МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
В.о. завідувача кафедри інтелектуальних інформаційних систем, канд.техн.наук, доц.
_____ Є. В. Сіденко
«____»_____2023 року

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

СИСТЕМА РОЗПІЗНАВАННЯ РУКОПИСНИХ СИМВОЛІВ НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

Спеціальність 122 «Комп'ютерні науки»

122 – МКР – 601.21710220

Виконала студентка 6-го курсу, групи 601
_____ А. В. Овсянникова
«16» лютого 2023 р.

Керівник: канд. пед. наук, доцент
_____Н. М. Болюбаш
«16» лютого 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри інтелектуальних інформаційних систем, канд.техн.наук, доц.
_____ Є. В. Сіденко
«___» _____ 20__ р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу

Овсянниковій Анлегіні Володимирівні

1. Тема магістерської кваліфікаційної роботи «Система розпізнавання рукописних символів на основі згорткових нейронних мереж».

Керівник роботи Болюбаш Надія Миколаївна, канд. пед. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «03» листопада 2022 р. № 199

2. Строк подання студентом роботи 16 лютого 2023 р.

3. Вхідні (початкові) дані до роботи: загальні відомості про сучасні підходи до розпізнавання рукописних символів, набір даних MNIST, який містить 70 тисяч зображень із рукописними цифрами.

Очікуваний результат роботи: система розпізнавання символів рукописних текстів, побудована на основі згорткових нейронних мереж.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- розкриття теоретичних засад розпізнавання зображень, здійснення аналізу сучасних підходів до розпізнавання символів рукописних текстів та вибір нейромережевої моделі для їх розпізнавання;

- обґрунтування вибору технологій та інструментальних засобів розробки системи розпізнавання рукописних символів;
- розробка та здійснення програмної реалізації системи розпізнавання рукописних символів на основі згорткових нейронних мереж.

5. Перелік графічного матеріалу: презентація, рисунки, таблиці.

6. Завдання до спеціальної частини: Охорона праці на підприємстві при загрозі або виникненні надзвичайних ситуацій.

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	докт.біол.н.,професор Л. І. Григор'єва	
Методична частина	канд.пед.н., доцент Н.М. Болюбаш	

Керівник роботи __канд. пед. наук, доц. Болюбаш Н. М._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Овсянникова А. В._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «__07__» ____листопада___ 2022 р.

КАЛЕНДАРНИЙ ПЛАН

виконання магістерської кваліфікаційної роботи

Тема: «Система розпізнавання рукописних символів на основі згорткових нейронних мереж»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Визначення керівника і теми МКР. Подання заяви на затвердження теми МКР	01.09.2022	20.10.2022	Виконано
2.	Отримання завдання на виконання МКР	21.10.2022	10.11.2022	Виконано
3.	Складання календарного плану	11.11.2022	15.11.2022	Виконано
4.	Огляд літератури за темою дослідження. Аналіз існуючих підходів до розпізнавання рукописних символів	16.11.2022	27.11.2022	Виконано
5.	Проходження переддипломної практики, збір та аналіз матеріалів до МКР	28.112022	18.12.2022	Виконано
6.	Аналіз предметної області розпізнаваня символів у рукописних текстах та розробка технічного завдання	19.12.2022	22.12.2022	Виконано
7.	Проєктування та програмна реалізація системи розпізнавання рукописних символів із аналізом отриманих результатів	23.12.2022	15.01.2023	Виконано
8.	Робота над розділами фахової частини МКР	16.01.2023	24.12.2023	Виконано
9.	Розробка методичної частини МКР та спеціальної частини з охорони праці	25.01.2023	01.02.2023	Виконано
10.	Обговорення отриманих результатів з керівником та попередній захист МКР	02.02.2023	3.02.2023	Виконано
11.	Корегування роботи за результатами попереднього захисту	4.02.2023	6.02.2023	Виконано
12.	Остаточне оформлення пояснювальної записки та слайдів доповіді до зихисту	7.02.2023	9.02.2023	Виконано
13.	Подання рецензенту та рецензування МКР	9.02.2023	12.02.2023	Виконано
14.	Подання МКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	15.02.2023	16.02.2023	Виконано
15.	Захист МКР перед ЕК	23.02.2023	23.02.2023	Виконано

Розробив студент Овсянникова А.В.
(прізвище та ініціали) (*підпис*)

Керівник роботи канд.пед.н., доц. Болюбаш Н.М.
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«12» листопада 2022 р.

АНОТАЦІЯ

до магістерської кваліфікаційної роботи
студентки групи 601 ЧНУ ім. Петра Могили

Овсянникової Ангеліни Володимирівни

на тему: «**СИСТЕМА РОЗПІЗНАВАННЯ РУКОПИСНИХ СИМВОЛІВ НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ**»

Керівник: канд.пед.н доцент Болюбаш Надія Миколаївна

Магістерська кваліфікаційна робота присвячена розробці та здійсненню програмної реалізації системи розпізнавання рукописних символів на основі згорткових нейронних мереж. Що є актуальним при розпізнаванні рукописних документів, оскільки підвищує ефективність автоматизації при переносі інформації з паперових рукописних носіїв до електронних документів.

Об'єкт дослідження – процес розпізнавання зображень в автоматизованих системах обробки інформації.

Предмет дослідження – програмні засоби для автоматичного розпізнавання рукописних символів, побудовані на основі згорткових нейронних мереж.

Мета дослідження – підвищення ефективності розпізнавання рукописних символів шляхом створення системи розпізнавання на основі згорткових нейронних мереж.

Дипломна робота складається з фахової, методичної і спеціальної частини з охорони праці. Пояснювальна записка дипломної роботи складається зі вступу, трьох розділів, висновків та додатку. У першому розділі розкрито теоретичні засади оптичного розпізнавання символів, розглянуто сучасні підходи до створення систем їх розпізнавання. У другому розділі обґрунтовано вибір технологій і засобів розробки системи розпізнавання рукописних символів. У третьому розділі описано проєктування та програмну реалізацію системи розпізнавання рукописних символів на основі згорткових нейронних мереж. У спеціальній частині з охорони праці розглядаються питання охорони праці та безпеки у надзвичайних ситуаціях.

Дипломна робота містить ___ сторінку (без додатків), ___ рисунків, ___ таблиці, ___ джерел, ___ додаток.

Ключові слова: оптичне розпізнавання символів, розпізнавання рукописних текстів, сегментація, згорткова нейронна мережа, класифікатор.

ABSTRACT

to the master's qualification work
by the student of the group 601 of Petro Mohyla Black Sea National University

Ovsiannykovoi Anheliny Volodymyrivny

on the subject: «**HANDWRITTEN CHARACTER RECOGNITION SYSTEM BASED ON CONVOLUTIONAL NEURAL NETWORKS**»

Leader: Ph.D., associate professor Bolyubash Nadiya Mikolaivna

The master's thesis is devoted to the development and implementation of a software implementation of a system for recognizing handwritten characters based on convolutional neural networks. What is relevant when recognizing handwritten documents, which contributes to the efficiency of automation when transferring information from paper handwritten documents to electronic documents.

Object of research – the process of image recognition in automated information processing systems.

Subject of research – software tools for automatic recognition of handwritten characters built on the basis of convolutional neural networks.

The purpose of the study is to increasing the efficiency of recognition of handwritten characters by creating a recognition system based on convolutional neural networks.

The thesis consists of a professional, methodical and special part on labor protection. The explanatory note of the thesis consists of an introduction, three sections, conclusions and an appendix. In the first chapter, the theoretical foundations of optical character recognition are revealed, modern approaches to the creation of character recognition systems are considered. In the second chapter, the choice of technologies and tools for the development of a character recognition system is substantiated. The third chapter describes the design and software implementation of a handwritten character recognition system based on convolutional neural networks. The special part on labor protection deals with issues of labor protection and safety in emergency situations.

Thesis contains ___ page (without appendices), ___ figures, ___ tables, ___ sources, ___.

Keywords: optical character recognition, handwriting recognition, segmentation, convolutional neural network, classifier.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

ЗНМ – згорткові нейронні мережі

ШІ – штучний інтелект

ШНМ – штучна нейронна мережа

ANN - Artificial Neural Network

CNN – Convolutional Neural Network

DNN – Deep Neural Network

ORC – Optical Character Recognition

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

«СИСТЕМА РОЗПІЗНАВАННЯ РУКОПИСНИХ СИМВОЛІВ НА ОСНОВІ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ»

Спеціальність 122 «Комп’ютерні науки»

122 – МКР – 601.21710220

Виконала студентка 6-го курсу, групи 601
_____ А. В. Овсянникова
«16» лютого 2023 р.

Керівник: канд. техн. наук, доцент
_____Н. М. Болюбаш
«16» лютого 2023 р.

Миколаїв – 2023

ВСТУП

Актуальність. В умовах високих темпів інформатизації усіх сфер сучасного суспільства зростає актуальність задачі оптичного розпізнавання символів (англ. Optical Character Recognition, OCR) та рукописних текстів (англ. Handwriting Recognition), яка включає їх розпізнавання із друкованих текстових документів та рукописних чи реальних зображень. Розпізнавання рукописних символів є непростою задачею, оскільки вони можуть мати великі варіації форми, розміру, нахилу, текстури та фону. Для вирішення цієї проблеми створюють інформаційні системи, здатні розпізнавати символи за їх рукописним зображенням, отриманим із різних джерел, класифікуючи їх, розділивши на попередньо визначені класи, кожен із яких відповідає зображенню окремого символу.

Розпізнавання символів широко застосовується у таких сферах, як автоматичне розпізнавання номерних знаків, сортування поштової кореспонденції, обробка банківських чеків, розпізнавання ідентифікаційних карток, автоматизація цифровізації рукописних документів. Фундаментальними етапами розпізнавання символів є сегментація, виділення ознак і класифікація. Науковці проводять чисельні дослідження, для вирішення цієї задачі, застосовуючи різні підходи та методи: нейронні мережі прямого поширення, рекурентні нейронні мережі, алгоритм k-ближніх сусідів, згорткові нейронні мережі. Найефективнішими є моделі згорткових нейронних мереж (англ. Convolutional Neural Network, CNN), які досягають високої точності класифікації й є найбільш прийнятним інструментом для розпізнавання зображень.

Застосування згорткових нейронних мереж дозволяє з високою точністю розпізнавати окремі рукописні символи. Складною та не є до кінця вирішеною є проблема сегментації зображення рукописного тексту на окремі символи. Тому

розпізнавання символів рукописних документів у процесі їх цифровізації усе ще не є надійним.

Мета дослідження – підвищення ефективності розпізнавання рукописних символів шляхом створення системи розпізнавання на основі згорткових нейронних мереж.

Досягнення поставленої мети обумовлює необхідність вирішення наступних **завдань**:

1) розкрити теоретичні засади розпізнавання зображень, здійснити аналіз сучасних підходів до розпізнавання символів рукописних текстів та обрати нейромережеву модель для їх розпізнавання;
2) обґрунтувати вибір технологій та інструментальних засобів розробки системи розпізнавання рукописних символів;
3) розробити та здійснити програмну реалізацію системи розпізнавання рукописних символів на основі згорткових нейронних мереж.

Об’єктом дослідження є процес розпізнавання зображень в автоматизованих системах обробки інформації.

Предметом дослідження є програмні засоби для автоматичного розпізнавання рукописних символів, побудовані на основі згорткових нейронних мереж.

Методологічною основою дослідження є загальнонаукові аналітичні методи та методи розпізнавання символів, які дозволили вивчити предмет та об’єкт дослідження, дослідити розвиток науково-методичних засад, напрямів та шляхів підвищення ефективності розпізнавання рукописних символів шляхом створення на основі згорткових нейронних мереж інформаційної системи, яка забезпечує цифровізацію рукописних текстових документів.

Наукова новизна одержаних результатів дослідження полягає у тому, що автором: запропоновано та обґрунтовано напрями вдосконалення архітектури згорткової нейронної мережі розпізнавання рукописних символів; одержали подальший розвиток підходи до підвищення точності та надійності розпізнання

символів рукописних документів; узагальнено теоретичні засади створення систем розпізнавання рукописних символів.

Результати дослідження обговорювалися на Всеукраїнській науково-практичній конференції молодих вчених, аспірантів і студентів «Інформаційні технології та інженерія» (7-10 лютого 2023 року) та отримали схвалення.

Практичне значення отриманих результатів полягає в тому, що сформульовані теоретичні положення та практичні рекомендації щодо підвищення точності та надійності розпізнавання рукописних символів можна застосувати при здійсненні цифровізації документів, які містять рукописні тексти трудових книжок, для автоматизації підрахунку стажу при нарахуванні пенсії.

Структура магістерської роботи. Відповідно до мети, завдань і предмета дослідження, магістерська робота містить основну, методичну та спеціальну частини. Основна частина магістерської роботи складається із вступу, трьох розділів, висновку, списку використаних джерел та __ додатків. Загальний обсяг магістерської роботи – __ сторінок, із них основного тексту основної частини – __ сторінок, методичної частини – __ сторінок, спеціальної – __ сторінок. Кількість використаних джерел – __.

1 ТЕОРЕТИЧНІ ЗАСАДИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ІЗ ВИКОРИСТАННЯМ НЕЙРОННИХ МЕРЕЖ

1.1 Розпізнавання рукописних символів при обробці зображень

В даний час однією з основних тенденцій документообігу є його реалізація в електронному вигляді, що дозволяє широко використовувати комп'ютерні технології. Документи, що зберігаються, досить часто є зображеннями сканованих текстів, які або повністю є рукописними, або частково (наприклад, підписи під текстами). При цьому найчастіше найбільший інтерес представляє саме рукописна частина.

У даному досліджені для розпізнавання рукописних символів було обрано розпізнавання дат у записах трудової крижки, яке необхідне для підрахунку стажу особам, що оформляють документи для отримання пенсії по вислузі років. Такі підрахунки здійснюють у відділеннях Пенсійного фонду України. Цифровізація інформації з трудових книжок та підрахунок стажу здійснюється вручну. Автоматизація частини цієї роботи дозволить підвищити ефективність відділів, які здійснюють оформлення пенсійних документів.

Розпізнавання образів є розділом інформаційних технологій, який містить міждисциплінарні дослідження та розвиває методи класифікації та ідентифікації об'єктів різної природи. При розв'язанні задач розпізнавання вихідним матеріалом є зображення отримане із різних джерел – друкованих текстових документів та рукописних чи реальних зображень. Розпізнавання образів із зображення – це завдання ідентифікувати об'єкт, визначити будь-які його властивості із зображення, а також віднести його до множини зображень за певним правилом [2].

Знаходить широке застосування розпізнавання символів та цифр у таких сферах, як автоматичне розпізнавання номерних знаків, сортування поштової

кореспонденції, обробка банківських чеків, розпізнавання рукописних записів у документах при здійсненні їх цифровізації.

Найважливішими кроками у процесі розпізнавання є наступні:

1) зчитування зображення: на цьому етапі отримують значення характерних властивостей об’єкта;

2) обробка зображення: цей етап містить бінаризацію та нормалізацію, основні труднощі – спотворення всіх видів: геометричні, фонові та шумові;

3) характеристична обробка (індексація): на цьому етапі вимірюються характерні властивості об’єкта;

4) розбиття на сегменти: у більшості випадків розбиття на сегменти зводиться до знаходження «однорідних» ділянок;

5) виділення ознак;

5) розпізнавання символів – класифікація;

6) виправлення помилок, прийняття рішення.

Ці кроки виконуються послідовно і результат кожного кроку подається на вхід наступного кроку (рис. 1.1).

Рисунок 1.1 – Послідовність етапів розпізнавання символів

На етапі попередньої обробки зчитаного зображення здійснюють виділення необхідної інформації шляхом приведення його до вигляду, який дозволяє виділяти символи на фоні, виконуючи операції очистки від шуму, фільтрації, згладжування та збільшення контрастності. Якщо текст рукописний, то додатково застосовують підхід по випрямленню символів, так як вони будуть написані з

нахилом. В основному та цьому етапі виконується бінаризація зображення, яка дозволяє чітко виділити рукописний текст та прибрати фон [3].

Розпізнавання рукописних символів – це нетривіальне завдання у галузі штучного зору. Складність задачі розпізнавання рукописного тексту – це велика різноманітність почерків, форм, розмірів букв та різноманіття мов. Також папір із рукописним текстом може містити шуми – дефекти паперу, сторонні плями, що ускладнює весь процес.

Фундаментальними етапами розпізнавання рукописних символів та цифр є сегментація, виділення ознак і класифікація.

Розпізнавання починається з нормалізації та сегментації. Нормалізацією називається зміна кута зображення символу в площині зображення так, щоб рядок із символами був розташований горизонтально. Сегментація зображення передбачає виділення корисної інформації з зображення, з наступною її обробкою.

Сегментація символів здійснюється за допомогою використання моделей розташування символів. Зіставленням різних моделей з реальним зображенням обирається модель, яка найбільш підходить, параметри котрої використовуються, щоб отримати координати символів [4]. Цей підхід дозволяє визначати належність символу до букв або цифр, а це в свою спрощує подальше розпізнання.

Сегментація зображення рукописного тексту містить такі етапи (рис. 1.2):

1) сегментація рядків: виділення на зображенні лініями фрагментів слів;

2) сегментація слів: виділення фрагментів, які містять слова;

3) сегментація символів: розділення зображень слів на окремі символи.

Сторінка рукописного тексту розбивається на рядки, потім рядок розбивається на слова, де пробіл є їх роздільником. Для цього на текст послідовно накладаються фільтри для видалення шумів та визначення меж слів. Текст розбивається на складові – компоненти і розраховується відстань між їх центрами. Як параметр алгоритму приймається якесь граничне значення відстані, яке надалі можна підібрати виходячи з успішності результатів (рис. 1.3).

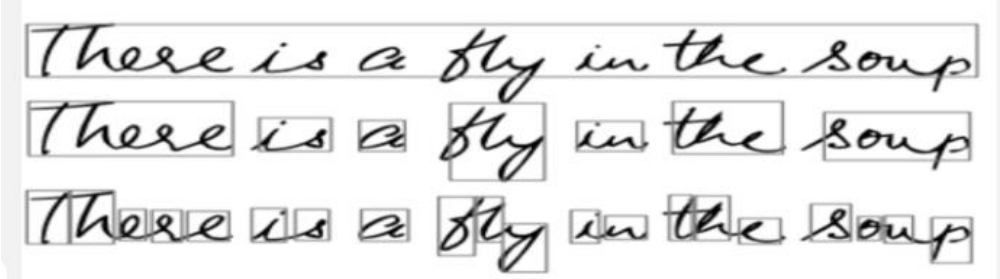


Рисунок 1.2 – Сегментація рукописного тексту

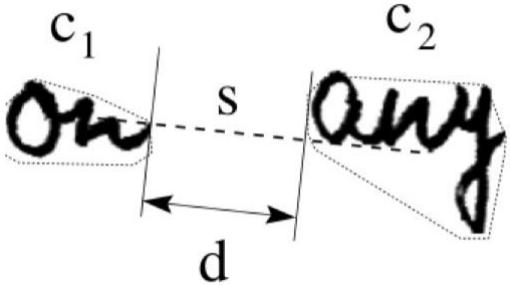


Рисунок 1.3 – Розрахунок відстані між центрами сегментованих фрагментів рукописного тексту

Наступний підхід також ґрунтується на розрахунку відстаней. За допомогою методу опорних векторів знаходиться площина, яка поділяє два різні типи даних (символи), а потім за допомогою порогового значення текст розбивається на слова не рисунку 1.4 опорні вектори відмічено кружочками.

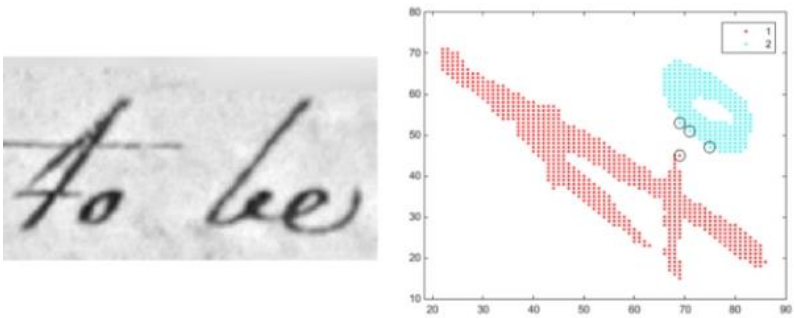
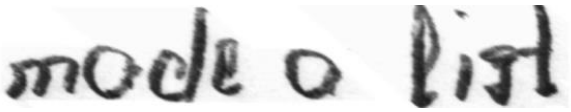


Рисунок 1.4 – Використання методу опорних векторів

Чим менше рядки у рукописному тексті схожі на прямі, тим гірше працюватиме алгоритм сегментації на рядки. Елементарна сегментація на слова працює за принципом, що відстань між словами більша, ніж між літерами.

Розглянемо детальніше випрямлення рукописного тексту. Кут нахилу – це кут між вертикаллю та напрямом аркушу. Випрямлення здійснюється на такий напрямок, щоб максимально знизити цей кут (рис. 1.5).

а) фрагмент тексту до випрямлення



б) фрагмент тексту після випрямлення

Рисунок 1.5 – Випрямлення рукописного тексту

Мовна модель допомагає алгоритму розпізнавання покращити результат за допомогою відомих мовних словосполучень. Мовна модель може пророкувати наступне слово на підставі попередніх і пропонувати варіанти з різним ступенем ймовірності. Таку модель можна тренувати на підставі великої кількості тексту з розрахунком повторень одного слова за іншими. Звичайно, немає сильної впевненості, що мовна модель зможе у всіх випадках передбачати наступне слово для будь-якого тексту, тому модель тренують на тексті тієї ж тематики, що й розпізнається. Використання мовної моделі є опціональним.

Класифікація дозволяє розпізнати символ із зображення та перенести його у формат, який потім можна використовувати у електронному документі і використовувати для вирішення поставлених задач.

В цілому розпізнавання рукописних символів включає:

1) вилучення окремих символів та їх розпізнавання – аналіз символів за основними характеристиками незалежно від масштабу, геометричних спотворень та розривів, форми, розміру, нахилу, текстури, фону;

2) уточнення результатів розпізнавання на основі інформації про тип символу та результати попередніх кадрів.

Результатом є інформація про розпізнаний символ.

Науковці проводять чисельні дослідження, для вирішення задачі розпізнавання рукописних символів, застосовуючи різні підходи та алгоритми: шаблонні, ознакові та нейромережеві.

Шаблонні алгоритми порівнюють кожен символ із наявними шаблонами. Символом, який більше всього підходить, є той, який має найменшу кількість точок, відмінних від зображення, що розпізнається. Стосовно сегментації символів можна використовувати підхід, заснований на підстановці під готовий шаблон розташування символів на зображенні. Кожен із шаблонів відповідає певному стандарту розташування символів. Зображення, які отриманні у результаті, дають можливість використовувати їх для подальшого розпізнавання. Цей дає можливість полегшити аналіз зображень. Однак недоліком є необхідність наявності достатньої кількості шаблонів, що є складним при розпізнаванні рукописних документів.

Ознакові алгоритми розглядають зображення як вектор ознак, а розпізнавання полягає у порівнянні його з набором еталонних векторів тієї ж розмірності. Прийняття рішення про схожість зображення з певним символом приймається на основі математичних рішень у рамках детерміністичних та ймовірнісних підходів. Створення вектора відбувається під час аналізу зображення, а процес носить назву вилучення ознак. Еталонні вектори для символів отримують аналогічною обробкою символів навчаючої множини. Недоліком такого підходу є нестійкість до різних дефектів зображення, шумів та втрата основної інформації на етапі вилучення ознак із зображення.

Для розпізнавання символів широко застосовують нейронні мережі прямого поширення, рекурентні нейронні мережі, згорткові нейронні мережі. Нейронні мережі підходять до розпізнавання зображень, використовуючи велику кількість рукописних чисел, які використовують для оптимального налаштування її параметрів по розпізнаванню – навчання нейронної мережі [5]. Навчена нейронна

мережа використовується для розпізнавання нових рукописних чисел у тій предметній сфері, де є така потреба.

Існує багато нейромережевих моделей для розпізнавання текстів, однак завжди у якості базової архітектури використовують згорткові нейронні мережі, а також функціонал збереження та накопичення результатів розпізнавання і рекурентні мережі для розпізнавання [6].

Вхідними даними для нейромережевих методів розпізнавання є зображення рядків та слів, вихідними – символи, які ідуть у порядку їх розпізнавання, що формують машинний текст. Приклад такої моделі зображено на рисунку 1.6, на останньому етапі використовують шар CTC для збереження послідовності символів, які вводяться. Основними недоліками нейромережевих методів є складність підбору навчаючої множини даних та вимоги до вертикального положення символів, що складно забезпечити при розпізнаванні саме рукописних текстів.

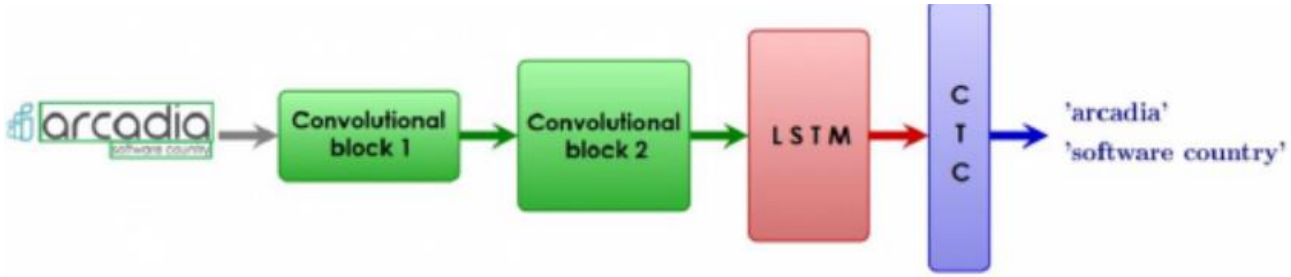


Рисунок 1.6 – Архітектура нейромережевого класифікатора

Найефективнішими у розпізнаванні рукописних символів є моделі згорткових нейронних мереж (англ. Convolutional Neural Network, CNN), які досягають високої точності класифікації й є найбільш прийнятним інструментом для розпізнавання зображень [7].

Одним із найважливіших етапів у процесі оптичного розпізнавання символів є сегментація рукописних текстів на символи, зокрема при оптичному розпізнаванні зображень документів. Важливість сегментації обумовлена тією обставиною, що в основі більшості сучасних систем оптичного розпізнавання

тексту лежать класифікатори (у тому числі – нейромережевих) окремих символів, а не слів або фрагментів тексту. У таких системах помилки неправильного проставлення розрізів між символами зазвичай є причиною левової частки помилок кінцевого розпізнавання.

Для здійснення сегментації також застосовують згорткові нейронні мережі, часто поєднуючи їх з рекурентними нейромережами для збільшення точності навчання.

В цілому розпізнавання рукописного тексту нерозривно пов’язане з серйозними технічними труднощами через величезну варіативність почерків. На результат можуть вплинути вік автора, провідна рука, рідна країна, навіть поверхня, на якій він пише, — і це без урахування впливу мови та алфавіту.

Почерк зі злитим написанням ще більше ускладнює завдання сегментації та розпізнавання окремих символів. Характерна їм відсутність чіткої структури робить автоматичний аналіз вмісту ще більш трудомістким. Не спрощує його наявність об’єктів з інших категорій: математичних висловів, графіків і таблиць. Максимальна точність розпізнавання рукописного тексту на англійській мові, результати якого вдалося знайти з відкритих джерел, досягає величини від 55 до 75%.

Важливим є також фактор часу: програмний застосунок для розпізнавання рукописного тексту має аналізувати дані, що вводяться користувачем, в режимі реального часу, в міру написання. Якщо користувач вносить правки, наприклад, закреслює слово, щоб видалити його, вставляє пробіл або переміщає абзац, то модуль розпізнавання повинен відреагувати своєчасно.

Крім того, від рішення з розпізнавання рукописного тексту потрібно вміння аналізувати як друковані символи, так і рукописні штрихи, щоб користувачі могли імпортувати надрукований текст з веб-сторінок або з інших програм, а потім анотувати його від руки, якщо потрібно. Надійний модуль розпізнавання повинен обробляти такі складні взаємодії з високою точністю, безпомилково

розрізняючи жести редагування, додавання розділових знаків та написання нових символів і слів [8].

Таким чином, задача розпізнання рукописного тексту на сьогоднішній день не є вирішеною у повному обсязі. Також є складність з датасет, які б містили достатню кількість зразків рукописного тексту на кирилиці. Системи розпізнавання, створені на основі згорткових нейронних мереж, добре справляються із розпізнаванням окремих рукописних символів. Проте точність розпізнавання документів, які містять рукописний текст, не є високою.

1.2 Застосування згорткових нейронних мереж для детектування об’єктів на зображенні

Багато задач, пов’язаних з детектуванням об’єктів на зображенні, вирішуються з допомогою згорткових нейронних мереж – ЗНМ (Convolutional Neural Networks). Завдяки своїй будові вони добре витягують ознаки з зображення та використовуються в задачах класифікації, розпізнавання, сегментації та багатьох інших [9].

Штучні нейронні мережі, створені за аналогією з людським мозком, здатні навчатися та аналізувати великі та складні набори даних, які за допомогою більш лінійних алгоритмів обробити вкрай складно. Незважаючи на наявність великої різноманітності варіацій нейронних мереж, у них наявні спільні риси через те, що вони складаються з великої кількості зв’язаних між собою однотипних елементів — нейронів, які імітують нейрони головного мозку. Такі системи у розпізнаванні зображень можуть навчатися ідентифікувати зображення, які містять певні об’єкти, аналізуючи приклади зображень, мічені як «об’єкт» і «не об’єкт», і використовуючи результати для ідентифікування об’єктів в інших зображеннях.

Базовим елементом будь-якої нейронної мережі є штучний нейрон [10]. Штучний нейрон є елементом штучних нейронних мереж, який моделює деякі функції біологічного нейрону та складається з: 1) синапсів: одно направлених

вхідних зв’язків, з’єднаних з виходами інших нейронів; 2) ядра нейрону, яке здійснює обробку вхідних сигналів; 3) аксона, що пов'язує нейрон з синапсами нейронів наступного шару шляхом передачі вихідного сигналу (в точку розгалуження).

Нейронна мережа є сукупністю нейронів, які організовані шарами. Нейрони кожного шару пов’язані з нейронами попереднього й наступного шарів. Головна функція штучного нейрону – формувати вихідний сигнал в залежності від сигналів, які поступають на його входи. Кожен синапс має вагу, яка визначає міру впливу вхідного сигналу нейрона впливає на його стан [11]. Стан нейрону можна визначити порахувавши зважену суму його входів, а значення аксона – виходу нейрону, визначається як функція його стану – активаційна функція.

У класичній багатошарової нейронної мережі міжшарові нейрони повністю взаємопов’язані, і зображення представляється у вигляді n-мірного вектора, який не враховує ні двовимірну локальну організацію пікселів, ні можливості деформації зображення. Надточна архітектура ЗНМ дозволяє подолати ці недоліки, в яких принципи архітектури неокогнітрона реалізовані, спрощені і доповнені алгоритмом навчання зі зворотним поширенням помилки.

Шари ЗНМ розташовують у трьох вимірах, які формують об’єм. Кожен нейрон згорткового шару застосовується тільки до локальної області попереднього шару, яка називається рецептивним полем. Рецептивні поля дозволяють виділяти елементарні візуальні ознаки (кут, край, тощо) а їх комбінація дозволяє отримувати складніші ознаки. ЗНМ використовує локальні поля рецепторів (забезпечують локальну двовимірну зв’язність нейронів), загальні ваги (забезпечують виявлення чисел, виявлених у будь-якому фрагменті зображення) і ієрархічну організацію з просторовими підвибірками (просторова підвибірка).

ЗНМ забезпечує частковий опір змінам масштабу, зсувам, вигинам, змінам кута і іншим спотворень. Архітектура ЗНМ є багаторівневою. Шари діляться на два типи: згорткові з нелінійними функціями активації (Convolutional) і

підвибірки – шари об’єднання (Subsampling), що чергуються один з одним. Для отримання вихідних значень застосовують згортки над кожним вхідним шаром. Кожний шар має набори з декількох площин, нейрони однієї площини мають однакові ваги, що надходять в усі локальні ділянки попереднього шару (як в зоровій корі людини), зображення попереднього шару «сканується» через невелике вікно і «зважується» за допомогою набору вагових коефіцієнтів, а результат буде відображатися на відповідному нейроні поточного шару.

Згорткові нейронні мережі є різновидом багатошарового перпептрона з використанням операцій згортки. У своїй основі розглядається як нейронна мережа, що використовує безліч ідентичних копій одного і того ж нейрона [12]. Це дозволяє мережі мати обмежену кількість параметрів при обчисленні великих моделей. Цей прийом з декількома копіями одного і того ж нейрона має близьку аналогію з абстракцією функцій. При програмуванні функція пишеться один раз і потім повторно використовується, не вимагаючи писати той самий код безліч разів у різних місцях, що прискорює виконання програми та зменшує кількість помилок. Аналогічно CNN, одного разу навчивши нейрон, використовує його в багатьох місцях, що полегшує навчання моделі та мінімізує помилки.

CNN має спеціальну архітектуру, яка дозволяє їй максимально ефективно розпізнавати образи. Сама ідея CNN ґрунтується на чергуванні згорткових та субдискретизуючих шарів пулінгу (pooling), а структура є односпрямованою. Сутність згортки полягає у тому, що кожен фрагмент зображення множиться на матрицю – ядро згортки поелементно, а результат сумується і записується у аналогічну позицію вихідного зображення – формується карта ознак [13].

Операція пулінгу дозволяє суттєво зменшити об’єм зображення. Якщо на попередній операції згортки вже були виявлені деякі ознаки, то для подальшої обробки настільки докладне зображення вже не потрібне, і воно ущільнюється до менш детального. Виконується зменшення розмірності сформованих карт ознак. Також кінцевий результат залежить від датасета – набору зображень для моделі для кожного символу різного почерку (рис. 1.7).

Архітектура згорткових шарів забезпечує інваріантність розпізнавання щодо зсуву об'єкта, поступово укрупнюючи «вікно», на яке «дивиться» згортка, виявляючи дедалі більші структури та патерни в зображенні.

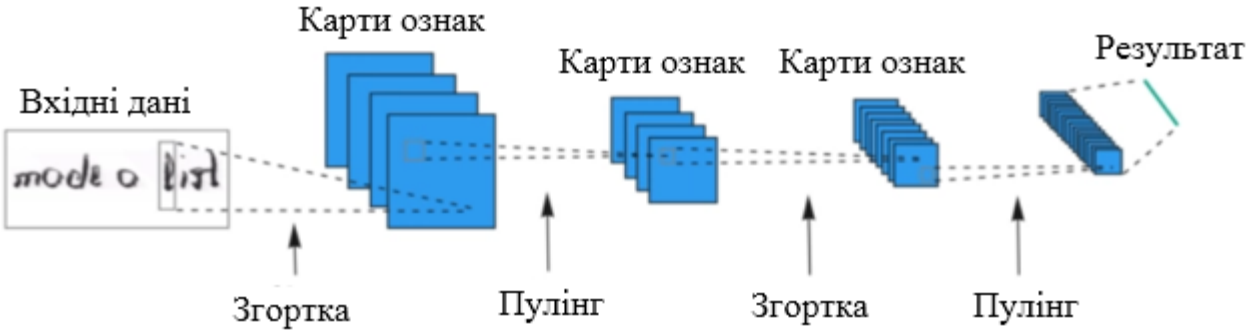


Рисунок 1.7 – Згорткова мережа для розпізнавання рукописних символів

Операцію згортки можна надати наступним алгоритмом.

1. Ковзне вікно, зване фільтром, з розміром (n, n) рухається за вхідною ознакою. Кількість рухів визначається заданою кількістю фільтрів.
2. Кожен отриманий шаблон має форму (n, n, d), де d – глибина вхідної ознаки.
3. Кожен шаблон множиться на своє ядро згортки, у результаті формується вихідна карта ознак. Отримана вихідна карта ознак має форму (h, w, N), де h і w - довжина і ширина, отримані в результаті відсікання, а N - кількість фільтрів.

Кількість фільтрів – є гіперпараметр, тому обирається самостійно. Зазвичай його підбирають як рівень двійки зі збільшенням кількості фільтрів у міру збільшення глибини архітектури. А ядра згортки є параметрами, які налаштовуютсья під час навчання.

Параметри шару згортки складаються з набору фільтрів для навчання, кожен з яких проходить (ковзає) по ширині і висоті вхідних даних. Мережа навчає фільтри, які активуються при виявленні певної візуальної особливості. Кожен

фільтр формує окрему двомірну карту активації – карту ознак. Карти ознак формують вихідний об’єм (рис. 1.8).

Ядром згортки є матриця ваг невеликого розміру, що проходить по усьому поточному шарі, формуючи після кожного зсуву сигнал активації для нейрона наступного шару з аналогічною позицією.

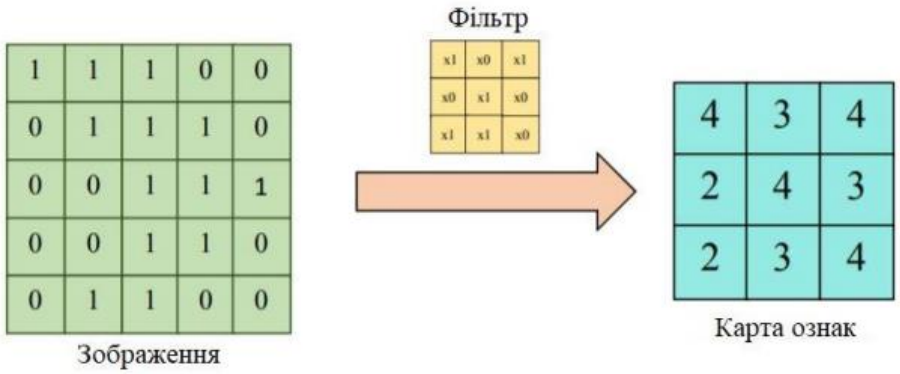


Рисунок 1.8 – Формування карти ознак. Операція згортки

Таким чином, площини називаються картами об'єктів, кожна з яких виділяє «свої» області зображення у будь-якому місці попереднього шару. Далі, після згорткового шару, вибірковий шар робить роботу зменшення масштабів площин, зменшує механізмом локального усереднення значень шару на виходах з нейронів тим самим досягає ієрархічної організації. Наступні шари виділяють найбільш поширені властивості, в меншій мірі залежать від спотворення зображення.

Розглянемо процес згортки на прикладі зображення у відтінках сірого розміром 28x28 пікселів. Глибина зображення у відтінках сірого дорівнює 1, якби це було RGB, то глибина входу дорівнювала б 3. Нехай розмір фільтра дорівнює 3x3, а їх 32 (рис. 1.9). На першому етапі сформуються 32 шаблони розміром 3x3x1, де 1 - глибина зображення. Отримані шаблони множаться на ядра згортки. Кожен перетворений у результаті множення шаблон формує вектор із довжиною, що дорівнює кількості фільтрів, тобто. 32. Усі перетворені шаблони поєднуються у вихідну карту ознак. Вона має розмір 26x26x32.

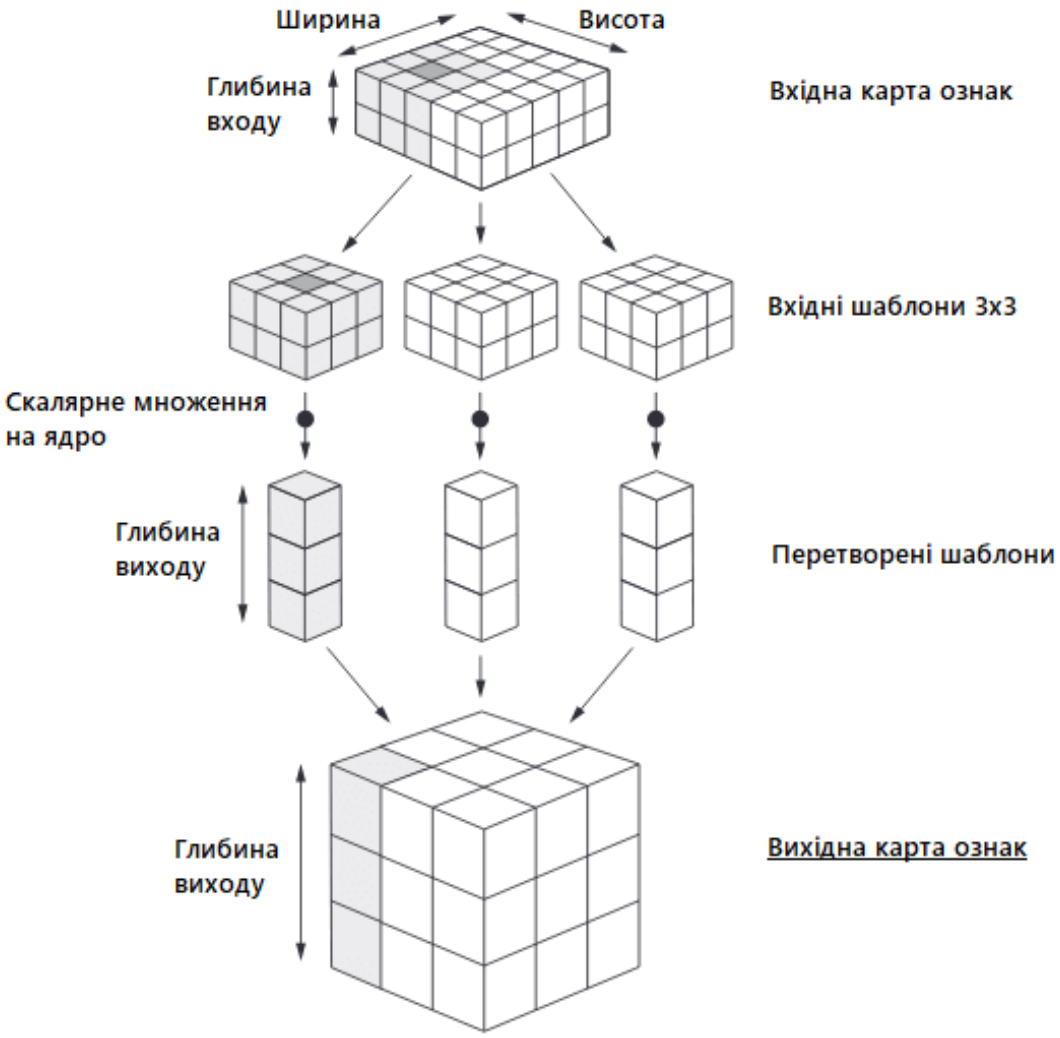


Рисунок 1.9 – Принцип дії операції згортки

Розглянемо матрицю 5x5 та фільтр 3x3. Справа в тому, що центр ковзного вікна може стати тільки в 9 клітин матриці 5x5, як це показано на малюнку нижче (рис. 1.10). Отже, після множення на ядра згортки сформується вихідна карта ознак із висотою та шириною 3x3.

Для фільтра з розміром 5x5 вихідне зображення 28x28 зменшилося до 24x24. Іноді таке обрізання можна уникнути шляхом ефекту доповнення (padding). Він полягає у додаванні рядків і стовпців так, що центр ковзного вікна можна помістити в кожну клітку. Для фільтра 3x3 додаються рядки зверху та знизу та стовпці ліворуч та праворуч. Для фільтра 5x5 додаються по 2 рядки знизу та зверху та 2 стовпці ліворуч та праворуч.

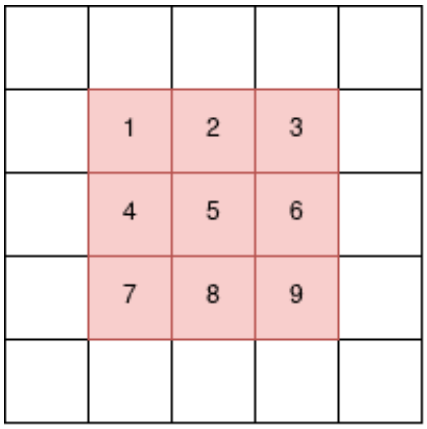


Рисунок 1.10 – Положення для фільтра 3x3

Суть шару Pooling полягає у зменшенні розмірності карти ознак. Pooling має два різновиди: max-pooling та average-pooling. Найчастіше застосовується max-pooling. Операція Pooling схожа з операцією згортки:

1) ковзне вікно, зазвичай це вікно 2x2, рухається по карті ознак;
2) з обраного шаблону вибирається максимальне (max-pooling) або середнє (average-pooling) значення;
3) формується зменшена у розмірі карта ознак.

На рисунку 1.11 показано, як із матриці 4x4 виходить вихідна карта 2x2 після операції max-pooling та average-pooling.

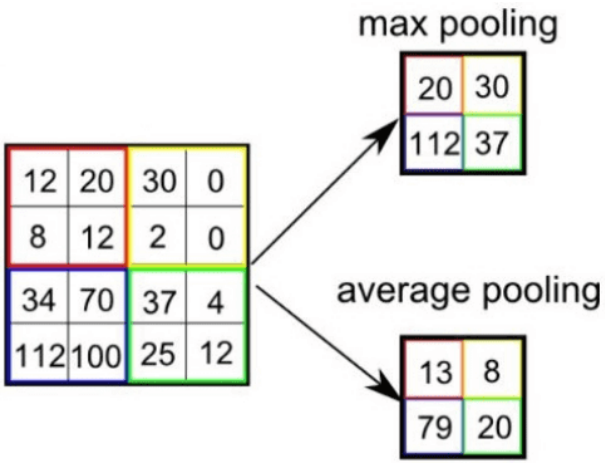


Рисунок 1.11 – Застосування Pooling до матриці 4x4 з фільтром 2x2

CNN є просторово-ієрархічною. На перших шарах вивчаються локальні шаблони, а наступні вивчають шаблони, отримані з перших шарів. Зменшують розмірність за допомогою Pooling для того, щоб підтримати ієрархічність. Архітектура згорткових нейронних мереж схожа на воронку, де все починається з великої картини з подальшим заглибленням в окремі деталі. Людський мозок налаштований аналогічно: спочатку він бачить на вулиці кішку, а потім починає розглядати її колір, плями, вуха, очі тощо. Це є основою Deep learning – навчання на уявленнях. Зменшення розмірності призводить до зменшення кількості параметрів, що налаштовуються під час навчання, тому це ще й виграш у обчислювальних ресурсах.

Глибинне навчання полягає у наборі алгоритмів, що виконують операції з моделювання високорівневих абстракцій у даних, використовуючи для цього глибинний граф з обробними шарами, які побудовані за допомогою лінійних чи нелінійних перетворень [14].

Глибинне навчання засноване на навчанні через зчитування ознак даних. Дані, наприклад, зображення можуть бути представлені різними способами, наприклад, як вектор значень яскравості пікселів, чи більш абстрактним способом - через множину кромок або областей певної форми тощо. Серед представлень є ті, які є кращими в області спрощення задачі навчання, наприклад для розпізнавання обличь чи виразів обличчя. Однією з переваг алгоритмів глибинного навчання є заміна ознак ручної роботи ефективними алгоритмами автоматичного чи напівавтоматичного навчання через ознаки, а також ієрархічне виділення ознак.

Роботи направленні на покращення у цій області намагаються знайти кращі представлення та створити моделі навчання для цих представлень з великомасштабними неміченими даними. Різні типи архітектур для алгоритмів глибинного навчання а також рекурентні нейронні мережі, які застосовуються у областях комп'ютерного зору, автоматичного розпізнавання мови, обробки

природної мови, а також у розпізнаванні звуків та біоінформатики, де вони мають змогу представляти передові результати у різноманітних задачах.

Алгоритми глибинного навчання мають за основу розподілені представлення. Припущення, яке є у основі розподілених представлень – спостережувані дані породжуються взаємодією деяких факторів, організованих на одному рівні. Глибинне навчання висуває припущення, що ці рівні факторів є відповідними різним за рівнями абстракціями або побудовам [15]. З метою забезпечення різних ступенів абстракції можуть бути застосовані змінна кількість та розмір шарів. Алгоритми глибинного навчання використовують цю ідею ієрархічних пояснювальних факторів, у якому з понять нижчого рівня відбувається процес навчання понять вищого рівня. Такі типи архітектур частіше за все побудовані з використанням жадібного пошарового методу. Алгоритми глибинного навчання дозволяють розплутувати ці абстракції та відокремлювати ознаки, які є корисними для навчання.

Для керованого навчання методи алгоритмів глибинного навчання мають уникати конструювання ознак, перетворюючи дані у формат компактних проміжних представлень, наприклад, головних компонентів, й виводять структури, які розбиті пошарово, що дозволяє усунути надмірність у представленнях [16, 17].

ЗНМ мають суттєві переваги у швидкості та надійності класифікації у порівняні з іншими архітектурами. Корисна функція CNN полягає в тому, що особливості, сформовані на виході верхнього шару структури, можуть бути використані для класифікації методом найближчого сусіда (наприклад, при обчисленні евклідової відстані), і CNN може успішно використовувати характеристики які були відсутні в навчальних наборах даних. CNN має характеристики високої швидкості навчання та високої продуктивності.

Згорткові нейронні мережі та їх модифікації вважаються найкращими алгоритмами пошуку об’єктів у зображеннях з точки зору точності та швидкості. Відомими моделями згорткових нейронних мереж є LeNet, AlexNet, ZfNet, VGG,

GoogleLeNet, R-CNN, Fast R-CNN, YOLO. Згорткові нейронні мережі відносять до глибинний нейронних мереж (англ. Deep Neural Network, DNN), які перетворюють вхідні дані у вихідні, ієрархічно виділяючи та агрегуючи ознаки шляхом підвищення рівня абстракції даних у напрямку від входів до виходів мережі.

Навчання штучних нейронних мереж розглядають як налаштування архітектури і ваг зв'язків між нейронами для виконання поставлених завдань. Існуючі методи навчання нейронних мереж можна розділити на дві великі групи методів: детерміновані та стохастичні методи навчання. До детермінованих відносять методи, які базуються на інтерактивній корекції параметрів мережі. Стохастичні методи навчання змінюють параметри мережі випадковим чином, зберігаючи ті з них, які призвели до поліпшення роботи мережі.

Основним детермінованим методом навчання є метод зворотнього поширення помилки. Основна його ідея полягає у поширенні сигналів помилки після її обчислення на виході мережі у напрямку, зворотному прямому поширенню сигналів. При зворотному проході синаптичні ваги налаштовуються з метою мінімізації помилки. Відбувається рух у багатомірному просторі ваг до мінімуму помилки в сторону, протилежну градієнту.

Епоха у навчанні включає прямий та зворотний прохід по усім тренувальним прикладам, і може містити декілька ітерацій, оскільки кожна ітерація може охопити певну кількість прикладів – серію (batch). Процес навчання мереж описують за допомогою кривої, де на горизонтальній осі відображено кількість навчальних зразків, які отримала мережа, а на вертикальній осі – значення отриманої помилки (рис. 1.12). Крива процесу навчання є індикатором правильності роботи мережі. Значення помилки на тестовому наборі ніколи не повинно бути нижчим за значення помилки навчального набору.

На сьогоднішній день важливу роль у реалізації нейронних мереж відіграють спеціалізовані пакети, які надають уже повністю реалізовані структури та алгоритми для розробки. Найбільш популярними є такі бібліотеки глибинного

навчання, як Torch, Theano, Tensorflow, Caffe. Здійснивши порівняльний аналіз бібліотек було прийнято рішення використовувати для програмної реалізації згорткової нейронної мережі бібліотеку Tensorflow.

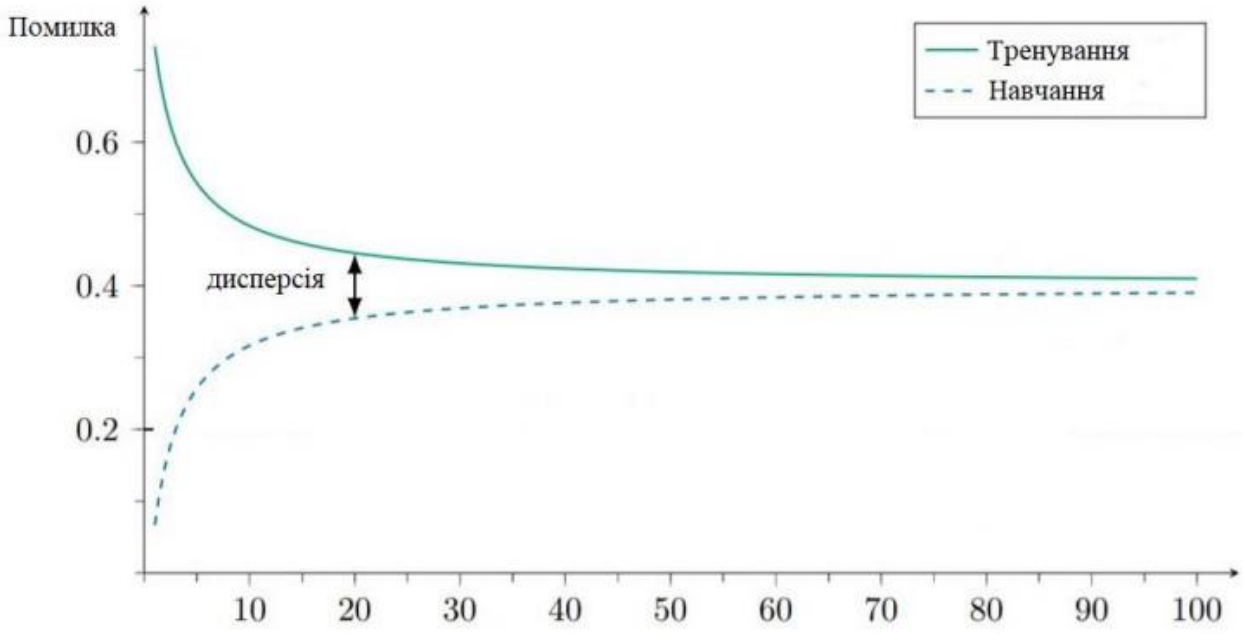


Рисунок 1.12 – Процес навчання мережі

1.3 Програмні засоби розпізнавання тексту

До програмних засобів, які застосовують для розпізнавання текстів, відносять ABBYY FineReader, що є розробкою компанії ABBYY, яка входить до числа провідних компаній із розпізнавання тексту. Даний продукт є програмою з графічним інтерфейсом користувача, де можна завантажувати документи і отримувати результат у вигляді файлу (рис. 1.12).

Також існує ABBYY Cloud OCR SDK API – хмарний сервіс, який використовує двигун ABBYY FineReader OCR. ABBYY Cloud OCR платний. ABBYY FineReader не має проблем із добре відсканованим текстом і непогано справляється з документами, які сфотографовані та, можливо, з якимись шумами та розворотами. Проте у рукописному документі він повністю не працює. Його

головна перевага – можливість вилучення таблиці. Крім осередків він витягує такі дрібні деталі як шрифти.

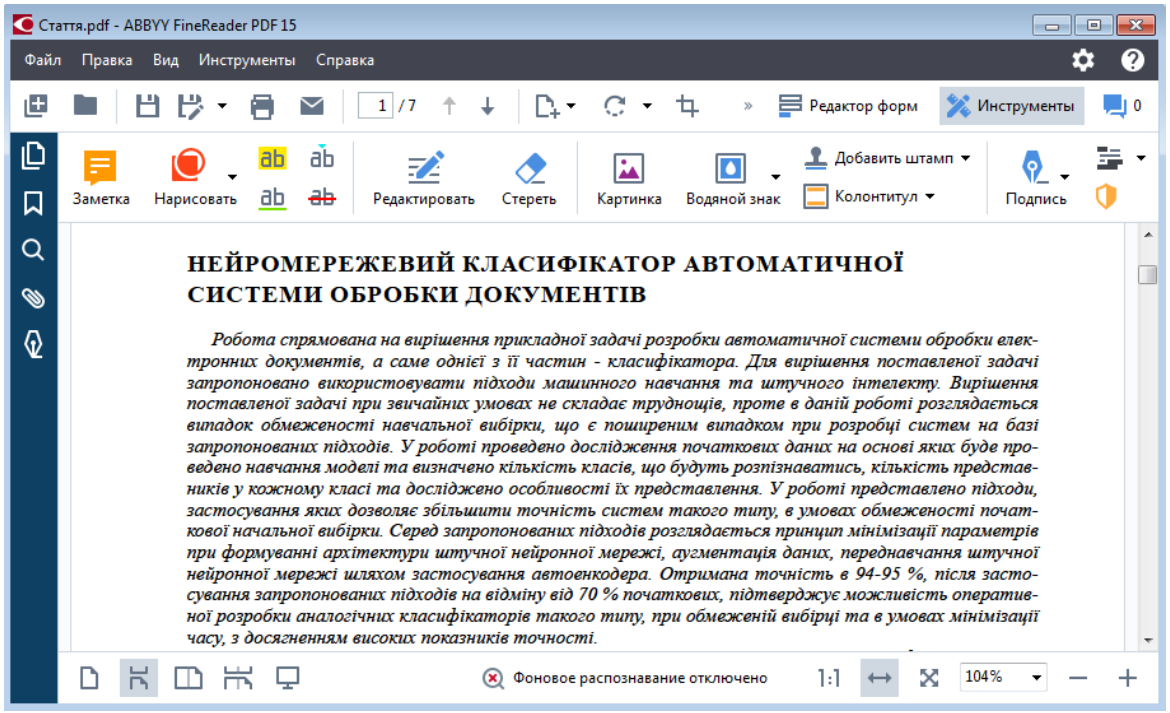


Рисунок 1.12 – Вікно програми ABBYY FineReader

Продукт Google Cloud Visio представляє собою хмарний сервіс з розпізнавання текстів із зображень (рис. 1.13). Він також, як і продукт ABBYY, є платним. Google добре справляється з відсканованим текстом і розпізнає текст у документі, знятому на камеру, як і ABBYY. Однак він набагато кращий, ніж Tesseract або ABBYY у розпізнаванні почерку.

Google Cloud Vision не дуже добре обробляє таблиці: він витягує текст, але це все. Фактично, результат роботи Cloud Vision є файлом JSON, що містить інформацію про позиції символів. На основі цієї інформації можна спробувати виявити таблиці, але ця функція не вбудована і необхідно задіяти додаткові ресурси та технології. Узагальнені відмінності проаналізованих технологій представлені у таблиці 1.1.

Рисунок 1.13 – Витягування тексту із зображення в Google Cloud Visio

Таблиця 1.1 – Аналіз програмних засобів розпізнавання символів

OCR системи	Розпізнавання відсканованого документу	Розпізнавання рукописного документу	Розпізнавання сфотографованого документу	Розпізнавання таблиць
Tesseract	Добре	Погано	Прийнятно	Погано, потрібні додаткові бібліотеки
ABBYY FineReader	Добре	Погано	Добре	Добре
Google Cloud Vision	Добре	Прийнятно, є помилки у розпізнаванні	Добре	Погано, необхідні додаткові бібліотеки

До програмних засобів розпізнавання друкованого та рукописного текстів відносять PDFelement Pro, OCR Desktop, SimpleOCR [18]. Ці програми дозволяють розпізнавати символи з pdf-документів, у тому числі рукописні, редагувати їх та конвертувати у документи інших форматів. Проте безкоштовні версії програм мають обмежений функціонал. Модуль, який дозволяє використовувати альтернативне рукописних текстів є у програмі MS Word. Засобом введення при цьому є планшети з сенсорним введенням та миша.

Однак еволюція технологій розпізнавання рукописного тексту ще не наблизилася до фінальної стадії. У багатьох системах оптичного розпізнавання рукописних та друкованих текстових документів результат, отриманий після класифікації, не вважається достатнім. Необхідно використовувати контекстну інформацію, яка дозволяє знаходити помилки та виправляти їх.

Самим розповсюдженим є використання словників. Наприклад, безкоштовна бібліотека Tesseract OCR досить добре розпізнає відсканований текст, однак при розпізнаванні рукописного тексту процент розпізнаного суттєво знижується. Є складність у розпізнаванні табличної інформації. У таких випадках необхідно самостійно обробляти вихідні дані з допомогою додаткових технологій та бібліотек. З цією метою застосовують ручну правку розпізнаного тексту, глобальні та позиційні діаграми, триграми, n-грами, словники та різні поєднання цих методів.

Здійснений аналіз дозволив установити, що на ринку програмного забезпечення повноцінних засобів, які б вирішувати задачу розпізнавання рукописних текстів, немає. Існує потреба створення систем, які б справлялися з цією задачею при цифровізації документів, що містять рукописний текст.

1.4 Постановка задачі дослідження

Провівши аналіз сучасних підходів до розпізнавання символів рукописних текстів із використанням нейронних мереж було виявлення складність задачі

розпізнавання рукописних текстів та зроблено висновок про розробку системи розпізнавання рукописних символів на згорткових нейронних мереж.

Об’єкт дослідження – процес розпізнавання зображень в автоматизованих системах обробки інформації.

Предмет дослідження – програмні засоби для автоматичного розпізнавання рукописних символів, побудовані на основі згорткових нейронних мереж.

Мета дослідження – підвищення ефективності розпізнавання рукописних символів шляхом створення системи розпізнавання на основі згорткових нейронних мереж.

Досягнення поставленої мети обумовлює необхідність вирішення наступних **завдань**:

1) розкрити теоретичні засади розпізнавання зображень, здійснити аналіз сучасних підходів до розпізнавання символів рукописних текстів та обрати нейромережеву модель для їх розпізнавання;
2) обґрунтувати вибір технологій та інструментальних засобів розробки системи розпізнавання рукописних символів;
3) розробити та здійснити програмну реалізацію системи розпізнавання рукописних символів на основі згорткових нейронних мереж.

Висновки до розділу 1

Установлено, що актуальність задачі розпізнавання рукописних символів в умовах інформатизації суспільства різко зросла. Для вирішення цієї проблеми створюють інформаційні системи, здатні розпізнавати символи за їх рукописним зображенням, отриманим із різних джерел. Однак розпізнавання зображень рукописних документів при їх цифровізації все ще не є надійним. Існує потреба у системах, які забезпечують високу точність та надійність розпізнавання у різних сферах автоматизації обробки документів.

У результаті проведеного дослідження установлено, що фундаментальними етапами розпізнавання рукописних символів є сегментація, виділення ознак і класифікація. Виявлено, що згорткові нейронні мережі відносять до глибинних нейронних мереж (англ. Deep Neural Network, DNN), які перетворюють вхідні дані у вихідні, ієрархічно виділяючи та агрегуючи ознаки шляхом підвищення рівня абстракції даних у напрямку від входів до виходів мережі. Найефективнішими є моделі згорткових нейронних мереж (англ. Convolutional Neural Network, CNN), які досягають високої точності класифікації й є найбільш прийнятним інструментом для розпізнавання зображень.

2 ТЕХНОЛОГІЇ ТА ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ СИСТЕМИ РОЗПІЗНАВАННЯ РУКОПИСНИХ СИМВОЛІВ

2.1 Платформи та фреймворки із вбудованими засобами побудови нейромережевих моделей

Для побудови нейромережевих моделей з метою розпізнавання рукописних символів можна скористатися програмними засобами та фреймворками, які мають вбудовані засоби їх створення, налаштування та навчання. До таких засобів можна віднести систему MatLab, Microsoft Azure Machine Learning Studio, фрейсворк машинного навчання ML.NET, платформа Google Cloud AI. Проаналізуємо їх можливості.

MatLab є платформою чисельного програмування, яка дозволяє користувачам аналізувати дані, розробляти математичні алгоритми та створювати моделі. Побудований з використанням мови програмування на основі матриць, він дозволяє користувачам аналізувати великі набори даних, а також ретельно розробляти та тестувати моделі [19]. Розроблений для інженерів і вчених, він часто застосовується для різних цілей, включаючи машинне навчання та глибоке навчання, обробку зображень, обчислювання фінансів, аналітику Інтернету речей та інше.

MatLab є корисним для фінансових організацій, оскільки він розробляє моделі тестування ризиків та стрес-тестування, які є високоякісними, документованими, прозорими та відтворюваними. MatLab також корисний у галузі робототехніки, оскільки він може розробляти та налаштовувати алгоритми, автоматично генерувати код та моделювати реальні системи на єдиній інтегрованій платформі.

Програмне забезпечення дозволяє користувачам підключатися до своїх роботів і управляти ними за допомогою розроблених алгоритмів. Існують також

інструменти для підключення до операційної системи роботів та створення апаратно-незалежних алгоритмів.

Переваги MatLab: форум онлайн-спільноти - чудовий ресурс підтримки, інтуїтивно зрозуміла обробка та аналіз даних, багатий набір функцій. Недоліки MatLab: крута крива навчання, програмне забезпечення потребує великих ресурсів і займає занадто багато місця для зберігання та пам’яті, мова програмування не є відкритим вихідним кодом, велика вартість ліцензії.

Microsoft Azure Machine Learning Studio – це програмне забезпечення для машинного навчання з можливістю перетягування та спільної роботи, яке допомагає розробникам та фахівцям з обробки даних розробляти, тестувати та виконувати рішення для прогнозної аналітики за лічені хвилини.

Azure Machine проста, але потужна програма, яка публікує моделі у вигляді веб-сервісів, які можуть використовуватися інструментами бізнес-аналітики та програмами, такими як Microsoft Excel. Він пропонує найкращі у своєму класі алгоритми та візуальне середовище розробки для нетехнічних фахівців, які дозволяють швидко реалізовувати ідеї без написання коду. Одна з найпривабливіших функцій – інтерактивний робочий простір.

Переваги Microsoft Azure Machine: зручність у використанні, програма заснована на хмарі, тому не покладається на доступні ресурси комп'ютера, сприятлива вартість продукту, підтримка мов R і Python для побудови коду користувача, чудова взаємодія із середовищем Microsoft Office.

Недоліки Microsoft Azure Machine: порівняно з іншими програмами машинного навчання є досить обмеженою, немає простого способу підключення до Tableau, завжди вимагає стабільного підключення до Інтернету, оскільки працює у хмарі.

ML.NET – це безкоштовний, кросплатформений та відкритий фреймворк машинного навчання, призначений для використання можливостей машинного навчання (ML) у додатках .NET. ML.NET дозволяє навчати, створювати та постачати користувацькі моделі машинного навчання, що використовують C# або

F# для таких сценаріїв, як sentiment analysis, issue classification, forecasting, recommendations та інших.

Переваги ML.NET: можливість розробки у .NET, проста інтеграція з Azure. Недоліки ML.NET: розрахунки підтримуються тільки на GPU.

Google Cloud AI – платформа, яка включає будівельні блоки AI, прискорювачі та інші рішення AI . Рішення AI націлені на бізнес-менеджерів, а не на спеціалістів з обробки даних, і є нещодавнім доповненням до платформи. Бізнес-менеджери можуть використовувати будівельні блоки ШІ без глибоких знань програмування або науки про дані. Але навіть досвідчені фахівці за даними використовують їх із практичних міркувань, особливо коли їм потрібно працювати без великого навчання моделі. Ці будівельні блоки є простими у використанні компонентами, які ви можете додати в будь-який з ваших додатків, щоб додати мову, вид, діалог і структуровані дані. Деякі з будівельних блоків є попередньо навчені нейронні мережі.

Переваги Google Cloud AI: легко налаштовувати, добре інтегрується з Google BigQuery та Google PubSub. Недоліки: деякі із вбудованих модулів AI не мають актуальної документації, налаштування існуючих модулів та бібліотек складне та потребує складного навчання, підтримка Python та інших мов програмування може бути покращена.

2.2 Мова програмування Python, бібіліотеки NumPy, Matplotlib

Python — це високорівнева інтерпретована мова програмування загального призначення. Мова орієнтована на підвищення продуктивності розробника і читання коду (рис 2.1). Python підтримує кілька парадигм програмування: структурне, об'єктно-орієнтоване, функціональне, імперативне та аспектно-орієнтоване. У мові присутня динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопотокових обчислень та зручні високорівневі структури даних [20].

Програмний код на Python організовується у функції та класи, які можуть об'єднуватись у модулі, а вони у свою чергу можуть бути об'єднані у пакети. Може бути скомпільований в байт-код Java та в MSIL (в рамках платформи .NET).

Рисунок 2.1 – Логотип мови програмування Python

Переваги мови програмування Python визначені наступними її особливостями.

1. Гнучкість — основна перевага мови, оскільки завдяки своїй гнучкості мова набула популярності серед багатьох розробників.
2. Розширюваність - один із слоганів мови звучить як - Just Import! — що повністю пояснює, наскільки мову розширюють та було розширено за останні роки. Існують бібліотеки та фреймворки під будь-який тип завдань та потреб [21]. Також величезним плюсом є те, що ми можемо використовувати C-код з Python.
3. Простота синтаксису. Код чистий і зрозумілий без зайвих дужок та виразів.
4. Інтерпретованість. Інтерпретатор Python існує для всіх популярних платформ і за замовчуванням входить до більшості дистрибутивів Linux, а значить, є на більшості серверів «з коробки».
5. PEP — єдиний стандарт для написання коду, що дає змогу підтримувати код і читати навіть при переході від одного програміста до іншого.
6. Open Source — код інтерпретатора Python є відкритим, що дозволяє будь-кому, хто зацікавлений у розвитку мови взяти участь у її розробці та покращити її. Якщо дивитися деталі релізу однієї з версій мови, можна помітити, що великі частини нового функціоналу реалізовані сторонніми розробниками.
7. Ком'юніті. Навколо Python утворилося досить дружнє і приємне ком'юніті, яке готове прийти на допомогу будь-якому початківцю або вмілому розробнику і розібратися в його проблемі.

Недоліки є наступними.

1. Продуктивність. Більшість розробників, та й сам автор мови, сходяться на думці, що Python не настільки спритний, наскільки хотілося б. Це пов'язано з тим, що Python інтерпретується. Але навіть у порівнянні з іншими мовами, що інтерпретуються, помітно, що Python програє у продуктивності. Але це легко можна нівелювати за допомогою C реалізацій тієї чи іншої проблемної ділянки коду. На сьогоднішній день це не дуже помітно.

2. Синтаксис – як плюс так і мінус, тому що якщо при переході з іншої мови програмування, синтаксис буде незвичним і трохи дивним.

3. Динамічна типізація – через динамічну типізацію Python споживає більше ресурсів, ніж міг би, але це часто компенсується внутрішнім кешуванням.

4. Global Interpriter Lock. На даний момент це є основною проблемою продуктивності Python, а також цим обумовлена погана реалізація багатопоточності. Код GIL не змінювався з першої версії мови. Це явно свідчить про те, що він застарів.

Переваги мови зробили її популярною і затребуваною на даний момент. Python часто використовують для роботи з ШІ та машинним навчанням [22]. Численні фреймворки та бібліотеки Python допомагають суттєво зменшити кількість часу, необхідного для розробки програм.

Бібліотека NumPy — це бібліотека Python, яка використовується для роботи з масивами. Вона також має функції для роботи в області лінійної алгебри, перетворення Фур’є та матриць. NumPy означає Numerical Python. У Python є списки, які служать для цілей масивів, але вони повільно обробляються. NumPy має на меті надати об’єкт масиву, який у 50 разів швидший за традиційні списки Python [23]. Масиви часто використовуються в Data Science, де швидкість і ресурси дуже важливі.

Бібліотека Python Matplotlib призначена для візуалізації даних. У ній можна побудувати двовимірні та тривимірні графіки. Python Matplotlib – альтернатива модуля візуалізації програми для технічних обчислень MatLab. Matplotlib має

об'єктно-орієнтований інтерфейс, тобто користувач безпосередньо взаємодіє з кожним об'єктом. За допомогою коду можна задавати будь-який елемент діаграми, у тому числі ярлики та позначки на осях.

2.3 Бібліотека TensorFlow

TensorFlow є бібліотекою програмного забезпечення з відкритим програмним кодом для задач глибинного машинного навчання, розробленою 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 [24].

TensorFlow підтримує мови програмування Python і C++ та представляє обчислення у вигляді графу потоку даних – графу обчислень, який є моделлю, що описує процес виконання обчислень. Граф складається з плейсхолдеров (tf.Placeholder), змінних (tf.Variable) і операцій. У ньому здійснюється обчислення тензорів – багатовимірних масивів, які можуть бути числом або вектором. Графи виконуються в сесіях (tf.Session), які бувають двох типів: звичайні та інтерактивні (tf.InteractiveSession). Сесія зберігає стан змінних (Variables) і черг (queues).

Складання графа обчислень і виконання операцій є двома різними процесами. У графі кожна вершина має 0 або більше входів і 0 або більше виходів, і являє собою реалізацію операції. Тензорами є ребра графа – масиви довільного розмір, тип яких вказується при побудові графа. Кожна операція має назву є абстрактним обчисленням (наприклад, операція суми). У операції можуть бути атрибути. Змінна – особливий вид операції, який повертає лічильник на постійно мінливий тензор, що дає можливість змінній не зникати після одиничного використання графа. При розв’язанні задач машинного навчання

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

TensorFlow добре працює на графічних процесорах, також чудово може споживати енергію від центральних процесорів комп'ютерів, на яких немає виділеного графічного процесора. Таким чином, розробники з обмеженими обчислювальними ресурсами, можуть використовувати програмне забезпечення для створення проектів.

TensorFlow має активне виконання, яке є функцією, яка спрощує налагодження, побудову та навчання динамічних графіків. Користувачі можуть використовувати потік управління Python у програмному API та миттєво перевіряти та налагоджувати свої графіки.

Переваги: TensorFlow можна використовувати безкоштовно, відмінна підтримка клієнтів, найкраща продуктивність. Недоліки: дуже крута крива навчання, підтримує лише мову програмування Python.

2.4 Фреймворк Keras

Keras – фрейморк для Deep Learning, є надбудовою над TensorFlow. Keras був розроблений для того, щоб максимально спростити процес створення нейронних мереж. Наголос був на розширюваності, модульності та мінімалізмі з підтримкою Python. Розробка Keras дозволила компанії Google зробити великий внесок у глибинне навчання та штучний інтелект. Пов’язано це насамперед із тим, що бібліотека містить сучасні алгоритми, які раніше були недоступні [25].

Keras має ряд позитивних факторів при використанні. Зручність користувача. Keras був розроблений для користувачів, а не для машин – він використовує спеціальні методи: пропонує узгоджений і простий API, мінімізує кількість дій користувача, необхідних для вирішення поширених завдань. У разі виникнення помилок завжди можна звернутися на підтримку зворотного зв'язку.

Модульність. Мається на увазі послідовність або граф автономних, повністю налаштованих модулів, які можуть бути підключені без додаткових обмежень. Наприклад, це можуть бути нейронні шари, функції помилки, оптимізатори, схеми ініціалізації, функції активації та схеми регулювання – всі ці модулі можна комбінувати для створення моделі.

Розширюваність. Можливість легко додавати нові класи, модулі та функції робить Keras відмінним засобом для проведення різноманітних досліджень.

Робота з Python. Всі моделі були написані мовою програмування Python, тому код завжди компактний, який легко можна прочитати [26].

В основі Keras лежать моделі, основний тип яких – послідовність, що є лінійним стек шарів. Створюється послідовність і до неї додаються шари у тому порядку, у якому потрібно виконати обчислення. Після визначення компілюється модель, що використовує базову платформу оптимізації обчислень.

Модель повинна відповідати даним – це можна зробити по одній партії даних за один раз або запустивши весь режим навчання моделі. Після завершення навчання модель можна використовувати для прогнозування нових даних. Також є ще один тип моделей – це клас Model, який використовується з функціональним API. Keras на даний час займає одне з провідних місць в роботі з штучним інтелектом і було використано при розробці системи розпізнавання рукописних цифр.

2.5 Платформа з відкритим кодом Kivy

Kivy – це фреймворк для Python з відкритим вихідним кодом і мультиплатформенністю, що дозволяє розробляти програми зі складними функціями, дружнім інтерфейсом користувача та мультисенсорними властивостями, і все це за допомогою інтуїтивно зрозумілого інструменту, орієнтованого на швидке створення прототипів із ефективним дизайном, який

допомагає мати коди для повторного використання, простий у розгортанні [27]. Встановити Kivy можна використовуючи команду pip install Kivy (рис. 2.1)

```
PS C:\Users\anhel\source\repos\Python\Recognition\Recognition> py -m pip install Kivy
Defaulting to user installation because normal site-packages is not writeable
Collecting Kivy
  Downloading Kivy-2.1.0-cp39-cp39-win_amd64.whl (4.0 MB)
     ---------------------------------------- 4.0/4.0 MB 10.6 MB/s eta 0:00:00
Collecting Kivy-Garden>=0.1.4
  Downloading Kivy_Garden-0.1.5-py3-none-any.whl (4.6 kB)
Collecting kivy-deps.angle~=0.3.2
  Downloading kivy_deps.angle-0.3.3-cp39-cp39-win_amd64.whl (4.8 MB)
     ---------------------------------------- 4.8/4.8 MB 11.3 MB/s eta 0:00:00
Collecting kivy-deps.sdl2~=0.4.5
  Downloading kivy_deps.sdl2-0.4.5-cp39-cp39-win_amd64.whl (3.1 MB)
     ---------------------------------------- 3.1/3.1 MB 12.2 MB/s eta 0:00:00
Collecting pypiwin32
  Downloading pypiwin32-223-py3-none-any.whl (1.7 kB)
Collecting pygments
  Downloading Pygments-2.14.0-py3-none-any.whl (1.1 MB)
     ---------------------------------------- 1.1/1.1 MB 11.8 MB/s eta 0:00:00
Collecting kivy-deps.glew~=0.3.1
  Downloading kivy_deps.glew-0.3.1-cp39-cp39-win_amd64.whl (123 kB)
     ---------------------------------------- 123.6/123.6 kB 7.6 MB/s eta 0:00:00
Collecting docutils
  Downloading docutils-0.19-py3-none-any.whl (570 kB)
     ---------------------------------------- 570.5/570.5 kB 12.2 MB/s eta 0:00:00
Requirement already satisfied: requests in c:\users\anhel\appdata\roaming\python\python39\site-packages (from Kivy-Garden>=0.1.4->Kivy) (2.28.1)
Collecting pywin32>=223
  Downloading pywin32-305-cp39-cp39-win_amd64.whl (12.2 MB)
```

Рисунок 2.2 – Встановлення Kivy

Kivy було розроблено з використанням Python у Keaton, він заснований на OpenGL ES 2 і підтримує велику кількість пристроїв введення, таким же чином інструмент оснащений великою бібліотекою віджетів, які допомагають додати багато функцій. Цей потужний фреймворк дозволяє генерувати базовий вихідний код, який можна використовувати в програмах для Linux, Windows, OS X, Android та iOS. Його чудова стабільність, чудова документація, широка спільнота та потужний API роблять його дуже корисним фреймворком для більшості програмістів Python [28].

Kivy є гарним вибором, оскільки він містить функції, які дозволяють слідувати галузевим стандартам і допомагають пришвидшити процес розробки програм. Однією з його найбільших особливостей є висока підтримка різних пристроїв введення та протоколів, а також можливість розробки основних програм, які потім можна перенести на різні операційні системи, що допоможе заощадити час.

2.6 Бібліотека OpenCV

OpenCV – це бібліотека з відкритим кодом, яка підтримує багато платформ, включаючи Windows, MacOs і Linux. Найчастіше її використовують для написання програм машинного навчання на Python, особливо в області комп'ютерного зору. Ця бібліотека також існує для багатьох інших мов програмування [29]. Окрім того, що бібліотека OpenCV є кросплатформною та підтримує багато мов програмування, які дозволяють використовувати програми в різних системах, вона дуже ефективна (порівняно з іншими подібними бібліотеками) з точки зору обчислень, оскільки майже всі функції та оператори в ній векторизовані.

Однією з основних функцій є виведення зображення на екран. Процес виведення зображення на екран складається з двох етапів. Спочатку ми повинні завантажити зображення, а потім вивести його на екран. Ці операції виконуються послідовно, і на кожну з них покладається окрема функція.

Щоб відобразити зображення на екрані, потрібно встановити дві речі.

1. Шлях до файлу, що містить зображення (працює як відносний, так і абсолютний шляхи).
2. Режим читання файлу (тільки читання, лише запис тощо).

Арифметика зображень включає додавання, віднімання, ділення та множення різних зображень і використовується для отримання нового зображення за допомогою арифметичних операцій над вхідними зображеннями. Арифметика зображень має багато практичних застосувань, таких як водяний знак на зображенні, змішування двох зображень, застосування різноманітних фільтрів до зображень тощо.

Наприклад згладжування зображення є надзвичайно корисною операцією, яка дуже часто використовується перед передачею зображення в модель машинного навчання для обробки. В основному це потрібно робити для фільтрації високочастотного шуму, використовуючи для цього фільтр низьких частот. Існує

багато різних фільтрів, таких як фільтр усереднення (блоковий фільтр), медіанний фільтр, фільтр хвильового типу (фільтр режиму), фільтр Гауса та багато інших.

Важливим є трансформація зображень. Це використовується в різних додатках, але окремо варто згадати про проблему доповнення даних для моделей машинного навчання. Мова йде про ситуації, коли в датасеті недостатньо даних для повноцінного навчання, таким чином, доповнюючи і модифікуючи існуючі картинки, збільшуємо початковий датасет до потрібного розміру. Це допомагає підвищити точність навченої моделі.

Список можливих перетворень дуже довгий і включає масштабування, перетворення зображень, обертання, транспонування та багато іншого.

OpenCV застосовується:

1) у робототехніці - для орієнтування робота в просторі, розпізнавання об'єктів та взаємодії з ними;
2) медичні технології — для створення точних методів діагностики, наприклад 3D-візуалізації органу при МРТ;
3) промислові технології — для автоматизованого контролю якості, зчитування етикеток, сортування продуктів тощо;
4) безпеки - для створення «розумних» камер відеоспостереження, які реагують на підозрілі дії, для зчитування та розпізнавання біометрії;
5) мобільної фотографії - для створення б’юті-фільтрів, що змінюють особу додатків;
6) на транспорті – для розробки автопілотів.

Опишемо ряд переваги OpenCV.

1. Вільний доступ. OpenCV поширюється за безкоштовною ліцензією як для освітнього, так і для комерційного використання. Бібліотека має відкритий код, який може переглядати будь-який програміст. Це дає велику гнучкість у роботі з OpenCV і дозволяє самостійно вивчати, як реалізована та чи інша функція.

2. Велика кількість алгоритмів. OpenCV містить понад 2500 інструментів і алгоритмів комп'ютерного зору та машинного навчання. Цього достатньо для вирішення складних завдань розпізнавання та обробки зображень [30].

3. Висока швидкість. Бібліотека є швидшою, ніж велике та важке математичне програмне забезпечення, таке як Matlab. Тому його можна використовувати в ситуаціях, коли потрібно швидко обробити зображення.

Враховуючи переваги та можливості бібліотеки, можна зробити висновок що OpenCV підходить для вирішення поставленої задачі.

2.7 Середовище розробки Visual Studio

Інтегроване середовище розробки Visual Studio (IDE) – це багато функціональна програма, яка використовується для розробки програмного забезпечення (ПО), яка дозволяє створювати, налагоджувати, редагувати та публікувати застосунки (рис. 2.2) Крім стандартного редактора і відладчика, які існують в більшості середовищ IDE, Visual Studio спрощує процес розробки ПО за допомогою компіляторів, графічних конструкторів, засобів виконання коду та інших функцій [31].

Visual Studio підтримує 36 мов програмування та дозволяє редактору коду підтримувати майже будь-яку мову програмування за умови існування послуги, що залежить від мови. Використовуючи плагіни доступна підтримка таких мов пррограмування, як Python, Node.js серед інших, Ruby, та M. Вбудовані мови включають JavaScript, TypeScript, XML, XSLT, HTML та CSS. Visual Basic .NET, C #, F #, C, C ++, C ++ / CLI [32].

До переваг середовища розробки Visual Studio слід віднести наступне.

1. Використовується при застосуванні технологій UWP і WPF.
2. Функціональність. За допомогою якісних плагінів розширюється функціональність програми і підключаються інші мови.

3. Підтримка платформ .NET. Visual Studio дає багато можливостей для розробки додатків під Windows, в тому числі в .NET-сегменті.

4. Хмарні сховища. Увійшовши в співтовариство Visual Studio є можливість отримати доступ до хмарного сховища, де можна розташовувати файли проектів.

5. Можливість розробки додатків для мобільних пристроїв Windows (Windows Phone).

6. Можливість розробки додатків для Microsoft Office.

7. Вбудована підтримка рефакторинга коду. Рефакторинг – це поліпшення існуючої кодової бази. У відсутності інструментів рефакторинга коду, даний захід вимагає багато часу.

8. Наявність інструментів візуального конструювання класів.

9. Наявність засобів підвищення продуктивності при розробці програмного забезпечення.

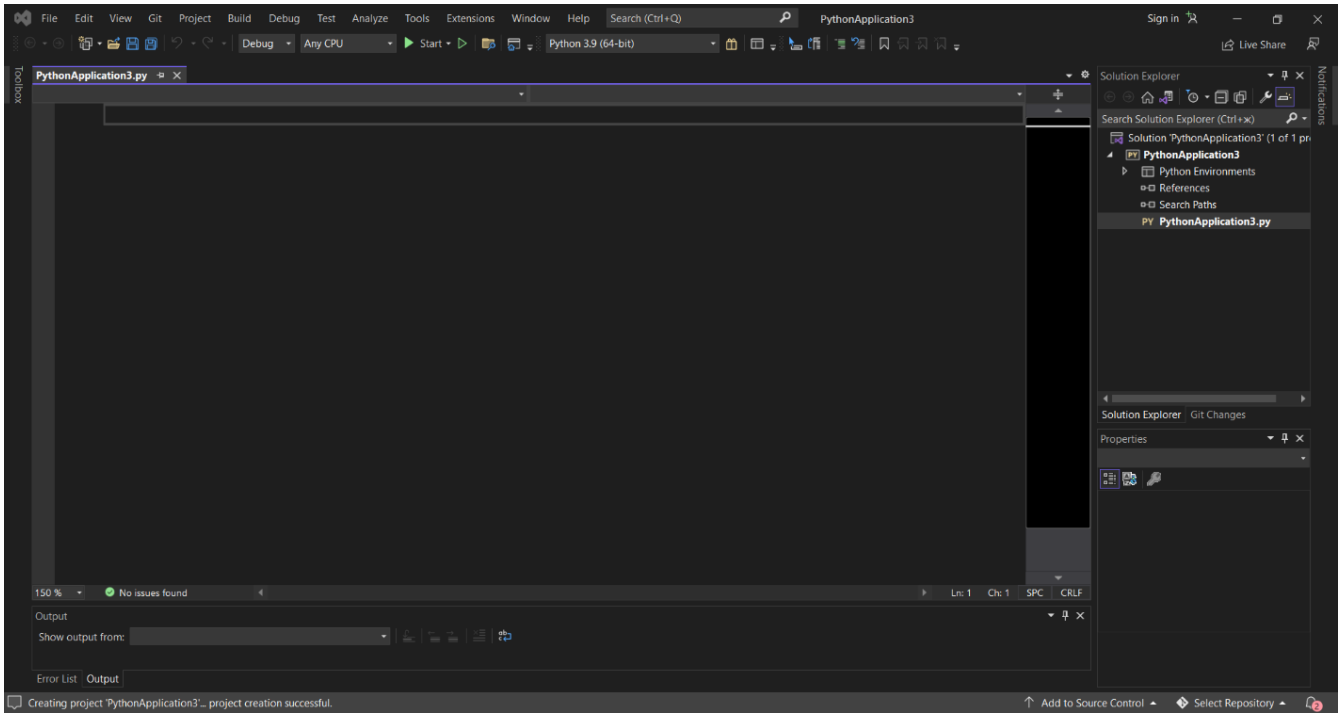


Рисунок 2.2 – Інтерфейс Visual Studio

Для розробки коду скриптів системи розпізнавання рукописних символів було вирішено обрати середовище розробки Visual Studio від Microsoft, яке підтримує мову програмування Python.

Висновки до розділу 2

Для розробки системи розпізнавання рукописних символів було вирішено використати мову програмування Python, середовище розробки Visual Studio, фреймворки TensorFlow і Keras. Бібліотека OpenCv була використана для виділення фрагментів зображення, бінаризації та сегментації. Для управляння даними та виведення результатів було використано бібліотеки NumPy, Matplotlib. Графічний інтерфейс створено за допомогою фреймворку Kivy.

3 ПРОЄКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ РОЗПІЗНАВАННЯ РУКОПИСНИХ СИМВОЛІВ

3.1 Програмна реалізація системи

Програмна реалізація системи розпізнавання рукописних символів здійснювалася на мові Python із використанням бібліотек NumPy та Matplotlib, фреймворків TensorFlow та Keras.

Бібліотека Python – це набір пов’язаних модулів. Вона містить пакети коду, які можна багаторазово використовувати в різних програмах. Для більш зручного використання модуля, створюємо псевдонім за допомогою ключового слова as. Для того щоб встановити необхідну бібліотеку використовуємо псевдонім py для Windows. Команда є наступною: py -m pip install <name module>, де <name module> - це ім’я модуля, який необхідно встановити (рис. 3.1).

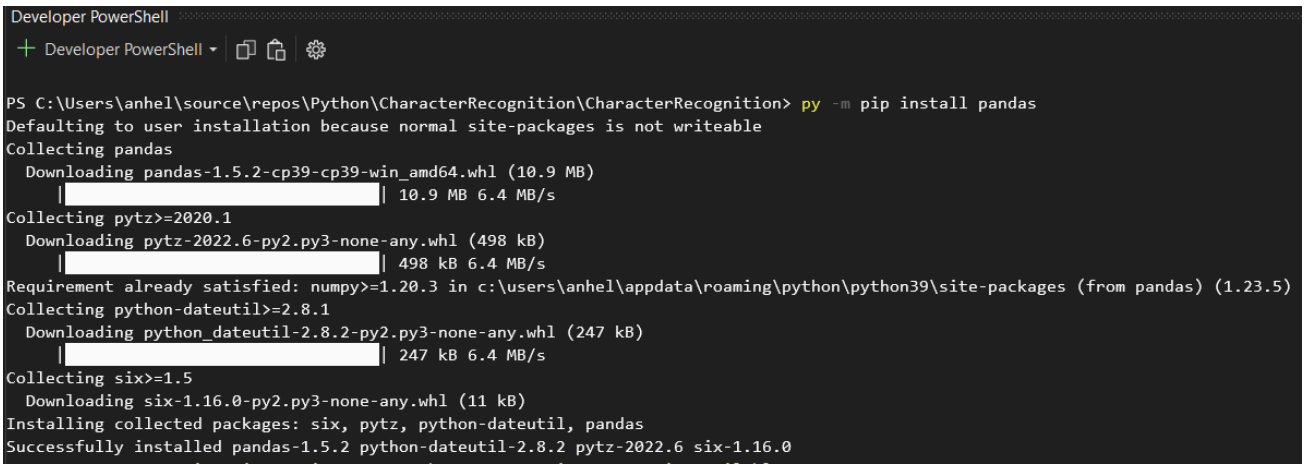


Рисунок 3.1 – Процес встановлення бібліотеки

Клас Keras ImageDataGenerator забезпечує швидкий і простий спосіб доповнити зображення. Він надає безліч різних методів збільшення, таких як стандартизація, обертання, зміщення, перевороти, зміна яскравості та багато

іншого. Однак головна перевага використання класу Keras ImageDataGenerator полягає в тому, що він розроблений для забезпечення збільшення даних у реальному часі. Це означає, що він генерує доповнені зображення дуже швидко, поки модель ще перебуває на стадії навчання.

Аргумент validation_split. Float. Частка зображень, зарезервована для перевірки (строго від 0 до 1).

Для дослідження було обрано набір даних MNIST, який містить 70 тисяч зображень розміром 28x28 пікселів із рукописними цифрами від 0 до 9 білого кольору на чорному фоні. Навчаюча множина містила 60 тисяч зображень, тестова – 10 тисяч зображень.

База даних MNIST (Modified National Institute of Standards and Technology database) є великою базою даних рукописних цифр, яка зазвичай використовується для навчання різних систем обробки зображень. База даних також широко використовується для навчання та тестування в галузі машинного навчання [33].

При створені моделі імпортуємо необхідні бібліотеки.

Sequential групує лінійний стек шарів у tf.keras.Model. Має наступні аргументи: layers список шарів для додавання до моделі. Не є обов'язковим
Name – необов'язкова назва для моделі. Метод add() додає екземпляр layer поверх стека шарів.

Метод summary() виводить підсумковий рядок мережі. Метод compile() налаштовує модель для навчання. Може приймати наступні аргументи: optimizer. Тип даних string або екземпляр оптимізатора.

Аргумент loss. Loss функція. Може бути рядком (назва функції loss) або екземпляром tf.keras.losses.Loss. Loss - це будь-яка функція, що викликається, із сигнатурою loss = fn(y_true,y_pred), де y_true — основні значення істинності, а y_pred — прогнози моделі. y_true має мати форму (batch_size, d0, .. dN) (за винятком розріджених функцій loss, таких як розріджена категорійна кросентропія, яка очікує цілих масивів форми (batch_size, d0, .. dN-1)). y_pred має

мати форму (batch_size, d0, .. dN). Функція loss має повертати float tensor. Якщо використовується спеціальний екземпляр Loss і для параметра Reduction встановити значення None, значення, що повертається, має форму (batch_size, d0, .. dN-1), тобто значення loss на вибірку або на крок часу; інакше це скаляр. Якщо модель має кілька виходів, можна використовувати різні loss для кожного виходу, шляхом передачі dictionary або list of losses.. Значення loss, яке буде мінімізовано моделлю, тоді буде сумою всіх окремих loss, якщо не вказано loss_weights.

Аргумент metrics. Список показників, які оцінює модель під час навчання та тестування. Кожен із них може бути рядком (назва вбудованої функції), функцією або екземпляром tf.keras.metrics.Metric. Зазвичай використовується metrics=['accuracy']. Функція — це будь-яка функція виклику з результатом сигнатури = fn(y_true,y_pred). Щоб вказати різні показники для різних результатів моделі з кількома виходами, також можна передати словник, наприклад metrics={'output_a':'accuracy', 'output_b':['accuracy', 'mse']}. Також є можливість передати список, щоб указати метрику, або список метрик для кожного результату, наприклад metrics=[['accuracy'], ['accuracy', 'mse']] або metrics=['accuracy', [' точність', 'mse']]. Коли передають рядки 'accuracy' або 'acc', вони на основі форм цілей і виходу моделі перетворюються на один із:

1) tf.keras.metrics.BinaryAccuracy;
2) tf.keras.metrics.CategoricalAccuracy;
3) tf.keras.metrics.SparseCategoricalAccuracy .

Ми також виконуємо подібне перетворення для рядків 'crossentropy' і 'ce'. Передані показники оцінюються без зважування вибірки. Якщо бажаємо застосувати зважування вибірки, можна вказати свої показники за допомогою аргументу weighted_metrics.

Keras Conv2D — це 2D Convolution Layer, цей шар створює ядро згортки, яке є вітром із введенням шарів, що допомагає створити tensor виходів.

Параметр kernel_size визначає розміри ядра. Загальні розміри включають 1 × 1, 3 × 3, 5 × 5 і 7 × 7, які можна передати як кортежі (1, 1), (3, 3), (5, 5) або (7, 7).

Це ціле число або кортеж/список із 2 цілих чисел, що визначає висоту та ширину вікна двовимірної згортки. Цей параметр має бути непарним цілим числом.

Параметр filters є обов'язковим параметром Conv2D є кількість фільтрів, які навчатимуться згорткові шари. Це ціле значення, яке також визначає кількість вихідних фільтрів у згортці.

Параметр padding класу Keras Conv2D може приймати одне з двох значень: «valid» або «same».

Встановлення значення «valid» параметра означає, що вхідний обсяг не доповнюється нулем, а просторові розміри можна зменшувати за допомогою природного застосування згортки.

Параметр kernel_initializer Цей параметр керує методом ініціалізації, який використовується для ініціалізації всіх значень у класі Conv2D перед фактичним навчанням моделі. Це ініціалізатор для матриці ваг ядра.

Параметр activation для класу Conv2D є допоміжним параметром, який дозволяє передати рядок, який визначає назву функції активації, яка застосується після виконання згортки.

При побудові моделі використовувалась функція активації ReLu (Rectified Linear Unit) (див. рис. 3.2).

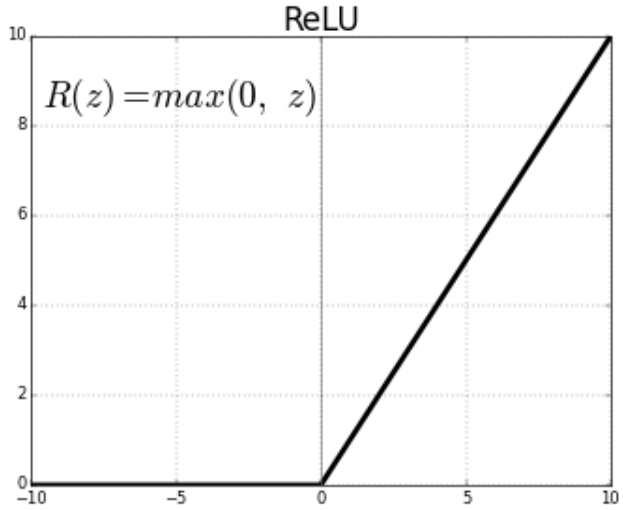


Рисунок 3.2 – Функція активації ReLu

Функція є досить простою. Вхідні значення, менші або рівні нулю, перетворюються на нуль; значення, що перевищують нуль, не змінюються. Зазвичай у згорткових шарах використовується більше одного фільтра. Коли це має місце, результати роботи кожного з фільтрів збираються вздовж деякої осі, що дає тривимірну матрицю вихідних даних [34].

MaxPool2D – операція максимального об’єднання для двовимірних просторових даних. Зменшує дискретизацію вхідного сигналу вздовж його просторових розмірів (висота та ширина), беручи максимальне значення у вхідному вікні (розміру, визначеному параметром pool_size) для кожного каналу вхідного сигналу. Вікно зсувається кроками вздовж кожного виміру.

BatchNormalization – рівень, який нормалізує свої входи. Пакетна нормалізація застосовує перетворення, яке підтримує середнє значення виходу близько до 0 і вихідне стандартне відхилення близько до 1. Важливо, що пакетна нормалізація працює по-різному під час навчання та під час висновку.

Dense реалізує операцію: output = activation(dot(input, kernel) + bias), де activation — це поелементна функція активації, яка передається як аргумент активації, kernel — це матриця ваг, створена шаром, а bias — це вектор зміщення, створений за шаром (застосовно, лише якщо use_bias має значення True). Це все атрибути Dense.

Рівень Dropout випадково встановлює одиниці введення на 0 із частотою швидкості на кожному кроці під час навчання, що допомагає запобігти переобладнанню. Вхідні дані, для яких не встановлено значення 0, масштабуються на 1/(1 - швидкість), щоб сума всіх вхідних даних не змінювалася.

Flatten використовується для конвертації вхідних даних у меншу розмірність. Створена модель згорткової нейронної мережі містить послідовність 14 шарів різних типів. Три згорткових шари Conv2D з функцією активації ReLu та розміром фільтру 3x3 застосовують операцію згортки, генеруючи результат згортки до наступного шару. Після кожного згорткового шару розміщено агрегувальний шар MaxPool2D із розміром фільтру 2x2 та шар нормалізації

BatchNormalization з метою зменшення перенавчання моделі. Потім розміщено шар Flatten, який перетворює дані 2D-матриці у вектор. Модель містить два шари Dropout, налаштовані на випадкове виключення частки нейронів у шарі для запобігання перенавчання моделі, і два повнозв'язні шари Dense. Вихідний шар має 10 нейронів і функцію активації softmax для виведення ймовірнісних прогнозів кожного класу [35].

Таким чином, створена модель згорткової нейронної мережі містить послідовність 14 шарів різних типів. Три згорткових шари Conv2D з функцією активації ReLu та розміром фільтру 3x3 застосовують операцію згортки, генеруючи результат згортки до наступного шару. Після кожного згорткового шару розміщено агрегувальний шар MaxPool2D із розміром фільтру 2x2 та шар нормалізації BatchNormalization з метою зменшення перенавчання моделі. Потім розміщено шар Flatten, який перетворює дані 2D-матриці у вектор. Модель містить два шари Dropout, налаштовані на випадкове виключення частки нейронів у шарі для запобігання перенавчання моделі, і два повнозв'язні шари Dense. Вихідний шар має 10 нейронів і функцію активації softmax для виведення ймовірнісних прогнозів кожного класу.

3.2 Навчання та тестування нейронної мережі

Для дослідження було обрано набір даних MNIST, який містить 70 тисяч зображень розміром 28x28 пікселів із рукописними цифрами від 0 до 9 білого кольору на чорному фоні. Навчаюча множина містила 60 тисяч зображень, тестова – 10 тисяч зображень. Усі зображення чисел мають розмір 28×28 пікселів. Вони мають чорний фон із білим номером. Фігуру потрібно розташувати посередині так, щоб її центр мас збігався з центром зображення. Сама вона трохи менше всієї картинки - її розмір становить 20 × 20 пікселів. База даних MNIST містить 60 000 навчальних зображень та 10 000 тестових зображень [36].

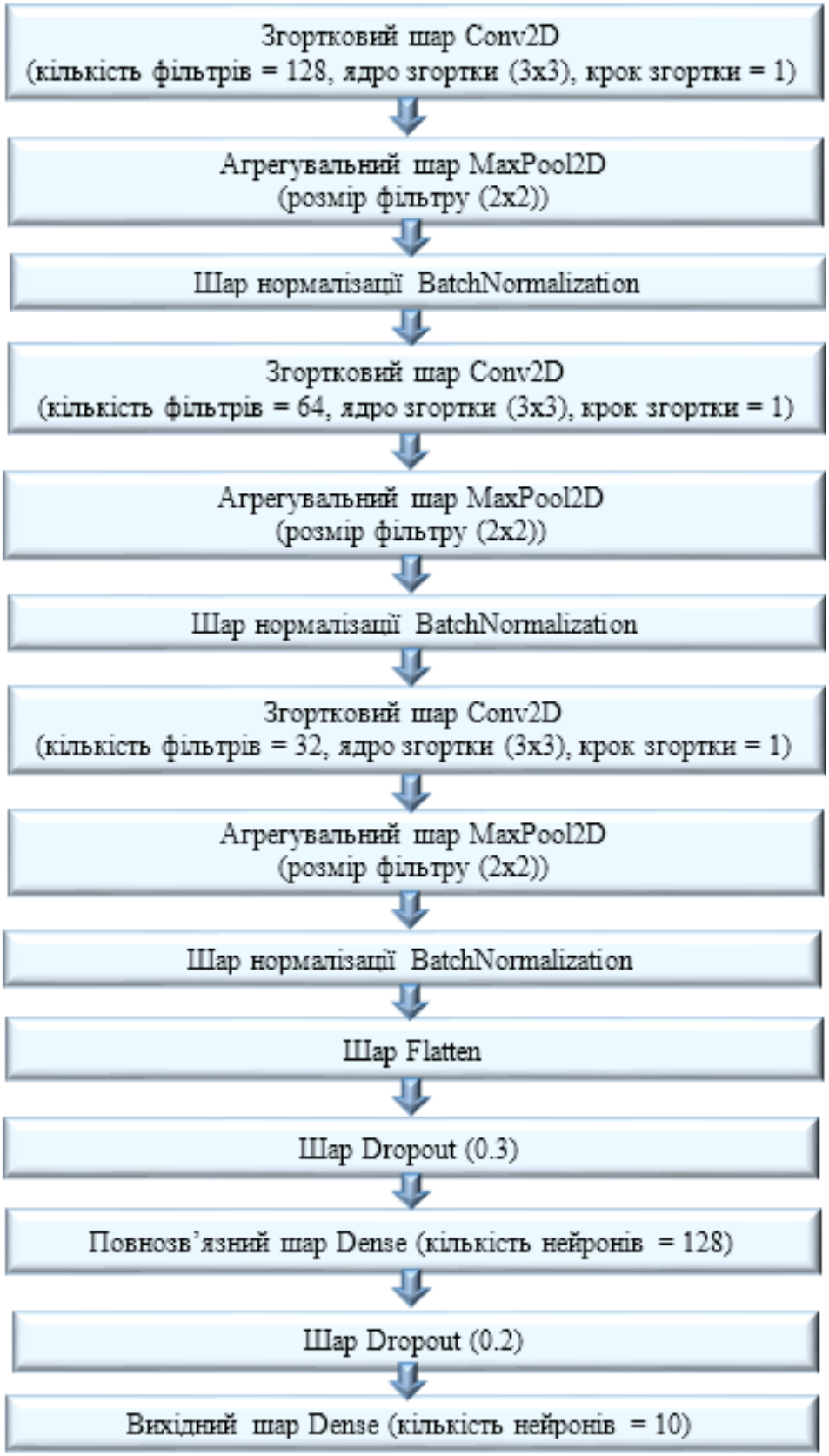


Рисунок 3.4 – Архітектура створеної згорткової нейронної мережі

Train — це 60 тисяч зображень, які варто використовувати для навчання. Програмі демонструються дані зображення, щоб вона «розуміла», як виглядають ті чи інші числа. У навчальному наборі вже є правильні результати, програма відразу отримує відповідь, що саме їй показують.

Test – становить 10 тисяч зображень тестового зразка. Вони не використовуються для навчання, але пізніше показуються нейромережі, щоб перевірити, наскільки правильно вона навчилася розпізнавати числа [37].

Завантаження MNIST відбувається за допомогою використання модуля Keras (див. рис. 3.5). Завантажуємо навчальний і тестовий набори в окремі змінні:

(x_train, y_train), (x_test, y_test) = tf.keras.datasets.mnist.load_data()

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz

    8192/11490434 [..............................] - ETA: 0s
  221184/11490434 [..............................] - ETA: 2s
  843776/11490434 [=>.............................] - ETA: 1s
 1458176/11490434 [==>............................] - ETA: 1s
 2080768/11490434 [====>..........................] - ETA: 0s
 2719744/11490434 [======>........................] - ETA: 0s
 3358720/11490434 [=======>.......................] - ETA: 0s
 3981312/11490434 [=========>.....................] - ETA: 0s
 4620288/11490434 [===========>...................] - ETA: 0s
 5423104/11490434 [=============>.................] - ETA: 0s
 6062080/11490434 [==============>................] - ETA: 0s
 6684672/11490434 [================>..............] - ETA: 0s
 7323648/11490434 [==================>............] - ETA: 0s
 7962624/11490434 [====================>..........] - ETA: 0s
 8601600/11490434 [======================>........] - ETA: 0s
 9224192/11490434 [========================>......] - ETA: 0s
 9846784/11490434 [==========================>....] - ETA: 0s
10469376/11490434 [============================>...] - ETA: 0s
11091968/11490434 [==============================>..] - ETA: 0s
11490434/11490434 [==============================] - 1s 0us/step
```

Рисунок 3.5 – Завантаження набору даних MNIST

Як би ми зберігали зображення розміром 28 x 28 пікселів, це був би формат RGB, де для кожного пікселя знадобиться 3 скаляри, тому це буде 28x28x3. Коли зображення у відтінках сірого, потрібен тільки один канал, це буде 28x28x1. -1 в основному це означає, що не потрібно обчислювати всі розміри, і обирається лише один [38].

Об’єднання даних у єдину структуру – це операція, яку виконують при завантажені даних. За допомогою неї можемо поєднувати дані з різних джерела, а також асинхронні відповіді сервера та результати паралельних та послідовних обчислень. Зменшуємо наші дані на 255. Це нормалізує значення до діапазону [0, 1], що робить навчання швидшим і легшим.

Побудуємо виведемо графічне зображення об’єктів набору (див. рис. 3.6), який використовується. Це дає уявлення про те, з якими даними буде проводитись робота.

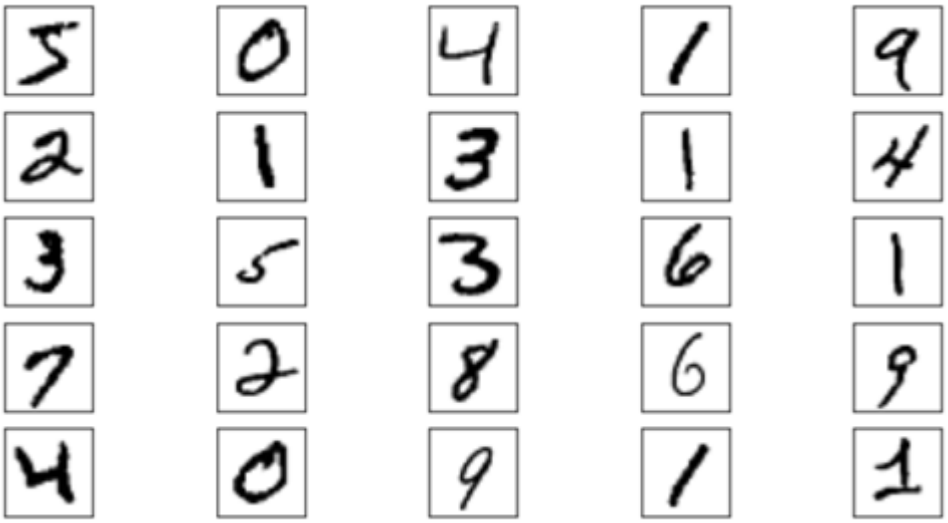


Рисунок 3.6 – Графічне зображення рукописних цифр набору даних

Використовуємо допоміжну функцію, яка виводить грона тренувальних/оціночних втрат і точності, щоб була змога легше визначити, чи був сенс у навчанні моделі для більшої кількості епох. Для створення функції використовуємо наступні функції.

Функція plot() використовується для малювання точок (маркерів) на діаграмі. За замовчуванням функція plot() малює лінію від точки до точки. Функція приймає параметри для визначення точок на діаграмі. Параметр 1 є масивом, що містить точки на осі X. Параметр 2 це масив, що містить точки на осі Y. Якщо, наприклад, потрібно побудувати лінію від (1, 3) до (8, 10), необхідно

передати два масиви [1, 8] та [3, 10] функції plot(). Вісь X – це горизонтальна вісь. Вісь Y – це вертикальна вісь.

Є можливість нанести будь-яку кількість точок, але переконавшись, що кількість точок на обох осях однакова. Якщо не вказувати точки на осі X, вони отримають значення за замовчуванням 0, 1, 2, 3 (і т. д., залежно від довжини точок у).

За допомогою Pyplot можна використовувати функції xlabel() і ylabel(), щоб установити мітку для осей X і Y. Використовуючи функцію title(), встановлюємо заголовок графіка. Параметр fontdict у xlabel(), ylabel() і title(), задає властивості шрифту для заголовка та міток. Параметр loc у title(), створений щоб позиціонувати заголовок (left, right, center). За замовчуванням значення дорівнює center.

Легенда – це область, що описує елементи графіка. У бібліотеці matplotlib є функція під назвою legend(), яка використовується для розміщення легенди на осях. Атрибут loc у legend() використовується для визначення розташування легенди. Значення loc за замовчуванням loc=”best” (upper left). Рядки «upper left», «upper right», «lower left», «lower right» розміщують легенду у відповідному куті осей фігури.

Функція show() відображає всі відкриті фігури. Параметр: block. Тип даних bool, необов'язковий. Чи чекати, поки всі фігури будуть закриті, перш ніж повернутися? Якщо True, потрібно заблокувати та запустити основний цикл GUI, доки не закриються всі вікна фігур. Якщо False, потрібно переконатись, що всі вікна фігур відображаються, і негайно повернутись. У цьому випадку не є відповідальності за те, щоб цикл подій був запущений, щоб мати відповідні цифри. За замовчуванням значення True в неінтерактивному режимі та значення False в інтерактивному режимі.

При створення моделі ми використовуємо 3 набори Conv2D з розміром ядра 3, MaxPool2D з розміром пулу 2 і BatchNormalization, потім вирівнюємо останній вихід BatchNormalizations і додаємо ще один прихований щільний шар.

Застосовуємо Droout після вирівнювання та після прихованого щільного шару, щоб ще більше зменшити дисперсію.

Першим параметром є epoch - ціле число, індекс епохи. Другим параметром є logs - результати метрики для цієї епохи навчання та для епохи перевірки, якщо перевірка виконується.

Навчаємо модель для 200 епох. Використовуємо функцію fit, яка навчає модель для фіксованої кількості епох (ітерацій набору даних).

Параметр x – це вхідні дані. Y - target дані. Як і вхідні дані x, це може бути масив Numpy або тензор TensorFlow. Epoch – є ціле число. Кількість епох для навчання моделі. Callbacks – є список екземплярів keras.callbacks.Callback. Використовуємо функцію створену раніше. Параметром verbose задаємо, як ми хочемо бачити прогрес навчання кожної епохи від 0 до 2. Встановлюємо значення 2 і таким чином бачимо, яка саме зараз епоха та її прогрес (рис. 3.7).

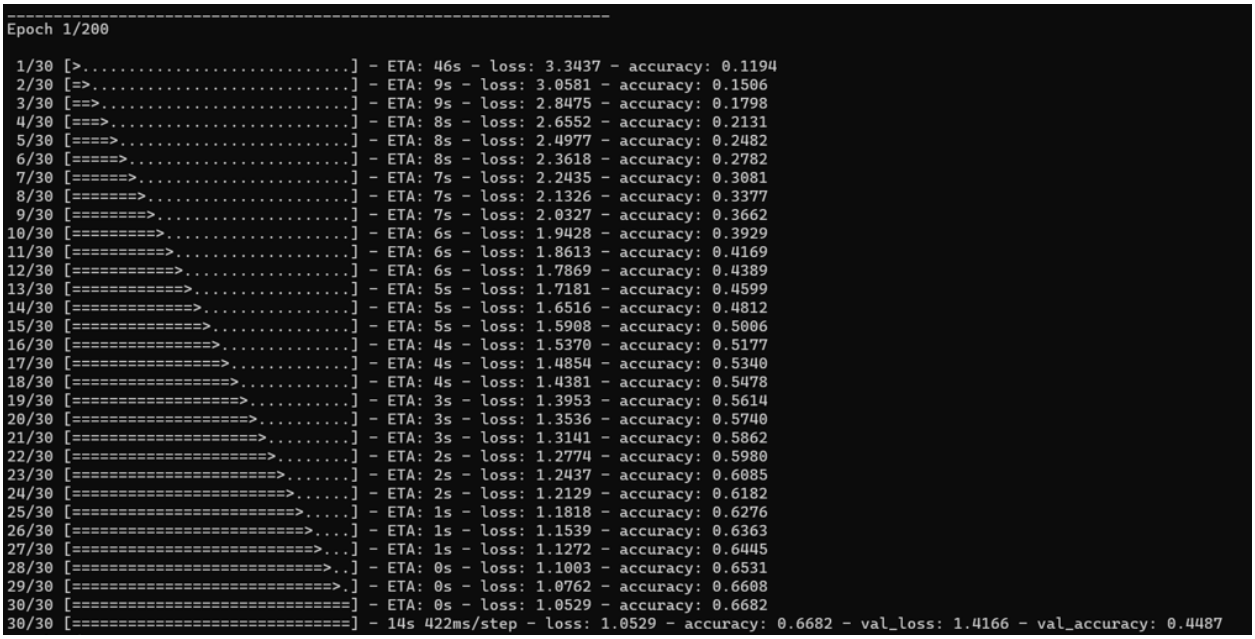


Рисунок 3.7 – Процес навчання нейронної мережі

Проаналізуємо результат, який отримано після тренування моделі протягом 200 епох. За визначенням, показник точності Acuuracy – це кількість отриманих правильних прогнозів (рис. 3.8).

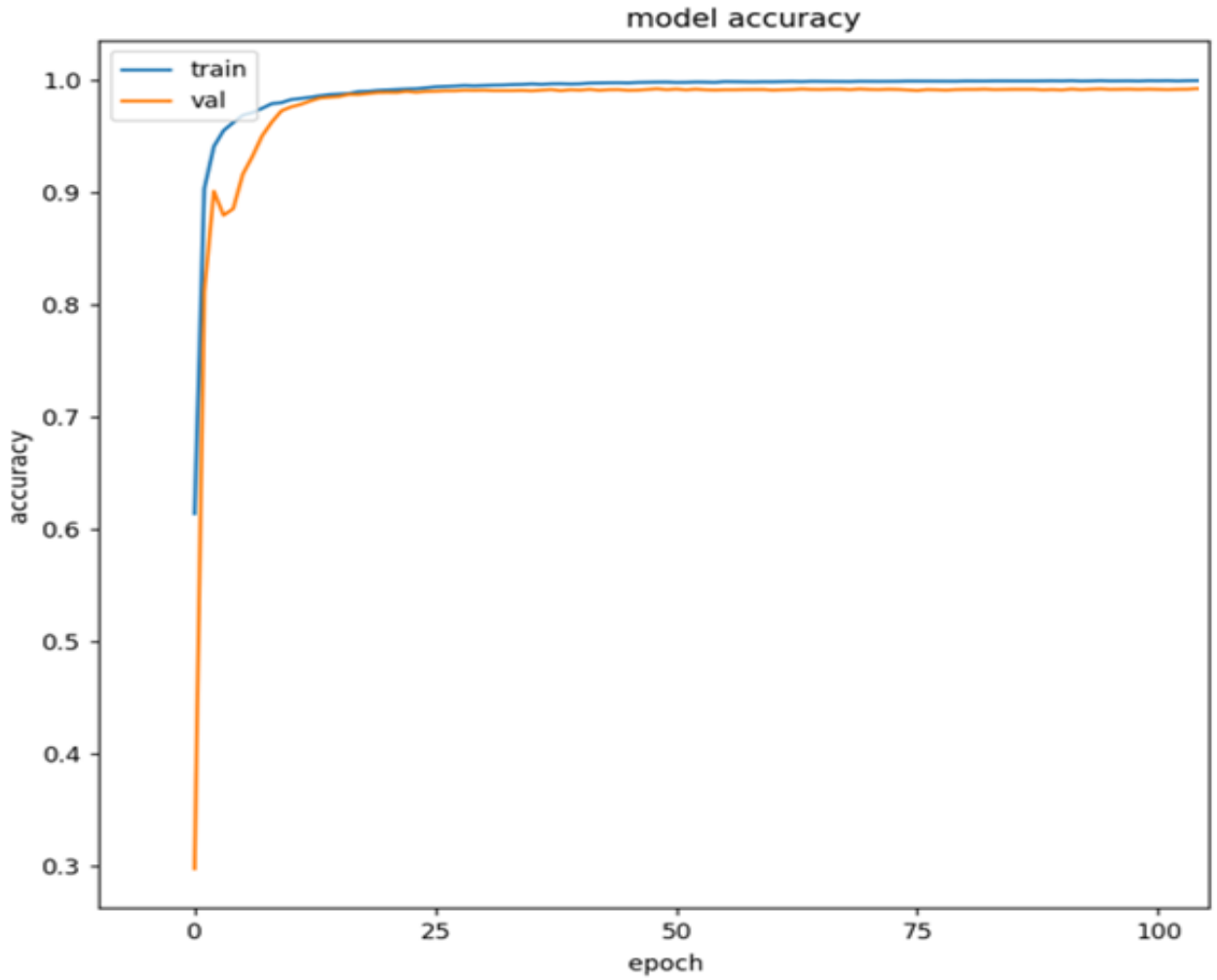


Рисунок 3.8 – Точність Accuracy під час навчання та тестування

У якості метрики для дослідження ефективності створеної моделі використовується стандартна функцію втрат categorical_crossentropy та accuracy – частка правильних відповідей алгоритму. Мета функція втрат в нейронній мережі є оцінка та оновлення ваг нейронів з метою поліпшення оцінки на наступному кроці. Функція втрат categorical_crossentropy намагається мінімізувати ентропію між справжніми та спрогнозованими ймовірностями із перехрестною втратою ентропії.

Loss (значення втрат) - це значення, що вказують на різницю з бажаним цільовим станом (рис. 3.9). Як можна помітити, loss під час навчання та валідації тут з невеликою розбіжністю.

Також бачимо, що acuuracy під час навчання та валідації підвищуються разом.

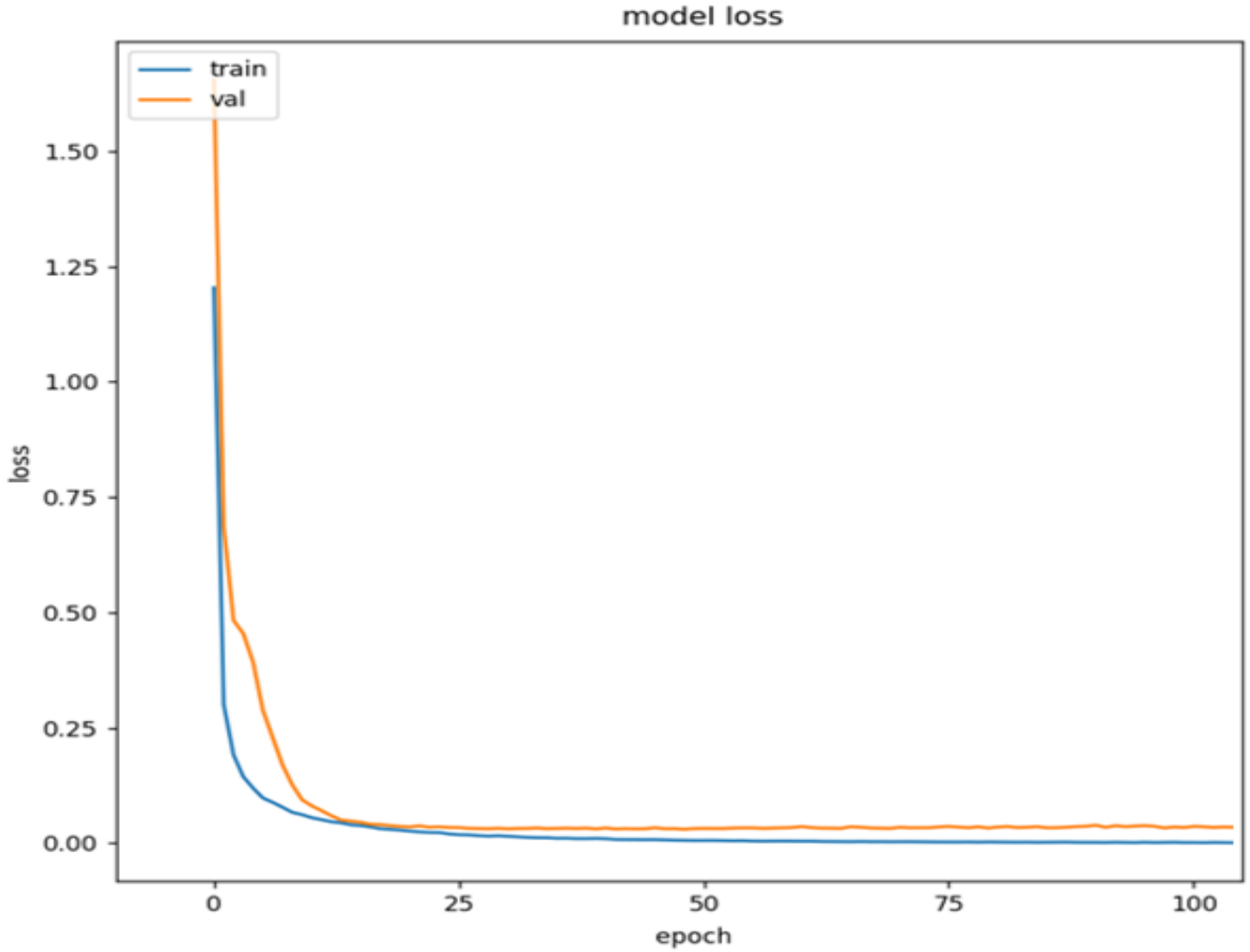


Рисунок 3.9 – Функція втрат Loss під час навчання та тестування

Після вдалого завершення навчання нейронної мережі завантажуємо тестові дані для моделі та отримуємо оцінку для певного зображення.

Функція model.predict() – модель може бути створена та оснащена навченими даними та використана для прогнозування.

Функція tf.math.argmax повертає індекс із найбільшим значенням по осях тензора. Параметр input - це тензор. Параметр axis є ціле число, вісь для скорочення. Встановлюємо значення -1, це означає, що індекс, який буде повернуто argmax, буде взято з останньої осі.

Побудована модель нейронної мережі може визначати рукописні цифри з високою точністю, оскільки поєднує шари згортки під час виділення ознак із повнозв'язними шарами й має засоби запобігання перенавчанню. Система забезпечує точність 99,19%, що є вище, ніж запропоновані раніше схеми. Крім того, значно скорочено час навчання та тестування, що забезпечує ефективність реалізованої моделі.

3.3 Сегментація зображень рукописних символів

Для сегментації зображення імпортуємо бібліотеки OpenCV, Numpy, Matplotlib. За допомогою бібліотеки Matplotlib на етапі розробки тестуємо як проходить процес сегментації [39]. Знаходимо необхідне зображення за вказаним шляхом використовуючи метод cv2.imread() та завантажуємо зображення із зазначеного файлу. Якщо зображення не може бути прочитане (через відсутність файлу, неправильних дозволів, непідтримуваного або неприпустимого формату), цей метод повертає порожню матрицю.

Створюємо функцію, для того щоб отримати порогове значення зображення. Спочатку ми перетворимо зображення на зображення в градаціях сірого за допомогою cv2.cvtColor(). Метод cv2.cvtColor() використовується для перетворення зображення з одного колірного простору на інший. В OpenCV є більше 150 методів перетворення колірного простору. Параметрами є src - це зображення, колірний простір якого має бути змінено. Параметр code є кодом перетворення колірного простору. Отримаємо висоту, ширину та кількість каналів зображення.

Далі використовуємо функцію cv2.threshold(). Для кожного пікселя застосовується однакове порогове значення. Якщо значення пікселя менше за порогове значення, воно встановлюється на 0, інакше встановлюється максимальне значення. Перший аргумент – вихідне зображення, яке має бути зображенням у градаціях сірого. Другим аргументом є порогове значення, яке

використовується для класифікації значень пікселів. Встановлюємо його зі значенням 80. Третій аргумент — це максимальне значення, яке призначається значенням пікселів, що перевищують порогове значення. Третій аргумент буде дорівнювати 255. OpenCV надає різні типи порогів, які визначаються четвертим параметром функції [40]. Використовуємо THRESH_BINARY. В результаті ми отримуємо бінарне зображення для його подальшої обробки.

Виділяємо що є рядком символів на заданому зображені, виконавши операцію dilation(). Ця операція полягає у згортанні зображення з деяким ядром, яке може мати будь-яку форму та розмір. Створюємо ядро з розміром прямокутника, який має меншу висоту та більшу ширину.

Ядро має певну точку прив'язки, що зазвичай є центром ядра. Коли ядро сканується за зображенням, ми обчислюємо максимальне значення пікселя, що перекривається ядром, і замінюємо піксель зображення в позиції точки прив'язки цим максимальним значенням.

Знаходимо зовнішні контури рядка з рукописними символами. Для цього використаємо функцію findContours(). Функція отримує контури з бінарного зображення. Контури є корисним інструментом для аналізу форми, виявлення та розпізнавання об’єктів. Передаємо зображення та використовуємо cv2.RETR_EXTERNAL та cv2.CHAIN_APPROX_NONE. При використанні cv2.RETR_EXTERNAL, повертає крайні зовнішні прапори. Усі дочірні контури залишаються позаду.

При використанні cv.CHAIN_APPROX_NONE, всі граничні точки зберігаються. Знаходимо координати контуру, щоб можна було намалювати прямокутник на лінії, також сортуємо контури зверху вниз використовуючи метод sorted().

Створюємо копію оригінального зображення. Використовуємо цикл для відображення контурів. Використовуємо cv2.boundingRect(cnt). Прямий обмежуючий прямокутник. Це прямий прямокутник, не враховує обертання об'єкта. Нехай (x, y) – верхня ліва координата прямокутника, а (w, h) – його

ширина та висота. Тоді, x,y,w,h = cv.boundingRect(cnt). За допомогою методу cv2.rectangle() малюємо прямокутник.

Після того як контури рукописних символів знайдено переходимо до розпізнавання символів.

Виконуємо обробку за допомогою вищезазначених методів із зміненими параметрами та сортування символів в кожному з заданих рядків. Сортування відбувається зліва направо. Для того, щоб відобразити результат сегментації, враховуємо розміри зображення, встановлюємо колір для відображення сегментації та подальшої обробки вхідного зображення.

Результат сегментації з використанням описаних методів зображено на рисунку 3.11.

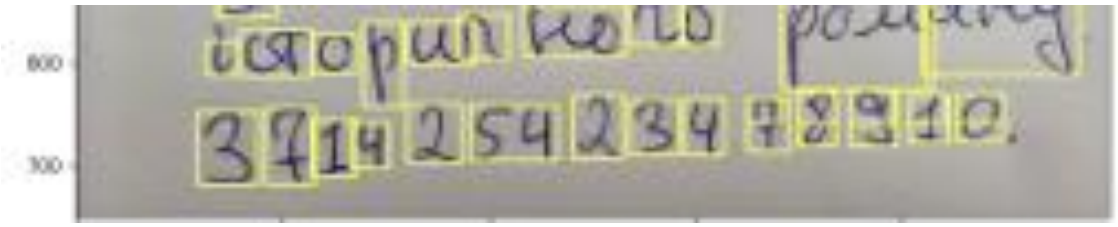


Рисунок 3.10 – Сегментація рукописних цифр

3.4 Інтерфейс системи розпізнавання рукописних символів

Розроблена система дозволяє автоматизувати роботу працівників пенсійного фонду, робота яких при оформленні людини на пенсію полягає у опрацюванні великої кількості рукописних документів, що займає велику кількість часу.

Розроблена система дозволяє вирішити лише одну складову роботи працівників пенсійного фонду по цифровізації записів трудової книжки – підрахунок стажу роботи, від якого залежить формула, за якою людині буде нарахована пенсія. Заміна етапу ручного розрахунку стажу та автоматизоване

розпізнавання дат, необхідних для цього, дозволяє зменшити час обробки документів та підвищити точність їх розрахунків.

Для створення інтерфейсу системи розпізнавання рукописних символів із зображень сторінок трудової книжки використовуємо фреймворк Python Kivy. В першу чергу імпортуємо клас App. Клас App є основою для створення програм Kivy [41]. Для того, щоб почати життєвий цикл програми викликаємо метод App.run().

Головне меню програми має 2 кнопки Open та Select відповідно (рис. 3.11). За допомогою кнопки Open ми відкриваємо файловий діалог. Бібліотека Tkinter містить зручні класи та функції для створення простих модальних діалогів для отримання значення від користувача. Метод tkinter.filedialog у поєднані з наступними класами та функціями забезпечує поєднання зовнішнього вигляду і параметрів конфігурації для налаштування поведінки. Параметр parent визначає чи буде діалогове вікно поверх інших вікон. Title задає заголовок вікна. Параметр filetypes є послідовність (мітка, шаблон) кортежів, дозволений символ підстановки «*».

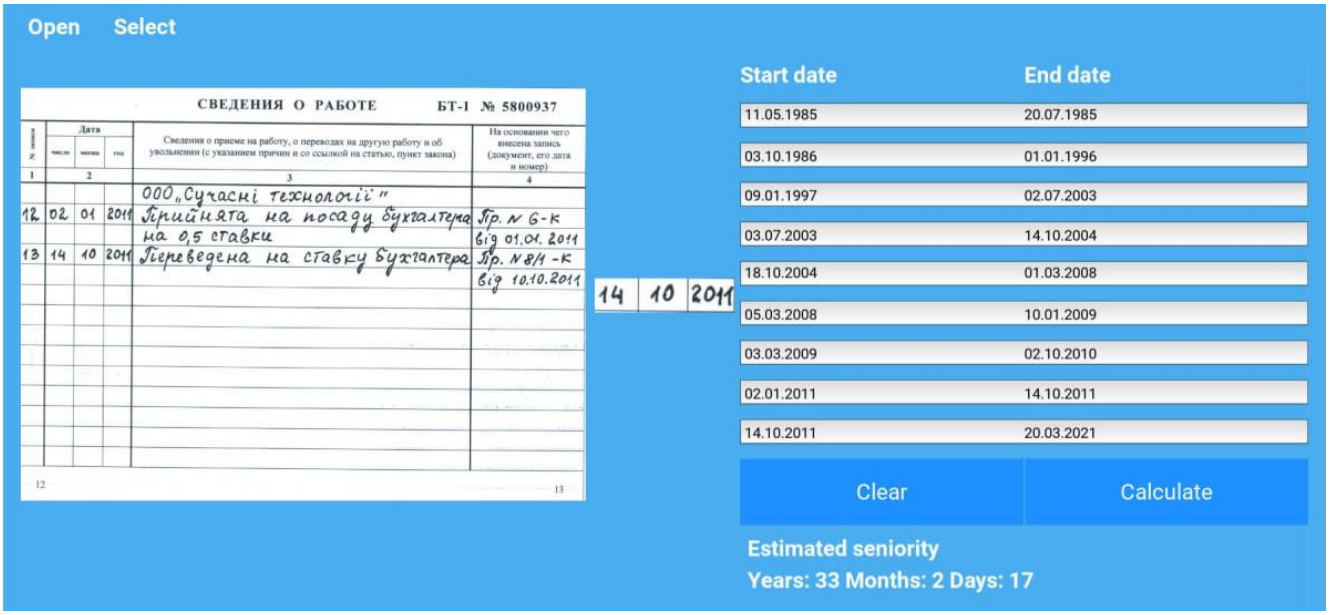


Рисунок 3.11 – Вікно системи розпізнавання рукописних символів

Створена система призначена для підрахунку стажу людини, яка оформлюється на пенсію за віком на основі розпізнавання цифр зі сканованих копій сторінок трудової книжки у полях, які містить дати прийняття та відрахування особи з певного місця роботи. Перед початком використання застосунку необхідно відсканувати сторінки трудової книжки.

Обравши пункт меню Open буде відкрито вікно папки, яка містить скановані копії, з яких по черзі необхідно обирати файли зображень. Обравши зображення певної сторінки трудової, воно буде відображено у лівій частину вікна програми (див. рис. 3.11).

Для того, щоб розрахувати стаж людини на певному місці роботи, необхідно мати дату прийняття на посаду і дату звільнення. Працівник відділу кадрів має обрати ці дати на зображенні сторінки трудової книжки, яке знаходиться у лівій частині вікна. Для цього необхідно натиснути кнопку Select.

Бібліотека алгоритмів комп'ютерного зору OpenCV надає зручні інструменти для реалізації поставленої задачі. За допомогою методу selectROI ми можемо вибрати діапазон вручну, вибравши область на зображенні. В якості параметрів передається window_name, що є назва вікна, де буде показано процес вибору. Source image зображення для вибору ROI, за вказаним шляхом та додаткові параметри showCrosshair та fromCenter, які відповідають за відображення прямокутника миші. Створення нового вікна є доречним, оскільки в такому випадку можна краще побачити яку саму дату необхідно виділити для подальшої обробки (рис 3.12).

Функцію selectRoi() повертає масив різних значень, які містять координату верхньої лівої точки виділеної області та ширину та висота ROI (область інтересу). Це фактично позиція пікселя у верхньому лівому куті + ширина та висота вибраного прямокутника на зображенні та вихідний масив у порядку:

[Top_Left_X, Top_Left_Y, Width, Height]

top_left_y = top_left_row = y1

top_left_x = top_left_col = x1

Після вибору ROI натискають кнопку пробілу або enter, щоб перейти до вибраної області. Клавіша c відповідає за скасування вибору. Використовуючи задані координати, обираємо конкретну виділену область. Обрізаємо зображення за допомогою масивів NumPy, результат буде показано у програмі поруч з зображенням трудової книги у середній частині вікна. У області виведення буде відображено обраний для розпізнання фрагмент тексту з датою (див. рис. 3.13).

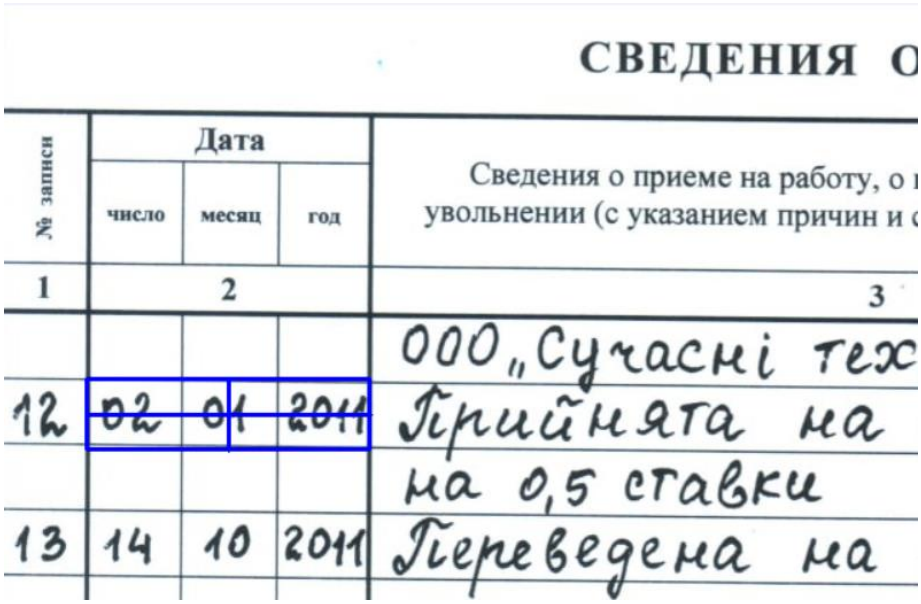


Рисунок 3.12 – Виділення необхідної дати

Рисунок 3.13 – Вирізаний для розпізнавання фрагмент зображення сторінки

У разі не чіткого вибору можна здійснити процедуру виділення фрагменту зображення з датою для розпізнання ще раз.

Після того, як дату було обрано, подане зображення обробляється, здійснюється його сегментація та подача на вхід нейронної мережі для розпізнавання чисел посимвольно.

Необхідно спочатку обрати дату зарахування на роботу, потім кінцеву дату на кожному місці роботи, які є на відкритому зображенні сторінки трудової книжки. Розпізнані дати будуть з'являтися у правій частині вікна під надписами Start Date та End Data у тому порядку, у якому вони були відібрані для розпізнавання. Зазвичай трудова книжка має декілька сторінок з записами. У програмі передбачено відкриття зображень сторінок трудової книжки із записами по черзі. На зображенні кожної сторінки дати відбираються у хронологічному порядку. Усі відібрані дати відображаються у правій частині вікна у відповідних полях. На рисунку 3.12 ми бачимо у правій частині вікна усі розпізнані дати однієї трудової книжки.

У програмі реалізована можливість редагування отриманих дат, оскільки при розпізнаванні можуть бути не усі цифри розпізнані правильно [42].

Після того, як у правій частині вікна буде відображено усі дати записів трудової книжки та відредаговано їх у разі необхідності, необхідно натиснути клавішу Calculate. Почнеться процес обробки отриманих результатів – розрахунку точної кількості років, місяців та днів, які становлять повний стаж людини. Розпізнані дані передаються до методу datetime модуля datetime. Клас datetime() потребує трьох параметрів для створення дати: рік, місяць, день. Розрахований стаж буде відображено у правій частині вікна під областю із датами (див. рис. 3.14).

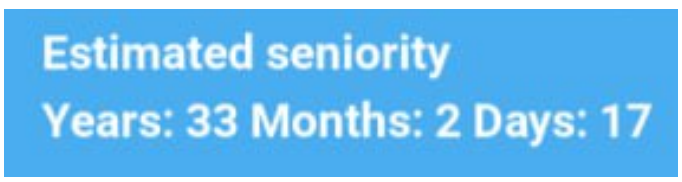


Рисунок 3.14 – Виведення розрахованого стажу

Натискання клавіши Clear надає можливість швидко очистити поля для початку роботи з іншою трудовою книжкою або перерахунку стажу поточної трудової з початку у разі необхідності.

Для визначення якості розпізнавання дат створеною системою було сформовано набір 200 зображень рукописних дат, на сторінках трудової книжки та виявлено, що точність їх розпізнавання становила 75%, що є непоганим результатом, оскільки система отримувала вирізані зображення дат, які потребували сегментації. Розрахунок стажу у разі корегування введених дат, система здійснює точно, що також було протестовано на вибірці з періодів роботи, які містили дати з охопленням різних часових періодів.

Висновки до розділу 3

Здійснено розробку, програмну реалізацію та тестування системи розпізнавання рукописних символів на зображеннях сканованих сторінок трудової книжки, яка дозволяє автоматизувати роботу працівників пенсійного фонду, які займаються опрацюванням великої кількості рукописних документів. Розроблена система дозволяє підвищити ефективність роботи працівників пенсійного фонду по цифровізації записів трудової книжки та підрахунку стажу при оформленні пенсії. Заміна етапу ручного розрахунку стажу та автоматизоване розпізнавання дат, необхідних для цього, дозволяє зменшити час обробки документів та підвищити точність їх розрахунків.

Система розпізнавання дозволяє здійснювати сегментацію дат із вирізаних фрагментів зображення сторінок трудової книжки та окремі символи, які подають на вхій розробленої нейронної мережі. Розроблена система базується на створеній моделі згорткової нейронної мережі містить послідовність 14 шарів різних типів. Три згорткових шари Conv2D з функцією активації ReLu та розміром фільтру 3x3 застосовують операцію згортки, генеруючи результат згортки до наступного шару. Після кожного згорткового шару розміщено агрегувальний шар MaxPool2D

із розміром фільтру 2x2 та шар нормалізації BatchNormalization з метою зменшення перенавчання моделі. Потім розміщено шар Flatten, який перетворює дані 2D-матриці у вектор. Модель містить два шари Dropout, налаштовані на випадкове виключення частки нейронів у шарі для запобігання перенавчання моделі, і два повнозв’язні шари Dense. Вихідний шар має 10 нейронів і функцію активації softmax для виведення ймовірнісних прогнозів кожного класу.

Побудована модель нейронної мережі забезпечує високу точність розпізнавання рукописних цифр, яка становить 99,19%. У цілому точність розпізнавання дат із сканованих зображень сторінок трудової книжки становила 75%, що також є прийнятним результатом.

ВИСНОВКИ

У результаті проведеного дослідження установлено, що актуальність задачі розпізнавання рукописних символів в умовах інформатизації суспільства різко зросла. Для вирішення цієї проблеми створюють інформаційні системи, здатні розпізнавати символи за їх рукописним зображенням, отриманим із різних джерел. Однак розпізнавання зображень рукописних документів при їх цифровізації все ще не є надійним. Існує потреба у системах, які забезпечують високу точність та надійність розпізнавання у різних сферах автоматизації обробки документів.

Виявлено, що фундаментальними етапами розпізнавання рукописних символів є сегментація, виділення ознак і класифікація. Виявлено, що згорткові нейронні мережі відносять до глибинних нейронних мереж (англ. Deep Neural Network, DNN), які перетворюють вхідні дані у вихідні, ієрархічно виділяючи та агрегуючи ознаки шляхом підвищення рівня абстракції даних у напрямку від входів до виходів мережі. Найефективнішими є моделі згорткових нейронних мереж (англ. Convolutional Neural Network, CNN), які досягають високої точності класифікації й є найбільш прийнятним інструментом для розпізнавання зображень.

Для розробки системи розпізнавання рукописних символів було використано мову програмування Python, середовище розробки Visual Studio, фреймворки TensorFlow і Keras. Бібліотека OpenCv була використана для виділення фрагментів зображення, бінаризації та сегментації. Для управляння даними та виведення результатів було використано бібліотеки NumPy, Matplotlib. Графічний інтерфейс створено за допомогою фреймворку Kivy.

Здійснено розробку, програмну реалізацію та тестування системи розпізнавання рукописних символів на зображеннях сканованих сторінок трудової книжки, яка дозволяє автоматизувати роботу працівників пенсійного

фонду, які займаються опрацюванням великої кількості рукописних документів. Розроблена система дозволяє підвищити ефективність роботи працівників пенсійного фонду по цифровізації записів трудової книжки та підрахунку стажу при оформленні пенсії. Заміна етапу ручного розрахунку стажу та автоматизоване розпізнавання дат, необхідних для цього, дозволяє зменшити час обробки документів та підвищити точність їх розрахунків.

Система розпізнавання дозволяє здійснювати сегментацію дат із вирізаних фрагментів зображення сторінок трудової книжки та окремі символи, які подають на вхід розробленої нейронної мережі. Розроблена система базується на створеній моделі згорткової нейронної мережі містить послідовність 14 шарів різних типів. Три згорткових шари Conv2D з функцією активації ReLu та розміром фільтру 3x3 застосовують операцію згортки, генеруючи результат згортки до наступного шару. Після кожного згорткового шару розміщено агрегувальний шар MaxPool2D із розміром фільтру 2x2 та шар нормалізації BatchNormalization з метою зменшення перенавчання моделі. Потім розміщено шар Flatten, який перетворює дані 2D-матриці у вектор. Модель містить два шари Dropout, налаштовані на випадкове виключення частки нейронів у шарі для запобігання перенавчання моделі, і два повнозв’язні шари Dense. Вихідний шар має 10 нейронів і функцію активації softmax для виведення ймовірнісних прогнозів кожного класу.

Побудована модель нейронної мережі забезпечує високу точність розпізнавання рукописних цифр, яка становить 99,19%. У цілому точність розпізнавання дат із сканованих зображень сторінок трудової книжки становила 75%, що також є прийнятним результатом.

Поставлені завдання виконано повністю, однак функціонал розробленого застосунку може бути розширений у подальшому шляхом надання можливості розпізнавання рукописних символів зі сторінок трудової книжки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A. Korotynskyi, O. Zhuchenko A system of automated control for the baking process that minimizes the probability of defects, Eastern-European Journal of Enterprise Technologies, 2020, (2-103), стр. 58–67.
2. Кобилін О.А., Творошенко І.С. (2021). Методи цифрової обробки зображень: навч. посібник. *Харків: ХНУРЕ.*
3. Гонсалес, Р., & Вудс, Р. (2019). *Цифровая обработка изображений.* Litres.
4. Как работает распознавание рукописного текста. URL: https://vc.ru/ml/96273-kak-rabotaet-raspoznavanie-rukopisnogo-teksta
5. Соколенко Д. Г., Корнага Я. І. «Система розпізнавання писемних символів за допомогою нейронної мережі», *Вчені записки ТНУ імені В.І. Вернадського. Серія Технічні науки* Том 29 (68) Ч. 2 № 5 2018 с. 56-58
6. Розпізнавання рукописного тексту. URL: https://idr.com.ua/info/rukopisniy-tekst.html
7. Сверточная нейронная сеть, часть 1: структура, топология, функции активации и обучающее множество [Електронний ресурс] // Хабр URL: https://habr.com/post/348000/
8. OCR-конвейер для обработки документов. URL: https://habr.com/ru/company/arcadia/blog/505950/
9. Карпович Артем Валерійович «Використання згорткових нейронних мереж для задачі класифікації текстів», *International scientific journal «Internauka»* // № 14(54), 2018 // Technical sciences // с. 69-73
10. Лазарев В. М., Свиридов А. П. Нейросети и нейрокомпьютеры. Монография. - М.: Академия, 2011. - 131 с
11. Джеффри Е. Хинтон. Как обучаются нейронные сети. // В мире науки – 2012. - № 11 - С. 103-107.

12. Нейросетевое моделирование: многослойный персептрон [Електронний ресурс] // Шитиков В.К., Розенберг Г.С., Зинченко Т.Д. – URL: www.ievbran.ru/kiril/Library/Book1/content394/content394.htm
13. Машинное зрение на Python. Обучаем нейросеть распознавать цифры. URL: https://medium.com/@enduranceprog/machine-vision-digits-94eb258c6ff8
14. Т. М. Басюк, В. В. Литвин, Л. М. Захарія, Н. Е. Кунанець Машинне навчання: Навчальний посібник Львів: Видавництво «Новий Світ - 2000», 2021. - 315 с.
15. Гудфеллоу Я., Бенджио И., Курвилль А. Г. Глубокое обучение / пер. с анг. А. А. Слинкина. – 2-е изд., испр. – М.: ДМК Пресс, 2018. – 652 с.
16. Кононова К. Ю. Машинне навчання: методи та моделі: підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка» / К. Ю. Кононова. – Харків: ХНУ імені В. Н. Каразіна, 2020. – 301 с.
17. Машинне навчання простими словами. Електронний ресурс. Код доступу: http://www.mmf.lnu.edu.ua/ar/1739
18. ТОП-4 программ для OCR распознавания рукописного текста. URL: https://pdf.iskysoft.com/ru/ocr-pdf/handwriting-ocr.html
19. Matplotlib. URL: Matplotlib — Visualization with Python
20. Welcome to Python.org: вебсайт. URL: https://www.python.org/
21. Creation of virtual environments. URL: venv — Creation of virtual environments — Python 3.11.2 documentation
22. Python Deep Learning Tutorial: вебсайт. URL: https://www.tutorialspoint.com/python_deep_learning/index.htm
23. NumPy: вебсайт. URL: https://numpy.org/
24. TensorFlow: вебсайт. URL: https://www.tensorflow.org/
25. Keras: the python deep learning API: вебсайт. URL: https://keras.io/
26. Франсуа Шолле. Глубокое обучение на Python. – СПб.: Питер, 2018. – 400 с.: ил. – (Серия «Библиотека программиста»). ISBN 978-5-4461-0770-4

27. Welcome to Kivy: вебсайт. URL: Welcome to Kivy — Kivy 2.1.0 documentation
28. Kivy: The Open Source Python App development Framework: вебсайт. URL: Kivy: Cross-platform Python Framework for GUI apps Development
29. OpenCV: веб-сайт. URL: Home - OpenCV
30. Python + OpenCV + Keras. URL: https://habr.com/ru/post/466565/
31. Visual Studio. URL: Visual Studio: IDE and Code Editor for Software Developers and Teams (microsoft.com)
32. Visual Studio documentation. URL: Visual Studio documentation | Microsoft Learn
33. Feiyang Chen, Nan Chen, Hanyang Mao, Hanlin Hu Assessing Four Neural Networks on Handwritten Digit Recognition Dataset (MNIST) / *Chuangxinban journal of computing*, june 2018, URL: https://arxiv.org/pdf/1811.08278. pdf
34. A Practiacal Guide to Relu: вебсайт. URL: https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7
35. Ali, S., Shaukat, Z., Azeem, M. et al. An efficient and improved scheme for handwritten digit recognition based on convolutional neural network. *SN Applied Science*. 2019. Vol. 1, Issue 9. doi: 10.1007/s42452-019-1161-5.
36. Распознавание рукописных цифр на P-ython + GUI. URL: https://pythonru.com/primery/raspoznavanie-rukopisnyh-cifr-na-p-ython-gui
37. Yifan Wang, Fenghou Li, Hai Sun, Wenbo Li, Cheng Zhong, Xuelian Wu, Hailei Wang, Ping Wang Improvement of MNIST Image Recognition Based on CNN, 7th Annual International Conference on GeoSpatial Knowledge and Intelligence IOP Conf. Series: Earth and Environmental Science 428 (2020). URL: https://iopscience.iop.org/article/10.1088/1755-1315/428/1/012097/pdf
38. Orhan G. Yalçın Image Classification in 10 Minutes with MNIST Dataset. URL: https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d

39. Matplotlib documentation. URL: Matplotlib documentation — Matplotlib 3.7.0 documentation
40. OpenCV Open Source Computer Vision. URL: OpenCV: OpenCV modules
41. Kivy Tutorial. URL: Kivy Tutorial - GeeksforGeeks
42. Python Kivy Tutorials. URL: Python Kivy Tutorials - How To Create Apps with Python - techwithtim.net
43. Конституція України: Закон від 28.06.1996 № 254к/96-ВР // Конституція України. URL: https://zakon.rada.gov.ua/laws/show/254%D0%BA/96-%D0%B2%D1%80#Text
44. Про охорону праці від 14.10.1992 № 2694-XII // Закон України про охорону праці. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text
45. Кодекс законів про працю України від 10.12.71 № 322-VII // Кодекс законів про працю України URL: https://zakon.rada.gov.ua/laws/show/322- 08#Text
46. Кодекс цивільного захисту України від 02.10.2012 № 5403-VI // Кодекс цивільного захисту України URL: https://zakon.rada.gov.ua/laws/show/5403-17#Text
47. Про затвердження Типового положення про службу охорони праці. (НПАОП 0.00-4.35-04) від 15.11.2004 № 255 // Наказ про затвердження Типового положення про службу охорони праці. URL: https://zakon.rada.gov.ua/laws/show/z1526-04#
48. Про затвердження Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці та Переліку робіт з підвищеною небезпекою від 26.01.2005 № 15 // Наказ Про затвердження Типового положення про порядок проведення навчання і перевірки знань з питань охорони праці та Переліку робіт з підвищеною небезпекою URL: https://zakon.rada.gov.ua/laws/show/z0231-05#Text
49. Про затвердження Порядку розслідування та обліку нещасних випадків, професійних захворювань та аварій на виробництві від 17.04.2019 р. № 337 // Постанова про затвердження Порядку розслідування та обліку нещасних

випадків, професійних захворювань та аварій на виробництві URL: https://zakon.rada.gov.ua/laws/show/337-2019-%D0%BF#Text

50. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН від 10.12.1998 № 80 // Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН URL: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text
51. Типову інструкцію розроблено Українським НДІ цивільного захисту відповідно до ст. 130 Кодексу цивільного захисту України. URL: ТИПОВА ІНСТРУКЦІЯ
52. Про затвердження Правил вибору та застосування засобів індивідуального захисту органів дихання. URL: Про затвердження Правил вибору т... | від 28.12.2007 № 331 (rada.gov.ua)