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: Методи та програмні засоби розпізнавання зображень

Виконав: студент II курсу
групи 2ПІ-18 м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Ель Жеддауї Хашем
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Хошаба О.М.
(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

Ель Жеддауї Хашему

1. Тема роботи – Методи та програмні засоби розпізнавання зображень.
Керівник роботи: Хошаба Олександр Мирославович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від "__"_____2019 року №___
2. Строк подання студентом роботи _____
3. Вихідні дані до роботи: базові методи детекції – методи MSER, салієнтності; методи розпізнавання – сегментація слів; вихідні дані для розпізнавання – навчена нейронна мережа MSER; навчена нейронна мережа Saliency; навчений класифікатор слів; словник образів; словник слів; вихідні дані – виявлені текстові області, розпізнаний текст, зображення, замінені текстом.
4. Зміст розрахунково-пояснювальної записки: Аналіз предметної області; Аналіз методів розпізнавання; аналіз методів фільтрації; аналіз нейронних мереж OCR, аналіз методів MSER, Saliency; розробка метода детекції; розробка метода сегментації; розробка метода розпізнавання; розробка моделі детекції; розробка моделі розпізнавання; розробка загального алгоритму роботи програми; розробка бібліотеки браузера; економічна частина.
5. Перелік графічного матеріалу: схема методу виявлення; схема методу виявлення; Діаграма станів для процесу робота програми; приклади методів розпізнавання; приклади роботи програми.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада	Підпис, дата

	консультанта	завдання видав	завдання прийняв
1-4	Хошаба Олександр Мирославович, к.т.н., доцент кафедри ПЗ		
5	Бальзан М.В., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз наявних методів і засобів розпізнавання зображень, зокрема розпізнавання тексту в зображенні	07.10.2018 – 27.10-2019	Вик.
2	Розробка методу детекції тексту в зображенні та його розпізнавання	28.10.2019 – 8.11.2019	Вик.
3	Розробка програмних компонентів моделі машинного навчання	9.11.2019 – 20.11.2019	Вик.
4	Розробка програмних компонентів бібліотеки JavaScript для застосування моделі машинного навчання у браузері	21.11.2019 – 29.11.2019	Вик.
5	Економічна частина	1.12.2019 – 05.12.2019	Вик.

Студент _____ **Ель Жеддауї Х.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Хошаба О.М.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі запропоноване програмне рішення розпізнавання зображень, а саме розпізнавання тексту в зображеннях на веб-сторінках. Розглянуті основні наявні методи розпізнавання. Запропоноване комбінування двох методів, що дозволило досягти вищої точності та ефективності детекції тексту в зображеннях шляхом зведення зон пошуку до конкретних локацій. В основу алгоритму, що застосовується в роботі, покладено селективний пошук, який є поєднанням вичерпного пошуку та сегментації. Разом із підготовленим класифікатором розпізнавання символів це дозволило досягти високих результатів розпізнавання. Нарешті, запропоновано метод сегментації слів, який не вимагає попередньої підготовки моделі і здатен розділити слова на символи, до розпізнавання яких і застосовується класифікатор. Для використання програмного рішення у браузері на веб-сторінках розроблено бібліотеку JavaScript. Для розробки програмного рішення використовувались мови програмування JavaScript і Python і редактор Microsoft Visual Studio Code.

ABSTRACT

In the master's qualification work an advanced solution for pattern and specifically text in image recognition on webpages is proposed. The basis of existing methods of recognition are considered. The choice of combining two advanced recognition methods allows the detection of the text’s locations from the beginning and then combine them to detect text. In that way the search area was reduced to specific locations in the image. Selective search is applied to guide the detection algorithm, which is a combination of exhaustive search and segmentation. In parallel with a trained character recognition classifier, that achieves satisfying recognition results. Finally, a word segmentation method is proposed, which doesn’t require any training and is able to segment the word into characters, where the character recognition classifier is applied. The software solution is developed using the Python and JavaScript programming languages in Microsoft Visual Studio Code.

ЗМІСТ

ВСТУП ……8
1 АНАЛІЗ І ПОСТАНОВКА ЗАДАЧІ ……12
1.1 Аналіз предметної області ……12
1.2 Аналіз програм розпізнавання зображень……...16
1.3 Аналіз методів вирішення поставленої задачі……..20
1.4 Постановка задачі ……...31
1.5 Висновки ……32
2 ПРОЕКТУВАННЯ МЕТОДУ РОЗПІЗНАВАННЯ ТЕКСТУ В ЗОБРАЖЕННЯХ ……..34
2.1 Загальна структура системи ……...34
2.2 Методи детекції символів і слів ……34
2.2.1 Метод детекції символів ……34
2.2.2. Метод формування слів ……37
2.3 Метод розпізнавання символів ……...41
2.4 Методи сегментації і розпізнавання слів ……..43
2.4.1 Сегментація слів ……43
2.4.2 Розпізнавання слів ……44
2.5 Висновки ……46
3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ СИСТЕМИ РОЗПІЗНАВАННЯ ТЕКСТУ В ЗОБРАЖЕННЯХ ……..48
3.1 Вибір засобів розробки програми ……48
3.2 Розробка програмних засобів для детекції і розпізнавання тексту в зображеннях ……50
3.2.1 Розробка програмних засобів для детекції тексту ……50
3.2.2 розробка програмних засобів розпізнавання тексту ……..55

3.3 Розробка бібліотеки JavaScript для застосування моделі машинного навчання у браузері ……..57
3.4 Тестування роботи програмних засобів ……...61
3.5 Висновки …….60
4 ЕКОНОМІЧНА ЧАСТИНА ……..65
4.1 Оцінювання комерційного потенціалу розробки ……65
4.2 Прогнозування витрат на виконання наукової роботи та впровадження результатів……...75
4.3 Прогнозування комерційних ефектів від реалізації результатів розробки……80
4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності…….82
4.5 Висновок ...……..86
ВИСНОВКИ …….87
Перелік посилань …….89
Додаток А - Технічне завдання ……...92
Додаток Б Лістинг програми ……...96
Додаток Г Графічна частина ……..103

ВСТУП

Обґрунтування вибору теми дослідження.

Сучасний рівень розвитку технічного прогресу тісно пов`язаний із використанням різноманітних елементів штучного інтелекту, автоматичних систем у технологічному процесі. Одним із таких елементів є розпізнавання образів. Зростання інтересу до задач розпізнавання образів обумовлений необхідністю автоматизації процесів людської діяльності, пов’язаних з ідентифікацією об’єктів навколишнього світу, функціями контролю й управління складними динамічними об'єктами в реальному часі, образними процесами комунікації в інтелектуальних системах. Тому продовжується пошук і реалізація ефективних принципів передачі розпізнавальної функції людини комп'ютеризованими системами.

Один із перспективних напрямків вирішення даної проблеми ґрунтується на застосуванні штучних нейронних мереж і нейрокомп'ютерів як найбільш адекватних та ефективних методів розв’язання задач розпізнавання образів. Розроблена ціла низка нейромережевих парадигм для вирішення цих задач. Не зупиняються спроби практичного втілення напрацьованого знання у прикладних програмах і їх вдосконалення.

Загалом розвиток даної сфери досяг значного прогресу, призвівши до появи багатьох програмних рішень, які працюють з високою точністю. Однак, більшість цих рішень є неспеціалізованими рішеннями загального характеру. Одним із спеціалізованих напрямків розпізнавання зображень, на якому зосереджується ця робота, є оптичне розпізнавання тексту в зображеннях у веб-просторі.

Безупинне вивчення та вдосконалення виявлення і розпізнавання тексту мотивується сучасним станом і широкою сферою застосування цифрових мультимедіа. Постійно зростає кількість візуальної інформації, яку збирають, зберігають, доставляють та керують нею у цифрових формах. Використання пристроїв мультимедіа, планшетів, смартфонів тощо, які сьогодні широко поширені у повсякденному житті, призводить до появи нових задач.

В еру існування інформаційного суспільства, масової комп'ютеризації доступ до віртуального світу стає одним із найважливіших джерел інформації для будь-якої людини, зокрема і для людей із вадами зору. Згідно з даними Всесвітньої оранізації охорони здоров'я на 2015 рік близько 940 мільйонів людей у світі мали ту або іншу форму порушення зору. 246 мільйонів мали слабкий зір, а 39 мільйонів були сліпими [1].

Використовуючи програми для зчитування тексту, ці люди отримують життєво важливу інформацію, яку їм важко, а інколи і неможливо було б отримати в інший спосіб. У країнах Європи цій проблемі приділяється значна увага. Так, у багатьох із них обов'язковою умовою створення інформаційних веб-порталів є доступність цих ресурсів для людей з інвалідністю. Проте навіть із удосконаленням програмного забезпечення для зчитування текстів і доступності веб-сайтів залишається проблема зображень із текстовими елементами[2]. Не існує програм забезпечення доступності браузерів, які б допомагали розпізнавати текст у зображеннях на веб-сторінках і зачитували його користувачам. Тому, у цій роботі ми пропонуємо розробку бібліотеки JavaScript, яка, із використанням можливостей машинного навчання, виявлятиме текст у зображеннях, розпізнаватиме його та надаватиме користувачу у зрозумілій формі. Тобто, ми плануємо використати можливості інструментів розпізнавання текстів у зображеннях для вдосконалення користувацького досвіду людей із вадами зору.

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

Мета і задачі дослідження. Метою роботи є підвищення продуктивності розпізнавання тексту за підтримки й використання моделі машинного навчання, що здатна розпізнавати текст у зображеннях на веб-сторінках, і розробки бібліотеки JavaScript для застосування мережі у браузері.

Основними задачами дослідження є:

- проаналізувати наявні методи й засоби розпізнавання зображень, зокрема розпізнавання тексту в зображенні;

- розробити методи детекції тексту в зображенні та його розпізнавання;

- розробити програмні компоненти моделі машинного навчання, що здійснюватиме розпізнавання тексту в зображеннях на основі визначених методів;

- розробити програмні компоненти бібліотеки JavaScript для застосування моделі машинного навчання у браузері.

- провести експериментальне дослідження розроблених програмних засобів.

Об’єкт дослідження – процес розпізнавання тексту в зображеннях у браузері.

Предмет дослідження – Методи та програмні засоби розпізнавання зображень і розпізнавання тексту в зображенні.

Методи дослідження. У дослідженні використовувались:

- теорія комп'ютерного зору для детекції, тобто визначення наявності та розташування тексту;

- теорія статистичного навчання для розробки класифікатора;

- методи розпізнавання зображень і методи розпізнавання тексту для розпізнавання літер і слів у зображеннях;

- методи машинного навчання для створення моделі розпізнавання.

Наукова новизна отриманих результатів.

1. Подальшого розвитку отримав метод керованої детекції тексту в зображенні, в основі якого лежить поєднання двох алгоритмів детекції: MSER і салієнтності, що дозволило вдосконалити результати детекції тексту в зображеннях.

2. Подальшого розвитку отримав метод розпізнавання слів, який поєднує навчений класифікатор слів із виявленим сегментом слова для розпізнавання тексту в зображенні.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих у магістерській кваліфікаційній роботі теоретичних положень запропоновано алгоритми та розроблено програмні засоби розпізнавання тексту в зображеннях на веб-сторінках у браузері. Отримані результати можуть використовуватись для вдосконалення користувацького досвіду людей із вадами зору і розширення їх можливості отримувати інформацію у веб-просторі.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто: аналіз методів розпізнавання зображень, метод виявлення тексту в зображенні, метод розпізнавання тексту, комп'ютерна програма для розпізнавання тексту в зображеннях, застосування програми у браузері.

Структура та обсяг роботи. Магістерська кваліфікаційна роботи складається зі вступу, чотирьох розділів, висновків, списку літератури, що містить 33 найменувань, 3 додатків. Робота містить 28 ілюстрацій, 6 таблиці.

РОЗДІЛ 1
АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області

Протягом тривалого часу об’єктом вивчення були лише біологічні або психологічні аспекти розпізнавання графічних образів. При цьому їх дослідження зосереджувалося переважно на якісних характеристиках, які не дозволяли точно описати механізм функціонування візуального розпізнавання. Отримання функціональних залежностей було, як правило, пов'язане з дослідженням рецепторів органів слуху, дотику або зору. Однак принципи формування рішення залишалися загадкою. Існувала думка, що мозок функціонує за певними алгоритмами, а отже, з'ясувавши цю систему правил, можна її відтворити за допомогою обчислювальних і технічних засобів.

Заснована Норбертом Вінером на початку XX століття нова наука, яка отримала назву кібернетики (наука про загальні закономірності процесів управління та передачі інформації в машинах, живих організмах і суспільстві), дозволила запровадити кількісні методи в дослідження розпізнавання образів. Іншими словами, представити цей процес (по суті – природне явище) математичними методами.

Створення пристроїв, які виконують функції розпізнавання різних об'єктів, у деяких випадках забезпечує можливість заміни людини автоматизованими системами. Ця можливість визначається здатністю системи перейняти деякі найважливіші функції притаманні живому організму, однією з яких є і розпізнавання об’єктів.

Слід зазначити, що якість робіт, здійснюваних людиною на робочому місці, залежить від багатьох факторів (кваліфікації, досвіду, сумлінності тощо). У той же час справний автомат діє одноманітно і забезпечує завжди ту саму якість. Автоматичний контроль складних систем дозволяє вести моніторинг і

забезпечувати своєчасне обслуговування та ідентифікацію перешкод, підвищувити якість передачі інформації. Також зрозуміло, що використання автоматичних систем у низці завдань може забезпечити неможливу для людини швидкість.

Особливо важливою є автоматизація рутинних операцій, які при цьому вимагають деякого інтелектуального аналізу інформації. До таких операцій відносять розпізнавання образів.

Протягом останніх десятиріч проблема розпізнавання графічних образів привертала увагу фахівців у сфері статистики, прикладної математики, інформатики. Сучасні алгоритми теорії розпізнавання образів базуються на роботах С.А.Айвазяна, А.Я.Червоненкіса, В.Н.Вапніка, В.Н.Фоміна, І.Форджі, Д.Фукунагі, Дж.Хартігана, Дж.Хопфілда, Я.З.Ципкіна та ін. Багато сучасних системи розпізнавання образів засновані на принципах нейронних мереж (С.Хайкін, Ф.Уоссермен, А.В.Тімофеев та ін.) [3].

Перші кроки розпізнавання образів пройшли під знаком статистичного підходу, який і диктував основну проблематику. Важливою вехою у його розвитку стали роботи Р.Фішера. Виконані ще у 20-х роках XX ст, вони призвели до формування дискримінантного аналізу як одного з розділів теорії та практики розпізнавання. Аналіз дискримінантних функцій корисний для визначення, чи є безліч змінних ефективною для прогнозу належності до певної категорії. У 40-х роках А.Н.Колмогоровим і А.Я.Хінчіним поставлена задача про поділ суміші двох розподілів.

Дещо пізніше, у 50-60-ті роки XX століття на основі великої кількості робіт з'явилася теорія статистичних рішень. У її рамках будувалися алгоритми, що забезпечували на основі експериментальних вимірювань параметрів (ознак), що характеризують об'єкт, а також деяких апріорних даних, що описують класи, визначення конкретного класу, до якого може бути віднесений об'єкт [4].

Це стало початком планомірного наукового пошуку і практичних розробок. Поступово в рамках кібернетики формується новий науковий напрям, пов'язаний із

розробкою теоретичних основ і практичної реалізації пристроїв, а потім і систем, призначених для розпізнавання об'єктів, явищ, процесів. Нова наукова дисципліна отримала назву «Розпізнавання образів».

Всупереч прагненням статистиків розглядати розпізнавання образів виключно як розділ статистики, у практику та ідеологію розпізнавання входили зовсім інші ідеї. Одна з них була викликана дослідженнями в області розпізнавання зорових образів і заснована на наступній аналогії. У повсякденному житті люди постійно вирішують проблеми розпізнавання різних ситуацій, слухових і зорових образів. Звідси деякими дослідниками був зроблений висновок, що рішення цих проблем на електронно-обчислювальних машинах має у загальних рисах моделювати процеси людського мислення.

Найбільш відомою спробою підійти до проблеми з цього боку було знамените дослідження Ф.Розенблатта [5], який розробив модель навчання розпізнаванню зорових образів, названу їм персептроном. Учений виходив із припущення, що сприйняття здійснюється мережею нейронів. Підхід виявився успішним для вирішення низки завдань розпізнавання і породив цілий напрям досліджень алгоритмів навчання, заснованих на нейронних мережах, окремим випадком яких був персептрон.

На сьогодні створення штучних систем розпізнавання образів залишається складною теоретичною і технічною проблемою. Традиційно завдання розпізнавання образів включають в коло завдань штучного інтелекту.

Розглянувши коротко історію формування науки про розпізнавання образів, спробуємо тепер визначити, що саме являє собою це поняття. Розпізнавання образів (об'єктів) ми можемо охарактеризувати як задачу ідентифікації об'єкта або визначення будь-яких його властивостей по його зображенню (оптичне розпізнавання) або аудіозапису (акустичне розпізнавання) та іншим характеристикам (рис 1.1). Згадуючи статистичний підхід, можна дати й інше визначення. Розпізнавання образів — це віднесення вихідних даних до певного

класу за допомогою виділення істотних ознак, що характеризують ці дані, із загальної маси несуттєвих даних.

Центральне поняття – образ – класифікаційне угрупування в системі класифікації, яка об'єднує (виділяє) певну групу об'єктів за певною ознакою. Образи мають характерні властивості, які виявляються в тому, що ознайомлення з кінцевим числом явищ певної безлічі дає можливість виявляти яке завгодно велике число її представників.

Розпізнавання образів завжди передбачає наявність деякої системи обробки інформації, що має вхід і вихід. На вхід системи дані можуть надходити від безлічі різних джерел: фізичного об'єкта, що знаходиться в деякому середовищі, деякого процесу або як результат експерименту. Дані, що поступають на вхід системи розпізнавання, як правило, дуже складні і включають сигнали, які являються просторовими і/або часовими функціями. Вихідна інформація порівняно проста: вона зводиться до вказівки одного з декількох класів [6].

Методика віднесення елемента до образу називається вирішальним правилом. Ще одне важливе поняття – метрика – спосіб визначення відстані між елементами універсальної множини. Чим менший цей період, тим більш схожими є об'єкти (символи, звуки тощо), що ми намагаємось розпізнати. Зазвичай елементи задаються у вигляді набору чисел, а метрика -– у вигляді функції. Від вибору уявлення образів і реалізації метрики залежить ефективність програми, один алгоритм розпізнавання з різними метриками буде помилятись з різною частотою.

Загалом вибір метрики (або заходи близькості) є вузловим моментом дослідження, від якого вирішальним чином залежить остаточний варіант розбиття об'єктів на класи при заданому алгоритмі розбиття. У кожній конкретній задачі цей вибір має проводитися по-своєму.

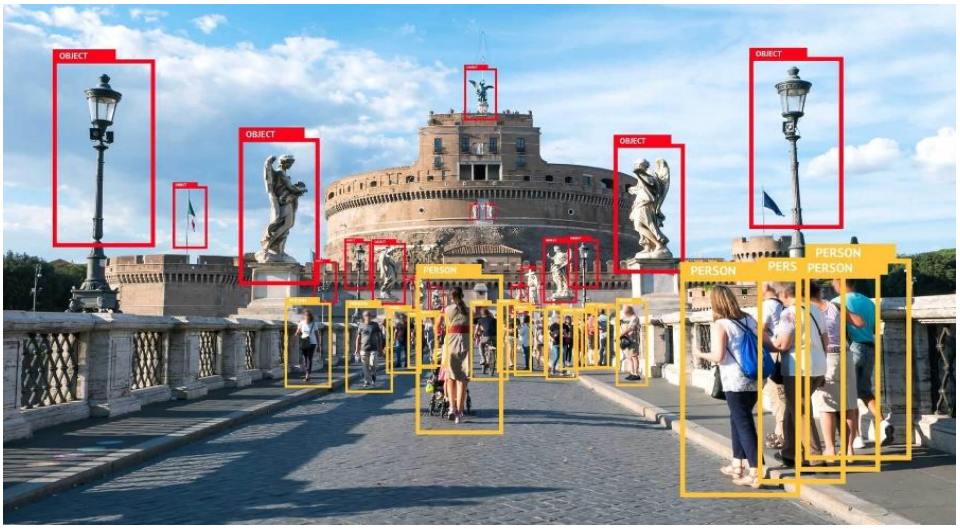


Рисунок 1.1 – Приклад розпізнавання графічних образів

Інше важливе поняття – адаптація, тобто процес зміни параметрів і структури системи, а можливо, і керуючих впливів, на основі поточної інформації з метою досягнення певного стану системи при початковій невизначеності і мінливих умовах роботи.

Навчання – це процес, внаслідок якого система поступово набуває здатності відповідати потрібними реакціями на певні сукупності зовнішніх впливів, а адаптація – це підстроювання параметрів і структури системи з метою досягнення необхідної якості управління в умовах безперервних змін зовнішніх умов.

Навчанням зазвичай називають процес вироблення в деякій системі тієї чи іншої реакції на групи зовнішніх ідентичних сигналів шляхом багаторазового впливу на систему зовнішнього корегування. Таке зовнішнє коригування у навчанні прийнято називати «заохоченнями» і «покараннями». Механізм генерації коригування практично повністю визначає алгоритм навчання. Самонавчання відрізняється від навчання тим, що тут додаткова інформація про вірність реакції системі не повідомляється.

1.2 Порівняльний аналіз програм розпізнавання зображень.

На сучасному IT-ринку існує доволі велика кількість різноманітних програмних засобів розпізнавання образів, які стали практичним втіленням попередніх науково-теоретичних розробок. Ці сервіси виконують широкий спектр функцій – від розпізнавання облич до класифікації об’єктів у зображеннях, від розпізнавання об’єктів у відео в реальному часі до розшифрування текстових елементів.

Найбільш помітними на сьогодні серед них є:

- Amazon Rekognition (рисунок 1.2) – сервіс, який дозволяє просто вбудовувати у додатки аналітику зображень і відео на основі глибокого навчання. Необхідно лише надати API Rekognition відповідне зображення або відео. Сервіс здатен розпізнавати об’єкти, людей, тексти, дії, а також виявляти неприйнятний контент. Крім того, Amazon Rekognition з високою точністю аналізує та розпізнає обличчя на зображеннях і відеоматеріалах клієнта. Шукати, аналізувати, порівнювати обличчя можна з метою перевірки користувачів, підрахування відвідувачів або забезпечення суспільної безпеки [7].

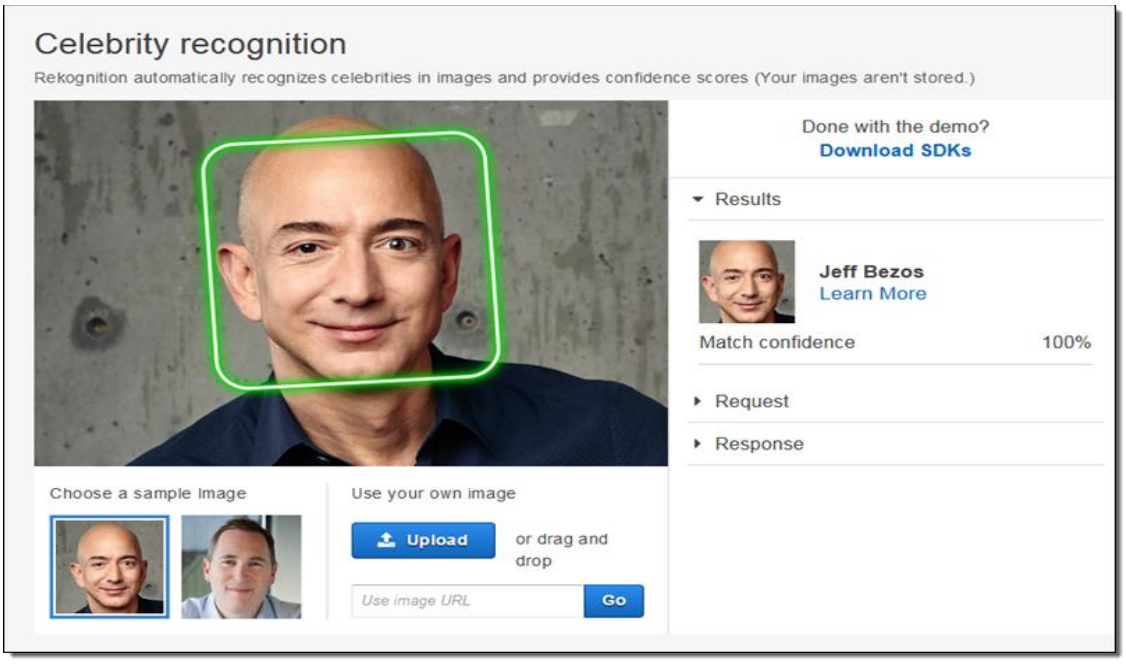


Рисунок 1.2 – Програма Amazon Rekognition

- OpenCV (рисунок 1.3) – бібліотека функцій та алгоритмів комп'ютерного зору, обробки зображень і чисельних алгоритмів загального призначення з відкритим кодом. Бібліотека надає засоби для обробки та аналізу вмісту зображень, зокрема розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей тощо), відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях [8].

1.3 – Програма OpenCV

- Google Cloud Vision API (рисунок 1.4) – потужний інструмент, який може стати основою для нескінченних можливостей застосування у поєднанні з бібліотеками мови Python. Google Cloud Vision API – це модель, яку навчили виявляти об’єкти та обличчя, виконувати задачі розпізнавання зображень, класифікації, маркування, а також розпізнавання текстів, надрукованих або рукописних, у зображеннях. Даний сервіс дозволяє розробникам програмних засобів легко інтегрувати вбудовані функції [9].

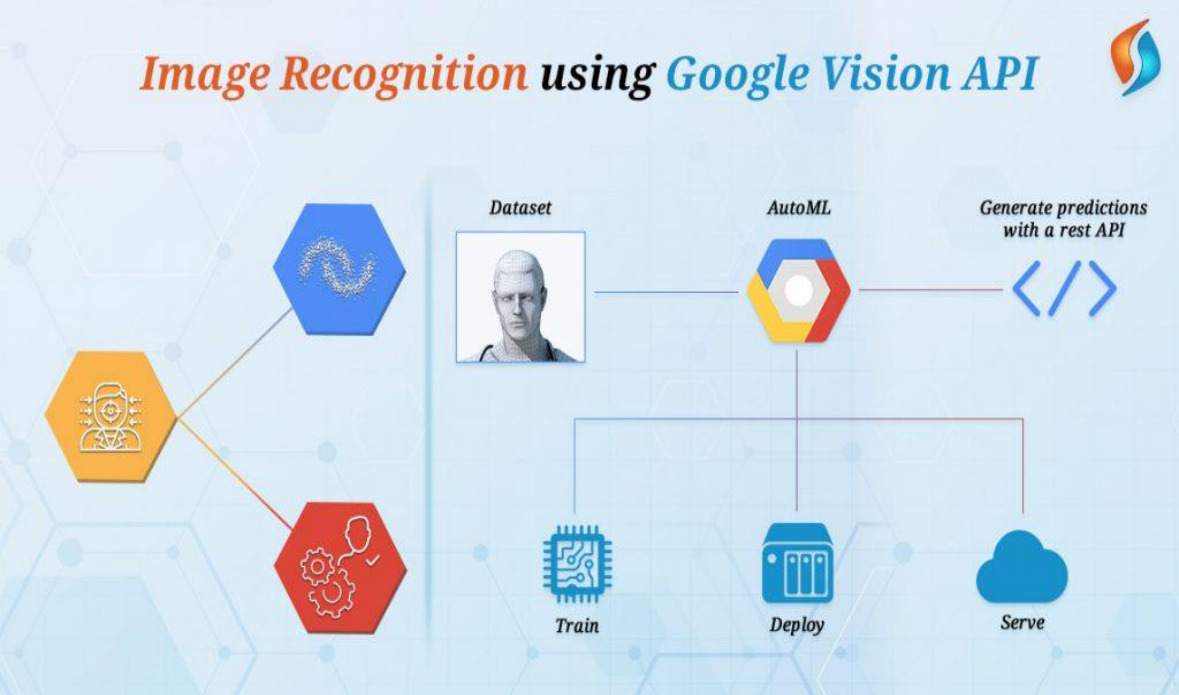


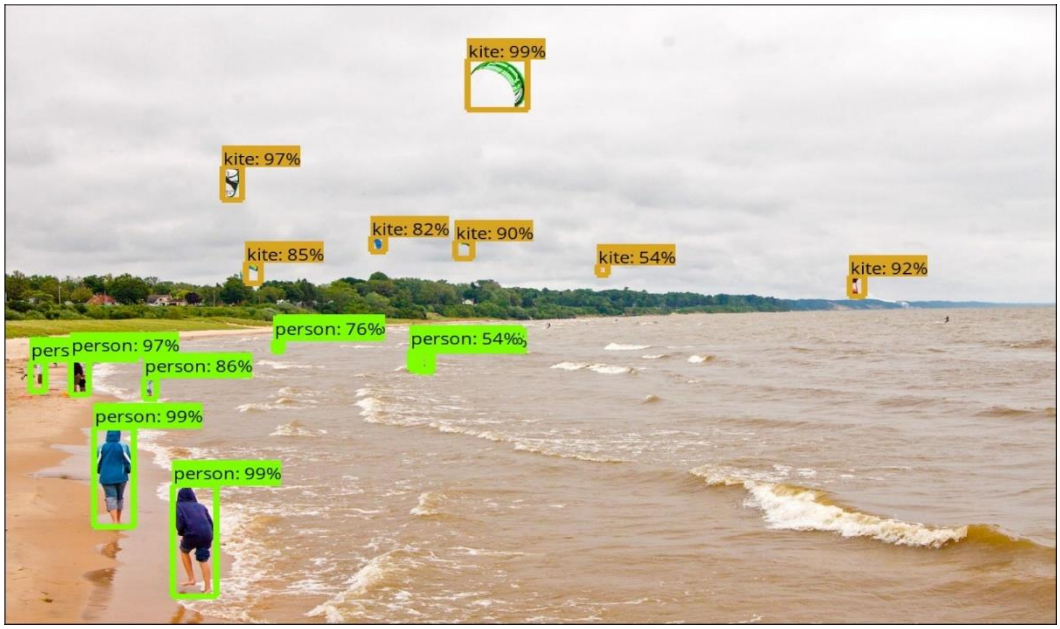
Рисунок 1.4 – Програма Google Vision API

- Watson – комп'ютерна система штучного інтелекту, що здатна відповідати на питання, задані природною мовою. Watson була створена в рамках проекту IBM DeepQA. При її розробці IBM застосовувала передові технології обробки природних мов, пошуку інформації, презентації знань, автоматизації формулювання логічних висновків, технології машинного навчання до галузі створення систем, здатних надавати відповіді на питання користувачів.

- Tesseract – програма для візуального розпізнавання текстів для різних операційних систем. Спочатку значна частина коду була написана мовою програмування C, але згодом було здійснено його доопрацювання для сумісності з C++ компіляторами. Це безкоштовне програмне забезпечення, випущене під ліцензією Apache; версія 2.0 та розробка спонсоровані Google з 2006 року. У 2006 році Tesseract вважався одним із найбільш точних наявних OCR з відкритим кодом.

- TensorFlow (рисунок 1.5) – відкрита програмна бібліотека для машинного навчання низці задач, розроблена компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявлення та розшифровування образів і кореляцій за аналогією з навчанням і мисленням, які

застосовують люди. Її наразі застосовують як для досліджень, так і розробки продуктів Google.



1.5 – Приклад роботи програми TensorFlow

Більшість цих програмних рішень є універсальними і пропонують загальні можливості машинного навчання. На сьогодні жодне з рішень не спеціалізується на наданні людям із вадами зору розширеного досвіду веб-перегляду. Ця робота орієнтується на розробку такого програмного рішення, яке із застосуванням загальних методів розпізнавання зображень, дозволить забезпечити подібний досвід. Від цього потенційно отримають користь розробники браузерів та програмного забезпення, орієнтованого на людей з особливими потребами, а також зазначена цільова категорія користувачів.

1.3 Аналіз методів вирішення поставленої задачи.

Існують різноманітні класифікації методів розпізнавання зображень. Одна з них виділяє наступні групи методів:

- структурний підхід застосовується до задач розпізнавання образів, у яких важлива інформація про структуру конкретного об'єкта. Алгоритм використовує тільки інформацію з відомими еталонними зразками.

- статистичні методи використовують похідну інформацію для вирішення задачі розпізнавання. Даний метод визначає чи відноситься об'єкт до певного класу на основі ймовірност чи ні.

- використання штучних нейронних мереж (згодом ми детальніше розглянемо цей метод).

Більш змістовною та глибокою нам видається інша класифікація. Згідно з нею методи розпізнавання графічних образів також можна розділити на три групи: попередня фільтрація і підготовка зображення, логічна обробка результатів фільтрації, алгоритми прийняття рішень на основі логічної обробки. Межі між групами умовні. Для вирішення завдання далеко не завжди потрібно застосовувати методи з усіх груп, буває достатньо двох, а іноді навіть одного.

Фільтрація. До цієї групи відносяться методи, які дозволяють виділити на зображеннях області без їх аналізу. Більшість цих методів застосовує єдине перетворення на всі точки зображення. На рівні фільтрації аналіз зображення не проводиться, але точки, які проходять фільтрацію, можна розглядати як області з особливими характеристиками.

Найпростіше перетворення – це бінаризація зображення по порогу. Для RGB зображення і зображення у градаціях сірого порогом є значення кольору. Зустрічаються ідеальні задачі, для вирішення яких достатньо такого перетворення. Припустимо, потрібно автоматично виділити предмети на білому аркуші паперу: вибір порога, за яким відбувається бінаризація, багато в чому визначає процес самої бінаризації. У даному випадку зображення було бінаризованим за середнім кольором. Зазвичай бінаризація здійснюється за допомогою алгоритму, який адаптивно обирає поріг.

У 1990-ті рр. для аналізу текстур напівтонових зображень були запропоновані локальні бінарні шаблони (ЛБШ). ЛБШ представляє собою опис околу пікселя зображення у двійковій формі. Оператор ЛБШ, що застосовується до пікселя зображення використовує вісім пікселів околу, приймаючи центральний піксел у якості порогу. Пікселі, які мають значення більші, ніж центральний піксель (чи дорівнюють йому), приймають значення «1», ті, які, менше центрального, приймають значення «0». Таким чином утворюється восьмирозрядний бінарний код, який описує окіл пікселя [10].

Методи розпізнавання, які використовують для виокремлення ознак ЛБШ та їх модифікації показують високі результати, як за швидкістю, так і за точністю розпізнавання. Бінаризація може давати дуже цікаві результати при роботі з гістограмами, зокрема в ситуаціях, де розглядаються зображення не в RGB, а в HSV. Наприклад, при сегментації кольорів. На цьому принципі можна побудувати, зокрема, детектор шкіри людини, на основі розрізнення «звичайних» пікселів зображення й пікселів шкіри [11].

Класична фільтрація: Фур'є, ФНЧ, ФВЧ. Класичні методи фільтрації з радіолокації та обробки сигналів можна з успіхом застосовувати у деяких завданнях із розпізнавання зображень. Традиційним методом у радіолокації, який майже не використовується в зображеннях у чистому вигляді, є перетворення Фур'є (конкретніше – БПФ). Одним із небагатьох винятків використання одновимірного перетворення Фур'є є компресія зображень. Для аналізу зображень одновимірного перетворення зазвичай не вистачає, потрібно використовувати більш ресурсомістке двовимірне перетворення.

Зазвичай набагато швидше і простіше використовувати згортку області з уже готовим фільтром, налаштованим на високі (ФВЧ) або низькі (ФНЧ) частоти. Такий метод не дозволяє зробити аналіз спектра, але для конкретного завдання інколи частіше потрібен не аналіз, а результат. І даний метод дозволяє його досягти.

Найпростіші приклади фільтрів, що реалізують підкреслення низьких частот (фільтр Калмана (рис. 1.6)) і високих частот (Фільтр Габора). Для кожної точки зображення обирається вікно і перемножується з фільтром того ж розміру. Результатом такої згортки є нове значення точки.

Інший метод, який слід згадати, – вейвлет аналіз. Вейвлет (від англ. Wavelet – невелика хвиля) – математична функція, яка дозволяє аналізувати різні частотні компоненти даних. Вейвлет аналіз, на відміну від аналізу Фур'є, спирається на спеціальні «малі хвилі», обмежені у часі (у випадку зображень – у просторі). Це дозволяє у вейвлет-представлені відразу мати і частотну і просторову інформацію. Вейвлет аналіз призначений, перш за все, для одночасного аналізу зображення в декількох масштабах, який отримав назву кратномасштабного аналізу [12].

Існує набір класичних функцій, застосовуваних у вейвлет-аналізі. До них відносяться вейвлет Хаара, вейвлет Морлі, вейвлет «мексиканський капелюх» тощо.

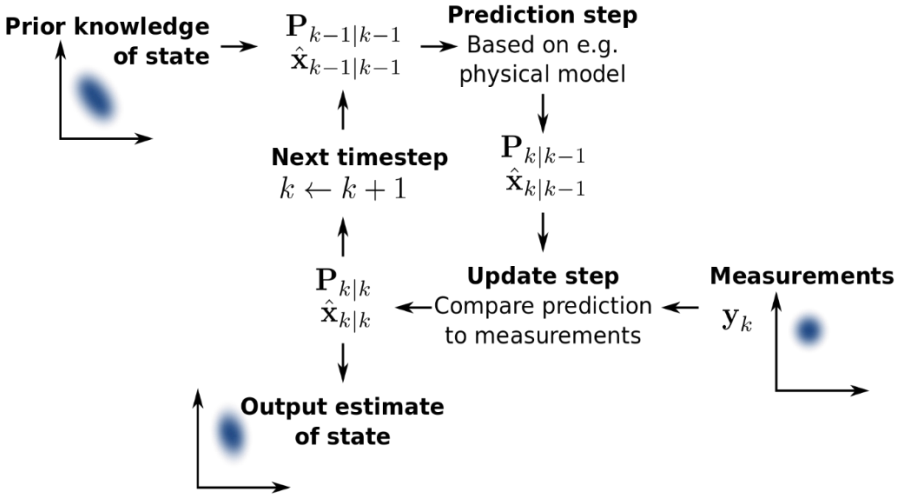


Рисунок 1.6 – Фільтр Калмана

Гарним прикладом використання розширеного трактування вейвлетів є завдача пошуку відблиску в оці, для якої вейвлетом є сам відблиск. Класичні вейвлети зазвичай використовуються для стиснення зображень або для їх класифікації.

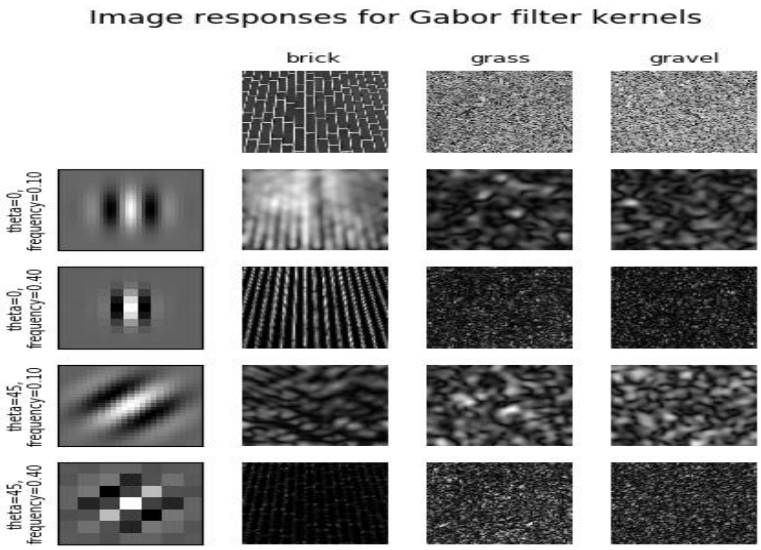


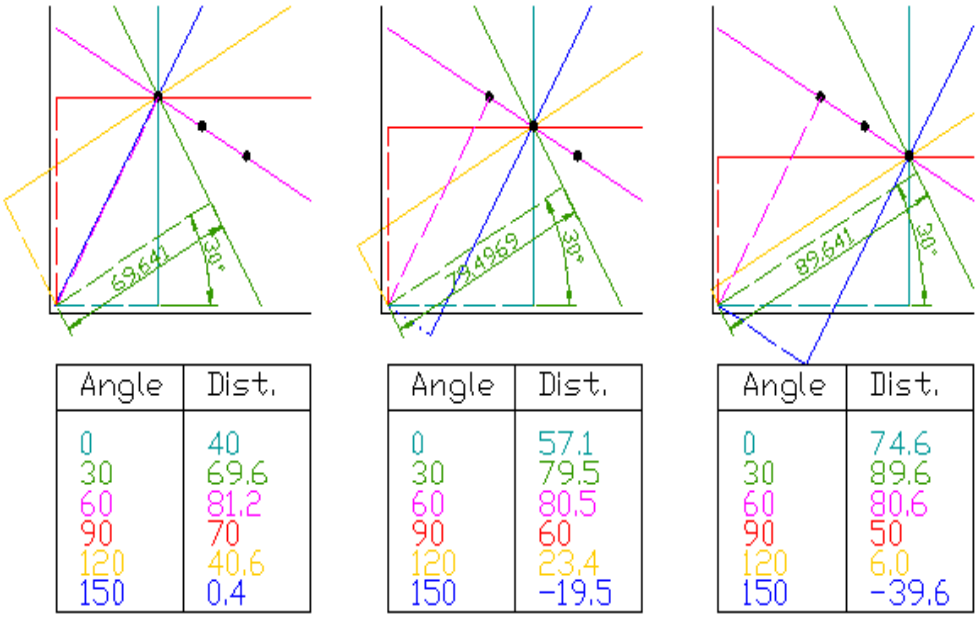
Рисунок 1.7 – Фільтр Гарбора

Розглядаючи вейвлет аналіз, варто згадати поняття кореляції, яке лежить у їх основі. При фільтрації зображень це незамінний інструмент. Кореляційними вважаються підходи, що базуються на побудові міри подібності між зображенням B та еталоном B0. Традиційна техніка порівняння зображення з еталоном грунтується на розгляді зображень як двовимірних функцій яскравості (дискретних двовимірних матриць інтенсивності). При цьому вимірюється або відстань між зображеннями, або міра їх близькості.

Кореляційні методи широко використовуються при виявленні та розпізнаванні зображень у системах навігації, стеження та промислових роботах [13] Класичне застосування – кореляція відеопотоку для виявлення зрушень або оптичних потоків. Найпростіший детектор зсуву також у певному сенсі є різницевим коррелятором. Там, де зображення не корелюють, можна вести мову про наявність руху.

Цікавим класом фільтрів є фільтрація функцій. Це суто математичні фільтри, які дозволяють виявляти прості математичні функції на зображенні (пряму, параболу, коло). Будується зображення, в якому для кожної точки вихідного зображення промальовується безліч функцій, які її породжують. Найбільш

класичним перетворенням є перетворення Хафа для прямих (рис. 1.8). У цьому перетворенні для кожної точки (x; y) вимальовується множина точок (a; b) прямої y = ax + b, для якої вірною є рівність.



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5

Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

Рисунок 1.8 – Перетворення Хафа для прямих

Перетворення Хафа дозволяє знаходити будь-які параметризовані функції, наприклад, окружності. Є модифіковане перетворення, яке дозволяє шукати будь-які фігури. Однак при обробці зображень воно, на жаль, працює далеко не завжди через надто повільну швидкість роботи, високу чутливість до якості бінаризації.

Аналогом перетворення Хафа для прямих є перетворення Радона. Воно обчислюється через БПФ, що дозволяє підвищити ефективність у ситуації, коли наявна дуже велика кількість точок. До того ж його можна застосовувати до небінаризованих зображень.

Окремий клас фільтрів – фільтрація кордонів і контурів. Контури є корисними, коли ми переходимо від роботи із зображенням до роботи з об'єктами на цьому зображенні. Коли об'єкт досить складний, але добре виділяється, то часто єдиним способом роботи з ним є виділення його контурів. Існує низка алгоритмів, які вирішують задачу фільтрації контурів.

Один із них – оператор Собеля – дискретний диференціальний оператор, який обчислює наближене значення градієнта яскравості зображення. Також варто згадати оператор Лапласа – оператор виділення країв, заснований на обчисленні симетричних кругових похідних.

Найчастіше використовується оператор (алгоритм) Кенні. Основні етапи алгоритму:

- розмиття зображення для видалення шуму;

- пошук градієнтів. Кордони відмічають там, де градієнт набуває найбільшого значення. Вони можуть мати різні напрямки, тому алгоритм Кенні використовує чотири фільтри для визначення горизонтальних, вертикальних і діагональних ребер у розмитому зображенні [14].

- пошук локальних максимумів;

- подвійна порогова фільтрація.

Хоча розробка алгоритму проводилася ще на зорі формування теорії комп'ютерного зору, детектор кордонів Кенні на сьогодні залишається одним із кращих.

Наступна група методів розпізнавання зображень – логічна обробка результатів фільтрації. Фільтрація дає набір придатних для обробки даних. Але часто ці дані можна використовувати без їх обробки. Наведемо декілька класичних методів, які дозволяють перейти від зображення до властивостей об'єктів або до самих об'єктів.

Переходом від фільтрації до логіки є методи математичної морфології. Фактично це найпростіші операції нарощування та ерозії бінарних зображень. Вони дозволяють прибрати шуми з бінарного зображення, збільшивши або зменшивши наявні елементи. На базі математичної морфології створені алгоритми оконтурювання, але зазвичай використовуються гібридні алгоритми або алгоритми у зв'язці.

Вище було згадано алгоритми отримання кордонів. Кордони досить просто перетворюються на контури. Із застосуванням алгоритму Кенні це відбувається автоматично, однак для інших алгоритмів потрібна додаткова бінаризація.

Контур є унікальною характеристикою об'єкта. Для систем ідентифікації, які розпізнають об’єкти на цифровому зображенні, найбільш корисною інформацією є відомості саме про контури, тобто про лінії, що проходять на границях однорідних областей. Вважається, що такими областями є об’єкти, для яких різниця яскравостей будь-яких двох елементів зображення (пікселів, групи пікселів), не перевищує певного порогу. Тому, по завершенні попередньої обробки зображення, система розпізнавання, в першу чергу, робить пошук контурів зображення [15].

Для виконання задачі визначення контурів був розроблений потужний математичний апарат, що отримав назву контурного аналізу. Недоліком методу контурного аналізу є необхідність його проведення у певних «ідеальних умовах».

Інший метод – метод особливих точок. Особливі точки – це унікальні характеристики об'єкта, які дозволяють порівнювати об'єкт із собою або зі схожими класами об'єктів. Метод базується на визначенні множини таких точок (ОТ) та їх описові у вигляді числового чи бінарного вектора – дескриптора, що відображає вміст множини локальних околиць ОТ зображення.

Значення дескриптора, як правило, інваріантне стосовно групи геометричних перетворень об’єктів на зображенні (зміщення, поворот, масштабування), а кількість утворених дескрипторів, що формують опис, повинна бути достатньою для прийняття результативного рішення відносно розрізнення розпізнаваних об’єктів [16].

Розглянемо спочатку методи, які допомагають знайти особливі точки, що не є стабільними, але які можна швидко розрахувати.

Перший клас. Особливі точки, які є стабільними протягом секунд. Такі точки служать для того, щоб вести об'єкт між сусідніми кадрами відео, або для зведення зображення з сусідніх камер. До таких точок можна віднести локальні максимуми

зображення, кути на зображенні, точки, в яких досягаються максимуми дисперсії, певні градієнти.

Другий клас. Особливі точки, які є стабільними при зміні освітлення і невеликих рухах об'єкта. Такі точки служать передусім для навчання і подальшої класифікації типів об'єктів. Наприклад, класифікатор пішохода або класифікатор особи – це системи, побудовані саме на таких точках. Деякі з раніше згаданих вейвлетів можуть стати базою для таких точок. Наприклад, примітиви Хаара, пошук відблисків, пошук інших специфічних функцій. До таких точок відносяться точки, знайдені методом гістограм спрямованих градієнтів (рис 1.9).

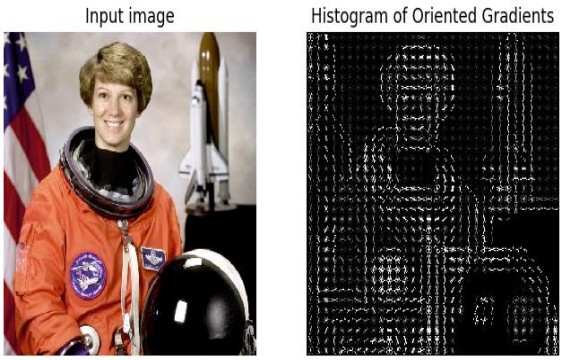


Рисунок 1.9 – Гістограм спрямованих градієнтів

Третій клас – стабільні точки. Серед методів, які дають повну стабільність, варто згадати SURF і SIFT. Вони дозволяють знаходити особливі точки навіть при повороті зображення. Розрахунок здійснюється довше порівняно з іншими методами, але досить обмежений час. Наприклад, метод SURF ефективний у пошуку об’єктів на зображенні, що мають велику кількість критичних точок. Критичні точки визначаються на переходах змін кольору геометричних форм. Метод SIFT дозволяє порівнювати зображення, піддані таким трансформаціям, як зміна масштабу, зміщення об'єкта на сцені, повороти камери або об'єкта. Алгоритм SIFT працює з чорно-білими зображеннями [17].

Третя група методів розпізнавання образів – методи навчання. Вони не працюють безпосередньо з зображеннями, але дозволяють приймати рішення. Мова йде про методи машинного навчання і прийняття рішень.

Методи машинного навчання поділяють на дві основні категорії: навчання з учителем (supervised learning) та навчання без учителя (unsupervised learning). Методи навчання з учителем поділяють вхідні дані на набір наперед заданих класів. Для навчання такого класифікатора потрібна навчальна вибірка, яка містить марковані зразки різних класів. Методи навчання без учителя не потребують навчальних даних, проте вони не ставлять у відповідність вхідним даним певний клас, а лише вивчають закономірності у вхідних даних та поділяють вхідні дані на схожі між собою групи (кластери) [18].

Якщо вести мову про класифікатори безпосередньо в області розпізнавання зображень, то по суті їх мета – виділити у просторі ознак області, які є характерними для об'єктів класифікації. Існує ціла низка класифікаторів. Кожен із них краще налаштований на «свою» сферу. Важливо вміти підібрати потрібний для вирішення конкретної задачі класифікатор.

Класифікатори можна поділити на параметричні та непараметричні. До параметричних належать, наприклад, метод максимальної правдоподібності (maximum likelihood), оскільки він працює на припущенні, що функція щільності ймовірності для кожного з класів подається гаусовим розподілом [19]. Непараметричні класифікатори, у свою чергу, не ґрунтуються на жодних припущеннях про розподіл вхідних даних. Ураховуючи той факт, що в більшості випадків функція розподілу невідома, непараметричні класифікатори набули значно більшого поширення.

Важливою властивістю класифікаторів є можливість не лише належності вхідних даних до певного класу (виходу класифікатора), а і визначення ймовірності належності до кожного з класів, на основі якої легко обрати найбільш

достовірний клас. Таку особливість має, наприклад, логістична регресія (logistic regression).

Найбільш поширеними методами машинного навчання для задач класифікації [20] є штучні нейронні мережі (artificial neural network), логістична регресія, метод опорних векторів Support Vector Machine (SVM) та random forest.

Штучна нейронна мережа — математична модель, а також її програмна або апаратна реалізація, побудована за принципом організації та функціонування біологічних нейронних мереж – мереж нервових клітин живого організму. Проблема навчання багатошарових нейронних мереж була вирішена в середині 1980-х років методом зворотного поширення помилки (backpropagation) [21]. Це ітеративний градієнтний алгоритм, який використовується з метою мінімізації помилки роботи нейронної мережі, що забезпечує отримання бажаного виходу.

Для визначення якості роботи нейронної мережі використовують функцію втрат (loss function). Зазвичай за таку функцію обирають евклідову відстань, середньоквадратичну похибку або функцію кросентропії [22]. Мережа вважається навченою, якщо функція втрат набуває мінімального значення. Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів у напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи мережі. Процедуру зворотного поширення помилки можна застосувати кілька разів, щоб поширити градієнти через усі шари, починаючи з виходу (результату прямого проходження нейронної мережі) і до входів, що подаються в мережу. У процесі навчання нейронної мережі ваги зв’язків між нейронами коригуються на основі методу градієнтного спуску (gradient descent). На практиці зазвичай використовують модифікацію цього методу, коли процедура градієнтного спуску застосовується до груп навчальних прикладів. Такий підхід називається методом стохастичного градієнта (stochastic gradient descent), що значно пришвидшує час навчання нейронної мережі.

Перевагами нейронної мережі як класифікатора є її непараметричність — нейронна мережа не потребує жодних попередніх знань про розподіл вхідних даних та можливості їх поділу. Вона швидко адаптується до нових даних. Нейронні мережі стійкіші, ніж інші статистичні методи при розпізнаванні зображень, якщо зображення входу має шуми. Тому у цій роботі для забезпечення розпізнавання текстових елементів у зображеннях будуть застосовані саме нейронні мережі.

Для різних задач застосовуються різні види і типи нейронних мереж, серед яких можна виділити: згорткові нейронні мережі, рекурентні нейронні мережі, нейронну мережу Хопфілда. Згорткові мережі є одними з найбільш популярних типів штучних нейронних мереж. Вони довели свою ефективність у розпізнаванні візуальних образів (відео та зображення), рекомендаційних системах і обробці мови.

Згорткові нейронні мережі видаються найбільш оптимальним методом для досягнення поставлених у даній роботі задач, адже вони відмінно масштабуються і можуть використовуватися для розпізнавання образів якого завгодно великого масштабу. Методи попіксельної класифікації часто потерпають від наявності шуму. Тому для класифікації важливо розглядати не лише спектральні характеристики окремих пікселів, а і просторові, що й робить згорткова нейронна мережа.

1.4 Постановка задачі

Додаток «AI Access» є реалізацією можливостей згорткових нейронних мереж із ефективного розпізнавання та класифікації патернів у зображеннях у комбінації з потенціалом веб-простору для вдосконалення користувацького досвіду людей із вадами зору.

Даний додаток включатиме наступні функціональні елементи:

- тренована модель штучного інтелекту, здатна розпізнавати текстові патерни у зображеннях;

- ефективний та зручний веб-інтерфейс виявлення зображень у веб-сторінках та їх заміни текстом;
- зручний для розробників API, який можна буде легко інтегрувати у програмне забезпечення.

Реалізація подібної системи вимагає створення та тренування згорткової нейронної системи, а також розробку бібліотеки Javascript для втілення можливостей даної моделі штучного інтелекта у веб-просторі.

Таким чином, ми можемо визначити основні задачі роботи:

1. Дослідження предметної області розпізнавання зображень;
2. Висвітлення наявних підходів, рішень та алгоритмів розв'язання проблеми розпізнавання зображень.
3. Створення моделі машинного навчання.
4. Реалізація обраних алгоритмів.
5. Тренування нейронної мережі, здатної розпізнавати текстові елементи у зображеннях.
6. Створення бібліотеки Javascript для використання розробленого AI у веб-просторі.
7. Тестування програмного забезпечення.

1.5 Висновки

У даному розділі досліджено предметну область розпізнавання зображень. Висвітлено розвиток основних науково-теоретичних розробок та найбільш популярних й ефективних практичних рішень у зазначеній сфері. Зроблено висновок щодо надто загального універсального характеру більшості наявних продуктів та неможливості їх застосування до специфічних випадків, пов'язаних із цільовою категорією користувачів. На основі аналізу найбільш популярних методів розпізнавання зображень згорткові нейронні мережі було обрано як метод, застосовуваний у цій роботі. Розроблено план подальшої роботи, у якому

виокремлено низку функцій, які має виконувати програмний продукт, та окреслено подальші кроки для його реалізації.

РОЗДІЛ 2
ПРОЕКТУВАННЯ МЕТОДУ РОЗПІЗНАВАННЯ ТЕКСТУ В ЗОБРАЖЕННЯХ

2.1 Загальна структура системи

Структура роботи програми має нагадувати наступну схему (рис.2.1).

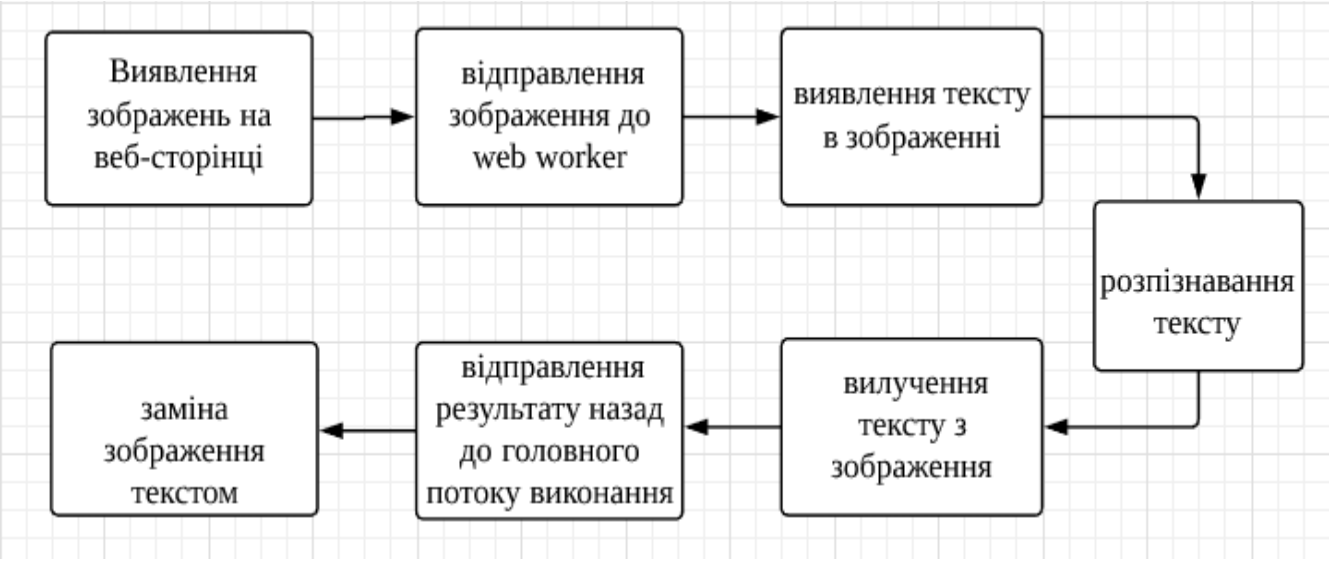


Рисунок 2.6 – Діаграма станів для процесу робота програми

Після окреслення загальної структури робочого процесу програми, першим кроком у її реалізації має бути створення підготовленої робочої моделі нейронної мережі, здатної здійснювати детекцію, розпізнавати та виокремлювати тексти із зображень. Для реалізації цієї задачі спочатку необхідно розробити алгоритм дій, якому буде слідувати нейронна мережа у процесі розпізнавання.

2.2 Методи детекції символів і слів

У цій роботі порядок дій будується на підході «знизу вгору». Першим кроком має бути визначення областей можливого розташування символів – літер. Потім відбуватиметься комбінування цих областей на основі певних визначених критеріїв і згодом здійснюватиметься детекція слів.

В основі реалізації даного кроку лежить поєднання двох визнаних методів детекції з метою досягнення вищої точності та ефективності виявлення тексту в зображеннях, а саме методів MSER і салієнтності.

Метод MSER (Maximally Stable Extremal Region) є інструментом пошуку відповідностей між двома варіантами одного зображення із різною перспективою з метою знаходження стерео-відповідності. Цей алгоритм встановлює порогові значення для сірого кольору вхідного зображення і з'єднує визначені зони в дерево компонентів із просторовим перекриттям. Алгоритм здатен виявляти гомогенні світлі зони на темному фоні так само, як і гомогенні темні зони на світлому фоні. Ця здатність є важливою для розробки методу детекції текстів у зображеннях, адже текст може бути знайденим в обох випадках [23].

Вищезгаданий алгоритм поєднується із альтернативним методом детекції тексту – салієнтності. Цей алгоритм заснований на показниках салієнтності кольорів і кривизни, оскільки текст у зображеннях має зазвичай відносно рівномірне забарвлення, що разом із викривленням відрізняє його від фону, на якому він розташований.

2.2.1 Метод детекції символів і слів

Згідно з означеним вище підходом розпочнемо з детекції символів – літер.

Для виявлення символів ми вилучаємо з зображення розмаїття кольорових каналів та обробляємо ці дані за допомогою методу MSER (Maximally stable extremal regions) та алгоритму салієнтності. Різні кольорові канали допомагають у виявленні літер у відмінних умовах освітлення. Хоча метод використовує різноманітність кольорових каналів, процеси є незалежними, тому всі розрахунки відбуваються паралельно.

Бінаризований результат застосування алгоритмів виявлення тексту, заснованих на методі MSER та салієнтності, надає нам пов'язані компоненти, які потенційно є областями розташування літер. Завдяки особливостям освітлення

кожного зображення літери, які не були виявлені одним кольоровим каналом, можуть бути знайдені іншим.

Наприклад, після обробки каналу G на зображенні, наведеному нижче, за допомогою алгоритму MSER, відсутніми є літери у слові FIRE. Це сталося через певні умови освітлення (Рис. 2.2). Однак використання каналу S алгоритмом MSER дозволяє виявити ці символи (Рис. 2.3), адже кольоровий канал S є інваріантним (незмінюваним) відносно даних умов освітлення.

Рис. 2.2. Приклад виявлення символів за допомогою каналу G

Рис. 2.3. Приклад виявлення символів за допомогою каналу S

2.2.2 Метод формування слів

Вхідною інформацією для реалізації наступного кроку -– формування слів – є результат попередньої стадії, а саме визначені потенційні області розташування символів-літер. Однак це не все. Також необхідно здійснити певну попередню обробку даних для того, щоб зменшити ці області без зниження точності детекції символів. Критерії комбінування засновані на припущенні, що літери у слові підпорядковані певним загальним правилам: аналогічні виміри, схожі відстані між символами у слові тощо. Після цього використовується вичерпний спосіб вилучення зон розташування слів. Нарешті, застосовується фільтрування, метою якого є зменшення долі неправильно визначених областей розташування слів у результаті, наданому програмою.

Через те, що на попередній стадії відбувається обробка різних кольорових каналів одного зображення, ми отримуємо кілька потенційних областей розташування слів для тієї ж самої зони зображення. Більше того, хоча ми й здатні виявити більшість літер, які є у зображенні, значна частина отриманих на цій стадії областей є фактично «шумом». З метою зменшення кількості таких областей без зменшення точності визначення символів застосовуються наступні методи фільтрування.

Спочатку фільтрування здійснюватиметься на основі попередніх знань, зокрема, видалення контрасту та співвідношення сторін. На стадії детекції літер можна помітити, що їх частини також інколи визначаються як літери. Головною причиною такого результату є переважно умови освітлення навколишнього середовища, такі як світові ефекти на блискучих рівних поверхнях.

Наприклад, обробка кольорового каналу H, після якої дані піддаються дії алгоритму салієнтності, на зображенні з єдиною літерою на блискучій поверхні може дати на виході близько тисячі потенційних зон розташування літер. Це можна побачити на рис. 2.4, де визначено 1146 таких зон. Для того, щоб зменшити кількість неправильно визначених областей зображення і при цьому не загубити

виявлену літеру, потрібно усунути зони, які не змінюються в загальному контрасті, якщо розширюється їх площа.

Крім того, ми застосовуємо видалення дублікатів виявлених зон. Як дублікати визначаються ті потенційні області розташування літер, де перекриття становить понад 95%. Зрештою, ми видаляємо аномально витягнуті зони, визначені алгоритмом.

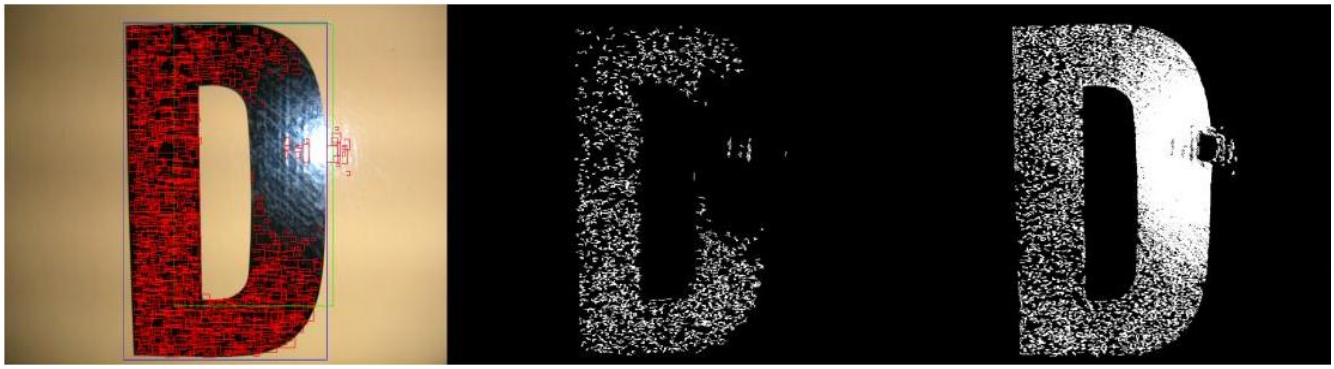


Рис 2.4. Приклад виявлення літери

Наступна задача – поєднання зон детекції символів-літер для формування слів. Цей процес здійснюватиметься на основі певних правил-критеріїв. Найчастіше літери в одному слові мають схожу висоту із сусідніми літерами. Для того, щоб визначити цю зону близькості ми використовуватиме термін «тестовий радіус». Тестовий радіус – це максимальний коефіцієнт, на який необхідно помножити виміри (висота/ширина) зони однієї літери, щоб отримати відстань до іншої літери у слові.

Ми надаємо перевагу цій динамічній мірі для визначення навколишньої області, оскільки відстань між символами залежить від розміру шрифту, і тому вона не може бути представленою абсолютним числом пікселів. Схожим чином ми запроваджуємо термін «тестовий нахил», під яким розуміється нахил між центроїдами двох послідовних символів у слові.

Ці два фактори, а саме тестовий нахил і тестовий радіус, визначають зону пошуку для кожної із потенційних областей розташування літер (рис. 2.5 зліва).

Якщо потенційна область відповідає вищезгаданим обмеженням, можна зробити висновок про зв’язок між двома такими областями (рис. 2.5 (справа)).

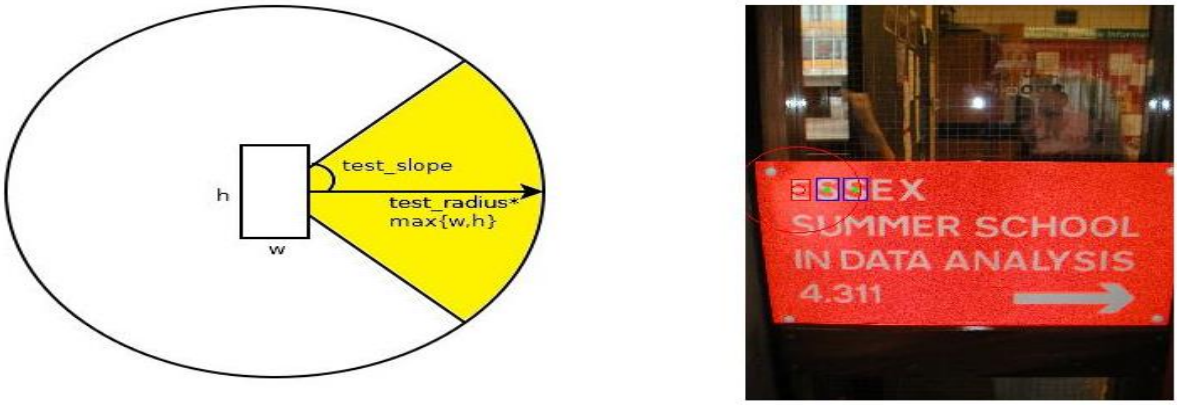


Рис. 2.5. Зона пошуку (зліва) і зона зв’язку (справа)

Якщо представити області потенційного розташування літер як вузли, а зв’язки між ними як ребра, ми отримуємо спрямований (орієнтований) ациклічний граф. Назвемо його графом з’єднання(зв’язку). Зазвичай на цій стадії кінець одного слова з’єднується з початком наступного в одному текстовому рядку. Наприклад, великі кластери на рис.2.6 є фактично другим і третім рядком у зображенні.

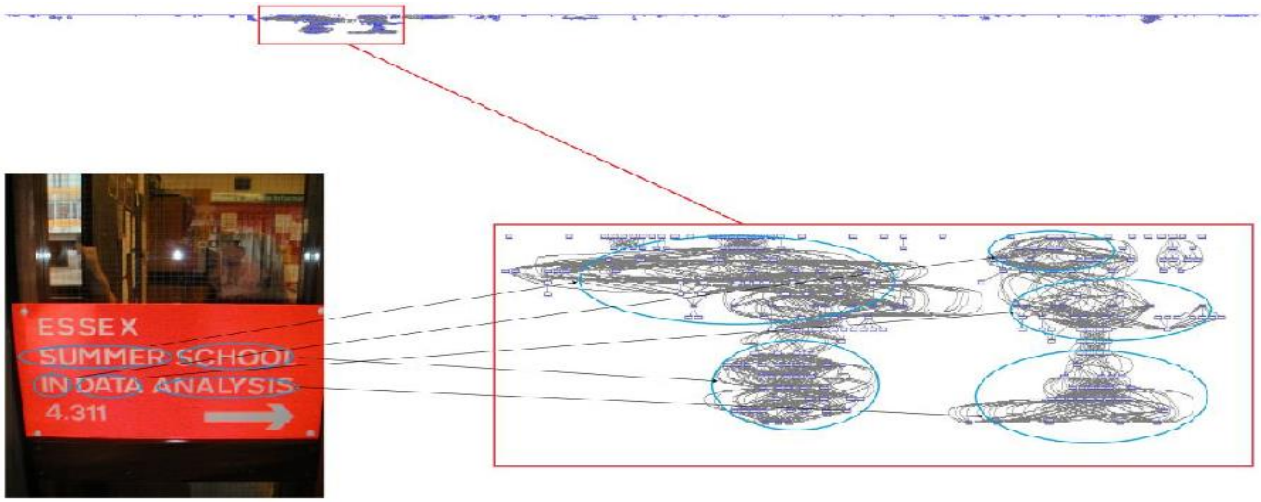


Рис. 2.6. Аналіз графу з'єднання

Для того, щоб отримати найбільш ймовірну область розташування слова, ми маємо відібрати інформацію про всі можливі шляхи з графу з'єднання.

Для отримання всіх можливих шляхів із кореневого вузла, необхідно застосувати алгоритм пошуку в глибину, допоки ми не досягнемо листового вузла. Як результат, послідовність усіх відвіданих вузлів надається як потенційний шлях слова, й останній зв’язок розривається.

Процедура повторюється, допоки у графі з’єднання не залишиться лише кореневий вузел. Коли це відбувається, ми продовжуємо, розглядаючи першу «дитину» як корінь і відбираючи усі шляхи із цього підграфа.

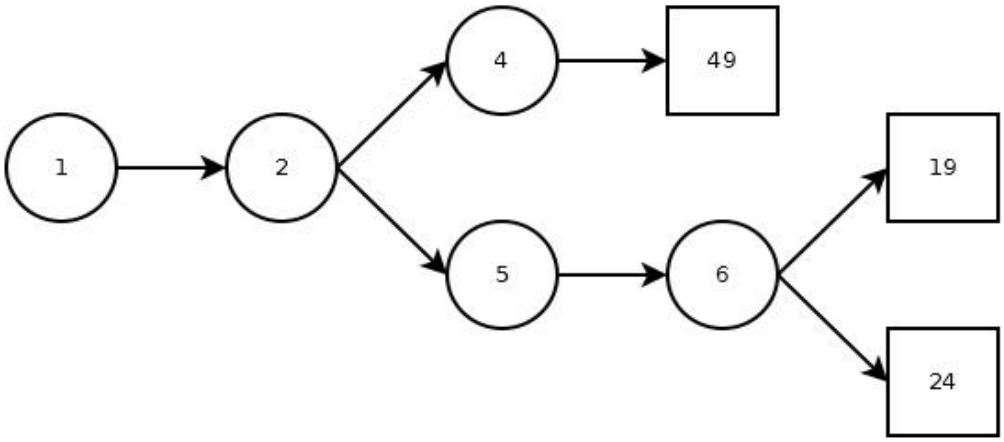


Рис. 2.7. Приклад графу з’єднання

Результатом даного процесу є значна кількість потенційних зон розташування слів. Ми спробуємо виключити більшість із них на наступній стадії. Цей підхід відбору шляхів націлений на отримання якомога більшого їх числа, беручи до уваги початок і кінець шляху та не опікуючись послідовністю вузлів.

Наприклад, якщо у попередньому випадку існував зв’язок між вузлом 4 і 6, то згідно із запропонованим підходом ми не зможемо вилучити шляхи [1 2 4 6 19] і [1 2 4 6 24].

Останнім кроком у визначенні потенційної області розташування слова є її післяобробка. Метою даного кроку є виключення із результату всіх помилково визначених на попередній стадії потенційних зон. Знову необхідно застосувати засноване на певному попередньому знанні фільтрування.

Здебільшого слова мають горизонтальне позиціювання, тому можна виключити зони із діагональною орієнтацією. Крім того, існує певний поріг максимальної довжини шляху слова. Будь-який шлях, який перевищує заданий поріг, вважається «шумом» і виключається. Через те, що ми застосовуємо вичерпний підхід до з'єднання областей із літерами, значну кількість шляхів, які займають ту саму зону, ми відкидаємо, залишаючи один.

Застосовувана у цій роботі методика визначення тексту дещо схожа із підходом Х. Чена [24], однак із використанням більшого числа алгоритмів і на різних кольорових каналах.

2.3 Метод розпізнавання символів

Для розпізнавання літер буде застосовану класифікацію зображення літер, яка передбачає існування 62 можливих класів: 26 класів для заголовних літер, 26 – рядкових літер (нижній регистр) і 10 класів для чисел (у цій роботі детекція і розпізнавання здійснюватиметься щодо літер англійської мови як однієї з найбільш поширених у веб-просторі).

Для вирішення задачі розпізнавання використовується простий, але ефективний підхід. По-перше, застосовується щільна сітка без перекриття на вхідному зображенні (рис. 2.8 зліва), яка розділяє зображення на клітини рядків * клітини стовпців. Потім для кожної клітини сітки визначаються характеристики гістограми напрямлених градієнтів (HOG – histogram of oriented gradients) (рис. 2.8 (справа)). Нарешті, ми об'єднуємо ці характеристики в один дескриптор, за допомогою якого навчатиметься модель нейронної мережі – мультикласовий класифікатор SVM.

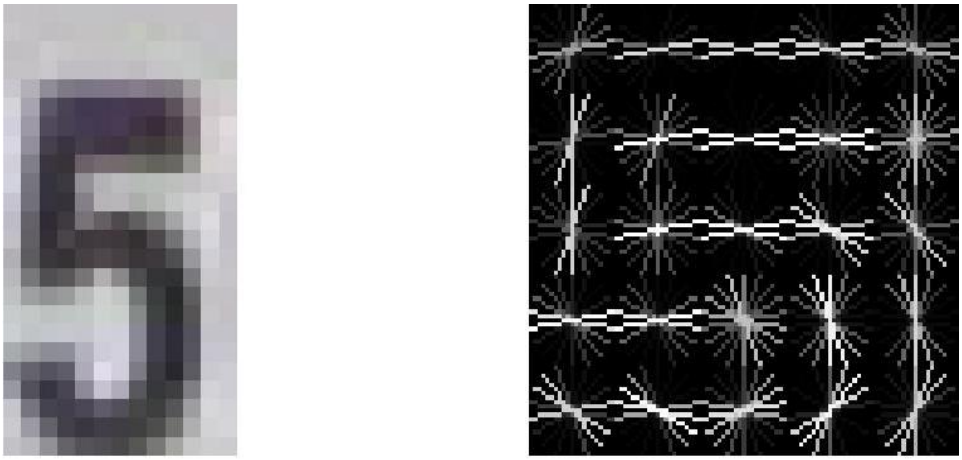


Рис. 2.8. Зразок зображення літери (зліва) і характеристики HOG у сітці з вимірами 5x5 (справа)

Особливість латинської абетки полягає в тому, що деякі літери у верхньому та нижньому регістрах мають аналогічну морфологію. Тому можна об'єднати символи, представлені в таблиці 2.1, оскільки розпізнавання як заголовних літер, так і рядкових жодним чином не впливає на загальне написання слова, яке містить цей символ.

Таблиця 2.1. Об’єднані класи літер

Об’єднані класи	
C	c
I	i
J	j
K	k
O	o
P	p
S	s
U	u
V	v
W	w
X	x
Y	y
Z	z

Водночас морфологія нуля (0) є подібною до літери «О» у верхньому регістрі та «о» в нижньому, так само як рядкова літера «l» нагадує одиницю (1), а також заголовну «І» і рядкову «і», однак помилкове розпізнавання цих символів може спотворити написання слів, тому в цьому випадку ми уникаємо об’єднання класів.

Наступним кроком розпізнавання є визначення найбільш підходящого розміру вищезгаданої сітки. Інтуїтивно ми перевіряємо сітки з рівним числом рядків і столпців. Крім того, ми пробуємо сітки, де кількість рядків перевищує кількість стовпців і навпаки. Нашою задачею на цьому етапі є підбір найбільш вдалої комбінації.

2.4 Методи сегментації і розпізнавання слів

Останній крок – розпізнавання тексту. Іншими словами, розпізнавання слова, враховуючи його потенційну площу і використовуючи розпізнавання літер, описане в попередньому розділі. Для цього застосовується декомпозиція зображення слова – його розділення на підзображення/символьні одиниці.

2.4.1 Методи сегментації слів

Ефективним шляхом розділення слова на сегменти – символьні одиниці (літери) – є отримання вертикального профілю. Для цього вертикально сумуються значення рівня сірого в зображенні слова. Крайні точки згладженого вертикального профілю є гарними показниками сегментів літер: локальні мінімуми (найнижчі точки) вказують на сегменти світлого текста на темному тлі (рис. 2.8), у той час як локальні максимуми є індикаторами сегментів темного текста на світлому тлі (рис. 2.9).

Рис. 2.8. Сегментація світлого тексту на темному тлі

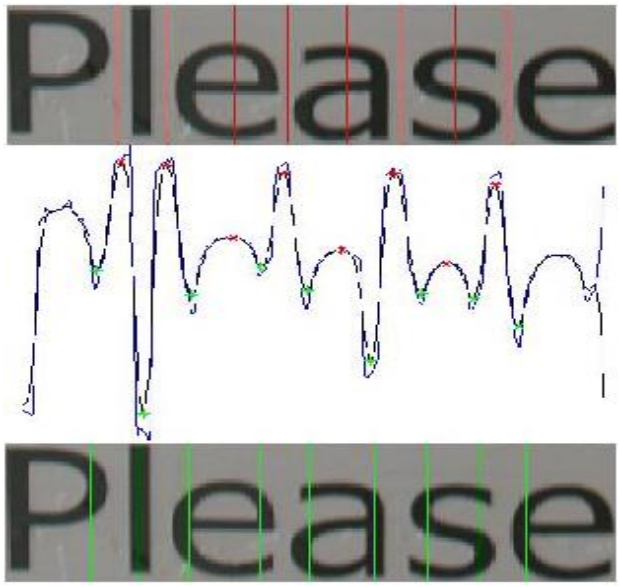


Рис. 2.9. Сегментація темного тексту на світлому тлі

2.4.2 Методи розпізнавання слів

Для завершення розпізнавання слова використовується ітеративний підхід вичерпного пошуку (метод «грубої сили»). На початку ми припускаємо, що кожен сегмент є символом-літерою, і застосовуємо їх розпізнавання. Для наступного циклу ітерації ми видаляємо сегмент, який найбільш імовірно є помилковим, враховуючи нижчий локальний максимум або вищий локальний мінімум, і

повторюємо розпізнавання, допоки не залишаться лише два сегменти. Наприклад, розпізнавання сегментів рисунка 2.8 виглядає наступним чином:

- [0BltEAIC0]
- [0BREAIC0]
- [0BREAK0]
- [0BRMK0]
- [0BMK0]
- [0MK0]

Завершальним кроком є застосування функції перевірки орфографії. Ця функція попередньо навчається на словнику слів із набору даних.

Функція Люсіна застосовує відстань Левенштейна для перевірки подібності слів. Це відбувається шляхом розрахування мінімального числа односимвольних модифікацій, необхідних для перетворення одного слова (групи слів) на інше. Функція повертає ранжований перелік можливих варіантів для даної області слова, навіть якщо розпізнавання по літерах зазнало невдачі для одного або декількох символів, і не надає жодного результату, якщо попереднє розпізнавання показало послідовність випадкових літер [25].

Наприклад, функція здатна розпізнати ’EJNGJNEERJNG’ як ’ENGINEERING’, тобто компенсувати помилки як неправильної сегментації, так і нерозпізнання деяких символів (рис. 2.10). З іншого боку, вона відкидає результат "EJNXSJJELELXXJJXX" для того самого надмірно сегментованого слова як невідомий. Подібна здатність може бути корисною для зменшення кількості помилково визначених областей розташування слів на етапі визначення наявності текста.

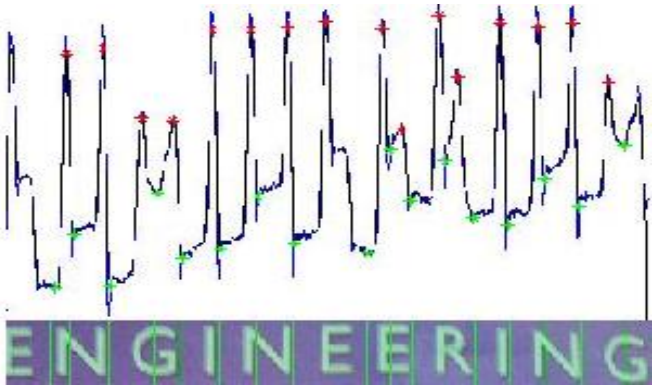


Рис. 2.10. Вертикальний профіль і сегментація слова «ENGINEERING»

2.5 Висновки

У цьому розділі розроблено метод детекції символів і слів у зображеннях, заснований на поєднанні двох алгоритмів детекції: MSER і салієнтності. Крім того, розроблені методи розпізнавання символів, сегментації і розпізнавання слів.

Дія алгоритму в цілому може бути описана наступним чином. Спочатку здійснюється детекція тексту на вхідних зображеннях, що включає наступні кроки: детекція символів на основі вилучення кольорових каналів і їх обробки за допомогою двох алгоритмів детекції; передобробка шляхом виключення зон, де відсутні символи, утворення слів шляхом вичерпного поєднання окремих зон, де було знайдено символи, післяобробка результатів детекції, а саме виключення зон, де відсутні слова.

Результати, отримані на цьому етапі, є основою для наступного – застосування методів розпізнавання слів і літер до визначених областей. Він також складається з низки кроків: сегментація слів, підхід «грубої сили» до вилучення сегментів символів, розпізнавання слів, що передбачає роботу підготовленого оффлайн класифікатора над сегментами потенційних символів-літер.

У загальному вигляді схема алгоритму може бути представленою таким чином:

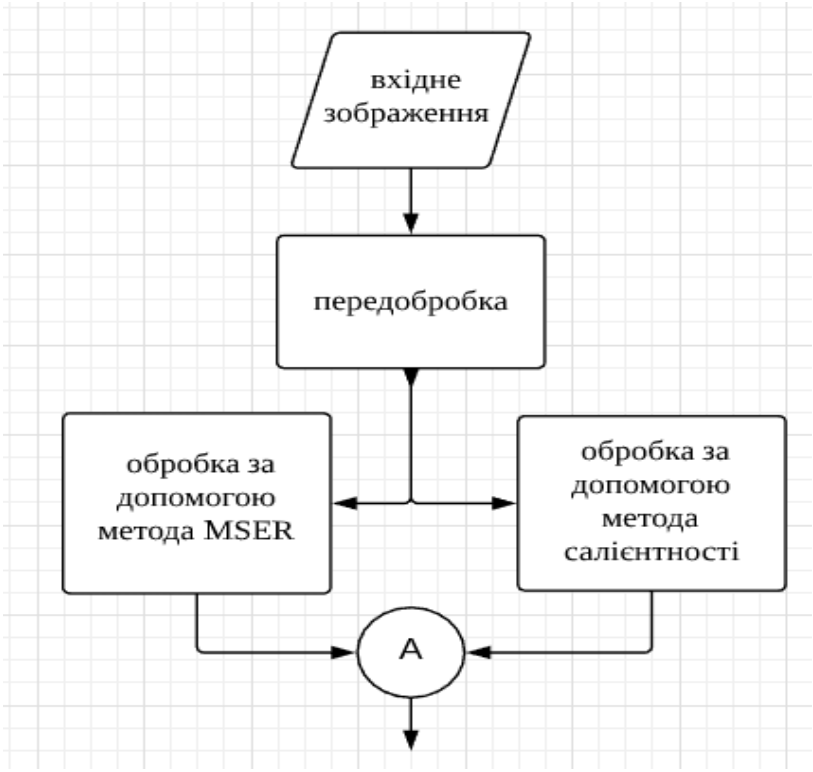


Рис. 2.11. Загальний алгоритм детекції символів і слів

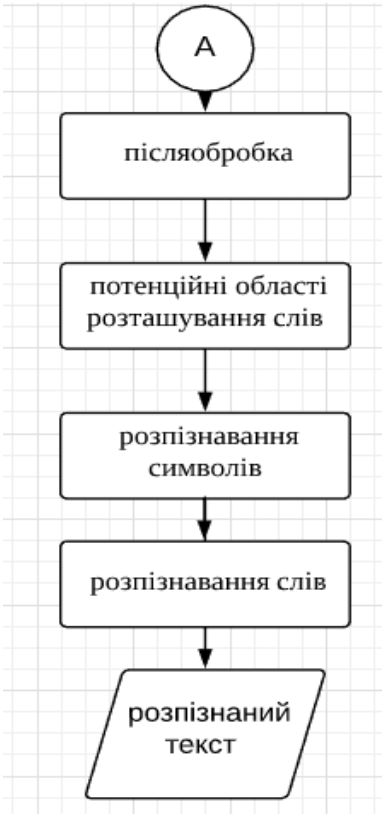


Рис. 2.12. Загальний алгоритм розпізнавання символів і слів

РОЗДІЛ 3
РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ СИСТЕМИ РОЗПІЗНАВАННЯ ТЕКСТУ В ЗОБРАЖЕННЯХ

3.1 Вибір засобів розробки програми

На сьогоднішній день у реалізації нейронних мереж важливу роль реалізації посталених архітектур відіграють, як правило, спеціалізовані пакети, що надають вже повністю реалізовані базові структури та алгоритми для зручної розробки. Найбільш рзповсюдженими є такі популярні бібліотеки, як Theano, OpenCV, Tensorflow та Torch та ін. Дані пакети реалізують програмний інтерфейс для здійснення низькорівневих розрахунків, і прийняття рішень передусім має базуватися на компромісі між продуктивністю та зручністю розробки.

Для детекції тексту в зображеннях за допомогою методів машинного навчання використовуватиметься бібліотека OpenCV. Ця бібліотека вважається найкращим програмним забезпеченням із відкритим кодом для розпізнавання образів.

OpenCV – бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Бібліотека надає засоби для обробки й аналізу вмісту зображень, зокрема розпізнавання об'єктів на фотографіях (наприклад, осіб і фігур людей, тексту тощо), відстежування руху об'єктів, перетворення зображень, застосування методів машинного навчання і виявлення загальних елементів на різних зображеннях. Сирцевий код бібліотеки написаний мовою C ++ і поширюється під ліцензією BSD. Біндіги підготовлені для різних мов програмування, таких як Python, Java, Ruby, Matlab, Lua тощо [26].

Для розпізнавання тексту, виявленого на етапі детекції, буде застосовуватись двигун Tesseract OCR.

Tesseract OCR – це двигун оптичного розпізнавання текстів для різних операційних систем. Це безкоштовне програмне забезпечення, ліцензоване Apache,

розвиток якого спонсорувався Google з 2006 [27]. Tesseract розглядається як найкращий двигун оптичного розпізнавання текстів на ринку, здатний до розпізнавання 100 різних мов.

У процесі розробки й навчання штучної нейронної мережі буде використовуватись мова програмування Python.

Ця мова має декілька переваг. Вона досить проста у вивченні і має відносно невисокий поріг входу. Людина, яка має базові знання з теорії алгоритмів і математики, може швидко засвоїти базовий функціонал, методи й синтаксис для вирішення задач прикладного характеру.

Саме ця мова лежить в основі значної кількості бібліотек, які надають у зручному виді більшість доступних алгоритмів. Здебільшого вони використовують бібліотеку NumPy. Це бібліотека, яка дозволяє швидко й ефективно працювати з числовими даними, матрицями, таблицями чисел у різних форматах, здійснювати велику кількість типових операцій, потрібних у процесі вирішення прикладних задач машинного навчання.

Після розробки й навчання моделі розпізнавання, програма буде перенесена у браузер із використанням WebAssembly.

WebAssembly або wasm — незалежний від браузера універсальний низькорівневий проміжний код для реалізації у браузері застосунків, скомпільованих з різних мов програмування. Серед основних завдань WebAssembly виділяється забезпечення перенесення між браузерами, передбачуваність поведінки та ідентичності виконання коду на різних платформах. Використання WebAssembly також дозволить істотно скоротити розмір застосунків, завдяки компактному проміжному коду, і збільшити швидкість декодування. Робоча група зі стандартизації технології WebAssembly сформована при організації W3C з представників проектів Google Chrome, Microsoft Edge, Firefox і WebKit [28].

І, нарешті, для розробки бібліотеки для конвертувння зображення в текст у браузері застосовуватиметься мова програмування JavaScript.

JavaScript – це динамічно типізована мова програмування, заснована на прототипах, яка слідує стандарту ECMA, а отже підтримує імперативну, об'єктно-орієнтовану і функціональну парадигми програмування.

JavaScript працює на всіх основних браузерах і платформах. Вона виконує задачі і на стороні сервера у всіх основних операційних системах. Жодна розробка веб-сайту сьогодні не обходиться без JavaScript, особливо якщо мова йде про front-end частину, тобто взаємодію з клієнтом. При цьому немає значення, яка саме мова використовується для розробки back-end – взаємодії з сервером. Це може бути Java, PHP, .NET, Node.js. Однак на стороні клієнта розробнику в будь-якому випадку буде необхідний JavaScript [29].

3.2 Розробка програмних засобів для детекції і розпізнавання тексту в зображеннях

3.2.1 Розробка програмних засобів для детекції тексту

У цій роботі для детекції текстів у зображеннях застосовуватиметься детектор статичної візуальної салієнтності (виразності) бібліотеки OpenCV. Даний детектор працює над логаріфмічним спектром зображення, розраховує залишки салієнтності у спектрі і створює карту салієнтності у просторі. Потім зображення завантажується у пам’ять. Це можна зробити за допомогою наступного програмного коду:

```
import argparse
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
    help="path to input image")
args = vars(ap.parse_args())

# load the input image
image = cv2.imread(args["image"])
```

Наступний крок – розрахування карти салієнтності зображення. Використовуючи модуль cv2.saliency і метод StaticSaliencySpectralResidual_create(), створюється версія об’єкта статичної спектральної залишкової салієнтності. Звідси ми викликаємо метод computeSaliency, задаючи зображення як значення параметру функції.

```
# compute the saliency map
saliency = cv2.saliency.StaticSaliencySpectralResidual_create()
(success, saliencyMap) = saliency.computeSaliency(image)
saliencyMap = (saliencyMap * 255).astype("uint8")
```

Результатом функції є карта салієнтності – saliencyMap, набір значень з плаваючою комою, зображення у відтінках сірого, яке підкреслює найбільш виразні ділянки зображення. Діапазон значень із плаваючою комою – [0, 1]. Значення, близькі до 1, є ділянками, що становлять об’єкт інтересу, а значення, близькі до 0, не є для нас цікавими.

Другий метод статичної салієнтності, що використовується в цій роботі, має назву «fine grained» («дрібнозернистий»). Він імітує перший метод, за тим винятком, що ми створюємо версію дрібнозернистого об'єкта. Також застосовується поріг для демонстрації бінарної карти, яку можна буде обробити для отримання контурів (тобто для вилучення кожної виразної області).

```
# initialize OpenCV's static fine grained saliency detector and
# compute the saliency map
saliency = cv2.saliency.StaticSaliencyFineGrained_create()
(success, saliencyMap) = saliency.computeSaliency(image)
```

Поєднання цих двох методів дає наступний результат.

Рис. 3.1. Зображення на вході

Рис. 3.2. Результат застосування методу салієнтності

Наступний крок – паралельне застосування до зображення методу MSER-детекції (maximally stable extremal regions – максимально стабільних граничних ділянок). Спочатку має бути змінений розмір зображення; потім воно завантажується у пам’ять. Згодом на виході мають бути отримані карти ознак двох шарів.

```
# resize the image and grab the new image dimensions
image = cv2.resize(image, (newW, newH))
(H, W) = image.shape[:2]
# define the two output layer names for the EAST detector model that
# second can be used to derive the bounding box coordinates of text
layerNames = [
    "feature_fusion/Conv_7/Sigmoid",
    "feature_fusion/concat_3"]
```

Перший шар – отримана на виході сигмоїдна активація, яка визначає ймовірність, містить дана ділянка текст або ні. Другий шар – карта ознак, що представляє «геометрію» зображення. Ці геометричні дані використовуватимуться для визначення координат кордонів ділянки, що містить текст у зображенні.

Нейромережа потім завантажується у пам’ять, задаючи шлях у детектор EAST бібліотеки OpenCV (Efficient and Accurate Scene Text) як параметр. Потім відбувається підготовка зображення шляхом конвертації його у блоб формат. Блоб задається як «вхідні дані» (input), і викликається метод net.forward. Надаючи layerNames як параметр методу net.forward, ми інструктуємо бібліотеку OpenCV надати нам на виході дві карти ознак, у яких зацікавлена програма: карту з геометричними даними для визначення координат кордонів ділянок із текстом на вхідному зображенні, а також карту балів, які відображують ймовірність того, що певна ділянка містить текст.

```
# load the pre-trained EAST text detector
net = cv2.dnn.readNet(args["east"])

# construct a blob from the image and then perform a forward pass of
# the model to obtain the two output layer sets
blob = cv2.dnn.blobFromImage(image, 1.0, (W, H),
    (123.68, 116.78, 103.94), swapRB=True, crop=False)
start = time.time()
net.setInput(blob)
(scores, geometry) = net.forward(layerNames)
end = time.time()
```

Дані двох вимірів необхідно перебрати в циклі для застосування немаксимального придушення. Ці два виміри: rects – містить геометричну інформацію про координати (x, y) потенційних текстових ділянок; confidences – містить інформацію про ймовірність наявності тексту для кожної ділянки в rects.

```
# apply non-maxima suppression to suppress weak, overlapping bounding
# boxes
boxes = non_max_suppression(np.array(rects), probs=confidences)

# loop over the bounding boxes
for (startX, startY, endX, endY) in boxes:
    # scale the bounding box coordinates based on the respective
    # ratios
    startX = int(startX * rW)
    startY = int(startY * rH)
    endX = int(endX * rW)
    endY = int(endY * rH)

    # draw the bounding box on the image
    cv2.rectangle(orig, (startX, startY), (endX, endY), (0, 255, 0), 2)
```

Зрештою, співставлення та комбінування цих двох методів в OpenCV дає наступний результат:

Рис. 3.3. Результат комбінування методів

3.2.2 розробка програмних засобів розпізнавання тексту

Наступним кроком після успішної детекції наявності та розташування тексту в зображенні є його розпізнавання. Для вирішення цієї задачі використовується Tesseract. Цей процес не є настільки ж складним, як попередній. Tesseract забезпечує API, здатний реалізувати всі кроки, необхідні для розпізнавання тексту із застосуванням лише декількох рядків коду.

Спочатку потрібно здійснити певну обробку зображення, отриманого на попередньому етапі, конвертувати його в зображення у відтінках сірого й видалити «шум» для того, щоб допомогти підготовленому класифікатору Tesseract краще ідентифікувати текст. Досягти цього можна за допомогою наступного коду:

```
# apply gray scale
image = cv2.imread(args["image"])
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# check to see if we should apply thresholding to preprocess the
# image
if args["preprocess"] == "thresh":
    gray = cv2.threshold(gray, 0, 255,
        cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

# make a check to see if median blurring should be done to remove
# noise
elif args["preprocess"] == "blur":
    gray = cv2.medianBlur(gray, 3)
```

Завершальний крок доволі простий. Ми застосовуємо метод Tesseract image_to_string до зображення і надаємо підготовленому класифікатору можливість виконати свою роботу.

```
# load the image as a PIL/Pillow image, apply OCR, and then delete
# the temporary file
text = pytesseract.image_to_string(Image.open(filename))
os.remove(filename)
print(text)
```

Остаточний результат процесу розпізнавання:

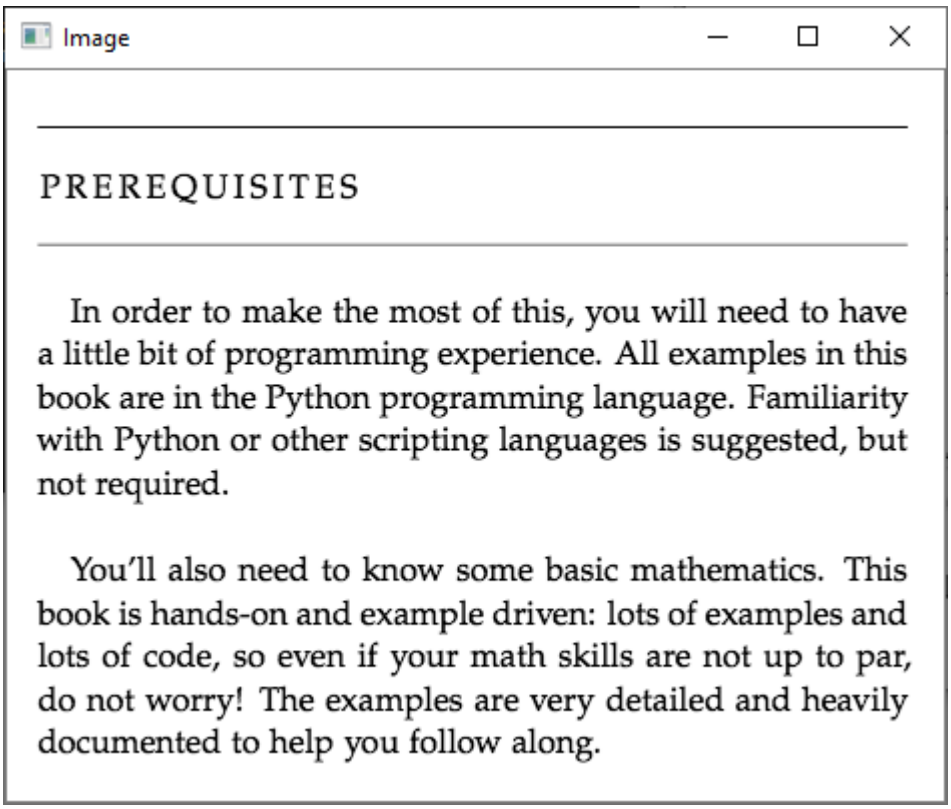


Рис. 3.4. Зображення на вході

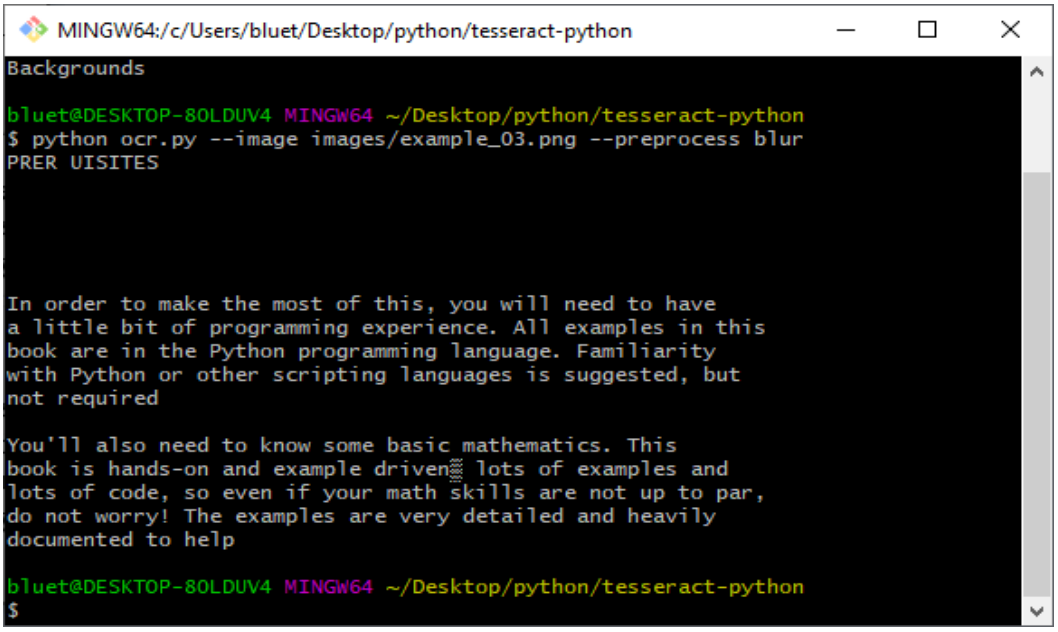


Рис. 3.5. Результат розпізнавання тексту

Наступною задачею після завершення розробки програми детекції та розпізнавання є її перенесення у браузер із використанням WebAssembly.

3.3 Розробка бібліотеки JavaScript для застосування моделі машинного навчання у браузері

Першим кроком у розробці бібліотеки є компіляція файлів за допомогою Emscripten, що конвертує код Python у мінімізований файл JS із ядром бібліотеки.

Emscripten — компілятор типу код-у-код або транскомпілятор. На вході він приймає LLVM-байткод, звичайно отриманий компіляцією сирцевого коду мовою C або C++. На виході він видає файл з кодом мовою JavaScript, що може бути запущений у веб-переглядачі.

Код із Python у Javascript конвертує не сам Emscripten, а бібліотека, яку він використовує – Empython. Застосовуючи цей інструмент, необхідно запустити лише один скрипт у командному рядку для компіляції із Python у C і потім у WebAssembly у Js файл, придатний для виконання у браузері.

BASECFLAGS=-m32 LDFLAGS=-m32 emconfigure ./configure --without-threads --without-pymalloc --disable-shared --without-signal-module –disable-ipv6

Наступний крок – розробка модуля, здатного завантажувати цей ключовий файл та використовувати його для розпізнавання тексту в зображенні.

Цей модуль має насамперед перевірити, чи підтримує браузер клієнта WebAssembly перед тим, як завантажувати зкомпільоване ядро програми.

```
module.exports = (corePath, res) => {
  if (typeof global.AIAcessCore === "undefined") {
    res.progress({ status: "loading core", progress: 0 });
    global.importScripts(corePath);
    /*
     * Depending on whether the browser supports WebAssembly
     */
    if (
      typeof global.AIAcessCore.WASM !== "undefined" &&
      typeof WebAssembly === "object"
    ) {
      global.AIAcessCore = global.AIAcessCore.CoreWASM;
    } else if (typeof global.AIAcessCore.CoreASM !== "undefined") {
      global.AIAcessCoreCore = global.AIAcessCoreCoreASM;
    } else {
      throw Error("Failed to load Core");
    }
    res.progress({ status: "loading core", progress: 1 });
  }
  return global.AIAcessCoreCore;
};
```

Імпортування підготовленої моделі штучного інтелекта та її застосування може бути достатньо вимогливим до ресурсів єдиного потоку виконання браузера. З метою забезпечення оптимального плавного досвіду користувача реалізація процесу розпізнавання тексту в зображеннях має відбуватись на окремому потоці. Для цього необхідне залучення Web Worker.

Web Worker – скрипт JavaScript, який виконується з HTML-сторінки, що працює у фоновому режимі, незалежно від скриптів інтерфейсу користувача, які

також можуть виконуватись із тієї ж самої HTML-сторінки. Web Workers часто здатні ефективніше використовувати мультиядерні процесори.

Наступний код – частина модуля, що забезпечує створення такого сценарію:

```
module.exports = (_options = {}) => {
  const id = getId('Worker', workerCounter);
  const {
    logger,
    ...options
  } = resolvePaths({
    ...defaultOptions,
    ..._options,
  });
  const resolves = {};
  const rejects = {};
  let worker = spawnWorker(options);

  workerCounter += 1;

  const setResolve = (action, res) => {
    resolves[action] = res;
  };

  const setReject = (action, rej) => {
    rejects[action] = rej;
  };

  const startJob = ({ id: jobId, action, payload }) => (
    new Promise((resolve, reject) => {
      log(`[${id}]: Start ${jobId}, action=${action}`);
      setResolve(action, resolve);
      setReject(action, reject);
      send(worker, {
        workerId: id,
        jobId,
        action,
        payload,
      });
    })
  );

  const load = jobId => (
    startJob(createJob({
      id: jobId, action: 'load', payload: { options },
    }))
  );
```

Поряд із додаванням допоміжних модулів потрібен головний, який виконує повністю процес детекції тексту та розпізнавання. Цей модуль приймає зображення як параметр, надсилає його до головної функції розпізнавання та, нарешті, повертає результат – розпізнаний текст. Наступний код є основною частиною цього модуля:

```
const createWorker = require('./createWorker');
const recognize = async (image, langs, options) => {
  const worker = createWorker(options);
  await worker.load();
  await worker.loadLanguage(langs);
  await worker.initialize(langs);
  return worker.recognize(image)
    .finally(async () => {
      await worker.terminate();
    });
};

const detect = async (image, options) => {
  const worker = createWorker(options);
  await worker.load();
  await worker.loadLanguage('osd');
  await worker.initialize('osd');
  return worker.detect(image)
    .finally(async () => {
      await worker.terminate();
    });
};

module.exports = {
  recognize,
  detect,
};
```

Завершальний етап програми – створення відкритого класу API, який можна викликати з веб-сторінки для пошуку зображень і їх заміни текстом. Цей клас ініціюється як екземпляр шаблона-одинака (англ. Singleton) і відповідає за завантаження сторінок HTML, перегляд їх DOM-дерева, виявлення тегів зображень, витягнення зображень, їх передачу основному модулю для

розпізнавання, а потім отримання розпізнаного тексту та заміну ним оригінального зображення. Вирішення цієї задачі досягається за допомогою наступного коду:

```
const AiAcessApi = new (function() {
  this.recongize = function() {
    const imgs = document.querySelectorAll("img");
    imgs.forEach(img => {
      img.parentNode.replaceChild(
        document.createElement("p"),
        this.parseTXT(img)
      );
    });
  };

  this.parseTxt = function(img) {
    AIAcess.recognize(img.src, "eng").then(({ data: { text } }) => {
      return `there was an image here that contains
        the follow text : ${text}`;
    });
  };
})();
```

3.4 Тестування роботи програмних засобів

Будь-яке програмне забезпечення після розробки потребує тестування. Це є необхідним для виявлення помилок, неточностей, невідповідностей технічному завданню, зауважень. Тестування включає не лише перевірку правильності роботи системи, а й визначення показників її роботи, наприклад, швидкості, продуктивності, надійності. Такий аналіз дає змогу оцінити програмний продукт.

Щоб переконатись у якості розроленого програмного продукту, потрібно провести тестування як кожного кроку окремо, так і програми загалом. Передусім необхідно протестувати основний модуль: наскільки безпомилково він здатен здійснити детекцію та розпізнавання тексту в зображеннях.

Під час тестування було використано 200 різних зображень із текстом. Проведено тест порівняння із програмами-конкурентами – google cloud vision та IBM Watson. Результати продемонстрували 96% точності детекції тексту; швидкість склала 0.142 секунд для одного зображення. Результати порівняння з конкурентним програмним забезпеченням відображені в наступній таблиці:

Таблиця 3.1 – Тест-порівняння детекції тексту розробленим програмним продуктом і програмами-конкурентами

	Точність	Швидкість (сек)
Розроблена програма	96%	0.142
Аналог (програма Google Vision)	90%	0.090
Аналог (програма IBM Watson)	91%	0.100

Обидві програми-конкуренти, Vision і Watson, є дещо швидшими, що можна пояснити загальною оптимізацією штучного інтелекта в цих продуктах. Водночас розроблений у цій роботі програмний засіб має мінімум 5% переваги за аспектом точності детекції. Це було досягнуто шляхом оптимізації під специфічну сферу застосування, а також завдяки поєднанню алгоритмів MSER і салієнтності, що дозволило вдосконалити відфільтрування помилкових результатів детекції.

Наступним кроком є тестування розпізнавання тексту після його детекції. У цьому тесті використовувались 100 зображень, у яких текст займав більшу частину поверхні, із різною роздільною здатністю та ефектами розмиття. Результати продемонстрували 89% точності. Швидкість складала в середньому 0.122 секунд на одне зображення. Наведена таблиця показує результати порівняння із програмами-конкурентами:

Таблиця 3.2 - Тест-порівняння розпізнавання тексту розробленим програмним продуктом і програмами-конкурентами

	Точність	Швидкість (сек)
Розроблена програма	89%	0.122
Аналог (програма Google Vision)	85%	0.190

Продовження таблиці 3.2

Аналог (програма IBM Watson)	82%	0.185

Отримані під час тестування результати показали, що розроблений програмний продукт має перевагу над програмами-конкурентами за обома аспектами, як точністю (мінімум 4%), так і швидкістю (0.06 секунд). Це було досягнуто завдяки оптимізації програми під конкретні цілі розпізнавання тексту в зображеннях.

Завершальним етапом стало тестування повного циклу роботи програмного засобу. Його проведення включило використання програми на 20 різних веб-сторінках, де були представлені зображення із текстовими елементами. Результати тестування: 100% успіху детекції зображення на веб-сторінці, 90% - детекції та розпізнавання тексту в зображеннях, 90% - заміни зображення текстом.

Досконалість детекції зображень на веб-сторінках пояснюється ефективністю JavaScript у перегляді DOM-дерева та знаходженні вузлів (nodes) зображень. Це відбувається шляхом виявлення тегів зображень. Результати детекції та розпізнавання тексту є аналогічними до результатів попередніх етапів тестування і цілком очікуваними.

3.5 Висновки

У цьому розділі розглянуто інструменти, використані для розробки програмного продукту. Застосування таких бібліотек, як OpenCV і Tesseract, дозволило, зокрема шляхом поєднання алгоритмів MSER і салієнтності, розробити методи детекції тексту в зображеннях. На наступному етапі було створено програмні інструменти, які реалізують сегментацію і розпізнавання тексту. І, нарешті, розроблено програмні засоби, які створюють можливість для використання завершеної моделі програми у браузері – просту та готову до

утилізації бібліотеку JavaScript. Після цього було здійснено тестування готового програмного продукту. Тестування проводилось для всіх трьох етапів розробки; його результати порівнювались із показниками конкуретнтних програмних рішень і продемонстрували переваги розробленої програми за аспектами швидкості та точності детекції і розпізнавання.

РОЗДІЛ 4
ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності. Результатом магістерської кваліфікаційної роботи «Методи та засобі розпізнавання зображень» є розробка програмної реалізації бібліотеки розпізнавання тексту в зображеннях на веб-сторінках з використанням машинного навчання. Для проведення технологічного аудиту залучено трьох незалежних експертів. У нашому випадку такими експертами є: Войтко Вікторія Володимирівна (к.т.н., доцент каф. програмного забезпечення ВНТУ), Хошаба Олександр Мирославович (к.т.н., доцент каф. програмного забезпечення ВНТУ) та Артюх Антон Владимирович (головний інженер ТОВ EXADEL»). Оцінювання комерційного потенціалу [31] буде здійснене за критеріями, що наведені в таблиці 4.1.

Таблиця 4.1 - Критерії оцінювання комерційного потенціалу розробки, бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Крит-ерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження таблиці 4.1

3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критер.	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 4.1

9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуют ься у військовопромисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3- х років
12	Необхідна розробка регламентих документів та отримання великої кількості дозвільних документів	Необхідно отримання великої кількості дозвільних документів, що вимагає значних коштів та часу	Процедура отримання дозвільних документів у вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу експертами розробки зведено в таблицю 4.2.

Таблиця 4.2 - Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1 – Колесницький	2 – Арсенюк	3 – Кулеба
	Бали, виставлені експертами:		
1	3	3	4
Ринкові переваги (недоліки):			
2	2	3	2
3	3	4	3
4	3	3	4
5	3	4	3
Ринкові перспективи			
6	3	2	2
7	2	2	3
Практична здійсненність			
8	4	3	4
9	3	3	3
10	4	4	4
11	4	4	4
12	4	4	4
Сума балів	СБ1=38	СБ2=39	СБ3=40
Середньоарифметична сума балів СБ	$\overline{СБ} = \frac{\sum_1^3 СБ_j}{3} = \frac{116}{3} = 39$		

За даними таблиці 4.3 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 4.3.

Таблиця 4.3 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 – 10	Низький
11 – 20	Нижче середнього
21 – 30	Середній
31 – 40	Вище середнього
41 – 48	Високий

Рівень комерційного потенціалу розробки, становить 39 балів, що відповідає рівню «вище середнього» [31].

Проаналізуємо суть технічної проблеми та розглянемо аналоги. Процеси розпізнання тексту добре вивчені, існує безліч додатків і працюючих прикладів. Однак, все ще ведуться дослідження і боротьба розробників за все нові можливості в якості та швидкості [32]. Досить мало існує універсальних систем, які здатні якісно розпізнати текст, працювати досить швидко, особливо якщо на шляху розпізнавання виявляються проблеми з якістю зображення (існують завади), або об’єкту розпізнавання.

Виникає потреба в такому продукті, який з одного боку зможе централізовано оброблювати вхідну графічну інформацію і використовуватись як для розпізнавання сканованих текстів, так і для роботи з запитами користувача, а з іншого боку працювати швидко і ефективно.

Нова розробка повинна відповідати двом критеріям: бути точною, а отже і спеціалізованою; діяти швидко і якісно.

Найкращі методи розпізнавання образів/тексту використовують оптичні характеристики. Тобто існує декілька характерних для символу властивостей і система перевіряє наявність таких в зображенні. Такий пошук виконується досить

швидко і має високі показники точності розпізнавання, однак має декілька суттєвих недоліків. По-перше для кожного символу і для кожного шрифту потрібно знайти та формалізувати десятки або і сотні ознак, причому перевірити їх на наявність перетинів. По-друге, виявлення завад або перешкод для розпізнавання лише виконується попередньою обробкою, а отже зникнення символу може призвести до втрати цієї інформації. Взагалі метод не придатний для розпізнання рукописних символів, він може бути модифікований до поставленої задачі, однак буде мати дуже низькі показники якості розпізнавання.

Нейронна мережа являє собою систему з однотипних елементів обробки, які на основі вхідних даних формують вихідний сигнал. В залежності від параметрів елементів змінюється характер роботи мережі, цей 69 факт використовується для використання системи для задач класифікації та машинного навчання.

Існують методи побудови, методика визначення оптимальних параметрів мережі, таких як кількість шарів, розмір шарів, топологія, основні методи подання вхідного сигналу та їх переваги та недоліки. Одним з найкращих (на практиці) методів подання вхідного сигналу є виділення знімків векторного поля зображення. На відміну від точкового знімку такому методу вистачає мережі меншого розміру і як наслідок меншого обсягу обчислень. Проте, такий метод вимагає більш складної попередньої обробки.

Нейронна мережа сама по собі виконує задачу класифікації. В багатовимірному просторі нейронна мережа виділяє області які відповідають деяким екземплярам класу. Основними перевагами нейромережевих структур є їх можливість навчання. Більш того вони можуть досить вдало вирішувати класифікацію об’єкта який не очікувався при навчанні, оскільки апроксимують об’єкт у відповідність до відомих критеріїв.

У цій магістерській роботі запропоноване поєднання методів. Застосування підготовленої моделі машинного навчання до детекції тексту за допомогою

комбінування алгоритмів салієнтності і MSER допоможе зменшити ймовірність помилки детекції.

Враховуючи описані вище методи та їх недоліки, можна зробити висновок, що задача розпізнавання все ще потребує більш швидкого та точного вирішення. Найбільш популярні та практичні програми використовують закономірності в символах для розпізнавання, що дає високі показники розпізнавання такі як швидкість або точність, однак можуть виявитись безпомічними при появі завади, або частковій втраті інформації.

Для визначення конкурентоспроможності нової розробки були розглянуті сучасні існуючі аналоги на ринку програмного забезпечення.

Першим аналогом є програма Watson – комп'ютерна система штучного інтелекту, що здатна відповідати на питання, задані природною мовою. Watson була створена в рамках проекту IBM DeepQA.

Другим аналогом є програма Tesseract – програма для візуального розпізнавання текстів для різних операційних систем. Спочатку значна частина коду була написана мовою програмування C, але згодом було здійснено його доопрацювання для сумісності з C++ компиляторами.

Третім аналогом є програма TensorFlow – відкрита програмна бібліотека для машинного навчання низці задач, розроблена компанією Google для задоволення її потреб у системах, здатних будувати та тренувати нейронні мережі для виявлення та розшифровування образів і кореляцій за аналогією з навчанням і мисленням, які застосовують люди.

Четвертим аналогом є програма Amazon Recognition – сервіс, який дозволяє просто вбудовувати у додатки аналітику зображень і відео на основі глибокого навчання. Необхідно лише надати API Recognition відповідне зображення або відео. Сервіс здатен розпізнавати об’єкти, людей, тексти, дії, а також виявляти неприйнятний контент.

П'ятийм аналогом є програма Google Cloud Vision API – потужний інструмент, який може стати основою для нескінченних можливостей застосування у поєднанні з бібліотеками мови Python.

Тому як аналог для розробки було обрано програму IBM Watson. Наряду з певними перевагами аналог має і недоліки, зокрема невисоку достовірність розпізнавання символів.

У розробці дана проблема вирішується тим, що використовується поєднання двох методів детекції. Також програмний продукт випереджає аналог за такими параметрами як вартість, простота використання та економія ресурсів комп’ютера.

У таблиці 4.4 наведені основні технічні показники аналога і нового програмного продукту.

Таблиця 4.4 - Основні технічні показники аналога і нового програмного продукту

Показники	Аналог	Нова розробка	Відношення параметрів нової розробки до параметрів аналога
Достовірність детекції тексту	91%	96%	5%
Достовірність розпізнавання тексту	82%	89%	7%
Простота використання (за 12-бальною шкалою)	8	9	1
Економія ресурсів комп‘ютера (за 12-бальною шкалою)	7	11	4

Аналіз таблиці 4.4 дозволяє зробити висновки, що нова розробка має кращі показники Достовірність детекції тексту 5% та достовірності розпізнавання тексту 7% .

Також із табл. 4.4 видно, що нова розробка має кращі такі технічні показники як простота використання (бібліотека дуже проста у використанні) та економія ресурсів комп‘ютера (програма менше займає місця на диску та при роботі використовує менший обсяг оперативної пам’яті.)

Новий програмний продукт буде реалізовувати як ліцензійна угода, що застосовується між власними майновими правдою на комп’ютерній програмі та її учасником, який намагається змінити і обмежує права інших.

Новий програмний продукт підходить для використання як індивідуальним, так і корпоративним користувачам. Крім того нижча ціна ніж у конкурентів буде сприяти поширенню нашого продукту та позитивно вплине на його конкурентоспроможність. Безпосередніми споживачами розробки можуть бути особи, яким потрібно переводити скановані рукописні тексти поганої якості та зображення у цифровий формат.

Слід враховувати те, що на ринку програмного забезпечення конкурентна перевага вимірюється не роками, а місяцями і тижнями. Новому товару погрожує конкуренція, тому необхідно адресуватися до широкого кола споживачів, для того щоб довести переваги нової розробки і зацікавити покупців придбати саме нашу продукцію. Для цього ми будемо максимально використовувати рекламу в мережі Інтернет.

Для виведення нового товару на ринок необхідно надати інформацію, переконати, нагадати, схилити до рішення про покупку нової розробки, тобто проводити рекламну компанію. Споживачі проходять через різні стадії: від “поінформованості” (про наявність потреби), “знання” (про продукт, що задовольнить потребу), “симпатії” і “перевазі” (певним маркам) до “переконання” (що саме цей товар краще) і “покупки”. Згодом вони випробують “задоволення”,

що рекламодавець прагне “підкріпити”, або “незадоволення”, що рекламодавець прагне перебороти.

Таким чином, нам необхідно визначити, якого стану досягли наші цільові споживачі (за допомогою маркетингових досліджень), і відповідно установити цілі реклами, тобто у випадку з новим програмним продуктом максимізувати “поінформованість” і “знання” того, що програмний продукт дозволяє робити.

Якісне сервісне обслуговування є заставою стабільної роботи і розвитку власного підприємництва. Саме тому необхідно приділити найсерйознішу увагу наданню високоякісних послуг з технічної підтримки і супроводу програмного продукту, що буде поставлятися користувачам. Післяпродажне супроводження користувача може включати:

1. Техпідтримку через систему он-лайн повідомлень.

2. Техпідтримку через електронну пошту.

3. Техпідтримку через форум. Є необхідним елементом техпідтримки, що дозволяє користувачам самостійно допомагати один одному у вирішенні поточних проблем з Програмою.

4. Випуск оновлень і виправлень. Випуску оновлень і виправлень передує процес збору та аналізу інформації в процесі експлуатації користувачами програми, що надходить по каналах технічної підтримки, а також отриманої в результаті моніторингу інформації в Інтернет.

5. Збір відгуків та пропозицій. Збір відгуків та пропозицій від користувачів, має на увазі отримання інформації, яка не відображена в скаргах і претензії, що надходять в техпідтрмку. Може реалізовуватися через форми он-лайн опитування, форми прямого зв'язку в Інтернет, а також в самій Програмі, формі напрямки коментаря про причини видалення програми. Також в разі реалізації даної процедури слід враховувати, що аналізом і переробкою даної інформації повинен хтось займатися і бути щиро в цьому зацікавленим.

6. Навчання навичкам роботи з програмою. Метою реалізації післяпродажного супроводу користувача є:

- отримання оперативної інформації від користувачів про властивості і якостях програми, на які необхідно звернути увагу, або виправити.

- збір інформації для розробки і випуску наступної версії програми, або згортання проекту. - формування у користувача іміджу відповідального суб‘єкта бізнесу, зі зрозумілими процедурами обслуговування користувачів, а відповідно поліпшення зручності використання програми користувачами і отримання таким чином конкурентної переваги.

- створення лояльного ставлення користувача до програми, як зручного інструменту вирішення їх проблем.

Розробка повністю готова до реалізації. Є наявними фінансові та трудові ресурси, а також фахівців відповідної кваліфікації для впровадження на ринок нового програмного продукту.

В зв‘язку з тим, що існує потенційно велика кількість зацікавлених осіб у новому програмному забезпеченні проводяться переговори з потенційними інвесторами для можливості подальшої комерціалізації розробки.

4.2 Прогнозування витрат на виконання наукової роботи та впровадження результатів

Проведемо прогнозування витрат [33] на виконання науково-дослідної, дослідно-конструкторської та конструкторсько-технологічної роботи для розробки програмного забезпечення, яке складається з таких етапів:

1-й етап: розрахунок витрат, які безпосередньо стосуються виконавців даного розділу роботи;

2-й етап: розрахунок загальних витрат на виконання даної роботи;

3-й етап: прогнозування загальних витрат на виконання та впровадження результатів даної роботи.

Виконаємо розрахунок витрат приймаючи до уваги те, що розробкою займався один розробник програмного забезпечення.

1. Основна заробітна розробника-дослідника Зо:

$$Зо = \frac{М}{Т_р} . t \text{ [грн]} \quad (4.1)$$

де М – місячний посадовий оклад – 7000 грн;

Тр – число робочих днів в місяці; приблизно Тр = (22) дні;

t – число робочих днів роботи розробника-дослідника - 60.

$$З_0 \frac{7000}{2000} . 60 = 16363{,}63 \text{ (грн)}$$

2. Додаткова заробітна плата Зд розробника розраховується як 10% від основної

заробітної плати:

$$Зд = 0{,}10 \cdot 16363{,}63 = 1636{,}36 \text{ (грн)}.$$

3. Нарахування на заробітну плату Нзп розробника становить:

$$Н_{зп} = (З_0 + З_0) . \frac{\beta}{100} \text{ [грн]} \quad (4.2)$$

де Зо – основна заробітна плата розробника;

Зд – додаткова заробітна плата розробника;

β – ставка єдиного внеску на загальнообов‘язкове державне соціальне страхування – 22%.

$$Нзп = (16363{,}63 + 1636{,}36) \cdot 0{,}22 = 3960{,}00 \text{ (грн)}.$$

Амортизація обладнання, комп‘ютерів та приміщень, які використовувались під час виконання даного етапу роботи. Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

У спрощеному вигляді амортизаційні відрахування А в цілому розраховуємо за формулою:

$$А = \frac{Ц \cdot Т}{12 \cdot Т_В} \text{ [грн]} \quad (4.3)$$

де Ц – загальна балансова вартість обладнання, приміщення тощо, грн;

Т – фактична тривалість використання, міс;

ТВ – термін використання обладнання, приміщень тощо, роки. Розробка програмного забезпечення проводилася протягом 3 місяців.

Зроблені розрахунки зведено до таблиці 4.5.

Таблиця 4.5 – Амортизаційні відрахування

Найменування	Балансо ва вартість, грн	Термін викори стання, р	Фактична трив. використання, міс.	Величина амортизаційних відрахувань, грн
Офісне приміщення	100000	25	3	100
Комп‘ютер	15000	5	3	750
Монітор	3700	6	3	154
Всього				1004

Інформацію про матеріали, що використовуються при виготовленні даного інноваційного продукту внесено до таблиці 4.6.

Таблиця 4.6 – Матеріали, що використовуються при виготовленні даного продукту

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено, шт.	Вартість витраченого матеріалу, грн
Папір (пачка)	87,00	1	78,00
Ручка	5,00	2	10,00
Всього	88,00		

Під час розробки програмного продукту використовувались лише безкоштовні програмні засоби.

Витрати на енергію визначаються на основі витрат на одиницю продукції та тарифів на енергію за допомогою формули 4.2:

$$В_е = В \cdot П \cdot Ф \cdot К_п \quad [грн] \qquad (4.4)$$

де В– вартість 1кВт електроенергії;

П – установлена потужність обладнання, кВт;

Ф – фактична кількість годин роботи комп‘ютера при створенні програмного продукту, годин;

Кп – коефіцієнт використання потужності (Кп = 0,7).

Отже, витрати на енергію становлять:

$$В_е = 1{,}88 \cdot 0{,}5 \cdot 480 \cdot 0{,}4 = 180{,}48 \text{ (грн)}$$

Також потрібно врахувати витрати на доступ до мережі Інтернет, що використовувався під час виконання роботи.

Витрати за доступ до Інтернет можна розрухувати за формулою:

$$В_{ді} = Ц_{ді} \cdot Т \text{ [грн]}$$

де Цді – це ціна доступу за місяць;

Т – кількість місяців використання доступу до мережі.

Отже, витрати на доступ до мережі Інтернет становлять:

$$Вді = 100 \cdot 3 = 300 \text{ (грн).}$$

В результаті сума усіх витрат, що вказані вище дає витрати на виконання даного етапу роботи В:

$$В = Зо+Зд+Нзп+А + Вмат + Ве + Вді \text{ [грн]},$$

$$В = 16363{,}63 + 1636{,}36 + 3960{,}00 + 1004 + 88 + 180{,}48 + 300 = 23532{,}47 \text{ (грн).}$$

2-й етап. Розрахунок загальних витрат на виконання даної роботи. Загальна вартість всієї наукової роботи визначається за Взаг формулою:

$$В_{заг} = \frac{В}{\alpha} \text{ [грн]}$$

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

Так, як над роботою задіяна одна людина, якою виконується уся робота, то α становить 1. Підставивши дані у формулу, отримуємо:

Взаг = 23532,47 (грн).

3-й етап. Прогнозування загальних витрат на виконання та впровадження результатів виконаної роботи. Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \text{ [грн]}$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи. Так, якщо розробка знаходиться:

- на стадії науково-дослідних робіт, то $\beta \approx 0,1$;

- на стадії технічного проектування, то $\beta \approx 0,2$;

- на стадії розробки конструкторської документації, то $\beta \approx 0,3$;

- на стадії розробки технологій, то $\beta \approx 0,4$; - на стадії розробки дослідного зразка, то $\beta \approx 0,5$; - на стадії розробки промислового зразка, $\beta \approx 0,7$;

- на стадії впровадження, то $\beta \approx 0,9$.

Отже, підставимо дані в формулу й отримаємо результат:

$$ЗВ = \frac{23532,48}{0,9} = 26147,19 \text{ (грн).}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спробуємо кількісно спрогнозувати, яку вигоду, можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Зрозуміло, що всі зроблені тут розрахунки будуть приблизними і не передбачають деталізації.

В умовах ринку узагальнюючим позитивним результатом, що його отримує підприємство від впровадження результатів тієї чи іншої розробки, є збільшення чистого прибутку підприємства (організації). Зростання чистого прибутку ми можемо оцінити у теперішній вартості грошей.

Саме зростання чистого прибутку забезпечить підприємству надходження додаткових коштів, які дозволять покращити фінансові результати діяльності та виплатити кредити (якщо вони потрібні для впровадження результатів розробки).

Оцінити збільшення чистого прибутку підприємства Δ Пі для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки можливо використовуючи формулу:

$$\Delta Пі = \sum_{i}^{n}(\Delta П_0 \cdot N + Ц_0 \cdot N \cdot \Delta N) \cdot \lambda \cdot \rho \left(1 - \frac{v}{100}\right) [\text{грн}] \qquad (4.8)$$

де Δ По – покращення основного оціночного показника від впровадження результатів розробки у даному році. Зазвичай таким показником може бути ціна одиниці нової розробки;

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

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

Цо – основний оціночний 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;

λ – коефіцієнт, який враховує сплату податку на додану вартість.

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати

ρ = 0,2...0,3;

υ – ставка податку на прибуток (18%).

В результаті впровадження результатів наукової розробки покращується якість продукту, що дозволяє підвищити ціну його реалізації на 50 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року – на 250 шт., протягом другого року – ще на 550 шт., протягом третього року – ще на 350 шт.

Орієнтовно: реалізація продукції до впровадження результатів наукової розробки складала 50 шт., а її ціна – 500 грн.

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

Збільшення чистого прибутку підприємства Δ Пі протягом першого року складе:

$$\Delta П_1 = [500 \cdot 50 + (50 + 1500) \cdot 250] \cdot 0{,}88 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 94750 \text{ (грн)}$$

Збільшення чистого прибутку підприємства Δ Пі протягом другого року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе:

$$\Delta П_2 = [500 \cdot 50 + (50 + 1500) \cdot (250 + 550)] \cdot 0{,}88 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 248200 \text{ (грн)}$$

Збільшення чистого прибутку підприємства Δ Пі протягом третього року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе:

$$\Delta П_3 = [500 \cdot 50 + (50 + 1500) \cdot (250 + 550 + 350)] \cdot 0{,}88 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 345850 \text{ (грн)}$$

Загальне збільшення прибутку підприємства від впровадження розробки за три роки становить:

$$\Delta П = 94750{,}00 + 248200{,}00 + 345850{,}00 = 688800{,}00 \text{ (грн)}.$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Основними показниками, які визначають доцільність фінансування наукової розробки інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності. Розрахунок ефективності вкладених інвестицій передбачає проведення таких робіт:

1-й крок. Розраховується теперішня вартість інвестицій PV, що вкладаються в наукову розробку. Такою вартістю можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, розраховану за формулою, тобто будемо вважати, що ЗВ = PV=26147,19.

2-й крок. Розраховується очікуване збільшення прибутку ΔПі , що його отримає підприємство (організація) від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження. Таке збільшення прибутку також було розраховане нами раніше.

3-й крок. Будуємо вісь часу, на яку наносимо всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів.

Платежі показуємо у ті терміни, коли вони здійснюються. Припустимо, що загальні витрати ЗВ на виконання та впровадження результатів НДДКР (або

теперішня вартість інвестицій PV) дорівнює 26147,19 грн. Результати вкладених у наукову розробку інвестицій почнуть виявлятись протягом трьох років. У першому році підприємство отримає збільшення чистого прибутку на 94750,00 грн відносно базового року, у другому році – збільшення чистого прибутку на 248200 грн (відносно базового року), у третьому році – збільшення чистого прибутку на 345850 грн (відносно базового року).

Тоді рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рис. 4.1.

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

4-й крок. Розраховується абсолютна ефективність вкладених інвестицій Еабс. Для цього використовується формула:

$$E_{абс} = (ПП - PV) \quad (4.9)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.; PV – теперішня вартість інвестицій PV = ЗВ, грн. У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_1^T \frac{\Delta Пі}{1+\tau} \quad (4.10)$$

де ΔПі – 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 „0”.

Якщо Еабс > 0, то результат від проведення наукових досліджень та їх впровадження принесе прибуток, але це також ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даного проекту (роботи).

Розрахуємо абсолютну ефективність інвестицій, вкладених у реалізацію проекту. Ставка дисконтування τ дорівнює 0,1. Отримаємо:

$$ПП = \frac{94750}{(1+0,1)^1} + \frac{248200}{(1+0,1)^2} + \frac{248200}{(1+0,1)^3} = 672921,91 \quad (грн)$$

Тоді,

$$Е_{абс} = 672921,91 - 26147,19 = 646774,72 \ (грн)$$

Оскільки $Е_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР є доцільним. 5-й крок. Розраховуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій Ев. Для цього використовуємо формулу:

$$Е_В = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1 \qquad (4.11)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн; PV –теперішня вартість інвестицій PV = ЗВ, грн;

Тж – життєвий цикл наукової розробки, роки.

Далі, розрахована величина Ев порівнюється з мінімальною (бар'єрною) ставкою дисконтування τ мін, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть.

У загальному вигляді мінімальна (бар'єрна) ставка дисконтування τ мін визначається за формулою 4.10:

$$\tau = d + f \quad (4.12)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; d = 0,22;

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = 0,15.

Якщо величина Ев > τ мін, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки. В іншому випадку фінансування наукової розробки здійснюватися не буде. Спочатку спрогнозуємо величину τ мін. Припустимо, що за даних умов τ мін = 0,22 + 0,15 = 0,37. Тоді відносна (щорічна) ефективність вкладних інвестицій в проведення наукових досліджень та впровадження їх результатів складе:

$$E_В = \sqrt[3]{1 + \frac{646774,72}{26147,19}} - 1 = \sqrt[3]{2,91} - 1 = 1,91 \text{ або } 191\%$$

Оскільки Ев = 191% > τ мін = 0,37 = 37%, то у інвестора буде зацікавленість вкладати гроші в дану наукову розробку, оскільки значно більші прибутки він отримає від того, що інвестує кошти розробку, а не розмістить гроші на депозит у комерційному банку.

6-й крок. Розраховуємо термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій Ток можна розрахувати за формулою:

$$T_{ok} = \frac{1}{E_в} \quad (4.13)$$

Якщо Ток буде менше 5-ти років, то фінансування даної наукової розробки загалом є доцільним. В інших випадках потрібні додаткові розрахунки та обґрунтування. Для нашої розробки термін окупності вкладених у реалізацію проекту інвестицій Ток складе:

$$T_{ok} = \frac{1}{1{,}91} = 0{,}52 \text{ (року)}$$

що свідчить про доцільність фінансування даної наукової розробки.

4.5 Висновок

У цьому розділі було виконано оцінювання комерційного потенціалу розробки, а саме бібліотеки розпізнавання тексту в зображеннях на веб-сторінках для застосування у браузері. Проведено технологічний аудит з залученням трьох незалежних експертів. Визначено, що рівень комерційного потенціалу розробки вище середнього. Аналіз комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає аналогічні програмні продукти і є перспективною розробкою. Він має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку. Існуючі переваги нової розробки дозволять швидко її поширити та популяризувати.

Згідно із розрахунками всіх статей витрат на виконання науководослідної, дослідно-конструкторської та конструкторсько-технологічної роботи загальні витрати на розробку складають 266147,19 грн. При цьому приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки 672921,91 грн. Розрахована абсолютна ефективність вкладених інвестицій в сумі 646774,72 грн свідчить про отримання прибутку інвестором від комерціалізації програмного продукту. Щорічна ефективність вкладених в наукову розробку інвестицій складає 191%, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 32%. Це означає потенційну зацікавленість інвесторів у фінансуванні розробки. Термін окупності вкладених у реалізацію проекту

інвестицій становить 0,52 року, що також свідчить про доцільність фінансування нової розробки.

ВИСНОВКИ

У магістерській дисертації було розроблено програмні засоби для конвертації зображень, представлених на веб-сторінках, в озвучений текст, для вдосконалення користувацього досвіду людей із обмеженнями зору. В основу розробки покладено поєднання двох методів детекції тексту в зображеннях і його сегментації та розпізнавання. Програму розроблено завдяки використанню методів машинного навчання, зокрема таких бібліотек, як OpenCV і Tesseract, а також із застосуванням мов програмування Python і JavaScript.

На першому етапі роботи було проаналізовано стан предметної області на сьогоднішній день. Розглянуто основні програмні рішення та засоби, які існують у сфері розпізнавання образів, а також визначено їх особливості, сильні сторони та недоліки. Було підкреслено загальний характер цих програм і відсутність спеціалізованих засобів, орієнтованих на детекцію текстових елементів у зображеннях на веб-сайтах у браузері та їх розпізнавання. На основі цього було спроектовано та розроблено програмний продукт, представлений у цій роботі.

На другому етапі роботи було проаналізовано найбільш поширені та ефективні методи вирішення задачі розпізнавання зображень і тексту в зображеннях. Цей аналіз дозволив розробити вдосконалений метод детекції текстових елементів у зображеннях і їх подальшого розпізнавання. Разом із основною схемою алгоритму детекції та розпізнавання було створено архітектуру програми.

Для реалізації програмного рішення і досягнення поставлених задач використовувались наступні мови програмування: Python для розробки підготовленої моделі детекції та розпізнавання; JavaScript для розробки бібліотеки, яка створює можливість для застосування моделі розпізнавання (як основного модуля бібліотеки) у браузері.

Кодування для реалізації поставлених задач здійснювалось у редакторі Microsoft Visual Studio Code. Бібліотека OpenCV застосовувалась для детекції тексту в зображеннях, а бібліотека Tesseract OCR – для розпізнавання тексту. Контроль версій розробки програми здійснювався із використанням Git.

Завершальним етапом роботи стало тестування готового програмного продукту. Було перевірено повну операційну здатність програми відповідно до поставленого завдання. Тестування проводилося за аспектами точності детекції і розпізнавання текстів у зображеннях, а також швидкості проведення цих операцій. Отримані результати порівнювались із відповідними результатами схожих програмних рішень, а саме Google Vision та IBM Watson. Програма показала високу точність розпізнавання на тому ж рівні, що й програми-конкуренти (до 90%), і високу швидкість роботи – приблизно 0.3 секунди для обох етапів, обто детекції і розпізнавання. Після цього надійність розробленого програмного рішення було перевірено безпосередньо під час роботи у браузері. Результати тестування продемонстрували здатність виявляти та замінювати всі зображення із текстом на відповідній веб-сторінці.

Наприкінці роботи було проведено економічний аналіз розробленого програмного продукту. Було визначено його потенційний вплив на ринок і доведено, що рівень комерційного потенціалу розробки вище середнього. Аналіз комерційного потенціалу розробленого програмного рішення показав, що продукт за своїми характеристиками випереджає аналогічні програмні продукти і є перспективною розробкою. Він має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку.

Перелік посилань

1. Manuel Castells, “The impact of the internet on society: A global perspective”, – 2014. – С. 21.
2. Доступність сайтів для незрячих людей [Електронний ресурс]. Режим доступу: URL: http://www.i-law.kiev.ua/доступність-сайтів-для-незрячих-люде
3. Шалымов Д.С. Математическое обеспечение для разработки и анализа систем распознавания образов, использующих рандомизированные алгоритмы [Електронний ресурс]. Режим доступу: URL: http://www.math.spbu.ru/user/gran/students/sh_diss.pdf?fbclid=IwAR17FrfVZ5bsa8hOiWzNBlXgauZ1SHOm2oN-vzTUgLLi_06aUVhMdQiFJz0. – С. 45.
4. Горелик А.Л. Методы распознавания / А.Л. Горелик, В.А. Скрипник. – М.: Высшая школа, 1989. – 232 с.
5. Шалымов Д.С. Математическое обеспечение для разработки и анализа систем распознавания образов, использующих рандомизированные алгоритмы [Електронний ресурс]. Режим доступу: URL: http://www.math.spbu.ru/user/gran/students/sh_diss.pdf?fbclid=IwAR17FrfVZ5bsa8hOiWzNBlXgauZ1SHOm2oN-vzTUgLLi_06aUVhMdQiFJz0. – С. 52.
6. Хом’юк В.В. Методи попередньої обробки інформації в процесі розпізнавання образів // Вісник Сумського державного університету. – 2002. - С. 7.
7. Amazon Rekognition [Електронний ресурс]. Режим доступу: URL: https://aws.amazon.com/rekognition/. – Назва з екрану.
8. OpenCV Wikipedia [Електронний ресурс]. Режим доступу: URL: https://en.wikipedia.org/wiki/OpenCV/. – Назва з екрану.
9. Google Vision API [Електронний ресурс]. Режим доступу: URL: https://cloud.google.com/vision/. – Назва з екрану.
10. Маслій Р.В. Використання локальних бінарних шаблонів для розпізнавання облич на напівтонових зображеннях // Наукові праці ВНТУ. 2008. – № 4. – С. 5.
11. Юр Т.В. Аналіз методів розпізнавання облич на зображеннях // Наукові праці ДонНТУ. Серія «Інформатика, кібернетика та обчислювальна техніка». – 2015. – № 2(21) – С.42-46 – С.45.
12. Сучасні тенденції у вирішенні задач виявлення та розпізнавання об′єктів на зображеннях / Г. Л. Лисенко, М. Г. Тарновський, Л. В. Кузьменко // Оптико-електронні інформаційно-енергетичні технології. – 2017. – № 1. – С. 18-23. – С.19
13. Методы корреляционного обнаружения объектов / А.В. Гиренко, В.В. Ляшенко, В.П. Машталир, Е.П. Путятин. – Х.: АО “БизнесИнформ”, 1996. – 112 с.
14. Canny J. F. Finding edges and lines in images. – M.I.T. Artificial Intell. Lab., Cambridge, MA, Rep. Al-TR-720, 1983.
15. Казакова Н.Ф., Фразе-Фразенко О.О. Синтез методу виділення контурів у системах ідентифікації на основі усереднення перепадів яскравості // Інформаційна безпека. – 2013. – №2 (10). – с.48-57. – С.48.

16. Гороховатський В.О., Пупченко Д.В., Солодченко К. Г. Аналіз властивостей, характеристик та результатів застосування новітніх детекторів для визначення особливих точок зображення // Системи управління, навігації та зв'язку. – 2018. – Вип. 1. – С. 93-98. – С. 7.
17. Яровий А., Поперечний С. Аналіз методів та моделей розпізнавання зображень дистанційного зондування землі. – 2014. – ст.9. [Електронний ресурс]. Режим доступу: URL: https://ir.lib.vntu.edu.ua/bitstream/handle/123456789/13393/97-100.pdf?sequence=1&isAllowed=y
18. Ghamisi P. Advanced Spectral Classifiers for Hyperspectral Images: A review / P. Ghamisi, J. Plaza, Y. Chen et al // IEEE Geoscience and Remote Sensing Magazine. – Vol. 5, N 1. – 2017. – P. 8–32.
19. Fukunaga K. Introduction to statistical pattern recognition / K. Fukunaga // Academic press, 2013. – 591 p.
20. Maulik U. Remote Sensing Image Classification: A survey of support-vectormachine-based advanced techniques / U. Maulik, D. Chakraborty // IEEE Geoscience and Remote Sensing Magazine. – Vol. 5, N 1. – 2017. – P. 33–52.
21. Rumelhart D.E. Learning representations by back-propagating errors / D.E. Rumelhart, G.E. Hinton, R.J. Williams // Cognitive modeling. – Vol. 5, N 3. – 1988. – P. 213-220.
22. Amari S.I. Statistical theory of learning curves under entropic loss criterion / S.I. Amari, N. Murata // Neural Computation. – Vol. 5, N 1. – 1993. – P. 140–153.
23. Matas J., Chum O., Urban M., Pajdla T. Robust wide baseline stereo from maximally stable extremal regions // BMVC. – 2002. – p. 384-393. – P. 390.
24. Hopcroft J.E., Tarjan R.E. Algorithm 447: Efficient algorithms for graph manipulation. Communications of the ACM. – 1973. – p. 372–378.
25. Neumann L., Matas J. A method for text localization and recognition in real-world images // In Proceedings of the 10th Asian Conference on Computer Vision. Berlin, Heidelberg. – 2011. – Vol. III. – p. 770-783.
26. Bradsky G. Learning OpenCV // O'Reilly / Kaehler A. – O'Reilly Media. – 2008 – 580 p.
27. Tessaract OCR [Електронний ресурс]. Режим доступу: URL: https://github.com/tesseract-ocr/. – Назва з екрану.
28. WebAssembly [Електронний ресурс]. Режим доступу: URL: https://webassembly.org/. – Назва з екрану.
29. JavaScript [Електронний ресурс]. Режим доступу: URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript. – Назва з екрану.
30. Emscripten [Електронний ресурс]. Режим доступу: URL: https://emscripten.org/. – Назва з екрану.
31. Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт / Уклад. В. О. Козловський – Вінниця: ВНТУ, 2012. – 22 с.

32. Алексєєв А. Алгоритм розпізнавання символів на основі структурного підходу / А. Алексєєв, В. Заяць, Д. Іванов // Вісник Нац. ун-ту „Львівська політехніка‖ «Комп‘ютерна інженерія та інформаційні технології». – 2002. – №468. – С.129 – 133
33. Методичні вказівки до виконання студентами-магістрантами економічної частини магістерських кваліфікаційних робіт / Уклад. В. О. Козловський – Вінниця: ВНТУ, 2012. – 25 с.

Додаток А

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 «Методи та програмні засоби розпізнавання зображень» за спеціальністю

121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____ к.т.н., доц. каф. ПЗ Хошаба О.М.
"____"_____2019 р.
Виконав:
_____студент гр.2ПІ-18м Ель Жеддауї Х.

"____"_____2019 р.

Вінниця – 2019 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Методи та програмні засоби розпізнавання зображень».

Галузь застосування - Машинне навчання.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою роботи є підвищення продуктивності розпізнавання тексту за підтримки й використання моделі машинного навчання, що здатна розпізнавати текст у зображеннях на веб-сторінках, і розробки бібліотеки JavaScript для застосування мережі у браузері.

Призначення роботи – розробка методів і засобів розпізнавання тексту в зображенні.

3 Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Ghamisi P. Advanced Spectral Classifiers for Hyperspectral Images: A review / P. Ghamisi, J. Plaza, Y. Chen et al // IEEE Geoscience and Remote Sensing Magazine. – Vol. 5, N 1. – 2017. – P. 8–32.

2. Fukunaga K. Introduction to statistical pattern recognition / K. Fukunaga // Academic press, 2013. – 591 p.

3. Maulik U. Remote Sensing Image Classification: A survey of support-vectormachine-based advanced techniques / U. Maulik, D. Chakraborty // IEEE Geoscience and Remote Sensing Magazine. – Vol. 5, N 1. – 2017. – P. 33–52.

24. Matas J., Chum O., Urban M., Pajdla T. Robust wide baseline stereo from maximally stable extremal regions // BMVC. – 2002. – p. 384-393. – P. 390.

4. Технічні вимоги

базові методи детекції – методи MSER, салієнтності; методи розпізнавання – сегментація слів; вихідні дані для розпізнавання – навчена нейронна мережа MSER; навчена нейронна мережа Saliency; навчений класифікатор слів; словник образів; словник слів;

5. Конструктивні вимоги.

Конструкція пристрою повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

6. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз наявних методів і засобів розпізнавання зображень, зокрема розпізнавання тексту в зображенні	

2	Розробка методу детекції тексту в зображенні та його розпізнавання	
3	Розробка програмних компонентів моделі машинного навчання	
4	Розробка програмних компонентів бібліотеки JavaScript для застосування моделі машинного навчання у браузері	
5	Економічна частина	

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

Додаток Б

Лістинг програми

text_detection.py

```
# import the necessary packages
from imutils.object_detection import non_max_suppression
import numpy as np
import argparse
import time
import cv2

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", type=str,
	help="path to input image")
ap.add_argument("-east", "--east", type=str,
	help="path to input EAST text detector")
ap.add_argument("-c", "--min-confidence", type=float, default=0.5,
	help="minimum probability required to inspect a region")
ap.add_argument("-w", "--width", type=int, default=320,
	help="resized image width (should be multiple of 32)")
ap.add_argument("-e", "--height", type=int, default=320,
	help="resized image height (should be multiple of 32)")
args = vars(ap.parse_args())

# load the input image and grab the image dimensions
image = cv2.imread(args["image"])
orig = image.copy()
(H, W) = image.shape[:2]

# set the new width and height and then determine the ratio in change
# for both the width and height
(newW, newH) = (args["width"], args["height"])
rW = W / float(newW)
rH = H / float(newH)

# resize the image and grab the new image dimensions
image = cv2.resize(image, (newW, newH))
(H, W) = image.shape[:2]

# define the two output layer names for the EAST detector model that
# we are interested -- the first is the output probabilities and the
# second can be used to derive the bounding box coordinates of text
layerNames = [
	"feature_fusion/Conv_7/Sigmoid",
```

```
        "feature_fusion/concat_3"]

# load the pre-trained EAST text detector
print("[INFO] loading EAST text detector...")
net = cv2.dnn.readNet(args["east"])

# construct a blob from the image and then perform a forward pass of
# the model to obtain the two output layer sets
blob = cv2.dnn.blobFromImage(image, 1.0, (W, H),
        (123.68, 116.78, 103.94), swapRB=True, crop=False)
start = time.time()
net.setInput(blob)
(scores, geometry) = net.forward(layerNames)
end = time.time()

# show timing information on text prediction
print("[INFO] text detection took {:.6f} seconds".format(end - start))

# grab the number of rows and columns from the scores volume, then
# initialize our set of bounding box rectangles and corresponding
# confidence scores
(numRows, numCols) = scores.shape[2:4]
rects = []
confidences = []

# loop over the number of rows
for y in range(0, numRows):
        # extract the scores (probabilities), followed by the geometrical
        # data used to derive potential bounding box coordinates that
        # surround text
        scoresData = scores[0, 0, y]
        xData0 = geometry[0, 0, y]
        xData1 = geometry[0, 1, y]
        xData2 = geometry[0, 2, y]
        xData3 = geometry[0, 3, y]
        anglesData = geometry[0, 4, y]

        # loop over the number of columns
        for x in range(0, numCols):
                # if our score does not have sufficient probability, ignore it
                if scoresData[x] < args["min_confidence"]:
                        continue

                # compute the offset factor as our resulting feature maps will
                # be 4x smaller than the input image
                (offsetX, offsetY) = (x * 4.0, y * 4.0)

                # extract the rotation angle for the prediction and then
                # compute the sin and cosine
```

```
                angle = anglesData[x]
                cos = np.cos(angle)
                sin = np.sin(angle)

                # use the geometry volume to derive the width and height of
                # the bounding box
                h = xData0[x] + xData2[x]
                w = xData1[x] + xData3[x]

                # compute both the starting and ending (x, y)-coordinates for
                # the text prediction bounding box
                endX = int(offsetX + (cos * xData1[x]) + (sin * xData2[x]))
                endY = int(offsetY - (sin * xData1[x]) + (cos * xData2[x]))
                startX = int(endX - w)
                startY = int(endY - h)

                # add the bounding box coordinates and probability score to
                # our respective lists
                rects.append((startX, startY, endX, endY))
                confidences.append(scoresData[x])

# apply non-maxima suppression to suppress weak, overlapping bounding
# boxes
boxes = non_max_suppression(np.array(rects), probs=confidences)

# loop over the bounding boxes
for (startX, startY, endX, endY) in boxes:
        # scale the bounding box coordinates based on the respective
        # ratios
        startX = int(startX * rW)
        startY = int(startY * rH)
        endX = int(endX * rW)
        endY = int(endY * rH)

        # draw the bounding box on the image
        cv2.rectangle(orig, (startX, startY), (endX, endY), (0, 255, 0), 2)

# show the output image
cv2.imshow("Text Detection", orig)
cv2.waitKey(0)
```

static_saliency.py

```
# import the necessary packages
import argparse
import cv2

# construct the argument parser and parse the arguments
```

```
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
	help="path to input image")
args = vars(ap.parse_args())

# load the input image
image = cv2.imread(args["image"])

# initialize OpenCV's static saliency spectral residual detector and
# compute the saliency map
saliency = cv2.saliency.StaticSaliencySpectralResidual_create()
(success, saliencyMap) = saliency.computeSaliency(image)
saliencyMap = (saliencyMap * 255).astype("uint8")
cv2.imshow("Image", image)
cv2.imshow("Output", saliencyMap)
cv2.waitKey(0)

# initialize OpenCV's static fine grained saliency detector and
# compute the saliency map
saliency = cv2.saliency.StaticSaliencyFineGrained_create()
(success, saliencyMap) = saliency.computeSaliency(image)

# if we would like a *binary* map that we could process for contours,
# compute convex hull's, extract bounding boxes, etc., we can
# additionally threshold the saliency map
threshMap = cv2.threshold(saliencyMap.astype("uint8"), 0, 255,
	cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

# show the images
cv2.imshow("Image", image)
cv2.imshow("Output", saliencyMap)
cv2.imshow("Thresh", threshMap)
cv2.waitKey(0)
```

get_worder.js

```
module.exports = (_options = {}) => {
  const id = getId('Worker', workerCounter);
  const {
    logger,
    ...options
  } = resolvePaths({
    ...defaultOptions,
    ..._options,
  });
  const resolves = {};
  const rejects = {};
  let worker = spawnWorker(options);
```

```
  workerCounter += 1;

  const setResolve = (action, res) => {
    resolves[action] = res;
  };

  const setReject = (action, rej) => {
    rejects[action] = rej;
  };

  const startJob = ({ id: jobId, action, payload }) => (
    new Promise((resolve, reject) => {
      log(`[${id}]: Start ${jobId}, action=${action}`);
      setResolve(action, resolve);
      setReject(action, reject);
      send(worker, {
        workerId: id,
        jobId,
        action,
        payload,
      });
    })
  );

  const load = jobId => (
    startJob(createJob({
      id: jobId, action: 'load', payload: { options },
    }))
  );
```

Main.js

```
const createWorker = require('./createWorker');
const recognize = async (image, langs, options) => {
  const worker = createWorker(options);
  await worker.load();
  await worker.loadLanguage(langs);
  await worker.initialize(langs);
  return worker.recognize(image)
    .finally(async () => {
      await worker.terminate();
    });
};

const detect = async (image, options) => {
  const worker = createWorker(options);
  await worker.load();
  await worker.loadLanguage('osd');
  await worker.initialize('osd');
```

```
  return worker.detect(image)
    .finally(async () => {
      await worker.terminate();
    });
};

module.exports = {
  recognize,
  detect,
};
```

get_core.js

```
          module.exports = (corePath, res) => {
            if (typeof global.AIAcessCore === "undefined") {
              res.progress({ status: "loading core", progress: 0 });
              global.importScripts(corePath);
              /*
               * Depending on whether the browser supports WebAssembly
               */
              if (
                typeof global.AIAcessCore.WASM !== "undefined" &&
                typeof WebAssembly === "object"
              ) {
                global.AIAcessCore = global.AIAcessCore.CoreWASM;
              } else if (typeof global.AIAcessCore.CoreASM !== "undefined") {
                global.AIAcessCoreCore = global.AIAcessCoreCoreASM;
              } else {
                throw Error("Failed to load Core");
              }
              res.progress({ status: "loading core", progress: 1 });
            }
            return global.AIAcessCoreCore;
          };
```

Recognition.py

```
# import the necessary packages
from PIL import Image
import pytesseract
import argparse
import cv2
import os


# construct the argument parse and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
        help="path to input image to be OCR'd")
```

```
ap.add_argument("-p", "--preprocess", type=str, default="thresh",
	help="type of preprocessing to be done")
args = vars(ap.parse_args())

# load the example image and convert it to grayscale
image = cv2.imread(args["image"])
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

cv2.imshow("Image", gray)

# check to see if we should apply thresholding to preprocess the
# image
if args["preprocess"] == "thresh":
	gray = cv2.threshold(gray, 0, 255,
		cv2.THRESH_BINARY | cv2.THRESH_OTSU)[1]

# make a check to see if median blurring should be done to remove
# noise
elif args["preprocess"] == "blur":
	gray = cv2.medianBlur(gray, 3)

# write the grayscale image to disk as a temporary file so we can
# apply OCR to it
filename = "{}.png".format(os.getpid())
cv2.imwrite(filename, gray)

# load the image as a PIL/Pillow image, apply OCR, and then delete
# the temporary file
pytesseract.pytesseract.tesseract_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'
text = pytesseract.image_to_string(Image.open(filename))
os.remove(filename)
print(text)

# show the output images
# cv2.imshow("Image", image)
cv2.imshow("Output", gray)
cv2.waitKey(0)
```

ДОДАТОК Г

Ілюстративний матеріал до захисту

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

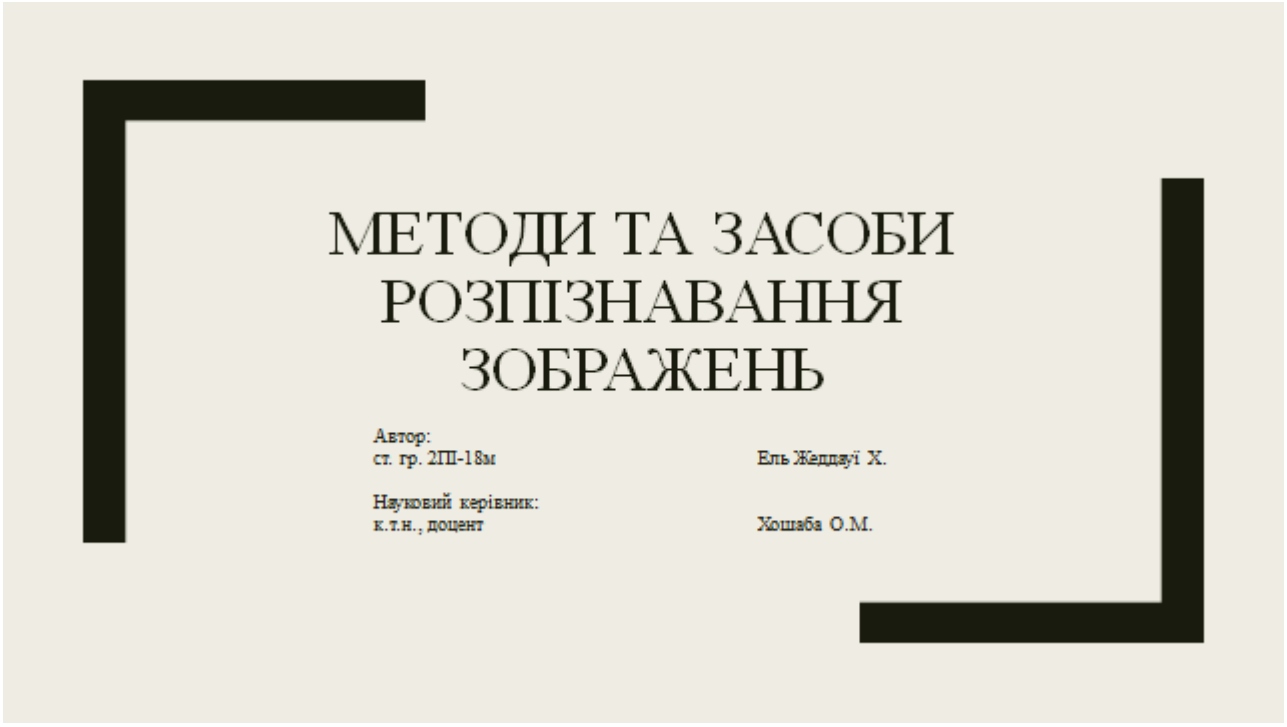
Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доцент кафедри ПЗ_____ О.М. Хошаба

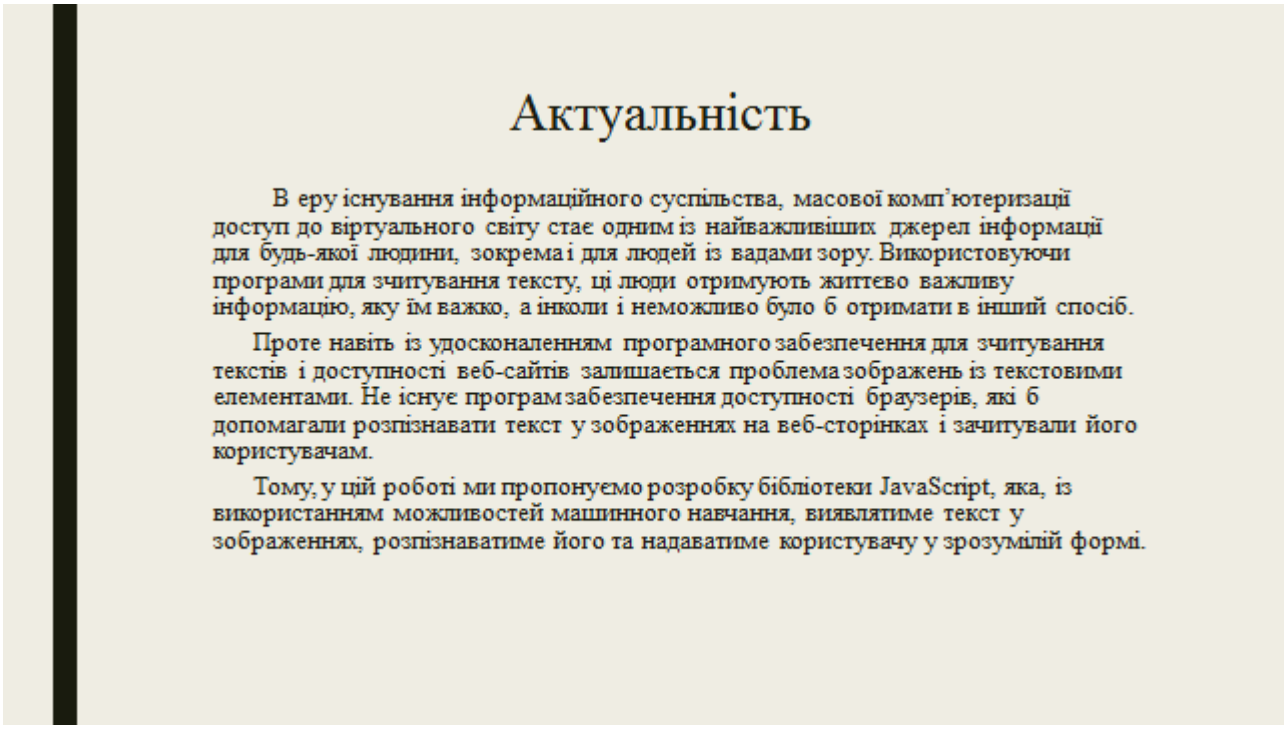
Рецензент, д.т.н, професор, зав. кафедри КН_____ Я.В. Іванчук

Нормоконтроль, к. т. н., доцент кафедри ПЗ _____В. В. Войтко

Виконавець, студент групи 2ПІ-18м _____ Х. Ель Жеддауї



Слайд 1 – Тема, автор, науковий керівник



Слайд 2 – Актуальність

Мета, об’єкт та предмет дослідження

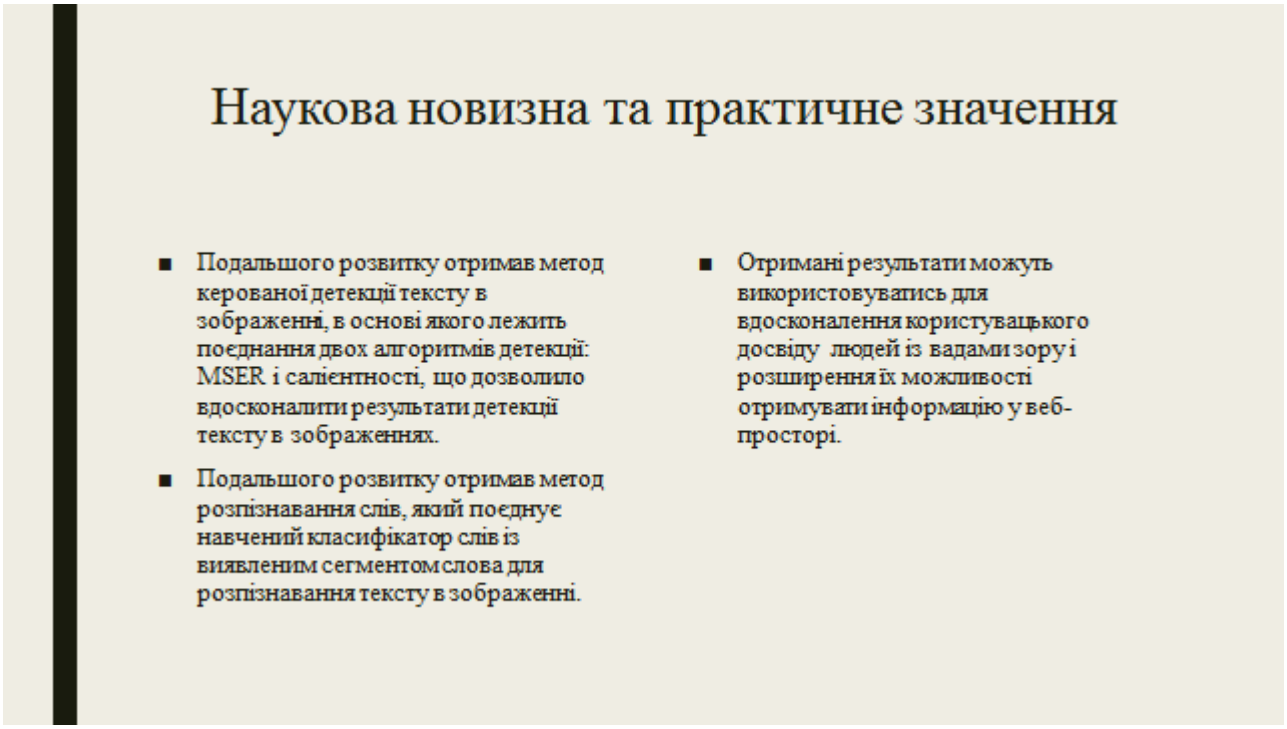
- **Мета** – підвищення продуктивності розпізнавання тексту за підтримки й використання моделі машинного навчання, що здатна розпізнавати текст у зображеннях на веб-сторінках, і розробки JavaScript library для застосування мережі у браузері.
- **Об’єкт** – процес розпізнавання тексту в зображеннях у браузері.
- **Предмет** – методи та засоби розпізнавання зображень і розпізнавання тексту в зображенні.

Слайд 3 – Мета, об’єкт та предмет

Задачі

- - проаналізувати наявні методи й засоби розпізнавання зображень, зокрема розпізнавання тексту в зображенні;
- - розробити методи детекції тексту в зображенні та його розпізнавання;
- - розробити програмні компоненти моделі машинного навчання, що здійснюватиме розпізнавання тексту в зображеннях на основі визначених методів;
- - розробити програмні компоненти бібліотеки JavaScript для застосування моделі машинного навчання у браузері.
- - провести експериментальне дослідження розроблених програмних засобів.

Слайд 4 – Задачі



Слайд 5 – Наукова новизна та практичне значення

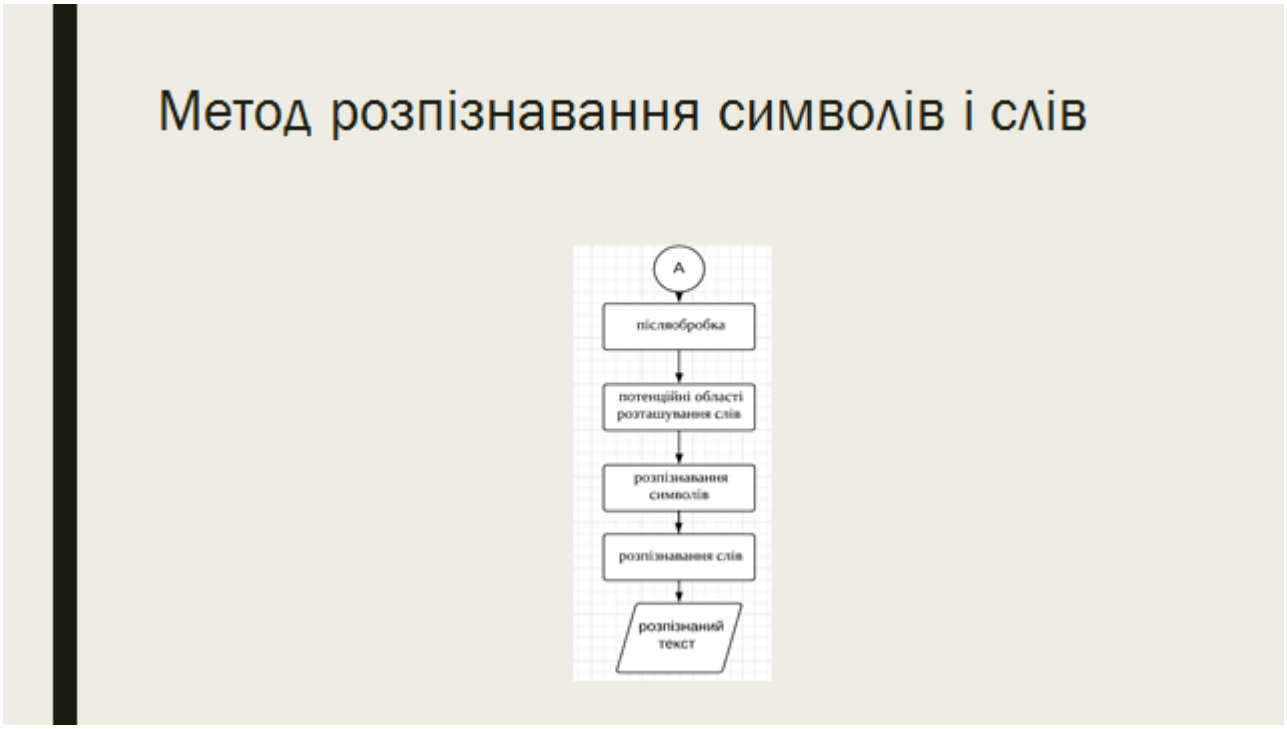
Слайд 6 – Існуючі аналоги

Слайд 7 – Використані технології

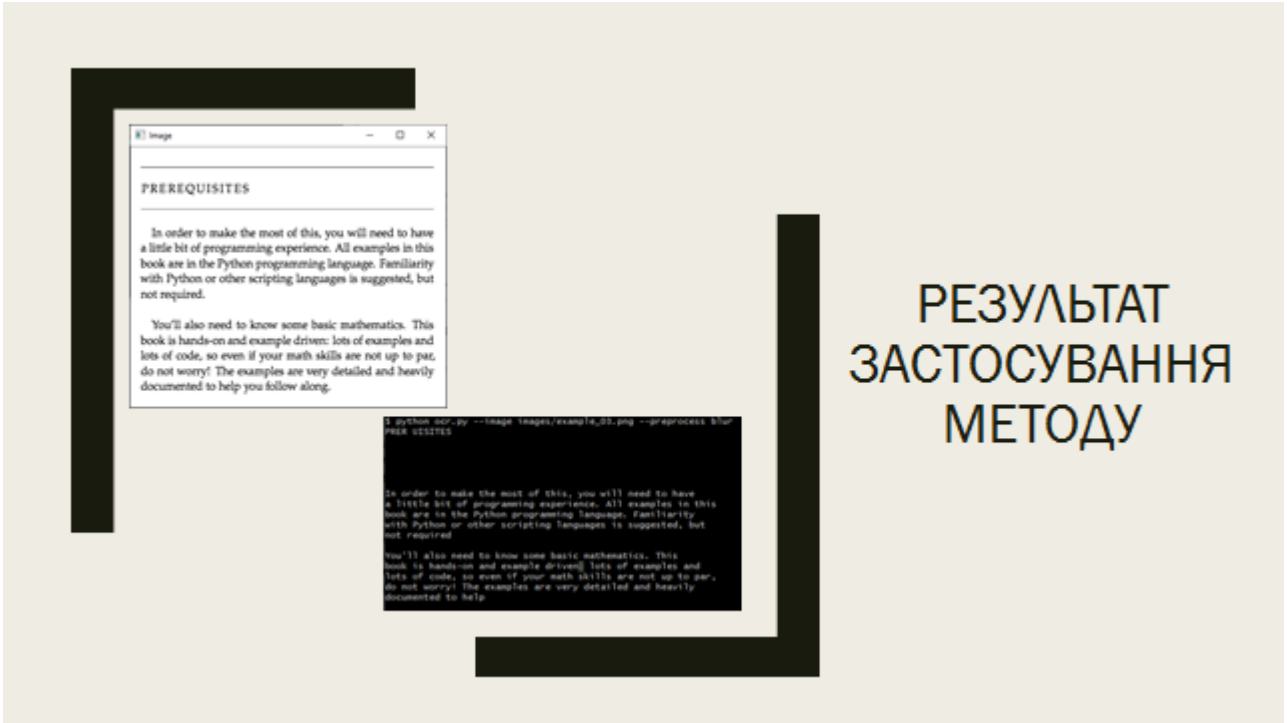
Слайд 8 – Модель системи

Слайд 9 – Метод детекції символів і слів

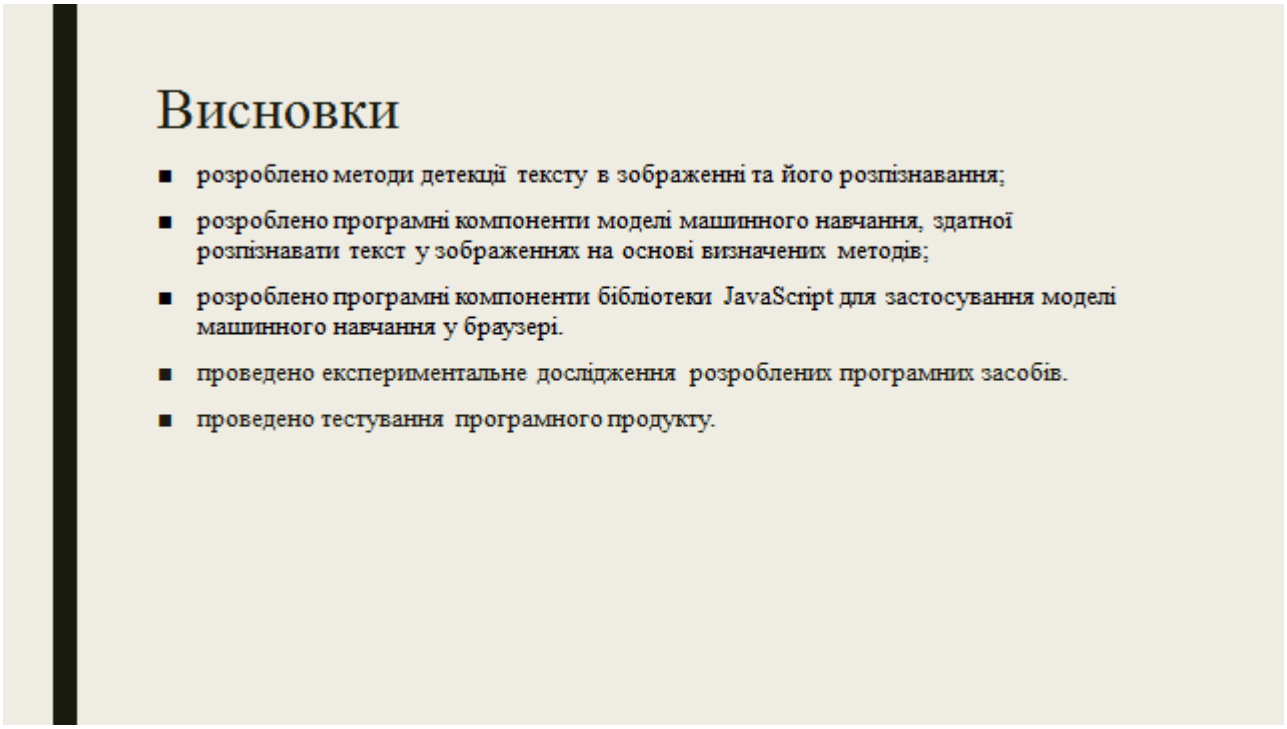
Слайд 10 – Результат застосування методу



Слайд 11 – Метод розпізнавання символів і слів



Слайд 12 – Результат застосування методу



Слайд 13 – Висновки