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

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних
інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2022 р.

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ІНФОРМАЦІЙНА МОДЕЛЬ ДЛЯ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЛІТЕР

Спеціальність 122 «Комп'ютерні наук»

122 – БКР – 402.1810224

Виконав: студент 4-го курсу, групи 402
_____О. М. Сова
«__» червня 2022 р.

Керівник: к.ф.-м.н., доцент
_____І. В. Кулаковська
«__» червня 2022 р.

Миколаїв – 2022

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

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2021 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук Сові Олександру Михайловичу.

1. Тема кваліфікаційної роботи «Інформаційна модель для розпізнавання рукописних літер».

Керівник роботи Кулаковська Інесса Василівна, к.ф.-м.н., доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «_7_» _грудня_ 2021 р. № 318

2. Строк представлення кваліфікаційної роботи студентом «___» _____ 20__ р.

3. Вхідні (початкові) дані до роботи: база даних з зображеннями рукописних літер.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки): аналіз предметної області і постановка завдання; моделі, методи та інформаційні технології для вирішення задачі розпізнавання рукописних літер; аналіз даних, створення та тренування моделі; розробка вебзастосунку та використання створеної моделі.

5. Перелік графічного матеріалу: презентація, 2 таблиці, 36 рисунків.

6. Завдання до спеціальної частини: «Охорона праці при користуванні екранними пристроями»

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	**Макарова О.В., ст. викладач**	

Керівник роботи _____к.ф.-м.н., доцент Кулаковська І. В. _____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання _____Сова О. М._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «__» _____ 2021 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Інформаційна модель для розпізнавання рукописних літер

№	Найменування роботи	Початок	Закінчення	Примітки
1	Визначення теми та керівника Подання заяви на затвердження теми та керівників ДР	01.10.2021	01.11.2021	Виконано
2	Огляд літератури за темою роботи	01.12.2021	08.12.2021	Виконано
3	Складання календарного плану роботи на весь період виконання ДР	08.12.2021	10.12.2021	Виконано
4	Проходження переддипломної практики, збір та аналіз матеріалів до ДР	23.05.2022	04.06.2022	Виконано
5	Аналіз предметної області та програмної частини	28.02.2022	05.03.2022	Виконано
6	Моделювання і реалізація нейронної мережі	06.03.2022	10.03.2022	Виконано
7	Реалізація графічного інтерфейсу для застосунку та тестування	19.03.2022	05.04.2022	Виконано
8	Розробка спеціальної частини з охорони праці	19.05.2022	23.05.2022	Виконано
9	Написання пояснювальної записки	23.05.2022	27.05.2022	Виконано
10	Обговорення результатів з керівником, створення презентації	27.05.2022	29.05.2022	Виконано
11	Попередній захист ДР на засіданні комісії кафедри	30.05.2022	31.05.2022	Виконано
12	Остаточне оформлення пояснювальної записки та слайдів доповіді для захисту	01.06.2022	14.06.2022	Виконано
13	Написання відгуку на БКР	14.06.2022	15.06.2022	Виконано
14	Подання БКР рецензенту	19.06.2022	22.06.2022	
15	Подання БКР, її електронної копії та інших документів (відгуку, рецензії) до захисту	20.06.2022	22.06.2022	
16	Захист БКР перед Державною екзаменаційною комісією (ЕК)	27.06.2022	29.06.2022	

Розробив студент Сова О.М. _____
(прізвище та ініціали) (*підпис*)

Керівник роботи к.ф.-м.н., доцент Кулаковська І. В. _____
(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«____» _____ 2021 р.

АНОТАЦІЯ

до бакалаврської дипломної роботи

«Інформаційна модель для розпізнавання рукописних літер»

Студент: Сова Олександр Михайлович

Керівник: к.ф.-м.н., доцент Кулаковська І.В.

Об'єкт дослідження – інформаційні моделі та їх методи.

Мета роботи – створення та тренування власної нейронної мережі для розпізнавання літер та розробка веб-застосунку для використання та наглядної перевірки роботи створеної нейронної мережі.

Предмет дослідження – розпізнавання рукописних літер.

Дипломна робота складається з загальної частини, та індивідуальних томів. Пояснювальна записка індивідуального тому роботи складається зі вступу, чотирьох розділів, висновків та спеціальної частини з охорони праці.

У вступі визначається мета роботи та етапи потрібні для її досягнення.

У першому розділі проводиться аналіз предметної області та огляд існуючих аналогів для розробки ПЗ.

У другому розділі проводиться вибір та обґрунтування моделей, методів та технологій розробки.

У третьому розділі описується процес реалізації моделі та побудова нейронної мережі.

У четвертому розділі описується створення веб-застосунку та аналіз результатів роботи моделі.

У висновках проводиться аналіз проведеної роботи та отриманих результатів.

У спеціальній частині з охорони праці аналізується стан охорони праці під час користування екранними пристроями. Робота містить 2 таблиці, 36 рисунків та 41 літературне джерело. Загальний обсяг дипломної роботи складає 64 сторінки.

Ключові слова: інформаційна модель, нейронна мережа, програмне забезпечення, моделювання.

ABSTRACT

to bachelor thesis

«Information model for handwritten character recognition»

Student: Sova Oleksandr Mykhaylovych

Leader: Docent Kulakovska Inessa Vasylivna

Object of research - information models and their methods.

The purpose of the thesis – creation and training of own neural network for handwritten character recognition and development of web application for use and visual inspection of the created neural network.

Subject of research – handwritten character recognition.

Thesis consist of general part, and individual volumes.

Explanatory note of this individual volume of thesis consists of introduction, four sections, conclusions and special part on labor protection.

The introduction defines the purpose of the work and the steps required to achieve it.

The first section analyzes the subject area and reviews the existing analogues for software development.

The second section selects and substantiates models, methods and technologies of development.

The third section describes the process of implementing the model and building a neural network.

The fourth section describes how to create a web application and analyze the results.

The special part on labor protection analyzes the state of labor protection during the use of screen devices.

The work contains 2 tables, 36 drawings and 41 literary sources. Total amount of thesis is 64 pages.

Keywords: information models, neural network, software, modeling.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ВДТ	–	Візуальний дисплейний термінал
ЕОМ	–	Електронна обчислювальна машина
КПО	–	Коефіцієнт природної освітленості
ПЕОМ	–	Персональні електронні обчислювальні машини
ПК	–	Персональний комп’ютер
ССБП	–	Система стандартів безпеки праці

API	–	Application programming interface
CNN	–	Convolutional neural network
CSV	–	Comma-separated values
GPU	–	Graphics processing unit
ERP	–	Enterprise resource planning
HTML	–	Hypertext markup language
LDA	–	Linear discriminant analysis
NDA	–	Normal discriminant analysis
NIST	–	National Institute of Standards and Technology
NLP	–	Natural language processing
OCR	–	Optical character recognition
PDF	–	Portable document format
ReLU	–	Rectified linear unit
RTF	–	Rich text format
TPU	–	Tensor processing unit
QDR	–	Quadratic discriminant analysis
QR	–	Quick response

Пояснювальна записка

до кваліфікаційної роботи

на тему:

ІНФОРМАЦІЙНА МОДЕЛЬ ДЛЯ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЛІТЕР

Спеціальність 122 «Комп’ютерні науки»

122 – БКР – 402.21810224

Виконав: студент 4-го курсу, групи 402
_____О. М. Сова
«27» травня 2022 р.

Керівник: к.ф.-м.н., доцент
_____І. В. Кулаковська
«__» червня 2022 р.

Миколаїв – 2022

ВСТУП

Розпізнавання рукописних літер та символів стають все більш важливими в сучасному оцифрованому світі через їх практичне застосування в різних повсякденних справах. Про це свідчить той факт, що в останні роки були розроблені або запропоновані до використання різні системи розпізнавання в різних галузях, де потрібна висока ефективність класифікації. Системи, які використовуються для розпізнавання рукописних букв, символів і цифр, допомагають людям вирішувати складніші завдання, які в іншому випадку були б трудомісткими та дорогими. Хорошим прикладом є використання систем автоматичної обробки, які використовуються в банках для обробки банківських чеків. Без автоматизованих систем обробки банківських чеків банку потрібно було б найняти багато співробітників, які можуть бути не такими ефективними, як комп’ютеризована система обробки. Системи розпізнавання рукописного тексту можуть бути натхненні біологічними нейронними мережами, які дозволяють людям і тваринам вчитися та моделювати нелінійні та складні відносини [1].

Теорія розпізнавання образів — це розділ кібернетики, який розвиває теоретичні основи й методи класифікації та ідентифікації явищ, предметів, ситуацій, процесів, сигналів і подібних об'єктів, що характеризуються кінцевим набором деяких властивостей і ознак. Такі задачі вирішуються досить часто, наприклад, при переході або проїзді вулиці за сигналами світлофора. Розпізнавання кольору лампи світлофора, що засвітилася, та знання правил дорожнього руху дозволяє прийняти правильне рішення про те, чи можна переходити вулицю в цей момент часу.

Розпізнавання образів — це автоматичне розпізнавання закономірностей і закономірностей у даних. Воно має застосування в статистичному аналізі даних, обробці сигналів, аналізі зображень, пошуку інформації, біоінформатиці, стисканні даних, комп’ютерній графіці та машинному навчанні. Розпізнавання образів бере початок у статистиці та інженерії; деякі сучасні підходи до розпізнавання образів

включають використання машинного навчання через збільшення доступності великих даних і нового великого обсягу обчислювальної потужності [2].

Однією з найпростіших задач розпізнавання образів є розпізнавання рукописного вводу літер — її досить часто використовують як вступ до технологій *deep learning*.

Мета даної роботи – створення та тренування власної нейронної мережі для розпізнавання літер та розробка вебзастосунку для використання та наглядної перевірки роботи створеної нейронної мережі.

Об'єкт дослідження – розпізнавання рукописних літер.

Предмет дослідження – інформаційні моделі та їх методи.

Для досягнення мети нам потрібно виконати такі **задачі**:

- проаналізувати і поставити задачу розпізнавання рукописних літер;
- розглянути сучасні аналоги розробки ПЗ для розпізнавання тексту;
- розглянути та проаналізувати методи і технології, які можуть бути застосовані для вирішення поставленої задачі розпізнавання рукописних літер;
- створити, розробити і натренувати модель багатошарового перцептрону для вирішення задачі мультикласифікації на прикладі рукописних літер із спеціальної бази даних NIST;
- створити вебзастосунок, інтерфейс якого містить полотно для роботи з моделлю, що буде інтегровано у вебзастосунок, та стовпчикову діаграму із літерами алфавіту, яка показує передбачення мережі на основі намальованої літери;
- перевірено коректність роботи застосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз задачі оптичного розпізнання символів

Зоровий апарат людини є одним із чудес світу. У кожній півкулі нашого мозку у людини є первинна зорова кора, також відома як V1, що містить 140 мільйонів нейронів, з десятками мільярдів зв’язків між ними. І все ж людський зір включає не лише V1, а й цілу серію зорових корок — V2, V3, V4 та V5 — які здійснюють ще більш складну обробку зображень. Ми носимо в голові суперкомп’ютер, налаштований еволюцією протягом сотень мільйонів років, який чудово адаптований для розуміння візуального світу. Розпізнати рукописні літери непросто. Скоріше, ми, люди, вражаюче добре розуміємо те, що показують нам очі. Але майже вся ця робота виконується несвідомо. Тому ми зазвичай не розуміємо, наскільки важку проблему вирішують наші візуальні системи.

Складність візуального розпізнавання образів стає очевидною, якщо ви спробуєте написати комп’ютерну програму для розпізнавання літер. Те, що здається легким, коли ми робимо це самі, раптом стає надзвичайно складним. Прості інтуїції щодо того, як ми розпізнаємо фігури виявляється не так просто виразити алгоритмічно. Коли ви намагаєтеся сформулювати такі правила точно, ви швидко губитеся в купі винятків, застережень і особливих випадків.

Нейронні мережі підходять до даної проблеми по-іншому. Ідея полягає в тому, щоб взяти велику кількість рукописних літер, відомих як навчальні приклади, а потім розробити систему, яка зможе вчитися на цих навчальних прикладах. Іншими словами, нейронна мережа використовує приклади для автоматичного визначення правил розпізнавання рукописних цифр. Крім того, збільшуючи кількість навчальних прикладів, мережа може дізнатися більше про почерк і таким чином підвищити його точність.

Ми зосереджуємось на розпізнаванні рукописного тексту, оскільки це чудовий прототип проблеми для вивчення нейронних мереж загалом. Будучи прототипом, він потрапив у солодке місце: це складно — розпізнати рукописні літери не так вже й просто, — але це й не так важко, щоб вимагати надзвичайно складного рішення чи величезної обчислювальної потужності. Крім того, це чудовий спосіб розробити більш продвинуті методи, такі як глибоке навчання [3].

1.2 Огляд існуючих аналогів розробки для розпізнання тексту

1.2.1 Google Lens

Google Lens — це технологія розпізнавання зображень, розроблена Google, призначена для отримання відповідної інформації, пов’язаної з об’єктами, які вона ідентифікує за допомогою візуального аналізу на основі нейронної мережі. Направляючи камеру телефону на об’єкт, Google Lens намагатиметься ідентифікувати об’єкт, зчитуючи штрих-коди, QR-коди, етикетки та текст, і показуватиме відповідні результати пошуку, вебсторінки та інформацію. Наприклад, якщо навести камеру пристрою на етикетку Wi-Fi, яка містить назву мережі та пароль, вона автоматично підключиться до відсканованої мережі Wi-Fi. Або якщо навести камеру на якусь квітку чи рослину, програма видає результати розпізнавання у вигляді зображень та інформацію з пошукової системи Google. Об’єктив порівнює об’єкти на вашій картинці з іншими зображеннями та ранжує ці зображення за їх схожістю та відповідністю до об’єктів на оригінальному зображенні. Lens також інтегровано з додатками Google Photos та Google Assistant [4]. Lens використовує більш просунуті процедури глибокого навчання, щоб розширити можливості виявлення, подібно до інших програм. Об’єктив також використовує своє розуміння об’єктів на вашому зображенні, щоб знайти інші відповідні результати в Інтернеті. Об’єктив також може використовувати інші корисні сигнали, такі як слова, мова та інші метадані на хост-сайті зображення, щоб визначити рейтинг і релевантність. Аналізуючи зображення, Lens часто генерує

кілька можливих результатів і оцінює ймовірну релевантність кожного результату. Інколи об’єктив може звузити ці можливості до одного результату. Скажімо, що Lens дивиться на собаку, яку він ідентифікує як, ймовірно, 95% німецьку вівчарку і 5% коргі. У цьому випадку Lens може показати результат лише для німецької вівчарки, яка, на думку Lens, найбільш схожа візуально. В інших випадках, коли Lens впевнений, що розуміє, який об’єкт на зображенні вас цікавить, він поверне результати пошуку, пов’язані з об’єктом. Наприклад, якщо зображення містить певний продукт, наприклад джинси чи кросівки, Lens може повернути результати, надавши більше інформації про цей продукт, або результати покупок для продукту. Для отримання таких результатів Lens також може покладатися на доступні сигнали, як-от оцінки користувачів продукту [5]. Приклад роботи ПЗ наведено на рис. 1.1.

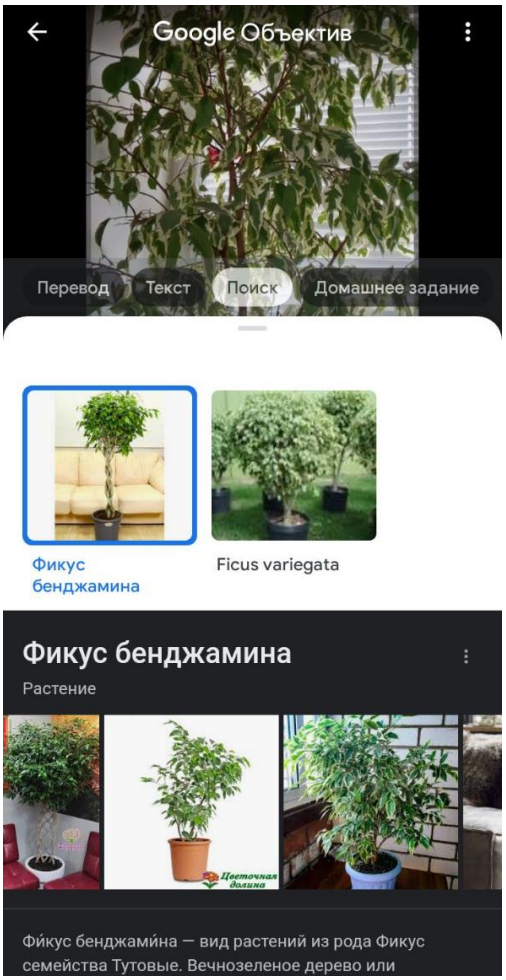

Рисунок 1.1 – Приклад розпізнавання рослини

1.2.2 ABBYY FineReader

ABBYY FineReader PDF — це програма для оптичного розпізнавання символів (OCR), розроблена компанією ABBYY з підтримкою редагування файлів PDF з версії 15. Програма працює під керуванням Microsoft Windows 7 або новішої версії та (без редагування PDF) Apple macOS 10.12 Sierra або новішої версії. Перша версія була випущена в 1993 році. Програма дозволяє конвертувати графічні документи (фотографії, скановані зображення, файли PDF) та знімки екрана у формати файлів, які можна редагувати, включаючи Microsoft Word, Microsoft Excel, Microsoft PowerPoint, формат Rich Text, HTML, PDF/A, PDF, CSV та txt з можливістю пошуку. (звичайний текст) файли. З 11 версії файли можна зберігати у форматі DjVu. Версія 15 підтримує розпізнавання тексту на 192 мовах світу і має вбудовану перевірку орфографії для 48 з них. FineReader розпізнає нові символи шляхом: навчання символів, щоб вони були додані до алфавіту розпізнавання; вибору додаткових символів зі списку та додавання їх до алфавіту вибраної мови (наприклад, додавання певних ісландських символів до німецького алфавіту для німецького тексту, що описує Ісландію); і додавання доменного словника до вбудованого лексикону FineReader. Програма також дозволяє користувачам порівнювати документи, додавати анотації і коментарі та планувати пакетну обробку [6].

Основні можливості:

1) **редактор форматів -** після завершення сканування документа можна редагувати формат і макет свого файлу. Також є можливість виправляти речення та коригувати абзаци;
2) **інструменти розмітки** - малюйте, коментуйте та анотуйте файли;
3) **наповнювач форм** - використовуйте ABBYY FineReader для створення форм, готових до заповнення або цифрового підпису.

Працює з Windows, Mac, iOS та Android.

1.2.3 Nanonets

Nanonets — це програмне забезпечення для розпізнавання символів на основі штучного інтелекту, яке автоматизує збір даних для інтелектуальної обробки документів рахунків-фактур, квитанцій, ідентифікаційних карток тощо. Nanonets використовує розширене OCR, обробку зображень машинного навчання та глибоке навчання для вилучення відповідної інформації з неструктурованих даних. Він швидкий, точний, простий у використанні, дозволяє користувачам створювати власні моделі OCR з нуля та має кілька акуратних інтеграцій Zapier. Оцифровуйте документи, витягуйте поля даних та інтегруйте їх із своїми повсякденними програмами за допомогою API у простому інтуїтивно зрозумілому інтерфейсі [7].

1) **вибіркова екстракція** - Nanonets можна навчити витягувати лише необхідні дані. Навіть з неструктурованими документами;
2) **широка інтеграція** - інтегрується з ERP, базами даних, а також службами хмарного сховища;
3) **безкоштовний OCR API** - швидкий час відповіді та необмежена кількість запитів.

Працює з Windows, Mac, iOS та Android.

1.2.4 Adobe Acrobat DC

Adobe Acrobat — це сімейство прикладного програмного забезпечення та вебсервісів, розроблених Adobe Inc. для перегляду, створення, керування, друку та керування файлами у форматі PDF (Portable Document Format). Сімейство включає Acrobat Reader (раніше Reader), Acrobat (раніше Exchange) і Acrobat.com. Базовий Acrobat Reader, доступний для кількох настільних і мобільних платформ, є безкоштовним; він підтримує перегляд, друк та анотування PDF-файлів.

Adobe Acrobat Export PDF підтримує оптичне розпізнавання символів або OCR, коли ви конвертуєте PDF-файл у Word (.doc і .docx), Excel (.xlsx) або RTF (формат розширеного тексту). OCR — це перетворення зображень тексту (сканованого тексту) у символи, які можна редагувати, щоб ви могли шукати,

виправляти та копіювати текст. Коли OCR увімкнено, Adobe Acrobat Export PDF виконує OCR для PDF-файлів, які містять зображення, векторну ілюстрацію, прихований текст або комбінацію цих елементів. (Наприклад, Adobe Acrobat Export PDF виконує розпізнавання файлів PDF, створених із відсканованих документів.) Adobe Acrobat Export PDF також виконує розпізнавання тексту, який він не може інтерпретувати, оскільки текст був неправильно закодований у вихідній програмі [8].

Основні можливості:

1) **розпізнавання форматування** - Adobe Acrobat DC достатньо інтуїтивно зрозумілий, щоб визначити, який шрифт спочатку використовувався в документі. Потім він перетворить текст у PDF-документ, використовуючи певний шрифт. Adobe Scan. Потім вони будуть перетворені в цифрові файли, які можна редагувати;
2) **сортування сторінок** - після створення PDF-файлу Adobe дозволяє змінювати порядок сторінок. Ви також можете розділити або об’єднати сторінки з інших документів.

Працює з Windows, Mac, iOS та Android.

1.2.5 Microsoft OneNote

Microsoft OneNote — це програма для створення нотаток для збору інформації у вільній формі та багатокористувацької співпраці. Він збирає: нотатки користувачів, малюнки, вирізки екрана та аудіокоментарі. Нотатки можна ділитися з іншими користувачами OneNote через Інтернет або мережу. OneNote доступний як частина пакету Microsoft Office, він також доступний у вигляді безкоштовної окремої програми на офіційному вебсайті та в магазинах додатків: Windows 10, MacOS, IOS та Android. Microsoft також надає вебверсію OneNote як частину OneDrive та Office для Інтернету. Microsoft OneNote має розширені функції OCR, які працюють як із зображеннями, так і з рукописними нотатками [9].

Microsoft Lens (раніше Office Lens) – це додаток для фотографування інформації з документів, дошок, візитних карток, квитанцій, меню, вивісок, рукописних заміщень та інших даних, що містять текст, який ви хочете імпортувати, але не вводити власноруч. При захопленні зображень Microsoft Lens видаляє тіні та неправильний ракурс, що спрощує перегляд остаточних результатів. Ви можете додати отриманий документ і зображення на дошку Microsoft OneNote, Word, PowerPoint або OneDrive, а також зберегти їх у форматі PDF або відправити електронною поштою [10].

Основні можливості OneNote:

1) **інструменти анотації** - використовуйте інструменти перо, введення або виділення, щоб допомогти коментувати наявні файли або відскановані документи. Використовуйте палець або стилус, сумісний з вашим планшетом, щоб робити нотатки на уроці;
2) **мультимедійна анотація** - вставляйте відеофайли у свої нотатки, завантажуйте вкладені файли або записуйте аудіонотатки;
3) **обріжте і збережіть** - використовуйте OneNote, щоб обрізати текст і зберегти його у своїх нотатках одним кліком;
4) **OneDrive -** скористайтеся перевагами інтеграції Microsoft з OneDrive, щоб захистити свої файли за допомогою виявлення програм-вимагачів та відновлення.

Працює з усіма пристроями.

1.2.6 IBM Datacap

IBM Datacap спрощує захоплення, розпізнавання та класифікацію ділових документів для вилучення з них важливої інформації. Його обробка природної мови, аналітика тексту та технології машинного навчання ідентифікують, класифікують та витягують вміст із неструктурованих або змінних паперових документів. Datacap має потужний механізм OCR, кілька функцій, а також

настроювані правила. Він працює через декілька каналів, включаючи сканери, мобільні пристрої, багатофункціональні периферійні пристрої та факс [11].

Основні можливості:

1) **багатоканальний вхід** - зберігайте редагований текст у документи з електронної пошти, факсу, PDF, зображень і сканерів;
2) **управління вмістом на основі ролей** - икористовуйте автоматичне редагування запитів на доступ. Доступ буде дозволено або заборонено залежно від ролі запитувача в компанії;
3) **розумне захоплення -** використовуючи машинне навчання, IBM Datacap може витягувати дані зі складних документів і впорядковувати їх у поля.

Працює з Windows, настільними комп’ютерами, вебпристроями, iOS, Android.

1.2.7 Amazon Textract

Amazon Textract дозволяє легко додавати визначення та аналіз тексту документа до ваших програм. За допомогою Amazon Textract клієнти можуть:

- виявляйти набраний та рукописний текст у різноманітних документах, включаючи фінансові звіти, медичні записи та податкові форми;
- витягувати текст, форми та таблиці з документів зі структурованими даними за допомогою Amazon Textract Document Analysis API;
- вказувати та витягувати інформацію з документів за допомогою функції запитів у Amazon Textract Analyze Document API;
- обробляти рахунки-фактури та квитанції за допомогою API AnalyzeExpense.

Amazon Textract включає прості у використанні API, які можуть аналізувати файли зображень і PDF-файли. Amazon Textract завжди навчається на нових даних, і Amazon постійно додає нові функції до служби [12].

Основні можливості:

1) **створення індексу інтелектуального пошуку** – за допомогою Amazon Textract ви можете створювати бібліотеки тексту, який виявляється в файлах зображень і PDF.
2) **використання інтелектуального вилучення тексту для обробки природної мови (NLP)** – Amazon Textract надає вам контроль над тим, як текст групується як вхідні дані для програм NLP. Він може витягувати текст у вигляді слів і рядків. Він також групує текст за клітинками таблиці, якщо увімкнений аналіз таблиці документів Amazon Textract;
3) **прискорення збору та нормалізації даних з різних джерел** – Amazon Textract дозволяє витягувати текстові та табличні дані з широкого спектру документів, таких як фінансові документи, звіти про дослідження та медичні записки. Завдяки API Amazon Textract Analyze Document ви можете легко та швидко витягувати неструктуровані та структуровані дані зі своїх документів;
4) **автоматизація збирання даних із форм** – Amazon Textract дозволяє витягувати структуровані дані з форм. За допомогою API Amazon Textract Analysis ви можете вбудувати можливості вилучення в існуючі бізнес-процеси, щоб дані користувачів, надіслані за допомогою форм, можна було витягувати у зручний для використання формат.

Працює з Windows, вебпристроями, Mac, Linux.

1.2.8 OnlineOCR

OnlineOCR — це безкоштовне програмне забезпечення, яке підтримує понад 40 мов. Вам не потрібно входити в систему або завантажувати будь-яку програму: просто перейдіть на сайт і завантажте файл, який ви хочете конвертувати. Сайт дозволяє конвертувати 15 файлів на годину без реєстрації облікового запису. Недоліком безкоштовного програмного забезпечення для розпізнавання тексту є те, що немає можливості редагувати текст на сайті, а метадані не можна витягти.

OnlineOCR робить речі неймовірно простими, дозволяючи конвертувати PDF-файли та відсканові PDF-файли в документи Word, тому є дуже мало специфічних функцій, які можна виділити.

1.3 Постановка задачі

1) Підібрати базу даних для тренування моделі.
2) Змоделювати нейронну мережу та навчити її на основі підібраної бази.
3) Створення графічного застосунку, за допомогою якого користувач зможе намалювати літеру та побачити результат прогнозування.
4) Тестування програми.

Висновки до розділу 1

Отже, під час виконання першого розділу було проаналізовано і поставлено задачу розпізнавання рукописних літер. Також було розглянуто сучасні аналоги розробки ПЗ для розпізнавання тексту.

2 МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ЗАДАЧІ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЛІТЕР

2.1 Методи для вирішення задачі розпізнавання образів

2.1.1 Лінійний дискримінантний аналіз (LDA)

Лінійний дискримінантний аналіз (LDA), нормальний дискримінантний аналіз (NDA) або аналіз дискримінантних функцій - це узагальнення лінійного дискримінанта Фішера, методу, який використовується в статистиці та інших областях, щоб знайти лінійну комбінацію ознак, яка характеризує або розділяє два або більше класів. предметів або подій. Отриману комбінацію використовують як лінійний класифікатор або, частіше, для зменшення розмірності перед подальшою класифікацією.

Для кожного класу $k \in \{1, \dots, K\}$ присвоюється апріорне $\hat{\pi}_k$ таке, що $\sum_{i=1}^{k} \hat{\pi}_k = 1$. За правилом Байєса апостеріорна ймовірність дорівнює:

$$\Pr(\mathrm{G} = \mathrm{k}|\mathrm{X} = \mathrm{x}) = \frac{f_k(x)\pi_k}{\sum_{i=1}^{k} f_1(x)\,\pi_1}, \tag{2.1}$$

де $f_k(x)$ – густина *X*, обумовлена *k*.

Максимальна апостеріорна оцінка спрощується до

$$G(x) = \arg\max_k \Pr(\mathrm{G} = \mathrm{k}|\mathrm{X} = \mathrm{x}) = \arg\max_k f_k(x)\pi_k. \tag{2.2}$$

LDA припускає, що густина є гауссовою:

$$f_k(x) = \left|2\pi \sum_k\right|^{-\frac{1}{2}} e^{-\frac{1}{2}\left(x-\mu_k\right)^T \Sigma_k^{-1}\left(x-\mu_k\right)}, \tag{2.3}$$

де $\sum_k$ – коваріаційна матриця для вибірок із класу *k*;

|.| – детермінант.

LDA передбачає, що всі класи мають однакову коваріаційну матрицю, тобто $\sum_k = \sum, \forall k$.

Підставляючи f_k до функції класифікації, ми приходимо до функції класифікації:

$$G(x) = \arg\max_k \delta_k(x) \tag{2.4}$$

де

$$\delta_k(x) = x^T \sum^{-1} \mu_k - \frac{1}{2} \mu_k^T \sum^{-1} \mu_k + \log \pi_k. \tag{2.5}$$

є дискримінантною функцією для класу *k*.

Отже, тепер, коли у нас є класифікатор, ми можемо його обчислити.

Щоб знайти коваріаційну матрицю, ми просто обчислюємо

$$\hat{\Sigma} = \sum_{k=1}^{K} \frac{1}{N-K} \sum_{g_i=k} \left(x_i - \hat{\mu}_k\right)\left(x_i - \hat{\mu}_k\right)^T. \tag{2.6}$$

Зверніть увагу, що відхилення від середнього ділиться на *N−K*, ступені свободи, щоб отримати незміщену оцінку. Середні значення класів, які також називають центроїдами, визначаються за допомогою

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{g_i=k} x_i. \tag{2.7}$$

Апріорні πk встановлюються як коефіцієнт поширеності спостережень, специфічних для класу:

$$\hat{\pi}_k = \frac{N_k}{N}. \tag{2.8}$$

Таким чином, ми визначили всі параметри, необхідні для класифікатора.

Процедура зменшення розмірності LDA включає дисперсію всередині класу $W = \hat{\sum}$, і дисперсію між класами *B*. Дисперсія між класами вказує на відхилення центроїдів від загального середнього $\hat{\mu} = \sum_{k=1}^{K} \hat{\pi}_k \hat{\mu}_k$, і визначається як:

$$B = \sum_{k=1}^{K} \hat{\pi}_k (\hat{\mu}_k - \hat{\mu})(\hat{\mu}_k - \hat{\mu})^T. \tag{2.9}$$

Пошук послідовності оптимальних підкроків включає три кроки:

1. обчислення $K \times p$ матриці M, що містить центроїди μ_k, і визначте загальну матрицю коваріації W;
2. обчислення $M^* = MW^{-\frac{1}{2}}$ використовуючи власний розклад W;
3. обчислення B^* (коваріацію між класами) та власного розкладу $B^* = V^* D_B V^{*T}$. Стовпці v_l^* у V^* визначають координати зменшеного підпростору.

Дискримінантна змінна l (один із нових розмірів $K-1$) визначається як

$$Z_l = v_l^T X, \tag{2.10}$$

де $v_l = W^{-\frac{1}{2}} v_l^*$.

LDA має такі властивості:

1. LDA припускає, що дані є гауссовими. Точніше, передбачається, що всі класи мають однакову коваріаційну матрицю;
2. LDA знаходить лінійні межі рішення в $K-1$ розмірному підпросторі. Таким чином, він не підходить, якщо між незалежними змінними існують взаємодії вищого порядку;
3. LDA добре підходить для вирішення проблем із кількома класами, але його слід використовувати обережно, коли розподіл класів є незбалансованим, оскільки пріоритети оцінюються на основі спостережуваних підрахунків. Таким чином, спостереження рідко будуть віднесені до нечастих класів;
4. LDA можна використовувати як метод зменшення розмірності.

2.1.2 Квадратичний дискримінантний аналіз (QDA)

У статистиці квадратичний класифікатор — це статистичний класифікатор, який використовує квадратичну поверхню прийняття рішень для розділення вимірювань двох або більше класів об’єктів або подій. Це більш загальна версія лінійного класифікатора. Квадратичний дискримінантний аналіз (QDA) тісно пов'язаний з лінійним дискримінантним аналізом (LDA), де передбачається, що вимірювання кожного класу розподілені нормально. QDA – це варіант LDA, в

якому для кожного класу спостережень оцінюється індивідуальна матриця коваріації. QDA особливо корисний, якщо є попередні знання про те, що окремі класи демонструють різні коваріації. Недоліком QDA є те, що його не можна використовувати як метод зменшення розмірності. У QDA нам потрібно оцінити $\sum_k$ для кожного класу $k \in \{1, \dots, K\}$, а не припускати $\sum_k = \sum$, як в LDA.

Дискримінантна функція LDA є квадратичною по *x*:

$$\delta_k = -\frac{1}{2}\log|\sum_k| - \frac{1}{2}(x-\mu_k)^T\sum_k^{-1}(x-\mu_k) + \log\pi_k. \quad (2.11)$$

Оскільки QDA оцінює коваріаційну матрицю для кожного класу, вона має більшу кількість ефективних параметрів, ніж LDA. Кількість параметрів можна отримати наступним чином.

- Нам потрібні пріори πk класу *K*. Оскільки $\sum_{k=1}^{K}\pi_k = 1$, нам не потрібен параметр для одного з пріорів. Таким чином, існує *K−1* вільних параметрів для пріорів.
- Оскільки є *K* центроїдів μ_k, кожен з яких має p записів, є параметри K_p, що відносяться середніх.
- З коваріаційної матриці $\sum_k$ нам потрібно розглянути лише діагональ і верхній правий трикутник. Ця область коваріаційної матриці має $\frac{p(p+1)}{2}$ елементів. Оскільки *K* таких матриць необхідно оцінити, існує $K\frac{p(p+1)}{2}$ параметрів, що відносяться до коваріаційних матриць.

Таким чином, ефективна кількість параметрів QDA дорівнює

$$K - 1 + K_p + K\frac{p(p+1)}{2}. \quad (2.12)$$

Оскільки кількість параметрів QDA є квадратичною в p, QDA слід використовувати обережно, коли простір функцій великий [13].

2.1.3 Наївний байєсів класифікатор

Наївний байєсів класифікатор — це імовірнісна модель машинного навчання, яка використовується для завдання класифікації [14]. Суть класифікатора заснована на теоремі Байєса:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \quad (2.13)$$

Використовуючи теорему Байєса, ми можемо знайти ймовірність того, що A станеться, враховуючи, що сталося B. Тут B — це доказ, а A — гіпотеза. Припущення, зроблене тут, полягає в тому, що предиктори/характеристики є незалежними. Тобто наявність однієї особливої ознаки не впливає на іншу. Тому його називають наївним. Розглянемо як приклад такий набір даних (рис 2.1):

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

Рисунок 2.1 – Набір даних для рішення задачі класифікації за допомогою наївного байєсового класифікатора

Класифікуємо, чи підходить день для гри в гольф, враховуючи особливості дня. Стовпці представляють ці функції, а рядки – окремі записи. Якщо ми візьмемо перший рядок набору даних, ми можемо помітити, що він не підходить для гри в гольф, якщо очікується дощ, висока температура, висока вологість і не вітер. Тут ми робимо два припущення, одне, як зазначено вище, ми вважаємо, що ці предиктори є незалежними. Тобто, якщо температура спекотна, це не обов'язково означає, що вологість висока. Інше припущення, зроблене тут, полягає в тому, що всі предиктори однаково впливають на результат. Тобто вітряний день не має більшого значення для рішення грати в гольф чи ні. Відповідно до цього прикладу теорему Байєса можна переписати так:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}, \tag{2.14}$$

де *y* – це змінна класу (грати в гольф), яка вказує, чи підходить день для гри в гольф, чи ні з урахуванням умов.

X – представляє параметри/характеристики.

X задається як,

$$X = (x_1, x_2, x_3, \dots, x_n), \tag{2.15}$$

де $x_1, x_2, x_3, \dots, x_n$ представляють ознаки, тобто їх можна відобразити на хмарність, температуру, вологість та вітряність.

Підставляючи X і розширюючи за допомогою правила ланцюга, отримуємо:

$$P(y|x_1, x_2, x_3, \dots, x_n) = \frac{P(x_1|y)P(x_2|y) \dots P(x_n|y)}{P(x_1)P(x_2) \dots P(x_n)}. \tag{2.16}$$

Тепер ми можемо отримати значення для кожного запису, подивившись набір даних і підставивши їх у рівняння. Для всіх записів у наборі даних знаменник не змінюється, він залишається статичним. Отже, знаменник можна прибрати і ввести пропорційність:

$$P(y|x_1, x_2, x_3, \dots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y). \tag{2.17}$$

У нашому випадку змінна класу (***y***) має лише два результати, так чи ні. Можуть бути випадки, коли класифікація багатоваріантна. Отже, нам потрібно знайти клас ***y*** з максимальною ймовірністю. Враховуючи предиктори:

$$y = \arg max_y P(y) \prod_{i=1}^{n} P(x_i|y). \quad (2.18)$$

2.1.4 Перцептрон

У машинному навчанні перцептрон — це алгоритм контрольованого навчання бінарних класифікаторів. Двійковий класифікатор - це функція, яка може вирішити, чи належить вхід, представлений вектором чисел, до певного класу. Це тип лінійного класифікатора, тобто алгоритм класифікації, який робить свої прогнози на основі функції лінійного предиктора, що поєднує набір вагових показників з вектором ознак. На рис. 2.2 наведена логічна структура елементарного перцептрону. Тут ваги зв'язків S-A можуть мати значення −1, 1 або 0 (тобто відсутність зв'язку), а ваги зв'язків A-R можуть мати будь-яке значення.

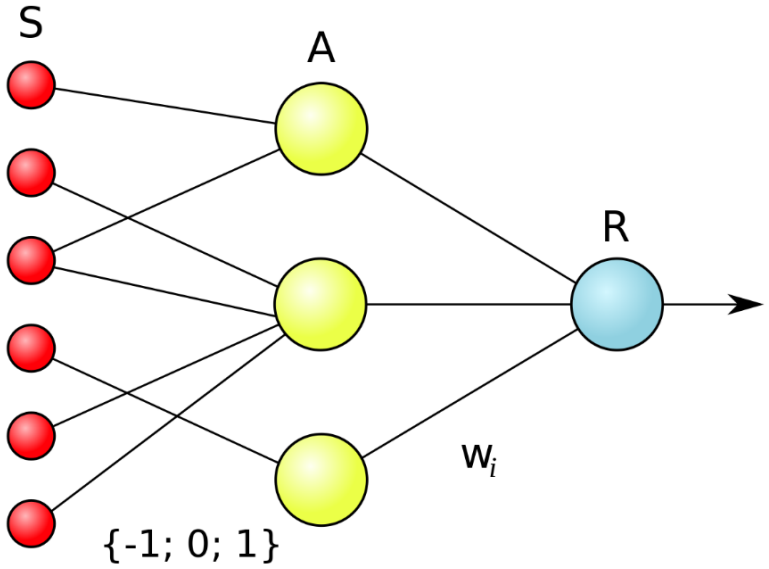

Рисунок 2.2 – Логічна схема елементарного перцептрону

Елементарний перцептрон включає в себе елементи трьох типів: S-елементи, A-елементи та однин R-елемент. S-елементи — це шар сенсорів (рецепторів). У фізичному втіленні вони можуть відповідати, наприклад, світлочутливим клітинам сітківки ока або фоторезисторам матриці камери. Кожен рецептор може перебувати

тільки в одному з двох станів — спокою чи збудження, і лише у другому випадку він передає одиничний сигнал до наступного шару A-елементів.

A-елементи називають асоціативними, тому що кожному такому елементу, як правило, відповідає цілий набір S-елементів. A-елемент активізується в той момент, коли кількість сигналів від шару сенсорів на його вході більша за певну величину θ.

Сигнали від збуджених A-елементів, своєю чергою, передаються до суматора R, при цьому сигнал від i-го A-елемента передається з коефіцієнтом ω_i – вагою A-R зв'язку.

Так само як і A-елементи, R-елемент розраховує суму значень вхідних сигналів, помножених на ваги (лінійну форму). R-елемент та елементарний перцептрон, видають “1”, якщо лінійна форма перевищує поріг θ, в іншому випадку на виході буде “-1”. Функцію, що реалізує R-елемент, можна математично записати так:

$$f(x) = sign\left(\sum_{i=1}^{n} \omega_i x_i - \theta\right). \quad (2.19)$$

2.1.5 Згорткова нейронна мережа (CNN)

До появи CNN для ідентифікації об’єктів на зображеннях використовувалися ручні, трудомісткі методи вилучення ознак. Однак згорткові нейронні мережі тепер забезпечують більш масштабований підхід до завдань класифікації зображень і розпізнавання об’єктів, використовуючи принципи лінійної алгебри, зокрема, множення матриці, для виявлення шаблонів у зображенні. Тим не менш, вони можуть бути вимогливими до обчислень, вимагаючи графічних процесорів (GPU) для навчання моделей. CNN використовують різновид багатошарових перцептронів, розроблений так, щоби вимагати використання мінімального обсягу попередньої обробки. Згорткові нейронні мережі відрізняються від інших

нейронних мереж їх високою продуктивністю з зображенням, мовленням або звуковим сигналом [15].

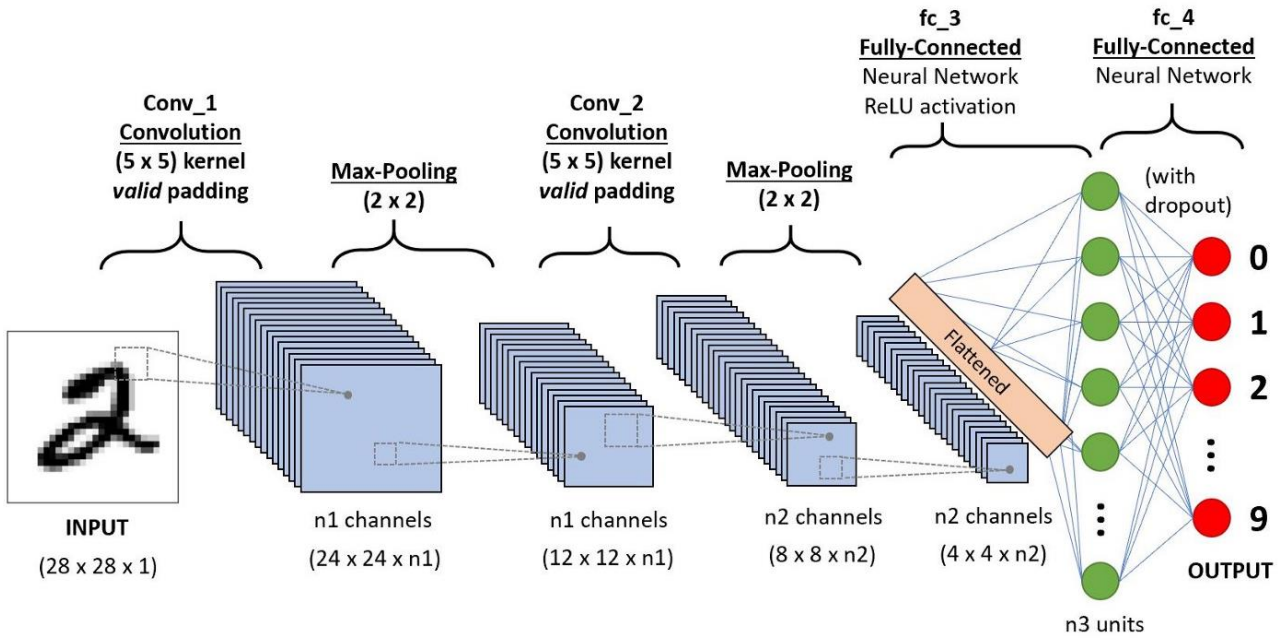

Рисунок 2.3 – Приклад структури CNN для класифікації рукописних цифр

CNN мають три основних типи шарів, а саме:

1) **згортковий шар**;

Згортковий шар є основним будівельним блоком CNN, і саме там відбувається більшість обчислень. Для цього потрібно кілька компонентів, а саме вхідні дані, фільтр і карта ознак. Припустимо, що вхідним буде кольорове зображення, яке складається з матриці пікселів у 3D. Це означає, що вхідні дані будуть мати три виміри — висоту, ширину та глибину — які відповідають RGB зображенню. У нас також є детектор ознак, також відомий як ядро або фільтр, який буде переміщатися по сприйнятливих полях зображення, перевіряючи, чи присутня ознака. Цей процес відомий як згортка.

Детектор ознак — це двовимірний (2D) масив ваг, який представляє частину зображення. Хоча вони можуть відрізнятися за розміром, розмір фільтра зазвичай є матрицею 3x3; це також визначає розмір рецептивного поля. Потім фільтр застосовується до області зображення, і між вхідними пікселями та фільтром обчислюється скалярний добуток. Цей добуток потім подається у вихідний масив.

Після цього фільтр зміщується на один крок, повторюючи процес, поки ядро не охопить усе зображення. Остаточний результат із ряду скалярних добутків від входу та фільтра відомий як карта ознак, карта активації або згорнута функція. Кожне вихідне значення на карті об’єктів не повинно з’єднуватися з кожним значенням пікселя у вхідному зображенні. Його потрібно лише підключитися до сприйнятливого поля, де застосовується фільтр. Оскільки вихідний масив не потребує безпосереднього відображення кожного вхідного значення, згорткові (і об’єднані) шари зазвичай називають «частково зв’язаними» шарами. Однак цю характеристику можна також описати як локальну зв’язність.

Зауважимо, що ваги в детекторі ознак залишаються фіксованими, коли він рухається по зображенню, що також відоме як спільне використання параметрів. Деякі параметри, як-от значення ваги, коригуються під час навчання за допомогою процесу зворотного поширення та градієнтного спуску. Проте є три гіперпараметри, які впливають на розмір виводу, який необхідно встановити перед початком навчання нейронної мережі. До них належать:

1. **кількість фільтрів** впливає на глибину виводу. Наприклад, три різні фільтри дадуть три різні карти об’єктів, створюючи глибину третього рівня;
2. **крок** – це відстань або кількість пікселів, на які ядро переміщається по вхідній матриці. Хоча значення кроку два або більше зустрічається рідко, більший крок дає менший результат;
3. **нульове заповнення** зазвичай використовується, коли фільтри не підходять до вхідного зображення. Це обнуляє всі елементи, які виходять за межі вхідної матриці, створюючи більший або однаковий за розміром вивід. Існує три типи заповнення:
    1. дійсне заповнення: це також відоме як відсутність заповнення. У цьому випадку остання згортка викидається, якщо розміри не співпадають;

2. ідентичне заповнення: це заповнення гарантує, що вихідний шар має той самий розмір, що і вхідний шар;
3. повне заповнення: цей тип заповнення збільшує розмір виводу, додаючи нулі до межі вхідних даних.

Після кожної операції згортки CNN застосовує перетворення зрізаного лінійного вузла (ReLU) до карти ознак, вносячи нелінійність у модель. Як ми згадували раніше, інший шар згортки може слідувати за початковим шаром згортки. В цьому випадку структура CNN може стати ієрархічною, оскільки пізні рівні можуть бачити пікселі в сприйнятливих полях попередніх шарів. Як приклад, припустимо, що ми намагаємося визначити, чи містить зображення велосипед. Ви можете думати про велосипед як про суму частин. Він складається з рами, керма, коліс, педалей тощо. Кожна окрема частина велосипеда складає шаблон нижнього рівня в нейронній мережі, а комбінація його частин представляє шаблон вищого рівня, створюючи ієрархію ознак у CNN (рис. 2.4).

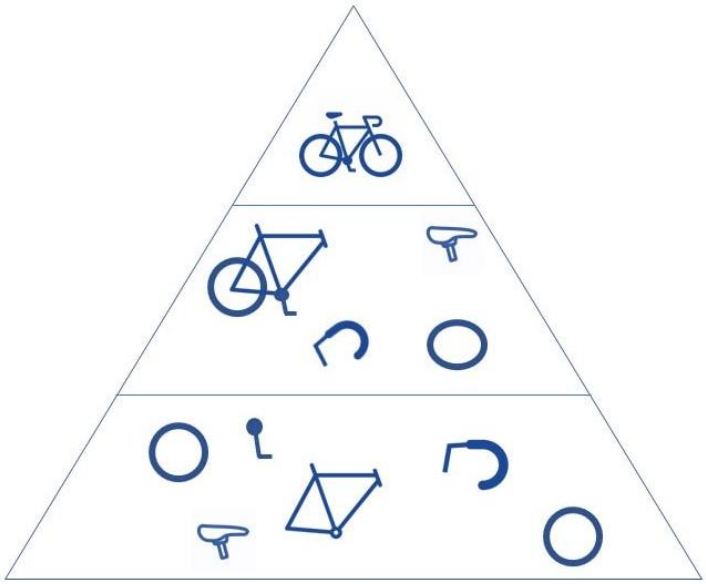

Рисунок 2.4 – Діаграма ієрархії ознак у згорткових нейронних мережах на прикладі велосипеда

Зрештою, згортковий шар перетворює зображення в числові значення, дозволяючи нейронній мережі інтерпретувати та витягувати відповідні шаблони.

2) **об’єднуючий шар**;

Об’єднуючі шари, також відомі як децимація (зниження дискретизації), зменшує розмірність, зменшуючи кількість параметрів у вхідних даних. Подібно до згорткового шару, операція об’єднання об’єднує фільтр по всьому входу, але різниця в тому, що цей фільтр не має жодних ваг. Замість цього ядро застосовує функцію агрегації до значень у сприйнятливому полі, заповнюючи вихідний масив. Існує два основних типи об’єднання:

1. максимальне об’єднання - під час переміщення фільтра по входу фільтр вибирає піксель з максимальним значенням для відправки на вихідний масив. Крім того, цей підхід, як правило, використовується частіше в порівнянні із середнім об’єднанням;
2. середнє об’єднання - коли фільтр переміщується по входу, він обчислює середнє значення в сприйнятливому полі для надсилання до вихідного масиву.

Хоча багато інформації втрачається на рівні об’єднання, це також має ряд переваг для CNN. Ці шари допомагають зменшити складність, підвищити ефективність та обмежити ризик перенавчання.

3) **повнозв'язний шар**.

Назва повнозв'язного шару влучно описує себе. Як згадувалося раніше, значення пікселів вхідного зображення не пов’язані безпосередньо із вихідним шаром у частково з’єднаних шарах. Однак у повнозв'язному шарі кожен вузол вихідного шару підключається безпосередньо до вузла попереднього шару.

Цей шар виконує завдання класифікації на основі ознак, витягнутих через попередні шари та їх різні фільтри. У той час як рівні згортки та об’єднання, як правило, використовують функції ReLU, рівні повнозв'язних шарів зазвичай використовують функцію активації softmax для належної класифікації вхідних даних, створюючи ймовірність від 0 до 1.

2.2 Технології для розробки системи

Процес виконання цієї роботи поділяється на дві основні частини: власне створення моделі та її використання на вебсайті. Для кожної із цих частин було виділено підзадачі та обрані технології.

1. **Модель нейронної мережі.**

– Мова програмування – **Python**.

Python – це мова комп'ютерного програмування, яка часто використовується для створення веб-сайтів і програмного забезпечення, автоматизації завдань і аналізу даних. Python — це мова загального призначення, що означає, що його можна використовувати для створення різноманітних програм і не спеціалізується на будь-яких конкретних проблемах. Ця універсальність разом із зручністю для початківців зробили її однією з найбільш використовуваних мов програмування сьогодні. Опитування, проведене аналітичною фірмою RedMonk, показало, що це друга за популярністю мова програмування серед розробників у 2021 році [16].

– Середовище – **Jupyter** (хмарне середовище розробки).

Jupyter Notebook (раніше IPython Notebooks) - це веб-інтерактивне обчислювальне середовище для створення документів для ноутбуків Jupyter. Термін «блокнот» може в розмовній формі посилатися на багато різних сутностей, головним чином на веб-додаток Jupyter, веб-сервер Jupyter Python або формат документа Jupyter в залежності від контексту. Документ Jupyter Notebook - це документ JSON, який слідує за версійною схемою, що містить упорядкований список клітинок вводу / виводу, який може містити код, текст (із використанням Markdown), математику, графіки та мультимедійні файли, як правило, закінчуючись розширенням ".ipynb". Блокнот Jupyter можна перетворити на безліч відкритих стандартних форматів виводу (HTML, слайди презентацій, LaTeX, PDF, ReStructuredText, Markdown, Python) за допомогою "Завантажити як" у веб-інтерфейсі за допомогою бібліотеки nbconvert або команди "jupyter nbconvert" лінійного інтерфейсу в оболонці. Для спрощення візуалізації документів ноутбуків

Jupyter в Інтернеті, бібліотека nbconvert надається як послуга через NbViewer, яка може взяти URL-адресу до будь-якого загальнодоступного документа блокнота, перетворити його в HTML на льоту і показати користувачеві.

– Аналіз даних – **Matplotlib** (візуалізація даних).

Matplotlib — це повна бібліотека для створення статичних, анімованих та інтерактивних візуалізацій на Python. Matplotlib надає такі можливості користувачу:

- створюйте графіки якості публікації;
- зробіть інтерактивні фігури, які можна масштабувати, панорамувати, оновлювати;
- налаштуйте візуальний стиль і макет;
- експортувати в багато форматів файлів;
- вбудовувати в JupyterLab і графічні інтерфейси користувача;
- використовуйте багатий набір пакетів сторонніх розробників, створених на Matplotlib.

– Створення та тренування моделі – **Keras**.

Keras — це бібліотека програмного забезпечення з відкритим вихідним кодом, яка надає інтерфейс Python для штучних нейронних мереж. Keras виступає в якості інтерфейсу для бібліотеки TensorFlow. Keras містить численні реалізації типових будівельних блоків нейронної мережі, таких як шари, цілі, функції активації, оптимізатори та безліч інструментів, які полегшують роботу з графічними та текстовими даними, щоб спростити кодування, необхідне для написання глибокого коду нейронної мережі. Код розміщено на GitHub, а форуми підтримки спільноти включають сторінку питань GitHub і канал Slack.

На додаток до стандартних нейронних мереж, Keras підтримує згорткові та рекурентні нейронні мережі. Він підтримує інші загальні рівні утиліти, такі як вилучення, нормалізація пакетів і об'єднання.

Keras дозволяє користувачам створювати глибокі моделі на смартфонах (iOS та Android), в Інтернеті або на віртуальній машині Java. Це також дозволяє використовувати розподілене навчання моделей глибокого навчання на кластерах графічних процесорів (GPU) і тензорних процесорів (TPU).

– Тренувальні дані – **бібліотека NIST**.

Спеціальна база даних 19 друкованих форм і символів NIST містить весь корпус навчальних матеріалів NIST для розпізнавання документів і символів від руки. Він публікує віддруковані від руки зразки форм від 3600 авторів, 810 000 зображень символів, виділених із їхніх форм, основні класифікації істинності цих зображень, довідкові форми для подальшого збору даних та програмні утиліти для керування зображеннями та їх обробки.

Особливості цієї бази даних:

- остаточне накопичення зразків даних, надрукованих від руки NIST;
- повносторінкові форми HSF з 3600 авторів;
- окремі поля для цифр, верхнього та нижнього регістрів, а також поля вільного тексту;
- понад 800 000 зображень із перевіреними вручну класифікаціями.

База даних є найбільшим і, ймовірно, остаточним випуском зображень NIST, призначених для обробки документів від руки та дослідження OCR [17].

2. Вебзастосунок.

– Фреймворк – **Angular 6**.

Angular — це платформа та фреймворк для створення односторінкових клієнтських програм за допомогою HTML і TypeScript. Він реалізує основні та додаткові функції у вигляді набору бібліотек TypeScript, які ви імпортуєте у свої програми.

Архітектура програми Angular спирається на певні фундаментальні концепції. Основними будівельними блоками фреймворка є компоненти Angular, що організовані у NgModules. NgModules збирає пов'язаний код у функціональні

набори; Angular додаток визначається набором NgModules. Додаток завжди має хоча б кореневий модуль, що дозволяє завантажувати, і також зазвичай має набагато більше модулів функцій.

Компоненти визначають представлення, що являють собою набори елементів екрану, які Angular може вибирати та змінювати відповідно до логіки та даних програми.

Компоненти використовують служби, які надають конкретні функції що не пов’язані безпосередньо з представленнями. Постачальники послуг можуть бути введені в компоненти як залежності, що робить код модульним, багаторазовим та ефективним.

Модулі, компоненти та служби — це класи, які використовують декоратори. Ці декоратори позначають свій тип і надають метадані, які повідомляють Angular як їх треба використовувати.

Метадані для класу компонента пов’язують його з шаблоном, який визначає представлення. Шаблон поєднує звичайний HTML з директивами Angular та розміткою прив’язки, які дозволяють Angular змінювати HTML перед обробкою його для відображення.

Метадані для класу служби надають інформацію, необхідну Angular, щоб зробити її доступною для компонентів за допомогою ін'єкції залежностей.

Компоненти зазвичай визначають багато представлень, розташованих ієрархічно. Angular надає службу Router, щоб допомогти вам визначити шляхи навігації між представленнями [18].

– Середовище розробки – **Visual Studio Code**.

Visual Studio Code — це легкий, але дуже потужний редактор вихідного коду, який працює на ПК та доступний для Windows, macOS та Linux. Він поставляється з вбудованою підтримкою JavaScript, TypeScript і Node.js і має багату екосистему розширень для інших мов (таких як C++, C#, Java, Python, PHP) і середовищ виконання (наприклад, .NET і Unity). Функції включають підтримку налагодження,

виділення синтаксису, інтелектуальне завершення коду IntelliSense, фрагменти, рефакторинг коду та вбудований Git. Користувачі можуть змінювати тему, комбінації клавіш та параметри.

– Робота з моделлю – Tensorflow.js.

TensorFlow служить основною платформою та бібліотекою для машинного навчання. API TensorFlow використовує Keras, щоб дозволити користувачам створювати власні моделі машинного навчання. Окрім створення та навчання моделі, TensorFlow також може допомогти завантажити дані для навчання моделі та розгорнути її за допомогою TensorFlow Serving.

TensorFlow забезпечує стабільний API Python, а також API без гарантії зворотної сумісності для JavaScript, C++ та Java. Пакети прив’язки сторонніх розробників також доступні для C#, Haskell, Julia, MATLAB, R, Scala, Rust.

Висновки до розділу 2

Отже, під час виконання другого розділу було розглянуто та проаналізовано методи і технології, які можуть бути застосовані для вирішення поставленої задачі розпізнавання рукописних літер. Обрана структура нейронної мережі – згорткова нейронна мережа (CNN).

3 АНАЛІЗ ДАНИХ, СТВОРЕННЯ ТА ТРЕНУВАННЯ МОДЕЛІ

3.1 Підготовка даних

3.1.1 Імпорт базових бібліотек

Імпортуємо такі бібліотеки:

– numpy - обробка масивів;

– pandas – зчитування даних;

– matplotlib - візуалізація даних;

– sklearn – робота з моделлю;

– keras - тренування моделі;

– zipfile – робота з архівами та їх вмістом.

Процес наведено на рис. 3.1.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Flatten, Conv2D, MaxPool2D, Dropout
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from keras.utils import np_utils, to_categorical
from keras.optimizers import SGD, Adam
from sklearn.model_selection import train_test_split
from sklearn.utils import shuffle
import zipfile
```

Рисунок 3.1 – Імпорт бібліотек

3.1.2 Отримання та візуалізація даних

Тепер завантажуємо набір даних за допомогою pd.read_csv(), з яким ми будемо працювати: датасет із 60,000 зображень літер від A до Z розміром 28x28, а також тестова вибірка із 10,000 зображень (рис. 3.2). Зчитуємо набір даних і друкуємо перші 10 зображень за допомогою data.head(10).

```
# open zipped dataset
with zipfile.ZipFile("A_Z Handwritten Data.zip") as z:
    # open the csv file in the dataset
    with z.open("A_Z Handwritten Data.csv") as f:

        # read the dataset
        data = pd.read_csv(f).astype('float32')

print(data.head(10))
```

✓ 15.5s

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	...	0.639	0.640	0.641	\
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	

Рисунок 3.2 – Завантаження даних

3.1.3 Розділення даних на зображення та їх мітки

«0» містить мітки, і тому ми скидаємо стовпець «0» із зчитуваного фрейму даних і використовуємо його в ***y*** для формування міток (рис. 3.3).

```
# Split data the X - Our data , and y - the prdict label
X = data.drop('0', axis=1)
y = data['0']
```

Рисунок 3.3 – Розбиття зчитуваних даних на зображення та відповідні мітки

3.1.4 Перетворення даних у файлі csv, щоб їх можна було візуалізувати

На рис. 3.4 ми поділяємо дані на набір даних для навчання та тестування за допомогою train_test_split(). Крім того, ми змінюємо форму даних зображень для тренування та тестування, щоб їх можна було відображати як зображення, оскільки

спочатку у файлі CSV вони були представлені як 784 стовпці піксельних даних. Тож ми конвертуємо його в 28×28 пікселів.

```
# Reshaping the data in csv file so that it can be displayed as an image...

train_x, test_x, train_y, test_y = train_test_split(X, y, test_size=0.2)
train_x = np.reshape(train_x.values, (train_x.shape[0], 28, 28))
test_x = np.reshape(test_x.values, (test_x.shape[0], 28, 28))


print("Train data shape: ", train_x.shape)
print("Test data shape: ", test_x.shape)

✓ 1.5s
Train data shape:  (297960, 28, 28)
Test data shape:  (74490, 28, 28)
```

Рисунок 3.4 – Розбиття даних для навчання та тестування

Усі мітки представлені у вигляді значень з плаваючою крапкою, які ми перетворюємо на цілі значення, і тому створюємо словник word_dict , щоб зіставити цілі значення з символами (рис. 3.5).

```
# Dictionary for getting characters from index values...
word_dict = {0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5: 'F', 6: 'G',
             7: 'H', 8: 'I', 9: 'J', 10: 'K', 11: 'L', 12: 'M', 13: 'N',
             14: 'O', 15: 'P', 16: 'Q', 17: 'R', 18: 'S', 19: 'T', 20: 'U',
             21: 'V', 22: 'W', 23: 'X', 24: 'Y', 25: 'Z'}
```

Рисунок 3.5 – Створення словника для отримання символів із значень індексів

3.1.5 Побудова графіка кількості алфавітів у наборі даних

Спочатку ми перетворюємо мітки в цілі значення і додаємо їх до списку лічильників відповідно до мітки. Цей список містить кількість зображень, присутніх у наборі даних, що належать до кожної літери. Тепер ми створюємо список – літер, що містять усі символи, використовуючи функцію .values() словника (рис. 3.6). Тепер, використовуючи списки лічильників та літер, ми малюємо горизонтальну смугу (рис. 3.7).

```
y_int = np.int0(y)
count = np.zeros(26, dtype='int')
for i in y_int:
    count[i] +=1
alphabets = []
for i in word_dict.values():
    alphabets.append(i)
fig, ax = plt.subplots(1,1, figsize=(10,10))
ax.barh(alphabets, count)
plt.xlabel("Number of elements ")
plt.ylabel("Alphabets")
plt.grid()
plt.show()
```

Рисунок 3.6 – Процес побудови графіка кількості літер в наборі даних

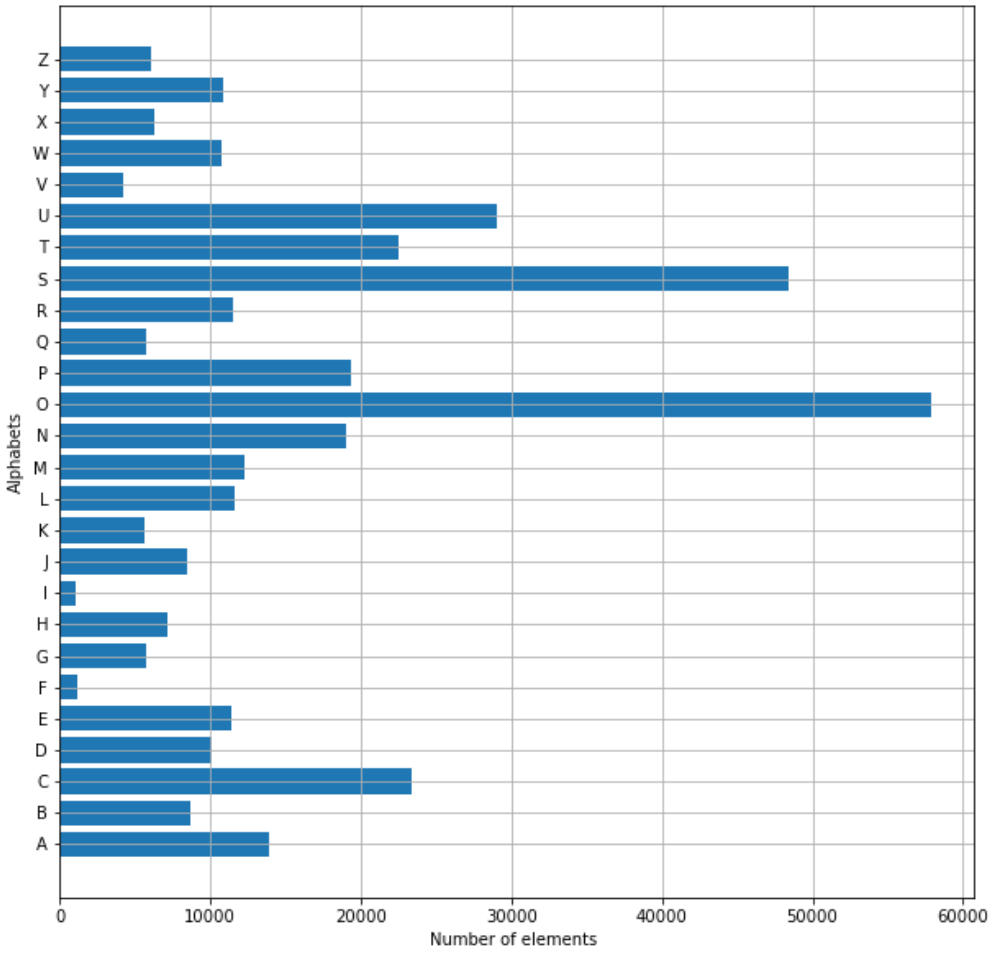

Рисунок 3.7 – Графік кількості літер в наборі даних

3.1.6 Перемішування даних

Тепер ми перемішаємо зображення з набору тренувальних даних (рис. 3.8). Перемішування виконується за допомогою функції .shuffle(), щоб ми могли відобразити деякі випадкові зображення. Потім ми створюємо 9 графіків у формі 3×3 і показуємо зображення 9 літер (рис. 3.9).

```
# Shuffling the data ...
shuff = shuffle(train_x[:100])

fig, ax = plt.subplots(3, 3, figsize=(10, 10))
axes = ax.flatten()

for i in range(9):
    axes[i].imshow(np.reshape(shuff[i], (28, 28)), cmap="Greys")
plt.show()
```

Рисунок 3.8 – Перемішування даних

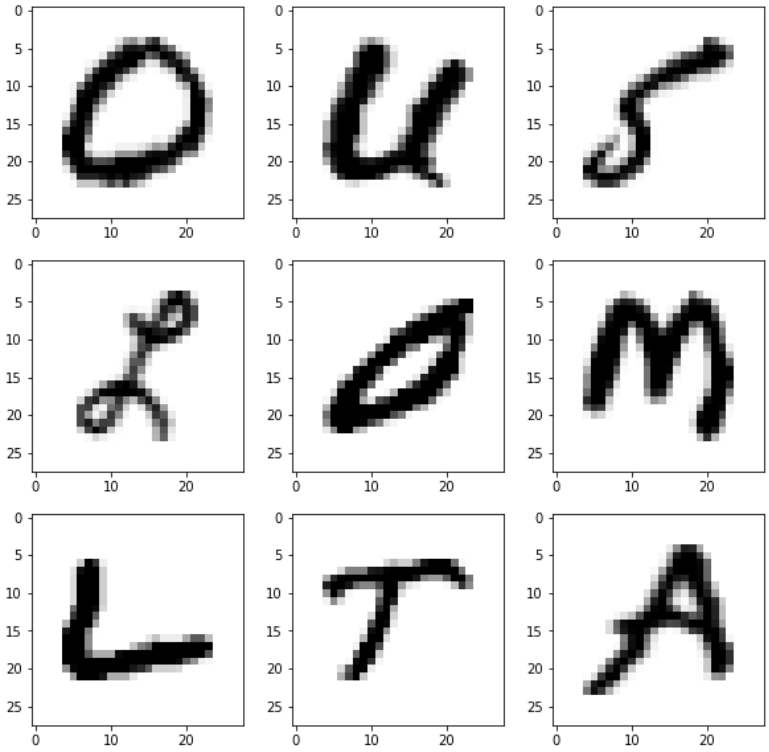

Рисунок 3.9 – Відображення випадкових даних з набору

Як ми бачимо, набір даних NIST зберігає тільки чорно-білі (grayscale) зображення.

3.1.7 Перетворення даних

Нам треба перетворити навчальний та тестовий набори даних, щоб їх можна було помістити в модель. Зробимо це за допомогою функції .reshape() (рис. 3.10).

```
# Reshaping the training & test dataset so that it can be put in the model...

train_X = train_x.reshape(
    train_x.shape[0], train_x.shape[1], train_x.shape[2], 1)
print("New shape of train data: ", train_X.shape)

test_X = test_x.reshape(test_x.shape[0], test_x.shape[1], test_x.shape[2], 1)
print("New shape of train data: ", test_X.shape)
```

✓ 0.1s

```
New shape of train data:  (297960, 28, 28, 1)
New shape of train data:  (74490, 28, 28, 1)
```

Рисунок 3.10 – Перетворення навчального та тестового наборів даних

На рис. 3.11 ми перетворюємо одиничні значення з плаваючою крапкою в категоріальні значення. Це робиться, оскільки модель CNN приймає вхідні дані міток і генерує вихід у вигляді вектору ймовірностей.

```
train_yOHE = to_categorical(train_y, num_classes=26, dtype='int')
print("New shape of train labels: ", train_yOHE.shape)

test_yOHE = to_categorical(test_y, num_classes=26, dtype='int')
print("New shape of test labels: ", test_yOHE.shape)
```

✓ 0.6s

```
New shape of train labels:  (297960, 26)
New shape of test labels:  (74490, 26)
```

Рисунок 3.11 – Перетворення одиничних значень з плаваючою крапкою в категоріальні значення

3.2 Створення структури нейромережі

За шарами згортки зазвичай слідують шари maxpool, які використовуються для зменшення кількості вилучених функцій, і зрештою вихідні дані maxpool, а також шари та шари згортки зводяться до одновимірного вектору та надаються як вхідні дані для щільного шару. Для цього використовуємо бібліотеку Keras. Для цієї архітектури ми можемо використовувати послідовну модель (Sequential), до якої ми будемо додавати шари методом .add().

До кожного прихованого шару ми додаємо активаційну функцію для того, щоб при тренуванні модель автоматично використовувала метод **зворотного розповсюдження помилки** (англ. **backpropagation**) для підрахування градієнту при використанні градієнтного спуску. В нашому випадку використовується *ReLU* – зрізаний лінійний вузол. Нарешті, вихідний шар мережі оброблюється активаційною функцією **softmax** (або **нормована експоненційна функція**). Ця функція дуже часто використовується для представлення категоріального розподілу.

На рис. 3.12 маємо модель CNN, яку ми розробили для навчання моделі над набором навчальних даних.

```
model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(3, 3),
                 activation='relu', input_shape=(28, 28, 1)))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=64, kernel_size=(3, 3),
                 activation='relu', padding='same'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=128, kernel_size=(3, 3),
                 activation='relu', padding='valid'))
model.add(MaxPool2D(pool_size=(2, 2), strides=2))

model.add(Flatten())

model.add(Dense(64, activation="relu"))
model.add(Dense(128, activation="relu"))

model.add(Dense(26, activation="softmax"))
```

Рисунок 3.12 – Модель згорткової нейронної мережі

3.3 Компіляція та тренування

Визначивши структуру мережі, тепер опишемо процес її навчання за допомогою функції .compile(). Використання цього методу наведено на рис. 3.13.

Спочатку опишемо цільову функцію. Найпопулярніша функція для вирішення проблеми логічної регресії (класифікаційної задачі) є **крос-ентропія** (або **логарифмічна функція втрат**), де

$$f(o) = -\sum_{c=1}^{M} y_{o,c} \ln(p_{o,c}), \tag{3.3}$$

де M – кількість класів;

o – приклад, для якого розраховуємо функцію;

$y_{o,c}$ – бінарний індикатор того, що клас c є коректною класифікацією для прикладу o;

$p_{o,c}$ – підрахована моделлю ймовірність того, що приклад o належить до класу c.

Також в моделі бібліотеки Keras слід зазначити оптимізатор. Був обраний оптимізатор Adam – вбудована імплементація **стохастичного градієнтного спуску**.

Наостанок визначимо метрику, тобто функцію, яка буде оцінювати продуктивність нашої моделі. Для нашої задачі вистачить точності у порівнянні з тестовими даними. Також визначимо дві додаткові функції для виконання під час тренування нашої моделі:

ReduceLROnPlateau – зменшує швидкість навчання, коли показник перестає покращуватися. Моделі часто виграють від зниження швидкості навчання в 2-10 разів після навчання. Цей зворотний виклик контролює кількість, і якщо не спостерігається покращення протягом кількох епох «спокою», швидкість навчання знижується.

EarlyStopping – припиняє навчання, коли контрольований показник перестає покращуватися. Припустимо, що мета тренінгу - мінімізувати втрати. При цьому показником для моніторингу буде «втрата», а режим — «мін». Цикл навчання model.fit() перевірятиме в кінці кожної епохи, чи втрати більше не зменшуються, враховуючи min_delta та «спокій», якщо це можливо. Як тільки буде виявлено, що він більше не зменшується, model.stop_training позначається як True, і навчання припиняється.

```
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='categorical_crossentropy', metrics=['accuracy'])
reduce_lr = ReduceLROnPlateau(
    monitor='val_loss', factor=0.2, patience=1, min_lr=0.0001)
early_stop = EarlyStopping(
    monitor='val_loss', min_delta=0, patience=2, verbose=1, mode='auto')
```

Рисунок 3.13 – Компіляція моделі

Тепер почнемо процес тренування за допомогою методу .fit() (рис. 3.14).

Визначимо **кількість епох** (англ. **epoch** – кількість ітерацій в процесі навчання) та **виклик зворотних функцій** (англ. **callback** – об’єкт, який може виконувати дії на різних етапах навчання (наприклад, на початку або в кінці епохи)).

Параметр **verbose** визначає особливості виводу інформації про тренування в консоль (в нашому випадку 1 – полінійний вивід даних про кожну епоху зі шкалою прогресу).

Наостанок ми вказуємо валідаційні дані для перевірки точності. Для цього обираємо тестові дані, які ми не використовували для тренування.

```
history = model.fit(train_X, train_yOHE, epochs=10, callbacks=[
                    reduce_lr, early_stop],  validation_data=(test_X, test_yOHE))
```

✓ 5m 13.6s Python

```
Epoch 1/10
9312/9312 [==============================] - 43s 5ms/step - loss: 0.0954 - accuracy: 0.9732
- val_loss: 0.0700 - val_accuracy: 0.9809 - lr: 0.0010
Epoch 2/10
9312/9312 [==============================] - 46s 5ms/step - loss: 0.0679 - accuracy: 0.9811
- val_loss: 0.0671 - val_accuracy: 0.9825 - lr: 0.0010
Epoch 3/10
9312/9312 [==============================] - 45s 5ms/step - loss: 0.0597 - accuracy: 0.9835
- val_loss: 0.0735 - val_accuracy: 0.9815 - lr: 0.0010
Epoch 4/10
9312/9312 [==============================] - 44s 5ms/step - loss: 0.0247 - accuracy: 0.9931
- val_loss: 0.0315 - val_accuracy: 0.9924 - lr: 2.0000e-04
Epoch 5/10
9312/9312 [==============================] - 44s 5ms/step - loss: 0.0145 - accuracy: 0.9960
- val_loss: 0.0296 - val_accuracy: 0.9937 - lr: 2.0000e-04
Epoch 6/10
9312/9312 [==============================] - 42s 5ms/step - loss: 0.0098 - accuracy: 0.9972
- val_loss: 0.0308 - val_accuracy: 0.9941 - lr: 2.0000e-04
Epoch 7/10
9312/9312 [==============================] - 42s 4ms/step - loss: 0.0046 - accuracy: 0.9988
- val_loss: 0.0305 - val_accuracy: 0.9952 - lr: 1.0000e-04
Epoch 7: early stopping
```

Рисунок 3.14 – Процес тренування моделі

Ми отримали вивід інформації про кожну епоху та зберегли її до змінної *history*. Для того, щоб краще оцінити продуктивність моделі, слід побудувати графіки точності та цільової функції втрат за кожну епоху (рис. 3.15-3.16).

```
# Displaying the accuracies & losses for train & validation set...

print("The validation accuracy is :", history.history['val_accuracy'])
print("The training accuracy is :", history.history['accuracy'])
print("The validation loss is :", history.history['val_loss'])
print("The training loss is :", history.history['loss'])

fig = plt.figure()

acc_plot = plt.subplot(2,1,1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='lower right')

loss_plot = plt.subplot(2,1,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['train', 'test'], loc='upper right')
acc_plot.set_xticklabels([0, 1, 2, 3, 4, 5, 6, 7])
loss_plot.set_xticklabels([0, 1, 2, 3, 4, 5, 6, 7])
fig.tight_layout()
```

Рисунок 3.15 – Процес візуалізації функцій

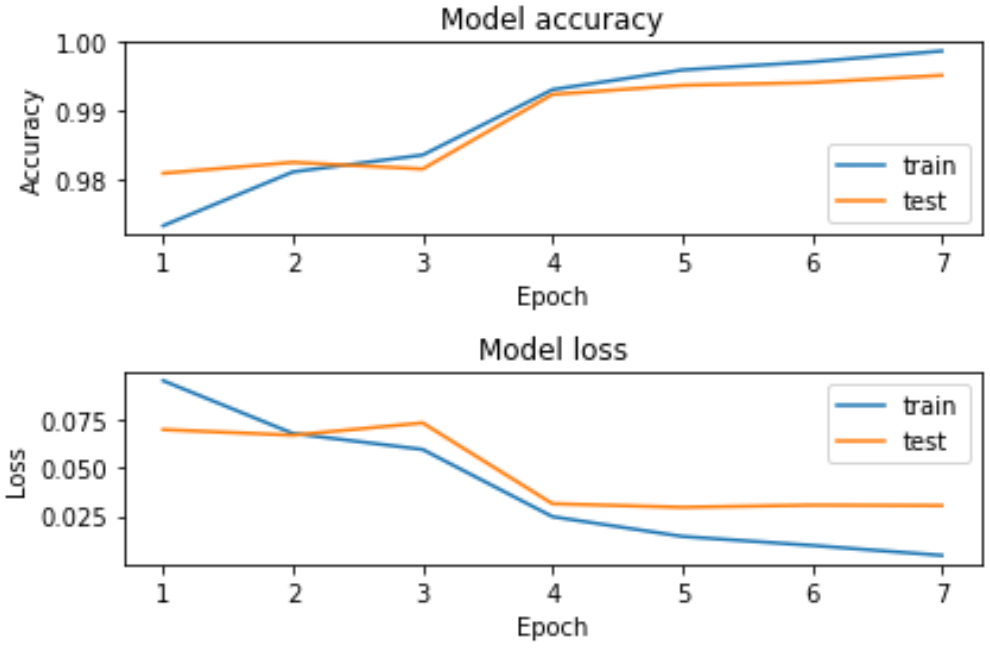

Рисунок 3.16 – Вивід графіків функцій точності та втрат

Як ми бачимо, функція втрат для тренувальних даних швидко зменшує значення під час перших 4 епох, після цього значення дещо зменшується до 7 епохи, а далі залишається маже незмінним (зменшується настільки повільно, що цим можна знехтувати). Це вказує на високу швидкість навчання та коректність роботи градієнтного спуску, адже значення тільки зменшується. Тепер нам треба перевірити роботу моделі на тестових даних. Для цього ми створюємо 9 графіків форми (3x3) і візуалізуємо деякі літери з тестового набору даних разом із їхніми прогнозами, які зроблені за допомогою функції model.predict() для розпізнавання літер (рис. 3.17).

```
# Making model predictions...

pred = model.predict(test_X[:9])
print(test_X.shape)


# Displaying some of the test images & their predicted labels...

fig, axes = plt.subplots(3, 3, figsize=(8, 9))
axes = axes.flatten()

for i, ax in enumerate(axes):
    img = np.reshape(test_X[i], (28, 28))
    ax.imshow(img, cmap="Greys")
    pred = word_dict[np.argmax(test_yOHE[i])]
    ax.set_title("Prediction: "+pred)
    ax.grid()
```

Рисунок 3.17 – Перевірка на тестових даних та виведення зображень

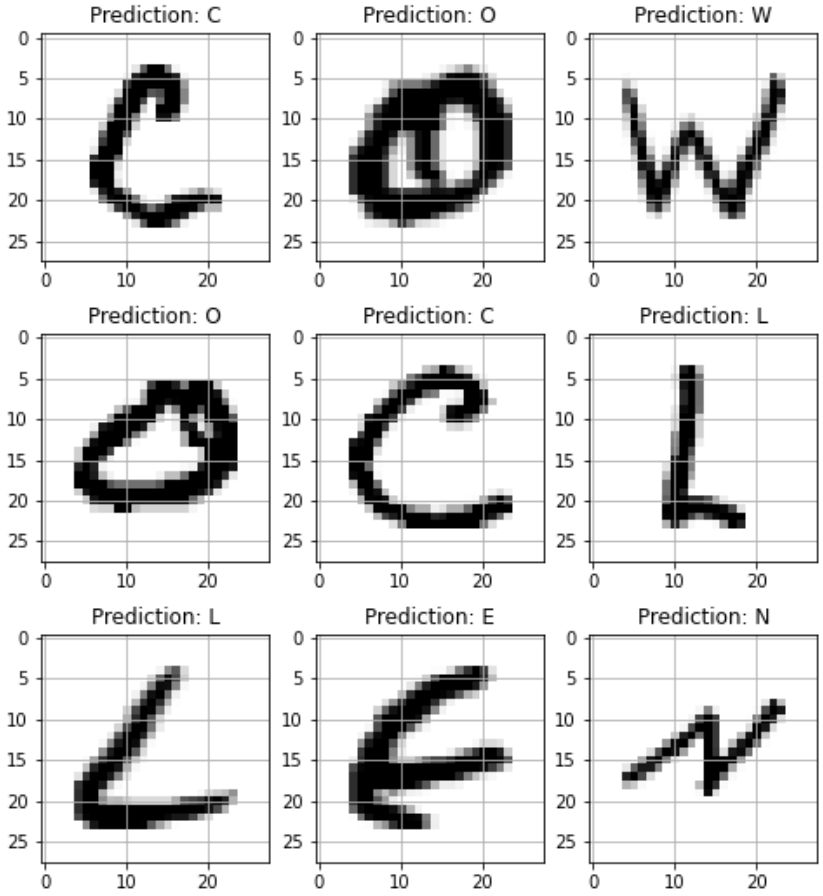

Рисунок 3.18 – Результат прогнозування моделі на тестових даних

Перевірка на тестових даних вказує на те, що модель дуже точно класифікує невідомі їй дані: точність на валідаційних даних відрізняється лише на декілька сотих відсотка.

3.4 Конвертація та зберігання моделі

Ми завершили побудову моделі нашого класифікатора. Тепер нам слід обробити його для використання. Раніше було зазначено, що для взаємодії вебзастосунку з мережею було обрано бібліотеку Tensorflow.js. На щастя, ця бібліотека надає механізм конвертації Keras-моделей.

Тепер конвертуємо та зберігаємо нашу модель (рис. 3.19).

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
model.summary()
model.save(r'model_hand.h5')

import tensorflowjs as tfjs
tfjs.converters.save_keras_model(model, 'C:/Users/ivan/Desktop/New folder/mnist-web-app-master/src/assets/modeldir')
```

Рисунок 3.19 – Конвертація та збереження моделі

Висновки до розділу 3

Отже, під час виконання третього розділу було проаналізовано вміст файлу, що містить набір вхідних даних, опрацьовано ці дані та підготовлено для подальшої роботи з моделлю. Була створена та реалізована структура згорткової нейронної мережі, скомпільовано та натреновано цю мережу, а також перевірено її роботу на тестових даних. Модель було збережено для подальшої роботи із вебзастосунком.

4 РОЗРОБКА ВЕБЗАСТОСУНКУ ТА ВИКОРИСТАННЯ СТВОРЕНОЇ МОДЕЛІ

В цьому розділі описується процес розробки вебзастосунку з використанням створеної класифікаційної моделі.

Розглянути код можна тут: https://github.com/sashasova1/mnist-web-app-master.

4.1 Розробка інтерфейсу вебзастосунку

Основна концепція сайту – користувач має здатність малювати білим кольором на чорному полотні (HTML-елемент canvas) і, натиснувши на кнопку “Predict” (“Передбачити”), отримати гістограму імовірностей відносно літер від A до Z того, що на полотні відповідна літера. При натисканні на кнопку “Clear” (“Очистити”) відбувається очищення полотна.

Як було зазначено раніше, для розробки вебзастосунку було обрано фреймворк *Angular 6*. Для відображення гістограми ж була обрана бібліотека *Chart.js*.

Інтерфейс створеного вебсайту наведено на рис. 4.1.

Рисунок 4.1 – Графічний інтерфейс вебзастосунку

4.2 Інтеграція створеної моделі до вебзастосунку

Перед інтеграцією моделі нейронної мережі до проекту була додана бібліотека *Tensorflow.js* (пакет *@tensorflow/tfjs*).

Переконавшись в наявності вищезазначеної бібліотеки, додаємо завантажену модель до файлів проекту та опишемо її асинхронне завантаження при ініціалізації сайту (рис. 4.2-4.3).

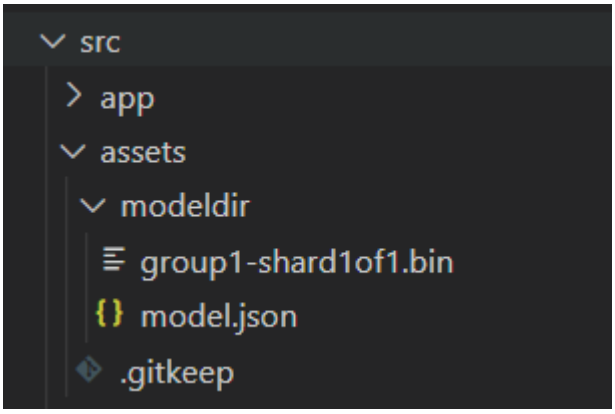

Рисунок 4.2 – Розташування моделі в проекті

```
ngOnInit(): void {
  this.loadModel();
}

async loadModel() {
  this.model = await tf.loadModel('/assets/modeldir/model.json');
}
```

Рисунок 4.3 – Завантаження моделі до проекту

Модель зберігається в об’єкті бібліотеки *@tensorflow/tfjs* типу Model. Для аналізу даних полотна ми пересилаємо його вміст як об’єкт типу ImageData та пересилаємо їх до метода Model.predict(). Реалізація зображена на рис. 4.4.

```
async predict(imageData: ImageData) {
  tf.tidy(() => {
    const img = tf.fromPixels(imageData)
      .resizeBilinear([28, 28]) // shrink data to 28x28 -> dims - [28, 28, 3]
      .mean(2) // (avg of rgb dimenson [28, 28, 3]-> grayscale [28, 28])
      .toFloat() // explicit declaration, mean already implicitly converts to float
      .reshape([1, 28, 28, 1]);
    const output = this.model.predict(img) as any;
    this.predictions = Array.from(output.dataSync());
  });
}
```

Рисунок 4.4 – Функція обробки даних с полотна

Слід описати послідовність дій та функції, що їх виконують:

1. завантажуємо масив піксельних – `.fromPixels()`;
2. перетворюємо масив до розміру 28x28 – `.resizeBilinear([28, 28])`;
3. перетворюємо 3-канальне зображення на чорно-біле методом середнього арифметичного (2 – простір з каналами) – `.mean(2)`;
4. явне перетворення чисел до float (попередній метод робить це неявно, але слід вказати для логічності) – `.toFloat()`;
5. реретворюємо масив 28x28 в [1,28,28,1] – `.reshape([1,28,28,1])`;
6. викликаємо метод `.predict()` на отриманий масив даних та отримуємо масив передбачень, який завантажуємо до гістограми.

Приклад роботи застосунку можна побачити на рис. 4.5.

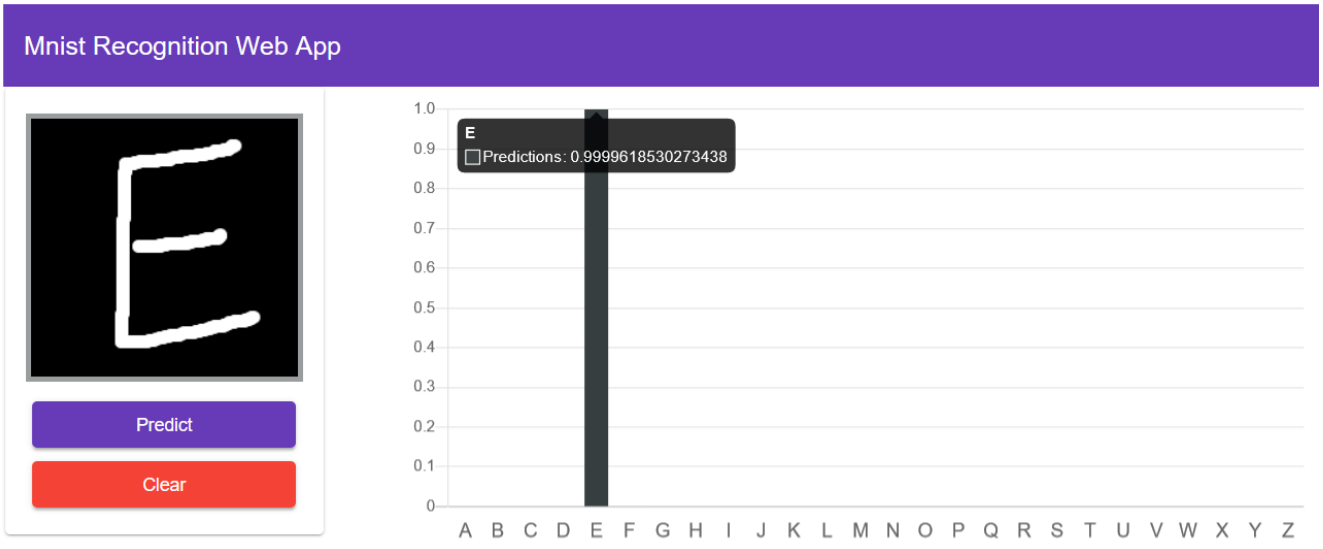

Рисунок 4.5 – Робота вебзастосунку

4.3 Аналіз результату

Створена нейронна мережа здатна відносно точно класифікувати намальовану цифру. Приклади зображені на рис. 4.6-4.9.

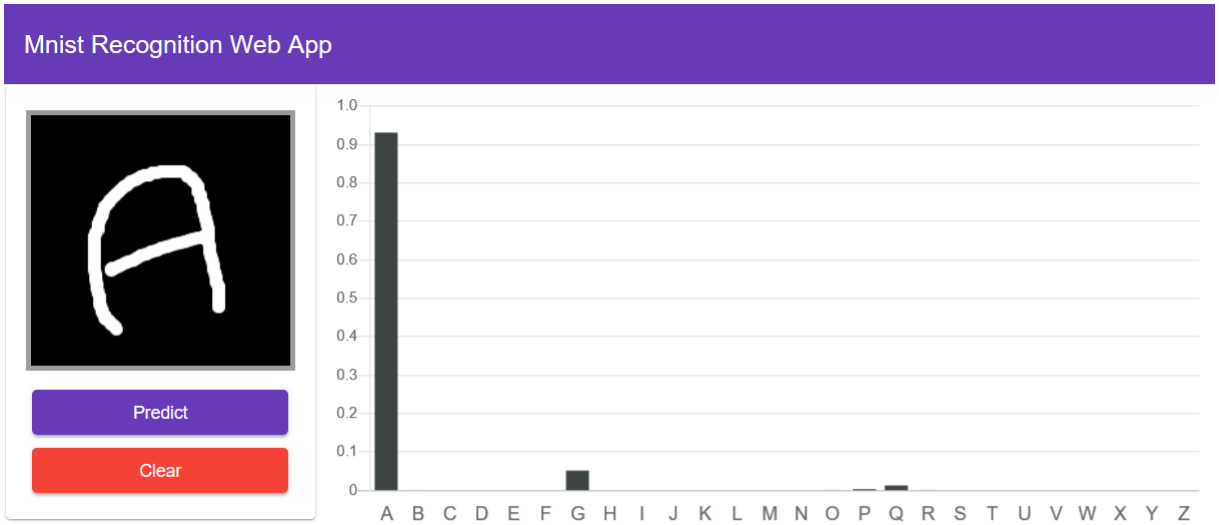

Рисунок 4.6 – Літера А

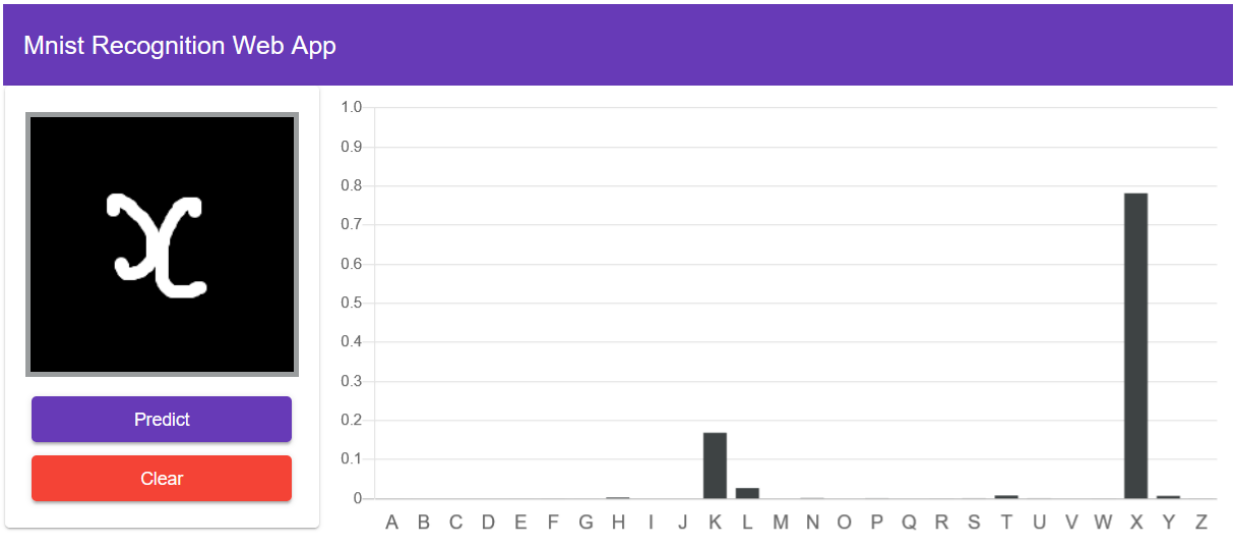

Рисунок 4.7 – Літера X

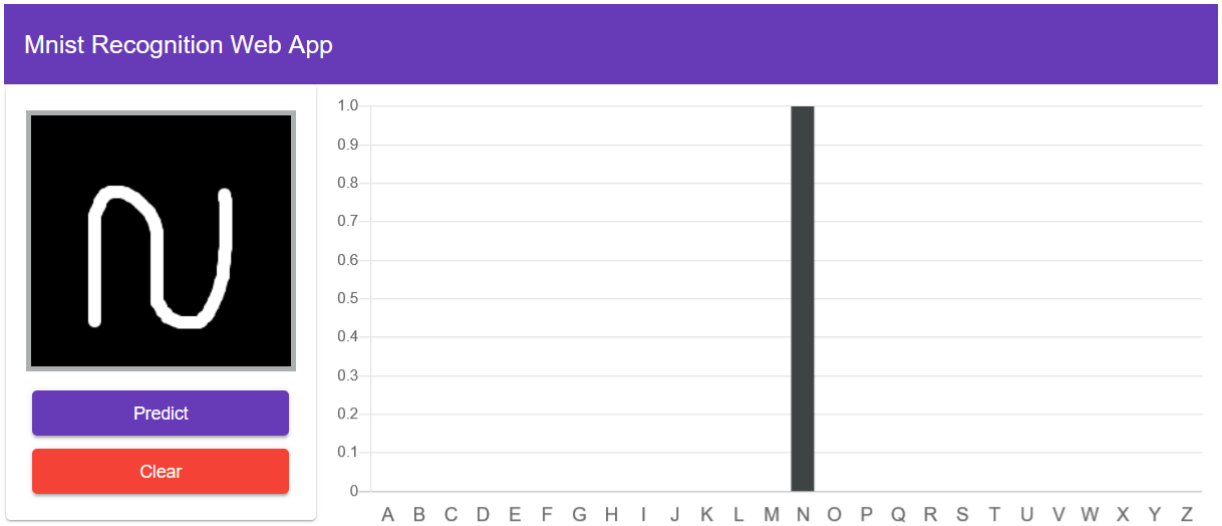

Рисунок 4.8 – Літера N

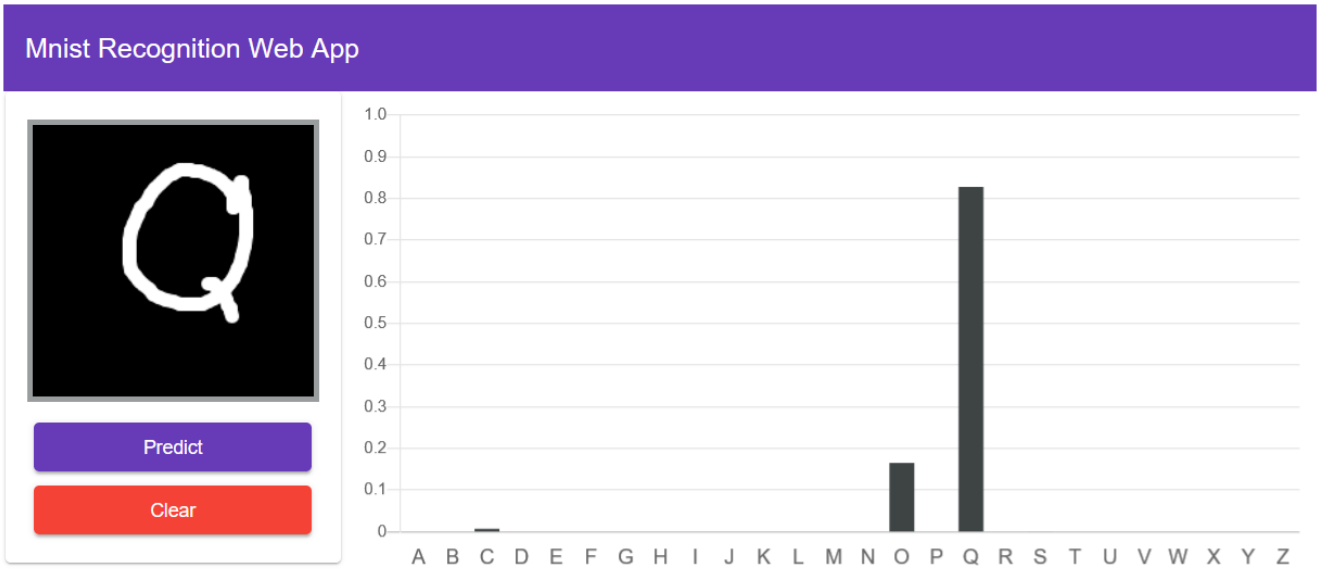

Рисунок 4.9 – Літера Q

Деякі з малюнків класифікуються неправильно (рис. 4.10.-4.13).

Це можна пояснити тим, що ми розглядаємо все полотно як цифру, а також тому, що крос-валідація проводилась на даних із вибірки NIST, а дані, що є рукописними в реальному часі, є більш непередбачуваними. Тим не менш, це не є проблемою перетренування моделі, адже валідаційні дані не були частиною тренування.

Рисунок 4.10 – Літера P/T

Рисунок 4.11 – Літера V/Y

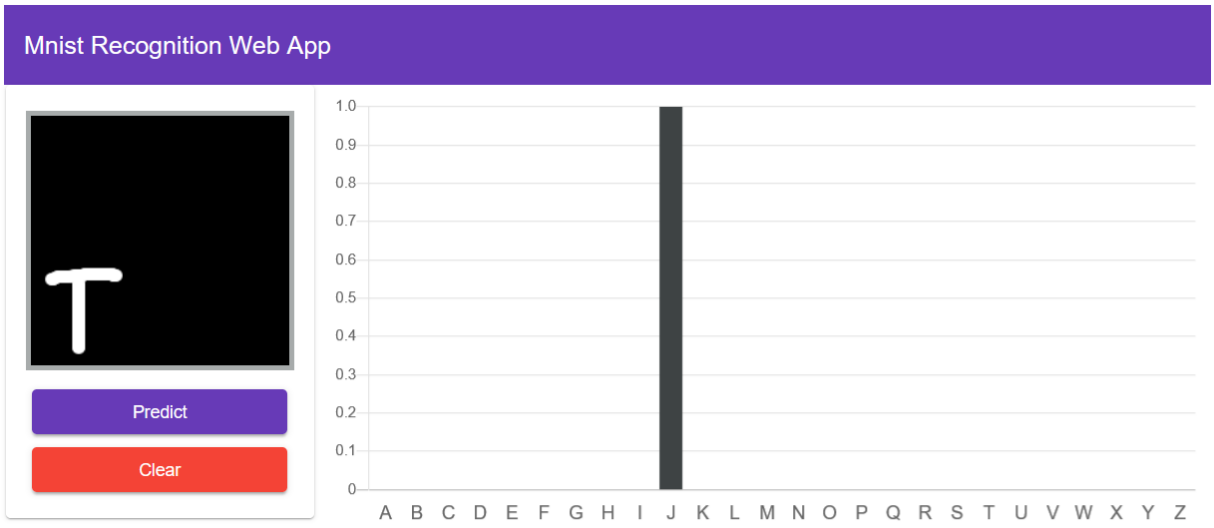

Рисунок 4.12 – Літера T/J

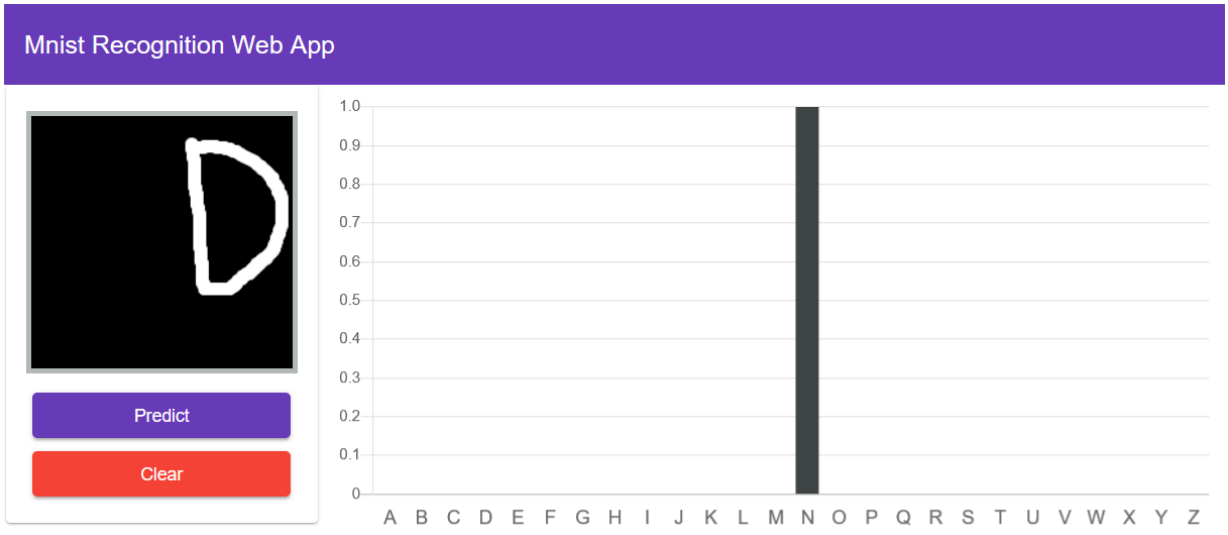

Рисунок 4.13 – Літера D/N

Виходячи з цього, можна визначити методи покращення продуктивності моделі.

Основним є збільшення тренувальних даних, наприклад:

1. включення тестових даних NIST до кінцевого тренування;
2. додати можливість використовувати малюнки у реальному часі як тренувальні дані.

Також для покращення роботи можна змінити інші особливості реалізації:

1. додати можливість вибору товщини лінії;
2. додати можливість вибору певного прямокутника із полотна як область з літерою.

Висновки до розділу 4

Отже, під час виконання четвертого розділу було розроблено інтерфейс вебзастосунку, інтегровано створену модель у застосунок та проаналізовано результати роботи програми, виявлено неточності та помилки при роботі. Наведено приклади для покращення продуктивності моделі.

Спеціальний розділ

ОХОРОНА ПРАЦІ

до кваліфікаційної роботи

на тему:

«ІНФОРМАЦІЙНА МОДЕЛЬ ДЛЯ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЛІТЕР»

Спеціальність 122 «Комп'ютерні науки»

122 – БКР – 402.21810224

Виконав студент 4-го курсу, групи 402

Сова О.М.

(підпис, ініціали та прізвище)

«___» _____ 202_ р.

Консультант_____*ст. викладач*_____

(наук. ступінь, вчене звання)

_____*Макарова О.В.*_____

(підпис, ініціали та прізвище)

«___» _____ 202_ р.

Миколаїв – 2022

5 ОХОРОНА ПРАЦІ

На сьогоднішній день кожне підприємство використовує інформаційні технології, а саме працює за допомогою різних технічних пристроїв. Наприклад, персональний комп’ютер використовується для збору, обміну та аналізу інформації всередині підприємства та між іншими установами. Таке використання персонального комп’ютера дуже полегшує різні процеси на підприємстві. Але треба зазначити, що вплив комп’ютера несе не тільки позитивні наслідки, але й негативні. Використання комп’ютера загострило проблеми суспільного, а насамперед власного здоров’я, що вимагає удосконалення існуючих робочих місць та розробки нових, більш позитивних до здоров’я робочих місць та вимагає проведення профілактичних заходів до запобігання розвитку негативних наслідків впливу персонального комп’ютера на здоров’я користувача.

При роботі з комп'ютером людина повністю залежить від положення дисплея. Крім того, зображення на екрані динамічно оновлюється, а низька частота оновлення викликає його мерехтіння. При цьому очні і внутрішньоочні м'язи, фокусують погляд, втомлюються від надмірного навантаження. Розвивається зорове стомлення, що сприяє виникненню короткозорості. Тривала робота з комп'ютером вимагає також підвищеної зосередженості, що призводить до появи головного болю, дратівливості, нервової напруги і стресу.

Метою даного розділу є створення безпечних і здорових умов праці на робочому місці або у виробничому приміщенні.

5.1 Нормативна документація щодо забезпечення охорони праці під час використання екранними пристроями

Існують правила, які працівники підприємств, установ повинні дотримуватися під час використання електронно-обчислювальних машин. Дотримування цих правил знизить наслідки несприятливої дії на працівників шкідливих та небезпечних факторів, які можуть супроводжуватися під час роботи з електронно-обчислювальними машинами. По перше, це стосується зорових та нервово-емоційних перевантажень, серцево-судинних захворювань та психічної складової працівника.

У правилах охорони праці під час експлуатації електронно-обчислювальних машин викладені гігієнічні та ергономічні вимоги до робочих приміщень та місць, параметрів робочого середовища, і якщо дотримуватися усіх цих правил, дає змогу уникнути порушення стану здоров'я користувача комп'ютера.

Відповідальність за виконання усіх цих правил покладається на посадових осіб, фізичних осіб , які займаються підприємницькою діяльністю та здійснюють застосування електронно-обчислювальних машин в адміністративних та промислових приміщеннях.

Державний санітарний нагляд за дотримання усіх цих правил державними органами, підприємствами, установами, організаціями незалежно від форми власності, а також фізичними особами, які займаються підприємницькою діяльністю, покладається на органи і установи санітарно – епідеміологічного профілю Міністерства охорони здоров'я України, відповідні установи, організації, частини та підрозділи Міністерства оборони України, закони України (ст. 31 Закону України «Про забезпечення санітарного та епідеміологічного благополуччя населення») [36].

5.2 Вимоги до приміщення з використанням екранних пристроїв

Усі приміщення, де робітники використовують персональні комп’ютери, повинні відповідати ряду вимог. Ці вимоги перевіряються лише спеціальними службами, які призначені для нагляду за дотриманням вимог охорони праці на підприємствах. Нижче наведено частину вимог з документу про правила охорони праці під час роботи на підприємствах:

− об'ємно-планувальні рішення будівель та приміщень для роботи з електронно-обчислювальними машинами мають відповідати вимогам ДСанПІН 3.3.2.007-98 [28];

− розміщення робочих місць з ВДТ ЕОМ і ПЕОМ у підвальних приміщеннях, на цокольних поверхах заборонено;

− площа на одне робоче місце становить не менше ніж 6,0 м, а об'єм - не менше ніж 20,0.м;

− приміщення для роботи з комп’ютерами повинні мати природне та штучне освітлення відповідно до СНиП II-4-79 [29];

− природне освітлення має здійснюватися через світлові прорізи, орієнтовані переважно на північ чи північний схід, і забезпечувати КПО не нижче, ніж 1,5%:

− виробничі приміщення повинні обладнуватись шафами для зберігання документів, магнітних дисків, полицями, стелажами, тумбами тощо, з урахуванням вимог до площі приміщень;

− у приміщеннях з електронно-обчислювальними машинами слід щоденно робити вологе прибирання;

− приміщення з електронно-обчислювальними машинами мають бути забезпечені аптечками першої медичної допомоги;

− на підприємствах, де є комп’ютери, мають бути обладнані побутові приміщення для відпочинку під час роботи, кімната психологічного

розвантаження. В кімнаті психологічного розвантаження слід передбачити встановлення пристроїв для приготування й роздачі тонізуючих напоїв, а також місця для занять фізичною культурою (СНиП 2.09.04.-87) [30].

Нижче наведено частину вимоги до безпеки робочих місць працівників з екранними пристроями з наказу Міністерством соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» на стан 18 травня 2018 року [21]:

− робочі місця працівників з екранними пристроями повинні бути спроектовані так і мати такі розміри, щоб працівники мали простір для зміни робочого положення і рухів;

− всі випромінювання від екранних пристроїв повинне бути зведене до гранично допустимого рівня з точки зору безпеки і охорони здоров'я працівників;

− організація робочого місця працівника з екранними пристроями повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним, антропологічними, психофізіологічним вимогам, а також характеру виконуваних робіт;

− освітлення робочого місця працівника з екранними пристроями повинно створювати відповідний контраст між екраном і навколишнім середовищем (з урахуванням виду роботи) і відповідати вимогам ДСанПІН 3.3.2.007-98 [9];

− мікроклімат виробничих приміщень з робочими місцями працівників з екранними пристроями повинен підтримуватися на постійному рівні і відповідати вимогам Санітарних норм мікроклімату виробничих приміщень ДСН 3.3.6.042-99 [31];

− робочий стіл або робочу поверхню повинні бути достатнього розміру і мати поверхню з низькою відбивною здатністю, допускати гнучкість при розміщенні екрана, клавіатури, документів і відповідного обладнання;

− робоче крісло має бути стійким та дозволяти працівникові з екранними пристроями легко рухатися і займати зручне положення. Сидіння повинна регулюватися по висоті, спинка сидіння - як по висоті, так і по нахилу;

− слід передбачати підставку для тих, кому це необхідно для зручності.

Крім тих вимог, які повинні дотримуватися усі робочі місця на підприємствах, є ще правила електронебезпеки. Нижче наведено частину вимог з документу про правила охорони праці під час експлуатації електронно-обчислювальної машини, розділ 2, пункт 2.3 – Вимоги електронебезпеки [22]:

− під час монтажу необхідно унеможливити виникнення електричного джерела загоряння через коротке замикання та перевантаження проводів;

− обов’язково встановити аварійний резервний вимикач, якщо у приміщення працює понад п’ять комп’ютерів;

− електромережу штепсельних розеток живлення для комп’ютерів та іншого обладнання прокладати у каналах, при цьому не допускається використання деяких матеріалів, що швидко загоряються.

Також існує вимоги до виробничого приміщення щодо параметрів мікроклімату, освітлення, шуму та вібрації, рівні електромагнітного та іонізуючого випромінювання. Це потрібно для зменшення впливу негативних факторів на праці працівника.

У виробничих приміщеннях на усіх робочих місцях з персональними комп’ютерами мають обов’язково забезпечуватися оптимальні значення параметрів мікроклімату, а саме температури відносної вологості й рухливості повітря (ГОСТ 12.1.005-88, СН 4088-86) [32, 33] (табл. 5.1).

Таблиця 5.1. Норми мікроклімату для приміщень з ВТД ЕОМ та ПЕМ

Пора року	Категорія робіт	Температура повітря, °С, не більше	Відносна вологість повітря, %	Швидкість руху повітря, м/с
Холодна	Легка - 1а	22-24	40-60	0,1
	Легка - 16	21-23	40-60	0,1
Тепла	Легка - 1а	23-25	40-60	0,1
	Легка - 16	22-24	40-60	0,2

До категорії робіт 1а належать, що виконується робота сидячі і не потребує фізичного напруження людини, до категорії робіт 1б належать, що робота виконується сидячі, стоячи або пов'язані з ходінням та потребує деякого фізичного напруження.

Рівні позитивних та негативних іонів в повітрі приміщень з ВДТ мають відповідати санітарно – гігієнічним нормам №2152-80 [34] (табл. 5.2).

Таблиця 5.2. Рівні іонізації повітря приміщень при роботі на ВДТ

Рівні	Кількість іонів в 1 см повітря	
	n+	n-
Мінімально необхідні	400	600
Оптимальні	1500-3000	3000-5000
Максимально допустимі	50000	50000

Штучне освітлення в приміщеннях з робочими місцями на підприємствах повинно бути обладнане ВДТ та має здійснюватися системою загального рівномірного освітлення. У виробничих та адміністративних приміщеннях, у разі роботою з документами, допускається застосування системи комбінованого освітлення.

Значення освітленості на поверхні робочого стола має становити 300-500лк. Якщо не має можливості забезпечити системою загального освітлення, допускається використання місцевим освітленням. Місцеве освітлення слід встановлювати так, щоб не створювати відблисків на поверхні екрану, а освітленість екрану персонального комп'ютера має не перевищувати 300лк.

Як джерело світу у приміщеннях для штучного освітлення мають використовуватися люмінесцентні лампи типу ЛБ. При використання відбитого освітлення у виробничих та адміністративних приміщеннях допускається використання метало-галогенних ламп, які мають потужність 250Вт.

Допускається використання ламп розжарювання у світильниках місцевого освітлення.

Інтенсивність потоків інфрачервоного випромінювання має не перевищувати допустимих значень, які зазначені у ДСН 3.3.6.042-99 [31].

Інтенсивність потоків ультрафіолетового випромінювання має не перевищувати допустимих значень, які зазначені у СН 4557-88 [35].

Потужність експозиційної дози рентгенівського випромінювання на відстані 0,05м від екрану та корпусу комп'ютера не повинна перевищувати 0,1мбер/рік (100мкР/рік).

5.3 Вимоги до роботи з екранними пристроями

На сьогодні існує ряд вимог, які повинні використовувати працівники на підприємствах, основним місцем роботи яких є місце за персональним комп'ютером. Нижче наведено частину вимог з наказу Міністерством соціальної політики України «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями» на стан 18 травня 2018 року [3]:

Мінімальні вимоги безпеки при роботі з екранними пристроями:

– щодня перед початком роботи необхідно очищати екран пристрою від пилу та інших забруднень;

– після закінчення роботи екранні пристрої слід відключати від електричної мережі;

– у разі виникнення аварійної ситуації необхідно негайно відключити екранний пристрій від електричної мережі.

Не допускається:

– виконувати технічне обслуговування, ремонт і налагодження екранних пристроїв безпосередньо на робочому місці працівника при роботі з екранними пристроями;

– відключати захисні пристрої, самочинно проводити зміни у конструкції та складі екранних пристроїв або їх технічне налагодження;

– працювати з екранними пристроями, в яких під час роботи з'являються нехарактерні сигнали, нестабільне зображення на екрані та інші несправності;

– при виконанні робіт операторського типу, пов'язаних з нервово-емоційним напругою, у приміщеннях при роботі з екранними пристроями, на пультах і постах керування технологічними процесами і в інших приміщеннях повинні дотримуватися оптимальні умови мікроклімату відповідно до вимог ДСН 3.3.6.042-99 [31].

Мінімальні вимоги безпеки до екранних пристроїв:

– екранні пристрої не повинні бути джерелом ризику для працівників.

– всі випромінювання, за винятком видимої частини електромагнітного спектра, повинні бути зведені до незначного рівня з точки зору безпеки і охорони здоров'я працівників.

– символи на екранних пристроях повинні бути чіткими, відповідного розміру. Між символами і рядками символів має бути належну відстань.

– зображення на екрані повинно бути стабільним, без спалахів або інших видів нестабільності.

− яскравість і/або контрастність символів має легко регулюватися працівником при роботі з екранними пристроями, а також швидко адаптуватися до навколишніх умов.

− вибираючи екрани, слід віддавати перевагу таким екранів, які легко і вільно повертаються і нахиляються у відповідності з потребою працівника.

− при необхідності може використовуватися окрема підставка або регульований стіл для розміщення екрану.

− екран не повинен відсвічувати або відбивати світло, щоб не викликати дискомфорту у працівника при роботі з екранними пристроями.

− вибираючи клавіатуру, слід віддавати перевагу такій клавіатурі, яка відкидається і є автономною (відокремленої від екрана), щоб працівник міг вибрати зручну робочу позу і уникнути втоми рук (кисті та верхньої частини руки).

− поверхня клавіатури має бути матовою, щоб уникнути відображення. Розташування клавіш і самі клавіші повинні полегшувати роботу з клавіатурою. Позначення клавіш має бути досить контрастним і розбірливим.

− при розробці, виборі, замовленні та модифікації програмного забезпечення, а також при розробці завдань, які передбачають використання обладнання з екранними пристроями, роботодавець повинен керуватися таким програмним забезпеченням, яке відповідає завданням і є простим у використанні, а де необхідно адаптованим до рівня знань і досвіду працівника.

5.4 Вимоги щодо режиму відпочинку та праці на підприємствах з екранними пристроями

При організації праці, яка пов’язана з електронними пристроями, для збереження здоров’я працівників, ухилення професійних захворювань і підтримки працездатності, повинно передбачатися внутрішньозмінні регламентовані перерви на відпочинок.

Внутрішньозмінні режими праці та відпочинку повинні містити додаткові нетривалі перерви в період, які передують появі об’єктивних в суб’єктивних ознак стомлення та зниження працездатності.

При виконанні роботи, що належать до різної трудової діяльності, за основну роботу з ВДТ треба вважати таку, що займає не менше 50% робочого часу. Під час робочої зміни повинно передбачатися:

- перерви для відпочинку та вживання їжі;
- перерви для відпочинку та особистих потреб згідно з трудовими нормами;
- додаткові перерви, які вводяться для окремих професій з урахуванням особливостей трудової діяльності.

Правила встановлюють внутрішньозмінні режим праці та відпочинку при роботі з електронними пристроями при 8-годинній денній робочій зміні від характеру праці. Для розробників програмного забезпечення при використання електронними пристроями слід призначити регламентовану перерву для відпочинку тривалістю 15 хвилин через кожну годину роботи за персональним комп’ютером. Робітникам, які працюють за комп’ютером, необхідно робити перерву для відпочинку кожну годину або дві під час робочого дня тривалістю 10-15 хвилин. У всіх випадках, коли виробничі обставини не дозволяють використовувати регламентовану перерву, тривалість безперервної роботи з електронними пристроями не повинна бути більше, ніж 4 години.

При 12-годонній робочій зміні регламентовані перерви повинні встановлюватися в перші 8 годин роботи, аналогічно перервам при 8-годинній робочій зміні, а протягом останніх 4-х годин роботи, незалежно від характеру трудової діяльності, через кожну годину тривалістю 15 хвилин.

5.5 Забруднення повітря на робочих місцях з використанням екранних пристроїв

Чимало досліджень було присвячено визначенню хімічного складу повітря на робочих місцях операторів ВДТ. Багатьма дослідниками було відмічено, що до кінця робочого дня в повітря робочої зони різко зростає концентрація CO_2 яка сягала від 0,12-0,13 до 0,19% (в атмосферному повітрі CO_2 міститься 0,03%).

Особливу небезпеку щодо впливу на здоров'я представляє підвищена концентрація озону - високотоксичного подразнюючого газу. З цієї причини він був внесений у список речовин, максимальні значення концентрації яких на робочих місцях обмежені та строго визначені. Надзвичайна небезпека озону для здоров'я людини пов'язана з тим, що він належить до так званих радіоміметичних речовин – хімічних сполук, що викликають в живих організмах зміни, схожі з тими, які виникають після дії іонізуючого випромінювання. Тому озон вважається не лише подразнюючою, а й канцерогенною речовиною.

Основними джерелами озону на комп'ютеризованих місцях є ЕПТ ВДТ та лазерні принтери. З огляду на це, необхідно виключати ВДТ у випадках, коли він не використовується, а лазерний принтер бажано розташовувати подалі від робочого місця оператора. Однак, це додаткові заходи, основним же заходом щодо запобігання несприятливого впливу озону та інших шкідливих речовин на здоров'я операторів є забезпечення функціонування припливно-витяжної вентиляції. Для того, щоб шкідливі речовини не проникали із сусідніх приміщень в приміщеннях з ВДТ необхідно створити деякий надлишковий тиск.

Відповідно до ГОСТ 12.1.005-88 [32] вміст озону в повітрі робочої зони не повинен перевищувати 0,1 мг/м3; вміст оксидів азоту - 5 мг/м3; вміст пилу – 4 мг/м3.

5.6 Виробничий шум та вібрація

Відомо, що шум несприятливо діє на слуховий аналізатор та інші органи та системи організму людини. Визначальне значення щодо такої дії має інтенсивність шуму, його частотний склад, тривалість щоденного впливу, індивідуальні особливості людини, а також специфіка виробничої діяльності.

Рівні звукового тиску в октавних смугах частот, рівні звуку та еквівалентні рівні звуку на робочих місцях, обладнаних ВДТ і ПК визначені ДСанПіН 3.3.2-007-98 [9].

Основними заходами та засобами боротьби з шумом є:

- зниження рівнів шуму в джерелі його утворення (застосовується, як правило, в процесі проектування);
- використання звукопоглинаючих та звукоізолюючих засобів;
- раціональне планування виробничих приміщень та робочих місць.

На комп'ютеризованих робочих місцях основними джерелами шуму є вентилятори системного блоку, накопичувані, принтери ударної дії. Для зниження рівнів шуму на робочих місцях рекомендується розмістити друкувальні пристрої ударної дії (матричні, шрифтові принтери тощо) в іншому приміщенні, або огородити їх звукоізолюючими екранами.

Оскільки зовнішні шуми (вулиця, суміжні приміщення) також можуть негативно впливати на функціональний стан операторів ВДТ, то стіни приміщень, в яких розташовані комп'ютеризовані робочі місця бажано облицювати звукопоглинаючими матеріалами. Однак доцільність їх застосування повинна бути обґрунтована спеціальними інженерно-акустичними розрахунками. Звукопоглинаюче облицювання стін (іноді й стелі) необхідно здійснювати матеріалами, що мають максимальний коефіцієнт звукопоглинання в межах частот 31,5-8000 Гц і дозволені для оздоблення приміщень органами державного санітарно-епідеміологічного нагляду.

Для зниження вібрації обладнання, пристрої, пристосування необхідно встановлювати на спеціальні амортизуючи прокладки, передбачені нормативними документами.

Висновки до розділу з охорони праці

Під час виконання спеціальної частини з охорони праці було досліджено умови праці людини на підприємствах та установах, а також роботу при користуванні електронними пристроями.

Правила, які прописані у нормативних документах є актуальними на сьогодні через те, що майже в кожному підприємстві є працівники, які працюють з електронними пристроями. Під час роботи за персональним комп’ютером людина швидко втомлюється та від цього страждає опорно-руховий, зоровий апарат, нервова система и в окремих випадках психічне здоров’я.

На сьогодні існує багато шкідливих умов, які погіршують процес трудової діяльності людини, такі як: шум, вібрація та інші умови.

Вимоги та нормативи, щодо охорони праці мають дотримуватися на підприємствах під час трудової діяльності людини. Дотримання поставлених вимог до працівників та власників підприємств дозволить мінімізувати шкідливі наслідки, які мають вплив на здоров’я людини.

ВИСНОВОК

Під час виконання дипломної роботи було проаналізовано і поставлено задачу розпізнавання рукописних літер, розглянуто сучасні аналоги розробки ПЗ для розпізнавання тексту. Розглянуто та проаналізовано методи і технології, які можуть бути застосовані для вирішення поставленої задачі розпізнавання рукописних літер.

Була створена, розроблена і натренована модель багатошарового перцептрону (CNN) для вирішення задачі мультикласифікації на прикладі рукописних літер із спеціальної бази даних NIST, були закріплені навички обробки, візуалізації та аналізу даних, а також роботи с фреймворками для нейронних мереж Keras, Tensorflow.js, програмними мовами Python, TypeScript, HTML/CSS та фреймворком Angular 6.

Створено вебзастосунок, інтерфейс якого містить полотно для роботи з моделлю, яку було інтегровано у вебзастосунок та стовпчикову діаграму із літерами алфавіту, яка показує передбачення мережі на основі намальованої літери та перевірено коректність роботи застосунку.

Теоретична значимість роботи полягає у можливості використання детально описаних математичних моделей для створення і налаштування нейронної мережі для інших задач.

Практична значимість полягає в роботі з різними бібліотеками, що надає змогу використовувати нейроні мережі на будь-якому пристрої. Окрім цього також практично-значимим є діюча система розпізнавання символів.

СПИСОК ДЖЕРЕЛ ТА ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Aqab S. Handwriting Recognition using Artificial Intelligence Neural Network and Image Processing / S. Aqab, M. T. Usman. // (IJACSA) International Journal of Advanced Computer Science and Applications. – 2020. – №7. – С. 137.
2. Pattern recognition - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Pattern_recognition (дата звернення: 24.05.2022).
3. Neural networks and deep learning. *Neural networks and deep learning*. URL: http://neuralnetworksanddeeplearning.com/chap1.html (дата звернення: 24.05.2022).
4. Google об'єктив – Вікіпедія. *Вікіпедія*. URL: https://uk.wikipedia.org/wiki/Google_Об'єктив (дата звернення: 23.06.2022).
5. Google lens - search what you see. *Google Lens*. URL: https://lens.google/howlensworks/ (дата звернення: 24.05.2022).
6. ABBYY FineReader - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/ABBYY_FineReader (дата звернення: 24.05.2022).
7. 10 best OCR software of 2022 (free & paid tools). *Nanonets AI & Machine Learning Blog*. URL: https://nanonets.com/blog/ocr-software-best-ocr-software/ (дата звернення: 25.05.2022).
8. Adobe acrobat - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Adobe_Acrobat (дата звернення: 25.05.2022).
9. Microsoft OneNote - Wikipedia. *Wikipedia, the free encyclopedia*. URL: https://en.wikipedia.org/wiki/Microsoft_OneNote (дата звернення: 25.05.2022).
10. Office Lens для windows. *Microsoft Support*. URL: https://support.microsoft.com/uk-UA/office/office-lens-для-windows-577ec09d-8da2-4029-8bb7-12f8114f472a (дата звернення: 25.05.2022).
11. IBM Datacap - overview. *IBM - Deutschland | IBM*. URL: https://www.ibm.com/products/data-capture-and-imaging/ (дата звернення: 25.05.2022).

12. What is Amazon Textract? - Amazon Textract. URL: https://docs.aws.amazon.com/textract/latest/dg/what-is.html (дата звернення: 26.05.2022).
13. Linear, quadratic, and regularized discriminant analysis. *Data Science Blog: Understand. Implement. Succeed.* URL: https://www.datascienceblog.net/post/machine-learning/linear-discriminant-analysis/ (дата звернення: 26.05.2022).
14. Gandhi R. Naive bayes classifier. *Medium*. URL: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c (дата звернення: 26.05.2022).
15. What are convolutional neural networks?. *IBM - Deutschland | IBM*. URL: https://www.ibm.com/cloud/learn/convolutional-neural-networks#toc-what-are-c-MWGVhUiG (дата звернення: 26.05.2022).
16. What is Python used for? A beginner's guide. *Coursera*. URL: https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python (дата звернення: 26.05.2022).
17. NIST special database 19. *NIST*. URL: https://www.nist.gov/srd/nist-special-database-19 (дата звернення: 26.05.2022).
18. Angular - introduction to Angular concepts. *Angular*. URL: https://angular.io/guide/architecture (дата звернення: 26.05.2022).
19. GitHub - sashasova1/mnist-web-app-master: handwritten character recognition. bachelor thesis. *GitHub*. URL: https://github.com/sashasova1/mnist-web-app-master (дата звернення: 26.05.2022).
20. Законодавство України про охорону праці // Збірник нормативних документів у 4 т. -К.: Держнаглядохоронпраці; Основа, 2006 р.

21. Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями. *Офіційний вебпортал парламенту України*. URL: http://zakon3.rada.gov.ua/laws/show/z0508-18 (дата звернення: 26.05.2022).
22. Про затвердження Правил охорони праці під час експлуатації електронно-обчислювальних машин (НПАОП 0.00-1.31-99). *Офіційний вебпортал парламенту України*. URL: http://zakon3.rada.gov.ua/laws/show/z0382-99 (дата звернення: 26.05.2022).
23. ДСТУ 2293-99 Охорона праці. Терміни та визначення основних понять. Київ -1999 р.
24. Москальова В. М. Основи охорони праці. Підручник. - Київ: ВД Професіонал, 2005. - 666 с.
25. Гандзюк М. П., Желібо Е. П., Халімовський М. О. Основи охорони праці / За ред.. Гандзюка М. П. - К.: Каравела 2003 - 405 с.
26. Ткачук К. Н., Халімовський М. О., Зацарний В.В., та інші. Основи охорони праці: Підручник. - К.: Основа, 2006. - 444 с.
27. Жидецький В.Ц. Основи охорони праці: Підручник. - К.: Основа, 2002. - 320 с.
28. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин ДСанПІН 3.3.2.007-98. *Нормативно-директивні документи МОЗ України*. URL: http://mozdocs.kiev.ua/view.php?id=2445 (дата звернення: 26.05.2022).
29. СНиП II-4-79. Природне і штучне освітлення. URL: https://dnaop.com/html/45036/doc- СНиП_II-4-79_ (дата звернення: 26.05.2022).
30. СНиП 2.09.04.-87. Адміністративні і побутові будівлі. URL: https://dnaop.com/html/54074/doc -СНиП_2.09.04-87_ (дата звернення: 26.05.2022).

31. ДСН 3.3.6.042-99. Санітарні норми мікроклімату виробничих приміщень. URL: https://dnaop.com/html/34094/doc -ДСН_3.3.6.042-99 (дата звернення: 26.05.2022).
32. ГОСТ 12.1.005-88.ССБП. URL: http://docs.cntd.ru/document/1200003608 (дата звернення: 27.05.2022).
33. СН 4088-86. Санітарні норми мікроклімату виробничих приміщень. URL: http://docs.cntd.ru/document/901710059 (дата звернення: 27.05.2022).
34. Санітарно – гігієнічним нормам №2152-80. URL: https://dnaop.com/html/2296/doc -ГН_2152-80 (дата звернення: 27.05.2022).
35. СН 4557-88. Санітарні норми ультрафіолетового випромінювання. URL: https://dnaop.com/html/2299/doc -СН_4557-88 (дата звернення: 27.05.2022).
36. Про забезпечення санітарного та епідемічного благополуччя населення. *Офіційний вебпортал парламенту України*. URL: http://zakon2.rada.gov.ua/laws/show/4004-12 (дата звернення: 27.05.2022).
37. MNIST handwritten digit recognition in Keras. *Nextjournal*. URL: https://nextjournal.com/gkoehler/digit-recognition-with-keras (дата звернення: 27.05.2022).
38. Recognizing digits using tensorflow.js in Google Chrome. *Gogul Ilango*. URL: https://gogul.dev/software/digit-recognizer-tf-js (дата звернення: 27.05.2022).
39. Ethem Alpaydin (2004). Introduction to Machine Learning, MIT Press, ISBN 978-0-262-01243-0.
40. Christopher Bishop (1995). Neural Networks for Pattern Recognition, Oxford University Press. ISBN 0-19-853864-2.
41. Stuart Russell & Peter Norvig, (2002). Artificial Intelligence – A Modern Approach. Prentice Hall, ISBN 0-136-04259-7.