробка текстового документу та виділяються рукописні цифри, на другому етапі використовується нейронна мережа для розпізнавання рукописних цифр.

У третьому розділі роботи була вибрана мова та інструментальні засоби програмування, сформована структура програмного продукту та розроблена програма по розпізнаванню рукописних цифр. Також були

приведені результати експериментальної перевірки реалізованого підходу по розпізнаванню рукописних цифр.

Розроблений програмний продукт може використовуватися в засобах розпізнавання рукописних цифр у текстових документах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Оптичне розпізнавання символів (OCR) [Електронний ресурс] – Режим доступу: https://hashdork.com/uk/optical-character-recognition/.

2. Коротун О. В. Система розпізнавання рукописних цифр з оцінкою якості. / О. В. Коротун, Г. В. Марчук, Д. К. Марчук, О. В. Талавер // Технічна інженерія. № 1 (85) 2020. с. 135 —146.

3. Жихаревич В. В. Аналіз методів розпізнавання символів тексту / В. В. Жихаревич, С. Е. Остапов, І. В. Миронів // Радіоелектронні і комп'ютерні системи. — 2016. — № 5. — С. 137 — 142.

4. Субботін С. О. Нейронні мережі : теорія та практика: навч. посіб. / С. О. Субботін. – Житомир : Вид. О. О. Євенок, 2020. – 184 с.

5. Курочкін Д. О. Розпізнавання рукописних цифр із використанням нейронної мережі. // Д. О. Курочкін, І. С. Колесник, М. А. Очкуров // Матеріали Міжнародної науково-практичної конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2025)». [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn.

6. Кушнір Н.О. Використання згорткових нейронних мереж у задачах розпізнавання та класифікації об'єктів зображень / Н.О. Кушнір, Т.М. Локтікова, А.В. Морозов, В.О. Юрченко // Технічна інженерія №1 (89), 2022. — С. 93 —100.

7. Shi B., Bai X., Yao C. An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition // IEEE transactions on pattern analysis and machine intelligence. 2017, vol. 39, no. 11, pp. 2298 — 2304.

8. Busta M., Neumann L., Matas J. Deep text spotter: an end-to-end trainable scene text localization and recognition framework // Proceedings of the IEEE International Conference on Computer Vision. 2017, pp. 2204 — 2212.

9. Ye Q., Doermann D. Text detection and recognition in imagery: A survey // IEEE transactions on pattern analysis and machine intelligence. 2015, vol. 37, №7, pp. 1480 — 1500.

10. Uchida S. Statistical Deformation Model for Handwritten Character Recognition // Advances in Digital Document Processing and Retrieval. 2014, pp. 157 — 174.

11. Doetsch P. Fast and robust training of recurrent neural networks for offline handwriting recognition / P. Doetsch, M. Kozielski, H. Ney. // 2014 14th International Conference on Frontiers in Handwriting Recognition. — 2014. – №14. – pp. 279 — 284.

12. Ковальчук А. М. Застосування згорткової нейронної мережі для розпізнавання рукописних символів / А. М. Ковальчук, Г. В. Марчук, Д. К. Марчук // Зб. наук. пр. «Вчені записки ТНУ імені В.І. Вернадського». Серія: технічні науки, Том 30 (69). Ч. 1. № 4, 2019, с. 68—73.

13. Що таке ABBYY FineReader [Електронний ресурс] – Режим доступу: https://help.abbyy.com/uk-ua/finereader/12/overview.

14. ABBYY FineReader. [Електронний ресурс]. – Режим доступу : https://pdf.abbyy.com.

15. Tesseract documentation. Tesseract OCR. [Електронний ресурс]. – Режим доступу: https://tesseract-ocr.github.io/.

16. Програма Readiris Pro [Електронний ресурс] – Режим доступу: https://readiris.ua.uptodown.com/windows

17. Cloud Vision API. [Електронний ресурс]. – Режим доступу : https://cloud.google.com/vision/.

18. Khaustov P.A. Algorithms for handwritten character recognition based on constructing structural models / P.A. Khaustov // Computer Optics. — 2017. — №41 (1). — P. 67—78.

19. Feuz S. RNN-Based Handwriting Recognition in Gboard [Електронний ресурс] / S. Feuz, P. Gonnet // Google AI Blog. — 2019. – Режим доступу до

ресурсу: https://ai.googleblog.com/2019/03/rnn-based-handwriting-recognition-in.html.

20. Матвійчук А.М. Аналіз архітектур нейронних мереж для розпізнавання рукописних цифр // Телекомунікаційні та інформаційні технології. 2023.№ 4(81) с. 118 — 127.

21. Круліковський Б. Б. Теоретичні основи розпізнавання багатомірних образів у Хеммінговому просторі / Б.Б. Круліковський, А.І. Сидор, О.М. Заставний, Я.М. Николайчук // Науковий вісник НЛТУ України. 2016. Вип. 26. — С. 361 — 367.

22. Порівняння систем оптичного розпізнавання. [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Comparison_of_optical _character_ recognition_software.

23. OpenCV library [Електронний ресурс]. – Режим доступу: https://opencv.org/.

24. AForge.NET: Framework [Електронний ресурс]. – Режим доступу: http://www.aforgenet.com/framework/.

25. LTI-Lib [Електронний ресурс]. – Режим доступу: http://ltilib.sourceforge.net/doc/homepage/index.shtml.

26. OpenCV vs VXL vs LTI: Performance Test - AI Shack [Електронний ресурс]. – Режим доступу: http://www.aishack.in/tutorials/opencv-vs-vxl-vs-lti-performance-test/.

27. LeCun Y. The mnist database / Y.LeCun [Електронний ресурс]. – Режим доступу : http://yann.lecun.com/exdb/mnist/ .

28. TensorFlow: [Електронний ресурс]. – Режим доступу :: https://www.tensorflow.org/.

29. Keras: the python deep learning API: [Електронний ресурс]. – Режим доступу :: https://keras.io.

ДОДАТОК А

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ ВНТУ

д.т.н., проф.

_____О. Д. Азаров

“__” _____ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання бакалаврської кваліфікаційної роботи

«Комп’ютерна система розпізнавання рукописних цифр на основі нейронної мережі»

08-54.БКР.011.00.000 ТЗ

Науковий керівник к.т.н., доц. каф. ОТ
_____ Колесник Л. А.

Студент групи 1КІ-23мс
_____ Курочкін Д. О.

Вінниця 2025

1 Підстава для виконання бакалаврської кваліфікаційної роботи (БКР)

1.1 Актуальність даного дослідження визначається необхідністю вирішення основних проблем існуючих засобів пошуку та розпізнавання рукописних символів текстових документів. Особливо актуальним дане дослідження є у зв’язку з тенденціями до впровадження таких засобів у різноманітних додатках для пошуку та розпізнавання рукописних цифр у системах електронного документообігу.

1.2 Наказ про затвердження теми бакалаврської кваліфікаційної роботи.

2 Мета і призначення БКР

2.1 Мета роботи полягає у підвищенні ефективності розпізнавання рукописних символів текстових документів. Підвищення ефективності розпізнавання рукописних цифр текстових документів досягається шляхом більш точної перевірки об’єктів на зображенні із набором еталонних символів.

2.2 Призначення розробки — виконання бакалаврської кваліфікаційної роботи із подальшим впровадженням та розвитком продукту.

3 Вихідні дані для виконання БКР:

— запропонувати нові підходи для реалізації методу розпізнавання рукописних цифр текстових документів.;

— вхідні дані: роздільна здатність зображення тексту не менше 300 dpi, оптична щільність зображення не менше 2,5Т, кількість градацій яскравості зображення — 256.

4 Вимоги до виконання БКР

Запропонувати нові підходи для реалізації методу пошуку та розпізнавання рукописних цифр текстових документів.

Розробити послідовність для розпізнавання рукописних цифр текстових документів у системах електронного документообігу.

Виконати розробку програмного забезпечення для розпізнавання рукописних цифр, провести його тестування. Схеми послідовності розпізнавання рукописних цифр та лістинги програми представити в додатках до роботи.

5 Етапи БКР та очікувані результати приведені у таблиці А.1.

Таблиця А.1 — Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Постановка задачі роботи. Вступ	21.03.25	23.03.25	Вступ
2	Аналіз методів та способів виділення та розпізнавання рукописних символів	24.03.25	30.03.25	розділ 1
3	Розробка способу та послідовності розпізнавання рукописних символів	01.04.25	21.04.25	Розділ 2, розробка способу
4	Розробка програми розпізнавання рукописних символів	22.04.25	05.05.25	Розділ 3, розробка програми
5	Тестування програми розпізнавання рукописних цифр, результати	06.05.25	10.05.25	Розділ 3
6	Оформлення пояснювальної записки та ілюстративного матеріалу	11.05.25	12.05.25	ПЗ, презентація
7	Перевірка якості виконання бакалаврської роботи та усунення недоліків	13.05.25	13.05.25	ПЗ, презентація

6 Матеріали, що подаються до захисту БКР: пояснювальна записка БКР, графічні і ілюстративні матеріали, протокол попереднього захисту БКР на кафедрі, відзив наукового керівника, відзив рецензента, протоколи складання державних екзаменів, анотації до БКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту БКР

Виконання етапів розрахункової та графічної документації БКР контролюється науковим керівником згідно зі встановленими термінами. Захист БКР відбувається на засіданні Державної екзаменаційної комісії, затвердженою наказом ректора.

8 Вимоги до оформлення БКР

8.1 При оформлювання БКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— Методичні вказівки до виконання бакалаврських кваліфікаційних робіт зі спеціальності 123 «Комп'ютерна інженерія» (освітня програма «Комп'ютерна інженерія»). Кафедра обчислювальної техніки ВНТУ 2022;

— документами, на які посилаються у вище вказаних.

8.2 Порядок виконання БКР викладено в «Положення про кваліфікаційні роботи на першому (бакалаврському) рівні вищої освіти СУЯ ВНТУ-03.02.02- П.001.01:21»

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Комп'ютерна система розпізнавання рукописних цифр на основі нейронної мережі.

Тип роботи: бакалаврська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 40%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Азаров О. Д., зав. каф. ОТ _____ _____
(прізвище, ініціали, посада) (підпис)

Крупельницький Л. В., доц. каф ОТ _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Захарченко С. М.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник роботи_____ Колесник І. С., доц. каф. ОТ
(підпис) (прізвище, ініціали, посада)

Здобувач_____ Курочкін Д. О.
(підпис) (прізвище, ініціали)

ДОДАТОК В

Графічна частина

КОМП’ЮТЕРНА СИСТЕМА РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

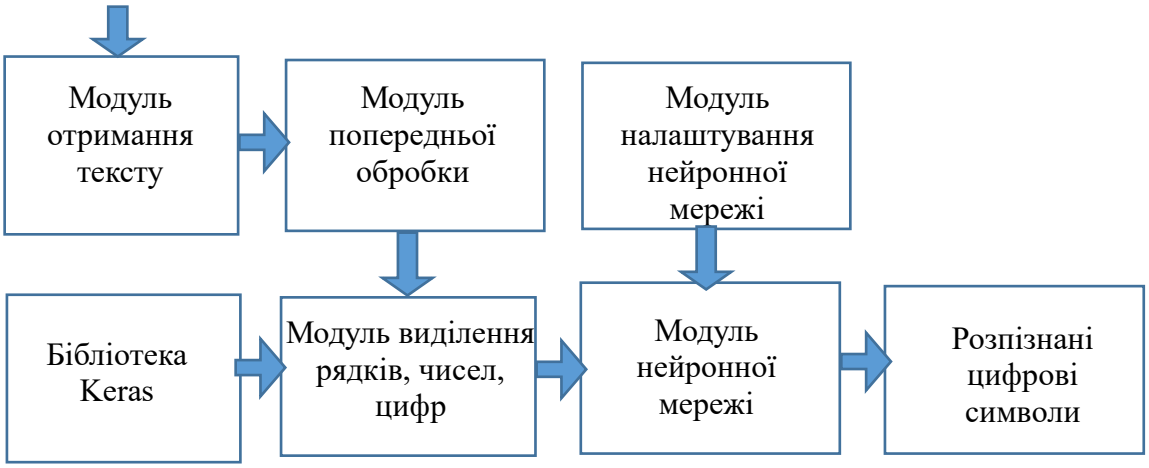

Рисунок В.1 — Структура програми розпізнавання рукописних цифр

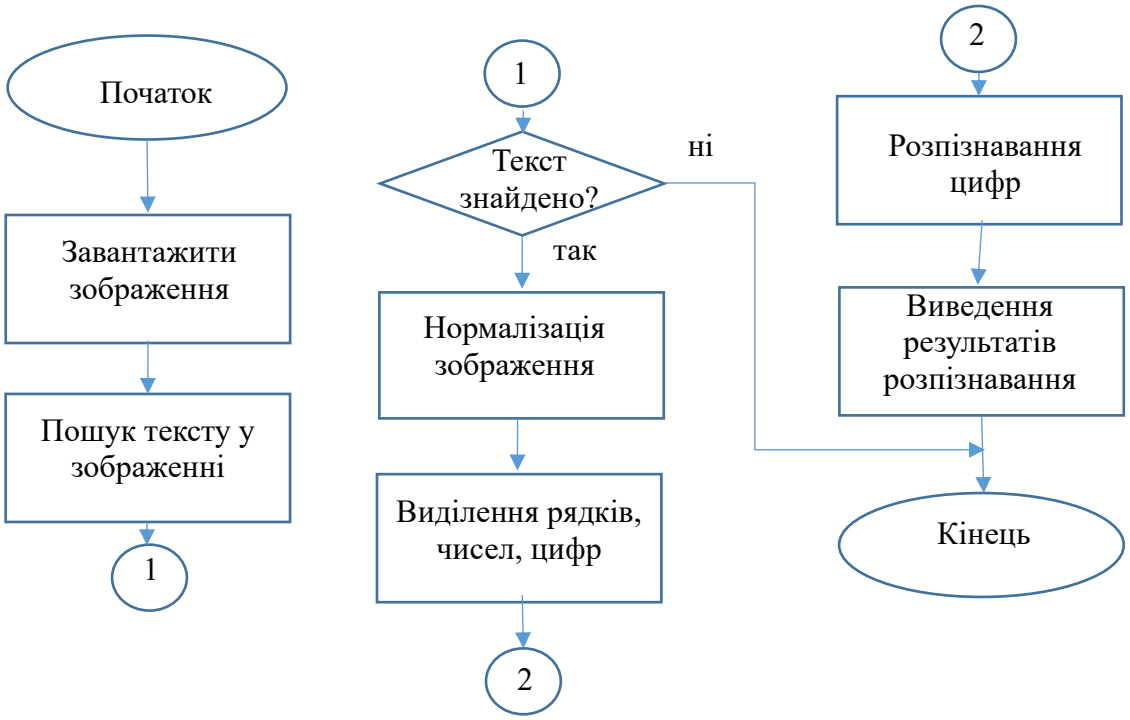

Рисунок В.2 — Блок-схема послідовності розпізнавання рукописних цифр

ДОДАТОК Г

Ілюстративна частина

КОМП’ЮТЕРНА СИСТЕМА РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

Рисунок Г.1 — Методи розпізнавання символів текстових документів

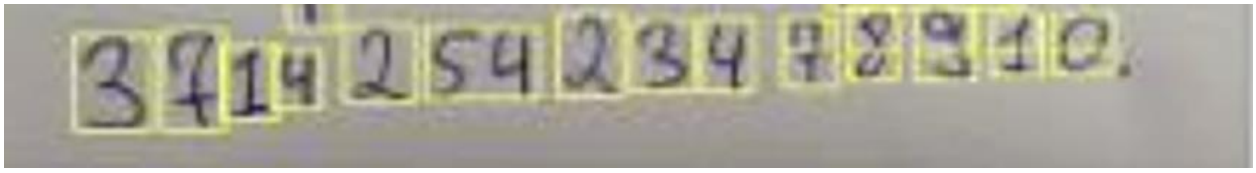

Рисунок Г.2 — Результати розподілу числа на виділені цифри

Рисунок Г.3— Результат виділення рукописних чисел у тексті

ДОДАТОК Д

Лістинг програми

```
import numpy as np
import matplotlib.pyplot as plt
import keras
import sys

image_file = "text.png"
img = cv2.imread(image_file)
gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
ret, thresh = cv2.threshold(gray, 0, 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)
output = img.copy()
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), (70, 0, 0), 1)
cv2.imshow("Input", img)
cv2.imshow("Enlarged", img_erode)
cv2.imshow("Output", output)
cv2.waitKey(0)
      def letters_extract(image_file: str, out_size=28) -> List[Any]:
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)))
      letters.sort(key=lambda x: x[0], reverse=False)
      return letters
cv2.imshow("0", letters[0][2])
cv2.imshow("1", letters[1][2])
cv2.imshow("2", letters[2][2])
cv2.imshow("3", letters[3][2])
cv2.imshow("4", letters[4][2])
cv2.waitKey(0)
from tensorflow import keras
from keras.models import Sequential
from keras import optimizers
from keras.layers import Convolution2D, MaxPooling2D, Dropout, Flatten, Dense,
Reshape, LSTM, BatchNormalization
```

```
        from keras.optimizers import SGD, RMSprop, Adam
        from keras import backend as K
        from keras.constraints import maxnorm
        import tensorflow as tf
        def emnist_model():
            model = Sequential()
            model.add(Convolution2D(filters=32, kernel_size=(3, 3), padding='valid',
input_shape=(28, 28, 1), activation='relu'))
            model.add(Convolution2D (filters=64, kernel_size=(3, 3), activation = 'relu'))
            model.add(MaxPooling2D(pool_size=(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='adadelta',
metrics=['accuracy'])
            return model
        import idx2numpy
        emnist_path = '/home/Documents/TestApps/keras/emnist/'
        X_train = idx2numpy.convert_from_file(emnist_path + 'emnist-byclass-train-images-
idx3-ubyte')
        y_train = idx2numpy.convert_from_file(emnist_path + 'emnist-byclass-train-labels-idx1-
ubyte')
        X_test = idx2numpy.convert_from_file(emnist_path + 'emnist-byclass-test-images-idx3-
ubyte')
        y_test = idx2numpy.convert_from_file(emnist_path + 'emnist-byclass-test-labels-idx1-
ubyte')
        X_train = np.reshape(X_train, (X_train.shape[0], 28, 28, 1))
        X_test = np.reshape(X_test, (X_test.shape[0], 28, 28, 1))
        print(X_train.shape, y_train.shape, X_test.shape, y_test.shape, len(emnist_labels))
        X_train = X_train[:X_train.shape[0] ]
        y_train = y_train[:y_train.shape[0] ]
        X_test = X_test[:X_test.shape[0]]
        y_test = y_test[:y_test.shape[0]]
        X_train = X_train.astype(np.float32)
        X_train /= 255.0
        X_test = X_test.astype(np.float32)
        X_test /= 255.0
        x_train_cat = keras.utils.to_categorical(y_train, len(emnist_labels))
        y_test_cat = keras.utils.to_categorical(y_test, len(emnist_labels))
        model = keras.models.load_model('emnist_letters.h5')
        def emnist_predict_img(model, img):
            img_arr = np.expand_dims(img, axis=0)
            img_arr = 1 - img_arr/255.0
            img_arr[0] = np.rot90(img_arr[0], 3)
            img_arr[0] = np.fliplr(img_arr[0])
            img_arr = img_arr.reshape((1, 28, 28, 1))
            result = model.predict_classes([img_arr])
            return chr(emnist_labels[result[0]])
        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 += emnist_predict_img(model, letters[i][2])
        if (dn > letters[i][1]/4):
          s_out += ' '
    return s_out

def loadBinData(pathToData):
    print('Завантаження файлу')
    with open(pathToData + 'imagesTrain.bin', 'rb') as read_binary:
        data = np.fromfile(read_binary, dtype = np.uint8)
     with open(pathToData + 'labelsTrain.bin', 'rb') as read_binary:
            labels = np.fromfile(read_binary, dtype = np.uint8)
    with open(pathToData + 'imagesTest.bin', 'rb') as read_binary:
        data2 = np.fromfile(read_binary, dtype = np.uint8)
     with open(pathToData + 'labelsTest.bin', 'rb') as read_binary:
        labels2 = np.fromfile(read_binary, dtype = np.uint8)
    return data, labels, data2, labels2
def load_mnist(pathToData, saveData):
    from mnist import MNIST
    mndata = MNIST(pathToData)
    mndata.gz = True
      imagesTrain, labelsTrain = mndata.load_training()
    imagesTest, labelsTest = mndata.load_testing()
    if saveData:
        fn = open(pathToData + 'imagesTrain.bin', 'wb')
        fn2 = open(pathToData + 'labelsTrain.bin', 'wb')
        fn3 = open(pathToData + 'imagesTest.bin', 'wb')
        fn4 = open(pathToData + 'labelsTest.bin', 'wb')
        fn.write(np.uint8(imagesTrain))
        fn2.write(np.uint8(labelsTrain))
        fn3.write(np.uint8(imagesTest))
        fn4.write(np.uint8(labelsTest))
        fn.close()
        fn2.close()
        fn3.close()
        fn4.close()
    return imagesTrain, labelsTrain, imagesTest, labelsTest
def makeNames(imgType):
    names = []
    if imgType == 'MNIST':
        for i in range(10): names.append(chr(48 + i))
    elif imgType == 'EMNIST':
        for i in range(26): names.append(chr(10 + i)) # ['0', '1', '2', ..., '9']
    elif imgType == 'CIFAR10':
    elif imgType == 'CIFAR100':
    return names
def load_data(allParams):
    num_classes = allParams[1]
```

```
        lossFunNum = allParams[3]
        categorical = False if lossFunNum == 10 else True
        img_rows = allParams[6]
        img_cols = allParams[7]
        pathToData = allParams[13]
        loadFromBin = allParams[14]
        show_img = allParams[19]
        saveData = allParams[20]
        nnType = allParams[24]
        useTestData = allParams[27]
        imgType = allParams[28]
        subtract_pixel_mean = allParams[33]
        if loadFromBin:
            imagesTrain, labelsTrain, imagesTest, labelsTest = loadBinData(pathToData)
        else:
            imagesTrain, labelsTrain, imagesTest, labelsTest = load_mnist(pathToData,
saveData)
        X_train = np.asarray(imagesTrain)
        y_train = np.asarray(labelsTrain)
        X_test = np.asarray(imagesTest)
        y_test = np.asarray(labelsTest)
        if imgType == 'EMNIST':
            y_train -= 1
            y_test -= 1
        if loadFromBin:
            X_train_shape_0 = int(X_train.shape[0] / (img_rows * img_cols)) # 60000 / 124800
            X_test_shape_0 = int(X_test.shape[0] / (img_rows * img_cols)) # 10000 / 20800
        else:
            X_train_shape_0 = X_train.shape[0]
            X_test_shape_0 = X_test.shape[0]
            print(X_train[0, :])
            print(y_train[0])
        if useTestData:
            allParams.append(X_test_shape_0)
            allParams.append(y_test)
        else:
            allParams.append(X_train_shape_0)
            allParams.append(y_train)
        K.set_image_data_format('channels_first')
        if nnType == 'conv' or show_img:
            if imgType == 'MNIST':
                X_train = X_train.reshape(X_train_shape_0, img_rows, img_cols, 1)
                X_test = X_test.reshape(X_test_shape_0, img_rows, img_cols, 1)
            elif imgType == 'EMNIST':
                X_train = X_train.reshape(X_train_shape_0, img_rows, img_cols,
1).transpose(0,2,1,3)
                X_test = X_test.reshape(X_test_shape_0, img_rows, img_cols,
1).transpose(0,2,1,3)
        elif nnType == 'mlp':

            im_size = img_rows * img_cols
            X_train = X_train.reshape(X_train_shape_0, im_size)
```

```
            X_test = X_test.reshape(X_test_shape_0, im_size)
        elif nnType == 'lstm':
            if imgType == 'MNIST':
                X_train = X_train.reshape(X_train_shape_0, img_rows, img_cols)
                X_test = X_test.reshape(X_test_shape_0, img_rows, img_cols)
            elif imgType == 'EMNIST':
                X_train = X_train.reshape(X_train_shape_0, img_rows,
img_cols).transpose(0,2,1)
                X_test = X_test.reshape(X_test_shape_0, img_rows, img_cols).transpose(0,2,1)
        if show_img:
            if useTestData:
            else:
            print(X_train.shape)
            print(y_train.shape)
            names = makeNames(imgType)
            for i in range(9):
                plt.subplot(3, 3, i + 1)
                ind = y_test[i] if useTestData else y_train[i]
                img = X_test[i][0:img_rows, 0:img_rows, 0] if useTestData else
X_train[i][0:img_rows, 0:img_rows, 0]
                plt.imshow(img, cmap = plt.get_cmap('gray'))
                plt.title(names[ind])
                plt.axis('off')
            plt.subplots_adjust(hspace = 0.5) # wspace
            plt.show()
            sys.exit()
        X_train = X_train.astype('float32')
        X_test = X_test.astype('float32')
```

Лістинг програми моделі мережі

```
import numpy as np
import tensorflow as tf

import vgg16
import rpn
import detector
import math_utils

class FasterRCNNModel(tf.keras.Model):
  def __init__(self, num_classes, allow_edge_proposals, custom_roi_pool,
activate_class_outputs, l2 = 0, dropout_probability = 0):
    super().__init__()
    self._num_classes = num_classes
    self._activate_class_outputs = activate_class_outputs
    self._stage1_feature_extractor = vgg16.FeatureExtractor(l2 = l2)
    self._stage2_region_proposal_network = rpn.RegionProposalNetwork(
      max_proposals_pre_nms_train = 12000,
      max_proposals_post_nms_train = 2000,
      max_proposals_pre_nms_infer = 6000,
      max_proposals_post_nms_infer = 300,
      l2 = l2,
      allow_edge_proposals = allow_edge_proposals
```

```
  )
  self._stage3_detector_network = detector.DetectorNetwork(
    num_classes = num_classes,
    custom_roi_pool = custom_roi_pool,
    activate_class_outputs = activate_class_outputs,
    l2 = l2,
    dropout_probability = dropout_probability
  )
def call(self, inputs, training=False):
  input_image = inputs[0]
  anchor_map = inputs[1]
  anchor_valid_map = inputs[2]
  if training:
    gt_rpn_map = inputs[3]
    gt_box_class_idxs_map = inputs[4]
    gt_box_corners_map = inputs[5]
  feature_map = self._stage1_feature_extractor(input_image = input_image, training = training)
  rpn_scores, rpn_box_deltas, proposals = self._stage2_region_proposal_network(
    inputs = [
      input_image,
      feature_map,
      anchor_map,
      anchor_valid_map
    ],
    training = training
  )
  if training:
    proposals, gt_classes, gt_box_deltas = self._label_proposals(
      proposals = proposals,
      gt_box_class_idxs = gt_box_class_idxs_map[0],
      gt_box_corners = gt_box_corners_map[0],
      min_background_iou_threshold = 0.0,
      min_object_iou_threshold = 0.5
    )
    proposals, gt_classes, gt_box_deltas = self._sample_proposals(
      proposals = proposals,
      gt_classes = gt_classes,
      gt_box_deltas = gt_box_deltas,
      max_proposals = 128,
      positive_fraction = 0.25
    )
    gt_classes = tf.expand_dims(gt_classes, axis = 0)
              gt_box_deltas = tf.expand_dims(gt_box_deltas, axis = 0)   #
              proposals = tf.stop_gradient(proposals)
    gt_classes = tf.stop_gradient(gt_classes)
    gt_box_deltas = tf.stop_gradient(gt_box_deltas)
  detector_classes, detector_box_deltas = self._stage3_detector_network(
    inputs = [
      input_image,
      feature_map,
      proposals
    ],
```

```
      training = training
    )
    if training:
      rpn_class_loss = self._stage2_region_proposal_network.class_loss(y_predicted = rpn_scores,
gt_rpn_map = gt_rpn_map)
      rpn_regression_loss = self._stage2_region_proposal_network.regression_loss(y_predicted =
rpn_box_deltas, gt_rpn_map = gt_rpn_map)
      detector_class_loss = self._stage3_detector_network.class_loss(y_predicted =
detector_classes, y_true = gt_classes, from_logits = not self._activate_class_outputs)
      detector_regression_loss = self._stage3_detector_network.regression_loss(y_predicted =
detector_box_deltas, y_true = gt_box_deltas)
      self.add_loss(rpn_class_loss)
      self.add_loss(rpn_regression_loss)
      self.add_loss(detector_class_loss)
      self.add_loss(detector_regression_loss)
      self.add_metric(rpn_class_loss, name = "rpn_class_loss")
      self.add_metric(rpn_regression_loss, name = "rpn_regression_loss")
      self.add_metric(detector_class_loss, name = "detector_class_loss")
      self.add_metric(detector_regression_loss, name = "detector_regression_loss")
    else:
      rpn_class_loss = float("inf")
      rpn_regression_loss = float("inf")
      detector_class_loss = float("inf")
      detector_regression_loss = float("inf")
    return [
      rpn_scores,
      rpn_box_deltas,
      detector_classes,
      detector_box_deltas,
      proposals,
      rpn_class_loss,
      rpn_regression_loss,
      detector_class_loss,
      detector_regression_loss
    ]
  def predict_on_batch(self, x, score_threshold):
    _, _, detector_classes, detector_box_deltas, proposals, _, _, _, _ = super().predict_on_batch(x
= x)
    scored_boxes_by_class_index = self._predictions_to_scored_boxes(
      input_image = x[0],
      classes = detector_classes,
      box_deltas = detector_box_deltas,
      proposals = proposals,
      score_threshold = score_threshold
    )
    return scored_boxes_by_class_index
  def load_imagenet_weights(self):
    keras_model = tf.keras.applications.VGG16(weights = "imagenet")
    for keras_layer in keras_model.layers:
      weights = keras_layer.get_weights()
      if len(weights) > 0:
        vgg16_layers = self._stage1_feature_extractor.layers +
```

```
self._stage3_detector_network.layers
        our_layer = [ layer for layer in vgg16_layers if layer.name == keras_layer.name ]
        if len(our_layer) > 0:
          print("Loading VGG-16 ImageNet weights into layer: %s" % our_layer[0].name)
          our_layer[0].set_weights(weights)
  def _predictions_to_scored_boxes(self, input_image, classes, box_deltas, proposals,
score_threshold):
    input_image = np.squeeze(input_image, axis = 0)
    classes = np.squeeze(classes, axis = 0)
    box_deltas = np.squeeze(box_deltas, axis = 0)
    if not self._activate_class_outputs:
      classes = tf.nn.softmax(classes, axis = 1).numpy()
       proposal_anchors = np.empty(proposals.shape)
    proposal_anchors[:,0] = 0.5 * (proposals[:,0] + proposals[:,2])
    proposal_anchors[:,1] = 0.5 * (proposals[:,1] + proposals[:,3])
    proposal_anchors[:,2:4] = proposals[:,2:4] - proposals[:,0:2]
    boxes_and_scores_by_class_idx = {}
    for class_idx in range(1, classes.shape[1]):
      box_delta_idx = (class_idx - 1) * 4
      box_delta_params = box_deltas[:, (box_delta_idx + 0) : (box_delta_idx + 4)]
proposal_boxes_this_class = math_utils.convert_deltas_to_boxes(
        box_deltas = box_delta_params,
        anchors = proposal_anchors,
        box_delta_means = [0.0, 0.0, 0.0, 0.0],
        box_delta_stds = [0.1, 0.1, 0.2, 0.2]
      )
      proposal_boxes_this_class[:,0::2] = np.clip(proposal_boxes_this_class[:,0::2], 0,
input_image.shape[0] - 1)
      proposal_boxes_this_class[:,1::2] = np.clip(proposal_boxes_this_class[:,1::2], 0,
input_image.shape[1] - 1)
      scores_this_class = classes[:,class_idx]
      sufficiently_scoring_idxs = np.where(scores_this_class > score_threshold)[0]
      proposal_boxes_this_class = proposal_boxes_this_class[sufficiently_scoring_idxs]
      scores_this_class = scores_this_class[sufficiently_scoring_idxs]
      boxes_and_scores_by_class_idx[class_idx] = (proposal_boxes_this_class, scores_this_class)
    scored_boxes_by_class_idx = {}
    for class_idx, (boxes, scores) in boxes_and_scores_by_class_idx.items():
      idxs = tf.image.non_max_suppression(
        boxes = boxes,
        scores = scores,
        max_output_size = proposals.shape[0],
        iou_threshold = 0.3
      )
      idxs = idxs.numpy()
      boxes = boxes[idxs]
      scores = np.expand_dims(scores[idxs], axis = 0)
      scored_boxes = np.hstack([ boxes, scores.T ])
      scored_boxes_by_class_idx[class_idx] = scored_boxes
    return scored_boxes_by_class_idx
  def _label_proposals(self, proposals, gt_box_class_idxs, gt_box_corners,
min_background_iou_threshold, min_object_iou_threshold):
    proposals = tf.concat([ proposals, gt_box_corners ], axis = 0)
```

```
    ious = math_utils.tf_intersection_over_union(boxes1 = proposals, boxes2 = gt_box_corners)
    best_ious = tf.math.reduce_max(ious, axis = 1)
    box_idxs = tf.math.argmax(ious, axis = 1)
            gt_box_class_idxs = tf.gather(gt_box_class_idxs, indices = box_idxs)
    gt_box_corners = tf.gather(gt_box_corners, indices = box_idxs)
    idxs = tf.where(best_ious >= min_background_iou_threshold)
    proposals = tf.gather_nd(proposals, indices = idxs)
    best_ious = tf.gather_nd(best_ious, indices = idxs)
    gt_box_class_idxs = tf.gather_nd(gt_box_class_idxs, indices = idxs)
    gt_box_corners = tf.gather_nd(gt_box_corners, indices = idxs)
    retain_mask = tf.cast(best_ious >= min_object_iou_threshold, dtype =
gt_box_class_idxs.dtype)
    gt_box_class_idxs = gt_box_class_idxs * retain_mask
       num_classes = self._num_classes
    gt_classes = tf.one_hot(indices = gt_box_class_idxs, depth = num_classes)    proposal_centers
= 0.5 * (proposals[:,0:2] + proposals[:,2:4
            proposal_sides = proposals[:,2:4] - proposals[:,0:2]
    gt_box_centers = 0.5 * (gt_box_corners[:,0:2] + gt_box_corners[:,2:4])      gt_box_sides =
gt_box_corners[:,2:4] - gt_box_corners[:,0:2]
      detector_box_delta_means = tf.constant([0, 0, 0, 0], dtype = tf.float32)
    detector_box_delta_stds = tf.constant([0.1, 0.1, 0.2, 0.2], dtype = tf.float32)
    tyx = (gt_box_centers - proposal_centers) / proposal_sides
    thw = tf.math.log(gt_box_sides / proposal_sides)
    box_delta_targets = tf.concat([ tyx, thw ], axis = 1)
    box_delta_targets = (box_delta_targets - detector_box_delta_means) /
detector_box_delta_stds
    gt_box_deltas_mask = tf.repeat(gt_classes, repeats = 4, axis = 1)[:,4:]
gt_box_deltas_values = tf.tile(box_delta_targets, multiples = [1, num_classes - 1])
    gt_box_deltas_mask = tf.expand_dims(gt_box_deltas_mask, axis = 0)
gt_box_deltas_values = tf.expand_dims(gt_box_deltas_values, axis = 0)    gt_box_deltas =
tf.concat([ gt_box_deltas_mask, gt_box_deltas_values ], axis = 0)    gt_box_deltas =
tf.transpose(gt_box_deltas, perm = [ 1, 0, 2])
    return proposals, gt_classes, gt_box_deltas
  def _sample_proposals(self, proposals, gt_classes, gt_box_deltas, max_proposals,
positive_fraction):
    if max_proposals <= 0:
      return proposals, gt_classes, gt_box_deltas
        class_indices = tf.argmax(gt_classes, axis = 1)
    positive_indices = tf.squeeze(tf.where(class_indices > 0), axis = 1)
    negative_indices = tf.squeeze(tf.where(class_indices <= 0), axis = 1)
    num_positive_proposals = tf.size(positive_indices)
    num_negative_proposals = tf.size(negative_indices)
    num_samples = tf.minimum(max_proposals, tf.size(class_indices))
    num_positive_samples = tf.minimum(tf.cast(tf.math.round(tf.cast(num_samples, dtype =
float) * positive_fraction), dtype = num_samples.dtype), num_positive_proposals)
    num_negative_samples = tf.minimum(num_samples - num_positive_samples,
num_negative_proposals)
    positive_sample_indices = tf.random.shuffle(positive_indices)[:num_positive_samples]
    negative_sample_indices = tf.random.shuffle(negative_indices)[:num_negative_samples]
    indices = tf.concat([ positive_sample_indices, negative_sample_indices ], axis = 0)
    return tf.gather(proposals, indices = indices), tf.gather(gt_classes, indices = indices),
tf.gather(gt_box_deltas, indices = indices)
```