Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему: Розробка методів розпізнавання рукописного тексту за допомогою нейромереж

Виконав: студент II курсу
групи 1ПІ-18 м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Пупко О. В.
(прізвище та ініціали)

Керівник: к.т.н., доц. Майданюк В. П.
(прізвище та ініціали)

Рецензент: к. т. н., доц. Богач І. В.
(прізвище та ініціали)

ВНТУ – 2019

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Освітньо-кваліфікаційний рівень – магістр

Спеціальність 121 – інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ

Романюк О. Н.

«___» _____ 2019 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Пупку Олександру Валерійовичу

1. Тема роботи – розробка методів розпізнавання рукописного тексту за допомогою нейромереж.

Керівник роботи: Майданюк Володимир Павлович, к.т.н., доц. каф. ПЗ, затверджені наказом вищого навчального закладу від "__"_____2019 року №___

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи: операційна система – Unix*, мова програмування – Python, технології – Veles Learning Machine.

4. Зміст розрахунково-пояснювальної записки: вступ, обґрунтування доцільності розробки та постановка задачі дослідження, розробка методів попередньої обробки та розпізнавання тексту, розробка програмних модулів системи розпізнавання рукописного тексту, економічна частина, висновки, додатки.

5. Перелік графічного матеріалу: алгоритми роботи первинної обробки зображень рукописного тексту та шарів згорткової нейронної мережі, результати

тестування кроків попередньої обробки, загальна схема роботи додатку, структурна та графічні схеми інтерфейсу.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
1-3	Майданюк В. П., к.т.н., доцент кафедри ПЗ		
4	Бальзан М. В., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Обґрунтування доцільності розробки та постановка задачі дослідження	04.09.2019 – 29.09.2019	Виконано
2	Розробка методів обробки та розпізнавання тексту	30.09.2019 – 26.10.2019	Виконано
3	Розробка програмних модулів системи обробки рукописного тексту	27.10.2019 – 12.11.2019	Виконано
4	Економічна частина	13.11.2019 – 17.11.2019	Виконано

Студент _____ Пупко О. В.
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ Майданюк В. П.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі «Розробка методів розпізнавання рукописного тексту за допомогою нейромереж» розроблено консольний додаток, який проводить попередню обробку зображень та дозволяє розпізнавати стародавні рукописні тексти мовою іврит.

В ході виконання проаналізовано сучасні методи первинної обробки зображень рукописного тексту та системи-аналоги. Обґрунтовано вибір засобів для розробки програмного забезпечення. До них належить розподілена програмна платформа Veles Machine Learning, середовище програмування PyCharm 2019, технології Python; фреймворки – OpenCV, NumPy, Pyplot. Розроблено модулі попередньої обробки зображень та систему розпізнавання рукописних текстів за допомогою згорткової нейронної мережі і підходу ковзного вікна.

ABSTRACT

In master's qualification thesis on «Development of methods of handwriting recognition using neural networks» developed a console application that performs pre - processing of images and allows you to recognize ancient handwritten texts in Hebrew.

In the course of implementation, modern methods of primary processing of handwritten text images and analog systems are analyzed. The choice of tools for software development is justified. These include the distributed software platform Veles Machine Learning, programming environment PyCharm 2019, Python technologies; frameworks – OpenCV, NumPy, Pyplot. Image preprocessing modules and handwriting recognition system using convolutional neural network and sliding window approach have been developed.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

AI (Artificial Intelligence) – штучний інтелект

ML (Machine Learning) – машинне навчання

CNN (Convolutional Neural Network) – згорткова нейронна мережа

ROC (Receiver Operating Characteristic) – робоча характеристика приймача

OCR (Optical Character Recognition) – оптичне розпізнавання символів

ВСТУП

Обґрунтування вибору теми дослідження. Рукопис залишається засобом спілкування та збору інформації в повсякденному житті навіть із впровадженням нових технологій. Більш того, величезна кількість історичних колекцій не є доступна у зручному форматі. В даний час підходи та способи перетворення зображень в цифровий текст розвиваються швидкими темпами, навіть незважаючи на те, що все ще є можливості для покращення та вирішення багатьох завдань.

В останні роки системи розпізнавання друкованого тексту стали досить ефективними. Рукописні цифри та інтерактивне написання розпізнаються якісно. Однак поточні технології все ще знаходиться на обмеженому рівні, щоб розпізнавати текстові зображення різних стилів та мов рукописного вводу [1].

У цьому документі запропонований підхід системи розпізнавання рукописного тексту, починаючи з ідеї, що хороша попередня обробка створює основу для успішного розпізнавання. До визнання себе як попередньої обробки підхід включає в себе бінаризацію, пошук текстової області, подальшу обробку, такі як коригування фону, сегментацію ліній, реконструкцію символів та обробку шумів.

Системи розпізнавання рукописного тексту можуть відрізнятися в багатьох аспектах і залежать від конкретних завдань. Бінаризація може виконуватися як перший крок або після сегментованих рядків або символів. Наш підхід використовує його як перший крок, оскільки наступні кроки потребують бінарних зображень, і, крім того, він ефективний завдяки появі вхідних зображень. Потім знайдеться текстова область, і зображення буде налаштовано таким чином, що є білий фон та чорні літери. Цей крок є досить значним, оскільки він впливає на етап сегментації ліній. Слід зазначити, що проекція гістограми для сегментації ліній є звичайною практикою. У цьому документі підхід до сегментації ліній змінюється способом пошуку піків у гістограмі. Слід

розглянути питання про деградовані історичні документи, реконструкцію символів. На цьому кроці запропонована система вирішує проблему шуму.

Після отримання текстових рядків виникає складність виконання точної сегментації символів. Існує практика сегментації букв та їх класифікації окремо, наприклад, використовуючи аналіз підключених компонентів або проектування гістограми лінії. Проте деградовані або перекриваючі символи можуть бути розпізнані неправильно. Тому завдання з розпізнавання рукописного тексту є актуальним.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення.

Мета та завдання дослідження. Метою роботи є підвищення достовірності розпізнавання рукописних текстів мовою іврит за рахунок удосконалення методів попередньої обробки зображень.

Основними задачами дослідження є:

- аналіз існуючих методів і засобів попередньої обробки зображень для визначення напрямків підвищення їх продуктивності та достовірності;
- модифікація методу попередньої обробки зображень рукописних текстів мовою іврит;
- розробка комплексного методу розпізнавання літер;
- розробка програмних компонент та систему розпізнавання рукописних текстів на основі запропонованих методів;
- проведення експериментальних досліджень розроблених засобів розпізнавання рукописних текстів.

Об'єкт дослідження – процес розпізнавання рукописних текстів.

Предмет дослідження – методи та програмні засоби первинної обробки зображень рукописних текстів на стародавньому івриті.

Методи дослідження. У процесі досліджень використовувались: теорія чисел та чисельних методів, лінійна алгебра, теорія розпізнавання образів для розробки методів та програмних модулів розпізнавання текстів мовою іврит;

комп’ютерне моделювання для аналізу та перевірки отриманих теоретичних положень.

Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод первинної обробки зображень при розпізнаванні рукописних текстів мовою іврит, який відрізняється від існуючих введенням додаткових кроків, таких як: бінаризація вихідного зображення, виявлення області тексту, сегментація текстових рядків, реконструкції символів та обробки шумів, моделі класифікатора символів, збагачення даних, сегментація символів на основі класифікатора, що покращує вихідне зображення, і як наслідок, збільшує достовірність розпізнавання.
2. Вперше розроблено метод розпізнавання літер за допомогою комплексного підходу, особливість якого полягає в обробці з використанням ковзного вікна і згорткової нейронної мережі, що підвищує ймовірність прогнозування символів.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби розпізнавання рукописного тексту за допомогою нейромереж.

Структура та обсяг роботи. Магістерська кваліфікаційна роботи складається зі вступу, чотирьох розділів, висновків, списку літератури, що містить 45 найменувань, 1 додаток. Робота містить 15 ілюстрацій, 20 таблиць.

У розділі 1 описано обґрунтування доцільності розробки. У розділі 2 описані запропоновані методи та всі його кроки. У розділі 3 представлені та проаналізовані експериментальні результати. Нарешті, деякі дискусії та заключні зауваження наведені в розділі 4.

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ РОЗРОБКИ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

Інформаційні технології проникли у життя соціуму у всіх його сферах, допомагаючи людству у розвитку. Яскравим прикладом цього є розробка нової форми розуму, а саме, – штучного інтелекту.

1.1 Штучний інтелект

Штучний інтелект є галуззю інформатики, метою якої є створення інтелектуальних машин. Це стало важливою частиною технологічної галузі.

Дослідження, пов'язані з штучним інтелектом, дуже технічні та спеціалізовані. Основні проблеми штучного інтелекту включають в себе програмування комп'ютерів для певних рис, таких як: знання, міркування, вирішення проблем, сприйняття, навчання, планування, можливість маніпулювати та переміщати об'єкти.

Інжиніринг знань є основною частиною досліджень AI. Машини часто можуть діяти і реагувати, як людина, лише в тому випадку, якщо вони мають багату інформацію, що стосується світу. Штучний інтелект повинен мати доступ до об'єктів, категорій, властивостей та відносин між ними для здійснення знань. Ініціювання здорового глузду, міркування та вирішення проблем у машинах - складне і нудне завдання.

Машинне навчання також є основною частиною AI. Навчання без будь-якого нагляду вимагає здатності виявляти закономірності у потоках вхідних даних, тоді як навчання з адекватним наглядом передбачає класифікацію та чисельні регресії. Класифікація визначає категорію, до якої об'єкт належить, і операції регресії з отриманням набору числових вхідних або вихідних прикладів, тим самим відкриваючи функції, що дозволяють створювати відповідні виходи з відповідних входів. Математичний аналіз алгоритмів машинного навчання та їх

продуктивність є чітко визначеною гілкою теоретичної інформатики, яку часто називають теорією обчислювальної навчання.

Машинне сприйняття стосується здатності використовувати сенсорні входи для виведення різних аспектів світу, а комп'ютерне бачення – для аналізу візуальних входів з кількома під-проблемами, такими як розпізнавання обличчя, об'єкта та жестів.

Робототехніка також є основним напрямком, пов'язаним з AI. Роботи вимагають інтелекту для вирішення таких завдань, як маніпулювання об'єктами та навігація, а також підпроблеми локалізації, планування руху та картографування.

Розробка штучного інтелекту як напряму бере розвиток з наук як психологія, математика , нейрофізіологія, та інформаційні технології.

Термін «штучний інтелект» був створений у 1956 році, однак AI став більш популярним сьогодні завдяки збільшенню об'ємів даних, розширенню алгоритмів та покращенню обчислювальної потужності.

Раннє дослідження AI у 1950-х роках вивчало такі теми, як вирішення проблем та символічні методи. У 1960-х роках Міністерство оборони США зацікавилось подібною роботою і почало навчання комп'ютерів, щоб імітувати базові людські міркування. Наприклад, в 70-х роках минулого століття в Агентстві передових дослідницьких проектів у галузі оборони (DARPA) було завершено проекти з планування вулиць. І DARPA виготовив інтелектуальних персональних помічників у 2003 році, задовго до того, як Siri, Alexa або Cortana стали персональними помічниками.

Ця рання робота проклала шлях до автоматизації та формальних міркувань, які ми бачимо в сучасних комп'ютерах, включаючи системи підтримки прийняття рішень та інтелектуальні пошукові системи, які можуть бути розроблені для доповнення та покращення людських здібностей.

Хоча голлівудські фільми та науково-фантастичні романи зображують AI як людиноподібних роботів, які захоплюють світ, поточна еволюція технологій

AI не така страшна, або і зовсім не така розумна. Замість цього AI розвивається, щоб забезпечувати багато конкретних переваг у кожній конкретній галузі.

У найближчій перспективі мета збереження впливу AI на суспільство вигідна, є мотивацією досліджень у багатьох сферах – від економіки та права до технічних тем, таких як перевірка, валідація, безпека та контроль. У той час це може бути трохи більше, ніж допомога при незначних незручностях, якщо ваш електронний пристрій дає збій або зламаний, стає все важливішим, щоб система AI робила те, що ви хочете зробити, наприклад, якщо вона контролює вашу машину, ваш літак, ваш кардіостимулятор, вашу автоматичну торгівлю або вашу енергосистему. Ще одним короткостроковим завданням є запобігання спустошливої гонки озброєнь в умовах смертельної автономної зброї.

У довгостроковій перспективі важливим питанням є те, що станеться, якщо квест за суперінтелектом досягне успіху, і система AI стане кращою за людей у будь-яких сферах та когнітивних завданнях. Як зазначив І.Я. Гарний у 1965 році, проектування більш розумних систем AI є когнітивним завданням. Така система може потенційно зазнати рекурсивного самовдосконалення, викликаючи вибух інтелекту, що залишає людський інтелект далеко позаду. Винаходячи революційні нові технології, така суперінтелекція може допомогти нам усунути війну, хвороби та бідність, і тому створення суперінтелекту може бути найбільшою подією в історії людства. Деякі експерти висловлюють занепокоєння, що це також може бути останнім днем людства, якщо ми не навчимося узгоджувати цілі AI з нашими, перш ніж він стане суперінтелектуальним.

Дехто сумнівається, чи буде взагалі досягнутий суперінтелект, але водночас є і ті, хто наполягають на тому, що створення суперінтелектуального AI гарантовано буде корисним. Сьогоднішні дослідження допоможуть нам краще підготуватися до таких потенційно негативних наслідків і запобігти невдалому майбутньому, таким чином, користуючись перевагами AI та уникаючи підводних каменів.

AI автоматизує повторюване навчання та пошук за допомогою даних. Але AI відрізняється від апаратної та автоматизованої роботи. Замість автоматизації ручних завдань, AI надійно та без втоми виконує великомасштабні комп'ютеризовані завдання. Для даного типу автоматизації, людський запит, як і раніше, є необхідним для створення системи та запиту правильних питань.

AI додає інтелект для існуючих продуктів. У більшості випадків AI не буде поширюватися як самостійний продукт. Швидше за все, продукти, які ви вже використовуєте, будуть покращені завдяки можливостям AI, подібно до того, як додано Siri як функцію нового покоління продуктів Apple. Автоматизація, розмовні платформи, боти і інтелектуальні машини можуть поєднуватися з великою кількістю даних для вдосконалення багатьох технологій вдома та на робочому місці, від розвідки безпеки до інвестиційного аналізу.

AI адаптується за допомогою алгоритмів прогресивного навчання, щоб дати дані програмуванню. AI знаходить структуру та закономірності в даних, так що алгоритм набуває вміння: ставати класифікатором або предиктором.

Отже, так само, як алгоритм може навчити себе, як грати в шахи, він може навчити себе, який продукт рекомендувати в Інтернеті та адаптувати моделі при наданні нових даних. Зворотне розповсюдження – це техніка AI, яка дає змогу коригувати модель шляхом навчання та додавання даних, коли початкова відповідь не зовсім правильна.

AI аналізує більш глибокі дані, використовуючи нейронні мережі, що мають багато прихованих шарів. Створення системи виявлення шахрайства з п'ятьма прихованими шарами було майже неможливо кілька років тому, але все змінилося, з неймовірною потужністю комп'ютера та Big Data. Потрібно багато даних, щоб тренувати глибокі моделі навчання, оскільки вони навчаються безпосередньо з даних, чим більше даних ви можете згодувати моделям, тим точніше вони стануть.

AI досягає неймовірної точності через глибокі нейронні мережі – це було раніше неможливо. Наприклад, ваша взаємодія з Alexa, Google Search та Google Photos залежить від глибокого навчання, вони постійно стають точнішими, чим

більше ми їх використовуємо. У галузі медицини методика AI з глибокого вивчення, класифікації зображень та розпізнавання об'єктів тепер може бути використана для пошуку раку на МРТ з такою ж точністю, як і висококваліфіковані радіологи.

Коли алгоритми самонавчаються, самі дані можуть стати інтелектуальною власністю. Рішення завжди в даних; ви просто повинні застосувати AI, щоб їх вивести. Оскільки роль даних зараз важливіша, ніж будь-коли раніше, вона може створити конкурентну перевагу. Якщо у вас є найкращі дані у конкурентній галузі, навіть якщо конкуренти застосовують подібні методи, найкращі дані переможуть.

1.2 Розпізнавання тексту за допомогою машинного навчання

Розпізнавання рукописного тексту стало активною та складною областю досліджень. Система розпізнавання рукописного тексту відіграє дуже важливу роль у сучасному світі. Розпізнавання рукописного тексту дуже популярне і обчислювально дорога робота. В даний час дуже важко знайти правильне значення рукописних документів. Є багато областей, де нам потрібно розпізнавати слова, алфавіти та цифри. Наприклад поштові адреси додатків, банківська перевірка, де потрібно розпізнавати почерк. Цей документ буде зосереджений на різній техніці, яка використовується при розпізнаванні рукописного тексту. В основному, існує два різних типи розпізнавання рукописного тексту – в режимі онлайн та автономний режим. Для автономного режиму розпізнавання рукописного вводу існує багато підходів. Цей документ розкриває обмеження та переваги різної техніки, яка використовується для розпізнавання рукописного тексту.

Отже, розпізнавання рукописного тексту було вивчено з багатьох десятиліть. Система розпізнавання рукописного тексту може бути використана для вирішення багатьох складних проблем та полегшити роботу людям.

Розпізнавання рукописного вводу – це здатність розпізнавати текст за допомогою системи, яка отримує вхідні дані від сенсорного екрана, електронного пера, сканера, зображень або паперових документів. Система розпізнавання рукописного вводу в автономному режимі є мистецтвом ідентифікації слова з зображення. Як ми знаємо, кожна людина має свій різний стиль письма, тому дуже важко визнати правильні рукописні символи та цифри. Система розпізнавання рукописного тексту розроблена для досягнення точності та надійності роботи.

Отже, розпізнавання рукописного тексту є найбільш складною областю, якщо розпізнавати зображення та образ. Розпізнавання рукописного тексту дуже корисно в реальному світі та вирішує багато практичних проблем, наприклад, таких, як аналіз документації, інтерпретація поштової адреси, обробка банківських чеків, перевірка підпису, поштової адреси тощо. Використовуються декілька підходів, як розпізнавання рукописного вводу в режимі онлайн, так і в автономному режимі, такі як статистичні методи, структурні методи, нейронна мережа та синтаксичні методи. Деякі системи розпізнавання визначають штрихи, інші застосовують розпізнавання на одному символі або цілі слова.

Система оптичного розпізнавання символів була вивчена протягом останніх десятиліть. У 1914 році Еммануель Голдберг розробив систему, яка читає рукописні символи та цифри і потім конвертується в телеграфний код. У той же час Едмунд Фурньє д'Альбе розробив сканер Optofon, який сканує друковану сторінку та видає результат. Голдберг продовжував розвивати систему розпізнавання рукописного введення даних. Через деякий час він запропонував підібрати зображення з шаблонами, що містять потрібну інформацію. Цей метод відомий як спосіб відповідності шаблону.

Після цього Пауль У. Хандель також зареєстрував патент на технологію рукописного впорядкування шаблонів у США у 1933 році. У 1994 році інженери RCA запропонували перший комп'ютерний тип оптичного розпізнавання символів для допомоги сліпим людям. Він призначений для перетворення рукописного звіту в перфокарти для введення в комп'ютер для допомоги у

обробці 20-25 мільйонів книг на рік. У 1965 році Reader's Digest та RCA співпрацювали з метою створення оптичної системи розпізнавання символів. У 1985 році запропоновано структурні підходи зі статистичними методами. У цих системах символи розбиті на безліч моделей, таких як горизонтальні та вертикальні лінії та різні криві. У даному методі система зосереджена на формі символів. Після 1990 року було досягнуто реальний прогрес за допомогою нових методів та методологій обробки зображень та розпізнавання образів. У сучасному світі використовуються більш потужні комп'ютери та більш точні пристрої, такі як електронна ручка, сканер та планшети, а також багато підходів, таких як HMM, нейронна мережа, алгоритм зворотного поширення, нечітка нейронна мережа для розпізнавання рукописних документів.

Розпізнавання рукописних символів є одною з важливих проблем у програмах розпізнавання образів. Програми розпізнавання розрядів включають в себе сортування поштових відправлень, обробку банківських чеків, введення даних у формі тощо. Основним завданням є здатність розробляти ефективний алгоритм, який здатний розпізнавати цифри, написані рукописним методом і зчитані за допомогою сканера, планшета та інших цифрових пристроїв.

Незважаючи на велику кількість інструментів для написання технологічних текстів, багато людей все-таки вирішують робити свої нотатки традиційно: за допомогою ручки та паперу. Проте рукописний текст має недоліки.

Важко ефективно зберігати і отримувати доступ до фізичних документів, здійснювати їх пошук та ділитися ними з іншими.

Таким чином, багато важливих знань втрачаються або не переглядаються через те, що документи ніколи не конвертуються в цифровий формат. Таким чином, ми вирішуємо цю проблему у нашому проекті, оскільки, значно більша зручність керування цифровим текстом порівняно з рукописним текстом допоможе людям більш ефективно отримувати доступ до нього, шукати, ділитися ним та аналізувати записи, досі дозволяючи використовувати бажаний метод запису.

Виявлення рукописного тексту – це техніка або здатність комп'ютера отримувати та інтерпретувати рукописні дані з джерел, як, наприклад, паперові документи, сенсорний екран, фотографії тощо. Коли ми щось пишемо і подаємо як вхід до системи, то система розпізнає ці слова та рукописні документи. Цей процес виконується системою розпізнавання рукописного тексту. Отже, можна сказати, що система розпізнавання рукописного тексту може бути використана як паралельний термін. Симуляція машини людського письма – це область застосування системи розпізнавання рукописного тексту. Система розпізнавання рукописного тексту є способом взаємодії між людиною та машиною.

Декілька десятиліть система розпізнавання рукописного тексту є найбільш розвиненою дослідницькою сферою, тому що сьогодні потреба системи розпізнавання рукописного тексту зростає з кожним днем. Написання – це природний спосіб зберігання інформації та передачі інформації. Рукопис не тільки забезпечує спосіб спілкування між людьми, але й взаємодію між людиною та машиною.

Розпізнавання рукописного вводу – це здатність комп'ютера або мобільного пристрою читати рукопис як фактичний текст. Найпоширенішим випадком використання в сучасному мобільному світі є розпізнавання рукописного тексту як написання на сенсорному екрані за допомогою стилуса або пальця. Це корисна можливість, оскільки дозволяє користувачеві швидко вписати числа та імена контактів у порівнянні з введенням тієї ж інформації за допомогою екранної клавіатури. Більшості людей комфортніше робити записи рукописним методом і вони можуть зробити це доволі швидко. Ця функція може бути непридатною для використання в більшості смартфонів або планшетів, але доступно багато програм для розпізнавання рукописного вводу.

Оптичний розпізнавання символів (OCR) – це основний метод, який використовується для розпізнавання рукописного тексту. Працює шляхом сканування рукописного документа, а потім перетворення його в текстовий документ або окремо взятого готового зображення. OCR – це форма

розпізнавання зображень, яка призначена для розпізнавання рукописного тексту замість обличчя або форми, наприклад орієнтирів.

Оптичний метод розпізнавання символів (OCR) став цікавою темою протягом багатьох років. Він представляє собою процес оцифровування зображення документа на складові символи. Уявіть систему, що може розшифрувати рецепт лікаря з поганим почерком. Незважаючи на десятиліття інтенсивних досліджень, розробка OCR з можливостями, порівнянними з можливостями людини, все ще залишається відкритим питанням.

Коли ви заповнюєте CAPTCHA, ви знаєте, що це працює за принципом вашої здатності перевершити завдання, яке вважається недоступним для комп'ютерів. Якщо ви вважаєте, що CAPTCHA буде залишатиметься важко розбірливою для комп’ютерів протягом декількох наступних років – це помилка.

За останні кілька років кількість академічних лабораторій та компаній, що займаються дослідженнями розпізнавання символів, різко зросла. OCR є складною проблемою через різноманітність мов, шрифтів і стилів, в яких текст може бути написаний, і не кажучи вже про складні мовні правила.

Отже, методи різних галузей інформатики, тобто обробка зображень, класифікація моделей та обробка природної мови тощо, використовуються для вирішення різних завдань. Перш ніж ми почнемо вирішувати завдання у цій галузі, і чим займається група Хурана, давайте коротко поговоримо про застосування OCR.

OCR дозволяє створювати велику кількість корисних додатків в режимі онлайн і офлайн, таких як:

1. Введення даних для бізнес-документів. Уявіть, що хіміки / аптеки зможуть сканувати рукописи лікарів без будь-яких проблем.

2. Автоматичне розпізнавання номерних знаків. Це легко призведе до покращення дотримання правил дорожнього руху.

3. Автоматичні страхові документи, які можуть витягувати ключову інформацію без втручання людини.

4. Витяг інформації візитної картки в список контактів.

5. Сканування книг.

6. Електронні зображення друкованих документів для пошуку, наприклад, Google Books.

7. Перетворення рукописного тексту в режимі реального часу для керування комп'ютером (ручне обчислення).

8. Допоміжні технології для сліпих та людей з вадами зору. Уявіть, що всі дорожні знаки доступні для сліпого на відстані.

Додатки практично у всіх сферах життя. Достатньо лише просто уявити, де можна зустріти рукописний текст, і це буде можливістю для реалізації.

Збіг шаблонів – це один із найпростіших і найстаріших підходів. У даному підході різноманітні шаблони кожного слова зберігаються для вхідного зображення, обчислюється помилка або різниця з кожним шаблоном, виводиться символ, що відповідає мінімальної похибці. Ця техніка ефективно працює для розпізнавання стандартних шрифтів, але погано працює з написаними вручну символами.

Матрична відповідність передбачає порівняння зображення зі збереженим гліфом на основі пікселів; він також відомий як "відповідність шаблону", "розпізнавання образів" або "співвідношення зображень". Метод оснований на тому, що вхідний гліф правильно ізольований від решти зображення, а в збереженому гліфі знаходиться такий же шрифт і в тому ж масштабі. Ця техніка найкраще працює з машинописним текстом і не працює належним чином з новими шрифтами.

Розпізнавання зображень – це ще один підхід, в якому аналізується статистичний розподіл точок та витягуються ортогональні властивості. Для кожного символу вектор функції розраховується та зберігається в базі даних. Причому розпізнавання виконується шляхом знаходження відстані вектору вхідного зображення до зображення, збереженого в базі даних, і виведення символу з мінімальним відхиленням.

З геометричного підходу, функції залежать від фізичних властивостей, таких як кількість стиків, відносних положень, кількості кінцевих точок,

співвідношення довжини до ширини тощо. Класи, створені на основі цих геометричних ознак, доволі відрізняються від незначного перекриття. Однак головним недоліком такого підходу є те, що цей підхід значною мірою залежить від набору символів, хіндійських алфавітів, латинських алфавітів або цифр тощо. Можливості, використані для одного набору символів, навряд чи працюють для іншого набору. Дані фактори впливають на рівень розпізнавання та багато інших показників продуктивності розпізнавання символів.

1.3 Інструменти і технології машинного навчання

Застосування популярних алгоритмів машинного навчання для великих обсягів даних спричинили нові практичні проблеми. Традиційні бібліотеки машинного навчання не підтримують обчислення великих масивів даних, тому необхідні нові підходи.

Як наслідок, парадигма хмарних обчислень вплинула на сферу машинного навчання. Один з підходів – це використання статистичних інструментів та бібліотек (R, Python), застосованих в хмарі. Інший напрям продуктів – додавання існуючих інструментів з плагінами, які дозволяють користувачам створити кластер Hadoop в хмарі. Далі в списку бібліотеки розподіленої реалізації для алгоритмів машинного навчання, а також на передумові розгортання складних систем для аналізу даних і інтелектуальний аналіз даних. Останній підхід машинного навчання – програмне забезпечення як послуга (Software-as-a-Service), декілька Big Data стартапів (а також великі компанії) вже виводять свої рішення на ринок [4].

Протягом більше двох десятиліть, продукти, які розробляються паралельно з базами даних, таких як Teradata, Oracle або Netezza надали кошти для реалізації алгоритмів машинного навчання в коді SQL, що є складним завданням для підтримки. Крім того, великомасштабні установки цих продуктів є дорогими і не є доступним варіантом у більшості випадків. Інший рушій для зсуву парадигми від реляційної моделі до інших альтернатив – це нова природа

даних. Приблизно п'ять років тому, велика частина даних мала транзакційний характер, що складається з числових або рядкових даних, які легко зберігаються в рядках і стовпцях реляційної бази даних. З тих пір, у той час як структуровані дані мають лінійне зростання, неструктурованих (наприклад, аудіо та відео) і напівструктуровані дані (наприклад, дані веб-трафіку, соціальний медіаконтенту і т. д.) демонструють експоненціальне зростання. Велика частина нових даних або напівструктурованного формату, тобто складається з заголовків, за якими слідують рядки тексту або чисті неструктуровані дані (фото, відео, аудіо). У той час як останні мають обмежений текстовий зміст і є більш складним для синтаксичного аналізу, напівструктуровані дані спричинили появу безліч нереляційних сховищ даних (NoSQL), адаптованих для обробки величезної кількості даних. Як наслідок, за останні 5 років дослідники бачили перехід до паралелізації машинного навчання з використанням цих нових платформ, таких як NoSQL сховищ даних, розподілених середовищах обробки (MapReduce), або хмарних обчислень. Варто згадати гарну метафору Бена Уертера [18], співзасновник Platfora, для обробки Big Data: «Якщо говорити у термінах промислової революції, ми в доіндустріальній епосі ремісництва, що передує масовому виробництву. Це еквівалентно необхідності залучати експерта коваля викувати вилки і ложки для нашого обіднього столу» [5].

Машинне навчання за своєю суттю є трудомістким завданням, тобто багато зусиль було проведено з метою зменшення часу виконання. Парадигма хмарних обчислень і провайдери хмарних технологій виявилися цінними елементами для прискорення платформ машинного навчання. Таким чином, популярні статистичні інструменти середовища, як R, Octave, Python перемістились в хмару також. Існує два основних напрямки для їх інтеграції з постачальниками хмарних технологій: створення кластера в хмарі з самозавантаженням його статистичних інструментів або розширення статистичних середовищ за допомогою плагінів, які дозволяють користувачам створювати кластери Hadoop у хмарі і запустити роботу на них.

Середовища R, Octave, Mapple та аналогічні пропозиції інфраструктур низького рівня для аналізу даних, які можуть бути застосовані для великих наборів даних, вже забезпечують постачальники хмарних технологій. Машинне навчання це те, що стоїть на вершині цього і полегшує витяг корисних знань з великих обсягів даних для клієнтів, які мають слабкі знання у статистиці або взагалі не мають таких знань, тепер вони мають можливість автоматично отримувати так звані моделі знань з великих наборів даних. Протягом останніх 5 років спостерігається вибух стартапів, деякі з них працюють ще в прихованому режимі, які пропонують послуги машинного навчання своїм клієнтам і послуги аналізу Big Data. Ці ініціативи можуть бути PaaS / SaaS платформи або продукти, які можуть бути розгорнуті на приватних середовищах.

Оглянувши літературу і ринок, можна зробити висновок, що машинне навчання має різноманітні прояви. Можна класифікувати 5 основних підходів:

1. Середовище машинного навчання в хмарі (створення комп'ютерного кластеру в хмарі та його розгортання з інструментами статистики).

2. Модулі для інструментів машинного навчання – доповнення статистичних інструментів за допомогою плагінів, які дозволяють користувачам створювати кластер Hadoop в хмарі і виконувати завдання машинного навчання в ньому.

3. Розподілені бібліотеки машинного навчання – колекції параллелізованних реалізацій алгоритмів машинного навчання для розподілених середовищ (Hadoop, Dryad і т. д.).

4. Комплексні системи машинного навчання – продукти, які повинні бути встановлені в приватних центрах обробки даних (або в хмарі) і пропонують інтелектуальний аналіз даних високої продуктивності).

5. Постачальники програмного забезпечення як послуг для машинного навчання - PaaS / SaaS рішень, які дозволяють клієнтам отримати доступ до алгоритмів машинного навчання через веб-сервіси).

1.4 Висновки

Аналіз предметної галузі показав що розпізнавання рукописного тексту є важливою і актуальною задачею, оскільки є багато областей де потрібно розпізнавати рукописний текст, зокрема у промисловості, науці, техніці, банківській сфері, роботі поштових компаній тощо.

В той час як задача розпізнавання друкованих текстів в основному вирішена, розпізнавання рукописних текстів вимагає додаткових досліджень, оскільки якість та точність розпізнавання найбільше всього залежить від проведеної попередньої обробки зображення.

2 РОЗРОБКА МЕТОДІВ ОБРОБКИ ТА РОЗПІЗНАВАННЯ ТЕКСТУ

У методології машинного навчання важливим кроком є побудова математичних моделей. Ці моделі можуть бути отримані методами лінійної регресії, логістичної регресії, дискримінантного аналізу, дерев рішень, нейронних мереж та ін. Однак згорткова нейронна мережа найбільш часто використовується на практиці для аналізу зображень.

2.1 Розробка методу первинної обробки зображень

Документальне зображення бінаризації являється сегментацією документа на передній план, тобто текст, та фон. Це зроблено для отримання чітких зображень, з яких можна легко отримати текст. Порогове значення використовується для сегментації зображень документа.

Бінаризація – це етап попередньої обробки для аналізу та обробки зображення документів, він покращує ефективність технологій обробки документів, таких як OCR та аналіз макета. Image Binarization – це перетворення зображення на подвійне зображення документа. Пікселі розділені на подвійний набір пікселів, тобто чорно-білий. Основною метою бінаризації зображення є сегментація документа на передній план тексту та фон.

Найпростіший підхід до бінаризації – пороговий. При пороговому значенні вибирається оптимальне порогове значення, і пікселі класифікуються як передній план або фон у порівнянні з цим пороговим значенням. Але в на практиці деякі документи страждають від деградації, такої як нерівномірне висвітлення, фоновий шум, забруднення та різноманітність освітленості і контрасту. Вибір оптимального порогового значення для таких документів є складним завданням [1]. Неправильна оцінка порогового значення призводить до неправильної класифікації пікселів на передньому плані або фоні [2]. Це впливає на результати бінаризації та точності прикладів розпізнавання образів.

Як правило, бінарізація класифікується як глобальна та локальна [3]. Глобальна бінаризація – це техніка, в якій застосовується єдине порогове значення для подвійного зображення всього зображення. Це швидкий процес, але не працює для документів зі складним фоном. В методі локальної бінаризації замість одиничного порогу на ціле зображення, для кожного пікселя вибирається різне значення порогу. Поріг вибирається в залежності від сусідніх пікселів. Локальні пороги не дають хороших результатів для документів, що страждають від фонового шуму.

Методи бінаризації також можна класифікувати на основі критеріїв для вибору порогового значення. Деякі з методів вибору порога базуються на гістограмі, кластеризації, ентропії та локальних адаптивних методах.

Метод Оцу – це глобальний пороговий метод, який перетворює чорно-біле зображення на подвійний рівень зображення. Цей метод розбиває пікселі на два класи: один – передній план, а інший – фон. Він вибирає оптимальний поріг, який розділяє зображення на два різних класу. Порогова величина вибирається таким чином, що дисперсія всередині класу зводиться до мінімуму та дисперсія між класами максимізується. Метод Оцу дає найкращі результати лише для тих зображень, які мають чіткий бімодальний шаблон. Але деградовані документи, як правило, не мають такої чіткої схеми. Крім того, він не працює добре для зображень з нерівномірним освітленням та тінями.

Логістична – популярна модель, яка використовується в аналізі виживання (англ. survival analysis), яка може бути використана для оцінки значущості різних коваріат на тривалість життя людей або об'єктів за допомогою функції ризику. Крім того, може бути описаний кількісний вплив цих змінних протягом життя на важливі змінні результату (наприклад, медіани виживання).

Логістична регресія (англ. logistic regression) або логіт регресія (англ. logit model) — статистичний регресійний метод, що використовується у випадку коли залежна змінна може набувати тільки двох значень (чи, більш загально, скінченну множину значень).

Логістична регресія – окремий випадок узагальненої лінійної регресії. Припускається, що залежна змінна приймає два значення і має біноміальний розподіл:

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_k x_k \quad (2.1)$$

Чи можна її використовувати для завдання оцінки ймовірності результату події? Так, можна, обчисливши стандартні коефіцієнти регресії. Наприклад, якщо розглядається результат по позиці, задається змінна у зі значеннями 1 і 0, де 1 означає, що відповідний позичальник розплатився за кредитом, а 0, що мав місце дефолт. Однак тут виникає проблема: множинна регресія не «знає», що змінна відгуку бінарна за своєю природою. Це неминуче призведе до моделі з пророкованими значеннями більше 1 і менше 0. Такі значення взагалі не припустимі для початкової задачі. Таким чином, множинна регресія просто ігнорує обмеження на діапазон значень для у.

Для вирішення проблеми завдання регресії може бути сформульоване інакше: замість прогнозування бінарної змінної, ми передбачаємо безперервну змінну зі значеннями на відрізку [0,1] при будь-яких значеннях незалежних змінних. Це досягається застосуванням наступного регресійного рівняння (логітеквіваленті):

$$f(z) = \frac{1}{1 + e^{-z}} \quad (2.2)$$

Зробимо деякі перетворення:

$$z = ln\frac{f(z)}{1-f(z)} = ln\frac{p}{1-p} = logit(p) \quad (2.3)$$

Відзначимо, що якщо x_i має вільний член, з’явиться проблема ідентифікованості – неможливо обчислити λ і β_0. З ряду причин (зручність, простота, чисельна стійкість, точність наближених процедур виведення), переважно краще оцінювати β_0 ніж λ, тому будемо використовувати цю параметризацію. Звичайно, отримавши оцінку β_0, можна легко оцінити значення та довірчі інтервали для λ через перетворення $\lambda = \exp(\beta_0)$.

Оцінка β за методом максимальної правдоподібності є складною у випадку експоненційної регресії через те, що потрібно розв'язувати нелінійну систему рівнянь. Воно не може бути розв'язане у явній формі, для розв'язання потрібна ітеративна процедура.

Основною ідеєю є створити лінійне наближення до нелінійної системи рівнянь, обчислити з неї β, обчислити нове наближення і повторювати до збіжності.

Як і з іншими регресійними моделями, інтерпретація коефіцієнтів регресії стосується ефекту від зміни одного з факторів за умови, що інші лишаються незмінними.

Розглянемо гіпотетичне порівняння між двома індивідами, чиї пре диктори є однаковими окрім змінної j, де вони відрізняються на $\delta_j = x_{1j} - x_{2j}$. Тоді:

Для будь-якої моделі пропорційних ризиків, $\lambda_1(t)/\lambda_2(t)$ є константою, незмінною з часом. Ця константа (2.18) відома як відношення ризиків, або відносний ризик. Таким чином, інтерпретацією коефіцієнта регресії в моделі пропорційних ризиків є те, що $e^{\delta\beta}$ є відношення ризиків для зміни в δодиниць в коваріаті.

Позначимо лінійний предиктор $\eta_i = x_i^T\beta$.

Враховуючи незалежне цензурування і вважаючи, що реальний час життя $T_i|x_i \sim Exp(\lambda_i)$, внесок логарифма правдоподібності і-го суб'єкта в експоненційну регресію наступний [3]:

$$l_i(\eta_i) = d_i\eta_i - t_i e^{\eta_i}, \tag{2.3}$$

де

$d_i = 1\{\tilde{t}_i \le c_i\}$, $t_i = \tilde{t}_i \wedge c_i = \min(\tilde{t}_i, c_i)$, c_i – час цензурування і-го суб'єкта,

$\tilde{t}i$ – реальний час життя і-го суб'єкта.

Тоді похідна логарифму правдоподібності і гессіан мають наступний вигляд:

$$u_i(\eta_i) = d_i - t_i e^{\eta_i} \tag{2.4}$$

$$H_i(\eta_i) = -t_i e^{\eta_i} \tag{2.5}$$

Нехай μ позначає вектор, у якого і-й елемент дорівнює $t_i e^{\eta_i}$ і W позначає діагональну матрицю з і-м діагональним елементом $t_i e^{\eta_i}$. Тоді можна переписати похідну логарифму правдоподібності і гессіан у вигляді:

$$u(\eta) = d - \mu \tag{2.6}$$

$$H(\eta) = -W \tag{2.7}$$

Як було відзначено раніше, вирішення для μ = 0 є складним, оскільки μ є нелінійною функцією від η. Тому, розглянемо наближення у вигляді ряду Тейлора

$$u(\eta) \approx u(\tilde{\eta}) + H(\tilde{\eta})(\eta - \tilde{\eta}) = d - \mu + W(\tilde{\eta} - \eta) \tag{2.8}$$

де μ і W фіксовані на $\tilde{\eta}$.

Підставляючи $\eta = X\beta$ у попереднє рівняння і розв'язуючи для β, отримуємо

$$\hat{\beta} \leftarrow (X^T W X)^{-1} X^T (d - \mu) + \tilde{\beta} \tag{2.9}$$

Так як маємо ітеративну процедуру, ми не отримуємо точне значення β. Натомість, потрібно обчислити значення β, перерахувати μ і W і повторювати до збіжності.

Алгоритм Ньютона-Рафсона збігається до оцінки максимальної правдоподібності (хоча це не гарантовано), так як правдоподібність

логарифмічно увігнута і зростаюча, і це також вірно (зазвичай) для експоненційної регресії.

Алгоритм у вигляді блок-схеми зображений на рисунку 2.1.

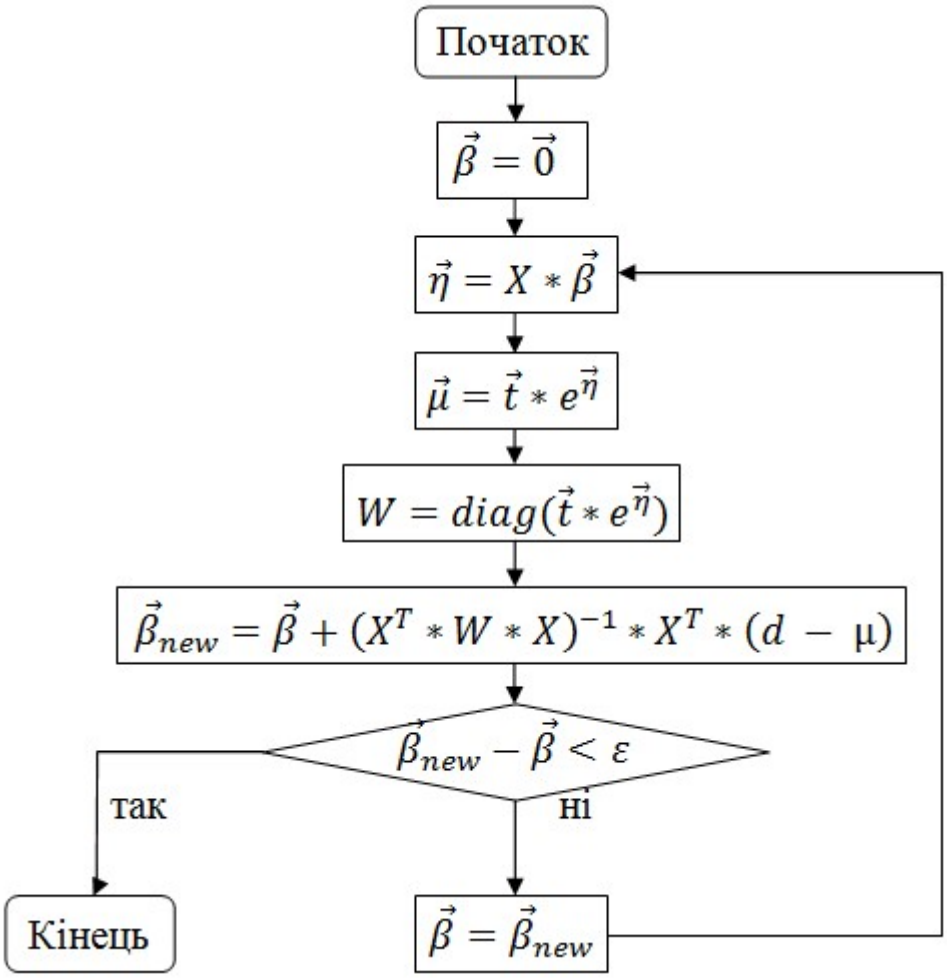


Рисунок 2.1 – Блок-схема алгоритму Ньютона-Рафсона

Існує три підходи до отримання довірчих інтервалів параметрів моделей виживання: заснований на похідній логарифму правдоподібності, метод Вальда і метод відношення правдоподібностей [3].

У методі, заснованому на похідній логарифму правдоподібності, перевіряється гіпотеза $H_0 : \theta = \theta_0$. Для цього обчислюється статистика

$$\frac{U(\theta_0)}{\sqrt{I(\theta_0)}} \tag{2.10}$$

і її значення порівнюється з нормальним розподілом. Як і звичайно, перетворюючи цей тест до $\alpha = 0.05$, можна отримати 95% довірчі інтервали для θ. Відмітимо, що для цього підходу, на відміну від двох наступних, не потрібно оцінювати значення θ.

Метод Вальда заснований на наближенні до похідної логарифму правдоподібності ряду Тейлора оцінки максимальної правдоподібності (H – матриця Гессе):

$$u(\theta) \approx H(\hat{\theta})(\theta - \hat{\theta}) \tag{2.11}$$

Таким чином,

$$\mathcal{I}^{\frac{1}{2}}\left(\hat{\theta} - \theta_0\right) \sim N(0,1), \text{ або } \hat{\theta} \sim N(\theta_0, \mathcal{I}^{-1}) \tag{2.12}$$

Тоді оцінка максимальної правдоподібності наближено нормальна, має середнє значення рівне справжньому значенню параметра і варіацію рівну оберненому значенню від інформації.

На основі цього результату, можна легко отримати тести або довірчі інтервали для θ.

Також, розглянемо асимптотичний розподіл відношення правдоподібностей. Цей підхід також застосовує розклад в ряд Тейлора, але тут наближується логарифм правдоподібності.

$$l(\theta) \approx l\left(\tilde{\theta}\right) + \frac{1}{2}\left(\theta - \hat{\theta}\right)^T H(\hat{\theta})(\theta - \hat{\theta}) \tag{2.13}$$

Таким чином,

$$2\left\{l\left(\hat{\theta}\right) - l(\theta_0)\right\} \sim \chi_p^2 \tag{2.14}$$

Отримана за алгоритмом Ньютона-Рафсона оцінка $\hat{\beta}$ є оцінкою максимальної правдоподібності, за методом Вальда $\hat{\beta} \sim N(\beta, I^{-1})$. Залишилось лише отримати матрицю інформації відносно β. Продовжуючи попередні міркування, маємо:

$$\hat{\beta} \sim N(\beta, (X^T W X)^{-1}) \tag{2.15}$$

Звідси отримуємо довірчі інтервали для β_j:

$$\beta_j \pm z_{1-\frac{\alpha}{2}} SE_j \tag{2.16}$$

$$SE_j = \sqrt{(X^T W X)^{-1}_{jj}} \tag{2.17}$$

Підхід відношення правдоподібностей на практиці дещо ускладнюється у випадку декількох параметрів, оскільки ми не маємо оцінок параметрів у явних формах. Якби деякий параметр β_j був єдиним параметром, проблема би зводилась до знаходження коренів, де ми б обчислили довірчий інтервал за правилом $2\left(l(\hat{\beta}_j) - l(\beta_j)\right) = \chi^2_{1,.95}$.

Проте, β_j не є єдиним параметром, тому всі оцінки максимальної правдоподібності відповідно зміняться. Іншими словами, оцінювання $l(\beta_j)$ не є простим, тому що потрібно перераховувати $\hat{\beta}_{-j}$ для кожного значення β_j для якого ми намагаємось знайти корені.

Правдоподібність $L(\beta_j, \hat{\beta}_{-j}(\beta_j))$ називається профілюючою правдоподібністю, а процедура перерахування називається профілюванням.

Отримання довірчих інтервалів за допомогою похідної логарифму правдоподібності або відношення правдоподібностей вимагають профілювання, на відміну від підходу Вальда. На практиці, швидше і зручніше використовувати довірчі інтервали Вальда [3]. У бібліотеці survival на мові програмування R застосовуються саме вони.

Для випадкової величини часу до події T, модель прискорення часу відмови [3] пропонує наступне відношення між коваріатами та $Y = \log T$:

$$Y_i = x_i^T \beta + W_i, \tag{2.19}$$

де $W_i \sim^{iid} f$ є похибкою (або залишком). Такі моделі часто називаються лог-лінійними моделями. Вищенаведені позначення описують широкий клас моделей: в залежності від розподілу, який буде вказано для W, будуть отримані різні моделі, але всі вони будуть мати схожу загальну структуру.

Очевидно, можна припустити, що $W_i \sim^{iid} N(0, \sigma^2)$. Припущення про те, що Y має нормальний розподіл, еквівалентне припущенню, що T має логнормальний розподіл. Тоді, за відсутності цензурування, можна просто використати звичайний метод найменших квадратів, щоб побудувати модель, отримати довірчі інтервали, тощо. Але звичайно, майже завжди у дослідженнях присутнє цензурування, тому потрібно розширити звичайні лінійні методи так, щоб вони справлялись з цензуруванням. Більше того, логнормальний розподіл, незважаючи на свою зручність, не описує точно більшість розподілів часу до події.

Для будь-якої моделі прискорення часу відмови, маємо:

$$T = e^{\eta_i} T_0, \tag{2.20}$$

де $T_0 = e^W$ і $\eta_i = x_i^T \beta$.

Іншими словами, в той час як в моделі пропорційних ризиків коваріати впливають мультиплікативно на ризик, в моделі прискорення часу відмови коваріати впливають мультиплікативно на час до виникнення події.

Функція виживання: $S_i(t) = S_0(e^{-\eta_i} t)$.

Функція ризику: $\lambda_i(t) = \lambda_0(e^{-\eta_i} t) e^{-\eta_i}$.

Якщо порівнювати модель пропорційних ризиків і модель прискорення часу відмови на графіку логарифм часу проти логарифму ризику, ефект припущення пропорційних ризиків впливає на зміну ризику і спричиняє вертикальне зміщення, в той час як ефект моделі прискорення часу відмови спричиняє горизонтальний зсув. В загальному, дві моделі не можуть бути приведені одна до одної, тому процес може бути краще описаний лише однією з моделей, а не обома одночасно [3].

Проте, існує виключення: якщо розподіл є лінійним (на шкалі логарифму часу проти логарифму ризику), тоді будь-якому горизонтальному зміщенню лінії можна поставити у відповідність вертикальне. У цьому масштабі є лінійним розподіл екстремальних значень, а також сімейство розподілів Вейбулла. Тому, розподіл Вейбулла є єдиним розподілом, який задовольняє обидві моделі: пропорційних ризиків і прискорення часу відмови [3].

Згідно з вищенаведеним, вплив зміни на δ_j одиниць в коваріаті j призводить до збільшення часу відмови у $\exp(\delta_i \beta_i)$ разів.

2.2 Сегментація ліній тексту

Одним з найпоширеніших підходів до сегментації видобутку рядків у друкованих документах є метод проектування. Він також був адаптований для рукописних документів з невеликим перекриттям [1]. Цей простий спосіб підраховує кількість чорних пікселів у кожному рядку зображення. Ті лінії, що містять кількість пікселів нижче певного значення, вважаються лініями без тексту, а інші – як текстові. Відхиленням від технології проекційного профілю є згладжений проекційний профіль [8], який використовує фільтр низьких частот для зменшення шуму. Існує інший цікавий метод, який застосовується до багатокожих рукописних документів для вилучення текстових ліній компонентів. Цей алгоритм передбачає, що гіпотетична вода тече з лівого та правого боків зображення, а символи текстових ліній працюють як дамба, яка блокує воду. Отримані ділянки, що залишилися незміцнені на зображенні, нарешті позначаються як кандидати на текстові рядки регіонів. Однак цей спосіб потребує використання структурованого елемента, який визначає кут нахилу для затоплення, який залежить від лінійних схилів кожного конкретного документа. Наприклад застосування цього методу до зразка рукописного документа. Оцінка відповідного значення параметра (кут нахилу необхідного структурованого елемента) для кожного документа є складною і визначає обчислювальні показники роботи методу. Зразок двоетапного документа та (справа) сегментація лінії гіпотетичним потоком води. Сірими областями є затоплені частини документа. Створення бінарної карти переходу зображення для пошуку імовірного розташування текстових ліній застосовується в [2]. Пізніше виявлені лінії очищаються за допомогою алгоритму розрізання графіків з мінімальним розрахунком / максимальним потоком. Інший підхід використовує перетворення Hough на основі блоків для виявлення та вилучення необмеженої лінії [10].

Ніколас та інші запропонували метод сегментації ліній у документах, який базується на структурі вирішення проблем AI виробничих систем. Лі та інші оцінюють карту вірогідності зображення документа, де кожен компонент представляє ймовірність того, що розглянутий піксель належить до текстової лінії. Вони розвивають початковий виявлення тексту та використовують метод встановлення рівня, щоб визначити межу сусідніх текстових ліній. Їхні експерименти показали, що запропонований алгоритм є надійним для масштабування зміни, обертання та шуму. У недавньому документі використовується запропонований метод вилучення текстових рядків на основі морфології для виявлення текстових областей у зображеннях. Для обробки з перекосом рукописних текстових рядків автори застосовують метод на основі моментів для оцінки лінійних орієнтацій. Лікформан-Сулем та інші опублікував у 2007 році оглядовий документ про текстові лінії (друковані та рукописні), сегментацію. історичних зображень документа. Ця робота детально описує складність цієї проблеми, а також описує класифікацію методів сегментації.

Процес вилучення рядків із документа використовується як основа для вилучення структури документів, розпізнавання рукописного тексту або вдосконалення тексту. Існує безліч методів, які стосуються проблеми вилучення рядка друкованого документа, яка, як правило, скорочується до глобального пошуку косусів (текстові рядки паралельні один одному, але не обов'язково горизонтальні). З іншого боку, при роботі з рукописним документом проблема стає більш складною лінії не паралельні один одному, однакові літери не мають однакових розмірів, текстові лінії мають літери, які поширюються на інші текстові лінії, висока текстова організація не може бути визначена (абзаци, підрозділи тощо). У будь-якому зображенні, документі чи ні, з рукописним або друкованим текстом кожен піксель може бути пов'язаний з важливістю (тобто, кількість пікселів впливає на загальне зображення).

2.3 Опрацювання шуму на зображеннях

Зображення часто деградуються шумами. Шум може виникнути під час захоплення зображень, передачі тощо. Вилучення шумів є важливим завданням при обробці зображень. Загалом, результати шумозаглушення сильно впливають на якість обробки зображень. Деякі способи видалення шумів добре зарекомендували себе при обробці кольорових зображень. Характер проблеми видалення шумів залежить від типу шуму, що руйнує зображення. У сфері зменшення шумів зображення було запропоновано декілька методів лінійного та нелінійного фільтрування. Лінійні фільтри не здатні ефективно усунути імпульсний шум, оскільки вони мають тенденцію до розмивання країв зображення. З іншого боку, нелінійні фільтри підходять для боротьби з імпульсним шумом. За останні кілька років виникли кілька нелінійних фільтрів на основі класичної та нечіткої техніки. Наприклад, більшість класичних фільтрів одночасно видаляють розмивання країв, а нечіткі фільтри мають можливість поєднувати збереження та згладжування краю. У порівнянні з іншими нелінійними методами нечіткі фільтри здатні представляти знання зрозумілим способом.

У комп'ютерному природному сценічному аналізі широко використовуються кольорові зображення завдяки кращим характеристикам над чорно-білими зображеннями. Розглянемо, наприклад, кілька сусідніх об'єктів з різним кольором, але однакову світність. Ці об'єкти можуть бути інтерпретовані як об'єднані об'єкти на графічному рівні зображення через постійні інтенсивності, але можуть бути відбиті яскраво в кольоровому зображенні. Додана інформація про кольори висвітлює будь-яку можливу двозначність. Окрім переносної інформації про відображену енергію, кольорові зображення містять спектральну інформацію про об'єкт та надають важливі деталі для швидкого візуального пошуку, перевірки та точної класифікації об'єктів. Більше того, колір поверхні відносно інваріантний для (інтенсивності) освітленості, що відбувається через властивість кольорової сталості. Тому обробка кольорової

інформації є відносно послідовною, коли умови освітлення навколишнього середовища змінюються, і забезпечує більш надійні та точні результати для сприйняття машини та аналізу природного місця.

Будь-яке зображення, отримане оптичним, електрооптичним чи електронним способом, може бути деградоване недосконалістю механізмів зондування. Потенційні деградації можуть мати місце у вигляді сенсора, шумів фотографічного зерна, розмивання (незфокусована камера, рух відносної об'єктної камери), випадкова атмосферна турбулентність тощо. Наявність шуму в зображеннях являє собою незворотну втрату інформації.

Шум є результатом помилок у процесі отримання зображень, що призводить до значень пікселів, які не відповідають дійсній інтенсивності справжньої сцени. Зменшення шуму – це процес видалення шуму від сигналу. Методи зменшення шуму є концептуально дуже схожі, незалежно від оброблюваного сигналу, однак знання характеристик очікуваного сигналу може означати, що реалізація цих прийомів сильно відрізняється залежно від типу сигналу. Зображення, зроблене датчиком, піддається фільтруванню різними фільтрами згладжування та отриманими зображеннями. Усі пристрої запису, як аналогові, так і цифрові, мають риси, які роблять їх чутливими до шуму. Основна проблема обробки зображень полягає в зменшенні шуму цифрового кольорового зображення.

Удосконалення якості зображення було проблемою у всій області обробки зображень. Зображення страждають від різних типів шумів. Шум зображення є небажаним, оскільки він погіршує якість зображення.

Методи зменшення шуму в обробці зображень є перспективним напрямом досліджень. Нечіткі технології вже застосовуються в декількох областях обробки зображень і мають численні практичні застосування. Звуковий шум, як правило, небажаний, різниця в яскравості або інформація про кольори розглядається як шум. Шум зображення може породжуватись у зерні плівок або в електронних шумах у датчику та схемі вхідного пристрою або в неминучому короткому шумі ідеального фотонного детектора. Шум зображення найбільш яскраво виражено

в області зображень з низьким рівнем сигналу, такими як область тіні або під виглядом зображень.

Алгоритми вилучення зображень часто приймають додатковий білий гаусовий шум (AWGN), який не залежить від фактичних значень RGB. Такі підходи не є повністю автоматичними і не можуть ефективно видалити колірний шум, вироблений сучасною цифровою камерою ПЗЗ. У даному документі пропонується єдина схема для двох завдань: автоматичну оцінку та видалення кольорового шуму з одного зображення за допомогою частково гладких моделей зображень. Ми вводимо функцію рівня шуму (NLF), яка є безперервною функцією, яка описує рівень шуму як функцію яскравості зображення. Потім оцінюємо верхню межу реальної функції шуму, встановивши нижній поріг відрази до стандартних відхилень відмінностей зображення на одному сегменті. Для видалення кольору кольоровий шум значно зменшується шляхом проеціювання значень пікселів на лінію, яка відповідає значенням RGB у кожному сегменті. Потім побудовано гаусівське умовне довільне поле (GCRF) для отримання нижчого чистого зображення із шумного вводу. Величезні експерименти проводяться для тестування пропонованого алгоритму, який, як показують, перевершує сучасні алгоритми стирання.

Розпізнавання зображень вивчалося десятиліттями в галузі комп'ютерного зору, обробки зображень та статистичної обробки сигналів. Ця проблема не тільки забезпечує хорошу платформу для вивчення природних моделей зображень та алгоритмів розподілу сигналів, але також стає важливою частиною систем придбання цифрових зображень для підвищення якості зображень. Ці два напрямки є важливими і будуть розглянуті в цьому документі.

Велика частина існуючої роботи з видалення зображення передбачає додавання білого гаусового шуму (AWGN) і видаляє шум, незалежний від каналів RGB. Однак тип і рівень шуму, який створюють цифрові камери, невідомі, якщо серія та марка камери, а також параметри камери (ISO, витримка затвора, діафрагма та вмикання / вимикання) невідомі. Наприклад, метадані формату файлів зображень (EXIF), додані до кожного зображення, можуть бути

втрачені при перетворенні формату зображення та переміщенні файлу зображення. Тим часом, статистика колірного шуму не залежить від каналів RGB через процес, вбудований в камери. Отже, поточні підходи, що використовуються для відміни, не є справді автоматичними і не можуть ефективно видаляти кольоровий шум. Цей факт перешкоджає практичному застосуванню методів видалення шуму до розпізнавання цифрових зображень та їх поширення.

У деякому програмному забезпеченні, що оброблює зображення, користувач повинен визначити кількість рівних зображень регіонів для оцінки рівня шуму. Це змусило нас прийняти сегментаційний підхід для автоматичного визначення рівня шуму з одного зображення. Оскільки рівень шуму залежить від яскравості зображення, ми пропонуємо оцінити верхню межу функції рівня шуму (NLF) з зображення. Зображення поділяється на кусочно-гладкі області, в яких середнє значення має оцінка яскравості, а стандартне відхилення - переоцінка рівня шуму. Попередні функції рівня шумів вивчаються шляхом імітації процесу цифрової обробки зображень, і вони використовуються для правильної оцінки кривої, коли відсутні дані.

Оскільки відокремлення сигналу та шуму від одного входу є недостатньо обмеженим, теоретично неможливо повністю відновити оригінальне зображення від шумового забрудненого. Метою вилучення зображень є якнайбільше збереження точності зображення при усуненні шуму.

2.3 Розробка комплексного методу розпізнавання з використанням ковзного вікна і згорткової нейронної мережі

Згорткові нейронні мережі мають іншу архітектуру, ніж звичайні нейронні мережі. Регулярні нейронні мережі перетворюють вхід, виводячи його через серію прихованих шарів. Кожен шар складається з набору нейронів, де кожен шар повністю з'єднаний з усіма нейронами в шарі раніше. Нарешті, є останній повний з'єднаний шар - вихідний шар, який представляє прогнози.

Згорткові нейронні мережі самі по собі бувають дещо різні. Перш за все, шари організовані в 3 вимірах: ширина, висота і глибина. Крім того, нейрони в одному шарі не з'єднуються з усіма нейронами в наступному шарі, але тільки в невеликій його області. Нарешті, кінцевий результат буде зведено до єдиного вектору оцінок імовірності, організованого вздовж вимірювання глибини.

CNNs мають два компоненти:

- Компонент "Приховані шари / вибір ознак";

У цій частині мережа виконуватиме ряд згортки і операцій об'єднання, під час яких виявляються функції. Якщо у вас є зображення зебри, це частина, в якій мережа визнає свої смуги, дві вуха і чотири ноги.

- Класифікаційний компонент;

Тут цілком зв'язані шари слугуватимуть класифікатором поверх цих отриманих функцій. Вони будуть призначати ймовірність того, що об'єкт на зображенні є тим, що передбачає його алгоритм.

CNN особливо корисні для класифікації та розпізнавання зображень. Вони мають дві основні частини: частину вилучення функції та класифікаційну частину.

Основною спеціальною технікою CNN є згортка, де фільтр слайд над входом і об'єднує вхідне значення та значення фільтра на карті об'єктів. Врешті-решт, нашою метою є подача нових зображень на нашу CNN, щоб вона могла дати вірогідність об'єкту, який вважає, що він бачить, або описує зображення з текстом.

CNNs мають широке застосування в області розпізнавання зображень та відео, систем рекомендацій та обробки природної мови. CNNs, як і нейронні мережі, складаються з нейронів з навчальними вагами та упередженнями. Кожен нейрон отримує кілька входів, приймає зважену суму за них, передає його через функцію активації і реагує на вихід. Вся мережа має функцію втрат, і всі поради та хитрощі, які ми розробили для нейронних мереж, все ще застосовуються на CNN.

Глибока згорткова нейронна мережа призвела до результатів прориву в численних практичних завданнях з машинного навчання, таких як класифікація зображень у наборі даних ImageNet, керування політикою навчання в іграх PlayAtari або настільну гру Go і підписи до зображення. Багато хто з цих додатків спочатку виконують функцію extraction, а потім подають їх результати в trainable classifier.

2.4 Висновки

Були розглянуті сучасні методи обробки зображень, методи побудови математичних моделей машинного навчання, та обрано для реалізації метод згорткової нейронної мережі оскільки вона має кращу архітектуру за звичайні нейронні мережі виражену у 3х шарах нейронів та способу їх взаємодії між собою.

Після огляду існуючих кроків обробки і аналізу зображень розроблений алгоритм, який включає 8 кроків: бінаризацію вихідного зображення, виявлення області тексту, сегментації текстових рядків, реконструкції символів та обробки шумів, модель класифікатора символів, збагачення даних та сегментацію символів на основі класифікатора.

3 РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ СИСТЕМИ ОБРОБКИ РУКОПИСНОГО ТЕКСТУ

3.1 Бінаризація вихідного зображення

Що стосується рукописних історичних документів, то етап попередньої обробки має важливе значення. Бінаризація є загальноприйнятою практикою для використання у випадку історичного набору даних [2]. Мета бінаризації полягає в тому, щоб отримати зображення, яке має лише дві можливі значення: чорний або білий для кожного пікселя.

У літературі методи бінаризації поділяються на глобальні або локальні (адаптивні) [3]. Глобальні методи використовують одиничне порогове значення для відокремлення чорно-білих пікселів, тоді як декілька значень використовуються для визначення локального порогу на основі локальної області.

Обидва підходи були перевірені на двозначний пошук образів рукописів. Місцеві методи гаусівського адаптивного порогового та адаптивного середнього порогу показують незрозумілі результати, як показано на рисунках 3.1 і рис 3.2. Як було зазначено в літературі, ці методи, як локальні методи визначення порогу, є добрими у випадку різноманітних фонових компонентів, різного роду колір, текстура або яскравість зображення, видимість тексту [4]. Проте в нашому випадку це не вдається, тому що на зображенні присутні шуми, але яскравість зображення та якість текстури навпаки – досить хороші.

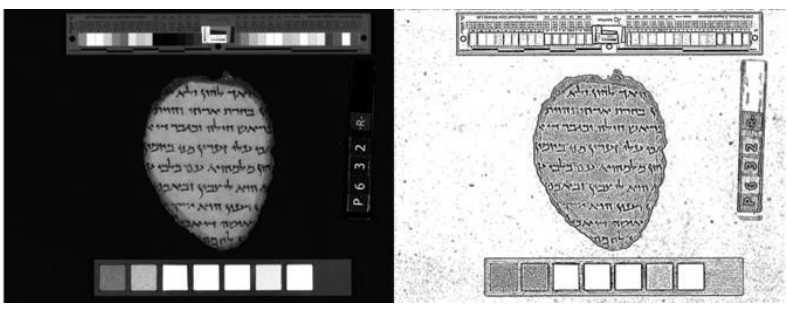


Рисунок 3.1 – Адаптивна бінаризація середнього порогу. Вихідне зображення ліворуч, бінаризоване – праворуч

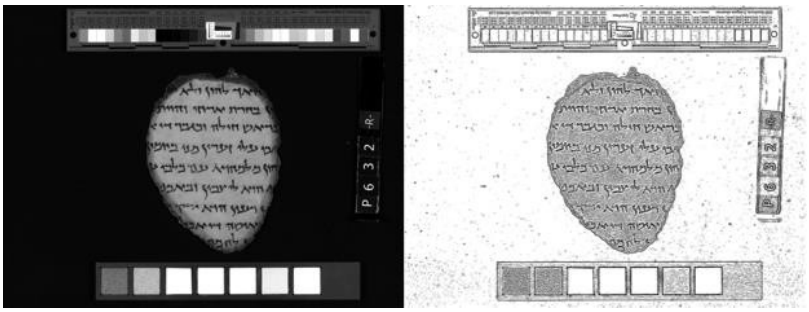


Рисунок 3.2 – Гаусівська адаптивна бінаризація. Вихідне зображення ліворуч, бінаризоване – праворуч

Серед усіх методів бінаризації підхід OTSU [5] найкраще застосовується для шумних рукописів, оскільки він вибирає порогове значення, щоб мінімізувати дисперсію між класами порогових чорно-білих пікселів. Результат бінаризації OTSU показаний на рисинку 3.3.

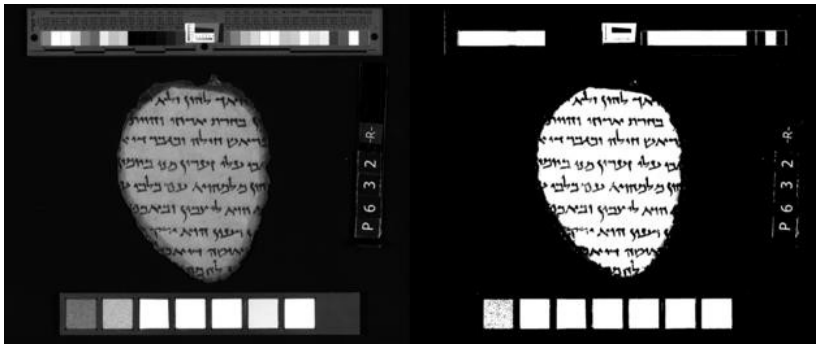


Рисунок 3.3 – Бінаризація OTSU. Вихідне зображення ліворуч, бінаризоване – праворуч.

3.2 Знаходження та попередня обробка текстової області

Після бінаризації, наступний крок полягає в обрізанні області, де розташований текст. Оскільки зображення складаються з додаткової інформації, такої як код та кольорова палітра, цей крок знаходить частину зображення, де знаходиться текст. Було проведено тестування двох методів для отримання текстової області:

1) Пов'язані компоненти;

Текстова область обрізана як найбільший з'єднаний компонент, як показано на рисунку 3.4. Метод має несправності у випадку перекриття символів або коли частина паперу не пов'язана з зображенням.

2) Контури;

Текстова ділянка обрізана як найбільший контур, як показано на рисунку 3.5. В основному цей метод знаходить криву, що з'єднує всі безперервні точки вздовж кордону. Як можна помітити, контурний метод дає кращі результати.

Це вирішує проблеми, що з'явилися з попереднім методом.

Рисунок 3.4 – Обрізаний текст за допомогою методу з’єднаних компонент

Причиною невдалого підходу підключених компонентів є відсутність незв'язаних частин рукопису, а також деякі збіги.

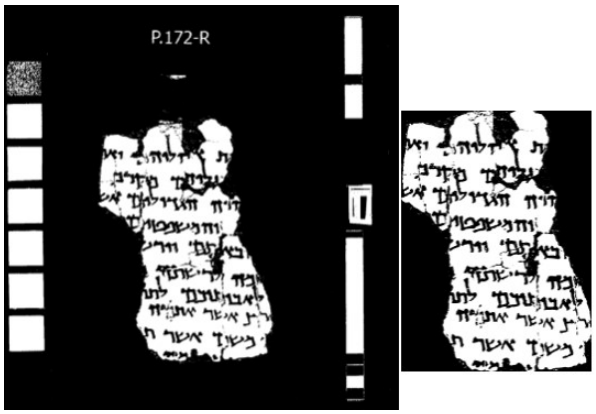


Рисунок 3.5 – Обрізаний текст за допомогою методу контурів

Отже, для основного алгоритму обрана методика контурів.

Під час цього кроку коректування фону здійснюється шляхом зміни кольору фону від чорного до білого. Фон вважається пікселями за межами знайденого контуру.

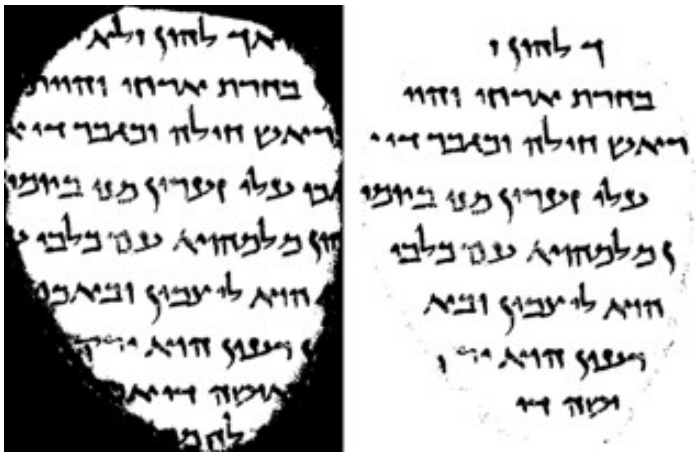


Рисунок 3.6 – Обрізане зображення як найбільший контур ліворуч, це ж зображення з налагодженим фоном справа

3.3 Сегментація ліній тексту

У літературі, часто трактується, що сегментація текстових рядків ґрунтується на проекції гістограми [6], [7].

Виходом з попереднього кроку є зображення з чорними літерами та білим фоном. Прогнозування гістограми отримується підсумовуванням значень пікселів уздовж горизонтальної осі для кожного значення у.

На рисунку 7 ясно, що кожна текстова лінія відповідає піку в гістограмі. Пусті простори між піками представляють можливі області між різними текстовими рядками.

Проте іноді граф коливається, тому більше ніж один пік на рядок. Для вирішення цієї проблеми задаються порогові та мінімальні відстані. Таким чином, буде виявлено лише піки з амплітудою, більшою за поріг, і мінімальна відстань між кожним виявленим піком. Як тільки отримано піки, обчислюється середня висота лінії для пошуку координат сегментованих ліній.

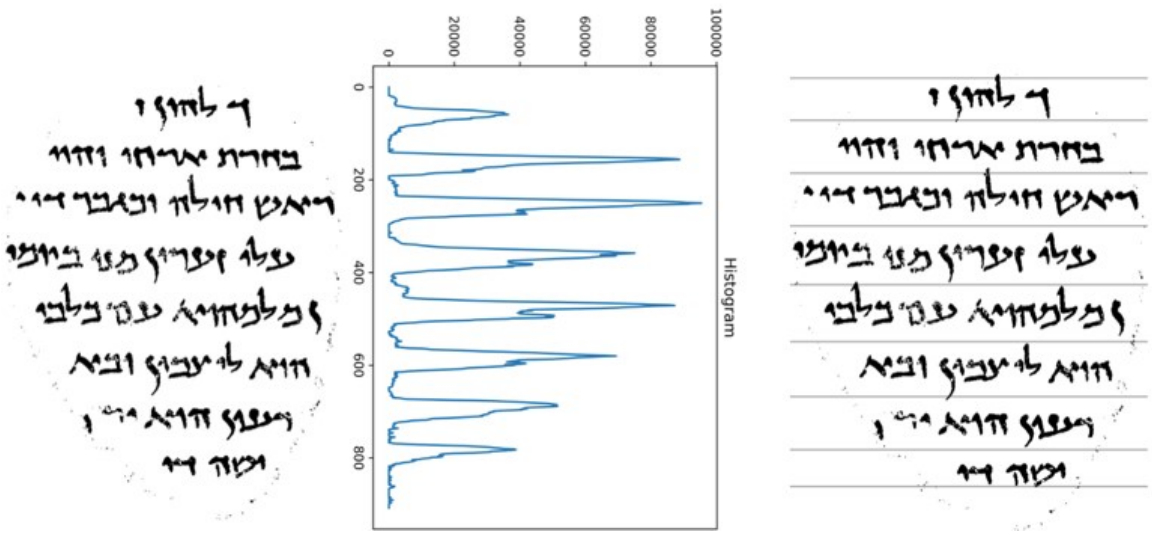


Рисунок 3.7 – Сегментація ліній тексту за допомогою гісторгамної проекції

Слід зазначити, що запропонована сегментація працює досить добре навіть із зображеннями низької якості.

3.4 Реконструкція символів та опрацювання шуму

Після того, як рядки сегментовані, зображення кожної лінії обробляється зверху на виявлених контурах, одночасно не включаючи контури менше встановленого порогу. При отриманні контурів символи на краях зображення спричинили проблему визначення всіх символів на краю як одного контуру. Це питання вирішується за допомогою простого та швидкого способу додати білі межі на 1 піксель на кожній стороні зображення. Як було зазначено вище, на цьому кроці наш спосіб обробки шумів полягає в тому, щоб ігнорувати їх під час виділення символів. Результат цього кроку показано на рисунку 8.

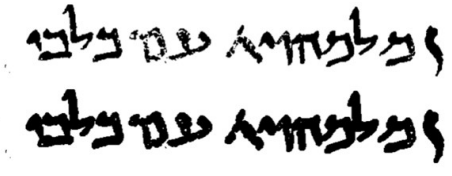


Рисунок 3.8 – Реконструкція символів та опрацювання шуму, малюючи контур

3.5 Модель-класифікатор літер

Модель CNN пройшла підготовку з набором івритських рукописних символів і перевірятиметься на зразок даних тестових даних того самого набору даних. Відсоток даних тесту з усього набору даних становить 30%. Кожна з вхідних зображень масштабується до розміру 32 × 32.

Модель підготовлена на 10 епох. Розмір партії 256 для тренувань та тестування. У цій моделі використовується категоріальна кросентропія як функція втрат. Оптимізатор RMSprop використовується для оптимізації навчального процесу. Серед 5116 зображень 3837 використовуються для навчання та 1279 використовуються для перевірки (25%).

Для реалізації CNN ми використовуємо 6 згорткових шарів і 1 повністю пов'язаний шар. Узор тричі аналогічний. Перший набір шарів може бути описаний наступним чином. Він має два згорткових шари з 32 фільтрами / ядрами з розміром вікна 3 × 3, а потім максимальним об'ємом шарів з розміром вікна 2 × 2. Також включено відсівний шар із коефіцієнтом відсіву 0,25, щоб уникнути переобладнання. У цьому випадку число ядра має такий самий порядок, що і ряд класів символів. Наступні два набори майже однакові, але вони мають 64 ядра. Нарешті, додається повністю з'єднаний щільний шар, а потім шар softmax, який виконує класифікацію серед 27 класів. Нижче наведено структуру моделі в таблиці 3.1.

Таблиця 3.1 – Структура згорткової нейронної мережі

Номер	Шар
1	2 Conv2D шарів з 32 ядрами з розміром вікна 3×3, relu як функція активації
2	Max pooling шар з вікном 2×2
3	A dropout шар з dropout з межою 0.25

4	2 Conv2D шари з 64 ядрами з вікном 3×3, relu як функція активації
5	Max pooling шар з вікном 2×2
6	A dropout шар з співвідношенням 0.25
7	2 Conv2D шари з 64 ядрами з розміром вікна 3×3, relu як функція активації
8	Max pooling шар з розміром вікна 2×2
9	Dropout шар з співвідношенням 0.25
10	Dense шар який виконує класифікацію серед 27 класів з relu як функцією активації
11	Dropout шар з співвідношенням 0.5
12	Dense шар з softmax як функцією активації

Модель оцінювалася з використанням 10-кратної крос-валідації. Середнє значення втрати k-fold становить 0,3, а середнє значення точності k-fold становить 0,93. Криві втрати та точності наведені на рисунках 9 та 10.

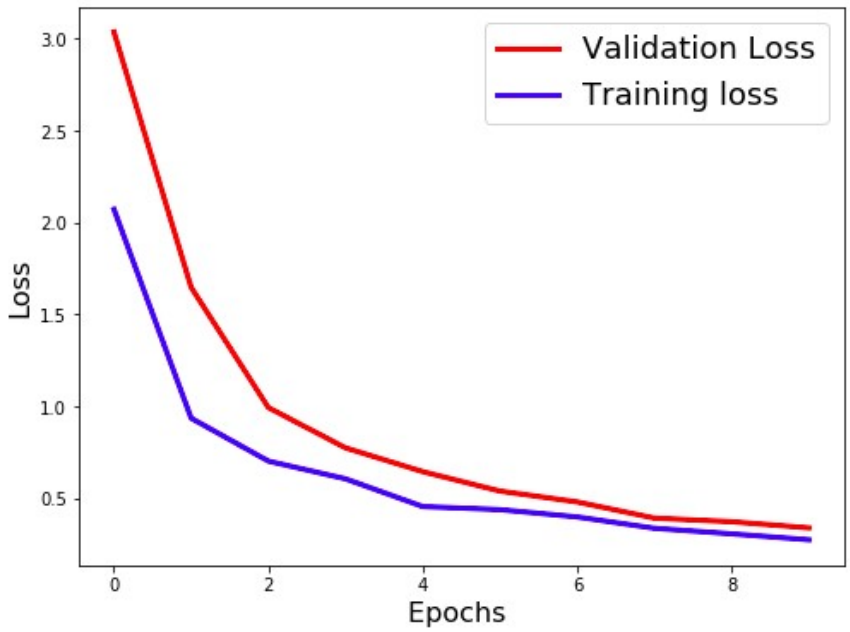


Рисунок 3.9 – Крива втрат

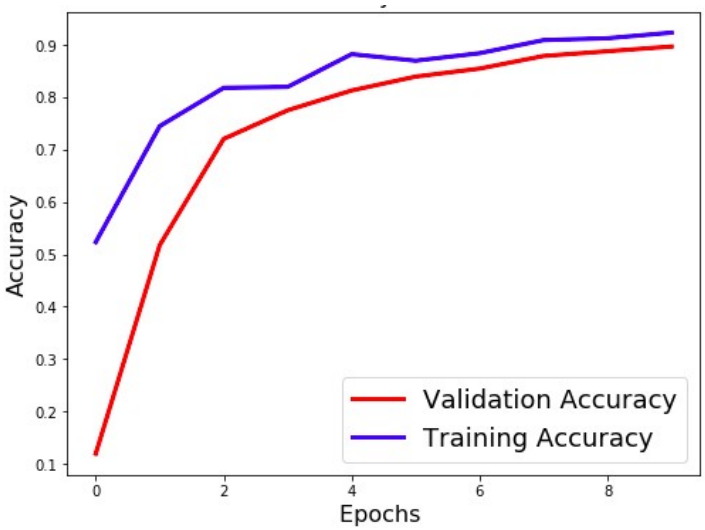


Рисунок 3.10 – Крива точності

Оцінені втрати за даними тесту дорівнюють 0,5, тоді як точність дорівнює 0,85. Як інструмент реалізації моделі CNN використовувалася високорівнева нейронна мережа API Keras, написана на Python і здатна працювати на вершині TensorFlow з огляду на те, що вона розроблена з урахуванням можливості швидкого експерименту.

3.6 Збагачення даних

Збагачення даних перетворює нашу базову інформацію. У нашому випадку, він просто приймає наш набір даних зображень і перетворює їх шляхом обертання, масштабування, зсуву. Це робить нашу модель більш стійкою до переобладнання та чисельно збільшує розмір набору даних [8]. Крім того, він додає еластичні гауссові спотворення та випадкове стирання. Використання пружних спотворень імітує природний процес почерку [9].

Мета випадкового стирання полягає в тому, щоб робити моделі надійними для закупорки шляхом випадкового заміщення прямокутних областей з довільними значеннями пікселів. Процентна область зображення, що закривається випадковим прямокутником, становить 0,4 [10].

Обертання та зсув обмежуються певними кутами, які імітують перекіс і рухомість, присутні в природному почерку. Максимальна кількість градусів для зсуву та обертання праворуч і ліворуч становить менше 10 градусів. Значення від 0 до 1, що представляє ймовірність виконання операції, встановлено на 0,5.

Приклад збагачення даних зображені на рисунку 3.11.

Рисунок 3.11 – Приклад збагачених даних

3.7 Розпізнавання літер за допомогою комплексного підходу (ковзного вікна і згорткової нейронної мережі)

Цілісний підхід використовується для сегментації символів та розпізнавання символів. Подібний підхід був запропонований Нільсоном [11] і Кейсі, Леколіне [12]. Ці два етапи об'єднані для проведення сегментації на основі визнання. Рішення щодо сегментації листа здійснюється на основі прогнозованої ймовірності за моделлю розпізнавання символів. Найвища ймовірність призводить до більш правильної сегментації. На цьому кроці образ попередньо сегментованої лінії подано у вигляді введення. Є два етапи, які повторюються для кожного потенційного символу. Перший – це генерація гіпотез сегментації однієї літери (горизонт). Другий – це вибір найкращої гіпотези (етап перевірки) [12]. Точніше, на першому етапі розмір вікна лінійного зображення дорівнює висоті рядка, оскільки існує припущення, що символ не має тенденцію бути ширшим, ніж це число. Розмір підвалу – це половина висоти рядка, якщо ця довжина є шириною літери. Розмір кроку становить 10 пікселів. Напрямок справа наліво, коли горизонтальна координата змінюється. Для кожного

зображення у вікні клас передбачається за його ймовірністю. На другому етапі з отриманого списку зображень від вікна приймається той, який має найвищу передбачувану ймовірність для цього вікна. Горизонтальна координата витягується з метою скорочення зображення та отримання залишку границі зображення як початкової точки до ітерації алгоритму ковзного вікна. Кожен наступний крок починається з витягнутого на попередньому кроці координати, зміщеної постійним ухилом, щоб запобігти розрізанню частин, що перекриваються. Ухилення дорівнює третині ширини підвалу. Повторюється, тоді коли залишкове зображення більше, ніж розмір підвалу. Розміри обох вікон і підсвічників постійні. Проте початкова координата підводячого проходу динамічно отримується на кожному етапі ітерації. Приклад результатів розсувного вікна показано на рисунках 3.12 та 3.13. Можливості знайдених букв, як прогнозованих символів, складають 0,8 та 0,9, відповідно.

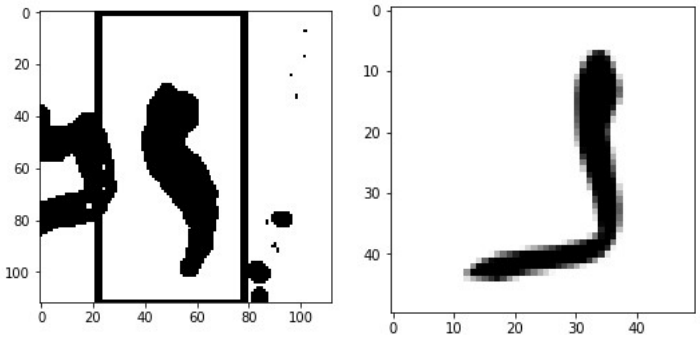


Рисунок 3.12 – Результат кроку вікна з неперетинаючими літерами зліва і спрогнозованою літерою справа

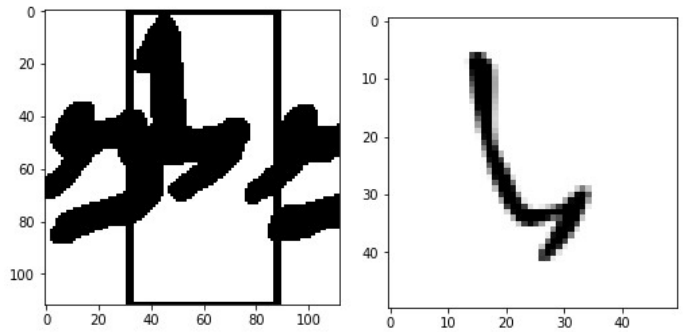


Рисунок 3.13 – Результат кроку вікна з перетинаючими літерами зліва і спрогнозаваною літерою справа

З урахуванням виходу списку передбачуваних класів з його ймовірністю виконується етап фільтрування. Ймовірності менше 0,1 відфільтровані.

Перевага цього підходу полягає в тому, що символи прогнозуються з відносно високою ймовірністю. Недоліком є те, що деякі символи пропускаються.

3.8 Оцінка моделі

Оцінка моделі подана у таблиці 3.1 за аналізом результатів відносно початкових даних і прогнозів за моделлю.

Таблиця 3.1 – Оцінка фактичних і передбачених значень

Фактичний клас	Передбачений клас	
	True(1)	False (0)
True (1)	TP – істинно позитивний	FN – хибно негативний
False (0)	FP – хибно позитивний	TN - істинно негативний

Для оцінки моделі використовують такі показники:

1. False positive rate – хибно позитивний показник

$$FPR = \frac{FP}{FP+TN} \quad (3.1)$$

2. Precision – акуратність

$$Precision = \frac{TP}{TP+FP} \quad (3.2)$$

3. Recall – відгук

$$Recall = \frac{TP}{TP+FN} \tag{3.3}$$

4. Accuracy – точність

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \tag{3.4}$$

3.9 Вибір програмної платформи

В останні роки машинне навчання перетворилося в тенденцію небувалої сили. Зростання даного напряму стимулюється не лише дешевизною хмарних середовищ, але й доступністю найпотужніших процесорів та відеокарт, застосовуваних для подібних обчислень, також з'явилася маса фреймворків для машинного навчання. Майже всі з них є відкритим програмним забезпеченням, але важливіше те, що ці фреймворки проектуються таким чином, щоб абстрагуватися від найважчих частин машинного навчання, роблячи ці технології більш доступними широкому класу розробників.

Машинне навчання — це наука змусити комп’ютер діяти, не будучи явно запрограмованим. В останнє десятиліття машинне навчання дало нам самокеровані автомобілі, розпізнавання мови, ефективний веб-пошук і значно покращене розуміння людського геному. Машинне навчання настільки розповсюджено в наш час, що ви можливо використовуєте його десятки разів на день, навіть не здогадуючись про це. Багато дослідників вважають, що це також найкращий шлях до штучного інтелекту, порівнянного з людським. У 1959 році Артур Семюель визначив машинне навчання як “Область науки, що надає комп’ютерам можливість навчатися, не будучи явно запрограмованими”. Є три основні причини необхідності машинного навчання, коли недостатньо просто запрограмувати комп’ютер. По-перше, розробники не можуть передбачити усі

ситуації, які може виявити машина. Наприклад, робот, розроблений для проходження лабіринтів мусить дізнаватися план кожного нового лабіринту, який йому трапляється. По-друге, розробники не можуть передбачити усі зміни з часом: програма для прогнозування цін на фондовій біржі мусить навчатися для адаптації, коли умови на ринку змінюються. По-третє, іноді люди не мають жодного уявлення про те, як запрограмувати розв'язок. Наприклад, більшість людей добре розпізнають обличчя своїх знайомих, але навіть найкращі програмісти не можуть створити програму для виконання цієї задачі, не використовуючи алгоритми навчання. Навчання, як і інтелект, охоплює такий широкий спектр процесів, що складно дати йому точне визначення. Словникові визначення включають такі фрази, як "отримувати знання, розуміння або вміння за допомогою вивчення, інструктування або досвіду" і "модифікація поведінкової тенденції через досвід." Зоологи і психологи вивчають навчання людей і тварин. Існують паралелі між тваринним і машинним навчанням. Звичайно, багато технік у машинному навчанні походять від намагань психологів покращити свої теорії людського і тваринного навчання через обчислювальні моделі. Щодо машин, ми можемо сказати, дуже широко, що машина навчається коли змінює свою структуру, програму або дані (у відповідь на зовнішню інформацію) таким чином, що її очікувана майбутня продуктивність покращується. Машинне навчання зазвичай застосовується для змін у системах, що виконують завдання, пов'язані зі штучним інтелектом. Такі завдання охоплюють розпізнавання, діагностику, планування, контролювання роботів, прогнозування і таке інше. "Змінами" можуть бути або покращення вже робочих систем, або синтез цілковито нових [8].

Проведемо аналіз передових платформ для машинного навчання. Амазонівський сервіс машинного навчання надає інструменти візуалізації, які направляють користувачів через процес створення моделей машинного навчання без необхідності вивчати складні алгоритми і технології МН. Після того, як моделі будуть готові, цей веб-сервіс машинного навчання дозволяє легко отримати передбачення для застосування за допомогою простого прикладного

програмного інтерфейсу. У Amazon є свій стандартний підхід до надання хмарних сервісів: спочатку зацікавленої аудиторії надається базова функціональність, ця аудиторія розробляє щось за допомогою сервісу, а компанія з'ясовує, що ж насправді потрібно людям.

Сервіс підключається до даних, що зберігаються в хмарі у базі даних чи у сховищі, він може виконувати двійкову класифікацію, багато класову категоризацію, а також регресію за вказаними даними для створення моделі. Мало того, що він використовує дані, що належать компанії та розташовані на її серверах, так ще і моделі не можна імпортувати або експортувати, а вибірки даних для тренувань не можуть бути більше 100 Гб. Але все ж це хороший інструмент для початку, ілюструє, що машинне навчання перетворюється з розкоші в практичний інструмент. Ілюстрація сервісу на рисунку 3.1.

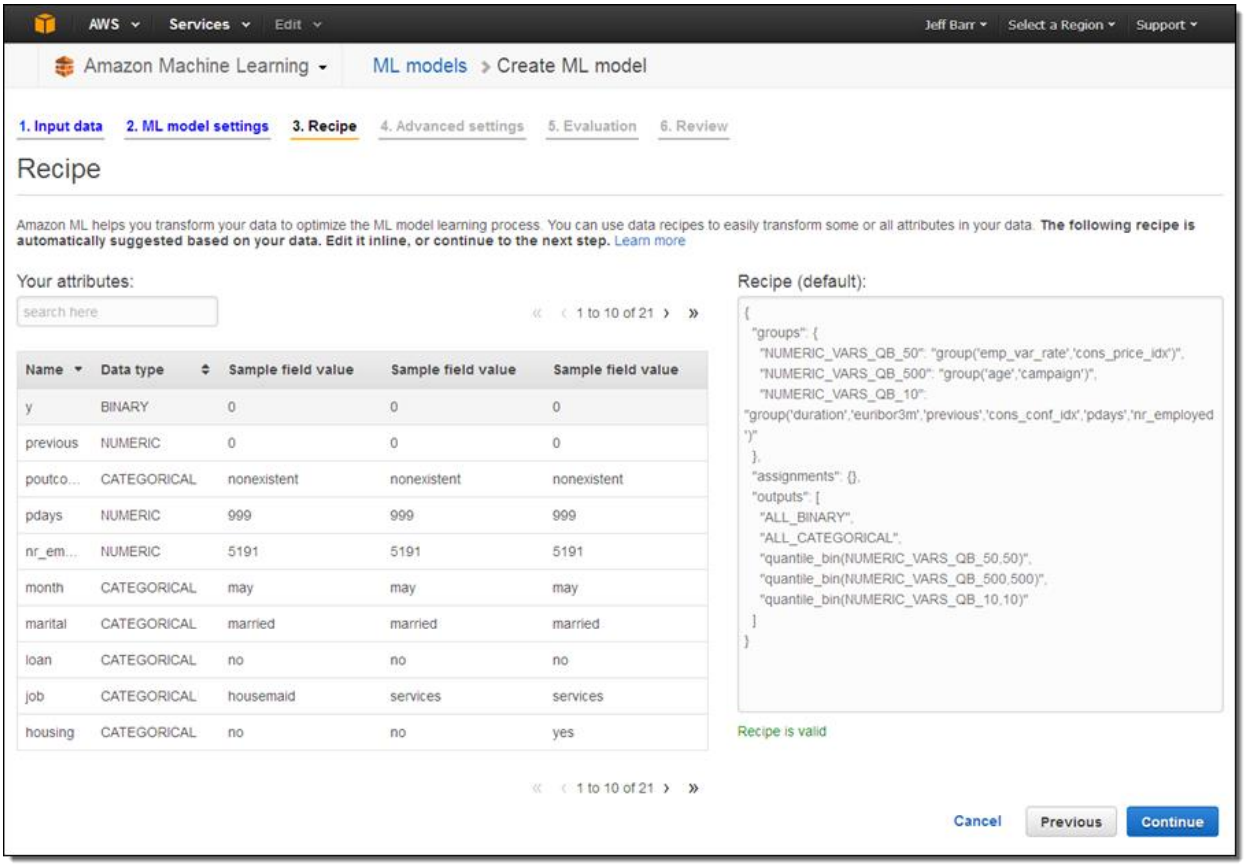


Рисунок 3.1 – Інтерфейс веб-сервісу машинного навчання Amazon Machine Learning

Майкрософтівська студія машинного навчання – Azure. Враховуючи величезний обсяг даних і обчислювальної потужності, необхідний для машинного навчання, хмари є ідеальним середовищем для ML-додатків. Microsoft вбудувала в Azure власний сервіс машинного навчання, за який можна платити тільки за фактом використання – студія МН Azure проілюстрована на рисунку 3.2. Доступні версії з помісячним і погодинною оплатою, а також безкоштовна.

Зокрема, за допомогою цієї системи створено новий проект. Студія МН Azure дозволяє створювати і тренувати моделі, перетворювати їх з інтерфейсом для надання іншим сервісам. На один акаунт може виділятися до 10 Гб місця для зберігання даних, хоча можна підключити і власне Azure-сховище. Доступний широкий спектр алгоритмів, створених Microsoft і сторонніми компаніями. Щоб спробувати сервіс, не треба навіть створювати акаунт, досить увійти анонімно, і можна використовувати студія МН Azure протягом восьми годин.

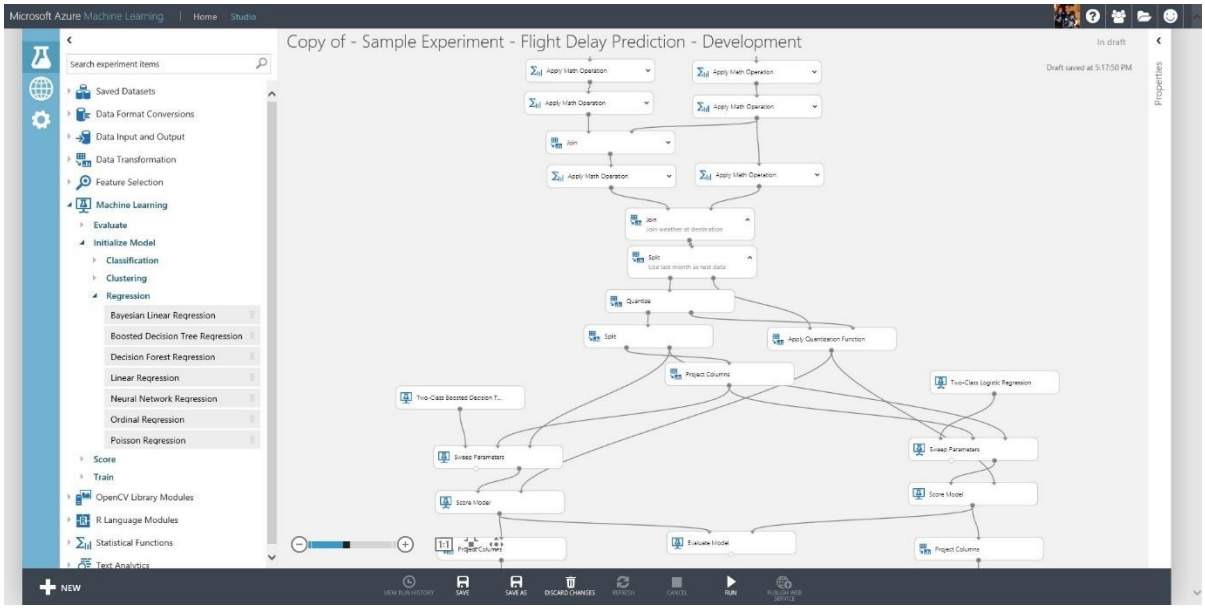


Рисунок 3.2 – Інтерфейс студії машинного навчання Microsoft Azure

Бібліотека МН Apache Spark. Apache Spark найбільш всього відомий завдяки своїй причетності до сімейства Hadoop. Але цей фреймворк для обробки даних всередині пам'яті з'явився поза Hadoop, і досі продовжує заробляти собі

репутацію за межами цієї екосистеми. Spark перетворився на звичний інструмент для машинного навчання завдяки зростаючої бібліотеці алгоритмів, які можна швидко застосувати та які знаходяться в пам'яті.

Google TensorFlow. TensorFlow – це фреймворк машинного навчання, створений для розподілу обчислень в рамках кластера. Цей фреймворк розроблявся для вирішення внутрішніх проблем Google, але пізніше компанія випустила його у відкрите плавання у вигляді продукту з відкритим кодом.

TensorFlow реалізує графи потоків даних, коли порції даних («тензори») можуть оброблятися серією описаних графом алгоритмів. Переміщення даних по системі називається «потоками». Графи можна збиратися з допомогою C++ або Python, і обробляти процесором або відеокартою. У Google є довгострокові плани розвитку TensorFlow силами сторонніх розробників, інтерфейс проілюстрований на рисунку 3.3.

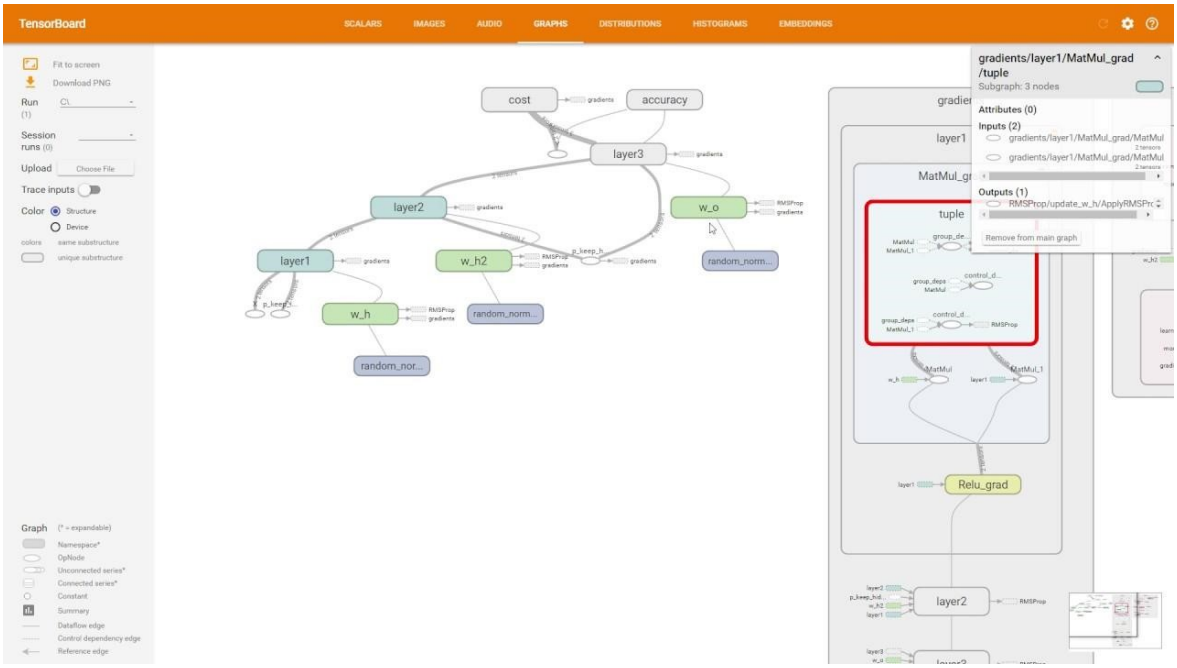


Рисунок 3.3 – Інтерфейс фреймворка машинного навчання TensorFlow

Samsung Veles – це розподілена платформа для створення програми глибокого навчання. Вона написана на C++, хоча для автоматизації та координації вузлів використовується Python. Перш ніж згодовуватися кластеру

вибірки даних, їх можна аналізувати і автоматично нормалізувати. REST API дозволяє негайно використовувати натреновані моделі в робочих проектах (якщо у вас досить потужне обладнання).

Використання Python у Veles виходить за межі простого «склеювання коду». Наприклад, Python (тепер Jupyter Notebook), інструмент для візуалізації та аналізу даних, може виводити дані з кластера Veles. Samsung сподівається, що статус відкритого коду допоможе стимулювати подальший розвиток продукту, як і портування під Windows і Mac OS X.

3.10 Висновки

Аналіз існуючих інструментів розробки нейронних мереж показав що найбільш доречним є використання розподіленої платформи Veles та мови програмування Python, оскільки дана платформа дозволяє швидко використовувати натреновані моделі в робочих проектах локально та не потребує хмарних обчислень.

Мовою програмування Python розроблено модулі попередньої обробки зображень, які включають в себе бінаризацію, пошук текстової області, подальшу обробку, такі як коригування фону, сегментацію ліній, реконструкцію символів, обробку шумів та систему розпізнавання рукописних текстів стародавньою мовою іврит за допомогою згорткової нейронної мережі і підходу ковзного вікна.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть к.т.н., доц. кафедри ПЗ Майданюк В. П. та к. т. н. доц. кафедри ПЗ Ракитянська Г. Б.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Експерт 1	2. Експерт 2
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	4	4
4	4	4
5	3	3
6	3	4
7	3	3
8	3	4
9	4	4
10	4	3
11	3	4
12	3	4
Сума балів	$СБ_1 = 42$	$СБ_2 = 44$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 43$	

Отже, з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою:

$$З_о = \frac{М}{Т_р} \cdot t, \quad (4.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 22 дні;

t - число днів роботи розробника, t = 40 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.2.

Таблиця 4.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5000	227,27	8	1818,16
Інженер-програміст	4600	163,63	40	6545,2
Всього:				8363,36

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 8363,36 = 836,33 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп} = (З_о + З_р) \cdot \frac{\beta}{100}, \quad (4.2)$$

$$Н_{зп} = (8363,36+836,33) \cdot \frac{36,3}{100} = 3339,48 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot\frac{Т}{12}, \quad (4.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т = 3 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн
Персональний комп’ютер	12000	25	2	500
Всього:				500

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (4.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 4.4 – Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	200	1	200
Пачка паперу	уп.	120	1	120
Ручка	шт.	5	1	5
Всього з урахуванням транспортних витрат				357,5

Витрати на силову електроенергію розраховуються за формулою:

$$В_е=В\cdot П\cdot Ф\cdot К_п\ ; \quad (4.5)$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=200 год.);

$К_п$ – коефіцієнт використання потужності ($К_п< 1$, $К_п = 0,7$).

$$В_е= 1,7 \cdot 0,6 \cdot 200 \cdot 0,7 = 142,8 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (4.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (8363,36 + 836,33) = 9199,69 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 8363,36+836,33+3339,48+500+142,8+357,5+9199,69= 22739,16 (грн.)

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (4.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{22739,16}{1} = 22739,16$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (4.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{22739,16}{0,9} = 25265,73 \text{ (грн.)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості

грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (4.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 35 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 35 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 200 користувачів, протягом другого року – на 150 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 700 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 300 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 35 \cdot 700 + (300 + 35) \cdot 200 = 91500 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 35 \cdot 700 + (300 + 35) \cdot (200 + 150) = 141750 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 35 \cdot 700 + (300 + 35) \cdot (200 + 150 + 100) = 175250 \text{ грн.}$$

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $Е_{абс}$вкладених інвестицій розраховується за формулою:

$$Е_{абс} = (ПП - PV), \quad (4.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

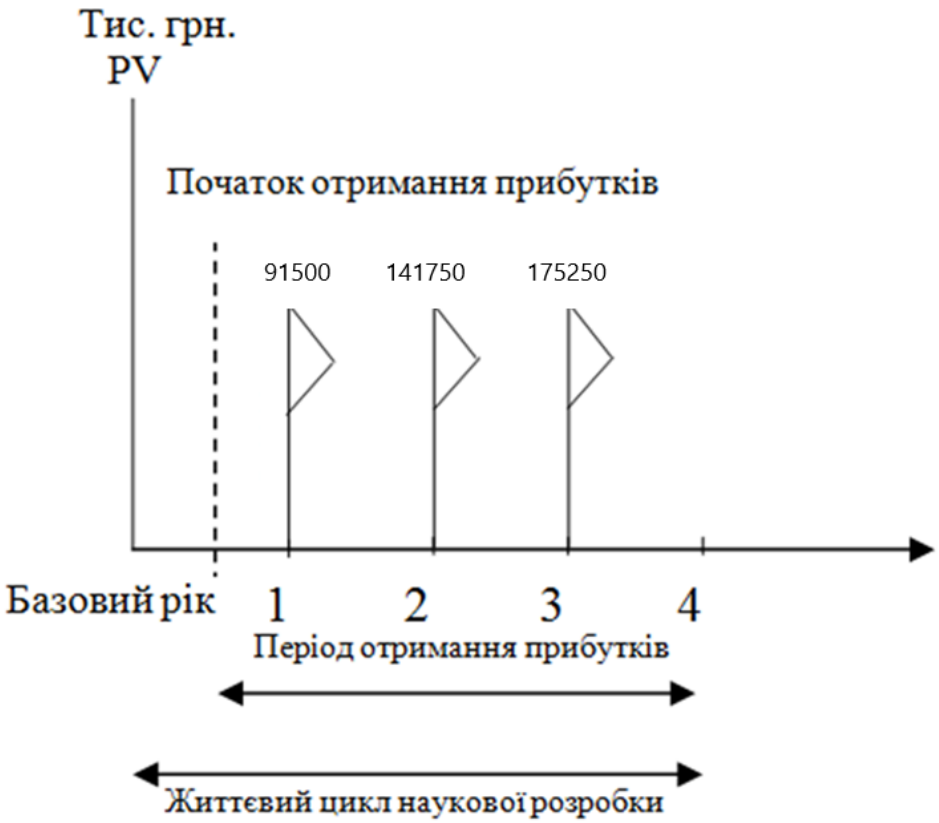


Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{25265,73}{(1+0,1)^0} + \frac{91500}{(1+0,1)^2} + \frac{141750}{(1+0,1)^3} + \frac{175250}{(1+0,1)^4} = 327082,53 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 327082,53 - 25265,73 = 301816,8 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (4.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[3]{1 + \frac{301816,8}{25265,73}} - 1 = 1,34 \text{ або } 134\ \%$$

Далі, розраховану величина $Е_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0{,}2 + 0{,}1 = 0{,}3$$

Оскільки $E_в = 134\% > \tau_{мін} = 0{,}3 = 30\%$, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

$$T_{ок} = \frac{1}{1{,}34} = 0{,}74 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи було розроблено методів розпізнавання рукописного тексту за допомогою нейромереж.

В результаті виконання кваліфікаційної роботи, було отримано наступні результати:

1. Проведено аналіз предметної галузі який показав, що розпізнавання рукописного тексту є важливою і актуальною задачею, оскільки є багато областей де потрібно розпізнавати рукописний текст, зокрема у промисловості, науці, техніці, банківській сфері, роботі поштових компаній тощо.
2. Визначено що, незважаючи на те, що проблема розпізнавання друкованих текстів в основному вирішена, розпізнавання рукописних текстів вимагає додаткових досліджень, оскільки якість та точність розпізнавання найбільше всього залежить від проведеної попередньої обробки зображення
3. Розглянуто сучасні методи обробки зображень, методи побудови математичних моделей машинного навчання, та обрано для реалізації метод згорткової нейронної мережі оскільки вона має кращу архітектуру за звичайні нейронні мережі виражену у 3х шарах нейронів та способу їх взаємодії між собою.
4. Розроблено алгоритм, який включає 8 кроків: бінаризацію вихідного зображення, виявлення області тексту, сегментації текстових рядків, реконструкції символів та обробки шумів, модель класифікатора символів, збагачення даних та сегментацію символів на основі класифікатора.
5. Обрано розподілену платформу Veles Machine Learning, оскільки дана платформа дозволяє швидко використовувати натреновані моделі в робочих проектах локально та не потребує хмарних обчислень.
6. Мовою програмування Python розроблено модулі попередньої обробки зображень, які включають в себе бінаризацію, пошук текстової області, подальшу обробку, такі як коригування фону, сегментацію ліній,

реконструкцію символів, обробку шумів та систему розпізнавання рукописних текстів стародавньою мовою іврит за допомогою згорткової нейронної мережі і підходу ковзного вікна.

Підсумовуючи усе вищесказане, можна зробити висновок, що задачі магістерської кваліфікаційної роботи було виконано у повному обсязі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Maad M. M. Handwriting Recognition Methods / Maad M. M. // IEEE Transactions on Signal Processing. - 2005. - pp. 1-3.
2. FischerA.Character prototype selection for handwriting recognition in historical documents with graph similarity features/ Fischer A., Bunke H. // Proc. 19th European Signal Processing Conference. - 2011. - pp. 1435–1439.
3. GatosB.Anadaptivebinarisationtecacpehniqueforlowqualityhistoricaldocuments. Document analysis systems. / Gatos B., Pratikakis I., Perantonis S. J. – Berlin: VI Lecture notes in computer science. - 2004. – Vol. 3163, pp. 102–113.
4. Thillou C., Color binarization for complex camera-based images / Thillou C., Gosselin B. // Electronic Imaging Conference of the International Society for Optical Imaging. - 2005. – pp. 1 – 8.
5. Otsu, N. A threshold selection method from grey-level histograms. / Otsu, N. // IEEE Transactions on System, Man, and Cybernetics. - 1979. – pp. 62-66.
6. Герасимов Б. М. Человеко-машинные системы принятия решений с элементами искусственного интеллекта / Герасимов Б. М. , Тарасов В. А., Токарев И. Б. – К.: Наукова Думка, 1993. – 184 с.
7. Химмельблау Д. Анализ процессов статистическими методами / Химмельблау Д. – М.: Мир, 1973. – 957 с
8. Shafer G. Probabilistic Expert Systems. [Text] / Society for Industrial and Applied Mathematics, Philadelphia, PA, 1996.
9. Lauritzen S. L. and Spiegelhalter D. J. Local computations with probabilities on graphical structures and their application to expert systems [Text] / Journal of Royal Statistics Society, series B – statistical methodology. – 1988. – Vol. 50, No2. – 157-194 p.
10. Lauritzen S. L. and Jensen F. V. Local computation with valuations from a commutative semigroup [Text] / Annals of Mathematics and Artificial Intelligence, – New York: Springer, 1997. – Vol. 21, No 1. – pp. 51-69.

11. Хабаров С. П. Экспертные системы [Электронный ресурс] : Конспект лекций. – Режим доступа : http://firm.trade.spb.ru/serp.
12. Casey, R. A survey of methods and strategies in character segmentation./ Casey R., Lecolinet, E.: // IEEE Transactions on System, 1996 - pp. 690–706.

ДОДАТКИ

ДОДАТОК А. Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ

д.т.н., проф. О. Н. Романюк

"____ "_____2019 р.

Технічне завдання

на магістерську кваліфікаційну роботу «Розробка методів розпізнавання рукописного тексту за допомогою нейромереж» за спеціальністю 121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:

_____ к.т.н., доц. Майданюк В. П.

"____"_____2019 р.

Виконав:

_____студент гр.1ПІ-18м О.В. Пупко

"____"_____2019 р.

Вінниця – 2019 року

1. Галузь застосування – комп’ютерні системи.

2. Підстава для розробки.
Підставою для розробки магістерської кваліфікаційної роботи є рішення засідання кафедри програмного забезпечення (протокол No__ від «__»_____ 2017 року).

3. Мета та призначення розробки.
Мета виконання магістерської кваліфікаційної роботи – підвищення достовірності розпізнавання рукописних текстів мовою іврит за рахунок удосконалення методів попередньої обробки зображень.
Призначення роботи – модифікація методу попередньої обробки зображень рукописних текстів мовою іврит.

4. Джерела розробки:
- завдання на магістерську кваліфікаційну роботу.

5. Технічні вимоги:
- Операційна система – Unix*.
- Мови програмування – Python.
- Технології – Veles Learning Machine, OpenCV.

6. Кліматичні умови.
Температурний діапазон +5оС +40оС, відносна вологість повітря не більше 75% та тиск – 720-740 мм. рт.ст.

7. Конструктивні вимоги.
Графічна та текстова документація повинна відповідати всім діючим стандартам України.

8. Перелік технічної документації, що пред’являється по закінченню робіт:

- пояснювальна записка до МКР;
- лістинги програми.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Обґрунтування доцільності розробки та постановка задачі дослідження	04.09.2019 – 29.09.2019	Виконано
2	Розробка методів обробки та розпізнавання тексту	30.09.2019 – 26.10.2019	Виконано
3	Розробка програмних модулів системи обробки рукописного тексту	27.10.2019 – 12.11.2019	Виконано
4	Економічна частина	13.11.2019 – 17.11.2019	Виконано

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком захисту.

Корегування технічного завдання допускається з дозволу керівника магістерської кваліфікаційної роботи.

ДОДАТОК Б. Лістинг коду

```
robot = Robot(sigma, delta, model_parameters["init_robot_pos"],
actual_angles=model_parameters["robot_actual_angles"],

compass_angles=model_parameters["robot_compass_angles"]) result =
run_model(robot, world, sigma, delta,

init_particles_pos=model_parameters["init_particles_pos"],
filtering_algorithm=algorithm)
error_norms = list(map(lambda x: np.linalg.norm(np.subtract(x[0], x[1])),
result))
plt.plot(error_norms)
plt.savefig(algorithm + ".png")

import cv2
from matplotlib import pyplot as plt import os
import numpy as np
import peakutils

# from skimage import measure
# from PIL import Image, ImageFont, ImageDraw

# Folder with original images
pathIn =
"/Users/olgamulava/Documents/Uni/5Groningen/HandwritingRecognition/Code/ImageDat
a/"

# Folders for output
pathOut1 =
"/Users/salexpro/Documents/Uni/5Groningen/HandwritingRecognition/Code/OutputOT
SU/"
pathOut2 =
"/Users/salexpro/Documents/Uni/5Groningen/HandwritingRecognition/Code/OutputGA
U/"
pathOut3 = "/Users/salexpro
/Documents/Uni/5Groningen/HandwritingRecognition/Code/OutputME AN/"
pathOut4 =
"/Users/salexpro/Documents/Uni/5Groningen/HandwritingRecognition/Code/OutputCr
opped/"
pathOut5 =
"/Users/salexpro/Documents/Uni/5Groningen/HandwritingRecognition/Code/OutputCr
oppedGAU/"
pathOut6 =
"/Users/salexpro/Documents/Uni/5Groningen/HandwritingRecognition/Code/OutputCr
oppedSmall/"
pathOut7 =
"/Users/salexpro/Documents/Uni/5Groningen/HandwritingRecognition/Code/Lines/"
pathOut8 =
"/Users/salexpro/Documents/Uni/5Groningen/HandwritingRecognition/Code/Lines/Co
nnectedComponents/"

# For output in console np.set_printoptions(threshold=np.inf)

# def largest_component(image):
# # image must be already binarized

# nb_components, labels, stats, centroids =
cv2.connectedComponentsWithStats(image, connectivity=8, ltype=cv2.CV_32S)
```

```
   #  # nb_components is the number of labels
   #  # Labels is a matrix the size of the input image where each element has a

value equal to its label.

   #  # Stats is the stat matrix - the most important
   #  # Centroids is a matrix with the x and y locations of each centroid. The
   row in this matrix corresponds to the label number.
   #
   #  sizes = stats[:, -1] #
   #  max_label = 1
   #  max_size = sizes[1]
   #  for i in range(2, nb_components):
   #  if sizes[i] > max_size:
   #  max_label = i
   #  max_size = sizes[i] #
   #  # Crop just connected component with background
   #  # img2 = np.zeros(labels.shape)
   #  # img2[labels == max_label] = 255 #
   #  # Crop image[y1:y2, x1:x2]
   #  img2 = image[stats[max_label, cv2.CC_STAT_TOP]:stats[max_label,
   cv2.CC_STAT_TOP]+stats[max_label, cv2.CC_STAT_HEIGHT], stats[max_label,
   cv2.CC_STAT_LEFT]:stats[max_label, cv2.CC_STAT_LEFT]+stats[max_label,
   cv2.CC_STAT_WIDTH]]
   #
   #      # Biggest component

   #  return img2

def largest_contour(img):
img2 = img.copy()
# Parameters: first one is source image, second is contour retrieval mode,

third is contour approximation method
_, contours, hierarchy = cv2.findContours(img2, cv2.RETR_TREE,

cv2.CHAIN_APPROX_SIMPLE)

# Find the index of the largest contour
areas = [cv2.contourArea(c) for c in contours] max_index = np.argmax(areas)
cnt = contours[max_index]

# Convex Hull
# hull = cv2.convexHull(cnt)#, returnPoints=False)

# cv2.drawContours(img2, [hull], -1, 50, 1)

# Mask initialized as white image
mask = np.ones(img2.shape[:2], dtype="uint8") * 255

# Draw on mask image filled biggest contour cv2.drawContours(mask, [cnt], -1, 0,
-1) # [hull]

# cv2.drawContours(img2, contours, -1, (255, 255, 0), -1) #
cv2.imwrite('TestCharacter.png', img2)

# Out initialized as white image
out = np.ones(img2.shape[:2], dtype="uint8") * 255 out[mask == 0] = img2[mask ==
0]
```

```
# Find coordinates of biggest contour cnt x, y, w, h = cv2.boundingRect(cnt)

# Crop image[y1:y2, x1:x2]
# imgReturn = out[y:y + h, x:x + w]

return out # imgReturn # np.array(out) # # Character Segmentation

# def #
#
#

char_contours(img):
# Image must be already binarized

# # Histogram for Line
# img_row_sum = np.sum(returnImg, axis=0) # axis=0 - x coordinate #
plt.figure(0)
# plt.plot(img_row_sum)
# plt.savefig('HistogramForLine.png', dpi=500)

# Copy the input image
# returnImg = img.copy()

#
#
#
#
#
#
#
#
#
#
#
#
method=cv2.CHAIN_APPROX_NONE)
# cv2.drawContours(returnImg, contours, -1, 100, 3) # last parameter is
thickness = 3

returnImg = np.asarray(img) # change
_, contours, hierarchy = cv2.findContours(returnImg, mode=cv2.RETR_LIST,

#
# return np.array(returnImg)

# Line segmentation function returns an array of lines images def
lines_segmentation(img_input):

img_copy = img_input.copy() img = np.invert(img_input)

# Make a graph from sum of pixels in each row
img_row_sum = np.sum(img, axis=1).tolist() # axis=1 - y coordinate

peaks = peakutils.indexes(np.array(img_row_sum), thres=0.15, min_dist=60) #
thres=0.02 / max(cb), min_dist=100)

# thres - Only the peaks with amplitude higher than the threshold will be
detected

# min_dist - Minimum distance between each detected peak print(peaks)
```

```
plt.title('Histogram')
# plt.hist(indexes, bins='auto', orientation='horizontal') plt.plot(img_row_sum)
plt.savefig('Histogram.png', dpi=500)

# Find average line height list_avg = []
for i in range(len(peaks) - 1):

list_avg.append((peaks[i+1] - peaks[i])) avg_height = int(sum(list_avg) /
float(len(list_avg))) print(avg_height)

H = img.shape[0]
lines = [] # np.zeros(len(peaks) + 1) lines.append(peaks[0] - avg_height / 2) #
first line for peak in peaks:

lines.append(min(peak + avg_height / 2, H))

# Converting float type to int lines = np.array(lines)
lines = lines.astype(int)
# print(lines)

   #  th = sum(img_row_sum) / float(len(img_row_sum))
   #  H, W = img.shape[:2] #
   #  lowers = np.array([y for y in range(H - 1) if img_row_sum[y] <= th and
         img_row_sum[y + 1] > th])
         # uppers = np.array([y for y in range(H - 1) if img_row_sum[y] > th and
         img_row_sum[y + 1] <= th]) #
   #  lines = np.zeros(len(uppers))
   #  lines[0] = max(uppers[0] - 20, 0)
   #  lines[-1] = min(lowers[-1] + 20, H) #
   #  for i in range(1, len(uppers) - 1):
   #  lines[i] = (lowers[i - 1] + uppers[i]) / 2
   #  # Converting float type to int
   #  lines = lines.astype(int)
       # print(lines)
       # for y in uppers:

# cv2.line(newImage, (0, y), (W, y), (255, 0, 0), 1) #
# for y in lowers:
# cv2.line(newImage, (0, y), (W, y), (255, 0, 0), 1)

# Width of the image W = img.shape[1]

#  for y in lines: # lines:
#  cv2.line(img_copy, (0, y), (W, y), 0, 1) # 0 - black #
#  cv2.imwrite("LineSegmentation.png", img_copy)

      lines_array = []

      for i in range(len(lines) - 1):
      # To crop image[y1:y2, x1:x2]
      img_line = np.array(img_copy[lines[i]:lines[i + 1], 0:W])
      lines_array.append(img_line)

          return lines_array

      k=0

def get_connected_components(imgline): global k
```

```
k=k+1
imgreturn = imgline.copy()
# image must be already binarized nb_components, labels, stats, centroids =

cv2.connectedComponentsWithStats(imgline, connectivity=8, ltype=cv2.CV_32S) #
charCandidates = np.ones(imgline.shape, dtype="uint8") * 255
print(nb_components)

# List of all components images from the line image conComponentsArray = []

# Use CC_STAT_AREA to filter?

for label in range(nb_components):
# Retrieving the width of the bounding box of the component width = stats[label,
cv2.CC_STAT_WIDTH]

# Retrieving the height of the bounding box of the component height =
stats[label, cv2.CC_STAT_HEIGHT]

# Retrieving the leftmost coordinate of the bounding box of the component

x = stats[label, cv2.CC_STAT_LEFT]

# retrieving the topmost coordinate of the bounding box of the component y =
stats[label, cv2.CC_STAT_TOP]

# creating the image of the component component = imgline[y:y + height, x:x +
width]

conComponentsArray.append(component)

cv2.rectangle(imgreturn, (x, y), (x + width, y + height), color=0, thickness=2)

cv2.imwrite("cc" + str(k) + ".png", imgreturn) print("len(conComponentsArray)
=", len(conComponentsArray)) return conComponentsArray

drawimg = imgline.copy()

for compLabel in range(1, nb_components, 1):
cv2.rectangle(drawimg, (stats[compLabel, 0], stats[compLabel, 1]),

(stats[compLabel, 0] + stats[compLabel, 2], stats[compLabel, 1] +
stats[compLabel, 3]),(0, 0, 255), 2)

cv2.imwrite(pathOut8 + "DrawConComp" + str(imgline) + ".png", drawimg)

sizes = stats[:, -1]

max_label = 1
max_size = sizes[1]
for i in range(2, nb_components):

if sizes[i] > max_size: max_label = i

max_size = sizes[i]

# Crop just connected component with background img2 = np.zeros(labels.shape)
img2[labels == max_label] = 255
```

```
# Crop image[y1:y2, x1:x2]
img2 = imgline[stats[max_label, cv2.CC_STAT_TOP]:stats[max_label,

cv2.CC_STAT_TOP]+stats[max_label, cv2.CC_STAT_HEIGHT], stats[max_label,

cv2.CC_STAT_LEFT]:stats[max_label, cv2.CC_STAT_LEFT]+stats[max_label,
cv2.CC_STAT_WIDTH]]

# Biggest component - img2 conComponentsArray.append(img2)

# Draw contours on a line image def draw_contours(imgline):

global k k=k+1

# imgLineCopy = cv2.subtract(255, imgline) # imgLineCopy = imgline.copy()

# Add white borders on each side
bordersize = 1
imgWithBorder = cv2.copyMakeBorder(imgline, top=bordersize,

bottom=bordersize, left=bordersize, right=bordersize,
borderType=cv2.BORDER_CONSTANT, value=[255, 255, 255])

cv2.imwrite("TestBorder.png", imgWithBorder)

# Parameters: first one is source image, second is contour retrieval mode, third
is contour approximation method

_, contours, hierarchy = cv2.findContours(imgWithBorder, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

contoursArray = []

for c in contours:
if 50 <= cv2.contourArea(c) <= 10000:

contoursArray.append(c)

# Find areas
# areas = [cv2.contourArea(c) for c in contours] # print("areas", areas)

# ? Why black line ? - Fix that

img_return = cv2.drawContours(imgWithBorder, contoursArray, -1, color=0,
thickness=2)

# Crop borders
w, h = img_return.shape[:2]
img_return = img_return[bordersize:w, bordersize:h-bordersize]

cv2.imwrite("TestContoursWords" + str(k) + ".png", imgWithBorder)

# Return the same image with contours drawn on the top return img_return
# return contoursArray

# Read an image
img = cv2.imread(pathIn + "P632-Fg002-R-C01-R01-fused.jpg", 0) # 0 - grayscale #
img = cv2.imread(pathIn + "P423-1-Fg002-R-C02-R01-fused.jpg", 0) # 0 - grayscale
# img = cv2.imread(pathIn + "124-Fg004.pgm", 0) # 0 - grayscale
```

```
# # Invert colors black to white, white to black # imgtest = np.invert(img)

# Show
# cv2.imshow('My image', img) # cv2.waitKey()

ret, imgBin = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
goodImage = largest_contour(imgBin)

# goodImage1 = largest_contour(imgBin)
# contours = get_contours(goodImage1)
# goodImage = cv2.drawContours(goodImage1, contours, -1, color=0, thickness=1)

cv2.imwrite('test.png', goodImage)

# Worse performance
# cropImage = largest_component(imgBin) # cv2.imwrite('cropTest.png', cropImage)
# goodImage = largest_contour(cropImage) # cv2.imwrite('test.png', goodImage)

lines_imgs = lines_segmentation(goodImage) lines_imgs_contours = []

for i in range(len(lines_imgs)):
# cv2.imwrite(pathOut7 + "SimpleTest" + "LineImage" + str(i) + ".png",

lines_imgs[i])

# CONTOURS
line_contoured = draw_contours(lines_imgs[i]) # lines_imgs[i]
# cv2.imwrite(pathOut7 + "Simple" + "LineImage" + str(i) + ".png",

line_contoured) lines_imgs_contours.append(line_contoured)

# for j in range(len(contours)):

# cv2.imwrite(pathOut8 + "test" + "LineImage" + str(i) + "CONTOUR" + str(j) +
".png", contours[j])

#  CONNECTED COMPONENTS ?
#  conComponents = get_connected_components(line_i)
#  # #
#  for #

print(conComponents)
j in range(len(conComponents)):
cv2.imwrite(pathOut8 + "test" + "LineImage" + str(i) + "OLYA" +

str(j) + ".png", conComponents[j])

# cv2.imwrite("Line Image", goodImage[20:100, 20:100])
# cv2.imwrite("LineImage.png", Image.fromarray(goodImage[1:40, 1:30])) #
Image.fromarray(goodImage[0:200, 0:200]).save("LineImage.png")

# Open the folder with input images listing = os.listdir(pathIn)

for file in listing:
if not file.startswith('.'): # check for hidden files

# Read an image
img = cv2.imread(pathIn + file, 0)
```

```
# OTSU Binarization

ret, imgi = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

cv2.imwrite(pathOut1 + file, imgi)

# Cropped important part as a largest connected component with borders, small
size

goodImage = largest_contour(imgi) cv2.imwrite(pathOut6 + file, goodImage)

# cv2.imwrite(pathOut7 + file, np.array(char_contours(goodImage)))

# # GAUSSIAN Binarization

# imgi = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 11, 2)

#  cv2.imwrite(pathOut2 + file, imgi) #
#  # reverse black to white to make this work
#  # cv2.imwrite(pathOut5 + file, largest_component(imgi)) #
# # MEAN Binarization

# imgi = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_MEAN_C,

cv2.THRESH_BINARY, 11, 2)
# cv2.imwrite(pathOut3 + file, imgi)

# Creat a list of segmented lines lines_imgs = lines_segmentation(goodImage)

# List of contoured line images lines_imgs_contours = []
for i in range(len(lines_imgs)):

# Draw contours
# line_contoured = draw_contours(lines_imgs[i]) # lines_imgs[i]
cv2.imwrite(pathOut7 + file + "LineImage" + str(i) + ".png",

line_contoured) # could be commented later # Append a contoured line image

lines_imgs_contours.append(line_contoured)

image1 = cv2.imread(pathOut1 + "P344-Fg001-R-C01-R01-fused.jpg", 0) #
cv2.imshow('My image', image1)
# cv2.waitKey()

# Template is one char
template = cv2.imread("CorrectData/" + "Bet.png", 0) w, h = template.shape[::-1]

# finding the location of a template image in a larger image res =
cv2.matchTemplate(image1, template, cv2.TM_CCORR_NORMED)

#n
min_val, max_val, min_loc, max_loc = cv2.minMaxLoc(res)

top_left = max_loc
bottom_right = (top_left[0] + w, top_left[1] + h)

cv2.rectangle(image1, top_left, bottom_right, (100, 100, 50), 2)
cv2.imwrite("my.png", image1)
```

```
# cv2.imshow("TM_CCORR_NORMED", image1)
cv2.waitKey()

print(top_left, bottom_right)

# plt.subplot(121)
# plt.imshow(res, cmap='gray') plt.xticks([])
plt.yticks([])
# plt.subplot(122) plt.imshow(image1, cmap='gray') plt.show()
```

ДОДАТОК В. Ілюстративний матеріал

Ілюстративний матеріал до захисту магістерської кваліфікаційної роботи

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доцент _____ В. П. Майданюк

Рецензент, к. т. н., доцент кафедри КН _____ І. В. Богач

Нормоконтроль, к. т. н., доцент _____ В. П. Майданюк

Виконавець, студент гр.1ПІ-18м _____ О. В. Пупко

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

Магістерська кваліфікаційна робота

на тему: Розробка методів розпізнавання рукописного тексту за допомогою нейромереж

Виконав: ст. гр. 1ПІ-18м – Пупко О. В.
Науковий керівник: к.т.н., доц. Майданюк В. П.

1

Рисунок В.1 – Слайд 1. Тема, автор та керівник бакалаврської дипломної роботи

Області використання

- Аналіз документації
- Інтерпретація поштової адреси
- Перевірка підпису
- Обробка банківських чеків
- Оцифрування історичних документів

2

Рисунок В.2 – Слайд 2. Області використання

Мета, об’єкт, предмет та завдання роботи

Метою роботи є підвищення достовірності розпізнавання рукописних текстів мовою іврит за рахунок удосконалення методів попередньої обробки зображень.

Об’єктом дослідження є процес розпізнавання рукописних текстів

Предметом дослідження є методи та програмні засоби первинної обробки зображень рукописних текстів стародавньою мовою іврит

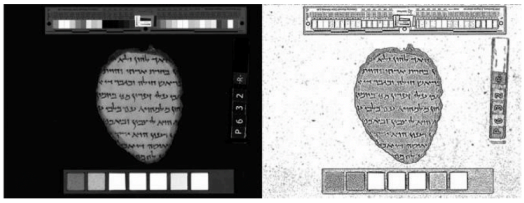
Завдання, що розв’язувались у роботі:

- аналіз існуючих методів і засобів попередньої обробки зображень для визначення напрямків підвищення їх продуктивності та достовірності;
- модифікація методу попередньої обробки зображень рукописних текстів мовою іврит;
- розробка комплексного методу розпізнавання літер;
- розробка програмних компонент та систему розпізнавання рукописних текстів на основі запропонованих методів;

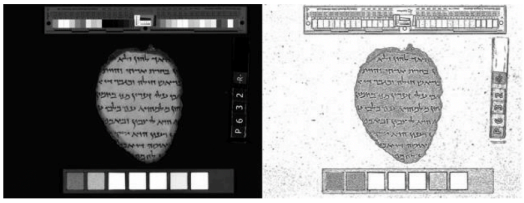
3

Рисунок В.3 – Слайд 3. Мета, об'єкт, предмет та завдання роботи

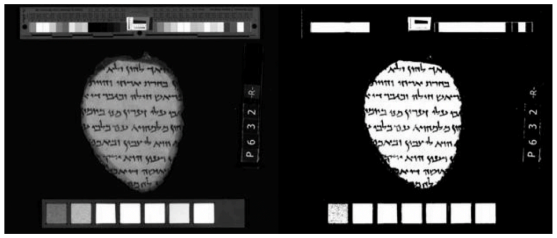
Бінаризація



Адаптивна середнього порогу



Гаусівська адаптивна



OTSU

4

Рисунок В.4 – Слайд 4. Бінаризація

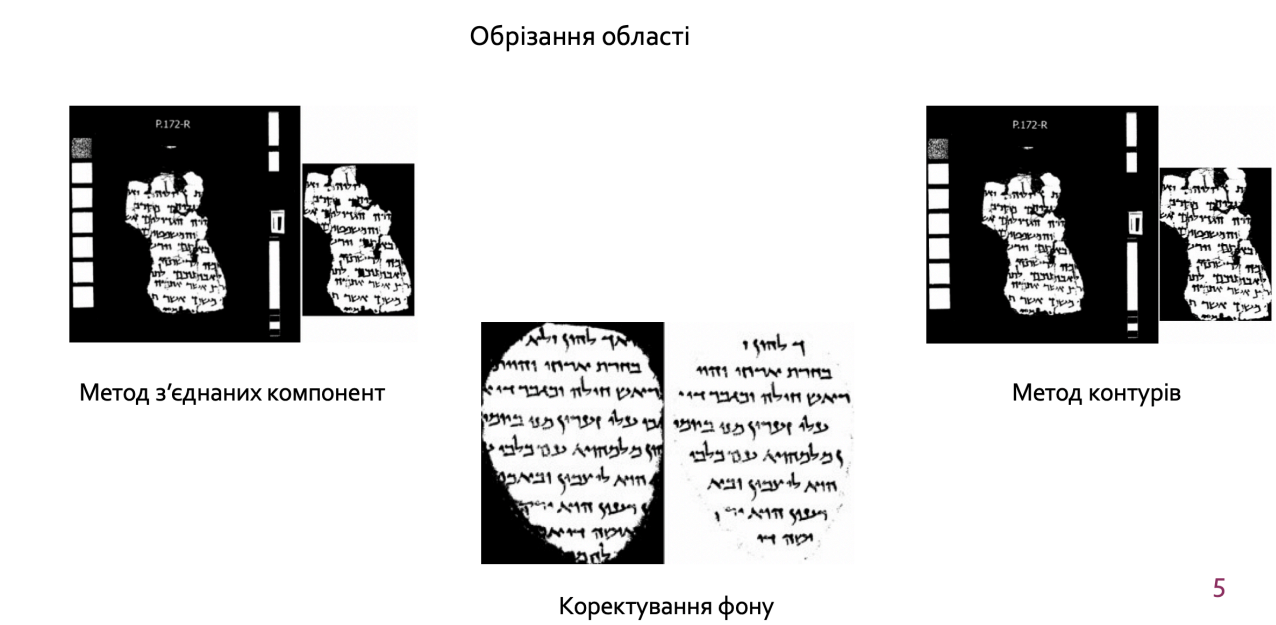


Рисунок В.5 – Слайд 5. Обрізання області

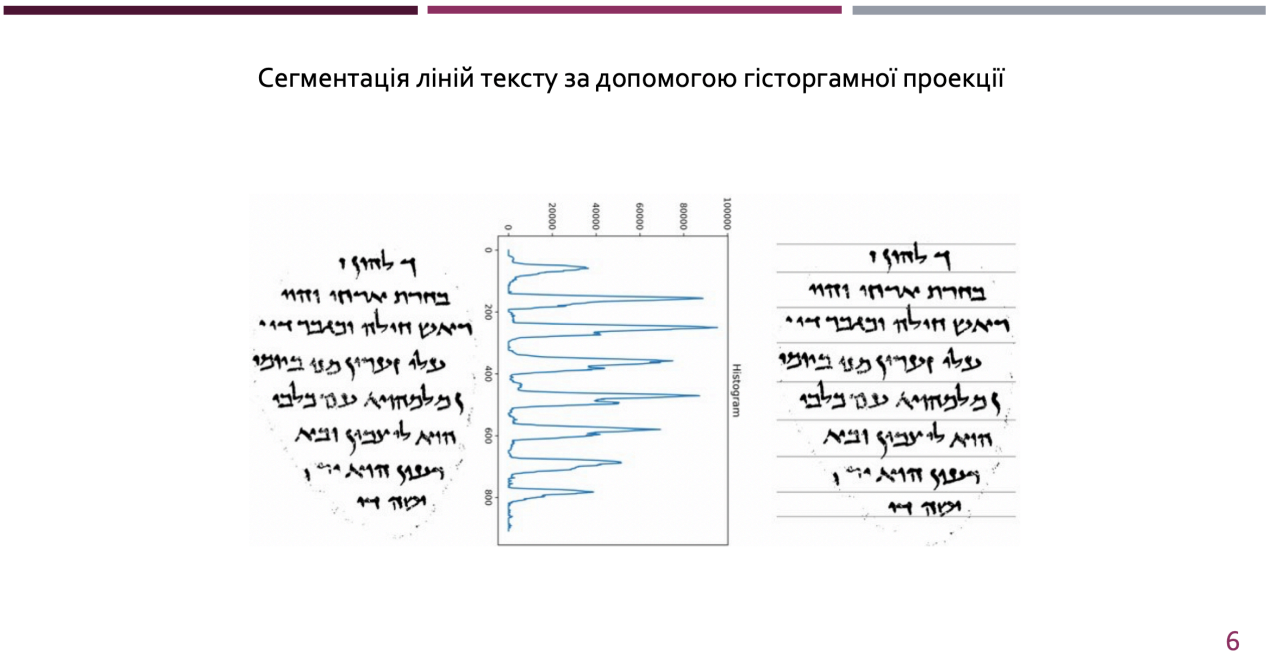
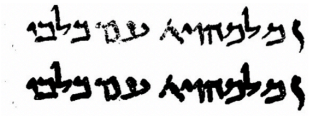


Рисунок В.6 – Слайд 6. Сегментація ліній тексту за допомогоюю гістограмної проекції

Реконструкція символів та опрацювання шуму

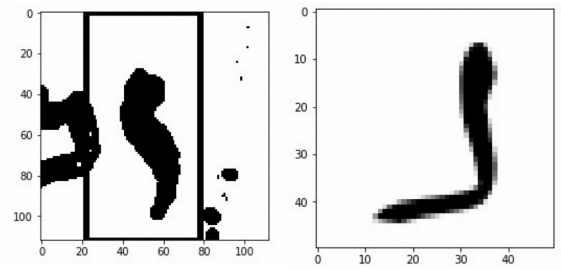


Збагачення даних

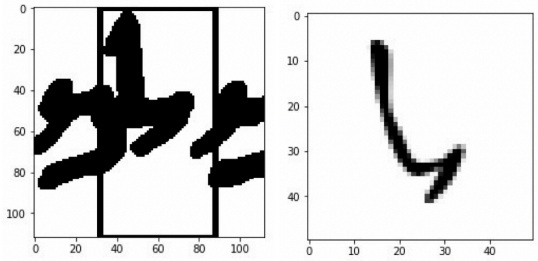
7

Рисунок В.7 – Слайд 7. Реконструкція символів та опрацювання шуму

Прогнозування літер методом сковзного вікна



Неперетинаючі літери



Перетинаючі літери

8

Рисунок В.8 – Слайд 8. Прогнозування літер методом сковзного вікна

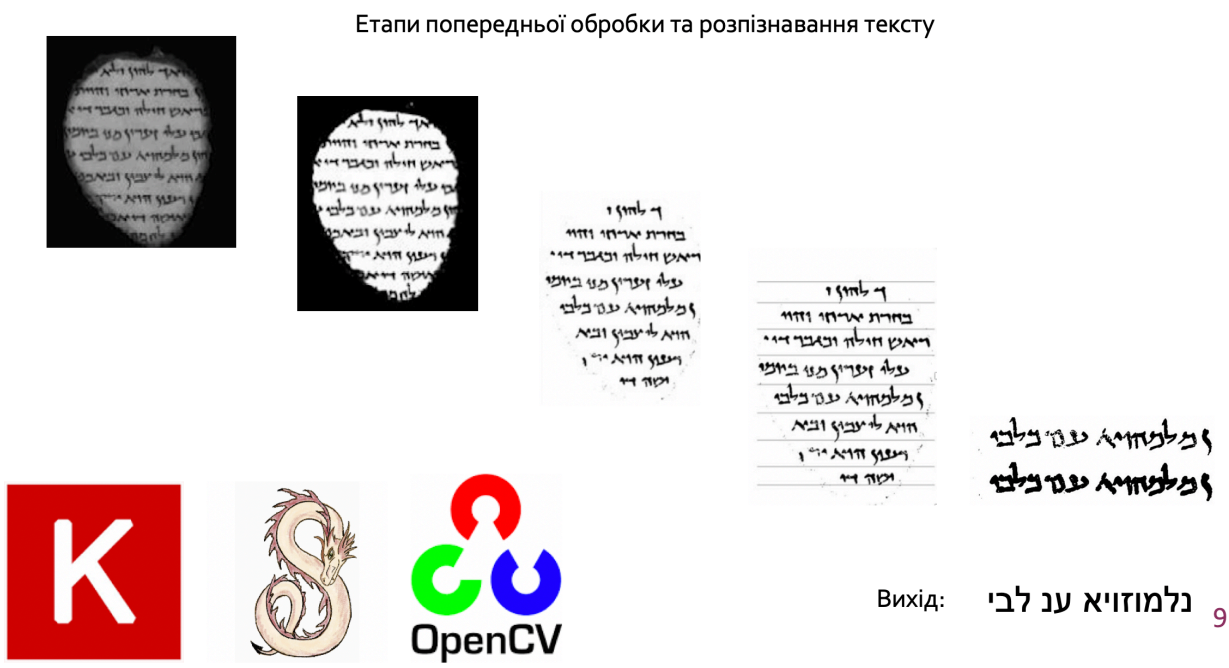


Рисунок В.9 – Слайд 9. Етапи попередньої обробки та розпізнавання тексту

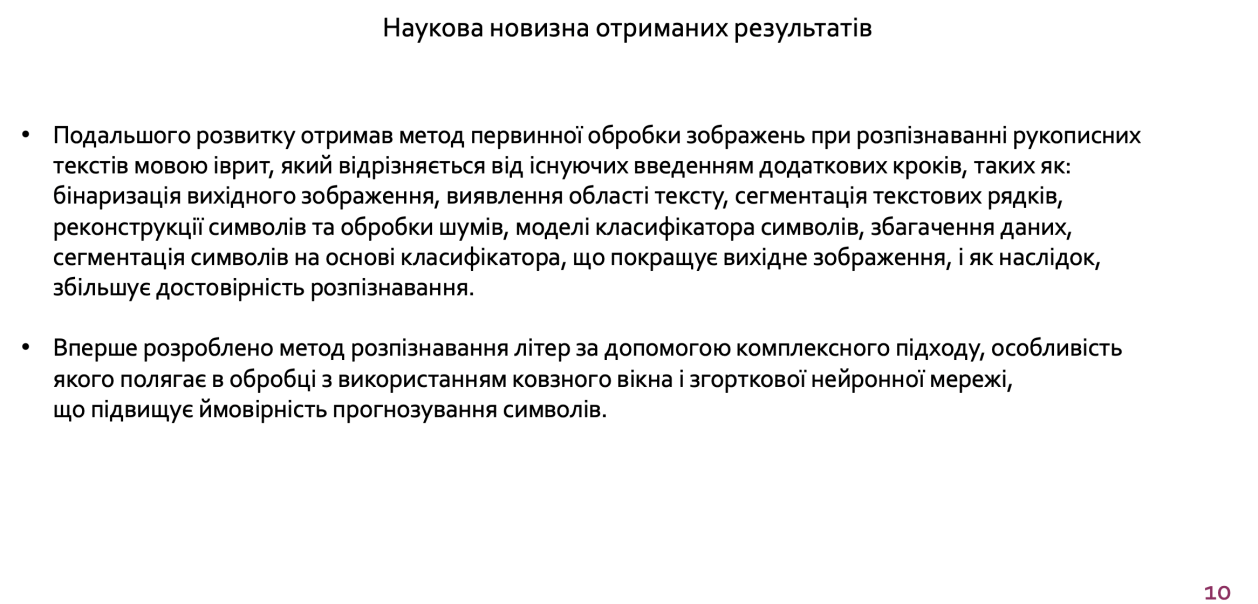


Рисунок В.10 – Слайд 10. Наукова новиизна отриманих результатів

Дякую за увагу!

Рисунок В.11 – Слайд 11. Дякую за увагу