Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра комп’ютерних наук

Пояснювальна записка

до магістерської кваліфікаційної роботи
на тему:

Інформаційна технологія розпізнавання рукописного тексту за допомогою згорткової нейронної мережі

Виконав: студент 2 курсу групи 2КН-19м
спеціальності 122 «Комп'ютерні науки»

Добера Р. О.
(прізвище та ініціали)

Керівник к.т.н., доц. Колодний В. В.
(прізвище та ініціали)

Рецензент к.т.н., доц. Романюк О. В.
(прізвище та ініціали)

Вінниця - 2020 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра комп’ютерних наук

ЗАТВЕРДЖЕНО
Завідувач кафедри КН
проф., д.т.н. Яровий А. А.
«____»_____2020 р.

З А В Д А Н Н Я

на магістерську кваліфікаційну роботу на здобуття кваліфікації магістра наук зі спеціальності: 122 «Комп'ютерні науки»
(шифр – назва спеціальності)

08-22.МКР.018.19.000.ПЗ

Магістранта групи __2КН-19м__ _Добери Романа Олександровича
(назва групи) (прізвище, ім’я і по батькові)

Тема магістерської кваліфікаційної роботи: «Інформаційна технологія розпізнавання рукописного тексту за допомогою згорткової нейронної мережі»

Вхідні дані: кількість входів нейронної мережі - 4; кількість виходів нейронної мережі – 5; кількість шарів нейронної мережі - 9; набір зображень рукописних текстів; мова програмування та середовище розробки повинні забезпечити маніпулювання та обробку даних.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: Граф-схема алгоритму роботи програми розпізнавання рукописного тексту, архітектура нейронної мережі, структура базової LSTM одиниці; схема моделі нейронної мережі, вид програмної реалізації імпульсної нейронної мережі, приклади роботи програми.

2. Текстова (пояснювальна записка): вступ, аналіз предметної області розпізнавання рукописного тексту, розробка інформаційної технології розпізнавання рукописного тексту на основі згорткової нейронної мережі, програмна реалізація інформаційної технології розпізнавання рукописного тексту, економічна частина, висновки, перелік використаних джерел, додатки.

КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз предметної області розпізавання рукописного тексту			Аналітичний огляд літературних джерел, задачі досліджень, розділ 1
2	Розробка методу та інформаційної технології розпізнавання рукописного тексту			Метод, інформаційна технологія, розділ 2
3	Програмна реалізація розробленої інформаційної технології, тестування			Програмне забезпечення, розділ 3
4	Підготовка економічної частини			розділ 4
5	Апробація та/або впровадження результатів дослідження			тези доповідей
6	Оформлення пояснювальної записки, графічного матеріалу та презентації			Пояснювальна записка, графічний матеріал, презентація

Консультанти з окремих розділів магістерської кваліфікаційної роботи

1. Науковий керівник _____ канд. техн. наук, доцент
(підпис) наук. ступінь, вчене звання (посада)
_____ В. В. Колодний
ініціали та прізвище

“____” _____20___ р.

2. Економічна частина _____ канд. екон. наук, доцент кафедри ЕПВМ
(підпис) наук. ступінь, вчене звання (посада)
_____ М. В. Бальзан
ініціали та прізвище

“____” _____20___ р.

Дата попереднього захисту роботи “____” _____20___ р.

Рецензент _____ _____
(підпис) наук. ступінь, вчене звання (посада)
_____
ініціали та прізвище

Завдання видав
науковий керівник _____ канд. техн. наук, доцент
(підпис) наук. ступінь, вчене звання (посада)
_____ В. В. Колодний
ініціали та прізвище

“____” _____20___ р.

Завдання отримав магістрант _____ Р.О. Добера
(підпис) ініціали та прізвище

“____” _____20___ р.

АНОТАЦІЯ

У роботі розглянуто проблему в області автоматизованого розпізнавання писемних символів на зображенні, показано основні особливості існуючих рішень та додатків, їх переваги та недоліки.

Для перенесення писемного тексту з будь-якого паперу на електронний пристрій розроблена нейронна мережа з розпізнавання писемних символів на зображенні. Дана система дозволяє зекономити час на перенесення тексту і може бути використані у сферах перекладу тексту з однієї мови на іншу.

Визначено завдання для системи розпізнавання писемних символів на зображенні за допомогою нейронної мережі, відібрано нейронну мережу та спосіб навчання, які найбільш підходять для даної задачі.

Ключові слова: нейронна мережа, розпізнавання рукописного тексту, навчання, зображення, алгоритм.

ABSTRACT

The paper considers the problem in the field of automated recognition of written characters in the image, shows the main features of existing solutions and applications, their advantages and disadvantages.

A neural network for recognizing written characters in an image has been developed to transfer written text from any paper to an electronic device. This system saves time on text transfer and can be used in the field of text translation from one language to another.

The tasks for the system of recognition of written symbols on the image by means of a neural network are defined, the neural network and a way of training which are most suitable for the given task are selected.

Key words: neural network, handwriting recognition, learning, image, algorithm.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

CNN (Convolutional Neural Network) – згорткова нейронна мережа

RNN (Recurrent Neural Network) – рекурентна нейронна мережа

CRNN (Convolutional Recurrent Neural Network) - згорткова рекурентна нейронна мережа

LSTM (Long-Short Term Memory) - довго-короткочасна пам’ять

CTC (Connectionist Temporal Classification) – класифікація по рейтингу

ВСТУП

Актуальність теми

Одним з найбільш актуальних і затребуваних завдань обробки документів і введення даних в комп'ютер є задача розпізнавання тексту, яка є окремим випадком завдання розпізнавання образів.

Розпізнавання образів - науковий напрям в теоретичній інформатиці, прикладній статистиці та суміжних дисциплінах, що розвиває принципи і методи класифікації об'єктів різної природи.

Автоматичне розпізнавання друкованих текстів є досить добре дослідженою областю інформатики та прикладної математики. Напрямок, пов'язаний з розпізнаванням рукописного тексту, досліджено набагато гірше. На сьогоднішній день існує два відпрацьованих підходи до вирішення цього завдання: системи розпізнавання форм, заповнених друкованими літерами від руки і розпізнавання роздільних рукописних букв, написаних особливим пером на спеціальному екрані. Методи, що реалізують читання комп'ютером злитих букв (звичайного рукописного письма), на практиці зустрічаються дуже рідко.

У студентів виникає ситуація коли потрібно перекинути конспект лекцій іншим колегам по групі для вивчення навчального матеріалу перед екзаменом. Студенти пересилають інформацію, фотографуючи кожну сторінку зошиту и скидуючи її через соціальні мережі своїм колегам, або просто передають зошит, щоб інші студенти змогли переписати конспект. На жаль, у вказаних випадках іншим студентам буває важко розпізнати почерк людини, яка писала конспект.

У туристів буває ситуація, коли не знаючи мову країни, яку вона відвідує, не може розпізнати навігаційні тексти в транспорті, на вулиці та в інших місцях, де англійська мова не використовується через непопулярність країни для туристів, але ця інформація буває корисною та важливою для самих відвідувачів іноземної держави. Особливо це може стосуватися країн

світу, де замість латинських або кирилицьких літер застосовують ієрогліфи, наприклад як в Китаї або Японії. Для такої ситуації було б чудово сфотографувати напис та надіслати його до сервісу, де цей напис розпізнають, перекладуть в зрозумілий для людини вигляд і вона не буде мати проблем у перебуванні в іноземній країні.

Розпізнавання писемних символів за допомогою нейронної мережі могло би бути корисним для психологічних досліджень взаємозв'язку психологічного стану людини з її почерком. Як можна дізнатися у почерку людини важливим є розмір нахил, напрямок літер, сила натиску на ручку та характер самого напису букв. Якщо об'єднати нейронну мережу розпізнавання писемних символів і бази даних почерку багатьох людей, то можна продивитися цікаву статистику і спробувати підтвердити кореляцію почерку з характером.

Для державних служб дуже важливим є елемент документообігу і загалом система документації. На будь-яку дію зазвичай вказується документ, який надає право тій або іншій людини працювати над питаннями, які вона повинна вирішувати. Одним із етапом такої роботи є написання заяви. В Україні дуже часто записують заяви у письмовому вигляді і для того, щоб перевести заяву в електронний вигляд зазвичай державні установи просто зберігають вказану заяву у себе і вона має шанси загубитися. Після цього державній установі ще потрібно доказати, що заява була написана. Для вирішення цієї задачі може пригодитися робота нейронної мережі, яка переводить текст заяви в електронний вигляд і надаєзмогу вирішити ще одно питання української бюрократії та зробити прозорий документообіг ще ближче до реальності.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп'ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних

інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета та завдання дослідження. Метою даної роботи є підвищення достовірності розпізнавання рукописного тексту.

Для досягнення мети необхідно розв’язати наступні завдання:

- дослідити постановку задачі розпізнавання образів
- дослідити основні алгоритми розпізнавання символів
- виконати огляд існуючих підходів до розпізнавання машинних і рукописних символів
- зібрати необхідну кількість тренувальних данних
- розробити ПЗ для підготовки данних до навчання
- розробити згорткову нейронну мережу й оптимізувати її для розпізнавання

Об’єктом дослідження даної роботи є процес розпізнавання рукописного тексту.

Предметом дослідження є інформаційна технологія оптичного розпізнавання у галузі глибинного навчання та штучного інтелекту.

Методи дослідження. В ході дослідження використовувався нейромережевий метод для розпізнавання тексту.

Для програмної реалізації було використано мову програмування Python разом з бібліотеками для глибинного навчання.

Наукова новизна отриманих результатів.

Розроблено метод розпізнавання рукописного тексту за допомогою згорткової нейронної мережі, що дозволило підвищити точність розпізнавання за відносно простої архітектури моделі.

У роботі розглядається альтернативний метод розпізнавання. Він не має на меті розбиття тексту на літери – натомість, він передбачає розпізнавання цілих слів. Це вводить певні обмеження на розмір словника розпізнавання, проте одночасно підвищує точність розпізнавання та швидкість навчання.

Практичне значення отриманих результатів.

Практичне значення отриманих результатів полягає в тому, що завдяки отриманим в роботі теоретичним положенням розроблено програмні засоби розпізнавання рукописного тексту, які можна використовувати в будь-яких сферах діяльності, пов’язаних з розпізнаванням рукописних символів.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується строгістю постановки задач, строгим виведенням аналітичних співвідношень, правильним застосуванням математичних методів під час доведення наукових положень, порівняння результатів роботи з відомими та збіжністю результатів математичного моделювання з отриманими результатами під час тестування програмного засобу.

Особистий внесок магістранта. Усі результати наведені у магістерській кваліфікаційній роботі отримані самостійно.

Апробація результатів роботи.

Результати роботи були апробовані на XLIX Науково-технічній конференції факультету інформаційних технологій та комп’ютерної інженерії (2020).

Публікації.

За результатами магістерської кваліфікаційної роботи опубліковано 1 тезу доповідей конференцій [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ

1.1 Постановка задачі розпізнавання рукописного тексту

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

Розпізнавання рукописного вводу – це здатність розпізнавати текст за допомогою системи, яка отримує вхідні дані від сенсорного екрана, електронного пера, сканера, зображень або паперових документів [2]. Система розпізнавання рукописного вводу в автономному режимі є мистецтвом ідентифікації слова з зображення. Як ми знаємо, кожна людина має свій різний стиль письма, тому дуже важко визнати правильні рукописні символи та цифри. Система розпізнавання рукописного тексту розроблена для досягнення точності та надійності роботи.

Отже, розпізнавання рукописного тексту є найбільш складною областю, якщо розпізнавати зображення та образ. Розпізнавання рукописного тексту дуже корисно в реальному світі та вирішує багато практичних

проблем, наприклад, таких, як аналіз документації, інтерпретація поштової адреси, обробка банківських чеків, перевірка підпису, поштової адреси тощо. Використовуються декілька підходів, як розпізнавання рукописного вводу в режимі онлайн, так і в автономному режимі, такі як статистичні методи, структурні методи, нейронна мережа та синтаксичні методи. Деякі системи розпізнавання визначають штрихи інші застосовують розпізнавання на одному символі або цілі слова.

Незважаючи на велику кількість інструментів для написання технологічних текстів, багато людей все-таки вирішують робити свої нотатки традиційно: за допомогою ручки та паперу. Проте рукописний текст має недоліки. Важко ефективно зберігати і отримувати доступ до фізичних документів, здійснювати їх пошук та ділитися ними з іншими.

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

В даній роботі будуть досліджуватися методи та засоби розпізнавання рукописного тексту на основі згорткової нейронної мережі. Буде розроблено програмний засіб для вирішення задачі розпізнавання рукописного тексту. Таке програмне забезпечення може стати корисним помічником як у повсякденному житті так і у професійній діяльності.

Вхідні дані являють собою зображення розширення png або jpg.

Вихідні дані – текст, розташований в текстовому полі.

1.2 Глибинне навчання

Глибинне навчання — це галузь машинного навчання, що базується на наборі алгоритмів, які моделюють високорівневі абстракції в даних, застосовуючи глибинний граф разом із кількома обробними шарами, що побудовано з кількох лінійних або нелінійних перетворень.

Методи глибинного навчання є методами навчання подань з багатьма їх рівнями, отриманими через об'єднання простих нелінійних модулів, кожен з яких переводить подання з одного рівня (починаючи з початкових даних) на вищий, більш абстрактний рівень. Об'єднання достатньої кількості таких трансформувань забезпечує навчання функції будь-якої складності. Ключовим аспектом глибинного навчання є те, що шари особливостей не розробляються експертами в тій чи іншій галузі, а автоматично виокремлюються із вхідних даних. Метод групового врахування аргументів (Group Method of Data Handling (GMDH)), запропонований українським науковцем Олексієм Івахненком, є одним з перших прикладів глибинного навчання для багатошарових нейронних мереж (у його працях розглядається мережа з 8 шарами), який використовує поліноми Колмогорова–Габора [3]. Інші архітектури глибинного навчання, побудовані зі штучних нейронних мереж, беруть свій початок з неокогнітрону, запропонованого в 1980 р. Куніхіко Фукусімою [4]. Неокогнітрон — це перша двовимірна згорткова нейронна мережа (Convolutional Neural Network (CNN)). Проте в цій мережі ваги налаштовувались не за допомогою методу зворотного поширення помилок, а локально за допомогою підходу «переможець отримує все» (winner-take-all). Тому цей тип мереж належить до навчання без учителя [5]. У 1989 р. метод зворотного поширення помилки успішно застосував Ян Лекун до глибинної двовимірної згорткової нейронної мережі в задачі розпізнавання рукописних цифр. Після цього багато уваги приділялось розвитку глибинних мереж переконань (Deep Belief Networks (DBN)), які

являють собою стек обмежених машин Больцмана (Restricted Boltzmann Machines (RBMs)), які, у свою чергу, є машиною Больцмана з тим обмеженням, що нейрони мусять формувати двочастковий граф: з'єднань між вузлами в межах групи немає. Почали розвиватись також рекурентні глибинні мережі для оброблення послідовностей даних, що залежать від часу. Проте в 1991 р. було помічено, що дуже глибокі нейронні мережі, особливо рекурентні нейронні мережі, складно навчити методом зворотного поширення помилки . Ця проблема називається проблемою зникнення або вибуху градієнта (Vanishing or Exploding Gradients): з використанням стандартних функцій активації сукупна помилка сигналу зворотного поширення або дуже швидко прямує до нуля, або дуже швидко експоненційно зростає. Для подолання цієї проблеми в 1997 р. було запропоновано рекурентні мережі з моделлю довгої короткочасної пам'яті (Long Short-Term Memory (LSTM)) [6]. Після разового успішного використання Яном Лекуном згорткових нейронних мереж вони не здобули популярності. І лише в 2012 р. А. Крижевський та інші науковці відновили інтерес до згорткових нейронних мереж після того, як показали вражаюче високу точність класифікації зображень на змаганні ImageNet Large Scale Visual Recognition Challenge (ILSVRC) . На цьому змаганні нейронні мережі застосовувалися до набору даних, що налічував понад мільйон зображень з інтернету, які містили більше ніж 1000 різних класів. Цей успіх розпочав революцію в напрямку комп'ютерного зору із застосуванням глибинних мереж у різноманітних напрямах.

1.3 Методи оптичного розпізнавання образів

Завдання оптичного розпізнавання образів має на меті виявлення об'єктів або визначення їх властивостей чи класів за заданим візуальним зображенням. Образ у даному контексті означає певну групу об'єктів,

об’єднаних за спільними ознаками. Наприклад, образом може бути обличчя людини, тварина певного виду, автомобіль, фізичний предмет, літера або слово і т.д. Тобто залежно від контексту задачі образ може позначати досить широкий клас об’єктів або щось більш конкретне.

Вирішальне правило – це методика, використовувана для того, щоб віднести елемент до якогось образу. Спосіб, яким визначається відстань між елементами множини, називається метрика [7]. Відстань між елементами зворотньо пропорційна подібності розпізнаваних об'єктів. Елементи прийнято задавати як числовий набір, а метрику –як функцію. Результативність програми залежить від правильного підбору образів тареалізації метрики. При використанні одного алгоритму з різними метриками буде отримано різні показники ефективності.

Виділяють три основні методи розпізнавання образів:

1) метод перебору;

2) аналіз характеристик образу;

3) використання ШНМ.

За методом перебору існує велика база даних, що містить багато модифікацій зображень та відповідні їм класи об'єктів. Перебираючи усю базу зразків, можна порівняти кожний з екземпляром, що аналізується, та обрати таку відповідність, за якої різниця між двома зразками буде найменшою. Графічні об'єкти в базі можуть бути трансформовані за масштабом, нахилом, зміщенням і т.д. У аналізі характеристик образу проводиться більш детальна обробка зображення. Визначаються геометричні характеристики зображення, межі об'єктів, аналізується колір, і таким чином уникається необхідність перебору усіх можливих комбінацій та прискорюється робота. Найбільш продвинутим методом оптичного розпізнавання є використання нейронних мереж. Він може застосовуватися також у комбінації з попередніми методами. Цей метод є більш ефективним та гнучким, але вимагає великої кількості зразків або

спеціальної структури моделі, що враховує специфіку даної задачі [8]. Серед завдань оптичного розпізнавання найбільш типовими є:-віднесення зразка до одного з відомих класів (класифікація, навчання із учителем);-автоматична класифікація –розбиття зразків на підмножини за спільними ознаками (кластеризація, навчання без учителя);-вибір інформативного набору ознак для подальшого розпізнавання;-динамічне розпізнавання об'єктів – співвіднесення оптичних зображень у часі, наприклад, для виявлення об'єктів, що рухаються;-завдання прогнозування.

Сьогодні методи глибинного навчання показують значні успіхи у вирішенні проблем, які неодноразово намагалися розв’язати протягом останніх десятиліть за допомогою штучного інтелекту: установлення рекордів у розпізнаванні зображень, виявленні транспортних засобів, розпізнаванні людської мови, реконструкції структури мозку, перекладі з однієї мови на іншу. Важливим для застосування методів глибинного навчання є наявність великих навчальних вибірок, оскільки недостатній обсяг навчальних даних завдає проблему «перенавчання» (overfitting), коли модель не узагальнює отриману інформацію, а просто її запам’ятовує. В цьому разі на навчальних даних модель показує хороші результати, але не показує такої точності на невідомих даних.

1.4 Аналіз існуючих аналогів

Існує велика кількість систем оптичного розпізнавання тексту, але в основному вони працюють з друкованим текстом. Точна обробка рукописного тексту є актуальним питанням.

Найпопулярнішим додатком для оптичного розпізнавання символів є FineReader від компанії ABBYY (рисунок 1.1). Програма швидко і з високою точністю розпізнає документи, перетворюючи їх в електронні редаговані формати. FineReader надає можливість розпочати роботу ще під час

сканування. Він надає доступ до документа у реальному часі, до того як усі сторінки будуть оброблені. Також програма здатна розпізнати текст на 190 мовах в будь-яких комбінаціях. ABBY FineReader може впоратися як із різними спотвореннями, характерними для цифрових фотографій (трапецієподібні спотворення, викривленння рядка, цифровий шум, і так далі), так і з дефектами зображення, пов'язаними зі станом вихідних паперових документів (пожовклий від часу папір, рукописні позначки, штампи) [9].

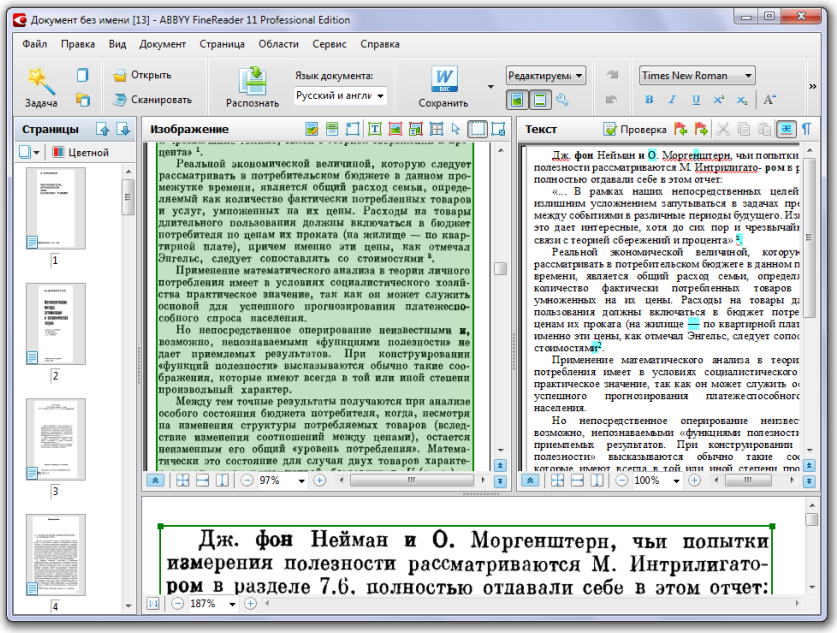


Рисунок 1.1 - Зразок інтерфейсу програми ABBYY FineReader

Багатьом відома світова компанія Google розробила додаток Google Translate, який допомагає людині перекласти слова та речення з однієї мови на іншу. До даного додатку декілька років тому була додана функція розпізнавання символів і її одноразовий переклад з однієї мови на іншу. Останні дослідження та публікації можна побачити на офіційному блогу компанії – AI Google Blog, де останні статті присвячені розпізнаванню музики або того, що зображено на фотографіях.

Ще одна відома компанія Adobe розробила комп’ютерний додаток Adobe Acrobat DC в якому присутня функція розпізнавання писемних

символів на папері в електронний вигляд. До цієї функції компанія Adobe намагається приділяти немалу увагу через її перспективність і у вказаному додатку вона доступна тільки після купівлі. Усю інформацію про розвиток даної технології в компанії можна дізнатися через її блог і зробити свої висновки щодо розвитку технології розпізнавання тексту на цей час.

1.5 Висновки

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

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

2 МЕТОД РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ

2.1 Нейронна мережа

Використання нейронної мережі на сьогоднішній день є дуже популярним серед великих корпорацій і таку мережу можна розробити не тільки використовуючи людські ресурси, які вичислюються сотнями або тисячами людей.

Нейронна мережа – це система обчислень, яка за своєю аналогією схожа на біологічну нейронну мережу [10]. Потрібно зазначити що нейронна мережа сама по собі вважається алгоритмом, а скоріше являє собою суміш різних алгоритмів машинного навчання, які працюючи разом оброблюють вхідні дані та на вихід виводять результат своєї роботи. Структура нейронної мережі відображає з себе сукупність вузлів, які з'єднані один з одним–дану структуру називають нейроном. Кожен вузол застосовується для передавання сигналу від одного нейрону до іншого. Кожний нейрон та вузол мають вагу, яка налаштовується у процесі навчання. Вага використовується для визначення того, які нейрони та вузли використовувати у процесах обробки інформації. Нейрони входять у склад шарів. Кожен шар виконує різний вид обробки інформації (рисунок 2.1).

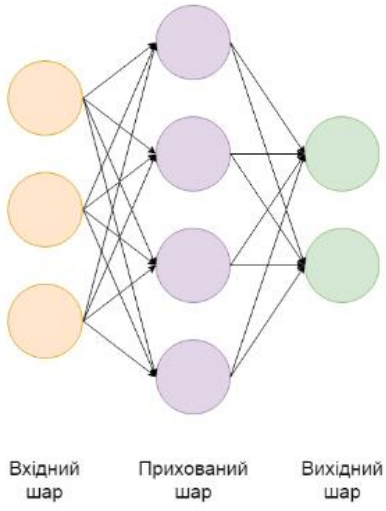


Рисунок 2.1 – Структура нейронної мережі

Головна формула нейронної мережі: $S = \sum_{i=1}^{n} X_i * W_i$, де X_i – вхідний сигнал, а W_i – значення ваги.

Найпопулярнішою, досліджуваною і застосовуваною нейронною мережею є багатошаровий перцептрон – Multi-Level Perceptron (MLP). Така структура, що навчається за допомогою зворотнього розширення помилки, є однією з універсальних форм нейронних мереж-класифікаторів і однією з найбільш часто використовуваних для розпізнавання тексту.

У цьому підході існує безліч різних методів. Найпопулярнішими можна назвати нечіткі нейронні мережі [11], мережу Хемінга [12], мережу Хопфилда [13], самоорганізаційну карту Кохонена та багато інших.

Основною перевагою застосування нейронних мереж та глибинного навчання взагалі є алгоритми, що навчаються. Вони поділяються на дві групи: алгоритми, що навчаються «з учителем» та «без учителя». Відмінність в першому випадку полягає у наявності зазделегідь відомих правильних результатів, на яких система може навчитися узагальненню. У випадку навчання без учителя система націлена на виокремлення груп екземплярів на основі спільних признаків.

2.2 Популярні типи нейронних мереж

2.2.1 Нейронна мережа прямого поширення

Ця нейронна мережа є однією з найпростіших форм нейронних мереж, де інформація рухається в одному напрямку. Інформація входить через вхідні вузли та виходить черезвихідні вузли. Вказаний тип нейронної мережіможе мати прихований шар. Нижче наведено приклад схеми мережі прямого поширення (рисунок 2.2). Як можна бачити система обчислює суму продуктів входів і ваг, які ідуть на вихід. Вихід вважається, якщо він перевищує певне значення, тобто порогову величину (зазвичай 0), і

нейрон випускається з активованим виходом (зазвичай 1), а якщо він не спрацьовує, то вимикається деактивоване значення (зазвичай -1).

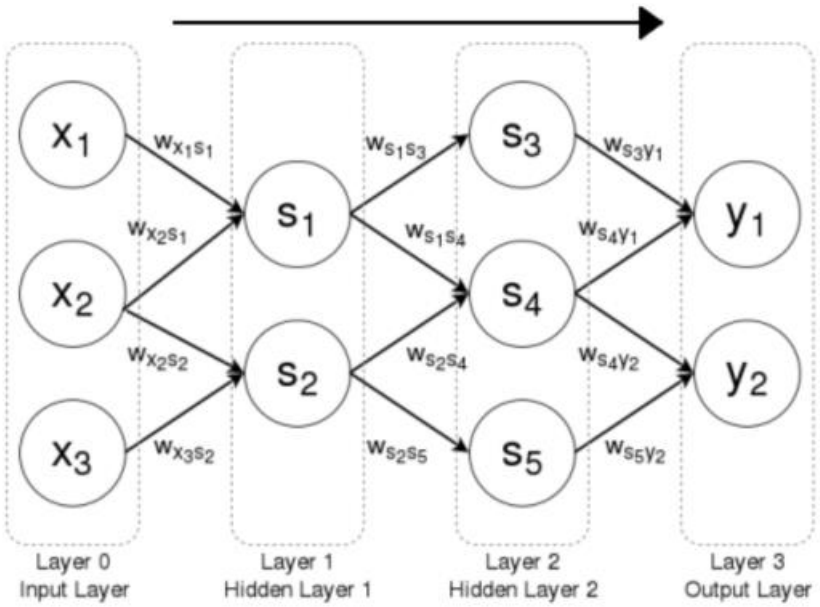


Рисунок 2.2 – Схема нейронної мережі прямого поширення

Мережа прямого поширення може знайти своє призначення в розвитку технології комп'ютерного бачення та розпізнаванні мовлення. Вказана нейронна мережа може з вхідними даними в яких присутній «шум» та легко підтримуються через свою просту систему роботи [14].

2.2.2 Самоорганізаційна карта Кохонена

Метою карти Кохонена є введення векторів довільної розмірності до дискретної карти, що складається з нейронів. Для правильної роботи самоорганізаційної карти Кохонена потрібно провести її навчання, щоб вона змогла створити свою власну структуру роботи. Цей тип нейронної мережі складається з одного або двох вимірів. Під час тренування карта розташування нейрона залишається постійною, але ваги відрізняються залежно від значення [15] (рисунок 2.3).

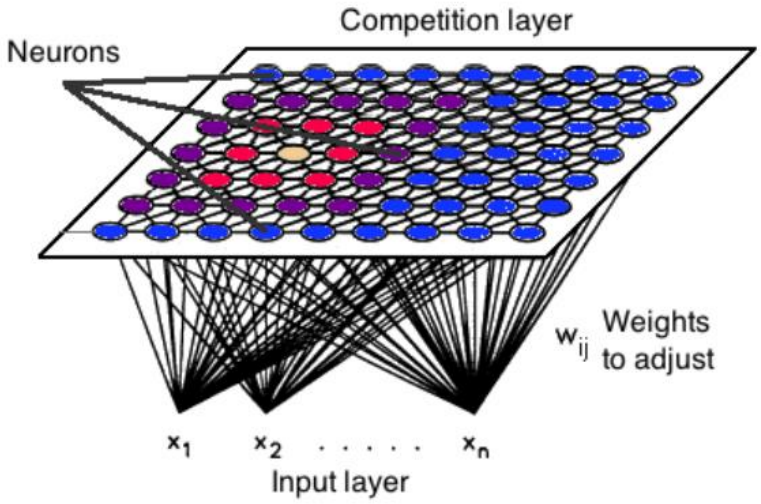


Рисунок 2.3 – Структура карти Кохонена

Процес самоорганізації має декілька етапів:

1. На першій фазі кожне значення нейрона ініціалізується з невеликою вагою та вхідним вектором.

2. На другому етапі нейрон, найближчий до точки, є "виграшним нейроном", а нейрони, з'єднані з виграшним нейроном, також рухаються до точки, подібної до наведеної нижче графіки.

Відстань між точкою і нейронами розраховується за евклідової відстані, нейрон з найменшою відстані перемагає. Через ітерації всі точки кластеризовані, і кожен нейрон представляє кожен вид кластеру.

Самоорганізаційна карта Кохонена використовується для розпізнавання шаблонів даних. Його застосування можна знайти в медичному аналізі для групування даних у різних категоріях.

2.2.3 Рекурентна нейронна мережа

Рекурентна нейронна мережа працює за принципом збереження виходу шару та подачі назад на вхід, щоб допомогти прогнозувати результат шару. Тут перший шар утворюється схожою на вихідну нейронну мережу з

продуктом суми ваг і функцій. Сам процес рекурентної нейронної мережі починає працювати після обчислення першого шару. Це означає, що від одного етапу до наступного кожного нейрона буде пам'ятати деяку інформацію, яку вона мала в попередньому кроці [16].

Така поведінка показує, що кожен нейрон виступає як комірка пам'яті у виконанні обчислень.У цьому процесі ми повинні дозволити нейронній мережі працювати над розповсюдженням спереду і пам'ятати, яка інформація потрібна для подальшого використання. Якщо під час процесу прогноз неправильний, ми використовуємо швидкість навчання або корекцію помилок, щоб внести невеликі зміни, щоб поступово працювати над правильним прогнозуванням під час зворотного розповсюдження. Застосування рекурентної нейронної мереж іможна знайти в моделях перетворення текстового мовлення (TTS).

2.2.4 Згорткова нейронна мережа

З появою великих обсягів інформації і великих обчислювальних можливостей почали активно використовуватися нейронні мережі. Дуже популярними стали згорткові нейронні мережі, архітектура яких була запропонована Яном Лекуном та націлена на ефективне розпізнавання зображень. Назву архітектура мережі отримала завдяки наявності операції згортки, суть якої в тому, що кожний фрагмент зображення множиться на ядро згортки поелементно, а результат сумується і записується в аналогічну позицію вихідного зображення. До архітектури мережі закладено апріорні знання з області комп’ютерного зору: піксель зображення сильніше пов’язаний з сусіднім і об’єкт на зображенні має змогу зустрітися в будь-якій частині зображення.

Згорткова нейронна мережа являє собою чергування згортальних шарів, агрегувальних шарів і при наявності повнозв’язних шарів на виході. Всі три види шарів можуть чергуватися в довільному порядку [17].

У згорткових шарах нейрони, які використовують одні і ті ж ваги, об’єднуються в карти ознак, а кожен нейрон карти ознак пов’язаний з частками нейронів попереднього шару. При обчисленні мережі виходить, що кожний нейрон виконує згортку певної області попереднього шару (яка визначається великою кількістю нейронів, пов’язаних з даними нейроном).

Шар в якому кожний нейрон з’єднаний з усіма нейронами на попередньому рівні, причому кожна зв’язок має свій ваговий коефіцієнт.

На відміну від повнозв’язного, в згортковому шарі нейрон з’єднаний тільки з обмеженою кількістю нейронів попереднього рівня, згортковий шар аналогічний застосуванню операції згортки, де використовується лише матриця ваг невеликого розміру, яку «рухають» по всьому оброблюваного шару. Ще одна особливість згорткового шару в тому, що він трохи зменшує зображення за рахунок крайових ефектів.

2.2.5 Модулярна нейронна мережа

Модулярна нейронна мережа містить сукупність різних мереж, що працюють незалежно один від одного та сприяють коректному фінальному результату роботи мережи. Кожна нейронна мережа має безліч входів, які є унікальними в порівнянні з іншими мережами, які будують та виконують підзавдання. Ці мережі не взаємодіють або не сигналізують один одного при виконанні завдань. Перевага модулярної нейронної мережі полягає в тому, що вона розбиває великий обчислювальний процес на менші компоненти, що зменшує складність обчислення. Вказана розбивка допомагає зменшити кількість з'єднань і конфліктів взаємодії цих мереж один з одним, що, в свою чергу, збільшує швидкість обчислень. Проте час обробки буде залежати від кількості нейронів та їх участі в обчисленні результатів[18].

2.2.6 Вибір типу нейронної мережі для вирішення задачі розпізнання рукописного тексту

Для роботи системи, яка буде виконувати розпізнавання писемних символів потрібно створити модель, архітектура мережі якої спеціально розроблена для розпізнавання об’єктів, а в нашому випадку – текст, на зображенні. Запропонована модель нейронної мережі називається згортковою рекурентною нейронною мережею (CRNN), оскільки вона є комбінацією згорткової нейронної мережі та рекурентної нейронної мережі.

Для об'єктів, подібних до послідовності, CRNN має ряд відмінних переваг перед звичайними моделями нейронних мереж:

- вона може навчитись безпосередньо з міток послідовностей (наприклад, слів), не вимагаючи детальних анотацій (наприклад, символів);
- вона має однакову властивість з RNN, будучи здатною виробляти послідовність міток;
- вона не обмежена довжиною подібних до послідовності об’єктів, вимагає лише нормалізації висоти як на етапі навчання, так і на етапі тестування;
- вона містить набагато менше параметрів, ніж стандартна модель згорткової, споживаючи менше місця для зберігання.

2.3 Архітектура нейронної мережі

Архітектура нейронної мережі складається з трьох компонентів, як показано на рисунку 2.4, включаючи згортковий шари, періодичні шари та шар транскрипції. У нижній частині нейронної мережі згорткові шари автоматично витягують послідовність об’єктів з кожного вхідного зображення. На вершині згорткової мережі – рекурентна мережа, побудована

для прогнозування для кожного кадру об’єкта послідовності, виведена згортковими шарами.

Транскрипційний шар у верхній частині CRNN прийнятий для перекладу прогнозування на кадр повторюваними шарами в послідовність міток. Хоча CRNN складається з різних видів мережевих архітектур (наприклад, CNN і RNN), вона може бути спільно навчена з однією функцією втрат.

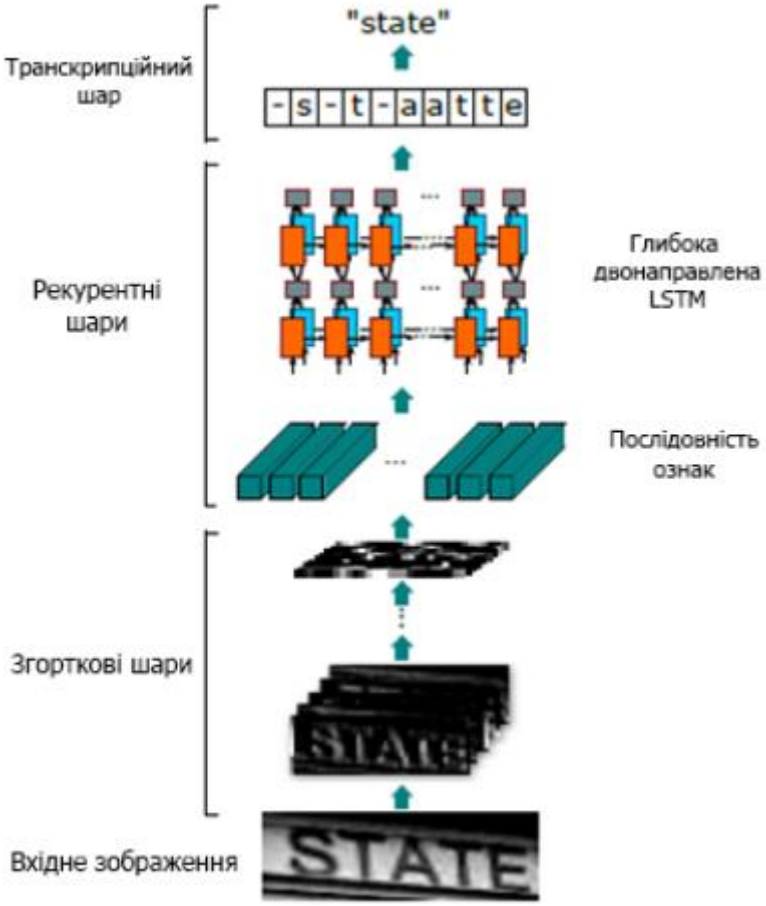


Рисунок 2.4 – Архітектура нейронної мережі

2.3.1 Особливості витягу послідовності

У моделі CRNN компонент згорткових шарів будується, беручи згорткові та максимально об'єднані шари зі стандартної моделі CNN (повністю з'єднані шари видаляються). Такий компонент використовується

для вилучення послідовності представлення ознак із вхідного зображення. Перед подачею в мережу всі зображення потрібно масштабувати до однакової висоти. Потім послідовність векторів ознак витягується з карт об’єктів, що створюються компонентом згорткових шарів, який є вхідним сигналом для повторюваних шарів. Зокрема, кожен вектор ознак послідовності ознак генерується зліва направо на картах об’єктів за стовпчиком. Це означає, що і-м вектором об'єднання є об'єднання і-го стовпців усіх карт. Ширина кожного стовпця в наших налаштуваннях фіксована до одного пікселя.

Оскільки згорткові шари, максимізаційне агрегування та елементно-активаційні функції у локальних регіонах, вони є інваріантними щодо трансляції. Отже, кожен стовпець карт об’єктів відповідає області прямокутника оригінального зображення (що називається сприйнятливим полем), і такі області прямокутника розташовані в однаковому порядку зі своїми відповідними стовпцями на картах об’єктів зліва направо. Кожен вектор у послідовності ознак пов'язаний з рецептивним полем і може розглядатися як дескриптор зображення для цієї області (рисунок 2.5).

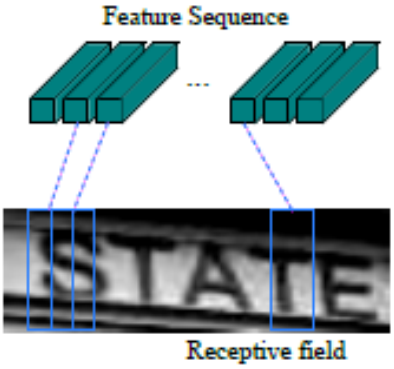


Рисунок 2.5 – Рецептивне поле

Будучи надійними, багатими та здатними до дресирування, згорткові особливості широко застосовуються для різних видів завдань візуального розпізнавання [19]. Оскільки CNN вимагає масштабування вхідних

зображень до фіксованого розміру, щоб задовольнити його фіксований вхідний розмір, він не підходить для об'єктів, подібних до послідовності, через їх велику варіацію довжини. У CRNN ми передаємо глибокі риси в послідовні подання, щоб бути інваріантними до варіації довжини подібних до послідовності об'єктів.

2.3.2 Маркування послідовності

Глибока двонаправлена рекурентна нейронна мережа побудована на вершині згорткових шарів, як рекурентні шари. Повторні шари передбачають розподіл міток y_t для кожного кадру x_t у послідовності ознак $x = x_1,..., x_T$. Переваги повторюваних шарів потрійні.

По-перше, RNN має потужну здатність фіксувати контекстну інформацію в межах послідовності. Використання контекстних сигналів для розпізнавання послідовностей на основі зображень є більш стабільним та корисним, ніж обробка кожного символу незалежно. Взявши за приклад розпізнавання тексту, широким символам може знадобитися кілька послідовних кадрів для повного опису. Крім того, деяких двозначних персонажів легше розрізнити під час спостереження за їхнім контекстом, наприклад легше розпізнати “il”, протиставляючи висоту символів, ніж розпізнаючи кожну з них окремо.

По-друге, RNN може зворотно поширювати диференціали помилок на свій вхід, тобто згортковий рівень, що дозволяє спільно навчати повторювані шари та згорткові шари в єдиній мережі.

По-третє, RNN здатна оперувати послідовностями довільної довжини, переходячи від початку до кінця.

Традиційна одиниця RNN має приєднаний прихований шар, що самостійно підключається, між вхідним та вихідним шарами. Кожного разу, коли він отримує кадр x_t у послідовності, він оновлює свій внутрішній стан h_t нелінійною функцією, яка приймає як вхід поточного входу x_t, так і минулого

стану h_{t-1}: $h_t = g(x_t, h_{t-1})$. Тоді прогноз y_t робиться на основі h_t. Таким чином, минулі контексти $\{x_{t\prime}\}_{t\prime<t}$ фіксуються та використовуються для прогнозування. Традиційна одиниця RNN, однак, страждає від проблеми зникнення градієнта, що обмежує діапазон контексту, який він може зберігати, і додає навантаження на тренувальний процес. Довго-короткочасна пам’ять (LSTM) - це тип RNN-пристрою, спеціально розроблений для вирішення цієї проблеми. LSTM (рисунок 2.6) складається з комірки пам'яті та трьох мультиплікативних шлюзів, а саме вхідних, вихідних та забутих.

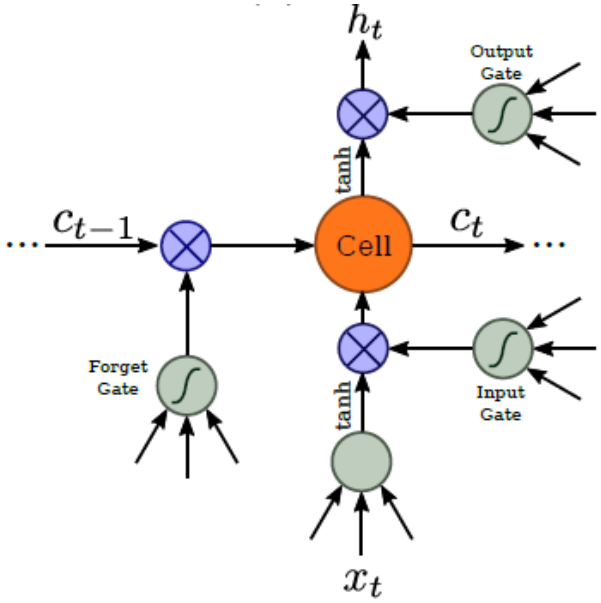


Рисунок 2.6 – Структура базової LSTM одиниці

Концептуально комірка пам'яті зберігає минулі контексти, а вхідні та вихідні ворота дозволяють комірці зберігати контексти протягом тривалого періоду часу. Тим часом пам'ять у комірці може очиститися воротами забуття. Спеціальний дизайн LSTM дозволяє йому фіксувати дальні залежності, які часто трапляються в послідовностях на основі зображень.

LSTM спрямована, вона використовує лише минулий контекст. Однак у послідовностях на основі зображень контексти з обох напрямків корисні та доповнюють один одного. Отже, поєднуємо два LSTM, один вперед та один назад, у двонаправлений LSTM. Крім того, можна скласти багатонаправлені двосторонні LSTM, що призводить до глибокого двонаправленого LSTM.

Глибока структура дозволяє вищий рівень абстракцій, ніж неглибокий, і досягла значного покращення продуктивності завдання розпізнавання мови [20].

2.3.3 Транскрипція

Транскрипція – це процес перетворення передбачень на кадр, зроблених RNN, у послідовність міток. Математично транскрипція полягає в тому, щоб знайти послідовність міток з найбільшою ймовірністю, зумовлену передбаченнями для кожного кадру. На практиці існує два способи транскрипції, а саме на ті, що не містять лексики, та транскрипції на основі лексикону. Лексикон – це набір послідовностей міток, для яких передбачення є обмеженням, наприклад словник перевірки орфографії. У режимі без лексики передбачення робляться без будь-якого лексикону. У режимі, заснованому на лексиконі, прогнозування здійснюється шляхом вибору послідовності міток, що має найбільшу ймовірність.

Ми приймаємо умовну ймовірність, визначену в шарі класифікації по рейтингу (Connectionist Temporal Classification (CTC)) [21]. Ймовірність визначається для послідовності міток l, обумовленої прогнозами на кадрі y = y_1,..., y_T, і він ігнорує позицію, де знаходиться кожна мітка в l. Отже, коли ми використовуємо негативну логарифмічну вірогідність цієї ймовірності як мету для навчання мережі, нам потрібні лише зображення та відповідні їм послідовності міток, уникаючи праці маркування позицій окремих символів.

Формулювання умовної ймовірності коротко описується таким чином: Вхідні дані - це послідовність y = y_1,..., y_T, де T - довжина послідовності. Тут кожен $y_t \in R^{|\mathcal{L}'|}$ – це розподіл ймовірностей за набором $\mathcal{L}' = \mathcal{L} \cup$, де $\mathcal{L}$ містить усі мітки в завданні (наприклад, усі англійські символи), а також «порожню» мітку. Функція відображення послідовності в послідовність $\mathcal{B}$ визначена на послідовності $\pi \in \mathcal{L}'^{\mathrm{T}}$, де T - довжина. $\mathcal{B}$ відображає π на l, спочатку видаляючи повторювані мітки, потім видаляючи «порожнечі».

Наприклад, $\mathcal{B}$ відображає «--hh-e-l-ll-oo--» ('-' означає «порожнє») на «hello». Тоді умовна ймовірність визначається як сума ймовірностей усіх π, які відображаються $\mathcal{B}$ на l:

$$p(l|y) = \sum_{\pi:\mathcal{B}(\pi)=l} p(\pi|y) \tag{2.1}$$

де ймовірність π визначається як $p(\pi|y) = \prod_{t=1}^{T} y_{\pi_t}^t$, $y_{\pi_t}^t$ – це ймовірність наявності мітки t на позначці часу t. Безпосередньо обчислювати рівняння 2.1 буде обчислювально нездійсненним через експоненціально велику кількість елементів підсумовування. Однак його можна ефективно обчислити, використовуючи алгоритм вперед-назад.

2.4 Навчання нейронної мережі

Мережа навчається за допомогою алгоритму RMSProp, який використовує адаптивну швидкість навчання [22]. RMSProp ділить швидкість навчання η на експоненціально спадаючу суму квадрат градієнтів в середньому, оновлення параметрів показано у формулі 2.2. В оригінальному формулюванні пропоноване значення коефіцієнта навчання становить 0.001 та 0.9 для коефіцієнту розпаду γ, однак Муккамала та Хайн [23] дають більш детальне пояснення щодо вибіру параметрів. Перевага перед іншими навчальними алгоритмами, такими як градієнтний спуск в тому, що швидкість навчання адаптується автоматично, потрібно встановити лише початкове значення.

$$\begin{aligned} w' &= w - \frac{\eta}{\sqrt{avg+\epsilon}} \cdot \nabla_w E\,(w) \\ avg' &= 0.9 \cdot avg + 0.1 \cdot (\nabla_w E\,(w))^2 \end{aligned} \tag{2.2}$$

Навчання глибокої штучної нейронної мережі може бути проблематичним оскільки оновлення параметра виконується з припущенням, що всі інші рівні залишилися без змін. Звичайно, це неправда, і ефект збільшується, чим більше шарів додається. Нормалізація пакета вирішує цю проблему, нормалізуючи пакет активацій введення (для вхідного або будь-якого прихованого шару) шляхом віднімання середнього та ділення на стандартне відхилення.

Під час навчання помилка навчального набору постійно зменшується, тоді як помилка набору перевірки починає знову збільшуватися в певний момент. Це пов'язано з переобладнанням. Алгоритм ранньої зупинки, використовується для запобігання перенавчання (рисунок 2.7). Параметр терпіння *p* позначає кількість навчальних епох, які алгоритм продовжує після епохи з найкращим результатом, досягнутий на сьогодні на наборі перевірки. Якщо кращого результату неможливо знайти після *p* випробування навчання закінчується.

```
1  i = 0;
2  v* = ∞;
3  while i < p do
4  |   train the ANN;
5  |   update w;
6  |   v = ValidationSetError(w);
7  |   if v < v* then
8  |   |   v* = v;
9  |   |   w* = w;
10 |   |   i = 0;
11 |   end
12 |   else
13 |   |   i = i + 1;
14 |   end
15 end
16 return w*
```

Рисунок 2.7 – Алгоритм ранньої зупинки

2.5 Сегментація слів

У методах розпізнавання символів сегментація є найважливішим процесом. Сегментація здійснюється для поділу зображення на окремі символи. Сегментація рукописного слова в різні зони (верхня середня і нижня) і символи складніша, ніж у друкованих документів. Це пояснюється в основному мінливістю міжсимвольних відстаней, перекосом, нахилом, розмірами, тобто почерком. Іноді компоненти двох послідовних символів можуть бути зачеплені або перекриті, і ця ситуація значно ускладнює завдання сегментації. В індійських мовах такий дотик або перекриття часто відбувається через змінні символи верхньої та нижньої зони. Сегментація є важливим етапом, тому що степінь, якого можна досягти при розділенні слів, рядків або символів, безпосередньо впливає на швидкість розпізнавання тексту [24].

Існує два типи сегментації:

- Зовнішня сегментація – розкладає макет сторінки на її логічні одиниці. Зовнішня сегментація – це ізоляція різних блоків запису, таких як абзаци, речення або слова. Це найважливіша частина аналізу документів. Аналіз і розпізнавання документів спрямовано на автоматичне вилучення інформації, представленої на папері, і від самого початку адресоване людському розумінню. Сегментація сторінок є одним з важливих кроків у аналізі макетів і особливо важка при роботі зі складними макетами. Аналіз макета сторінки здійснюється у два етапи: Перший етап - це структурний аналіз, який стосується сегментації зображення на блоки компонентів документа (абзац, рядок, слово тощо). Другий - функціональний аналіз, який використовує розташування, розмір і різні правила розмітки для позначення функціонального змісту компонентів документа. Потім сегментація сторінок реалізується шляхом пошуку текстурованих областей у чорно-білих або кольорових зображеннях.
- Внутрішня сегментація - це операція, яка намагається розкласти зображення послідовності символів на суб-зображення окремих символів. Хоча за останні десять років ці методи надзвичайно розвилися, і з'явилися

різноманітні методики, сегментація рукописного шрифту на символи залишається невирішеною проблемою.

Для запропонованого підходу необхідна система розпізнавання рукописних слів повинна бути заснована на штучній нейронній мережі, навченій з втратою CTC. Алгоритм декодування найкращого шляху CTC модифікований для виведення найкращого шляху без його згортання. Межі слів кодуються (помножують) пробілами в найкращому шляху. Для сегментації цього шляху можна використовувати чотири обмежувальні поля (рисунок 2.8). Перше обмежувальне поле (синє) просто розбиває шлях на пробіли та розглядає решту підпутей як слова. Другий (зелений) і третій (чорний) обмежувальні рамки видаляють пробіли ліворуч або праворуч від підпроходів. Нарешті, четвертий (червоний) видаляє заготовки з обох сторін.

ANN масштабує вхідне зображення на фіксований коефіцієнт, тому можна обчислити позицію слова на вхідному зображенні, помноживши позицію на найкращому шляху на згаданий коефіцієнт масштабування. Залишається питання, чи розпізнані символи на найкращому шляху знаходяться поблизу свого місця на вхідному зображенні. Наприклад, RNN може вирівняти всіх символів ліворуч, незалежно від їх реального положення, оскільки втрата CTC працює без сегментації.

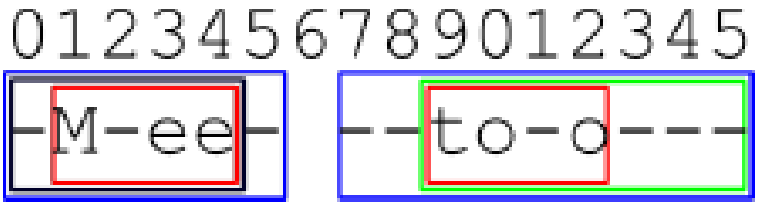
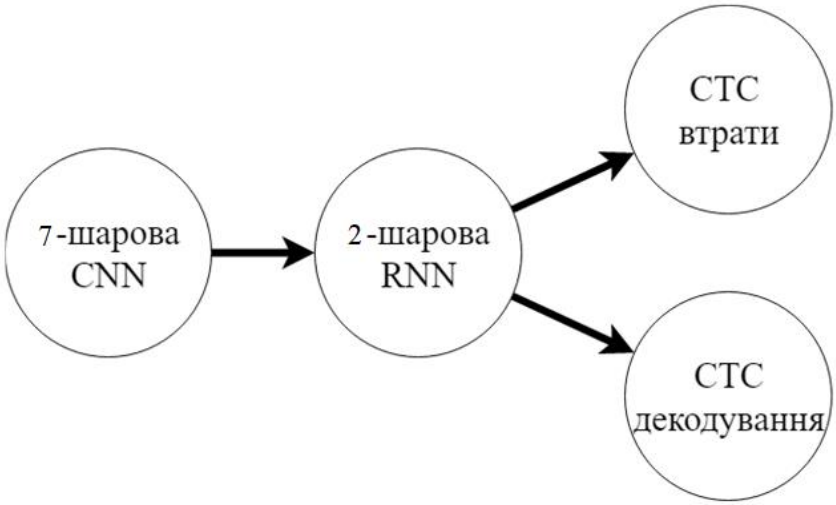


Рисунок 2.8 – Сегментація тексту чотирьма обмежувальними полями

2.5 Модель нейронної мережі

Модель нейронної мережі для реалізації системи розпізнавання рукописного тексту в даному випадку складається з шару згорткової

нейронної мережі (CNN), шару рекурентної нейронної мережі (RNN) та фінального шару часового класифікатора з'єднань (CTC). Схема моделі нейронної мережі для розпізнавання рукописного тексту наведена на рисунку 2.9.



2.9 – Схема моделі нейронної мережі

Нейронну мережу формально можна розглядати як функцію 2.3), яка відображає зображення (або матрицю) M розміру W*N в послідовність символів $(C_1, C_2, C_3, \dots C_n)$ з довжиною від 0 до L

$$M \rightarrow (C_1, C_2, C_3, \dots C_n), \text{ де } 0 \leq n \leq L \qquad (2.3)$$

Текст розпізнається на рівні символів, тому також можна розпізнати слова або тексти, які не містяться у навчальних даних (якщо окремі символи правильно класифікуються).

CNN: вхідне зображення подається в шари CNN. Ці шари навчені отримувати відповідні признаки з зображення. Кожен шар складається з трьох операцій. Спочатку виконується операція згортки, яка застосовує до входу ядро фільтру розміром 5 × 5 в перших двох шарах та 3 × 3 в останніх п'яти шарах. Потім застосовується нелінійна функція RELU (2.4).

$$A(x) = \max(0,\ x) \tag{2.4}$$

Нарешті, шар об'єднання сумує області зображень і виводить зменшену версію вводу. У той час як висота зображення зменшується на 2 в кожному шарі, додаються карти (канали) об'єктівтак що карта виходу (або послідовність) має розмір 100 × 512.

RNN: послідовність признаків містить 512 признаків за такт, RNN поширює відповідну інформацію через цю послідовність. Застосовується популярна реалізація RNN з довгостроковою пам’яттю, оскільки вона здатна поширювати інформацію на великі відстані і забезпечує більш надійну тренувальну характеристику, ніж звичайна RNN. Вихідна послідовність RNN відображається в матрицю розміром 32 × 80. База даних IAM складається з 79 різних символів, ще один додатковий символ необхідний для операції CTC (порожня мітка CTC), тому існує 80 записів для кожного зі ста тимчасових кроків.

CTC: під час навчання нейронної мережі, CTC отримує вихідну матрицю RNN і текстову інформацію про правдивий текст на зображенні, і обчислює значення втрат. Підчас розпізнавання, CTC отримує тільки матрицю і розшифровує її в кінцевий текст. Довжина правдивого тексту і розпізнаного тексту може бути не більше 100 символів.

На вхід даної нейронної мережі подаються чорно-білі зображення розміром 800 × 64. Зазвичай, зображення з набору даних який використовується для навчання нейронної мережа не мають точно такого розміру, тому ми змінюємо його розмір (без спотворень), щоб змінене зображення мало ширину 800 або висоту 64. Потім ми копіюємо зображення в пусте зображення розміром 800 × 64. Після, цього ми нормалізуємо значення кольору зображення, що спрощує завдання для нейронної мережі. Збільшення даних можна легко інтегрувати, скопіювавши зображення у

випадкові позиції, замість вирівнювання його ліворуч, або шляхом випадкової зміни розміру зображення.

Результат RNN: на рисунку 2.10 показана візуалізація вихідної матриці RNN для зображення, що містить текст «the fake friend of the family, like the». Матриця, показана на верхньому графіку, містить бали для символів, включаючи порожню мітку CTC як її останній запис. Інші записи матриці, зверху вниз, відповідають наступним символам:

“!”#&’()*+,./0123456789:;?ABCDEFGHIJKLMNOPQRSTUVWXYZabcd efghijklmnopqrstuvwxyz”

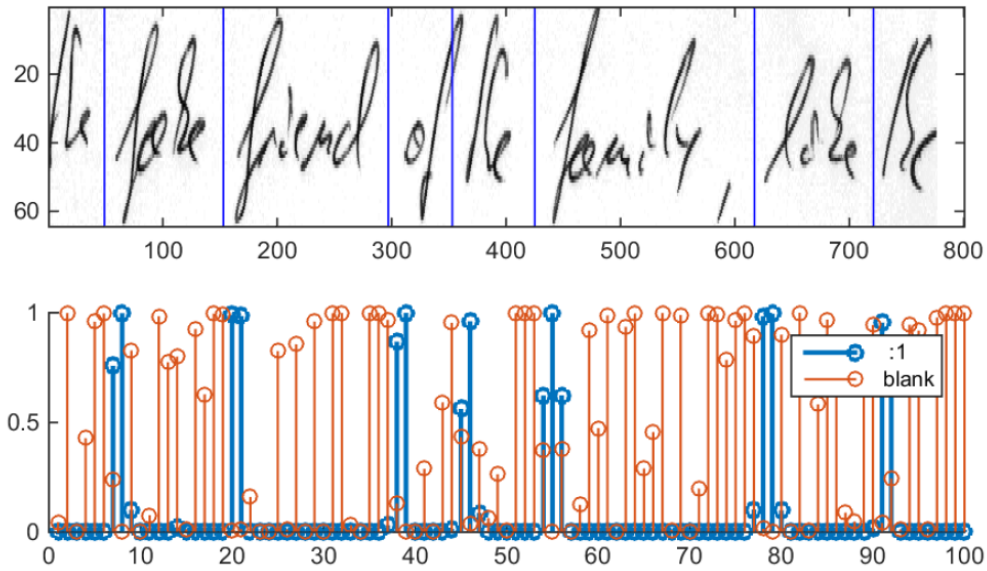


Рисунок 2.10 – Результат розпізнавання після шару RNN

2.5 Висновки

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

Було проведено опис роботи моделі нейронної мережі для розпізнавання рукописного тексту. Розроблено структуру нейронної мережі,

яка проектувалась із розрахунком на можливе покращення для збільшення точності розпізнавання рукописного тексту.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ

3.1 Обґрунтування вибору мови програмування

Мова програмування - це набір лексичний абстракцій, що забезпечує опис конкретних проблем, які сформовані людиною й потребують вирішення за допомогою комп'ютера. Мовою програмування пишеться програма, що дозволяє при її виконанні комп'ютером одержати конкретні результати. Мови програмування мають дві складові: синтаксис та семантика.

Здійснимо вибір мови програмування для розробки системи розпізнавання рукописного тексту.

Порівняємо 3 відомі мови програмування, які підійшли б для виконання даної роботи, а саму Python, Java, C++ та оберемо кращу.

Python проста у використанні, та одночасно повноцінна мова програмування, яка надає набагато більше засобів до структурування і підтримки великих програм. Вона краще за C обробляє помилки, і, будучи мовою високого рівня, має вбудовані типи даних високого рівня, такі як гнучкі масиви і словники, ефективна реалізація яких на C потребує великих втрат часу [25].

Завдяки загальним типам даних, Python використовують до більш широкого кола задач, ніж Awk і навіть Perl, водночас багато речей на ній робляться настільки ж просто.

Python дозволяє розбити програми на модулі, які потім можуть бути використані в інших програмах. Python постачається з потужною бібліотекою стандартних модулів, які можна використовувати як основу для нових додатків або як приклади при вивченні мови. Стандартні модулі забезпечують інструменти для роботи з файлами, системними викликами, мережними з'єднаннями і навіть інтерфейсами до графічних бібліотек.

Python – це інтерпретована мова, яка економить значну кількість часу, який зазвичай витрачається на компіляцію. Інтерпретатор може використовуватися інтерактивно, що дозволяє експериментувати з можливостями мови, писати шаблони програм або тестувати функції при розробці “знизу-вверх”. Він також зручний як настільний калькулятор. Python дозволяє писати компактні й читабельні програми. Програми, написані мовою Python, звичайно значно коротші еквівалента на C або C++ з декількох причин:

- типи даних високого рівня дозволять Вам виразити складні операції однією інструкцією;
- групування інструкцій виконується за допомогою відступів замість фігурних дужок;
- немає необхідності в оголошенні змінних;

Python це розширювана мова: знання C дозволяє додавати нові функції, що вбудовуються, або модулі для виконання критичних операцій з максимальною швидкістю або писати інтерфейс до комерційних бібліотек, доступним тільки у двійковій формі. Інтерпретатор Python можна вбудувати в програму, написану на мові C, і використовувати як розширення або мову команд для цієї програми. В даний час Python використовується десятками тисяч програмістів по всьому світу, і кількість людей, які його використовують, швидко зростає, щороку збільшуючись удвічі та втричі. Python приваблює користувачів з ряду причин. Він використовується для розробки програм і дозволяє розвиватись набагато швидше, ніж традиційні мови, такі як C, C ++ або Java. Ця мова однаково добре працює в Windows, UNIX, Macintosh та OS/2 і може використовуватися для легкої розробки невеликих додатків або сценаріїв, а також для розгортання великих програм. Python пропонує доступ до потужного та простого у використанні набору з 29 інструментів графічного інтерфейсу користувача. Традиційні машинні мови, такі як C та Pascal, мають ряд характеристик, таких як суворе введення тексту, основні типи, складні (і, як правило, довгі) цикли, а також

необхідність у великій кількості коду для виконання порівняно невеликих завдань. Java є досить новою, але поділяє більшість функцій, включених до цього списку. Програмісти, знайомі з традиційними мовами, погодяться, що відсутність суворого набору тексту полегшує роботу з Python [26].

Зазвичай очікується, що Python програми виконуються повільніше ніж програми Java, але вони в той же час вимагають набагато менше часу для розробки. Python програми типово повільніше в 3-5 разів, ніж еквівалентні Java програми. Ця різниця може бути пояснена за рахунок вбудованих високорівневих типів даних Python, і його динамічної типізації. Наприклад, Python програміст не витрачає часу, описуючи типи аргументів або змінних, а потужні типи поліморфних списків і словників Python, для яких багата синтаксична підтримка вбудована прямо в саму мову, можуть знайти застосування майже в кожній Python програмі. Через тіпізірованія під час виконання, Python повинен виконувати більше роботи, ніж Java.

Наприклад, при обробці вираження a + b, він повинен спершу дослідити об'єкти a і b, щоб з'ясувати їх типи, які не відомі під час компіляції. Потім викликається відповідна операція додавання, яка може виявитися перевантаженим користувачем методом. Java, з іншого боку, може виконувати ефективне додавання цілих або чисел з плаваючою точкою, але вимагає опису змінних a і b, і не дозволяє перевантажувати оператор + для примірників класів, визначених користувачем.

З цих причин, Python набагато більш підходить як "склеює" мову, в той час як Java краще характеризується як низькорівневий мову для реалізації. Фактично, вони разом можуть утворити відмінну пару. Компоненти можна реалізовувати на Java, а потім використовувати в додатках на Python; Python також корисно використовувати для прототипів компонент, поки їх розробка не "затвердіє" в Java реалізації. Для підтримки такого типу розробки, створюється реалізація Python, написана на Java, вона дозволяє викликати Python код з Java і навпаки. У цій реалізації вихідний код Python

транслюється в байт-код Java (за допомогою бібліотеки часу виконання, для підтримки динамічної семантики Python).

Майже все сказане для Java, також можна застосувати до C ++, просто тим більше, що там де код Python зазвичай в 3-5 разів коротше, ніж еквівалентний код Java, він часто в 5-10 разів коротше еквівалентного коду C ++. Анекдотичний підтвердження говорить: те, що один програміст Python може завершити за два місяці, два програміста C ++ не зможуть зробити і за рік. Python блискуче використовується як клей, що з'єднує компоненти, написані на C ++ [26].

Через простоту розробки і наявність широкого спектру програмних інструментів обрано мову програмування Python. Ця мова підходить для розробки і тестування продукту, який є ціллю даної роботи.

3.2 Обґрунтування вибору середовища розробки

Інтегроване середовище розробки, ICP (англ. IDE, Integrated development environment) – система програмних засобів, використовувана програмістами для розробки програмного забезпечення.

Зазвичай, середа розробки включає в себе:

- текстовий редактор,
- компілятор або інтерпретатор,
- засоби автоматизації збирання,
- відладчик.

Іноді містить також засоби для інтеграції з системами управління версіями і різноманітні інструменти для спрощення конструювання графічного інтерфейсу користувача. Багато сучасних середовища розробки також включають браузер класів, інспектор об'єктів і діаграму ієрархії класів - для використання при об'єктно-орієнтованої розробки ПЗ. Хоча й існують ICP, призначені для декількох мов програмування – такі як Eclipse, Embarcadero RAD Studio, Qt Creator, останні версії NetBeans, Xcode або

Microsoft Visual Studio, але зазвичай ІСР призначається для одного певної мови програмування – як, наприклад, Visual Basic, Delphi, Dev-C++.

Інтегровані середовища розробки були створені для того, щоб максимізувати продуктивність програміста завдяки тісно пов'язаним компонентам з простими користувача інтерфейсами. Це дозволяє розробнику зробити менше дій для перемикання різних режимів, на відміну від дискретних програм розробки.

ІСР, зазвичай, являє собою єдину програму, в якій проводилася вся розробка. Вона, зазвичай, містить багато функцій для створення, зміни, компілювання, розгортання і налагодження програмного забезпечення. Мета середовища розробки полягає в тому, щоб абстрагувати конфігурацію, необхідну, щоб об'єднати утиліти командного рядка в одному модулі, який дозволить зменшити час, щоб вивчити мову, і підвищити продуктивність розробника. Також вважається, що важка інтеграція завдань розробки може далі підвищити продуктивність. Наприклад, ІСР дозволяє проаналізувати код і тим самим забезпечити миттєвий зворотний зв'язок і повідомити про синтаксичні помилки.

Сьогодні існує досить велика кількість ІСР для Python: Boa Constructor, Eclipse, PyDev , Eric, IDLE, Komodo, PyCharm, PyScripter, SPE, Wing IDE. Проте їхні можливості поки що не можуть задовольнити усіх потреб розробників програмного забезпечення. Винятками є Eclipse та PyCharm.

Середовищем для розробки ігрового додатку було обрано PyCharm, інтерфейс головного вікна якого зображено на рисунку 3.1.

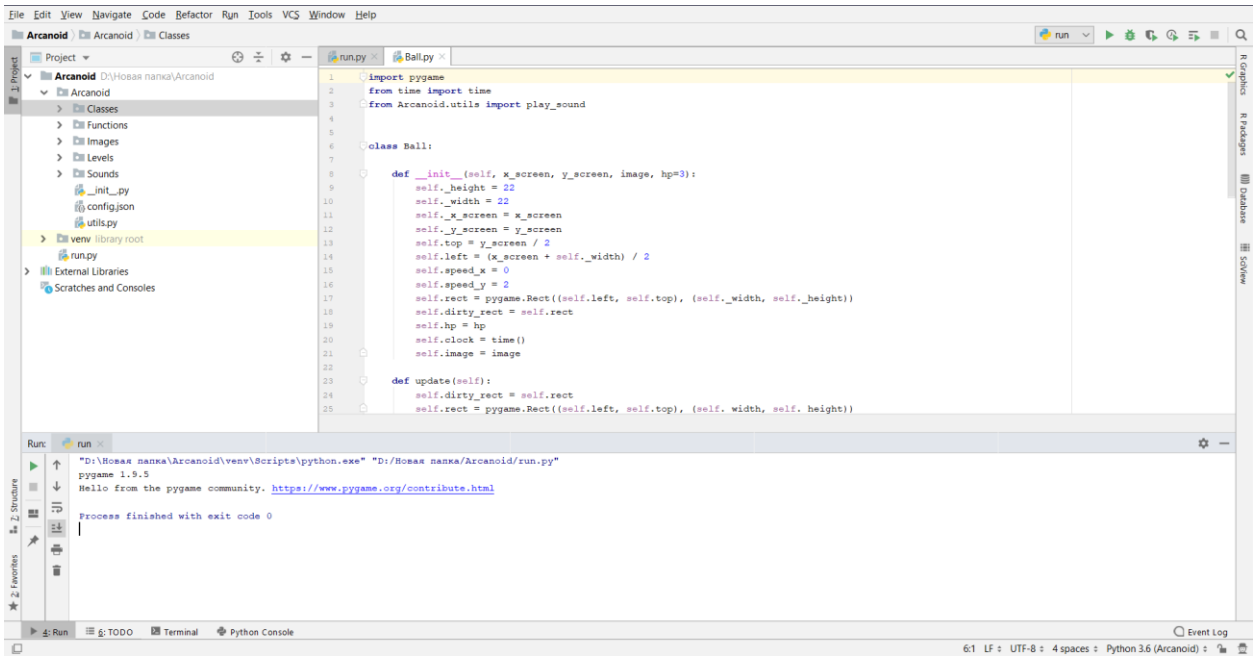


Рисунок 3.1 – Головне меню ICP PyCharm

PyCharm – інтегроване середовище розробки для мови програмування Python. Надає засоби для аналізу коду, графічний відладчик, інструмент для запуску юніт-тестів і підтримує веб-розробку на Django та Flask. PyCharm розроблена чеською компанією JetBrains на основі IntelliJ IDEA [27].

PyCharm працює під операційними системами Windows, Mac OS X і Linux.

Властивості ICP PyCharm:

- статичний аналіз коду, підсвічування синтаксису і помилок
- навігація за проектом і вихідного коду: відображення файлової структури проекту, швидкий перехід між файлами, класами, методами і використаннями методів
- рефакторинг: перейменування, витяг методу , введення змінної, введення константи, підйом і спуск методу і т. д.
- інструменти для веб-розробки з використанням фреймворку Django та Flask
- вбудований відладчик для Python
- вбудовані інструменти для юніт-тестування

- підтримка систем контролю версій: загальний користувальницький інтерфейс для Mercurial, Git, Subversion, Perforce і CVS з підтримкою списків змін та злиття
- велика кількість плагінів, що дозволяють значно розширити можливості ICP

3.3 Обґрунтування вибору бібліотек глибинного навчання

На сьогоднішній день у реалізації нейронних мереж важливу роль реалізації посталених архітектур на налаштувань відіграють, як правило, спеціалізовані пакети, що надають вже повністю реалізовані базові структури та алгоритми для зручної розробки. Найбільш широко використовуються такі популярні бібліотеки, як Theano, Tensorflow та Torch та ін. Дані пакети реалізують програмний інтерфейс для реалізації низькорівневих розрахунків, тому прийняття рішень, в першу чергу, повинно базуватися на суперечностях про продуктивність та зручність розробки. Варто відзначити, що Torch реалізує інтерфейс для мови Lua, не дуже популярної серед програмістів, що істотно знижує зручність і швидкість розробки. Згідно дослідженням Theano and Torch показують приблизно однакову продуктивність на одну й тому ж наборі даних.

Окрім описаної вище незручності використання бібліотеки Torch, існує ще одна, також пов'язана зі зручністю розробки. Інші два згаданих вище інструменти здатні працювати з бібліотекою більш високого рівня Keras [28], що надає загальні, для Tensorflow [29] і Theano, заздалегідь визначені алгоритми та структури. За результатами порівняння та аналізу документації бібліотек було прийнято рішення використовувати для програмної реалізації згорткової нейронної мережі зв'язок бібліотек Tensorflow і Keras.

Бібліотека Keras [28] - програмний продукт, написаний мовою python, що надає високорівневе API для роботи зі штучними нейронними мережами. Даний продукт створювався з метою запобіганню труднощів в розробці

нейронних мереж, пов'язаних із синтаксичними особливостями конкретних пакетів. В січні 2016 року даний інструмент було придбано компанією Google і адаптовано як основне високорівневе надлаштування над бібліотекою Tensorflow.

Бібліотека Theano була написана в першу чергу як розширення мови Python, яка дозволяє ефективно розрахувати математичні вирази, що містять багатомірні масиви. В бібліотеці реалізується базовий набір інструментів для побудови нейронних мереж. Процес створення моделі та визначення її параметрів вимагає написання об'ємного коду, що включає реалізацію класу моделі, самостійного визначення її параметрів, реалізації методів, що визначають функцію помилок, правило обрахування градієнтів, способи зміни ваги нейронів. Бібліотека Caffe реалізована на мові програмування C++. Топологія нейромереж, вихідні дані та спосіб навчання задаються за допомогою конфігураційних протоколів (застосовується технологія і протокол передачі даних) у прототиповому форматі. Побудова структури мережі виконується з простотою та зручністю.

Бібліотека Caffe підтримується досить великою спільнотою розробників та користувачів і на сьогоднішній день є самою розповсюдженою бібліотекою глибинних навчальних програм широкого кола застосування. Через те, що в останні роки методи машинного навчання отримали велику популярність і застосовуються в безлічі різних областей, існує великий вибір бібліотек для реалізації моделей машинного навчання. В таблиці 3.1 представлені найбільш відомі та поширені бібліотеки.

Таблиця 3.1 - Порівняння бібліотек глибинного навчання

Назва бібліотеки	Theano	Caffe	Torch	TensorFlow
Мова програмування	Python	Python, C++	Lua	Python
Спеціалізація на методах машинного навчання	Різноманітні види регресій, нейронні мереж	Нейронні мережі	-	Різноманітні види регресій, нейронні мережі
Швидкість роботи	Середня	Висока	Низька	Висока
Якість позпаралелювання між декількома пристроями	-	-	-	+
Допоміжі функції (автоматичне обрахування градієнта та ін.)	-	+ (зарахунок допоміжних бібліотек Python)	+ (зарахунок допоміжних бібліотек Python)	+

3.3.1 TensorFlow

TensorFlow [29] - це бібліотека програмного забезпечення з відкритим програмним кодом для задач машинного навчання, розробленого компанією Google. Вона дозволяє створювати та навчати нейронні мережі різної архітектури, що знаходять своє застосування у виявленні та розпізнаванні образів, пошуку взаємозв'язків. TensorFlow також включає в себе TensorBoard, який представляє собою засоби візуалізації в браузері для оцінки ефективності навчання та мережевих параметрів моделі.

TensorFlow досягає своєї продуктивності завдяки паралельній обробці задач між центральними та графічними процесорами GPU (у тому числі декількох одночасно) і навіть горизонтальне масштабування за допомогою gRPC. Tensorflow підтримує мови програмування Python і C++.

Кожне обчислення в TensorFlow представляється у вигляді графу потоку даних, графу обчислень. Граф обчислень є моделлю, яка описує як будуть виконуватися обчислення. Важливо зауважити, що складання графа обчислень і виконання операцій в заданій структурі - два різних процеси. Граф складається з плейсхолдеров (tf.Placeholder), змінних (tf.Variable) і операцій. У ньому виробляється обчислення тензорів - багатовимірних масивів, які можуть бути числом або вектором.

Графи виконуються в сесіях (tf.Session). Існують два типи сесій: звичайні та інтерактивні (tf.InteractiveSession); інтерактивна сесія підходить для виконання в консолі. Сесія зберігає стан змінних (Variables) і черг (queues). Явне створення сесій і графів гарантує належне звільнення ресурсів пам'яті. У графі кожна вершина має 0 або більше входів і 0 або більше виходів, і являє собою реалізацію операції. Тензорами є ребра графа, а саме масиви довільного розміру (тип масиву вказується під час побудови графа). Особливі вершини, що управляють залежності (control dependencies), також можуть бути в графі: вони вказують, що вихідний вузол для контрольної залежності повинен закінчити виконання до того, як вузол одержувача контрольної залежності почне виконуватися.

Кожна операція має назву і являє собою абстрактне обчислення (наприклад, операція суми). У операції можуть бути атрибути. Ядро - специфічна реалізація операції, яка може бути виконана на певному типі пристрою (центральний або графічний процесор).

Змінна - особливий вид операції, який повертає лічильник на постійно мінливий тензор: така змінна не зникає після одиничного використання графа. Лічильники на подібні тензори передаються чисельним операціями, які потім змінюють вказаний тензор. У завданнях машинного навчання,

параметри моделі зазвичай зберігають тензори в змінних, які оновлюються на кожному кроці навчання.

3.3.2 OpenCV

OpenCV (Open Source Computer Vision Library) - це бібліотека програмного забезпечення для комп'ютерного зору та машинного навчання. OpenCV побудований для забезпечення загальної інфраструктури програм комп'ютерного зору та прискорення використання машинного сприйняття в комерційних продуктах [30].

Бібліотека має понад 2500 оптимізованих алгоритмів, що включає повний набір як класичних, так і сучасніших алгоритмів комп’ютерного зору та машинного навчання. Їх можна використовувати для виявлення та розпізнавання облич, ідентифікації об'єктів, класифікації людських дій у відео, відстеження рухів камери, відстеження рухомих об'єктів, вилучення 3D-моделей об'єктів, створення 3D-хмар точок із стереокамер, зшивання зображень для отримання якісної роздільної здатності зображення цілої сцени, знайти подібні зображення з бази даних зображень, видалити червоні очі із зображень, зроблених за допомогою спалаху, стежити за рухами очей, розпізнавати декорації та встановлювати маркери, щоб накласти їх на доповнену реальність тощо. OpenCV налічує понад 47 тисяч користувачів користувачів спільноти та передбачає кількість завантажень, що перевищує 18 мільйонів. Бібліотека широко використовується в компаніях, дослідницьких групах та державних установах.

3.3.3 Numpy

Numpy — це розширення мови Python, яке додає підтримку потужних багатовимірних масивів і матриць, разом з великою бібліотекою математичних функцій для операцій з цими масивами. Попередник Numpy,

Numeric, був створений Jim Hugunin. Numpy — відкрите програмне забезпечення і має у свому штабі багато розробників [1].

Оскільки Python є інтерпретованою мовою, математичні алгоритми часто працюють в ньому значно повільніше чим у компільованих мовах, таких як C або навіть Java. NumPy намагається вирішити цю проблему для великої кількості обчислювальних алгоритмів, забезпечуючи підтримку багатовимірних масивів, багатьох функцій та операторів для роботи з ними. Таким чином, будь-який алгоритм, який може бути виражений переважно як послідовність операцій над масивами та матрицями, працює так само швидко, як і еквівалентний код, написаний на C.

NumPy можна хорошою вільною альтернативою MATLAB, оскільки мова програмування MATLAB зовні нагадує NumPy: обидві інтерпретовані та дозволяють користувачам писати швидкі програми, тоді як більшість операцій проводяться над масивами або матрицями, а не над скалярами. Перевагою MATLAB є велика кількість доступних додаткових наборів інструментів, зокрема таких, як пакет Simulink. Основними пакетами, що доповнюють NumPy, є: SciPy — бібліотека, що додає більше MATLAB-подібної функціональності; Matplotlib — пакет щоб створювати графіку в стилі MATLAB. Внутрішньо як MATLAB, так і NumPy базується на бібліотеці LAPACK, призначеної для вирішення основних задач лінійної алгебри.

3.4 Граф-схема алгоритму

Граф-схема алгоритму (ГСА) — кінцевий зв'язний орієнтований граф, вершини якого відповідають операторам відповідають операторам, а дуги задають порядок проходження вершин алгоритму, де число вершин графа – це число дуг. У широкому сенсі вершинам графа відповідають не тільки операторні вершини, а й умовні, початкова та кінцева вершини і т.д. При перегляді паралельних алгоритмів вводиться поняття «паралельної граф-

схеми алгоритму», до якої входять вершини розпаралелювання / синхронізації, функціональність яких зазвичай поєднується. Іноді до ГСА вводяться вершини додаткових типів: об'єднання альтернативних, фіктивні операторні вершини, вершини маркування, очікувальні вершини.

Однак не будь-який орієнтований граф, складений з вершин зазначених типів, може бути ототожнений з коректним алгоритмом. Наприклад, з операторної вершини не може виходити більше однієї дуги. Тому на практиці зазвичай обмежуються розглядом підкласу граф-схем алгоритмів, що задовольняють умовам безпеки, живучості і стійкості. Алгоритми перетворення ГСА, що є підмножиною алгоритмів обробки графів загального вигляду, зачасту мають суттєві відмінності через використання особливих властивостей ГСА, що дозволяє їх спрощення, зниження часової або об'ємної складності.

До складу граф-схеми алгоритму можуть бути виділені більші елементи, представлені підмножинами її вершини і дуг: гілки (лінійні ланцюжки або ділянки вершин) і фрагменти (початковий, паралельний, альтернативний, циклічні з перед-, постумовою і перериванням). Еквівалентним записом граф-схеми коректного алгоритму є дерево фрагментів, що відбиває порядок вкладеності фрагментів.

Граф схема алгоритму роботи програми розпізнавання рукописного тексту зображена на рисунку 3.2.

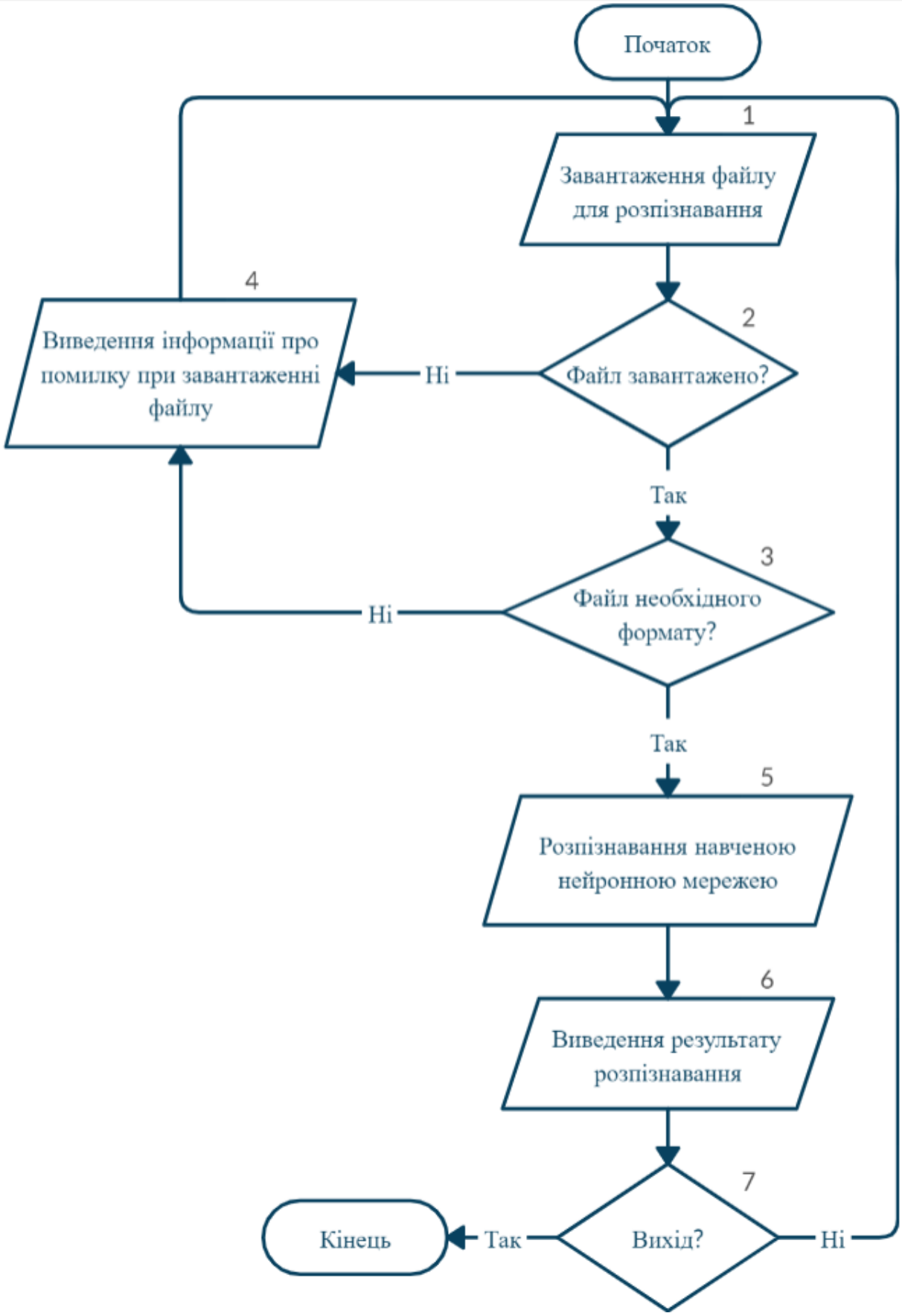


Рисунок 3.2 – Граф-схема алгоритму програми

3.5 Тестування та аналіз результату роботи програми розпізнавання рукописного тексту

Розроблювана програма працює на серверній стороні. Web-додатки зазвичай взаємодіють з користувачем за допомогою елементів форм і змінних GET або POST. У разі використання методу GET всі значення, передані додатком можуть бути видні безпосередньо в URL, в той час як у випадку POST-запитів для визначення того, що вводить користувач, часто доводиться вивчити вихідний текст сторінки, що містить форму.

Для того, щоб отримати доступ до розпізнавання спочатку необхідно перейти до нього у меню, натиснувши кнопку «Розпочати» (рисунок 3.3).

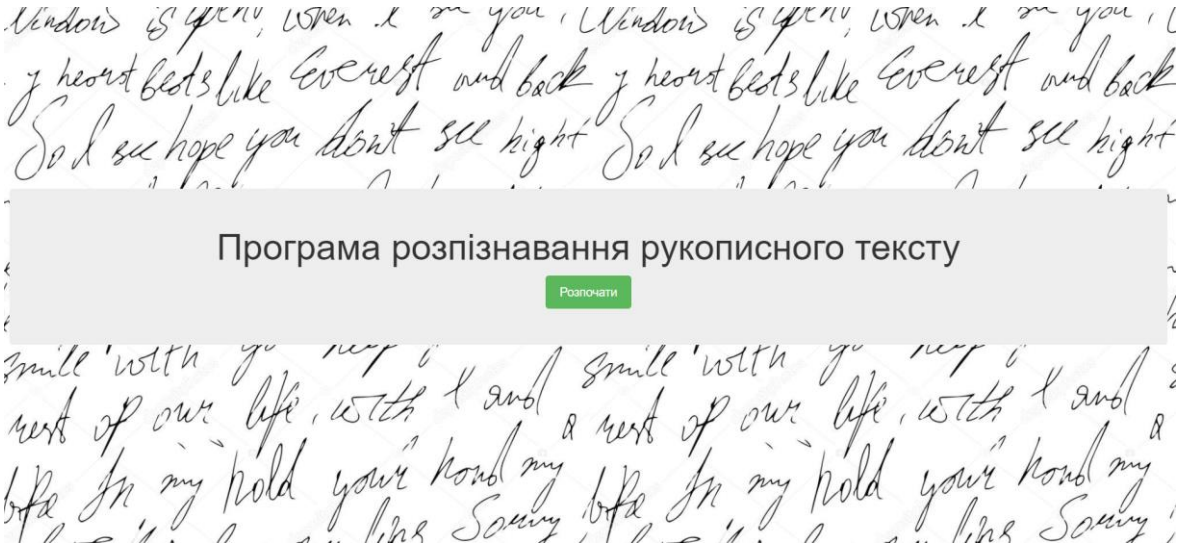


Рисунок 3.3 – Початкова сторінка програми

Після цього буде направлено на сторінку для завантаження зображення. Допустимі формати для завантаження «.png» «.jpg» (рисунок 3.4).

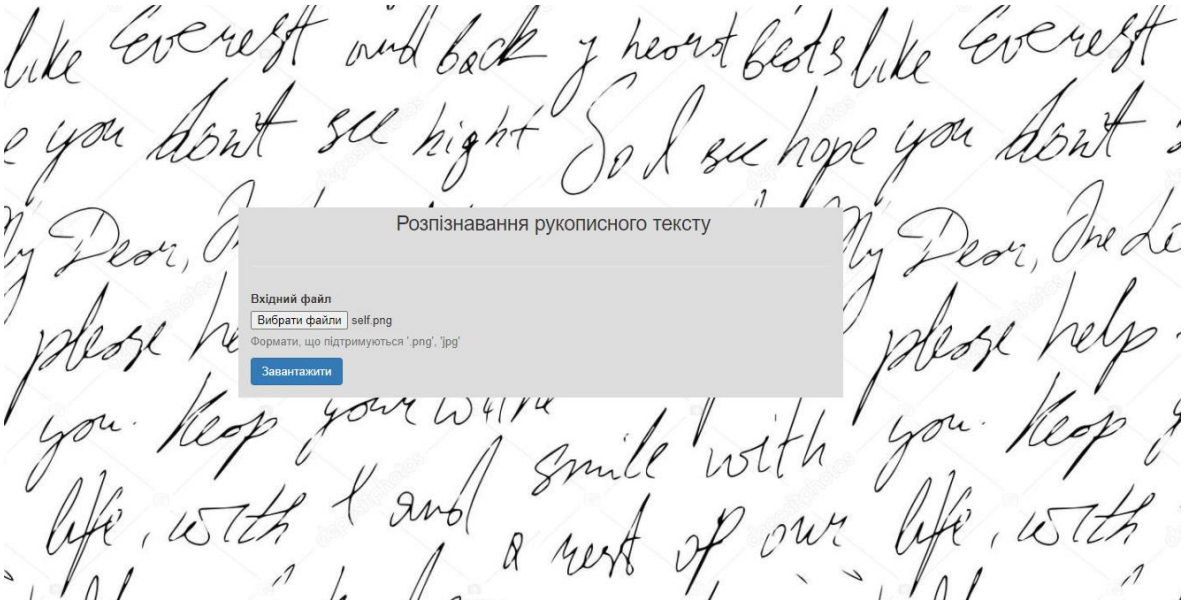


Рисунок 3.4 – Сторінка завантаження зображення

Якщо буде завантажено файл неправильного формату, або ж не завантажено зовсім, програма видасть помилку, як зображено на рисунку 3.5.

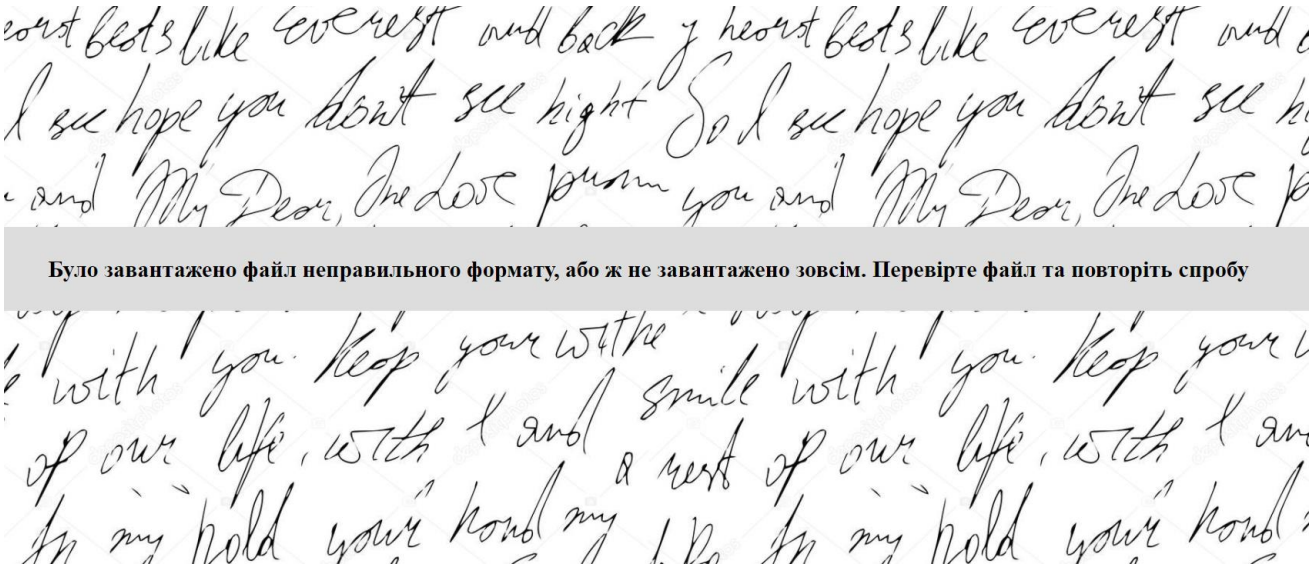


Рисунок 3.5 – Повідомлення про помилку при виборі неправильного формату

Якщо користувач завантажив корректний файл, то після натиснення кнопки «Завантажити» його буде перенаправлено на сторінку з результатом розпізнавання (рисунок 3.6).

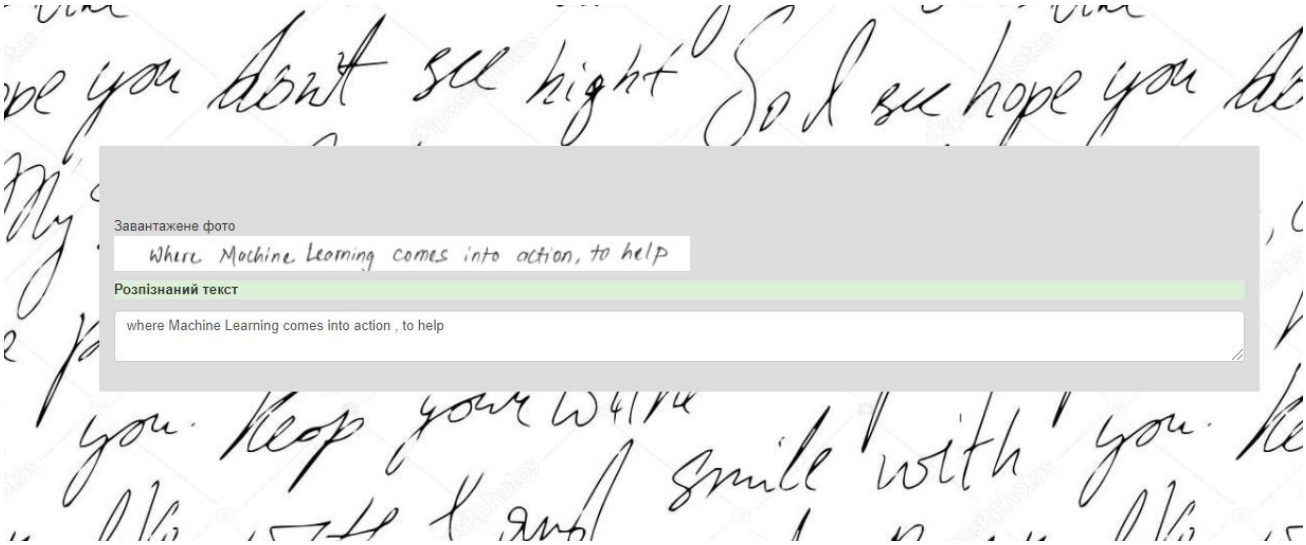


Рисунок 3.6 – Результат розпізнавання

Для тестування роботи програми класифікації тональності речень взяли загальнодоступуний в мережі Інтернет набір даних IAM Handwriting Database [31]. Цей набір даних містить форми рукописного англійського тексту, які можна використовувати для навчання та тестування розпізнавачів рукописних текстів та проведення експериментів з ідентифікації та перевірки письменника. Обсяг навчальної вибірки був 6000 речень. Обсяг тестової вибірки був 1000 речень. Порівняння результатів роботи розробленої програми розпізнавання рукописного тексту з аналогами, розробленими іншими дослідниками, наведено у таблиці 3.2.

Таблиця 3.2 – Порівняння результатів роботи програми розпізнавання рукописного тексту з аналогами

Коефіцієнт помилок	CITlab	LIMSI	Розроблена програма розпізнавання
Символи	5.44 %	5.5 %	5.23 %
Слова	15.5 %	15.0 %	13.45 %

Коефіцієнт похибки слів (WER) визначається за формулою:

$$WER = \frac{(i_w + s_w + d_w)}{n_w},$$

де n_w - кількість слів у довідковому тексті, s_w - кількість заміщених слів, d_w - кількість видалених слів та i_w - кількість вставлених слів, необхідних для перетворення вихідного тексту в цільовий.

Коефіцієнт похибки символів (CER) визначається за формулою:

$$CER = \frac{(i + s + d)}{n},$$

де n – загальна кількість символів та мінімальна кількість вставок символів i, заміни s та видалення d.

Із таблиці 3.2 видно, що розроблена програма має кращі коефіцієнти помилок ніж дві програми-аналога, тобто точність розпізнавання була покращенна, отже мета роботи була досягнута.

3.4 Висновки

В результаті порівняльного аналізу на основі об'єктивних переваг було обрано мову програмування Python.

Після аналізу існуючих середовищ програмування для обраної мови та платформи було обрано середовище програмування PyCharm, яке задовольняє усім потребам при розробці програмного забезпечення та має зручний графічний інтерфейс.

Було досліджено декілька основних бібліотек глибинного навчання, в результаті чого обраною бібліотекою для створення програми стала бібліотека TensorFlow, а допоміжними стали OpenCV для обробки зображень та Numpy для алгоритмічних обчислень. Тестування програми показало, що збоїв у її роботі немає, а точність розпізнавання була покращена.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть к.т.н., доц. кафедри КН Колодний В. В. та к. т. н. ст. викладач кафедри КН Озеранський В. С.

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

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали експерта	
	1. Колодний В. В.	2. Озеранський В. С.
	Бали, виставлені експертами:	
1	4	4
2	3	3
3	4	4
4	4	4
5	3	3
6	3	4
7	3	3
8	3	4
9	4	4
10	4	3
11	3	4
12	3	4
Сума балів	$СБ_1 = 42$	$СБ_2 = 44$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_1}{2} = 43$	

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

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

Для розробки нового програмного продукту потрібні наступні витрати.

Основна заробітна плата для розробників визначається за формулою:

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

де М – місячний посадовий оклад конкретного розробника;

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

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

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

Таблиця 4.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5400	227,27	8	1818,16
Інженер-програміст	5000	163,63	40	6545,2
Всього				8363,36

Розрахуємо додаткову заробітну плату:

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

Нарахування на заробітну плату операторів НЗП розраховується як 22% від суми їхньої основної та додаткової заробітної плати:

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

$$Нзп= (8363,36 + 836,33) * \frac{22}{100} = 2023,93 \text{ (грн.).}$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

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

де Ц – балансова вартість обладнання, грн;

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

Т – Термін використання (Т = 2 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

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

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

$$К = \sum_1^n Н_і * Ц_і * К_і , \quad (4.4)$$

де n – кількість комплектуючих;

Ні – кількість комплектуючих і-го виду;

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

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

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

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

Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В * П * Ф * К_п, \quad (4.5)$$

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

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

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

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

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

Розрахуємо інші витрати $В_{ін}$.

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

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

Отже, розрахуємо інші витрати:

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

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

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

$$В = 8363,36+836,33+2023,93+500+142,8+357,5+9199,69= 21477,61 \text{ (грн.)}$$

Розрахуємо загальну вартість наукової роботи за формулою:

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

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

$$В_{заг} = \frac{21477,61}{1} = 21477,61$$

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

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

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

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{21477,61}{0,9} = 23864,01 \text{ (грн)}$$

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності.

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

$$\Delta П_і = \sum_1^n (\Delta П_я * N + П_я \Delta N)_i \text{а} , \quad (4.9)$$

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

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

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

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

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

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 35 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 35 грн), а кількість користувачів, які будуть користуватись збільшиться:

протягом першого року – на 200 користувачів, протягом другого року – на 150 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 700 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 300 грн.

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

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

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

Протягом другого року:

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

Протягом третього року:

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

4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

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

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

$$Е_{абс} = (ПП\text{-}PV), \quad (4.10)$$

де ПП – вартість чистих прибутків;

PV – теперішня вартість інвестицій PV = ЗВ, грн.

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

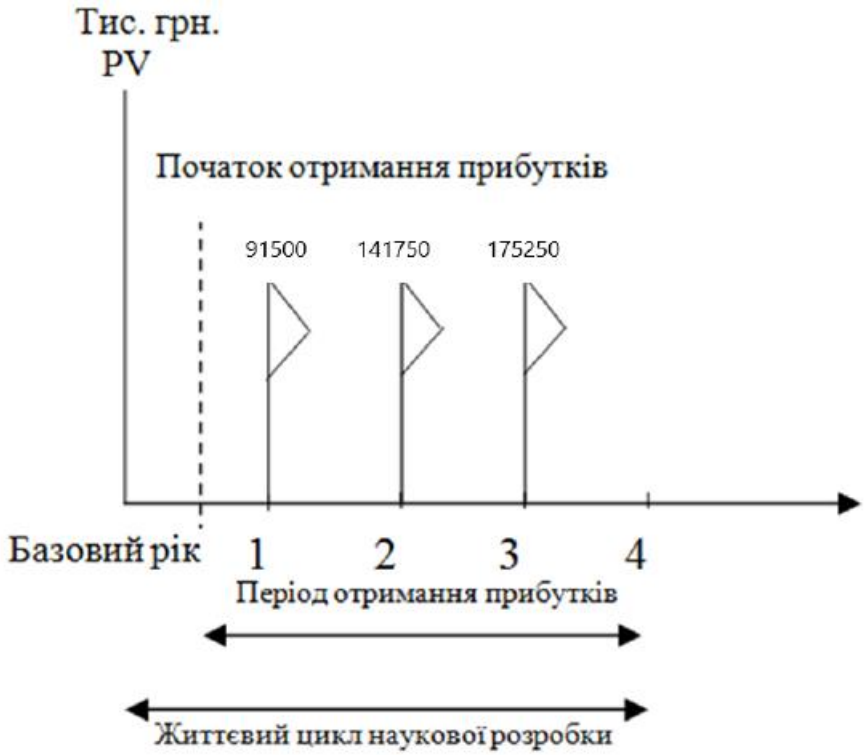


Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

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

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

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

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

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

Розрахуємо вартість чистого прибутку:

$$ПП = \frac{23864,01}{(1+0,1)^0} + \frac{91500}{(1+0,1)^2} + \frac{141750}{(1+0,1)^3} + \frac{175250}{(1+0,1)^4} = 325680,81 \text{ (грн)}$$

Розрахуємо $Е_{абс}$:

$$Е_{абс} = 325680,81 - 23864,01 = 301816,8 \text{ грн.}$$

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

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

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

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

PV – теперішня вартість інвестицій PV = ЗВ, грн;

Тж – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$Е_в = \sqrt[Т]{1 + \frac{301816,8}{23864,01}} - 1 = 1,38 \text{ або } 138\ \%$$

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

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2020 році в Україні d = 0,2;

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

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

Оскільки $E_в = 138\ \% > \tau_{мін} = 0{,}3 = 30\%$, то й інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

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

$$T_{ок} = \frac{1}{E_в}$$

Отже, розрахуємо термін окупності вкладених інвестицій:

$$T_{ок} = \frac{1}{1{,}38} = 0{,}72 \text{ року}$$

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

4.5 Висновок

В даному розділі було проведено економічне обґрунтування доцільності розробки програми для розпізнавання рукописного тексту з використанням нейронної мережі. Незалежними експертами було здійснено оцінювання комерційного потенціалу розробки, за результатами якого було визначено, що нова розробка має високий рівень комерційного потенціалу, оскільки середньоарифметична сума балів становить 43. Також було виконано прогнозування витрат на виконання розробки, де розраховано основну заробітну плату кожного із розробників, додаткову заробітну плату всіх розробників, нарахування на заробітну плату, амортизацію обладнання, комп’ютерів та приміщень, витрати на допоміжні матеріали, витрати на силову електроенергію тощо. Загальна сума витрат на виконання означених робіт склала 23864,01 грн. Виконано розрахунок ефективності вкладених

інвестицій та періоду їх окупності. Визначено, що абсолютна ефективність вкладених інвестицій становить 325680,81 грн, і це свідчить про те, що вкладання коштів на виконання та впровадження результатів НДДКР є доцільним. Було розраховано відносну (щорічну) ефективність вкладених в наукову розробку інвестицій – 138 %, її величина більша за мінімальну (бар'єрну) ставку дисконтування, отже інвестор буде зацікавлений у фінансуванні даної наукової розробки. Проведено розрахунок терміну окупності - 0,72 року (9 місяців). Це означає, що вже починаючи з 10 місяця розробка буде приносити прибуток. Взагальному можна зробити висновок, що фінансування розробки програми для розпізнавання рукописного тексту з використанням нейронної мережі є економічно доцільним проектом.

ВИСНОВКИ

В результаті виконання магістерської кваліфікаційної роботи розв’язано задачу розробки інформаційної технології та програмного забезпечення розпізнавання рукописного тексту згортковою нейронною мережею.

В першому розділі аналіз предметної області показав, що розпізнавання рукописного тексту є важливою і актуальною задачею, оскільки є багато областей де потрібно розпізнавати рукописний текст, зокрема у промисловості, науці, техніці, банківській сфері, роботі поштових компаній тощо. Було визначено, що тоді як задача розпізнавання друкованих текстів вирішена, розпізнавання рукописних текстів потребує додаткових досліджень.

В другому розділі були розглянуті сучасні методи обробки зображень, методи побудови математичних моделей машинного навчання, та обрано для реалізації метод згорткової рекурентної нейронної мережі, оскільки вона має кращу архітектуру за звичайні нейронні мережі виражену у сімох згорткових шарах та двох рекурентних шарах нейронів і способу їх взаємодії між собою. Також було проведено опис роботи моделі нейронної мережі для розпізнавання рукописного тексту. Розроблено структуру нейронної мережі, яка проектувалась із розрахунком на можливе покращення для збільшення точності розпізнавання рукописного тексту.

В третьому розділі в результаті порівняльного аналізу на основі об'єктивних переваг було обрано мову програмування Python. Після аналізу існуючих середовищ програмування для обраної мови та платформи було обрано середовище програмування PyCharm, яке задовольняє усім потребам при розробці програмного забезпечення та має зручний графічний інтерфейс. Було досліджено декілька основних бібліотек глибинного навчання, в результаті чого обраною бібліотекою для створення програми стала бібліотека TensorFlow, а допоміжними стали OpenCV для обробки зображень

та Numpy для алгоритмічних обчислень. Тестування програми показало, що збоїв у її роботі немає.

У четвертому розділі було проведено економічне обґрунтування доцільності розробки програми для розпізнавання рукописного тексту з використанням згорткової нейронної мережі. Було визначено, що нова розробка має високий рівень комерційного потенціалу, оскільки середньоарифметична сума балів становить 43. Також було виконано прогнозування витрат на виконання розробки. Визначено, що абсолютна ефективність вкладених інвестицій становить 325680,81 грн, і це свідчить про те, що вкладання коштів на виконання та впровадження результатів НДДКР є доцільним. Було розраховано відносну (щорічну) ефективність вкладених в наукову розробку інвестицій – 138 %, її величина більша за мінімальну (бар'єрну) ставку дисконтування, отже інвестор буде зацікавлений у фінансуванні даної наукової розробки. Проведено розрахунок терміну окупності - 0,72 року (9 місяців). Це означає, що вже починаючи з 10 місяця розробка буде приносити прибуток. Взагальному можна зробити висновок, що фінансування розробки програми для розпізнавання рукописного тексту з використанням нейронної мережі є економічно доцільним проектом.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Добера Р. О. Пакети Python для Data Science. НТКП ВНТУ. Факультет інформаційних технологій та комп'ютерної інженерії./Добера Р. О. [Електронний ресурс] – режим доступу : https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/10534.
2. Розпізнавання образів – [Електронний ресурс]. – Режим доступу: http://helpiks.org/5-72193.html
3. Ivakhnenko A.G. The group method of data handling-a rival of the method of sto-chastic approximation / A.G. Ivakhnenko // Soviet Automatic Control. — Vol. 13, N 3. — 1968. — P. 43–55.
4. Fukushima K. Neocognitron: A self-organizing neural network model for a mecha-nism of pattern recognition unaffected by shift in position / K. Fukushima // Biol. Cybern. — Vol. 36. —1980. — P. 193–202.
5. Schmidhuber J. Deep learning in neural networks: An overview / J. Schmidhuber // Neural networks. — Vol. 61. — 2015. — P. 85–117.
6. Hochreiter S. Long short-term memory / S. Hochreiter, J. Schmidhuber // Neural computation. — Vol. 9, N 8. — 1997. — P. 1735–1780.
7. GloVe: Global Vectors for Word Representation – [Електронний ресурс]. – Режим доступу: https://nlp.stanford.edu/pubs/glove.pdf
8. Sentence Classification CNN – [Електронний ресурс]. – Режим доступу: https://github.com/umairqadir97/Sentense-Classification-with-CNN
9. ABBY FineReader – [Електронний ресурс]. – Режим доступу: https://www.abbyy.com/uk/finereader/
10. Штучні нейронні мережі – [Електронний ресурс]. – Режим доступу: http://archive.ws-conference.com/wp-content/uploads/pw0060
11. Srihari S. N. Offline Chinese Handwriting Recognition: A Survey./ Srihari S. N., Yang X., Ball G. R. –Frontiers of Computer Science in China, 1(2), 2007.

12. Zhang G. P.Neural networks for classification: a survey./ Zhang G.P. –IEEE Transactions on Systems, Man, and Cybernetics, 2000. –Part C: Applications and Reviews, 30(4). –p. 451–462.

13. Gomez Sanchez E. On-Line Character Analysis and Recognition with Fuzzy Neural Networks./ Gomez Sanchez E., Dimitriadis Y. A., Mas M.S.-R., Garcia P. S., Cano Izquierdo J. M., Coronado J. L.–Intelligent Automation and Soft Computing, Vol. 7, No. 3., 1998. –p. 161–162

14. Нейронні мережі прямого поширення – [Електронний ресурс]. – Режим доступу: http://reference.wolfram.com/applications/neuralnetworks/NeuralNetworkTheory/2.5.1

15. Карти Кохонена – [Електронний ресурс]. – Режим доступу: http://matlab.exponenta.ru/neuralnetwork/book2/20/kokhonen.php

16. Hammer, Barbara; Micheli, Alessio; Sperduti, Alessandro; Strickert, Marc (2004). Recursive self-organizing network models. Neural Networks 17: 1061–1085.

17. Згорткові нейронні мережі – [Електронний ресурс]. – Режим доступу: http://ru.datasides.com/code/cnn-convolutional-neural-networks/

18. Happel, Bart and Murre, Jacob. The Design and Evolution of Modular Neural Network Architectures. Neural Networks, 7: 985—1004; 1994.

19. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

20. Y. Bengio, P. Y. Simard, and P. Frasconi. Learning longterm dependencies with gradient descent is difficult. NN, 5(2):157–166, 1994.

21. A. Graves, S. Fern´andez, F. J. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In ICML, 2006.

22. A. Graves, S. Fern´andez, F. J. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In ICML, 2006.

23. M. Mukkamala and M. Hein. Variants of RMSProp and Adagrad with Logarithmic Regret Bounds. In Proceedings of the 3\ th International Conference on Machine Learning, pages 2545-2553, 2017.

24. V. Papavassiliou, T. Stafylakis, V. Katsouros, and G. Carayannis. Handwritten document image segmentation into text lines and words. Pattern Recognition, 43(l):369-377, 2010.

25. Що таке Python? [Електронний ресурс]. – Режим доступу: URL: http://www.plug.org.ua/documentation/about-python

26. . Сравнение Python с другими языками [Електронний ресурс]. – Режим доступу: URL: http://www.edu-main.narod.ru/Programming/stats/python

27. PyCharm — интеллектуальная Python IDE [Електронний ресурс]. – Режим доступу: URL: https://jetbrains.ru/products/pycharm

28. F. Chollet, “Keras” [Електронний ресурс]. – Режим доступу: URL: https://github.com/fchollet/keras, 2015

29. Tensorflow: Large-scale machine learning on heterogeneous distributed systems [Електронний ресурс]. – Режим доступу: URL: https://arxiv.org/abs/1603.04467

30. OpenCV [Електронний ресурс]. – Режим доступу: URL: https://opencv.org/about/

31. IAM Handwriting Database [Електронний ресурс]. – Режим доступу: URL: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database