Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка інформаційної системи для розпізнавання рукописного тексту з використанням нейронних мереж

Виконав: студент IV курсу, групи СН-43
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Ясюк А.А.
(підпис) (прізвище та ініціали)

Керівник _____ Липак Г.І.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Липак Г.І.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«_27_» _січня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Ясюк Ангеліна Андріївна
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка інформаційної системи для розпізнавання рукописного тексту з використанням нейронних мереж

Керівник роботи Липак Галина Ігорівна, канд. соц. ком., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*07* « *травня* 2025 року № *4/7-444*

2. Термін подання студентом завершеної роботи 21 червня 2025 р.

3. Вихідні дані до роботи Завдання на кваліфікаційну роботу, наукові літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області. 1.1. Поточний стан в галузі розпізнавання тексту. 1.2. Порівняння архітектур нейронних мереж для розпізнавання рукописного тексту. 1.3. Визначення функціональних вимог до моделі розпізнавання тексту. 1.4. Висновки до першого розділу. 2. Теоретичні основи розпізнавання рукописного тексту. 2.1. Відмінності рукописного і друкованого текстів. 2.2. Розвиток технологій розпізнавання текстів. 2.3. Процес навчання моделі. 2.4. Застосування згорткових нейронних мереж для розпізнавання символів. 2.5. Висновки до другого розділу. 3. Практична реалізація системи розпізнавання рукописних символів. 3.1. Інструменти, використані в роботі. 3.2. Модель для розпізнавання символів. 3.3. Висновки до третього розділу. 4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин В.С., доцент	12.06.2024	15.06.2024

7. Дата видачі завдання 27 січня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	30.01.2025	*Виконано*
2.	Підбір джерел про розпізнавання рукописного тексту	31.01.2025-03.02.2025	*Виконано*
3.	Опрацювання джерел по темі кваліфікаційної роботи	04.02.2025-06.02.2025	*Виконано*
4.	Виконання дослідження щодо розпізнавання	07.02.2025-11.02.2025	*Виконано*
	рукописного тексту		
5.	Оформлення розділу «Аналіз предметної області»	15.05.2025-25.05.2025	*Виконано*
6.	Оформлення розділу «Теоретичні основи	26.05.2025-31.05.2025	*Виконано*
	розпізнавання рукописного тексту»		
7.	Оформлення розділу «Практична реалізація системи	01.06.2025-07.06.2025	*Виконано*
	розпізнавання рукописних символів»		
8.	Виконання завдання до підрозділу «Безпека життєдіяльності»	08.06.2025-10.06.2025	*Виконано*
9.	Виконання завдання до підрозділу «Основи охорони	11.06.2025-12.06.2025	
	праці»		
10.	Оформлення кваліфікаційної роботи	13.06.2025-14.06.2025	*Виконано*
11.	Нормоконтроль	15.06.2025-16.06.2025	
12.	Перевірка на плагіат	20.06.2025	
13.	Попередній захист кваліфікаційної роботи	21.06.2025	
14.	Захист кваліфікаційної роботи	29.06.2025	

Студент _____ Ясюк А.А.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Липак Г.І.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка інформаційної системи для розпізнавання рукописного тексту з використанням нейронних мереж // Кваліфікаційна робота освітнього рівня «Бакалавр» // Ясюк Ангеліна Андріївна // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СН-43 // Тернопіль, 2024 // С. – 65, рис. – 10, табл. – 2, додат. – 1, бібліогр. – 46.

Ключові слова: штучний інтелект, глибинне навчання, згорткові нейронні мережі, розпізнавання рукописного тексту, обробка зображень, рукописний текст.

В цій кваліфікаційній роботі досліджується застосування згорткових нейронних мереж для розпізнавання рукописного тексту. Метою даної роботи була розробка моделі глибинного навчання, яка буде класифікувати рукописні символи. У роботі були вивчені сучасні методи обробки тексту з застосуванням нейронних мереж.

Реалізована модель відповідає поставленій меті і здатна класифікувати рукописні символи.

ANNOTATION

Development of an Information System for Handwritten Text Recognition Using Neural Networks // Qualification work of the educational level «Bachelor» // Yasiuk Anhelina // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SN-43 // Ternopil, 2024 // P. – 65, fig. – 10, tabl. – 2, annexes. – 1, references – 46.

Keywords: artificial intelligent, deep learning, convolutional neural networks, hardwritten text recognition, image processing, hardwritten text.

This thesis explores the application of convolutional neural networks for handwritten text recognition. The aim of this work was to develop a deep learning model that would classify handwritten characters. The thesis examined modern methods of text processing using neural networks.

The implemented model meets the set goal and is capable of classifying handwritten characters.

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

CNN (з англ. Convolutional Neural Network) – Згорткова нейронна мережа;

CRNN (з англ. Convolutional Recurrent Neural Network) – Згортково-рекурентна нейронна мережа;

CTC (з англ. Connectionist Temporal Classification) – Коннекціоністська часова класифікація;

GPU (з англ. Graphics Processing Unit) – Графічний процесор;

HTR (з англ. Handwritten Text Recognition) – Розпізнавання рукописного тексту;

LSTM (з англ. Long Short-Term Memory) – Довготривала короткочасна пам’ять;

OCR (з англ. Optical Character Recognition) – Оптичне розпізнавання тексту;

ReLU (з англ. Rectified Linear Unit) – Зрізаний лінійний вузол, функція втрат;

RNN (з англ. Recurrent Neural Network) – Рекурентна нейронна мережа;

ШІ – Штучний інтелект.

ЗМІСТ

ВСТУП

Актуальність теми. Результатом глобалізації з її прискореним розвитком обробки інформації стала підвищена потреба в автоматизації обробки даних. Автоматизація процесу розпізнавання рукописного тексту стала важливою операцією в економіці, оскільки вона дозволить значно пришвидшити швидкість та ефективність обробки інформації, поданої в текстовому вигляді [1].

Через це розробка інформаційної системи розпізнавання рукописного тексту з використанням нейронних мереж є актуальним напрямом досліджень в галузях комп'ютерного зору та глибинного навчання. Дані дослідження закладуть базу для підвищення автоматизації та ефективності обробки даних у багатьох галузях, таких як освіта, охорона здоров'я, фінансах і т.д, а також відкриє нові можливості для застосування передових технологій у різноманітних галузях.

Мета і задачі дослідження. Метою даної бакалаврської роботи є розробка інформаційної системи розпізнавання рукописного тексту на основі використання нейронних мереж для того, щоб зробити обробку інформації, поданої у вигляді рукописного тексту, більш ефективною та автоматизованою. Для досягнення такої мети необхідно вирішити ряд завдань, а саме:

– Провести аналіз стану досліджень у сфері розпізнавання рукописного тексту;

– Ознайомитися з сучасними підходами до реалізації нейронних мереж для задач розпізнавання тексту;

– Визначити вимоги до функціоналу та архітектури інформаційної системи;

– Розробити модель нейронної мережі для розпізнавання рукописного тексту.

Практичне значення одержаних результатів

Отримані в результаті даного дослідження результати мають велике практичне застосування, оскільки вони можуть сприяти розвитку галузі автоматичного розпізнавання рукописних текстів.

Отримана інформаційна система може бути використана в широкому спектрі галузей, де автоматизація читання даних з рукописних текстів дозволить значно прискорити час обробки та зменшити шанс людської помилки.

У освіті технологія полегшує оцифрування рукописних матеріалів, таких як конспекти лекцій або рукописні нотатки, полегшуючи їх зберігання та доступ до них студентам і викладачам. В охороні здоров'я технологія допомагає обробляти рукописні історії хвороби і рецепти, роблячи заповнення медичних форм швидшим і точнішим. Ця доступність, таким чином, підвищує зручність і ефективність життя людей [2].

В музейній справі дана система може застосовуватися для оцифрування рукописних артефактів, що надасть доступ до історичних пам’яток широкому загалу людей [3].

Фінансова індустрія може використовувати систему для полегшення автоматизації обробки чеків, рахунків та інших документів, скорочуючи витрати і час. У наукових дослідженнях технологія уможливлює обробку рукописних даних і документів, що є корисним для істориків і дослідників.

Її впровадження має значний економічний і соціальний вплив завдяки підвищенню продуктивності праці, зниженню витрат і покращенню якості обслуговування [4].

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поточний стан в галузі розпізнавання тексту

Розпізнавання рукописного тексту є однією з ключових задач комп’ютерного зору, яка активно розвивається протягом останніх десятиліть. Застосування технологій OCR охоплює широке коло сфер: від цифровізації архівів, сканування рукописних та друкованих документів до обробки фотографій, номерних знаків і зображень у реальному часі. Завдяки стрімкому розвитку алгоритмів машинного навчання та глибинних нейронних мереж, точність таких систем наблизилася до рівня людського ока.

На початковому етапі розвитку OCR-системи здебільшого базувалися на класичних методах обробки зображень, евристичних алгоритмах і шаблонах. Такі системи працювали задовільно з друкованими текстами у добре структурованих форматах, однак їх ефективність значно знижувалася у випадках спотворених, рукописних або низькоякісних документів. Яскравим прикладом є система Tesseract у версіях до 3.0, яка широко використовувалася, однак покладалася на вручну сконструйовані ознаки. Інші приклади включають ABBYY FineReader, OmniPage та Readiris, що були орієнтовані переважно на обробку сканованих сторінок і мали обмеження при роботі з рукописним текстом або складними зображеннями.

Поява глибоких згорткових нейронних мереж [5] кардинально змінила підхід до OCR. Ці мережі автоматично витягують ознаки зображень без необхідності ручного налаштування, демонструючи стійкість до шуму, викривлень і неідеальних умов зйомки. Значним проривом стало поєднання CNN з рекурентними нейронними мережами у так званих CRNN-моделях. Такий підхід дозволив одночасно працювати як з просторовими, так і з послідовними характеристиками тексту. Крім того, функція втрат CTC

дозволила уникнути попередньої сегментації тексту на символи, що спростило побудову системи розпізнавання.

На основі цих моделей було створено низку сучасних open-source та комерційних рішень. Наприклад, оновлений Tesseract 4+ інтегрував LSTM-архітектуру [6], значно покращивши точність розпізнавання. EasyOCR — бібліотека з відкритим кодом, яка підтримує понад 80 мов та орієнтована на багатомовність. PaddleOCR від Baidu забезпечує розпізнавання тексту навіть у складних випадках, а Calamari OCR дозволяє створювати ансамблі моделей з можливістю кастомного навчання. Серед хмарних сервісів варто виокремити Google Cloud Vision OCR, Microsoft Azure OCR, Amazon Textract та Adobe Sensei, які дозволяють інтегрувати OCR у більші системи без потреби в локальних обчисленнях.

Окрему увагу заслуговує сегмент розпізнавання рукописного тексту. Це найбільш складний підрозділ OCR, оскільки почерк людини має велику кількість варіацій, а також в ньому часто бувають відсутні чіткі розмежування між літерами. Для розпізнавання рукопису застосовуються ті самі архітектури CRNN + CTC [7], однак із залученням спеціалізованих датасетів. Google Cloud Vision, MyScript, Transkribus є прикладами систем, які вже працюють із рукописним введенням.

У сучасних дослідженнях активно розвиваються нові архітектури, зокрема моделі-трансформери. Прикладом є TrOCR [8], який об'єднує зорову та мовну частину і досягає високих результатів у задачах без попереднього навчання. Поширюється також підхід Multilingual OCR, де моделі здатні одночасно розпізнавати текст кількома мовами. Іншим перспективним напрямком є zero-shot OCR — здатність моделі розпізнавати нові мови або шрифти без додаткового процесу навчання, що особливо важливо для рідкісних мов.

Інтеграція з мовними моделями також дозволяє покращити кінцеву якість тексту завдяки автокорекції, контекстному передбаченню або перевірці граматики. Це стає особливо важливим при обробці великого масиву

документів, де навіть незначні помилки будуть накопичуватися, значно погіршуючи кінцевий результат.

Таким чином, сучасний стан у галузі OCR демонструє глибоку інтеграцію комп’ютерного зору з методами глибокого навчання та обробки природної мови. Хоча технологія OCR значно просунулась уперед, ця галузь все одно ще потребує уваги дослідників. Для повноцінного використання OCR у локалізованих інформаційних системах необхідно створювати нові датасети, адаптувати архітектури моделей та налагоджувати відповідні процеси обробки даних.

1.2 Порівняння архітектур нейронних мереж для розпізнавання рукописного тексту

Розпізнавання рукописного тексту є складним завданням через широку варіативність почерків, відсутність чітких меж між символами та спотворення, які часто зустрічаються на зображеннях. Для вирішення цієї проблеми використовуються різні архітектури глибокого навчання, серед яких найпоширенішими є CNN, CRNN та трансформери. У цьому підрозділі розглядається їхній принцип роботи, переваги та недоліки, а також те, як їх можна застосувати для обробки рукописного тексту.

1.2.1 Згорткові нейронні мережі

Згорткові нейронні мережі традиційно використовуються для обробки зображень, оскільки вони здатні ефективно виявити просторові патерни. У контексті розпізнавання рукописного тексту CNN використовуються для вилучення візуальних ознак із вхідного зображення тексту, які потім передаються наступним модуля.

Основні переваги CNN:

- Висока ефективність у виділенні локальних ознак (контури, лінії, криві) зі зображення тексту.
- Низька кількість параметрів у порівнянні з повнозв'язними мережами, що знижує ризик перенавчання.
- Можливість застосування попередньо натренованих моделей на великих наборах даних для покращення результатів.

Недоліки:

- CNN не враховують послідовність символів — вони працюють лише з фіксованими зображеннями.
- Для повноцінного розпізнавання тексту їх потрібно поєднувати з іншими архітектурами (наприклад, з RNN або трансформаторами).

1.2.2 Згорткові рекурентні нейронні мережі

Згортково-рекурентні нейронні мережі - це комбінація згорткової і рекурентної нейронної мережі, яка поєднує переваги просторового і послідовного аналізу. Після того, як CNN виокремлює ознаки із зображення, RNN обробляє ці ознаки як послідовність, дозволяючи моделі зрозуміти порядок символів. Зазвичай CNN використовує функцію втрат CTC для зіставлення послідовності символів з вхідними даними [9].

Основні переваги:

- Забезпечує високу точність розпізнавання без необхідності попередньої сегментації символів.
- Добре працює з неструктурованими або викривленими зображеннями рукопису.
- Сумісність з різними алфавітами та мовами, включно з українською.

Недоліки:

- Висока обчислювальна складність та повільне навчання через послідовну природу RNN.

- Потребує ретельного підбору гіперпараметрів для ефективного функціонування.

1.2.3 Моделі на базі трансформерів

Трансформери поступово входять у сферу HTR, замінюючи RNN-компоненти. Вони дозволяють моделі повністю аналізувати контекст усієї послідовності завдяки механізму self-attention. У TrOCR, наприклад, використовується візуальний енкодер і мовний декодер на основі архітектури трансформера, що дозволяє безпосередньо перетворювати зображення на текст [10]. На рис. 2.1 представлено візуалізацію роботи моделі-трансформера для розпізнавання зображення.

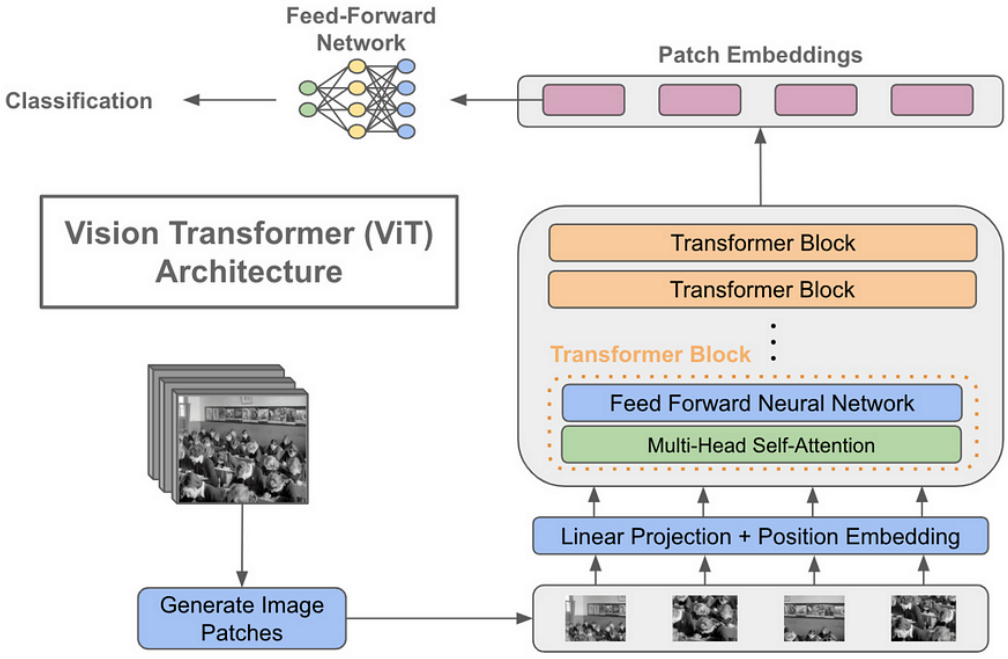

Рисунок 2.3 – Розпізнавання зображення з допомогою трансформера

Основні переваги:

- Паралельна обробка, що значно пришвидшує тренування у порівнянні з RNN.
- Вища точність розпізнавання на довгих послідовностях завдяки глобальному контексту.

- Гнучкість архітектури, що дозволяє її адаптувати до складних задач (мультимовність, генерація).

Недоліки:

- Велика кількість параметрів, що потребує потужних обчислювальних GPU.
- Потребує великих наборів даних для повноцінного навчання.
- Порівняно нова технологія для HTR, що ще не має такої широкої підтримки у відкритому доступі, як CRNN.

У таблиці 1.1 зображено коротке порівняння характеристик даних архітектур.

Таблиця 1.1 – Порівняння моделей мереж

Характеристика	**CNN**	**CRNN**	**Трансформери**
Тип даних	Зображення	Зображення + послідовність	Зображення + послідовність
Необхідність сегментації символів	Так	Ні	Ні
Запам’ятовування контексту	Локальне	Коротко- та довготривале	Повний контекст
Потреба в ресурсах	Помірна	Помірно-висока	Висока
Точність на складних рукописах	Низька-середня	Висока	Дуже висока

1.3 Визначення функціональних вимог до моделі розпізнавання тексту

У цій роботі була реалізована модель штучного інтелекту, здатна розпізнавати рукописні літери англійського алфавіту. В її основі — згорткова нейронна мережа, яка імітує роботу людського зору: вона «дивиться» на зображення і намагається зрозуміти, яка саме літера на ньому зображена. Це складне завдання, оскільки кожна літера може бути написана по-різному, в різному стилі або нерозбірливо.

Мережа тренувалася на наборі зображень EMNIST [11]. У цьому наборі кожне зображення містить одну окрему букву, написану від руки, а також мітку — тобто інформацію про те, яку саме букву вона зображає. На основі цих прикладів мережа поступово вчилася розпізнавати шаблони та особливості, притаманні кожній літері.

Для кращої роботи моделі використовувалась спеціальна підготовка даних: кожне зображення було приведене до єдиного формату та масштабу, а також до нього застосовувалися дрібні випадкові зміни (наприклад, легке обертання чи зсув), щоб мережі не «запам’ятовувала» картинки, а дійсно розпізнавати унікальні ознаки символів. Такий підхід дозволив досягти більшої гнучкості та точності при розпізнаванні нових, незнайомих прикладів.

Побудована модель має кілька рівнів обробки: на перших етапах вона виділяє основні риси зображення (лінії, контури), потім — складніші поєднання форм, а на завершальному етапі приймає рішення на основі унікальних ознак, наявних на зображенні, як саме класифікувати його. Візуалізація цього процесу представлена на рис. 2.2. На виході мережа видає ймовірності для кожної з 26 літер, і як відповідь обирається та, яка має найвищу ймовірність.

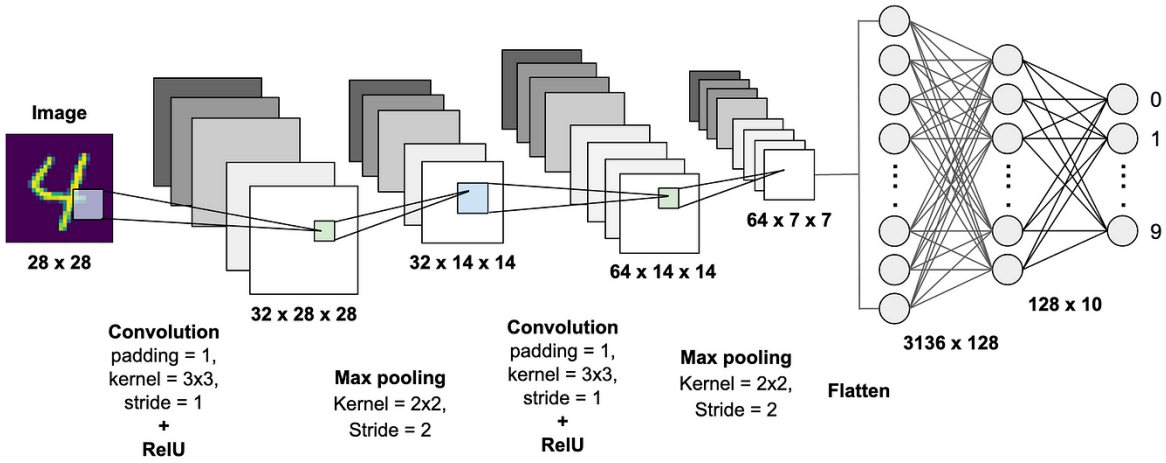

Рисунок 2.2 – Принцип розпізнавання зображень з допомогою CNN

Протягом навчання постійно відстежувалась точність роботи моделі на тестових зображеннях, які не використовувалися безпосередньо в процесі навчання. Це дозволяло уникнутих ситуації, коли мережа «запам’ятовує» навчальні зображення, замість того щоб чесно класифікувати символи[12]. Найкраща версія моделі автоматично зберігалась для подальшого використання.

Після завершення навчання модель була протестована на незалежному наборі зображень. Результати показали високу точність — це означає, що мережа успішно навчилася впізнавати більшість літер навіть тоді, коли вони написані інакше, ніж у прикладах, на яких вона вчилася.

Загалом реалізована система демонструє те, як сучасні технології штучного інтелекту можуть бути використані для автоматичного розпізнавання рукописного тексту.

1.4 Висновки до першого розділу

В першому розділі кваліфікаційної роботи:

– Розглянуто предметну область розпізнавання тексту, зокрема оптичне розпізнавання символів та рукописного тексту. Описано особливості задачі, а також основні підходи, які застосовуються для її вирішення;

– Проаналізовано найпоширеніші архітектури нейронних мереж, які застосовуються в даній області, такі як згорткові нейронні мережі, моделі-трансформери, а також комбінації CNN та RNN;

– На основі проведеного аналізу обрано архітектуру мережі, яка буде застосовуватися в даній роботі. Мною було обрано згорткову нейронну мережу, оскільки вона показує високу ефективність в класифікації зображень.

– Визначено функціональні вимоги для розроблюваної нейронної мережі, такі як підтримка класифікації зображень розміром 28 на 28, збереження моделі для подальшого використання та інші.

РОЗДІЛ 2. ТЕОРЕТИЧНІ ОСНОВИ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ

2.1 Відмінності рукописного і друкованого текстів

Рукописний і друкований текст принципово відрізняються за своєю природою і характеристиками, що впливає на процес машинного розпізнавання текстів. Рукописний є продуктом індивідуальної моторики людини і тому відрізняються високою варіативністю форми, нахилу, розміру, товщини ліній, їх з'єднання, а також інших видимих ознак. Друкований текст же - машинний результат типографських процесів, де кожна літера має чітку форму, розмір і відстань між символами. Саме це впливає на складність процесу розпізнавання: в той час як машинне зчитування друкованих літер зазвичай є точним і безпомилковим, рукописний почерк може бути прочитаний лише з допомогою алгоритмів високої складності, здатних впоратися з високим рівнем індивідуальних варіацій. Щоб проілюструвати ці відмінності, на рисунку 2.1 показано приклади рукописних і друкованих літер.

Рисунок 2.1 – Різниця між друкованим і рукописним варіантами деяких літер

Основна відмінність полягає в тому, що не існує єдиної нормальної форми тексту, яким пишуть люди. У той час як шрифти в поліграфії виглядають як чітко окреслені векторні фігури, символи в рукописному тексті є продуктом живого руху руки, і, відповідно, будуть відрізнятися навіть у різних авторів залежно від темпераменті, швидкості письма або інструменту, яким вони послуговуються. Літера «а» в друкованому вигляді завжди буде однакової форми в будь-якому шрифті, в той час як на письмі вона буде відкритою або закритою, закругленою або ламаною, нахиленою вправо або вліво, з продовженнями або без них. Це також ускладнює системи розпізнавання, які повинні бути адаптивними для розпізнавання характеру символів незалежно від його зовнішньої форми [13].

Іншою фундаментальною відмінністю є наявність міжсимвольних зв'язків між символами, особливо в скорописі. У друкованому тексті кожен символ є самостійним елементом з чіткими краями, а в почерку символи зливаються в єдину контурну лінію. Це ускладнює процес сегментації - розділення зображення тексту на окремі символи з метою розпізнавання - і в деяких випадках системі потрібно розпізнати форму щоб зрозуміти яке ж слово написане, використовуючи контекстну інформацію або статистичні ймовірності [14].

Крім того, почерк має елементи індивідуального стилю, яких бракує друкованій формі. Це можуть бути надмірне або недостатнє розтягування символів у кривих, нерівномірність розміщення літер і рядків, розмашистість, а також пропуски та правки, зроблені в процесі письма. На відміну від типографічного друку, де артефакти майже відсутні, рукописний текст може бути спотворений шорсткістю паперу, патьоками чорнила, плямами, чи тремтінням руки. Все це робить рукописний текст таким складним об'єктом аналізу.

Рукописний текст різниться в різних системами письма і типах почерку - окрім дитячого, каліграфічного чи медичного - і це додає йому різноманітності. Такий почерк можна прочитати лише в тих випадках, коли вдивляєшся не лише

у форму літер, а й у інші сигнали, які дають певну підказку про те, що написано в цьому місці. Друковане письмо менш вимогливе до такої підтримки у вигляді розпізнавання, оскільки помилка в розташуванні символів або у формуванні символів малоймовірна.

Таким чином, основними відмінностями машинописного письма від рукописного є ступінь стандартизації, варіативність форм літер, наявність міжлітерних залежностей, рівень спотворень, залежність від інтерпретації навколишнього середовища. Всі ці обставини досить сильно впливають на вибір методів і алгоритмів автоматичного зчитування: в той час як машинопис легко зчитується в рамках стандартних систем розпізнавання, рукописний почерк вимагає застосування більш глибоких моделей згорткових нейронних мереж, рекурентних моделей чи трансформаторів, здатних обробляти складніші патерни і пристосовуватися до різних форм написання [15].

2.2 Розвиток технологій розпізнавання текстів

Процес розпізнавання рукописного тексту є одним з найнагальніших проблем в сферах комп’ютерного зору та обробки природньої мови, який мав визначне значення для досліджень та повсякденного застосування. В порівнянні з друкованим текстом рукописний має значно вищу варіативність стилів, розмірів і нахилів символів, та інших особливостей, тому в даній галузі завдання класифікації було ще складнішим, оскільки модель повинна працювати з дуже великою кількістю різноманітних ознак. І через це розробка ефективних методів розпізнавання рукописних символів здійснюється за допомогою гнучких і потужних алгоритмів, які також здатні обробляти неструктуровані дані у вигляді зображень [16].

Раніше розпізнавання тексту здійснювалося за допомогою традиційних методів штучного інтелекту, які включали зіставлення шаблонів, евристичні правила та статистичні методи [17], такі як аналіз головних компонентів, машини опорних векторів або байєсівські класифікатори [18]. Але вони

вимагали ручної обробки інформації на вході, а також були мало масштабованими та схильними до шуму та спотворення. З еволюцією машинного навчання, а ще більше після впровадження глибоких нейронних мереж, стало можливим навчати системи безпосередньо на зображеннях без попереднього витягання ознак. CNN здатні автоматично вивчати корисні шаблони в растровому представленні символів, що значно підвищило точність класифікації.

У сучасних дослідженнях рукописні символи та слова розглядаються як цілі файли, що подаються в багатошарові згорткові нейронні мережі з фільтром, дискретизацією, нормалізацією та повністю з'єднаними шарами. Цей метод виявився практично нечутливим до шуму, спотворень та варіацій розміру літер. Однією з переваг глибинного навчання в даному випадку є те, що модель здатна навчатися на численних прикладах і узагальнювати закономірності, виявлені в різних написах [19].

Передумов для побудови таких моделей є наявність високоякісних наборів даних із значним обсягом рукописних символів в цифровому форматі та мітками. У цій роботі ми використовували набір даних EMNIST, який є розширенням стандартного набору даних MNIST, що, крім рукописних цифр, включає латинські літери. EMNIST пропонує стандартизований набір даних із чорно-білих зображень розміром 28×28 пікселів, щоб згорткові мережі могли бути застосовані для їх класифікації. EMNIST може обробляти 26 літер, 62 символи або весь набір літер із розрізненням регістру, залежно від підмножини, тому він є досить гнучким для навчання моделей із різним рівнем складності [20].

Для класифікації символів у проекті використовується архітектура згорткової нейронної мережі, спеціально створеної для розмірності зображень EMNIST. Після попередньої обробки даних (нормалізація, масштабування та перетворення у тензорний формат) модель навчається на категоріальній функції втрат та оптимізаторі Adam [21]. Така конфігурація дозволяє досягти високої точності класифікації на тестувальному наборі за відносно короткий час. Після

навчання модель зберігається у форматі TensorFlow для подальшого використання в реальних додатках або включення в інтерфейси розпізнавання тексту. На рис. 2.2 наведено блок-схему процесу розпізнавання символу нашою моделлю.

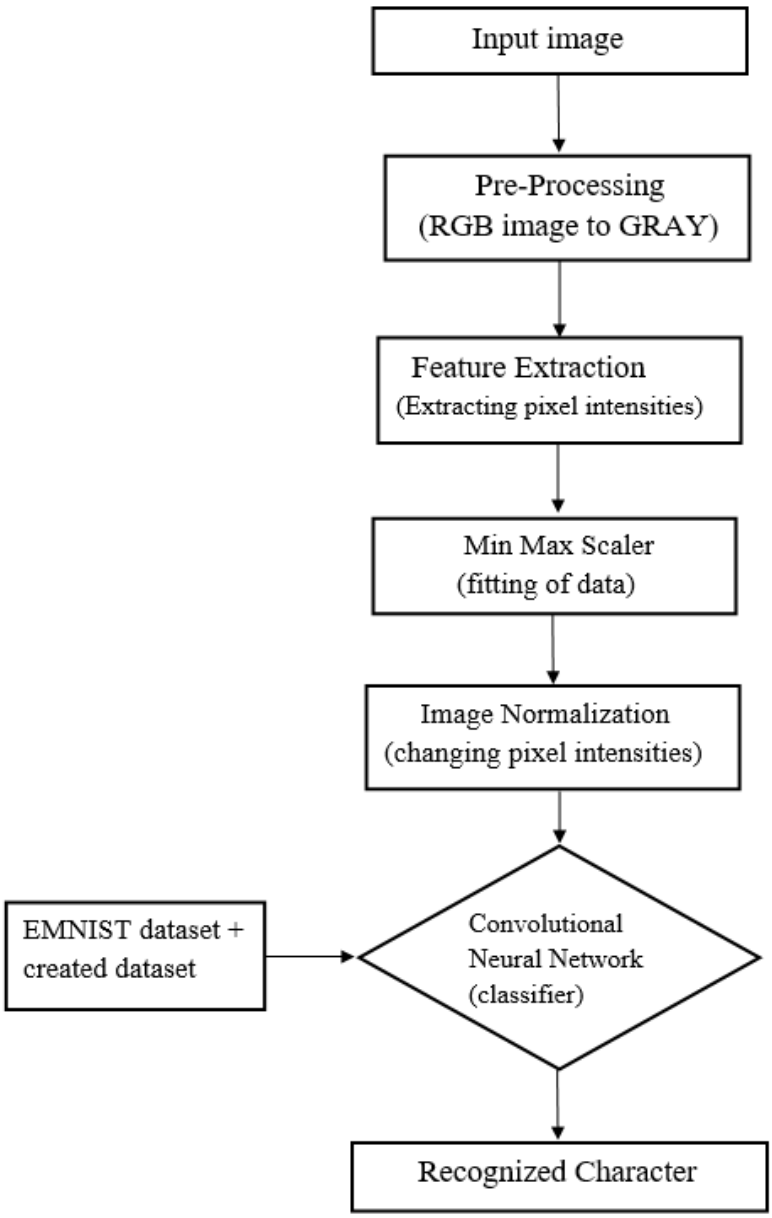

Рисунок 2.2 – Блок-схема процесу розпізнавання символів

Отже, напрям дослідження в цій роботі — це сучасні рішення для розпізнавання рукописних англійських символів з використанням сучасних

підходів глибокого навчання та відкритих наборів даних. Запропоноване рішення має потенціал для майбутнього розширення з метою виконання завдань розпізнавання символів, що є важливим при роботі над збереженням артефактів культурної спадщини [22].

2.3 Процес навчання моделі

Наша модель навчається шляхом прогону через велику кількість навчальних зображень, з яких вона буде поступово витягувати ознаки для класифікації різних символів. На відміну від традиційних алгоритмів, які вимагають ручного задання правил, сучасні моделі самі формують ці правила, знаходячи закономірності між вхідними даними і міткою. Це схоже на те, як щколярі навчаються писати і читати, порівнюючи свій результат з прикладом, знаходячи помилки й відмінності [23].

Секрет навчання полягає в тому, щоб згодувати мережі велику кількість промаркованих вхідних зображень. У кожному з них зображення символу супроводжується відповідною міткою — інформацією про те, якому символу воно відповідає. Це вчить систему інтерпретувати те, що вона бачить. Модель створює внутрішні концепції того, як виглядає окрема літера, та адаптується до нових, раніше не бачених форм. Потрохи вона отримує здатність класифікувати символи, яких ніколи раніше не бачила, якщо ці символи є близькими до тих, що використовувалися в процесі її навчання [24].

Цей процес передбачає постійне вдосконалення. Спочатку модель буде робити багато помилок, але кожна з них через процес зворотнього поширення похибки [25] лише покращить точність моделі, що дозволить системі оновлювати свою «думку» про те, як виглядає той чи інший символ. Оскільки таких ітерацій відбувається багато, модель стає все більш детальною, ідеалізованою і менш схильною до впливу нерелевантних варіацій, таких як зображення низької якості або дивні літери.

Розпізнавання символів є невід'ємною складовою більших систем розпізнавання тексту або мови. Це будівельний блок, на якому можна побудувати просунуті системи, що розпізнають цілі слова, речення або тексти. Отже, успіх у навчанні моделі розпізнавання символів є ключем до побудови гнучких і адаптивних систем машинного читання, здатних самостійно інтерпретувати візуальну інформацію, яка до недавнього часу була доступна лише живим істотам [26].

Функціональна система розпізнавання символів не просто вирішує актуальну проблему, а вказує напрямок розвитку сучасних інтелектуальних алгоритмів з великими наборами даних. Це все більш важливий напрямок в інформаційній епосі, коли інформація зазвичай зберігається або отримується в графічній формі і повинна швидко інтерпретуватися з високою точністю. У такій ситуації навчання моделі розпізнавання символів вже не є інженерною діяльністю, а спрямовано на створення систем, які навчаються самостійно розпізнавати видимі дані в реальному світі [27].

.

2.4 Застосування згорткових нейронних мереж для розпізнавання символів

Згорткові нейронні мережі є, мабуть, одними з найважливіших в області глибинного навчання, які здійснили революцію в комп'ютерному зорі. Цей набір моделей оптимізований для обробки зображень і розпізнавання ознак на них таким чином, що зберігається просторова ієрархія вхідних даних. Він базується на концепції локального сприйняття, згідно з якою кожен нейрон реагує лише на певну підмножину вхідних даних, дозволяючи йому розпізнавати типові особливості: краї, кути та текстури, які потім інтегруються в складні візерунки [28].

Основною будівельною одиницею CNN є згортковий шар, який визначає згортку зображення з декількома навченими в процесі оптимізації фільтрами. Представлення даної мережі наведено на рис. 2.3. Кожен із цих шарів здатний

розпізнавати одну особливість, що з'являється в декількох частинах зображення, а результатом обчислення є карта особливостей. Завдяки цій фішці мережа отримує можливість розпізнавання символів незалежно від їхнього розташування на зображенні.

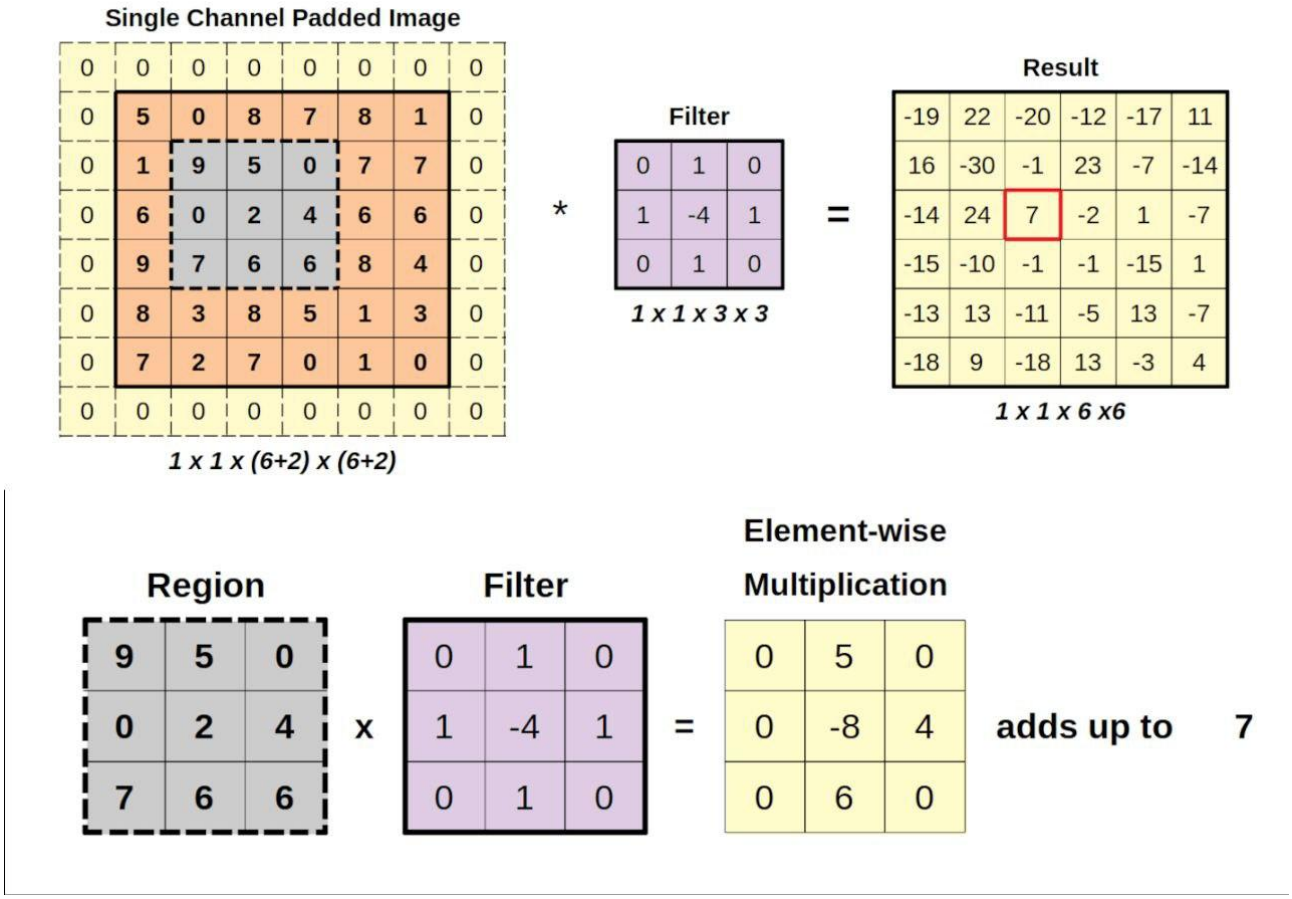

Рисунок 2.3 – Принцип роботи CNN

Після згортки зазвичай додається функція активації, така як ReLU [29], яка вносить елемент нелінійності і дозволяє мережі вивчати складні закономірності між пікселями. Графік цієї функції показано на рис. 2.4. Розмірність простору ознак зазвичай зменшується за допомогою додискретизації, яка вводиться після шарів згортки, щоб зробити мережу інваріантною до перетворення, тобто зберегти максимальне значення в межах кожної області карти.

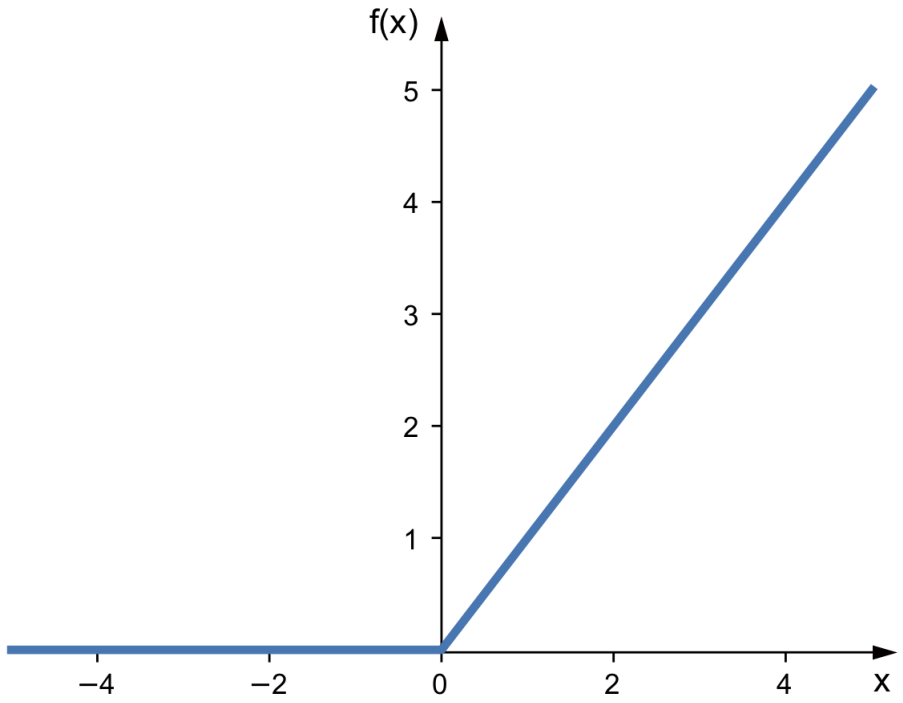

Рисунок 2.4 – Графік функції ReLU

Після кількох шарів згортки та субдискретизації архітектура мережі зазвичай перетворюється на одну або кілька повністю зв'язних архітектур. Останні класифікують, об'єднуючи ознаки, отримані на попередніх рівнях. На даному рівні досягається остаточна класифікація того, що саме представлено на вході. Ця архітектурна ієрархія від вхідних пікселів до високорівневих представлень є причиною високої продуктивності CNN

Згорткові мережі мають певні важливі переваги. По-перше, завдяки спільним вагам та локалізації, кількість параметрів цих типів мереж значно менша порівняно з повністю зв'язаними мережами. По-друге, такі мережі менш чутливі до варіацій об'єктів за формою, розміром або розташуванням. По-третє, CNN здатні навчатися ознакам автоматично, без явного кодування правил розпізнавання [30].

Такі мережі виявилися особливо ефективними в задачах розпізнавання рукописного тексту. Вони розпізнають символи в почерку з високою точністю в різних стилях письма. Крім того, вони не потребують попередньої нормалізації

зображень або ручної сегментації зображення - мережа здатна самостійно визначати форму символу.

Але у них є і свої недоліки. Вони вимагають значних обчислювальних ресурсів, особливо якщо працюють з великими зображеннями або використовують дуже глибокі мережі. Згорткові мережі також потребують великих наборів даних для навчання для нормального функціонування. Існує також проблема інтерпретованості - важко визначити, чому модель вибрала той чи інший варіанти [31].

Але дані обмеження все частіше долаються за допомогою новітніх методів, таких як попередньо навчені моделі, механізми уваги, нормалізація даних та гібридні архітектурі, що використовують поєднання згорткових мереж з іншими типами мереж. CNN є важливим елементом систем розпізнавання рукописного тексту, машинного перекладу рукописного тексту в цифрову форму, додатків для смартфонів, а також серверних пропозицій для обробки документів.

2.5 Висновки до другого розділу

В другому розділі кваліфікаційної роботи:

– Розглянуто відмінності між рукописним і друкованим текстом;

– Описано область розпізнавання рукописного тексту. Розглянути основні проблеми, з якими стикаються дослідники при роботі в даній області;

– Детально досліджено принцип роботи згорткової нейронної мережі при роботі з класифікацією симолів. Обґрунтовано вибір саме цієї архітектури для використання в нашій моделі.

РОЗДІЛ 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПІЗНАВАННЯ РУКОПИСНИХ СИМВОЛІВ

3.1 Інструменти, використані в роботі

В процесі виконання даної бакалаврської роботи було використано низку сучасних програмних засобів, бібліотек та технологій для створення, навчання та тестування нейронної мережі для розпізнавання рукописних символів. Кожен з використаних інструментів відіграв важливу роль у загальній структурі проекту, і саме їх взаємодія дозволила досягти потрібного результату. У цьому розділі наведено детальний огляд кожного використаного інструменту.

Першим і найважливішим інструментом є TensorFlow - потужна бібліотека з відкритим вихідним кодом, розроблена компанією Google. Вона призначена для створення моделей машинного навчання [32]. Однією з головних переваг TensorFlow є її гнучкість: вона підтримує як низькорівневий програмування (через тензорні операції), так і високорівневі API, в тому числі Keras, щоб значно полегшити роботу розробника. У цій роботі Keras використовувався для створення архітектури моделі, задання шарів нейронної мережі, компіляції моделі та налаштування функцій втрат, метрик та оптимізаторів. TensorFlow також забезпечує ефективну обробку даних, а завдяки апаратному прискоренню дозволяє значно скоротити час навчання нашої моделі. Візуалізація принципу роботи даної бібліотеки представлена на рис. 3.1.

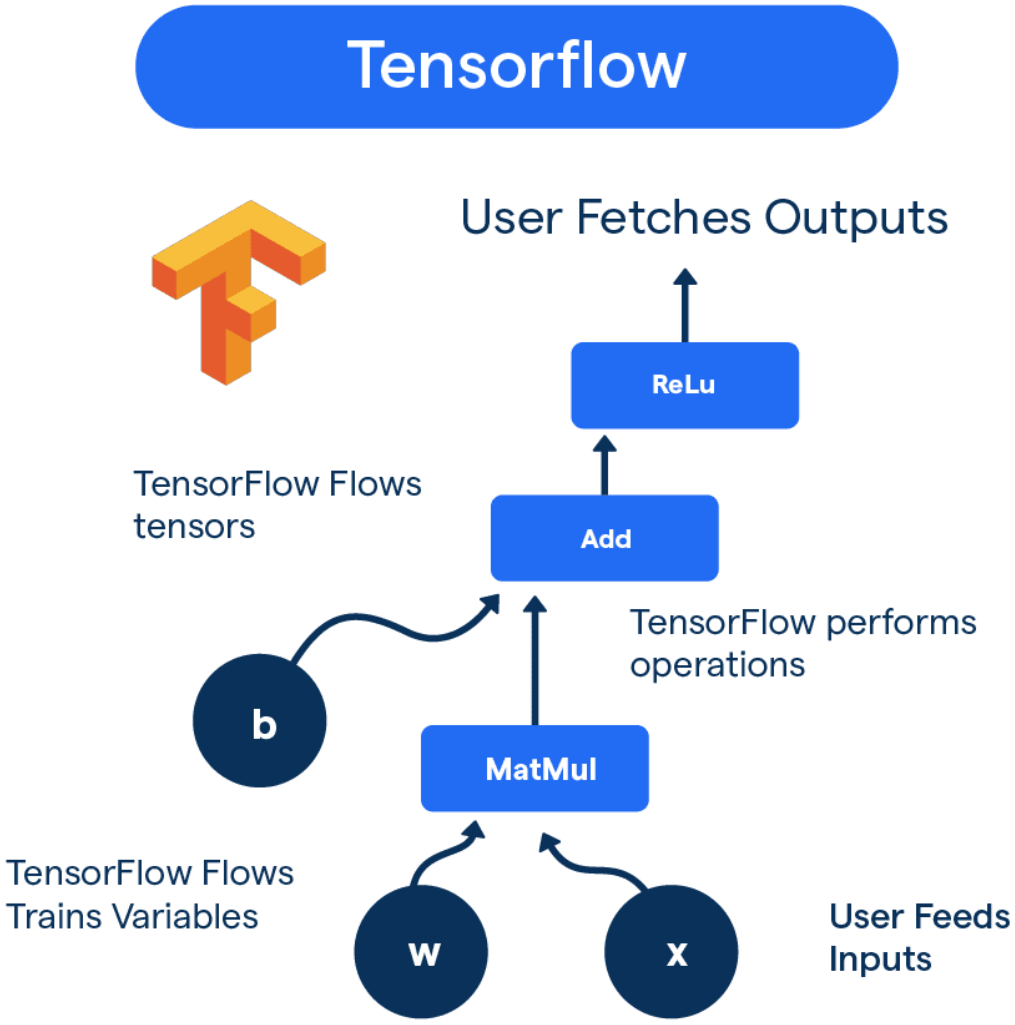

Рисунок 3.1 – Принцип роботи Tensorflow

Інтегрованим рішенням для TensorFlow є Keras API - високорівневий інтерфейс, який абстрагує складну математику, пов'язану з навчанням нейронних мереж, у простий і зрозумілий синтаксис. Замість того, щоб вручну реалізовувати формули, градієнти, зворотнє поширення помилок тощо, розробник може побудувати модель з готових блоків: згортки, повного з'єднання, нормалізації, шарів активації та інших. Це дає можливість не тільки швидко реалізувати функціональну модель, але й експериментувати з її архітектурою, кількістю шарів та іншими гіперпараметрами [33].

Для завантаження та попередньої обробки навчальних даних було використано бібліотеку TensorFlow Datasets. Вона надає простий спосіб

доступу до широкого спектру стандартних наборів даних, включаючи EMNIST. TensorFlow Datasets дозволяє автоматично завантажувати, розпаковувати, готувати та представляти дані у формі, придатній для подальшого використання в моделі. У цьому проекті ми використовували навчальну та тестову частини набору emnist/letters, які містять понад 140 тисяч прикладів літер, написаних різними людьми [34].

Окрім завантаження, важливу роль відіграє попередня обробка даних, для якої ми використовували вбудовані інструменти TensorFlow. Це включало нормалізацію значень пікселі зображень (переведення їх у діапазон від 0 до 1), зміну розмірів тензорів, додавання необхідних розмірностей та інші операції. Це важливо, оскільки нейромережа краще навчається, коли вхідні дані мають однаковий масштаб і формат.

Особливу роль у покращенні якості навчання відіграє аугментація даних - штучне збільшення розміру навчальної вибірки шляхом варіювання наявних прикладів. У даній роботі це було реалізовано за допомогою tf.keras.layers, зокрема шарів випадкового повороту, зсуву та масштабування зображень. Такий підхід робить модель більш стійкою до змін почерку, підвищує її здатність до узагальнення і, як наслідок, покращує точність розпізнавання на нових даних.

Для візуалізації результатів, зразків зображень і динаміки навчання використовували бібліотеку Matplotlib. Вона дозволяє легко створювати графіки, гістограми, зображення тощо. У даній мережі Matplotlib використовувався для відображення прикладів зображень рукописних літер, перевірки якості даних і, за необхідності, побудови графіків точності та втрат під час навчання моделі. Це дозволяє краще зрозуміти поведінку моделі та вчасно відреагувати на потенційні проблеми, такі як перенавчання [35].

У роботі також використовувалися колбеки - спеціальні об'єкти в Keras, які дозволяють автоматизувати важливі процеси під час навчання. Серед них - збереження найкращої моделі на основі результатів на тестовому наборі, автоматична зміна швидкості навчання, якщо немає покращення, та логування

даних для подальшого перегляду в TensorBoard - інтерактивному інструменті для візуалізації процесу навчання нейронної мережі. TensorBoard дозволяє відстежувати метрики (наприклад, точність або втрати), переглядати архітектуру моделі, аналізувати зміни ваг і навіть перевіряти зміни поведінка моделі з часом.

Для запуску та виконання коду ми використовували середовище Jupyter Notebook. Це зручний інструмент для інтерактивної роботи з Python-кодом, особливо популярний у сфері машинного навчання. Він дозволяє об'єднати код, графіки, текстові пояснення та результати виконання в одному документі. Це дає можливість легко документувати кожен крок дослідження, перевіряти проміжні результати та робити висновки без необхідності щоразу запускати програму з нуля. У цій роботі Jupyter був особливо корисним для налагодження моделі, тестування функцій доповнення, візуалізації прикладів і покрокової побудови всієї логіки [36].

Окремо варто згадати механізм апаратного прискорення за допомогою GPU. TensorFlow автоматично визначає доступність графічного процесора і, якщо він вільний, виконує обчислення на ньому. Це надзвичайно важливо для задач глибокого навчання, оскільки скорочує час навчання моделі в кілька разів. Це дає можливість тренуватися на повному обсязі даних з додаванням аугментацій, не витрачаючи на це надто багато часу [37].

Для збереження моделей і подальшого використання ми використовували формат .h5, який дозволяє зберігати як архітектуру моделі, так і її натреновані ваги. Він дозволяє зупинити навчання на будь-якому етапі, а потім відновити збережений стан або використати модель для нових прогнозів.

Все програмне забезпечення, що використовується в проекті, має відкритий вихідний код і є безкоштовним. Це важливо не тільки з точки зору доступності, але й відтворюваності результатів - будь-хто може відтворити результати дослідження, внести зміни, оптимізувати модель або використовувати її в інших додатках. Крім того, всі інструменти добре

задокументовані, підтримуються активними спільнотами розробників і мають регулярні оновлення.

3.2 Модель для розпізнавання символів

Для виконання цього завдання ми вирішили використати існуючі інструменти з відкритим вихідним кодом. Далі ми описуємо фрагменти коду, написані для нашої моделі.

Код, показаний у лістингу 4.1, готує основні параметри та завантажує набір даних, необхідних для навчання нейронної мережі для класифікації зображень. Спочатку встановлюються три базові параметри: BATCH_SIZE = 128 визначає кількість зображень, які модель обробляє одночасно протягом одного кроку навчання; це важливо для балансування швидкості та точності обчислень. Крім того, EPOCHS = 30 означає, що модель пройде весь навчальний набір даних 30 разів, що дозволяє їй поступово покращувати якість класифікації. А AUTOTUNE = tf.data.AUTOTUNE - забезпечує автоматичну оптимізацію під час обробки даних, дозволяючи TensorFlow самостійно вибирати ефективну кількість паралельних потоків для прискорення завантаження та підготовки зображень. Крім того, набір даних поділяється на навчальну та тестову частини. Даний фрагмент є важливою підготовчою частиною, яка забезпечує доступ до даних у зручному форматі та задає основні параметри для подальшого навчання моделі [38].

Лістинг 4.1 – Код мережі

```
BATCH_SIZE = 128
EPOCHS = 30
AUTOTUNE = tf.data.AUTOTUNE
(ds_train, ds_test), ds_info = tfds.load(
    'emnist/letters',
    split=['train', 'test'],
    shuffle_files=True,
```

```
    as_supervised=True,
    with_info=True
)
```

У лістингу 4.2 відбувається підготовка даних для подачі в нейронну мережу, а також візуалізація прикладів з набору даних. Спочатку змінна NUM_CLASSES отримує значення кількості класів у наборі даних EMNIST Letters, яке зчитується з метаінформації об'єкта ds_info. У даному випадку це число дорівнює 26, що відповідає кількості літер англійського алфавіту (від «a» до «z»).

Функція preprocess() виконує базову обробку вхідних зображень та міток. Зображення типу uint8 (цілі числа від 0 до 255) конвертуються у формат float32 і нормалізуються до діапазону [0, 1], що полегшує навчання нейронної мережі. Далі зображення розгортається вздовж останньої осі, щоб надати йому форму (28, 28, 1) - тобто зробити його чорно-білим «канальним» зображенням, як того вимагають згорточні шари. Крім того, мітку класу зменшено на одиницю (label - 1), оскільки в EMNIST літери кодуються від 1 до 26, а TensorFlow очікує, що класи починаються з нуля [39].

Функція visualize_samples() використовується для відображення прикладів з набору даних. Вона отримує підмножину даних, список назв класів та кількість прикладів для відображення. Функція створює горизонтальний ряд зображень, де кожне з них позначено відповідною літерою. Це дозволяє візуально перевірити, що дані були завантажені та оброблені правильно, і допомагає зрозуміти, як виглядають вхідні зображення перед тим, як завантажити їх у модель.

Лістинг 4.2 – Код мережі

```
NUM_CLASSES = ds_info.features['label'].num_classes
def preprocess(image, label):
    image = tf.cast(image, tf.float32) / 255.0
    image = tf.expand_dims(image, -1)
```

```
    label = label - 1  # Зсув класів на 0-based
    return image, label
def visualize_samples(dataset, class_names, num_samples=10):
    plt.figure(figsize=(12, 2))
    for i, (image, label)
in enumerate(dataset.take(num_samples)):
        plt.subplot(1, num_samples, i + 1)
        plt.imshow(tf.squeeze(image), cmap='gray')
        plt.title(class_names[label.numpy()])
        plt.axis('off')
    plt.show()
```

У лістингу 4.3 реалізовано два важливих кроки: візуалізація зображень з набору даних та підготовка навчальних даних. Перш за все, визначено функцію visualize_samples() для відображення вибраних прикладів з навчального набору. Вона приймає в якості аргументів сам набір даних, список назв класів та кількість зображень для відображення (10 за замовчуванням). За допомогою бібліотеки matplotlib.pyplot створюється нове зображення певного розміру, після чого в циклі for перебираються перші num_samples прикладів з набору даних. Кожне зображення відображається у сірій палітрі кольорів (cmap=«gray»), а його мітка перетворюється на літеру алфавіту за допомогою списку class_names. Функція також видаляє осі для зручності перегляду і викликає plt.show() для відображення результату. Така візуалізація дозволяє оцінити якість і зміст даних перед навчанням моделі.

Після визначення функції, наступний рядок виконує попередню обробку навчального набору даних ds_train за допомогою функції preprocess(), описаної раніше. Вона нормалізує зображення, розширює його до (28, 28, 1) і переводить мітки класів у формат з нульовою індексацією. Результат зберігається у новій змінній ds_train_proc, яка стане основою для подальшої підготовки до навчання.

Останній рядок створює список назв класів, що містять символи англійського алфавіту від «a» до «z». Це реалізовано за допомогою генератора списків, який перебирає коди літер Unicode від ord(«a») до ord(«z») включно і

перетворює кожне число назад у відповідний символ. Список використовується для підписів на візуалізаціях, а також може знадобитися в інших частинах коду, наприклад, для відображення результатів класифікації. Разом блок коду дозволяє попередньо переглянути дані, переконатися в правильності обробки зображень і підготувати набір даних до завантаження в модель [40].

Лістинг 4.3 – Код мережі

```
def visualize_samples(dataset, class_names, num_samples=10):
    plt.figure(figsize=(12, 2))
    for i, (image, label)
in enumerate(dataset.take(num_samples)):
        plt.subplot(1, num_samples, i + 1)
        plt.imshow(tf.squeeze(image), cmap='gray')
        plt.title(class_names[label.numpy()])
        plt.axis('off')
    plt.show()
ds_train_proc = ds_train.map(preprocess)
class_names = [chr(i) for i in range(ord('a'), ord('z') + 1)]
```

У лістингу 4.4 реалізовано етап аугментації (штучного розширення) навчального набору даних, підготовку навчальних і тестових даних та ініціалізацію моделі. Спочатку створюється об'єкт data_augmentation, який є послідовною моделлю трьох аугментаційних шарів, що випадковим чином змінює вхідні зображення. Зокрема, RandomRotation(0.1) випадково повертає зображення на 10% від повного повороту, RandomZoom(0.1) збільшує або зменшує його, а RandomTranslation(0.1, 0.1) зсуває зображення по горизонталі та вертикалі на 10% від його розміру. Такі стохастичні зміни зберігають розпізнавання символів, але підвищують стійкість моделі до варіацій написання літер від руки, імітуючи різні стилі письма.

Далі визначається функція augment(), яка застосовує описану вище модель доповнення до кожного зображення і повертає модифіковане зображення разом з оригінальною міткою. Потім створюється новий

навчальний набір даних ds_train, в якому аугментація застосовується до раніше оброблених прикладів. Для прискорення виконання використовується багатопотокова обробка (num_parallel_calls=AUTOTUNE). Отриманий набори даних перемішується (shuffle(10000)), що допомагає уникнути локального перенавчання на схожих прикладах. Після цього дані об'єднуються в пакети по 128 зображень (batch(BATCH_SIZE)) і попередньо завантажуються в пам'ять (prefetch(AUTOTUNE)), що збільшує швидкість навчання.

Тестовий набір даних ds_test не доповнюється - він лише нормалізується за допомогою функції preprocess(), після чого також пакується та оптимізується для читання. Це важливо для об'єктивної оцінки моделі, яка повинна демонструвати результати на незмінених, «чистих» даних. Наприкінці фрагменту ми переходимо до побудови самої нейронної мережі через models.Sequential, яка визначає шари мережі [41]. Інформація про шари моделі наведена у таблиці 3.1.

Лістинг 4.4 – Код мережі

```
data_augmentation = tf.keras.Sequential([
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
    layers.RandomTranslation(0.1, 0.1),
])
def augment(image, label):
    image = data_augmentation(image)
    return image, label
ds_train =
ds_train_proc.map(augment, num_parallel_calls=AUTOTUNE)
ds_train                                                    =
ds_train.shuffle(10000).batch(BATCH_SIZE).prefetch(AUTOTUNE)
ds_test =
ds_test.map(preprocess, num_parallel_calls=AUTOTUNE)
ds_test = ds_test.batch(BATCH_SIZE).prefetch(AUTOTUNE)
model = models.Sequential
```

Таблиця 3.1 – Шари мережі

№	Назва	Тип	Вихідна форма	Кількість параметрів
1	**2**	**3**	**4**	**5**
1	Input	Вхід	(28, 28, 1)	0
2	Conv2D	Згортка (64, 3×3)	(28, 28, 64)	640
3	BatchNormalization	Нормалізація	(28, 28, 64)	256
4	Activation (ReLU)	Активація	(28, 28, 64)	0
5	Conv2D	Згортка (64, 3×3)	(28, 28, 64)	36,928
6	BatchNormalization	Нормалізація	(28, 28, 64)	256
7	Activation (ReLU)	Активація	(28, 28, 64)	0
8	MaxPooling2D	Пулінг (2×2)	(14, 14, 64)	0
9	Dropout (0.25)	Випадкове вимкнення	(14, 14, 64)	0
10	Conv2D	Згортка (128, 3×3)	(14, 14, 128)	73,856
11	BatchNormalization	Нормалізація	(14, 14, 128)	512

Продовження таблиці 3.1

1	2	3	4	5
12	Activation (ReLU)	Активація	(14, 14, 128)	0
13	Conv2D	Згортка (128, 3×3)	(14, 14, 128)	147,584
14	BatchNormalization	Нормалізація	(14, 14, 128)	512
15	Activation (ReLU)	Активація	(14, 14, 128)	0
16	MaxPooling2D	Пулінг (2×2)	(7, 7, 128)	0
17	Dropout (0.25)	Випадкове вимкнення	(7, 7, 128)	0
18	Flatten	Вирівнювання	(6272)	0
19	Dense	Повнозв'язний (256)	(256)	1,605,888
20	BatchNormalization	Нормалізація	(256)	1,024
21	Activation (ReLU)	Активація	(256)	0
22	Dropout (0.5)	Випадкове вимкнення	(256)	0
23	Dense (26)	Класифікація (Softmax)	(26)	26

Код у лістингу 4.5 завершує підготовку моделі до навчання. Спочатку нейронна мережа компілюється за допомогою методу model.compile(). На цьому етапі вказується, що в якості оптимізатора використовується алгоритм Адама - один з найпопулярніших та найефективніших варіантів глибокого

навчання, який поєднує в собі переваги методів Momentum та RMSProp. Функцію втрат визначено як розріджену категоріальну крос-ентропію, що є стандартною для задач багатокласової класифікації, де мітки класів подано у вигляді цілих чисел. Метрикою оцінки обрано точність, яка дозволяє відстежувати частку правильно класифікованих прикладів.

Після компіляції викликається метод model.summary(), який виводить структуровану інформацію про архітектуру моделі: кількість шарів, типи операцій, розміри тензорів кожному етапі та загальну кількість параметрів, які потрібно навчити. Це корисно для перевірки правильності побудови структури мережі перед початком навчання.

Далі створюється набір лабораторій. Перший колбек ModelCheckpoint відповідає за збереження найкращої версії моделі у файл best_emnist_cnn_colab.h5, орієнтуючись на максимальне значення точності на валідаційному наборі (val_accuracy). Друга лабораторія - TensorBoard - створює лог-файли в спеціальному каталозі logs/emnist/, в якому будуть записані всі дані про процес навчання (втрати, точність, графіки моделі тощо), які згодом можна буде переглянути через веб-інтерфейс TensorBoard. Нарешті, ReduceLROnPlateau автоматично зменшує швидкість навчання вдвічі (factor=0.5), якщо втрати при валідації не покращуються протягом трьох епох (patience=3). Це дозволяє уникнути ситуацій, коли модель застряє на одному рівні і більше не навчається. Разом ці елементи підвищують ефективність, гнучкість і керованість процесу навчання.

Лістинг 4.5 – Код мережі

```
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy'])
model.summary()
checkpoint_cb = callbacks.ModelCheckpoint(
    'best_emnist_cnn_colab.h5',
```

```
save_best_only=True, monitor='val_accuracy')
log_dir = «logs/emnist/»
+ datetime.datetime.now().strftime(«%Y%m%d-%H%M%S»)
tensorboard_cb = callbacks.TensorBoard(log_dir
=log_dir, histogram_freq=1)
lr_scheduler_cb = callbacks.ReduceLROnPlateau(
    monitor='val_loss', factor=0.5, patience=3, verbose=1)
```

Код у лістингу 4.6 безпосередньо навчає нейронну мережу та оцінює її на тестовому наборі даних. Навчання запускається за допомогою методу model.fit(), куди передається попередньо підготовлений навчальний набір даних ds_train. Модель навчається на кількох епох, вказаній у змінній EPOCHS (в даному випадку 30), що означає повний прохід по всіх навчальних прикладах 30 разів.

Крім основних даних, передаються також validation_data=ds_test - набір тестових прикладів, які використовуються для проміжної оцінки моделі після кожної епохи. Це дозволяє відстежувати не тільки успішність навчання, але і якість узагальнення, тобто здатність моделі працювати з новими, невідомими даними.

Особливу роль відіграють зворотні виклики, які передаються до списку callbacks. Як було описано раніше, checkpoint_cb зберігає найкращу з точки зору точності версію моделі на валідаційному наборі, tensorboard_cb записує лог-файли для візуалізації процесу навчання, а lr_scheduler_cb відповідає за динамічну зміну швидкості навчання в разі зупинки поліпшення [42].

Після завершення навчання модель тестується на повному тестовому наборі даних ds_test за допомогою методу model.evaluate(), який обчислює втрати (test_loss) та точність (test_acc). Отримане значення точності виводиться на консоль у чотиризначному форматі з десятковою комою, що дозволяє оцінити, наскільки добре модель розпізнає рукописні літери на незалежних даних. Цей блок завершує процес створення, тренування та перевірки нейронної мережі.

Лістинг 4.6 – Код мережі

```
history = model.fit(
    ds_train,
    epochs=EPOCHS,
    validation_data=ds_test,
    callbacks=[checkpoint_cb,
tensorboard_cb, lr_scheduler_cb])
test_loss, test_acc = model.evaluate(ds_test)
print(f'Test accuracy: {test_acc:.4f}')
```

Після 30 епох, на навчання яких було витрачено 3 години, була отримана мережа з точністю 0,9278. Ці мережа здатна розпізнавати рукописні символи. На рис. 3.2 показано результат класифікації випадкових символів з набору.

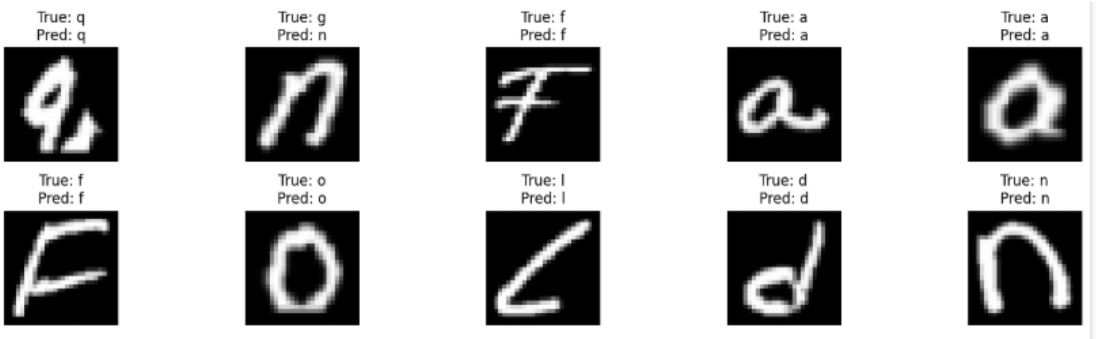

Рисунок 3.2 – Результат класифікації символів нашою мережею

Графік точності, представлений на рис. 3.3, показує стабільне зростання точності моделі протягом перших 10-15 епох, після чого темп зростання сповільнюється і наближається до плато. Кінцева точність, досягнута на навчальній вибірці, становить приблизно 95,5%, тоді як на валідаційній вибірці вона зупинилася на позначці 93%. Це нам говорить про певний ступінь узагальненості мережі, а також про відносно невелике перенавчання. Різниця в точності між навчанням і перевіркою є помірною і очікуваною для задач класифікації з великою кількістю класів, особливо у випадку рукописних символів, де висока варіабельність зображення ускладнює завдання.

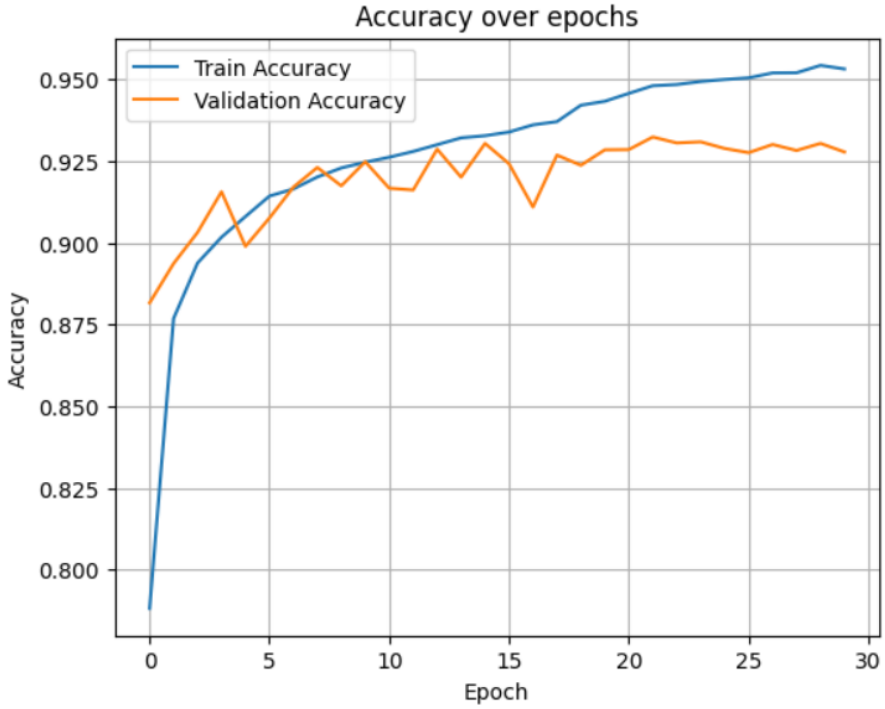

Рисунок 3.3 – Графік точності моделі на різних періодах навчання

Графік втрат, показаний на рис. 3.4, відображає стрімке падіння функції втрат у перші кілька епох як для навчальної, так і для валідаційної вибірок. Після цього крива втрат навчальної вибірки продовжує плавно спадати, тоді як крива валідації демонструє стабільний рівень з незначними коливаннями. Це може свідчити про той факт, що модель вже вичерпала потенціал простого узагальнення і подальше підвищення точності потребує складніших архітектурних рішень, збільшення обсягу даних або використання методів регуляризації. Водночас коефіцієнт втрат не перевищує 0,2 після 10 епох, що вважається гарним результатом для класифікації рукописних символів.

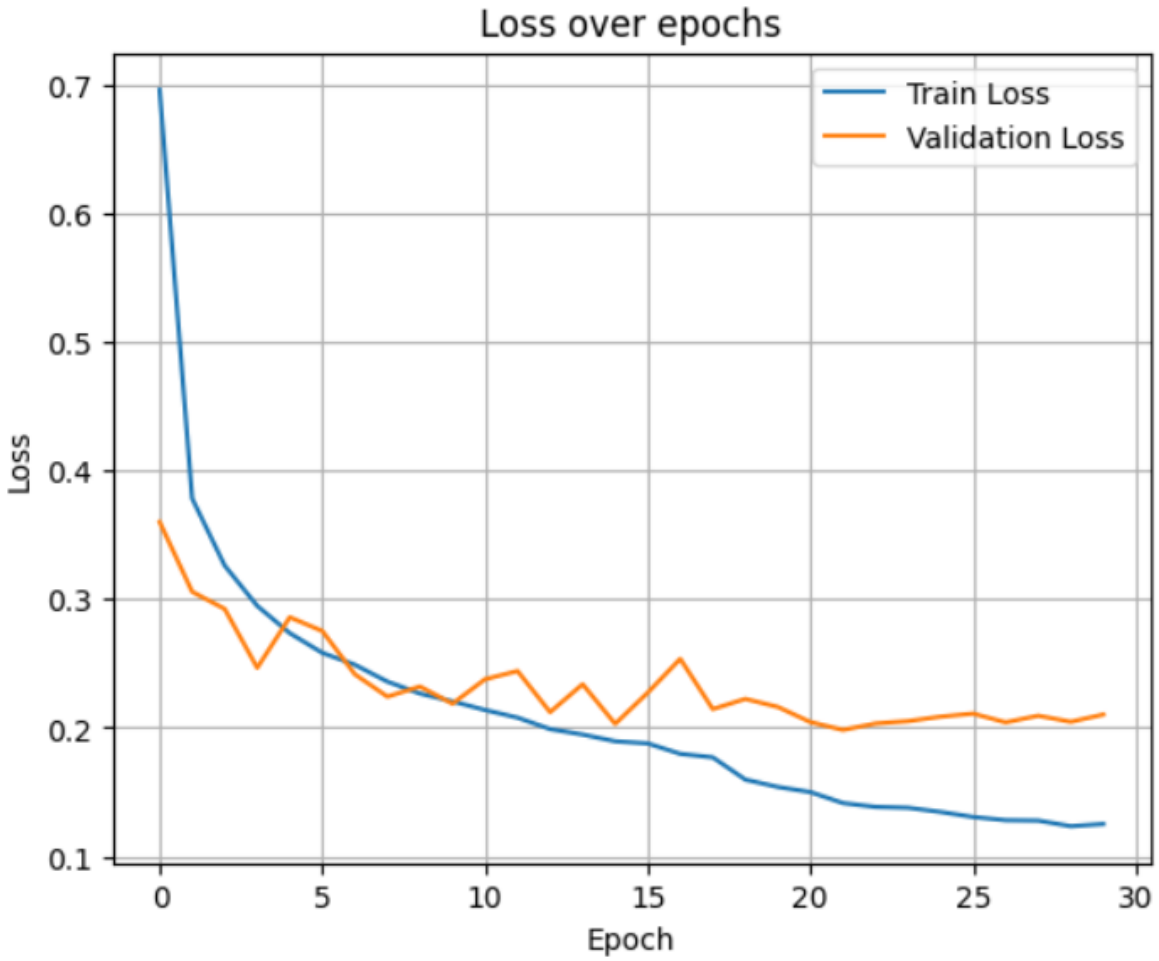

Рисунок 3.4 – Графік втрат моделі на різних періодах навчання

Ілюстрація передбачень моделі на тестових прикладах дозволяє оцінити якість класифікації з візуальної точки зору. Видно, що в більшості випадків модель правильно розпізнає символи, навіть якщо вони мають неоднозначну або спотворену форму. Наприклад, літери «a», «f», «o», «l», «d» і «q» були класифіковані правильно, хоча деякі з них мають змінну геометрію. Однак були й помилки, наприклад, помилкова класифікація «g» як «n». Ця помилка є типовою для задач розпізнавання почерку і зазвичай спричинена схожістю форми окремих літер у певних почерках.

Ця модель навчалася на візуальних зображення символів, тому її ефективність значною мірою залежить від якості сегментації символів перед подачею в мережу. У реальних умовах, коли на вході можуть бути цілі слова або речення, які необхідно розбити на окремі символи, загальна точність

системи може бути нижчою, якщо не передбачити додаткових етапів обробки. У той же час, результати демонструють здатність моделі успішно розпізнавати більшість класів, що відкриває можливість подальшого використання в реальних додатках, таких як цифровий рукописний ввід, автоматизоване оцифрування документів або освітні системи.

В результаті модель показала високу точність на навчальних і валідаційних даних з незначним рівнем перенавчання. Графіки втрат і приклади класифікації демонструють ефективність архітектури для задачі розпізнавання символів, з потенціалом для подальшої оптимізації шляхом покращення структури, використання глибших мереж, регуляризації або додавання мовного контексту. Ці результати можна вважати успішним етапом розробки моделі в рамках даної роботи.

3.3 Висновки до третього розділу

В третьому розділі кваліфікаційної роботи:

– Спроєктовано і розроблено нейронну мережу для розпізнавання рукописного тексту. При побудові були враховані особливості використаного датасету, що дозволило знизити показники втрат, а також збільшити точність розпізнавання;

– Проаналізовано результати створеної мережі. Дана мережа показала високі показники точності розпізнавання на тестовому наборі, що робить її непоганим підгрунтям для застосування в складніших мережах, які вже зможуть розпізнавати повноцінні рядки тексту.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

4.1 Загальні вимоги безпеки до обладнання та технологічних процесів

Усі технологічні процеси, що здійснюються в умовах експлуатації обладнання, мають відповідати чинним вимогам з охорони праці, технічного регламенту та нормам промислової безпеки. Це дозволяє забезпечити належний рівень захисту працівників, уникати аварійних ситуацій, мінімізувати шкідливий вплив на здоров’я людей і навколишнє середовище, а також забезпечити стабільність технологічного процесу. Вимоги безпеки охоплюють як конструктивні особливості обладнання, так і умови його експлуатації, обслуговування, ремонту й модернізації.

Усі одиниці обладнання, що використовуються на виробництві, повинні мати технічну документацію, сертифікати відповідності, пройти первинну та періодичну перевірки, бути укомплектованими необхідними засобами захисту — як колективного, так і індивідуального. Зокрема, до колективних засобів належать захисні екрани, кожухи, системи вентиляції, сигналізації й методи аварійного відключення. Індивідуальні засоби захисту — це спецодяг, респіратори, захисні окуляри, навушники, рукавички тощо. Перед допуском до роботи з обладнанням персонал повинен пройти відповідне навчання та перевірку знань з питань охорони праці, а також мати допуск до виконання робіт підвищеної небезпеки, якщо такі передбачені.

Конструкція обладнання повинна виключати можливість травмування, ураження електричним струмом, опіків, механічного затискання чи контакту з рухомими частинами. Рухомі елементи машин і механізмів повинні бути закриті або обладнані засобами блокування, які унеможливлюють роботу без встановлення захисних кожухів. Робочі поверхні повинні бути антиковзкими, а розміщення органів управління — ергономічним, зрозумілим і доступним з робочого місця оператора. Крім того, важливо дотримуватись принципу

енергетичної безпеки: усі джерела енергії мають бути обладнані системами аварійного відключення та захисту від перенавантаження [43].

До технологічних процесів висуваються вимоги щодо їхньої керованості, стабільності, передбачуваності результату та безпечності для персоналу. Усі операції повинні бути чітко регламентовані інструкціями, що враховують безпечну послідовність дій, допустимі режими роботи обладнання, обмеження щодо матеріалів та умов експлуатації. Особливу увагу слід приділяти операціям, пов’язаним із підвищеним ризиком: роботі з електричними установками, гарячими поверхнями, хімічно активними речовинами, при підйомі вантажів, при обслуговуванні під тиском тощо. У таких випадках необхідно проводити додатковий інструктаж, організовувати захисні зони, використовувати попереджувальні знаки та огородження.

Організація робочого простору повинна забезпечувати вільний доступ до обладнання для проведення ремонтних і профілактичних робіт, мати достатнє освітлення, вентиляцію та температуру, що відповідає санітарним нормам. Забороняється захаращення проходів, розміщення горючих матеріалів поблизу джерел тепла, недотримання нормативних відстаней між елементами обладнання. Усі приміщення мають бути оснащені засобами пожежогасіння, аварійним освітленням та вказівниками евакуаційних виходів.

Окрему увагу слід приділити безпечній організації роботи з комп’ютерною технікою, що є невід’ємною складовою більшості сучасних технологічних процесів. Комп’ютерне обладнання, включно з системними блоками, моніторами, серверами, джерелами безперебійного живлення, мережевими пристроями та периферією, повинне відповідати стандартам електробезпеки, мати заземлення та справні блоки живлення. Особливу небезпеку становлять перегрів компонентів, коротке замикання, перенапруга або несправність вентиляційних систем — усе це повинно бути враховано під час монтажу та експлуатації техніки.

Слід дотримуватись ергономічних вимог до робочого місця користувача ПК: зокрема, забезпечити правильну висоту монітора, кут огляду, відстань до очей (не менше 50–70 см), зручне розташування клавіатури й миші, а також належне освітлення — переважно природне розсіяне або з використанням джерел світла зі спектром, наближеним до природного. Регламентовано й тривалість безперервної роботи за комп'ютером — передбачено перерви для зменшення навантаження на зір та опорно-руховий апарат [44].

У разі використання комп'ютерів для автоматизованого управління технологічними процесами, необхідно передбачити засоби резервного живлення, захисту від збоїв у мережі, а також можливість ручного втручання в процеси в разі відмови програмного забезпечення. Усі системи повинні регулярно оновлюватися та проходити тестування на надійність, особливо у випадках, коли від їхньої роботи залежить безпека інших компонентів виробництва.

Також потрібно впроваджувати політику кібербезпеки — забезпечити захист локальних мереж, даних користувачів, технологічних алгоритмів від зовнішнього втручання або вірусного зараження. Для цього застосовують антивірусне ПЗ, фаєрволи, контролюють доступ до адміністрування систем, створюють резервні копії критично важливих даних.

Дотримання загальних вимог безпеки під час експлуатації обладнання й ведення технологічних процесів є запорукою не лише збереження життя та здоров'я працівників, а й безперебійної та якісної роботи виробництва. Важливо пам'ятати, що безпечне середовище — це результат зусиль усіх учасників процесу: від інженера-конструктора до оператора, від відповідального за охорону праці до керівника підприємства.

4.2 Протипожедні заходи в офісних приміщеннях

Протипожежна безпека в офісних приміщеннях є однією з ключових складових загальної системи охорони праці та безпеки життєдіяльності. Попри відносно низький рівень пожежонебезпеки у порівнянні з промисловими або виробничими об'єктами, офісні приміщення містять значну кількість електрообладнання, легкозаймистих матеріалів, документів та меблів, що при певному поєднанні чинників може спричинити стрімке поширення пожежі. Тому впровадження ефективних протипожежних заходів є обов’язковим у будь-якому офісному середовищі, незалежно від розміру установи чи кількості персоналу.

Основними джерелами пожеж у офісах, згідно з аналізом ДСНС України [45], є порушення правил експлуатації електроприладів, короткі замикання в електромережі, залишення ввімкнених електронагрівальних пристроїв без нагляду, необережне поводження з вогнем, а також несправність систем електропостачання. До додаткових чинників ризику належать перевантаження електромереж, використання несправних подовжувачів або несертифікованого обладнання, а також куріння в заборонених місцях.

Першочерговим завданням адміністрації будь-якого офісу є впровадження чіткої системи організаційних і технічних заходів з протипожежної безпеки. Зокрема, усі працівники повинні пройти вступний та первинний інструктаж з пожежної безпеки. Для окремих категорій осіб, відповідальних за утримання приміщень, проведення ремонтів або використання електрообладнання, необхідне також проходження повторного інструктажу та навчання дій у разі виникнення пожежі. У приміщеннях має бути розроблено план евакуації людей на випадок пожежі з позначенням шляхів руху, місць розміщення засобів пожежогасіння, аварійних виходів та виклику екстрених служб.

Важливою технічною вимогою є наявність справних первинних засобів пожежогасіння. В кожному офісному блоці повинні бути встановлені

вогнегасники відповідного типу — найчастіше це вуглекислотні (для електрообладнання) або порошкові універсального застосування. Вогнегасники повинні бути розміщені на видимих місцях, позначені відповідними знаками та не заблоковані меблями або іншими предметами. Перевірка стану вогнегасників повинна здійснюватися не рідше одного разу на рік із відповідним маркуванням дати перевірки. Крім того, доцільним є встановлення автоматичних систем виявлення задимлення або підвищення температури, які можуть сповістити персонал про небезпеку навіть у нічний або неробочий час.

Окрему увагу слід приділяти технічному стану електроустановок та побутової техніки. Усі прилади мають бути сертифіковані, мати справні вилки та ізоляцію, бути підключеними через захищені розетки з заземленням. Забороняється підключення кількох потужних пристроїв в одну розетку, використання саморобних подовжувачів або блоків живлення. Перед завершенням робочого дня обов’язковим є відключення комп’ютерів, принтерів, чайників, кавоварок та іншої техніки, за винятком обладнання, що має працювати цілодобово. Для таких пристроїв необхідно передбачити джерела безперебійного живлення та регулярний технічний огляд.

Організація офісного простору також повинна відповідати протипожежним вимогам. Шляхи евакуації мають бути вільними, достатньо широкими, без порогів та перешкод. Двері аварійних виходів повинні відкриватися у напрямку евакуації та не блокуватися. На кожному поверсі будівлі має бути передбачене аварійне освітлення та план-схема евакуації, розміщена на видимих місцях. Освітлення евакуаційних виходів та аварійні покажчики мають бути автономними — тобто здатними функціонувати навіть у разі знеструмлення будівлі.

У разі оренди приміщення особливу увагу слід звертати на дотримання загальних протипожежних вимог усіма орендарями: забороняється самовільне перепланування, прокладання додаткових кабелів, зберігання легкозаймистих речовин або матеріалів у службових приміщеннях. Усі меблі, обшивки та оздоблювальні матеріали мають відповідати класам вогнестійкості відповідно

до чинних будівельних норм. У разі проведення ремонтних або зварювальних робіт необхідне оформлення наряду-допуску та обов’язковий пожежний нагляд [46].

Ключовим елементом протипожежної безпеки є підготовка персоналу до дій у надзвичайній ситуації. Усі працівники повинні чітко знати свої дії при виявленні задимлення або вогню. Бажано проводити періодичні навчання з евакуації та протипожежні тренування, які дозволяють закріпити навички й виявити можливі слабкі місця в організації безпеки.

Таким чином, дотримання протипожежних заходів в офісних приміщеннях вимагає системного підходу, що поєднує нормативне регулювання, технічне забезпечення, контроль стану засобів безпеки та підготовку персоналу. Лише комплексне виконання цих заходів дозволяє мінімізувати ризик виникнення пожежі, зменшити ймовірні збитки та, найголовніше, зберегти людські життя.

4.3 Висновки до четвертого розділу

В четвертому розділі кваліфікаційної роботи:

– Розглянуто загальні вимоги безпеки до обладнання та технологічних процесів, що спрямовані на зменшення виробничих ризиків;

– Проаналізовано протипожежні заходи в офісних приміщеннях, які забезпечують належну готовність до надзвичайних ситуацій та захист працівників.

ВИСНОВКИ

В результаті виконання бакалаврської кваліфікаційної роботи було розроблено згорткову мережу для класифікації рукописних літер.

В першому розділі кваліфікаційної роботи:

– Розглянуто предметну область розпізнавання тексту;

– Проаналізовано основні алгоритми, які застосовуються в даній області;

– Описано функціональність розробленої мережі;

В другому розділі кваліфікаційної роботи

– Описано область розпізнавання рукописного тексту;

– Досліджено етапи розробки згорткової нейронної мережі;

В третьому розділі кваліфікаційної роботи:

– Спроєктовано і розроблено нейронну мережу для розпізнавання рукописного тексту;

– Проаналізовано результати створеної мережі. Дана мережа показала високі показники точності розпізнавання на тестовому наборі, що робить її непоганим підґрунтям для застосування в складніших мережах, які вже зможуть розпізнавати повноцінні рядки тексту.

В четвертому розділі кваліфікаційної роботи:

– Розглянуто загальні вимоги безпеки до обладнання та технологічних процесів, що спрямовані на зменшення виробничих ризиків;

– Проаналізовано протипожежні заходи в офісних приміщеннях, які забезпечують належну готовність до надзвичайних ситуацій та захист працівників.

ПЕРЕЛІК ДЖЕРЕЛ

1 Optical Character Recognition (OCR): Emerging Trends and Future Applications. Jumio. URL: https://www.jumio.com/optical-character-recognition-trends-and-applications/.

2 What is OCR (Optical Character Recognition)? Benefits, Challenges, and Use Cases [Infographic]. Medium. URL: https://weareshaip.medium.com/what-is-ocr-optical-character-recognition-benefits-challenges-and-use-cases-infographic-225dbb067549.

3 Digitized Historical and Cultural Heritage Consolidation Technologies: From a Territorial Resource to a National Portal / Halyna Lypak та ін. 10th International Conference on Advanced Computer Information Technologies. Conference Proceedings, Deggendorf, Germany, 16 верес. 2020 / Ternopil National Economaical University.

4 Дубчак А. О. Напрямки використання штучного інтелекту в сучасних умовах / А. О. Дубчак, Я. В. Литвиненко // Матеріали міжнародної наукової конференції «ван Пулюй: життя в ім’я науки та України» (до 175-ліття від дня народження), 28-30 вересня 2020 року. — Т. : ФОП Паляниця В. А., 2020. — С. 64–65.

5 What are convolutional neural networks?. *IBM*. URL: https://www.ibm.com/think/topics/convolutional-neural-networks.

6 What is LSTM – Long Short Term Memory? [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/deep-learningintroduction-to-long-short-term-memory/.

7 Text Recognition With CRNN-CTC Network. Wandb. URL: https://wandb.ai/authors/text-recognition-crnn-ctc/reports/Text-Recognition-With-CRNN-CTC-Network--VmlldzoxNTI5NDI.

8 TrOCR. Huggingface. URL: https://huggingface.co/docs/transformers/model_doc/trocr.

9 Building a Handwriting Recognition System with CRNN: A Beginner’s Guide. *Medium*. URL: https://medium.com/@pavitharan2020/building-a-handwriting-recognition-system-with-crnn-a-beginners-guide-58a51a46dd15.

10 TrOCR — Transformer-based Optical Recognition Model. Medium. URL: https://medium.com/@tejpal.abhyuday/trocr-transformer-based-optical-recognition-model-811f7b3217da.

11 Emnist. *Tensorflow*. URL: https://www.tensorflow.org/datasets/catalog/emnist.

12 Перенавчання. Google for Developers. URL: https://developers.google.com/machine-learning/crash-course/overfitting/overfitting?hl=uk.

13 Natural Language Processing (NLP) – Overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/natural-languageprocessing-overview/.

14 Exploring Natural Language Processing (NLP) in Translation [Електронний ресурс] – Режим доступу до ресурсу: https://www.shaip.com/blog/nlp-in-translation/.

15 Fundamentals of Statistical Natural Language Processing [Електронний ресурс] – Режим доступу до ресурсу: https://www.proxet.com/blog/fundamentalsof-statistical-natural-language-processing

16 Handwriting Text Recognition (HTR). PlanetAI. URL: https://planet-ai.com/handwriting-text-recognition-htr/.

17 Бабак В. П., Марченко М. Є., Фриз. Б. Г. Теорія ймовірностей, випадкові процеси та математична статистика. К.: Техніка, 2004. 288 с.

18 Roscoe Goble Loveth. Transforming OCR Accuracy: The Impact of Deep Learning vs. Bayesian Inference in Handwritten Character Recognition. ResearchGate. 2024.

19 OCR Deep Learning: The Curious Machine Learning Case. *Labelyourdata*. URL: https://labelyourdata.com/articles/ocr-with-deep-learning.

20 The EMNIST Dataset. *NIST*. URL: https://www.nist.gov/itl/products-and-services/emnist-dataset.

21 What is Adam Optimizer?. GeeksforGeek. URL: https://www.geeksforgeeks.org/deep-learning/adam-optimizer/.

22 Липак Г., Липак Т., Кунанець Н. Проєктування інформаційної системи на основі машинного навчання для збереження та класифікації артефактів документальної спадщини. *Вісник Хмельницького національного університету: технічні науки.* Т. 334 № 4 (2024). С. 176-182. https://doi.org/10.31891/2307-5732-2024-339-4-29

23 Fryz M., Scherbak L., Mlynko B., Mykhailovych T. Linear Random Process Model-Based EEG Classification Using Machine Learning Techniques. Proceedings of the 1st International Workshop on Computer Information Technologies in Industry 4.0 (CITI 2023). Ternopil, Ukraine: CEUR Workshop Proceedings, 2023. Vol. 3468. P. 126–132.

24 Multi-Layer Neural Network [Електронний ресурс] – Режим доступу до ресурсу: http://deeplearning.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/.

25 What is gradient descent? [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/think/topics/gradient-descent.

26 Що таке обробка природної мови (NLP) та як вона може використовуватися у бізнесі [Електронний ресурс] – Режим доступу до ресурсу: https://metinvest.digital/ua/page/1052.

27 Як ми вчимося. Чому мозок навчається краще, ніж машина… Поки що / пер. з англ. Юлія Костюк. – 3-тє вид. – К. :Лабораторія, 2023. – 288 с

28 Luis Serrano. Grokking Machine Learning, 2023. – 512 с.

29 Функції активації: Ступінчаста, лінійна, сигмоїда, ReLU та Tanh. *Robotdreams*. URL: https://robotdreams.cc/uk/blog/327-funkciji-aktivaciji-stupinchasta-liniyna-sigmojida-relu-ta-tanh.

30 What are Convolution Layers?. GeeksforGeek. URL: https://www.geeksforgeeks.org/machine-learning/what-are-convolution-layers/.

31 A Brief History of Natural Language Processing [Електронний ресурс] – Режим доступу до ресурсу: https://www.dataversity.net/a-brief-history-of-naturallanguage-processing-nlp/.

32 TensorFlow і машинне навчання. Foxminded. URL: https://foxminded.ua/tensorflow-shcho-tse/.

33 Keras 3 API documentation. Keras. URL: https://keras.io/api/.

34 TensorFlow Datasets: a collection of ready-to-use datasets.. Tensorflow. URL: https://www.tensorflow.org/datasets.

35 Matplotlib: Visualization with Python. Matplotlib. URL: https://matplotlib.org/.

36 Jupyter for Data Science [Електронний ресурс] – Режим доступу до ресурсу: https://docs.jupyter.org/en/latest/use/use-cases/data_science.html.

37 Why Deep Learning Need GPU. GeeksforGeek. URL: https://www.geeksforgeeks.org/deep-learning/why-deep-learning-need-gpu/.

38 Ерік Маттес. Пришвидшений курс Python / Ерік Маттес. – Львів: Видавництво Старого Лева, 2021. – 600 с

39 Neural networks and statistical techniques: A review of applications [Електронний ресурс] – Режим доступу до ресурсу: https://www.sciencedirect.com/science/article/abs/pii/S0957417407004952?__cf_chl_tk=nhhMjiamBzsEVB5EwXrkCHoUKIFsLCmLjsR6OU_WZD0-1734599769-1.0.1.1-OvKUGdsIHDUcx3Qujo9P.nL0wOl5J9yuylTXZkZ1RQQ.

40 Python for Machine Learning [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/python-for-machine-learning/.

41 Кліщ, М., Липак, Г., Кунанець, Н., Пасічник, С., & Липак, Т. Структура інформаційної системи передбачення та інтерпретації зміни стану користувача сервісу. Вісник Національного Університету «Львівська Політехніка». Інформаційні системи та мережі, 17 (2025). С. 226 - 238. https://doi.org/10.23939/sisn2025.17.226

42 Python for Mathematicians [Електронний ресурс] – Режим доступу до ресурсу: https://www.math.purdue.edu/~bradfor3/ProgrammingFundamentals/Python/.

43 Безпечність машин. Загальні принципи проектування. Оцінювання ризиків та зменшення ризиків: ДСТУ EN ISO 12100:2016. – Київ : Мінекономрозвитку України, 2016.

44 Державні санітарні правила і норми. Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПіН 3.3.2.007-98 – Київ : Міністерство охорони здоров’я України, 1998.

45 Аналітична довідка про пожежі та їх наслідки в Україні за 11 місяців 2024 року. *ДСНС*. URL: https://dsns.gov.ua/upload/2/2/8/5/6/0/7/analitychna-dovidka-pro-pojeji-112024.pdf.

46 Будівлі та споруди. Пожежна безпека. Загальні вимоги : ДБН В.1.1-7:2016. – [Чинний від 01.06.2017]. – Київ : Мінрегіон України, 2016.

ДОДАТКИ

Додаток А

Код нейронної мережі для розпізнавання рукописних символів

```
!pip install -q tensorflow-datasets

import tensorflow as tf
from tensorflow.keras import layers, models, callbacks
import tensorflow_datasets as tfds
import datetime
import matplotlib.pyplot as plt
import IPython.display as display

print(«GPU is», «available»
if tf.config.list_physical_devices('GPU')
else «NOT AVAILABLE»)

BATCH_SIZE = 128
EPOCHS = 30
AUTOTUNE = tf.data.AUTOTUNE

(ds_train, ds_test), ds_info = tfds.load(
    'emnist/letters',
    split=['train', 'test'],
    shuffle_files=True,
    as_supervised=True,
    with_info=True
)

NUM_CLASSES = ds_info.features['label'].num_classes

def preprocess(image, label):
    image = tf.cast(image, tf.float32) / 255.0
    image = tf.expand_dims(image, -1)
    label = label - 1  # Зсув класів на 0-based
    return image, label

def visualize_samples(dataset, class_names, num_samples=10):
    plt.figure(figsize=(12, 2))
    for i, (image, label) in enumerate(dataset.take
(num_samples)):
        plt.subplot(1, num_samples, i + 1)
        plt.imshow(tf.squeeze(image), cmap='gray')
        plt.title(class_names[label.numpy()])
        plt.axis('off')
    plt.show()

ds_train_proc = ds_train.map(preprocess)

class_names = [chr(i) for i in range(ord('a'), ord('z') + 1)]

visualize_samples(ds_train_proc, class_names)

data_augmentation = tf.keras.Sequential([
```

```
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
    layers.RandomTranslation(0.1, 0.1),
])

def augment(image, label):
    image = data_augmentation(image)
    return image, label

ds_train = ds_train_proc.map(augment,
num_parallel_calls=AUTOTUNE)ds_train
= ds_train.shuffle(10000).batch(BATCH_SIZE).prefetch
(AUTOTUNE)

ds_test = ds_test.map(preprocess, num_parallel_calls=
AUTOTUNE)
ds_test = ds_test.batch(BATCH_SIZE).prefetch(AUTOTUNE)

model = models.Sequential([
    layers.Input(shape=(28,28,1)),

    layers.Conv2D(64, 3, padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.Conv2D(64, 3, padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.25),

    layers.Conv2D(128, 3, padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.Conv2D(128, 3, padding='same'),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.25),

    layers.Flatten(),
    layers.Dense(256),
    layers.BatchNormalization(),
    layers.Activation('relu'),
    layers.Dropout(0.5),

    layers.Dense(NUM_CLASSES, activation='softmax')
])

model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()
```

```
    checkpoint_cb = callbacks.ModelCheckpoint(
        'best_emnist_cnn_colab.h5',
    save_best_only=True, monitor='val_accuracy'
    )

    log_dir = «logs/emnist/» +
    datetime.datetime.now().strftime(«%Y%m%d-%H%M%S»)
    tensorboard_cb = callbacks.TensorBoard
    (log_dir=log_dir, histogram_freq=1)

    lr_scheduler_cb = callbacks.ReduceLROnPlateau(
        monitor='val_loss', factor=0.5, patience=3, verbose=1
    )

    history = model.fit(
        ds_train,
        epochs=EPOCHS,
        validation_data=ds_test,
        callbacks=[checkpoint_cb, tensorboard_cb, lr_scheduler_cb]
    )

    test_loss, test_acc = model.evaluate(ds_test)
print(f'Test accuracy: {test_acc:.4f}')

def plot_training_history(history):
    fig, axs = plt.subplots(1, 2, figsize=(14,5))

    axs[0].plot(history.history['accuracy'],
label='Train Accuracy')
    axs[0].plot(history.history['val_accuracy'],
label='Validation Accuracy')
    axs[0].set_title('Accuracy over epochs')
    axs[0].set_xlabel('Epoch')
    axs[0].set_ylabel('Accuracy')
    axs[0].legend()
    axs[0].grid(True)

    axs[1].plot(history.history['loss'], label='Train Loss')
    axs[1].plot(history.history['val_loss'],
label='Validation Loss')
    axs[1].set_title('Loss over epochs')
    axs[1].set_xlabel('Epoch')
    axs[1].set_ylabel('Loss')
    axs[1].legend()
    axs[1].grid(True)

    plt.show()

def fix_emnist_image(img):
    rotated = np.rot90(img, k=-1)
    flipped = np.fliplr(rotated)
    return flipped
```

```
plot_training_history(history)


import random
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import load_model

model = load_model('best_emnist_cnn_colab.h5')

images_list = []
labels_list = []

for images, labels in ds_test.unbatch():
    images_list.append(images.numpy())
    labels_list.append(labels.numpy())

images_array = np.array(images_list)
labels_array = np.array(labels_list)

N = 10
indices = random.sample(range(len(images_array)), N)
sample_images = images_array[indices]
sample_labels = labels_array[indices]

preds = model.predict(sample_images)
pred_labels = tf.argmax(preds, axis=1).numpy()

plt.figure(figsize=(15, 4))
for i in range(N):
    plt.subplot(2, 5, i+1)
    fixed_img = fix_emnist_image(sample_images[i].squeeze())
    plt.imshow(fixed_img, cmap='gray')
    plt.title(f»True: {class_names[sample_labels
[i]]}\nPred: {class_names[pred_labels[i]]}»)
    plt.axis('off')
plt.tight_layout()
plt.show()
```