ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП‘ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «**НЕЙРОМЕРЕЖЕВІ МОДЕЛІ ДЛЯ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ**»

на здобуття освітнього ступеня бакалавра
зі спеціальності 123 – Комп’ютерна інженерія
(код, найменування спеціальності)

освітньо-професійної програми Комп’ютерна інженерія
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ *Дмитро ХОМЕНКО*
(підпис) *Ім’я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти групи КІД-41
Дмитро ХОМЕНКО
Ім’я, ПРІЗВИЩЕ
Керівник Юлія БОРОВА
Ім’я, ПРІЗВИЩЕ
Рецензент _____
Ім’я, ПРІЗВИЩЕ

Київ – 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально–науковий інститут інформаційних технологій

Кафедра Комп'ютерної інженерії
Ступінь вищої освіти – Бакалавр
Спеціальність – 123 – Комп'ютерна інженерія
Освітньо-професійна програма Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедрою
Комп'ютерної інженерії
Наталія ЛАЩЕВСЬКА
"__"_____2025 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Хоменко Дмитру Володимировичу

1.Тема кваліфікаційної роботи: «Нейромережеві моделі для розпізнавання рукописного тексту»
Керівник кваліфікаційної роботи: Борова Ю.О. Асистент,
затверджені наказом вищого навчального закладу від "24"лютого 2025 року №56.
2.Строк подання кваліфікаційної роботи 05.06.2025
3.Вхідні дані до кваліфікаційної роботи:
Науково-технічна література про архітектури штучних нейронних мереж
Статті про сучасні підходи до розпізнавання рукописного тексту (OCR).
Опис відкритих датасетів для навчання моделей розпізнавання (IAM, EMNIST).
Матеріали з реалізації нейронних мереж у середовищах TensorFlow та PyTorch.
Наукові публікації про застосування CRNN, CTC, та Transformer архітектур

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).
4.1. Постановка задачі
4.2. Дослідження штучних нейронних мереж
4.3. Аналіз застосування штучних нейронних мереж
4.4. Аналіз отриманої інформації
5. Перелік графічного матеріалу
5.1. Мета, об'єкт, предмет
5.2. Актуальність
5.3. Модель персептрона

5.4. Багатошарова штучна нейронна мережа
5.5. Властивості штучних нейронних мереж
5.6. Завдання штучних нейронних мереж
5.7. Переваги та недоліки штучних нейронних мереж
5.8. Практичне застосування штучних нейронних мереж у медицині
5.9. Висновки
5.10. Перелік публікацій
6. Дата видачі завдання «24» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір джерел інформації	08.04.25	
2	Постановка задачі	11.04.25	
3	Дослідження штучних нейронних мереж	25.04.25	
4	Аналіз застосування штучних нейронних мереж	06.05.25	
5	Написання роботи	08.05.25	
6	Висновки	09.05.25	
7	Підготовка демонстраційного матеріалу	10.05.25	
8	Попередній захист роботи	28.05.25	
9	Пред'явлення роботи в деканат	05.06.25	

Здобувач вищої освіти _____ Дмитро ХОМЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Юлія БОРОВА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 54 с., 8 табл., 12 рис., 1 графік, 31 джерел.

Об'єкт дослідження – нейромережеві моделі.

Предмет дослідження – методи розпізнавання рукописного тексту за допомогою штучних нейронних мереж.

Мета роботи – дослідити, порівняти та реалізувати сучасні нейромережеві підходи до розпізнавання рукописного тексту, а також оцінити їх ефективність на практиці.

Короткий зміст роботи: Метою кваліфікаційної роботи є розробка нейромережевої системи для автоматичного розпізнавання рукописного тексту. У роботі проаналізовано історію розвитку OCR-систем, виділено основні труднощі, пов'язані з рукописами, а також охарактеризовано архітектури нейронних мереж, що використовуються в таких задачах (CNN, RNN, CRNN, Transformer).

Здійснено порівняння точності та ефективності моделей на відкритих датасетах (IAM, EMNIST). Обґрунтовано вибір оптимальної архітектури та реалізовано систему на основі глибокої нейронної мережі з використанням сучасних фреймворків (TensorFlow / PyTorch).

На основі експериментальних результатів зроблено висновок, що нейромережеві моделі мають високу точність розпізнавання та значно перевершують класичні методи у випадку нестандартного або складного почерку.

Упровадження таких систем може бути корисним у багатьох сферах: архівній справі, освіті, медицині, мобільних додатках тощо. Робота підтверджує важливість і перспективність подальших досліджень у цьому напрямі.

КЛЮЧОВІ СЛОВА: ГЛИБОКЕ НАВЧАННЯ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, РОЗПІЗНАВАННЯ ТЕКСТУ, OCR, ШТУЧНИЙ ІНТЕЛЕКТ.

ЗМІСТ

ВСТУП.....9
1 ТЕОРЕТИЧНІ ОСНОВИ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ.....11
1.1 Історія та сучасний стан розпізнавання рукописного тексту.....11
1.1.1 Витоки технологій OCR.....11
1.1.2 Етапи розвитку технологій.....11
1.1.3 Сучасні рішення та їх використання.....12
1.2 Основні проблеми та особливості рукописного тексту.....12
1.2.1 Індивідуальність почерку.....12
1.2.2 Перекриття символів, шум, варіативність.....12
1.2.3 Мультимовність.....13
1.3 Традиційні методи оптичного розпізнавання символів (OCR).....13
1.3.1 Алгоритми виділення контурів.....13
1.3.2 Метод шаблонів.....13
1.3.3 Метод евристик.....14
1.3.4 Порівняння з сучасними підходами.....14
1.4 Переваги використання нейронних мереж у розпізнаванні.....15
1.4.1 Адаптивність.....15
1.4.2 Автоматичне виділення ознак.....15
1.4.3 Перевага над класичними алгоритмами.....15
1.5 Підходи до навчання моделей розпізнавання.....15
1.5.1 Навчання з учителем.....15
1.5.2 Безвчительське навчання.....16
1.5.3 Transfer Learning.....16
1.5.4 Fine-tuning.....16
1.6 Огляд відкритих датасетів для рукописного текста.....16
1.6.1 IAM Handwriting Database.....16
1.6.2 EMNIST.....16
1.6.3 RIMES, CVL, Bentham.....17
1.6.4 Порівняльна характеристика датасетів.....17
1.7 Метрики якості розпізнавання тексту.....17
1.7.1 Accuracy.....17
1.7.2 Character Error Rate (CER).....17
1.7.3 Word Error Rate (WER).....17
1.7.4 Комбіноване використання метрик.....18
2 АНАЛІЗ ТА ВИБІР НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ ДЛЯ РОЗПІЗНАВАННЯ.....19
2.1 Огляд існуючих архітектур нейронних мереж для OCR.....19
2.1.1 Згорткові нейронні мережі (CNN).....19
2.1.2 Рекурентні нейронні мережі (RNN).....22
2.1.3 Згортково-рекурентні нейронні мережі (CRNN).....25

2.1.4 Трансформер та модифікації....28
2.2 Порівняльна характеристика моделей....33
2.2.1 Точність, швидкість, ресурси....33
2.2.2 Глибина моделі....37
2.2.3 Переваги і недоліки....39
2.3 Аналіз відкритих датасетів для розпізнавання....41
2.3.1 MNIST, EMNIST....41
2.3.2 IAM, інші набори....43
2.3.3 Придатність до задач....45
2.4 Вибір моделі та обґрунтування....46
2.4.1 Вибір датасету....46
2.4.2 Вибір архітектури....48
2.4.3 Очікування та обґрунтування....50
3 РОЗРОБКА НЕЙРОМЕРЕЖОВОЇ СИСТЕМИ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ....53
3.1 Архітектура програмного забезпечення....53
3.1.1 Схема рішення....53
3.1.2 Використані бібліотеки....55
3.1.3 Інтерфейс користувача (за наявності)....56
3.2 Передобробка даних....57
3.2.1 Нормалізація зображень....57
3.2.2 Бінаризація, сірий формат....58
3.2.3 Сегментація тексту....58
3.2.4 Видалення шуму....58
3.3 Навчання моделі та вибір гіперпараметрів....59
3.3.1 Функція втрат....59
3.3.2 Оптимізатори....60
3.3.3 Кількість епох, батчі, LR....60
3.3.4 Використання обчислювальних ресурсів....61
3.4 Тестування та валідація результатів....62
3.4.1 Метрики точності....62
3.4.2 Графіки втрат....63
3.4.3 Візуалізація передбачень....63
ВИСНОВКИ....65
ПЕРЕЛІК ПОСИЛАНЬ....67
ДОДАТОК А....71
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (ПРЕЗЕНТАЦІЯ)....78

ВСТУП

Сьогодні дедалі більше інформації переходить у цифровий формат. Але багато важливих даних все ще зберігаються у вигляді рукописних документів — наприклад, архівні матеріали, особисті записи, шкільні зошити або медичні картки. Щоб швидко й зручно з ними працювати, їх треба розпізнавати та переводити в текст. Проблема в тому, що почерки у всіх різні, і комп'ютеру важко зрозуміти, що саме написано.

Класичні OCR-системи (оптичне розпізнавання символів) непогано справляються з друкованим текстом, але часто дають помилки при розпізнаванні рукописів. Тому зараз активно застосовують нейромережеві моделі, які можуть навчатися розпізнавати складні зображення, зокрема й рукописний текст. Ця тема є актуальною, бо такі технології вже зараз використовуються в реальних додатках, а з кожним роком стають дедалі потрібнішими.

Метою роботи є дослідити сучасні нейромережеві моделі для розпізнавання рукописного тексту та реалізувати одну з них, щоб перевірити її ефективність.
Для досягнення цієї мети треба виконати такі завдання:

- вивчити існуючі методи OCR та особливості рукописного тексту;
- ознайомитися з різними архітектурами нейромереж (CNN, RNN, Transformer тощо);
- вибрати відповідний датасет і модель для розпізнавання;
- навчити модель на обраному наборі даних;
- провести тестування та оцінити результат;
- порівняти отриману модель з іншими OCR-системами;
- зробити висновки та запропонувати ідеї для покращення.

Об'єкт дослідження — нейромережеві моделі для розпізнавання тексту. Предмет дослідження — саме ті методи, які дозволяють розпізнавати рукописний текст за допомогою таких моделей. У роботі використано теоретичний аналіз наукових

джерел, огляд і порівняння різних моделей, практична реалізація нейромережі, а також експериментальне тестування з подальшим аналізом результатів.

Результати можуть бути корисними в різних сферах: від розпізнавання шкільних або медичних записів до застосування в мобільних додатках. Якщо правильно налаштувати таку модель, вона може автоматично "читати" навіть складні почерки, що суттєво спрощує обробку даних.

1 ТЕОРЕТИЧНІ ОСНОВИ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ

Для того, щоб зрозуміти, як працює розпізнавання рукописного тексту, варто спочатку розглянути історію створення та розвиток цього процесу, а також проблеми . Адже розпізнавання рукописного тексту доволі складний процес, що поєднує в собі обробку зображень та роботу штучного інтелекту. Найпоширеніша проблема в розпізнаванні тексту коли мова йде не про друковані, а саме про рукописні документи складність котрих залежить від стилю, мови, якості зображень та почерку.

1.1 Історія та сучасний стан розпізнавання рукописного тексту

1.1.1 Витоки технологій OCR

Перші розробки в автоматичному розпізнаванні символів були зроблені ще в 1950-х роках. Першими вважаються роботи компаній IBM та Reader's Digest, які розробляли OCR-системи на основі простого порівняння з шаблонами (template matching). Ці системи використовували точне піксельне співставлення й були здатні розпізнавати лише друковані символи з вузького набору шрифтів.

1.1.2 Етапи розвитку технологій

Починаючи з 1980-х років, з розвитком обчислювальної техніки з’явилися гнучкіші алгоритми, зокрема методи евристичного аналізу, геометричної сегментації та статистичного моделювання. У 1990-х роках починають активно використовуватись методи машинного навчання, наприклад SVM та k-NN. У

подальшому, з поширенням методів глибокого навчання, основними архітектурами стали згорткові нейронні мережі (CNN), рекурентні мережі (RNN) і трансформери.

1.1.3 Сучасні рішення та їх використання

Серед провідних сучасних систем: Google Cloud Vision, Microsoft Azure OCR, Amazon Textract, Tesseract v5 та Adobe Sensei. Більшість із них використовують глибокі нейромережі, особливо комбінації згорткових та рекурентних шарів, або архітектури типу encoder-decoder з attention. Крім того, з 2020-х років активно застосовуються Transformer-архітектури (наприклад, TrOCR від Microsoft), які поєднують CNN як екстрактор ознак і декодер на основі трансформерів. Такі моделі демонструють надзвичайну точність, зокрема у випадках попереднього навчання на великих наборах даних із застосуванням архітектури "encoder-decoder".

1.2 Основні проблеми та особливості рукописного тексту

Розпізнавання рукописного тексту залишається однією з найскладніших задач у галузі комп’ютерного зору. Це пов’язано з великою кількістю індивідуальних, стилістичних та структурних відомостей, які не характерні друкованому тексту.

1.2.1 Індивідуальність почерку

Кожна людина має унікальний стиль письма. Це ускладнює роботу систем, оскільки не існує єдиного універсального шаблону для розпізнавання. Почерк може змінюватися залежно від інструмента, положення руки, швидкості письма, а також психоемоційного стану.

1.2.2 Перекриття символів, шум, варіативність

Рукописний текст нерідко супроводжується артефактами: плямами, розмиттям, накладенням символів. Такі завади ускладнюють процес сегментації і викликають хибні позитивні або негативні розпізнавання. У таких випадках потрібна або глибока сегментація, або обхідний шлях розпізнавання без попередньої сегментації.

1.2.3 Мультимовність

Системи мають розпізнавати символи з різних мов, враховуючи різні алфавіти, варіанти почерку та морфологічні особливості. Наприклад, латиниця має інші правила з’єднання символів, ніж кирилиця чи арабський алфавіт. Тому для ефективного розпізнавання необхідне або багатомовне навчання, або fine-tuning під кожну мову.

1.3 Традиційні методи оптичного розпізнавання символів (OCR)

1.3.1 Алгоритми виділення контурів

Контурні методи застосовують обробку країв зображення, зокрема оператор Собеля, фільтр Кенні, морфологічну обробку. Як результат, система виконує поділ тексту на символи, що потім проходять етап розпізнавання.

1.3.2 Метод шаблонів

Template Matching використовує просте піксельне зіставлення. Попри обмежену гнучкість, він і досі використовується для розпізнавання фіксованих шрифтів (наприклад, OCR-A, OCR-B).

1.3.3 Метод евристик

Евристичний підхід базується на заздалегідь прописаних правилах: кількість замкнених областей, перетинів, вертикальна та горизонтальна симетрія. Його слабке місце — обмеженість до заздалегідь заданого набору символів.

1.3.4 Порівняння з сучасними підходами

На відміну від класичних методів, глибокі нейронні мережі здатні самостійно виділяти ознаки та адаптуватися до нових форм символів. Наприклад, у тестах на IAM датасеті сучасні CRNN-моделі показують точність до 94%, тоді як традиційні методи — не вище 70%.

Таблиця 1.1 – Порівняння класичні методи з сучасними нейромережевими для задач OCR

Параметр	**Класичні методи**	**Нейромережі (Deep Learning)**
Адаптивність	Низька	Висока
Необхідність ручного налаштування	Висока	Мінімальна
Якість для рукописів	Низька	Висока (особливо CRNN, Transformer)
Витрати на навчання	Мінімальні	Високі (потрібен обчислювальний ресурс)
Точність для друкованого тексту	Висока	Висока

1.4 Переваги використання нейронних мереж у розпізнаванні

1.4.1 Адаптивність

Нейромережі легко пристосовуються до нових стилів, почерків, алфавітів. Це можливо завдяки властивості узагальнення, притаманній глибоким моделям.

1.4.2 Автоматичне виділення ознак

Замість ручної розмітки або вибору характеристик, мережа сама навчається витягати важливі ознаки. Особливо ефективні CNN при роботі з образами символів.

1.4.3 Перевага над класичними алгоритмами

Нейронні мережі здатні обробляти нестандартні та нерівномірні документи, автоматично підлаштовуючись під вхідні дані без потреби в ручному налаштуванні.

1.5 Підходи до навчання моделей розпізнавання

1.5.1 Навчання з учителем

Для навчання нейромережевої моделі потрібен датасет із підписаними зображеннями. Це найбільш поширений підхід, що демонструє високу ефективність у задачах OCR.

1.5.2 Безвчительське навчання

Дозволяє розпочати навчання моделі без залучення вручну анотованих наборів даних. Використовуються автокодери або контрастивне навчання (SimCLR, MoCo).

1.5.3 Transfer Learning

Модель, навчена на великому наборі даних, наприклад, на англомовному IAM, може бути донавчена для української або арабської мови з використанням меншого набору.

1.5.4 Fine-tuning

Цей процес полягає в додатковому навчанні вже існуючої моделі під конкретну задачу — наприклад , документи медичного спрямування, які мають нетиповий формат оформлення.

1.6 Огляд відкритих датасетів для рукописного текста

1.6.1 IAM Handwriting Database

Містить понад 1500 сторінок рукописного тексту англійською мовою. Поширений у задачах розпізнавання повних речень.

1.6.2 EMNIST

Підходить для класифікації окремих символів (латинські літери, цифри). Перевагою є велика кількість прикладів на символ.

1.6.3 RIMES, CVL, Bentham

Французький (RIMES), німецький (CVL) та історичний англійський (Bentham) набори даних, що дозволяють тестувати мультимовні та історичні моделі.

1.6.4 Порівняльна характеристика датасетів

IAM — для послідовностей; EMNIST — символи; CVL — стилі почерку. Кожен має переваги та обмеження залежно від задачі.

1.7 Метрики якості розпізнавання тексту

1.7.1 Accuracy

Оцінка точності на рівні символів або слів. Просте співвідношення правильних передбачень до загальної кількості.

1.7.2 Character Error Rate (CER)

CER = (заміни + видалення + вставлення) / кількість символів у референсі. Часто використовується в рукописному OCR.

1.7.3 Word Error Rate (WER)

Застосовується у випадках, коли важливо оцінити логічну цілісність речення. Обчислюється аналогічно CER, але на рівні слів.

1.7.4 Комбіноване використання метрик

Для об’єктивної оцінки моделі бажано використовувати як CER, так і WER, а також інші показники, наприклад BLEU або ROUGE для довших фрагментів.

2 АНАЛІЗ ТА ВИБІР НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ ДЛЯ РОЗПІЗНАВАННЯ

У даному розділі буде проведено аналіз архітектури нейронних мереж, що використовуються для оптичного розпізнавання символів (OCR), з акцентом на рукописний текст. Окрему увагу буде приділено порівнянню моделей за основними технічними критеріями: точністю, швидкістю обробки, споживанням ресурсів і складністю реалізації.

2.1 Огляд існуючих архітектур нейронних мереж для OCR

Архітектури нейронних мереж, що використовуються для задач оптичного розпізнавання символів, зазвичай поділяють на кілька основних типів. Серед них вирізняють згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN), а також їх поєднання у вигляді згортково-рекурентних моделей (CRNN). Окремо варто зазначити сучасні трансформерні архітектури, які демонструють високі результати у задачах розпізнавання завдяки здатності обробляти послідовності без втрати контексту.

2.1.1 Згорткові нейронні мережі (CNN)

Згорткові нейронні мережі (Convolutional Neural Networks, CNN) стали фундаментальним інструментом в задачах комп'ютерного зору, включаючи OCR. Натхненням для їх архітектури були біологічні процеси зорової кори головного мозку, які ефективно виявляє ієрархічні ознаки в зображеннях.

Принцип роботи та архітектура. CNN зазвичай складаються з кількох типів шарів:

– *Згорткові шари (Convolutional Layers).* Застосовують набір фільтрів (ядер згортки) до вхідного зображення для створення карт ознак. Кожен фільтр навчається виявляти специфічні локальні патерни, такі як краї, кути, текстури тощо. Наприклад, перші шари можуть виявляти прості ознаки, а глибші шари – складніші структури, що є комбінаціями ознак попередніх шарів;

– *Шари підвибірки (Pooling Layers, або Subsampling Layers).* Зменшують розмірність карт ознак, зберігаючи найважливішу інформацію. Найпоширенішими типами є MaxPooling (вибір максимального значення в локальній області) та AveragePooling (обчислення середнього значення). Це допомагає зменшити обчислювальну складність та зробити модель більш стійкою до невеликих зсувів та деформацій об'єктів на зображенні;

– *Функції активації (Activation Functions).* Вводять нелінійність в модель, дозволяючи їй навчатися складним залежностям. Популярними функціями активації є ReLU (Rectified Linear Unit), Sigmoid, Tanh. ReLU ($f(x)=\max(0,x)$) є найпоширенішою завдяки своїй простоті та ефективності у боротьбі з проблемою згасання градієнтів;

– *Повнозв'язні шари (Fully Connected Layers).* Після проходження через декілька згорткових та підвибіркових шарів, отримані карти ознак часто "випрямляються" в одновимірний вектор і подаються на вхід повнозв'язним шарам, які виконують класифікацію на основі вивчених ознак.

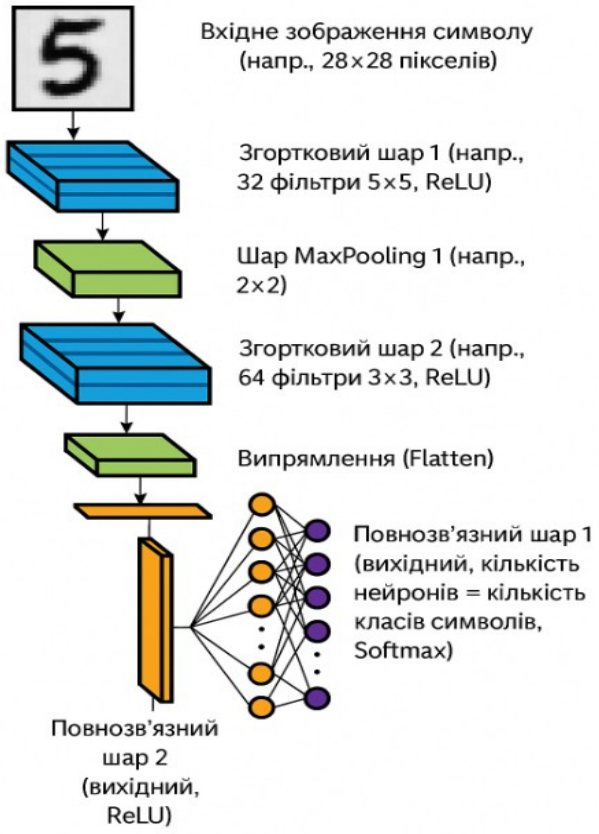


Рисунок 2.1 – Загальна архітектура CNN для класифікації зображень

Застосування в OCR:

В OCR CNN переважно використовуються для виділення візуальних ознак з зображень окремих символів або цілих слів/рядків тексту;

– *Розпізнавання окремих символів.*

CNN можуть бути навчені класифікувати зображення кожного символу на відповідний клас (літера, цифра, розділовий знак);

– *Виділення ознак для послідовних моделей.*

У більш складних системах OCR, CNN можуть слугувати енкодером ознак, вихід якого потім обробляється рекурентними мережами для розпізнавання послідовностей символів.

Переваги CNN в OCR:

– *Інваріантність до зсувів та локальних деформацій.* Завдяки операціям згортки та підвибірки;

– *Ієрархічне виділення ознак.* Від простих до складних, що дозволяє ефективно розпізнавати різноманітні шрифти та стилі написання;

– *Зменшення кількості параметрів.* Завдяки спільному використанню ваг фільтрів (weight sharing).

Недоліки CNN в OCR:

– *Чутливість до глобальних просторових трансформацій та обертань*. Якщо такі трансформації не були представлені в навчальних даних;

– *Обмеженість у моделюванні довгих послідовностей.* Стандартні CNN менш пристосовані для прямого розпізнавання цілих рядків тексту без додаткових механізмів.

2.1.2 Рекурентні нейронні мережі (RNN)

Рекурентні нейронні мережі (Recurrent Neural Networks, RNN) спеціально розроблені для обробки послідовних даних, де порядок елементів має значення. Це робить їх природним вибором для розпізнавання тексту в OCR, оскільки текст за своєю суттю є послідовністю символів.

Принцип роботи та архітектура:

На відміну від CNN, RNN мають "пам'ять" завдяки наявності зворотних зв'язків у своїй архітектурі. Вихід RNN у певний момент часу залежить не тільки від поточного вхідного сигналу, але й від попередніх обчислень (прихованого стану).

Стандартна архітектура RNN включає:

– *Вхідний шар (Input Layer).* Приймає елемент послідовності на кожному кроці часу. В OCR це можуть бути вектори ознак, що відповідають "зрізам" 1(slices) зображення слова або рядка, отримані, наприклад, з CNN;

– *Прихований шар (Hidden Layer).* Містить рекурентні нейрони. Прихований стан h_t на кроці t обчислюється на основі вхідного сигналу x_t та прихованого стану попереднього кроку h_t−1: h_t=f(W_hhh_t−1+W_xhx_t+b_h), де W – матриці ваг, b_h – вектор зсуву, f – функція активації;

– *Вихідний шар (Output Layer)*. Генерує вихід y_t на основі поточного прихованого стану: y_t=g(W_hyh_t+b_y), де g – функція активації (наприклад, Softmax для класифікації символів).

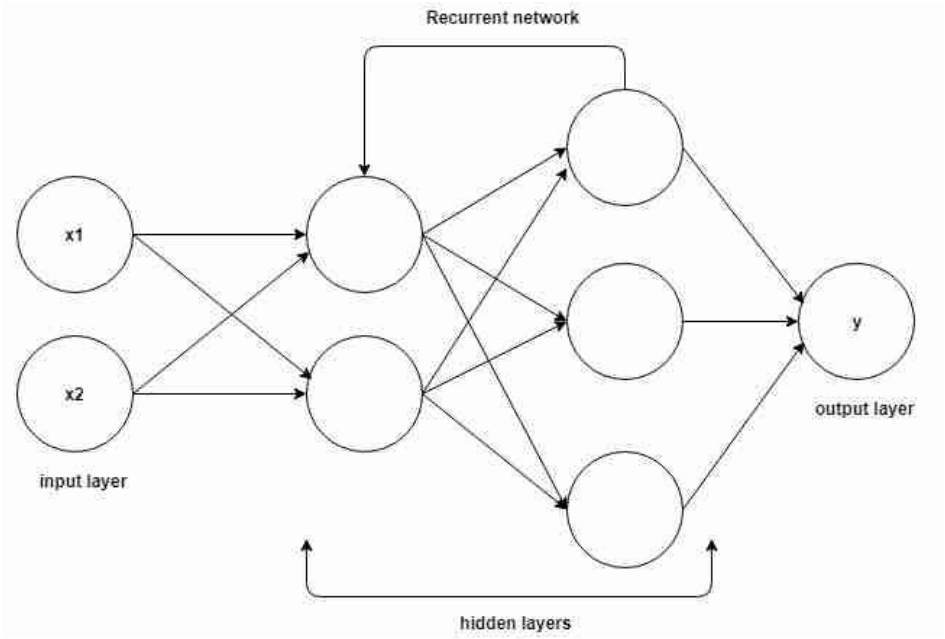


Рисунок 2.2 – Приклад рекурентної нейронної мережі

Проблеми стандартних RNN:

Стандартні RNN страждають від проблеми *згасання (vanishing gradient)* та *вибуху (exploding gradient)* градієнтів при навчанні на довгих послідовностях. Це ускладнює вивчення довгострокових залежностей.

Модифікації RNN - LSTM та GRU. Для подолання цих проблем були розроблені більш складні рекурентні комірки:

– *Довга короткочасна пам'ять (Long Short-Term Memory, LSTM).* LSTM-комірки мають спеціальну структуру з трьома "вентилями" (gate): вхідним (input gate), вентилем забування (forget gate) та вихідним (output gate). Ці вентилі контролюють потік інформації, дозволяючи мережі вибірково зберігати, оновлювати та забувати інформацію в своєму внутрішньому стані (cell state). Це робить LSTM значно ефективнішими для моделювання довгих залежностей;

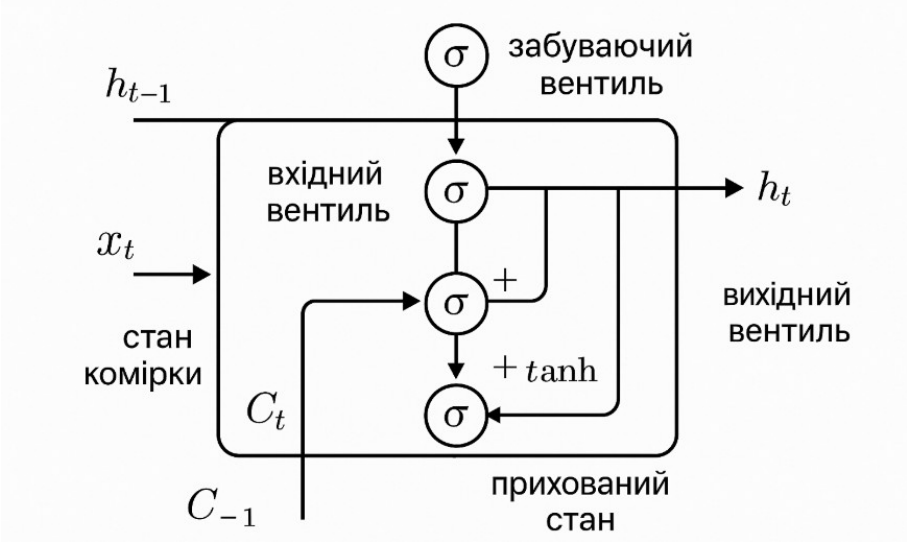


Рисунок 2.3 – Структура LSTM-комірки з вентилями

– *Керований рекурентний блок (Gated Recurrent Unit, GRU).* GRU є спрощеною версією LSTM з двома вентилями: оновлення (update gate) та скидання (reset gate). GRU часто демонструють схожу продуктивність з LSTM, але мають менше параметрів і можуть бути швидшими в навчанні.

Застосування в OCR. RNN (особливо LSTM та GRU) використовуються в OCR для:

– *Розпізнавання послідовностей символів.* Після того, як CNN виділила ознаки з зображення, RNN обробляє ці ознаки як послідовність і генерує ланцюг розпізнаних символів;

– *Моделювання мови.* RNN можуть використовуватися для врахування контекстуальної інформації та лінгвістичних закономірностей, покращуючи точність розпізнавання.

Переваги RNN в OCR:

– *Здатність моделювати послідовності.* Ефективно обробляють дані змінної довжини;

– *Врахування контексту.* Можуть використовувати інформацію з попередніх символів для розпізнавання поточного;

– *Ефективність LSTM/GRU.* Дозволяють вивчати довгострокові залежності в тексті.

Недоліки RNN в OCR:

– *Обчислювальна складність.* Особливо для довгих послідовностей, навчання може бути повільним;

– *Послідовна обробка.* Неможливість паралелізації обчислень по часових кроках (хоча існують спроби це обійти);

– *Складність у вивченні дуже довгих залежностей.* Навіть LSTM/GRU мають свої обмеження.

2.1.3 Згортково-рекурентні нейронні мережі (CRNN)

Згортково-рекурентні нейронні мережі (Convolutional Recurrent Neural Network, CRNN) є потужною гібридною архітектурою, що поєднує переваги CNN для виділення ознак та RNN для обробки послідовностей. CRNN стали фактичним стандартом для багатьох задач розпізнавання тексту.

Принцип роботи та архітектура. Типова архітектура CRNN для OCR складається з трьох основних компонентів:

– *Згорткові шари (Convolutional Layers).* На початку мережі знаходяться CNN (наприклад, VGG, ResNet або спеціалізовані архітектури), які приймають на вхід зображення тексту (наприклад, рядок або слово). Їхня задача – виділити з зображення послідовність векторів ознак (feature sequence). Кожен вектор у цій послідовності відповідає певній локальній області (фрейму) на вхідному зображенні. Наприклад, після проходження через CNN, зображення шириною W може бути перетворене на послідовність L векторів ознак, де $L\<W$;

– *Рекурентні шари (Recurrent Layers).* Виділені згортковими шарами ознаки подаються на вхід рекурентним шарам (зазвичай двонаправлені LSTM або GRU). Двонаправленість означає, що послідовність ознак обробляється у двох напрямках (зліва направо та справа наліво), що дозволяє RNN враховувати контекст з обох боків від поточного фрейму. Це особливо корисно для розпізнавання символів, значення якого може залежати від сусідніх;

– *Шар транскрипції (Transcription Layer / CTC Layer).* Вихід рекурентних шарів – це послідовність розподілів ймовірностей для будь-якого символу алфавіту на кожному часовому кроці (фреймі). Шар транскрипції перетворює цю послідовність ймовірностей у кінцевий текстовий рядок. Найчастіше для цього використовується функція втрат *Connectionist Temporal Classification (CTC).* CTC дозволяє навчати модель без необхідності точного вирівнювання (alignment) між вхідними фреймами та вихідними символами. Це дуже важливо, оскільки сегментація тексту на окремі символи є складною задачею сама по собі. CTC обробляє виходи RNN, допускаючи "порожні" мітки (blank labels) та повторення символів, і знаходить найімовірнішу послідовність символів.

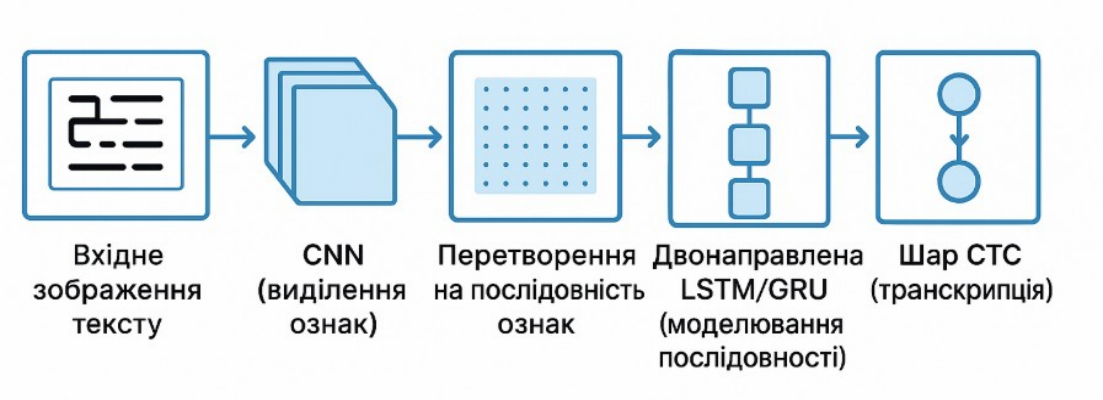


Рисунок 2.4 – Загальна архітектура CRNN для OCR

Переваги CRNN в OCR:

– *End-to-End навчання:* Модель навчається розпізнавати текст безпосередньо з зображення без проміжних етапів ручної сегментації;

– *Обробка тексту довільної довжини:* Завдяки рекурентним шарам;

– *Висока точність:* Поєднання сильних сторін CNN та RNN;

– *Незалежність від попередньої сегментації:* CTC-loss дозволяє уникати складного етапу сегментації символів.

Недоліки CRNN в OCR:

– *Складність архітектури та навчання.* Поєднання двох типів мереж;

– *Чутливість до сильних викривлень тексту.* Хоча CNN можуть впоратися з невеликими варіаціями, сильні викривлення (наприклад, дугоподібний текст) можуть потребувати додаткових механізмів, таких як модулі просторової трансформації (Spatial Transformer Networks, STN);

– *Обмежене моделювання двовимірних залежностей.* Хоча CNN захоплюють 2D інформацію, RNN обробляє її як 1D послідовність.

Таблиця 2.1 – Порівняння CNN, RNN та CRNN для задач OCR

Характеристика	CNN	RNN (LSTM/GRU)	CRNN
Основне призначення	Виділення візуальних ознак	Моделювання послідовностей, контексту	End-to-end розпізнавання тексту
Тип вхідних даних	Зображення (фіксованого розміру)	Послідовності векторів ознак	Зображення тексту (змінної ширини)
Обробка послідовностей	Обмежена	Основна перевага	Ефективно через RNN компонент
Інваріантність	До локальних зсувів, деформацій	До варіацій довжини послідовності	Поєднання переваг CNN та RNN
End-to-End навчання	Для класифіка	Потребує попередньо	Так, з використан

	ції окремих символів	виділених ознак	ням CTC
Складність	Середня	Середня/ Висока	Висока
Типові проблеми	Чутливість до глобальних трансформацій	Згасання/ вибух градієнтів (частково вирішено LSTM/GRU)	Чутливість до сильних викривлень тексту, складність навчання

2.1.4 Трансформер та модифікації

Архітектура Трансформер (Transformer), вперше представлена у статті "Attention Is All You Need" (Vaswani et al., 2017) для задач машинного перекладу, швидко революціонізувала обробку природної мови (NLP) і згодом була успішно адаптована для задач комп'ютерного зору, включаючи OCR.

Принцип роботи та архітектура:

Ключовим нововведенням Трансформера є *механізм уваги (attention mechanism), зокрема самоувага (self-attention)*, який дозволяє моделі зважувати важливість різних частин вхідної послідовності при обробці кожної її частини. На відміну від RNN, які обробляють дані послідовно, Трансформери можуть обробляти всі елементи послідовності паралельно, що значно прискорює навчання на великих обсягах даних.

Основні компоненти Трансформера:

а) *Вбудовування (Embedding).* Вхідні дані (наприклад, токени слів або патчі зображення) перетворюються на вектори фіксованої розмірності. Додається позиційне кодування (positional encoding) для врахування порядку елементів, оскільки сама архітектура не має рекурентності;

б) *Кодувальник (Encoder).* Складається з декількох однакових шарів. Кожен шар має два підшари:

1) *Багатоголовкова самоувага (Multi-Head Self-Attention).* Дозволяє моделі одночасно фокусуватися на різних частинах послідовності та вивчати різні аспекти залежностей між елементами;

2) *Повнозв'язна мережа прямого поширення (Feed-Forward Neural Network).* Застосовується до кожного елемента послідовності окремо. Навколо кожного підшару використовуються залишкові з'єднання (residual connections) та нормалізація шарів (layer normalization);

в) *Декодувальник (Decoder).* Також складається з декількох однакових шарів. Окрім підшарів, аналогічних кодувальнику, декодувальник має додатковий шар уваги, який фокусується на виході кодувальника (encoder-decoder attention). У задачах OCR, де декодується послідовність символів, декодувальник часто є авторегресійним, тобто генерує один символ за кроком, використовуючи раніше згенеровані символи як контекст.

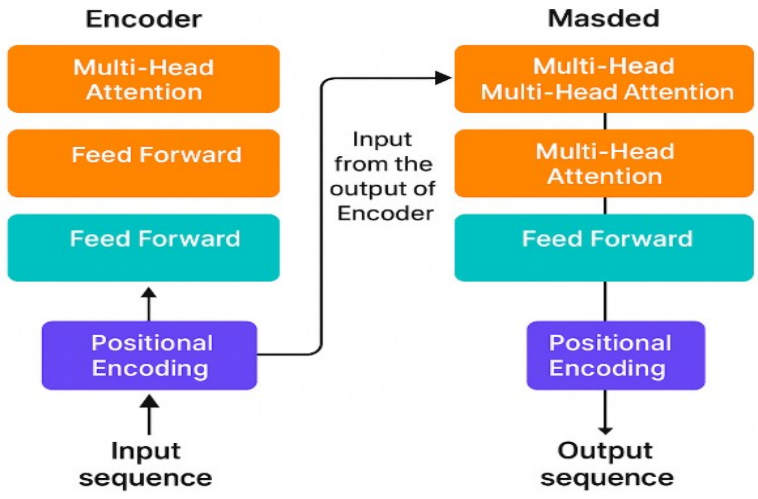


Рисунок 2.5 – Загальна архітектура Трансформера (кодувальник-декодувальник)

Застосування в OCR. Трансформери та їх модифікації застосовуються в OCR на різних етапах:

а) *Розпізнавання тексту (Text Recognition):*

1) *Використання Трансформер-декодераб.* CNN або ViT (Vision Transformer) можуть використовуватися як енкодер для виділення візуальних ознак, а Трансфор мер-декодер – для генерації послідовності символів;

2) *End-to-End Трансформери.* Деякі моделі намагаються використовувати Трансформер для всієї задачі, мінімізуючи роль традиційних CNN. Наприклад, *Vision Transformer (ViT)* розбиває зображення на послідовність патчів, які потім обробляються стандартним Трансформер-кодувальником;

б) *Виявлення тексту (Text Detection).* Трансформери (наприклад, DETR - DEtection TRansformer) також адаптуються для задач виявлення об'єктів, включаючи текстові області;

в) *Комбіновані моделі.* Моделі, що одночасно виявляють та розпізнають текст, можуть використовувати трансформерні архітектури для обох завдань або їх комбінації.

Модифікації Трансформерів для OCR:

– *Vision Transformer (ViT).* Адаптує архітектуру Трансформера безпосередньо до зображень. Зображення ділиться на фіксовані за розміром патчі, лінійно вбудовується, додається позиційне кодування, і отримана послідовність векторів подається на вхід Трансформер-кодувальнику;

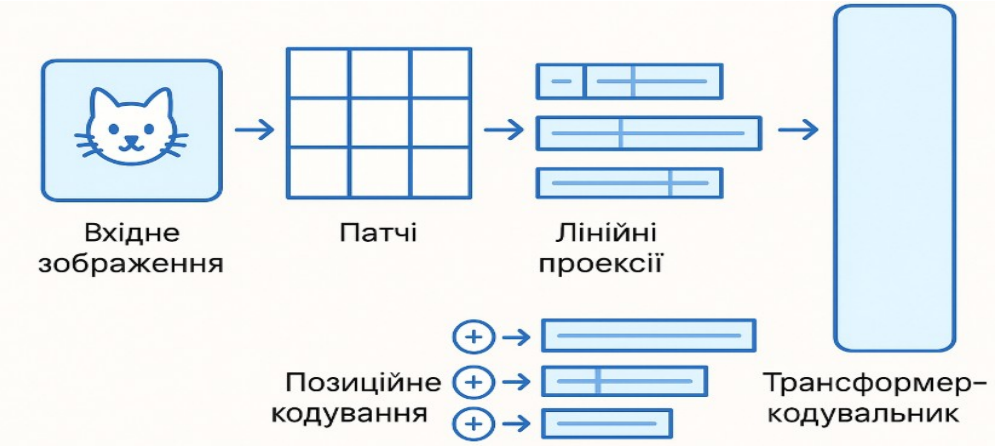


Рисунок 2.6 – Принцип роботи Vision Transformer (ViT)

– *Swin Transformer.* Ієрархічний Трансформер, який використовує зсунуті вікна (shifted windows) для обчислення уваги, що дозволяє моделювати залежності на різних масштабах і є більш ефективним для обробки зображень високої роздільної здатності;

– *Моделі з 2D увагою.* Деякі модифікації використовують двовимірні механізми уваги для кращого захоплення просторових залежностей у зображеннях тексту;

– *Гібридні моделі.* Поєднання CNN (для виділення низькорівневих ознак) та Трансформерів (для моделювання глобальних залежностей).

Переваги Трансформерів в OCR:

– *Потужне моделювання глобальних залежностей.* Завдяки механізму самоуваги;

– *Паралелізація обчислень.* На відміну від RNN, що прискорює навчання;

– *Висока продуктивність.* Часто демонструють state-of-the-art результати на багатьох бенчмарках OCR;

– *Гнучкість.* Можуть бути адаптовані для різних аспектів OCR (розпізнавання, виявлення).

Недоліки Трансформерів в OCR:

– *Високі вимоги до даних.* Для ефективного навчання Трансформери зазвичай потребують великих наборів даних;

– *Обчислювальна складність.* Механізм самоуваги має квадратичну складність відносно довжини послідовності, що може бути проблемою для дуже довгих послідовностей або зображень високої роздільної здатності (хоча існують методи для зменшення цієї складності, наприклад, у Swin Transformer);

– *Складність інтерпретації.* Розуміння того, на що саме "звертає увагу" модель, може бути нетривіальним;

– *Потреба в позиційному кодуванні.* Оскільки архітектура не враховує порядок елементів за замовчуванням.

Після того як ми ознайомились з основними архітектурами нейронних мереж для ORC можна зробити у підсумку таку таблицю

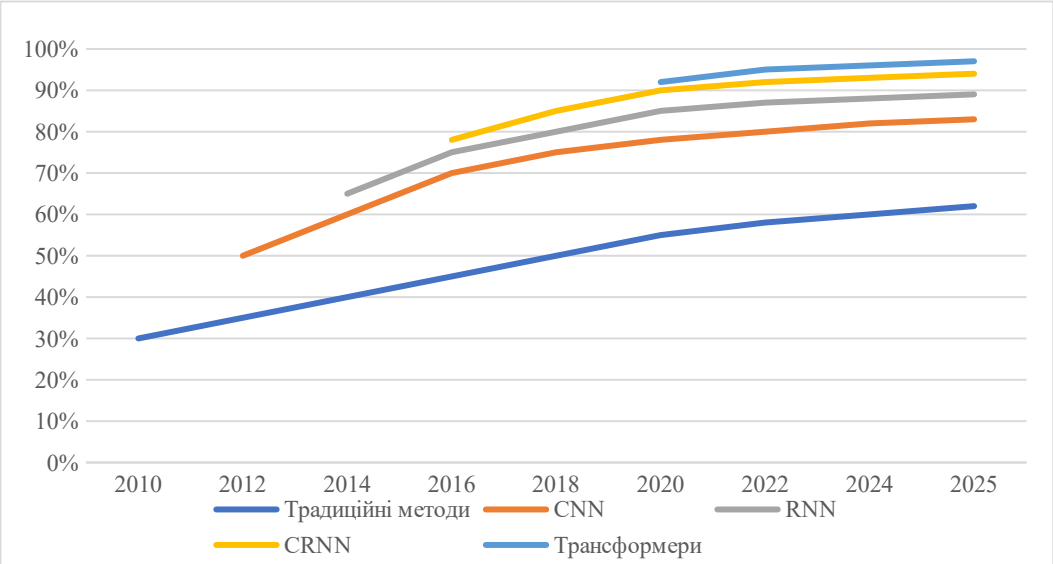
Таблиця 2.2 – Зведена порівняльна таблиця архітектур для OCR

Архітектура	**Основна ідея**	**Переваги**	**Недоліки**	**Типове застосування в OCR**
CNN	Ієрархічне виділення локальних візуальних ознак	Інваріантність до зсувів, ефективне виділення ознак, спільне використання ваг.	Обмежене моделювання послідовностей, чутливість до глобальних трансформацій.	Класифікація окремих символів, виділення ознак для RNN/Transformer.

Продовження Таблиці 2.2 – Зведена порівняльна таблиця архітектур для OCR

Архітектура	**Основна ідея**	**Переваги**	**Недоліки**	**Типове застосування в OCR**
RNN (LSTM/GRU)	Моделювання послідовностей з урахуванням контексту	Обробка даних змінної довжини, врахування контексту, моделювання довгострокових залежностей (LSTM/GRU).	Послідовна обробка (повільно), проблеми з дуже довгими залежностями, згасання/вибух градієнтів (частково).	Розпізнавання послідовностей символів після виділення ознак, моделювання мови.
CRNN	Комбінац	End-to-end	Складність	End-to-end

	ія CNN (ознаки) та RNN (послідовності) з CTC	навчання, обробка тексту довільної довжини, висока точність, незалежність від сегментації.	архітектури та навчання, чутливість до сильних викривлень.	розпізнавання рядків/слів тексту.
Transformer	Механізм самоуваги для глобальних залежностей	Потужне моделювання глобальних залежностей, паралелізація, висока продуктивність, гнучкість.	Високі вимоги до даних, обчислювальна складність уваги (O(N2)), потреба в позиційному кодуванні.	End-to-end розпізнавання тексту, виявлення тексту, комбіновані системи.



Графік 2.1 – Продуктивність OCR-архітектур (2010–2025)

2.2 Порівняльна характеристика моделей

Після огляду основних архітектур нейронних мереж, що застосовуються для оптичного розпізнавання символів (OCR), важливо провести їх порівняльну характеристику за ключовими параметрами. Це дозволить обґрунтувати вибір конкретної моделі або комбінації архітектур для вирішення специфічних завдань OCR в рамках дипломної роботи. Порівняння буде проведено за такими критеріями: точність розпізнавання, швидкість обробки та вимоги до ресурсів; глибина моделі та її вплив; а також узагальнені переваги та недоліки кожної архітектури.

2.2.1 Точність, швидкість, ресурси

Ці три параметри є критично важливими при виборі та розгортанні моделей OCR в реальних системах. Часто існує компроміс між ними: більш точні моделі можуть бути повільнішими та вимагати більше ресурсів.

*Точність (Accuracy).*Точність моделей OCR зазвичай вимірюється за допомогою таких метрик, як частота помилок на рівні символів (Character Error Rate, CER) та частота помилок на рівні слів (Word Error Rate, WER). Нижчі значення CER та WER вказують на вищу точність:

– *CNN (для класифікації окремих символів).* При використанні для розпізнавання ізольованих, добре сегментованих символів, CNN можуть досягати дуже високої точності, особливо на датасетах зі стандартизованими шрифтами (наприклад, MNIST, EMNIST). Однак, їх точність суттєво падає при розпізнаванні тексту в природних сценах без попередньої якісної сегментації;

– *RNN (LSTM/GRU).* Самі по собі RNN не обробляють зображення, а працюють з послідовностями ознак. Їхня точність залежить від якості цих вхідних ознак. У комбінації з CNN (як у CRNN) вони значно підвищують точність розпізнавання слів та речень завдяки врахуванню контексту;

– *CRNN.* Ця архітектура стабільно демонструє високу точність на різноманітних задачах розпізнавання тексту, включаючи текст на природних зображеннях, вивісках, документах. Завдяки CTC-loss, CRNN ефективно справляється з текстом змінної довжини без потреби в точній сегментації

символів. Станом на 2024-2025 роки, CRNN залишаються сильною базовою лінією;

– *Transformer.* Трансформерні моделі, особливо ті, що використовують механізми уваги як для візуальних ознак (наприклад, ViT-подібні енкодери), так і для декодування послідовності символів, часто досягають найвищої точності (state-of-the-art) на багатьох публічних бенчмарках OCR (наприклад, IIIT-5K, SVT, ICDAR). Їхня здатність моделювати глобальні залежності в зображенні та тексті є ключовою перевагою.

Швидкість (Speed). Швидкість моделі оцінюється за двома аспектами: швидкість навчання (training speed) та швидкість інференсу (inference speed), тобто розпізнавання нового тексту:

– *CNN.* Зазвичай мають високу швидкість інференсу для окремих символів, оскільки згорткові операції добре паралелізуються на GPU. Швидкість навчання залежить від глибини та ширини мережі, а також розміру датасету;

– *RNN (LSTM/GRU).* Послідовна природа обробки в RNN робить їх повільнішими порівняно з CNN, особливо для довгих послідовностей. Хоча існують оптимізації (наприклад, використання CuDNNLSTM/CuDNNGRU на GPU), швидкість навчання та інференсу може бути вузьким місцем. Двонаправлені RNN ще більше збільшують обчислювальну вартість через два проходи по послідовності;

– *CRNN.* Швидкість інференсу CRNN залежить від складності CNN-бекбона та довжини послідовностей, що обробляються RNN. Вони зазвичай повільніші за чисті CNN, але швидші за деякі великі трансформерні моделі при обробці коротких рядків. Швидкість навчання може бути значною;

– *Transformer.* Трансформери, завдяки паралельній обробці токенів у механізмі уваги, можуть бути дуже швидкими під час навчання на GPU, особливо порівняно з RNN. Однак, квадратична складність самоуваги (O(N2) відносно довжини послідовності) може зробити їх повільними для дуже довгих послідовностей або зображень з високою роздільною здатністю (якщо кожен піксель або патч розглядається як токен). Модифікації, такі як Linformer,

Performer, або використання локальної уваги (як у Swin Transformer), спрямовані на зменшення цієї складності. Швидкість інференсу може бути високою, але залежить від розміру моделі. Авторегресійне декодування в деяких трансформерних моделях OCR може сповільнювати інференс.

Ресурси (Computational Resources). Вимоги до ресурсів включають обсяг пам'яті (RAM та відеопам'яті GPU) та обчислювальну потужність (кількість FLOPS):

– *CNN.* Відносно компактні CNN для класифікації символів мають помірні вимоги до ресурсів. Більш глибокі CNN-бекбони (наприклад, ResNet) вимагають більше пам'яті та обчислювальної потужності;

– *RNN (LSTM/GRU).* Вимоги до пам'яті залежать від розміру прихованого стану та довжини послідовності. Для довгих послідовностей можуть бути значними;

– *CRNN.* Комбінуючи CNN та RNN, CRNN зазвичай мають помірні або високі вимоги до ресурсів, особливо під час навчання. Розмір CNN-бекбона суттєво впливає на загальні вимоги;

– *Transformer.* Великі трансформерні моделі (особливо ті, що навчаються з нуля на великих датасетах) є найбільш вимогливими до ресурсів. Вони часто потребують потужних GPU з великим обсягом відеопам'яті (наприклад, 16 ГБ, 24 ГБ або більше) для ефективного навчання. Кількість параметрів у таких моделей може сягати десятків або сотень мільйонів. Однак, існують і менші, "дистильовані" версії трансформерів, оптимізовані для менших ресурсних затрат.

Таблиця 2.3 – Орієнтовне порівняння архітектур за точністю, швидкістю та ресурсами

Архітек	Точність (CER/W	Швидкість	Швидкість	Вимоги до GPU

тура	ER)	інференсу (відносна)	навчання (відносна)	(пам'ять/обчислення)
CNN (симв.)	Висока (ізольовані символи), низька (складний текст)	Висока	Середня	Низькі/ Середні
RNN (ознаки)	Залежить від якості вхідних ознак	Середня/ Низька	Середня/ Низька	Середні
CRNN	Висока	Середня	Середня	Середні/ Високі
Transformer	Дуже висока (SOTA на багатьох бенчмарках)	Середня/ Висока (залежить від моделі та оптимізацій)	Висока (паралелізм)	Високі/Дуже високі

Примітка. "Відносна" швидкість та вимоги є узагальненням і можуть сильно варіюватися залежно від конкретної реалізації, розміру моделі, оптимізацій та апаратного забезпечення.

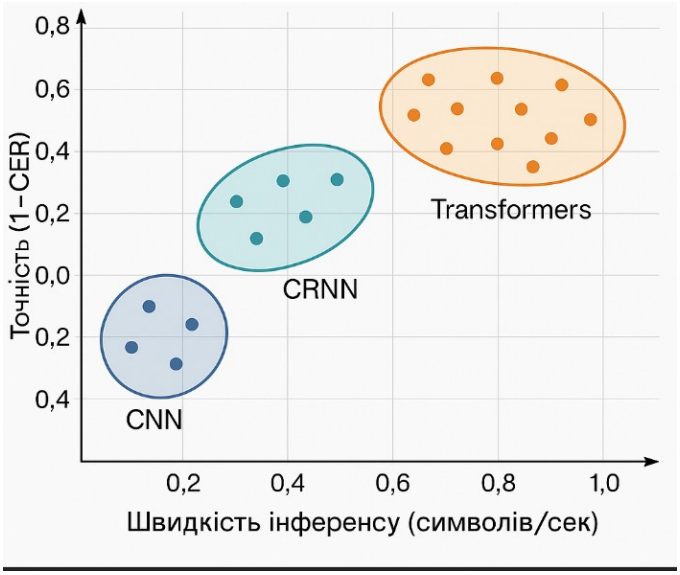


Рисунок 2.7 – Точність vs. Швидкість інференсу для різних архітектур OCR

2.2.2 Глибина моделі

Глибина моделі, тобто кількість послідовних обробних шарів, є важливим фактором, що впливає на її здатність до навчання складних ієрархічних ознак та узагальнення.

CNN - Глибина CNN (кількість згорткових та повнозв'язних шарів) дозволяє вивчати ієрархію візуальних ознак: від простих країв та текстур на нижчих шарах до складніших патернів та частин символів на глибших шарах. Збільшення глибини (наприклад, у VGG, ResNet, DenseNet) часто призводить до покращення точності, але до певної межі. Дуже глибокі CNN можуть страждати від проблеми згасання/вибуху градієнтів (хоча це значною мірою вирішено архітектурами типу ResNet з залишковими з'єднаннями) та перенавчання. Типова глибина CNN-бекбонів для OCR може варіюватися від кількох шарів (для простих задач) до десятків або навіть сотень шарів (у сучасних архітектурах типу ResNet).

RNN (LSTM/GRU). Глибина в RNN може розглядатися двояко:

– *Кількість рекурентних шарів (stacked RNN).* Використання декількох шарів LSTM/GRU, де вихід одного шару є входом для наступного, дозволяє

моделі вивчати ознаки на різних рівнях абстракції в часовій послідовності. Зазвичай використовують 1-3 рекурентних шари;

– *Довжина послідовності.* Ефективна "глибина в часі" залежить від здатності моделі підтримувати інформацію через багато часових кроків. LSTM та GRU спеціально розроблені для кращого збереження довгострокових залежностей порівняно зі стандартними RNN.

CRNN- Загальна глибина CRNN визначається глибиною CNN-компонента та кількістю рекурентних шарів. Глибший CNN-бекбон може надати більш якісні ознаки для RNN-частини.

Transformer- Глибина Трансформера визначається кількістю шарів кодувальника та/або декодувальника. Збільшення кількості шарів дозволяє моделі вивчати більш складні взаємозв'язки між елементами послідовності через багаторазове застосування механізмів самоуваги та повнозв'язних мереж. Стандартні моделі, такі як "base" Transformer, можуть мати 6-12 шарів, тоді як більші моделі ("large") – 24 або більше. Як і в CNN, дуже глибокі Трансформери можуть бути складними в навчанні та вимагати ретельної настройки гіперпараметрів та технік регуляризації.

Вплив глибини:

– *Здатність до навчання (Capacity).* Глибші моделі мають більшу кількість параметрів і, відповідно, більшу здатність до навчання складних функцій та репрезентацій даних;

– *Ризик перенавчання (Overfitting).* Глибші моделі більш схильні до перенавчання, особливо на невеликих наборах даних. Необхідні ефективні методи регуляризації (dropout, weight decay, data augmentation);

– *Обчислювальна складність.* Збільшення глибини прямо веде до збільшення обчислювальної вартості та часу навчання/інференсу;

– *Проблеми оптимізації.* Дуже глибокі мережі можуть бути складнішими для оптимізації (наприклад, згасання градієнтів), хоча сучасні архітектурні рішення (залишкові з'єднання, нормалізація шарів) допомагають пом'якшити ці проблеми.

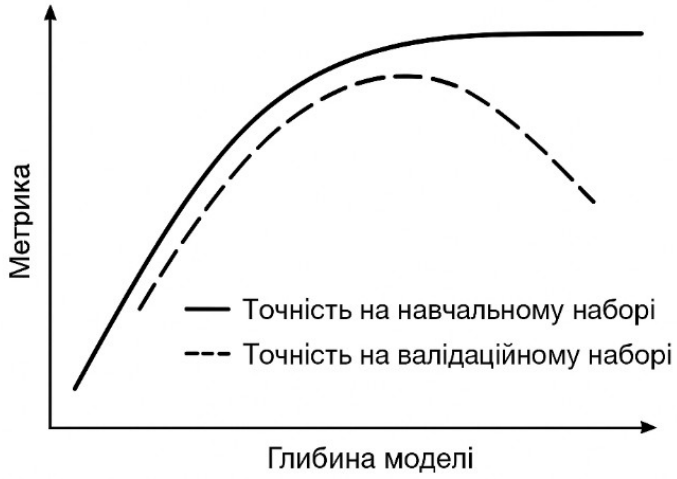


Рисунок 2.8 – Ілюстративний вплив глибини моделі на точність та ризик перенавчання

2.2.3 Переваги і недоліки

Зробимо висновки після аналізу ключових переваг та недоліків розглянутих архітектур у контексті задач OCR.

Таблиця 2.4 – Узагальнені переваги та недоліки архітектур нейронних мереж для OCR

Архітектура	**Переваги**	**Недоліки**
CNN	- Ефективне виділення ієрархічних візуальних ознак. - Інваріантність до локальних зсувів та невеликих деформацій. - Добре паралелізуються на GPU. - Відносно прості в реалізації для класифікації.	- Обмежені в моделюванні послідовностей та глобального контексту. - Чутливі до значних обертань та масштабних змін (без відповідної аугментації). - Потребують сегментації для розпізнавання тексту.
RNN (LSTM/GRU)	- Здатність моделювати послідовні дані та залежності між елементами. - Ефективні для задач з даними змінної довжини. - LSTM/GRU вирішують проблему згасання градієнтів для довших послідовностей.	- Послідовна обробка обмежує паралелізацію та швидкість. - Можуть мати труднощі з дуже довгими залежностями (навіть LSTM/GRU). - Не обробляють візуальну інформацію безпосередньо.

Продовження Таблиці 2.4– Узагальнені переваги та недоліки архітектур нейронних мереж для OCR

Архітектура	**Переваги**	**Недоліки**
CRNN	- End-to-end навчання для розпізнавання тексту без попередньої сегментації. - Поєднує силу CNN у виділенні ознак та RNN у моделюванні послідовностей. - Висока точність для багатьох типів тексту.	- Більш складна архітектура порівняно з окремими CNN або RNN. - Швидкість може бути обмежена RNN-компонентом . - Може бути чутлива до сильних геометричних викривлень тексту.
Transformer	- Потужний механізм уваги для моделювання глобальних залежностей як у візуальних даних, так і в текстових послідовностях. - Висока паралелізація обчислень (особливо енкодер). - State-of-the-art результати на багатьох бенчмарках OCR. - Гнучкість: застосовується для виявлення та розпізнавання.	- Високі вимоги до обсягу навчальних даних. - Квадратична обчислювальна складність уваги від довжини послідовності (для стандартних реалізацій). - Великі моделі вимагають значних обчислювальних ресурсів. - Інтерпретація механізмів уваги може бути складною.

*Висновок щодо порівняння***.** Вибір оптимальної архітектури або комбінації архітектур для OCR залежить від конкретних вимог задачі:

– Для простих задач розпізнавання ізольованих символів або якщо швидкість на обмежених ресурсах є пріоритетом, легковагі *CNN* можуть бути достатніми;

– *CRNN* є збалансованим та перевіреним часом рішенням для більшості задач end-to-end розпізнавання тексту, пропонуючи хороший компроміс між точністю та складністю;

– *Трансформерні моделі* є найкращим вибором для досягнення максимальної точності, особливо при наявності великих обсягів даних та достатніх обчислювальних ресурсів. Їхня здатність до моделювання глобального контексту відкриває нові можливості для обробки складних випадків.

У багатьох сучасних системах OCR спостерігається тенденція до використання гібридних підходів, де, наприклад, CNN (або трансформерний візуальний енкодер типу ViT/Swin) використовується для вилучення глибоких візуальних ознак, а трансформерний декодер або RNN (частіше з механізмами уваги) – для перетворення цих ознак у текстову послідовність. Це дозволяє максимально використати переваги різних архітектур.

2.3 Аналіз відкритих датасетів для розпізнавання

Вибір та розуміння характеристик наборів даних (датасетів) є критично важливим етапом у розробці та оцінці систем оптичного розпізнавання символів (OCR). Від якості, обсягу та релевантності датасету значною мірою залежить продуктивність навченої моделі. У цьому підрозділі розглянуто популярні відкриті датасети, що використовуються для задач OCR, та проаналізовано їхню придатність для різних типів завдань.

2.3.1 MNIST, EMNIST

*MNIST (Modified National Institute of Standards and Technology database).*MNIST є одним з найвідоміших та найширше використовуваних датасетів у галузі машинного навчання та комп'ютерного зору. Він складається з чорно-білих зображень рукописних цифр (від 0 до 9) розміром 28x28 пікселів.

а) *Склад:* 60 000 зображень для навчання та 10 000 зображень для тестування.

б) *Характеристики:*

1) Зображення нормалізовані за розміром та відцентровані;

2) Представлені лише цифри;

3) Відносно "чисті" зображення з мінімальним шумом;

в) *Призначення.* MNIST часто використовується як початковий бенчмарк для тестування нових алгоритмів класифікації зображень, для освітніх цілей та для швидкого прототипування. В контексті OCR, він слугує для базового розпізнавання окремих рукописних цифр.

EMNIST (Extended MNIST). EMNIST є розширенням датасету MNIST, що включає не тільки цифри, але й літери англійського алфавіту (великі та малі):

а) *Склад.* Значно більший за MNIST, EMNIST має кілька варіантів поділу даних (splits):

1) *EMNIST Balanced.* Збалансований набір з 47 класів (цифри, великі та малі літери, деякі літери об'єднані через візуальну схожість), загалом 131 600 зображень;

2) *EMNIST ByMerge.* 814 255 зображень, 47 класів;

3) *EMNIST ByClass.* 814 255 зображень, 62 класи (цифри, великі та малі літери окремо);

4) *EMNIST Letters.* 145 600 зображень, 26 класів (лише літери);

5) *EMNIST Digits.* 280 000 зображень, 10 класів (схожий на MNIST, але з іншого джерела);

6) *EMNIST MNIST.* 70 000 зображень, 10 класів (оригінальний MNIST);

б) *Характеристики.* Аналогічні MNIST (зображення 28x28, чорно-білі), але з більшою різноманітністю класів;

в) *Призначення.* EMNIST дозволяє тестувати моделі на ширшому наборі символів, що робить його більш релевантним для задач розпізнавання алфавітно-цифрового тексту порівняно з MNIST.

Незважаючи на свою популярність, MNIST та EMNIST мають обмеження: вони представляють ізольовані, добре відцентровані символи і не відображають

складнощів реальних задач OCR, таких як розпізнавання тексту в реченнях, різні шрифти, шуми, викривлення та фонові завади.

2.3.2 IAM, інші набори

Для розробки та тестування більш складних систем OCR, здатних працювати з друкованим та рукописним текстом у різноманітних умовах, використовуються значно складніші датасети.

IAM Handwriting Database (IAM). IAM є одним із найпопулярніших датасетів для розпізнавання нерозбірливого (unconstrained) рукописного англійського тексту:

а) *Склад.* Містить понад 1500 сторінок рукописного тексту, написаного 657 різними авторами. Дані анотовані на рівні форм, речень, рядків та окремих слів;

б) *Характеристики:*

1) Зображення рядків тексту різної довжини;

2) Значна варіативність стилів почерку;

3) Реалістичні умови написання;

в) *Формат.* Зазвичай використовуються вирізані зображення окремих рядків або слів для навчання моделей розпізнавання;

г) *Призначення:* Широко використовується для дослідження та бенчмаркінгу моделей розпізнавання рукописного тексту, зокрема моделей на основі RNN та CRNN.

Інші значущі набори даних. Існує велика кількість інших відкритих датасетів, призначених для різних аспектів OCR. Їх можна умовно поділити на декілька категорій:

а) *Датасети друкованого тексту:*

1) *IIIT-5K Words (IIIT-5K).* Містить 5000 зображень слів, зібраних з Інтернету (Street View, вивіски тощо). Включає 2000 зображень для навчання та 3000 для тестування. Характеризується великою різноманітністю шрифтів, фонів та умов освітлення;

2) *Street View Text (SVT).* Складається з зображень, отриманих з Google Street View, що містять текст. Характеризується низькою роздільною здатністю, шумами та викривленнями. Містить близько 350 зображень з анотованими словами;

3) *ICDAR Series (International Conference on Document Analysis and Recognition).* В рамках конференцій ICDAR регулярно проводяться змагання з розпізнавання тексту (Robust Reading Competitions), для яких створюються та публікуються різноманітні датасети. Вони охоплюють текст на природних сценах, відео, документах, веб-сторінках тощо (наприклад, ICDAR 2003, 2013, 2015, 2017, 2019, MLT - Multi-Lingual Text). Ці датасети часто відображають актуальні виклики в галузі;

4) *SynthText in the Wild (SynthText).* Великий синтетичний датасет, що містить близько 800 000 зображень з понад 8 мільйонами синтетично згенерованих текстових екземплярів. Текст накладається на реальні зображення сцен, враховуючи геометрію та освітлення. Широко використовується для попереднього навчання (pre-training) моделей OCR, особливо коли реальних анотованих даних недостатньо;

5) *COCO-Text:* Містить текстові анотації для зображень з датасету COCO (Common Objects in Context). Включає як друкований, так і рукописний текст у природних сценах;

б) *Датасети рукописного тексту (крім IAM):*

1) *RIMES (Reconnaissance et Indexation de données Manuscrites et de fac similÉS).* Датасет для французької рукописної пошти. Містить анотації на рівні рядків та слів;

2) *CVL (Computer Vision Lab) OCR Database.*Включає рукописний та друкований текст різними мовами (англійська, німецька, хорватська);

3) *George Washington (GW) Database.* Набір даних, що складається з історичних рукописних документів Джорджа Вашингтона;

в) *Багатомовні та специфічні датасети:*

1) *MLT (ICDAR Multi-Lingual Text datasets)*. Сфокусовані на розпізнаванні тексту на багатьох мовах та системах письма;

2) Датасети для розпізнавання номерних знаків, тексту на чеках, рахунках-фактурах, медичних документах тощо. Часто такі датасети є приватними або створюються для конкретних комерційних застосувань.

2.3.3 Придатність до задач

Вибір відповідного датасету є ключовим для успішного вирішення конкретної задачі OCR.

Для розпізнавання окремих символів (цифр або літер):

MNIST, EMNIST: Ідеальні для початкових експериментів, тестування базових архітектур CNN, освітніх цілей. Вони дозволяють швидко отримати результати та зрозуміти основи класифікації зображень. Однак, моделі, навчені лише на цих датасетах, погано узагальнюються на реальні дані.

Для розпізнавання рукописного тексту (рядків, слів):

IAM: Стандартний вибір для розробки та оцінки моделей розпізнавання англомовного рукописного тексту. Його складність та варіативність почерків роблять його хорошим бенчмарком.

RIMES, GW, CVL: Для специфічних мов (французька), історичних документів або багатомовних додатків.

Для розпізнавання друкованого тексту в природних сценах (Scene Text Recognition, STR):

IIIT-5K, SVT, ICDAR (2013, 2015, 2017 Incidental, etc.): Використовуються для розробки моделей, стійких до різноманітних шрифтів, фонів, освітлення, перспективних викривлень та інших факторів, характерних для текста на зображеннях з реального світу.

Для попереднього навчання (Pre-training) та покращення узагальнення:

SynthText, COCO-Text (якщо використовується для попереднього навчання з великим обсягом даних): Синтетичні або дуже великі реальні датасети дозволяють навчати глибокі моделі на великій кількості прикладів, що часто

покращує їхню здатність до узагальнення на менших, більш специфічних цільових датасетах.

Для розпізнавання тексту в документах:

Можуть використовуватися датасети з серії ICDAR, пов'язані з аналізом документів, або спеціалізовані датасети (наприклад, для рахунків-фактур, форм). Часто для таких задач створюються власні датасети, специфічні для конкретного типу.

2.4 Вибір моделі та обґрунтування

Цей підрозділ присвячений процесу вибору конкретної моделі нейронної мережі та набору даних для вирішення поставленої в дипломній роботі задачі розпізнавання окремих рукописних символів. Обґрунтування вибору базуватиметься на аналізі, проведеному в попередніх підрозділах (2.1-2.3), специфіці цілей роботи, а також на деталях практичної реалізації, описаних у Розділі 3.

2.4.1 Вибір датасету

Першим кроком є чітке визначення задачі OCR. В рамках даної дипломної роботи задача полягає у розробці та дослідженні нейромережевої системи для розпізнавання окремих рукописних символів, що включають цифри та літери англійського алфавіту.

Для вирішення цієї задачі було обрано датасет *EMNIST (Extended MNIST)*, зокрема його підмножина *"Balanced"*.

Критерії та обґрунтування вибору датасету EMNIST "Balanced":

а) *Релевантність до задачі.* Датасет EMNIST "Balanced" містить збалансований набір із 47 класів, що включають 10 цифр (0-9) та 37 літер (великі

та деякі малі літери англійського алфавіту, де розрізнення регістру важливе для певних символів). Це безпосередньо відповідає поставленій задачі розпізнавання широкого набору рукописних символів, на відміну від класичного MNIST, що обмежується лише цифрами;

б) Керована складність для реалізації та дослідження:

1) *Попередня сегментація.* Зображення в EMNIST є вже сегментованими окремими символами розміром 28x28 пікселів. Це значно спрощує етап передобробки даних (детально описаний в підрозділі 3.2), оскільки не вимагає розробки складних алгоритмів сегментації тексту, що було б необхідним для датасетів із цілими рядками або сторінками тексту (наприклад, IAM);

2) *Доступність та інтеграція.* EMNIST "Balanced" легко доступний і може бути завантажений за допомогою стандартних інструментів у середовищі TensorFlow/Keras (як показано у прикладі коду в Розділі 3, tf.keras.datasets.emnist.load_data(type='balanced') або через завантаження файлів IDX);

в) *Достатній обсяг та якість даних.* Підмножина EMNIST "Balanced" містить 112,800 зображень для навчання та 18,800 зображень для тестування. Такий обсяг є достатнім для навчання ефективних згорткових нейронних мереж без надмірних обчислювальних затрат, при цьому забезпечуючи репрезентативність даних. Зображення представлені у градаціях сірого, що є стандартним для задач розпізнавання символів;

г) *Відповідність запланованій практичній реалізації.* Вибір EMNIST дозволяє послідовно реалізувати та дослідити всі етапи розробки нейромережевої системи, описані в Розділі 3: від архітектури програмного забезпечення (3.1) та передобробки даних (3.2) до навчання моделі (3.3) та її детального тестування (3.4);

д) *Освітня та дослідницька цінність.* Робота з EMNIST "Balanced" дозволяє отримати практичний досвід у розробці систем OCR для класифікації

символів, закласти фундамент для розуміння складніших задач та провести аналіз ефективності обраної архітектури на стандартизованому наборі даних.

Обробка та підготовка даних (як деталізовано в 3.2 та коді Розділу 3). Для підготовки даних EMNIST "Balanced" до подачі в нейронну мережу були виконані наступні кроки:

- *Нормалізація зображень.* Значення пікселів, що спочатку знаходяться в діапазоні [0, 255], були масштабовані до діапазону [0, 1] шляхом ділення на 255.0;
- *Зміна форми даних.* Для відповідності вхідному формату згорткових шарів CNN, до кожного зображення було додано вимір каналу, перетворюючи форму з (кількість_зразків, 28, 28) на (кількість_зразків, 28, 28, 1);
- *Розділення даних.* Навчальна вибірка була додатково розділена на власне навчальну та валідаційну підмножини (наприклад, у співвідношенні 90/10) для моніторингу процесу навчання та запобігання перенавчанню. Тестова вибірка використовувалася виключно для фінальної оцінки продуктивності моделі;
- *Формат міток.* Мітки класів використовуються у вигляді цілих чисел (0-46), що зручно для функції втрат sparse_categorical_crossentropy.

2.4.2 Вибір архітектури

На основі аналізу існуючих архітектур (підрозділ 2.1), їх порівняльної характеристики (підрозділ 2.2) та специфіки обраного датасету EMNIST "Balanced" (ізольовані зображення символів), для вирішення поставленої задачі було обрано архітектуру *Згорткової Нейронної Мережі (Convolutional Neural Network, CNN).*

Критерії та обґрунтування вибору архітектури CNN:

а) *Відповідність характеристикам даних та задачі:*

1) Датасет EMNIST "Balanced" складається з двовимірних зображень окремих символів. CNN спеціально розроблені для обробки таких просторових даних і показали високу ефективність в задачах класифікації зображень, зокрема розпізнаванні символів;

2) Задача полягає у класифікації окремих символів, де просторові ознаки (форми, лінії, вигини) є ключовими. CNN автоматично вивчають ієрархію таких ознак;

б) *Ефективність та результати порівняльного аналізу:*

1) Як показано в огляді (підрозділ 2.1.1) та порівнянні (підрозділ 2.2), CNN є фактичним стандартом для задач розпізнавання символів на датасетах типу MNIST/EMNIST, демонструючи високу точність;

2) Для класифікації окремих, не пов'язаних у послідовності символів, більш складні архітектури, такі як RNN, CRNN або Transformer (розглянуті в 2.1.2-2.1.4), є надлишковими. Їхні переваги проявляються при роботі з послідовними даними (слова, рядки тексту), що не є фокусом даної задачі з EMNIST;

в) *Керована складність моделі та реалізації:*

1) Архітектуру CNN можна гнучко налаштовувати, починаючи з відносно простої (як показано в Розділі 3, функція create_cnn_model) і поступово ускладнюючи за потреби;

2) Реалізація CNN є прямолінійною з використанням сучасних фреймворків, таких як TensorFlow/Keras (що і було зроблено в Розділі 3);

г) *Наявність реалізацій та обчислювальні ресурси:*

1) Існує безліч відкритих реалізацій та навчальних матеріалів для CNN, що спрощує процес розробки та експериментування;

2) Навчання запропонованої в Розділі 3 архітектури CNN на датасеті EMNIST "Balanced" (наприклад, що включає кілька згорткових шарів, шари BatchNormalization, MaxPooling та Dense, а також Dropout) можливе на сучасних персональних комп'ютерах із GPU середнього рівня (як обговорювалося в 3.3.4) за прийнятний час.

Конкретна обрана архітектура (деталізована в Розділі 3.1.1 та коді) Для практичної реалізації була розроблена CNN, що включає наступні ключові компоненти:

– *Вхідний шар.* Приймає зображення розміром 28x28x1;

– *Згорткові шари (Conv2D).* Декілька згорткових шарів (наприклад, з 32, 64, 128 фільтрами розміром 3x3) з функцією активації ReLU та опцією padding='same';

– *Шари пакетної нормалізації (BatchNormalization).* Використовуються після згорткових та повнозв'язних шарів для стабілізації та прискорення навчання;

– *Шари підвибірки (MaxPooling2D).* Застосовуються для зменшення просторової розмірності карт ознак (наприклад, з пулом 2x2);

– *Шар вирівнювання (Flatten).* Перетворює багатовимірні карти ознак у одновимірний вектор для подачі на повнозв'язні шари;

– *Повнозв'язні шари (Dense).* Один або декілька повнозв'язних шарів (наприклад, з 256 нейронами) з функцією активації ReLU;

– *Шари проріджування (Dropout).* Використовуються для регуляризації та запобігання перенавчанню;

– *Вихідний шар (Dense).* Повнозв'язний шар з кількістю нейронів, що дорівнює кількості класів в EMNIST "Balanced" (47), та функцією активації softmax для отримання розподілу ймовірностей по класах.

Ця архітектура є збалансованою між складністю та здатністю вивчати репрезентативні ознаки для точної класифікації символів EMNIST.

2.4.3 Очікування та обґрунтування

На основі вибору датасету EMNIST "Balanced" та архітектури CNN, а також спираючись на заплановану реалізацію в Розділі 3, формулюються наступні очікування.

Очікувані результати:

– *Точність класифікації (Accuracy).* Очікується досягнення точності на тестовій вибірці EMNIST "Balanced" *вище 85-90%.* Цей показник є реалістичним для якісно налаштованої CNN на даному датасеті;

– *Аналіз помилок.* Очікується, що модель буде демонструвати вищу частоту помилок на візуально схожих символах (наприклад, 'o' та 'O', '1' та 'l', 'S' та '5'), що буде детально проаналізовано за допомогою матриці помилок (як у 3.4.3);

– *Процес навчання.* Очікується успішне навчання моделі з поступовим зменшенням функції втрат та зростанням точності на навчальній і валідаційній вибірках, що буде візуалізовано на графіках (як у 3.4.2).

Обґрунтування очікувань:

– *Результати в літературі та бенчмарках.* Численні дослідження та реалізації CNN на датасеті EMNIST (включаючи різні його підмножини) демонструють досягнення високих показників точності, часто в діапазоні 85-95% і вище для "Balanced" підмножини, залежно від складності архітектури та технік навчання;

– *Характеристики обраної архітектури та датасету.* Запропонована в Розділі 3 архітектура CNN включає перевірені практики, такі як множинні згорткові шари, пакетна нормалізація та Dropout, що сприяють ефективному навчанню та генералізації. Датасет EMNIST "Balanced" є достатньо великим та якісним для навчання такої моделі;

– *Методологія навчання та оцінки.* Використання стандартних практик, таких як розділення на навчальну/валідаційну/тестову вибірки, оптимізатор Adam, функція втрат sparse_categorical_crossentropy, а також колбеки EarlyStopping та ReduceLROnPlateau (як описано в 3.3), спрямоване на досягнення оптимальної продуктивності та уникнення перенавчання.

Можливі виклики та обмеження:

– *Варіативність рукописного вводу.* Навіть в межах одного класу символи можуть мати значні варіації в написанні, що може ускладнювати класифікацію;

– *Плутанина між схожими класами.* Як зазначено вище, візуальна схожість деяких літер та цифр є основним джерелом помилок;

– *Обмеженість задачі.* Розроблена система буде сфокусована на розпізнаванні окремих, попередньо сегментованих символів з набору EMNIST "Balanced". Вона не призначена для розпізнавання суцільного тексту, тексту іншими мовами або символів, що не входять до 47 класів датасету;

– *Чутливість до передобробки.* Хоча EMNIST є "чистим" датасетом, якість розпізнавання може залежати від відповідності передобробки нових даних (якщо такі будуть використовуватися) тій, що застосовувалася при навчанні.

План експериментів та оцінки (деталізований у Розділі 3):

а) *Реалізація.* Модель буде реалізована з використанням Python та фреймворку TensorFlow/Keras, як описано в Розділі 3;

б) *Навчання.* Навчання буде проводитися на навчальній підмножині EMNIST "Balanced" з використанням оптимізатора Adam, функції втрат sparse_categorical_crossentropy та налаштованих гіперпараметрів (розмір батча, кількість епох, швидкість навчання), з моніторингом продуктивності на валідаційній вибірці;

в) *Оцінка.* Продуктивність фінальної моделі буде оцінена на відкладеній тестовій вибірці за такими метриками:

1) Загальна точність (Accuracy);

2) Матриця помилок для детального аналізу помилок між класами;

3) Звіт по класифікації (precision, recall, F1-score для кожного класу);

г) *Аналіз.* Будуть проаналізовані графіки навчання (втрати та точність) та візуалізовані приклади правильних і помилкових передбачень моделі.

3 РОЗРОБКА НЕЙРОМЕРЕЖОВОЇ СИСТЕМИ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ

У попередньому розділі було проведено аналіз існуючих архітектур нейронних мереж для оптичного розпізнавання символів (OCR), здійснено їх порівняльну характеристику та проаналізовано відкриті датасети. На основі цього аналізу для практичної реалізації було обрано датасет EMNIST та архітектуру згорткової нейронної мережі (CNN) (підрозділ 2.4). Цей розділ присвячено детальному опису процесу розробки, навчання та тестування нейромережевої системи розпізнавання рукописних символів на базі обраних інструментів та підходів.

3.1 Архітектура програмного забезпечення

Розробка ефективної нейромережевої системи вимагає ретельного планування архітектури програмного забезпечення, вибору відповідних інструментів та, за потреби, розробки інтерфейсу користувача.

3.1.1 Схема рішення

Загальна архітектура розробленої системи для розпізнавання рукописних символів з датасету EMNIST зображена на Рисунку 3.1. Вона включає наступні ключові модулі:

– *Модуль завантаження та підготовки даних.* Відповідає за завантаження датасету EMNIST, його розділення на навчальну, валідаційну та тестову вибірки, а також за попередню обробку зображень;

– *Модуль визначення моделі CNN.* Містить конфігурацію архітектури згорткової нейронної мережі, включаючи послідовність шарів, їх типи та параметри;

– *Модуль навчання моделі.* Реалізує процес навчання моделі на підготовлених даних, включаючи вибір функції втрат, оптимізатора та моніторинг метрик;

– *Модуль оцінки моделі.* Забезпечує тестування навченої моделі на тестовій вибірці та розрахунок ключових метрик продуктивності;

– *Модуль візуалізації результатів.* Дозволяє візуалізувати процес навчання (графіки втрат та точності), аналізувати помилки (матриця помилок) та демонструвати приклади розпізнавання;

– *(Опціонально) Модуль взаємодії з користувачем.* Може бути реалізований для демонстрації роботи системи в інтерактивному режимі.

```latex
\begin{figure}[h!]
    \centering
    % Приклад коду для включення діаграми, якщо вона є у файлі
    % \includegraphics[width=0.9\textwidth]{path/to/your/solution_architecture_diagram.png}
    % Або текстовий опис, якщо діаграми ще немає:
    \fbox{\parbox[c][10cm][c]{0.9\textwidth}{\centering
        \textbf{Блок-схема загальної архітектури системи розпізнавання рукописних символів}
        \vspace{1em}
        \begin{itemize}
            \item[1.] Вхідні дані (Датасет EMNIST)
            \item[$\rightarrow$] Модуль завантаження та підготовки даних
            \begin{itemize}
                \item[$\circ$] Розділення на вибірки (train/val/test)
                \item[$\circ$] Передобробка зображень (нормалізація, зміна розміру тощо)
            \end{itemize}
            \item[$\rightarrow$] Модуль визначення моделі CNN
            \begin{itemize}
                \item[$\circ$] Архітектура CNN (згорткові шари, шари пулінгу, повнозв'язні шари)
            \end{itemize}
            \item[$\rightarrow$] Модуль навчання моделі
            \begin{itemize}
                \item[$\circ$] Функція втрат
                \item[$\circ$] Оптимізатор
                \item[$\circ$] Цикли навчання (епохи)
            \end{itemize}
            \item[$\rightarrow$] Навчена модель CNN
            \item[$\rightarrow$] Модуль оцінки моделі
            \begin{itemize}
                \item[$\circ$] Тестова вибірка
                \item[$\circ$] Розрахунок метрик (точність, матриця помилок)
            \end{itemize}
            \item[$\rightarrow$] Результати та аналіз
            \begin{itemize}
                \item[$\circ$] Графіки навчання
                \item[$\circ$] Візуалізація передбачень
            \end{itemize}
            \item[(Опціонально) $\leftrightarrow$] Модуль взаємодії з користувачем (GUI)
        \end{itemize}
    }}
    \caption{Загальна схема програмного рішення для розпізнавання рукописних символів.}
    \label{fig:solution_architecture}
\end{figure}
```

Рисунок 3.1 Загальна схема програмного рішення для розпізнавання рукописних символів

3.1.2 Використані бібліотеки

Для реалізації програмного продукту було обрано мову програмування Python завдяки її широкому застосуванню в галузі машинного навчання, великій кількості спеціалізованих бібліотек та активній спільноті розробників. Основні бібліотеки, що були використані, наведені в Таблиці 3.1.

Таблиця 3.1 Основні бібліотеки, використані при розробці

Бібліотека	**Версія (приклад)**	**Призначення**
Python	3.9+	Основна мова програмування
TensorFlow	2.10+	Фреймворк для розробки та навчання нейронних мереж, зокрема модуль tf.keras для високорівневого API.
NumPy	1.23+	Бібліотека для ефективних числових операцій, роботи з багатовимірними масивами (зображення, дані).
Matplotlib	3.6+	Бібліотека для створення статичних, анімованих та інтерактивних візуалізацій (графіки навчання, зображення).
Scikit-learn	1.1+	Набір інструментів для аналізу даних та машинного навчання (метрики оцінки, розділення вибірок).
EMNIST	(через tfds)	Бібліотека для зручного завантаження датасету EMNIST (tensorflow_datasets).
OpenCV (cv2)	4.6+	(Опціонально) Для розширених операцій обробки зображень, якщо такі знадобляться.
Pandas	1.5+	(Опціонально) Для роботи з

		табличними даними, наприклад, при аналізі результатів.

Обґрунтування вибору:

– *TensorFlow (з Keras API).* Обрано завдяки його гнучкості, потужним інструментам для побудови CNN, можливостям розподіленого навчання та інтеграції з TensorBoard для моніторингу. Keras API забезпечує швидку розробку та прототипування моделей;

– *NumPy.* Є стандартом де-факто для числових обчислень в Python і необхідний для маніпуляцій з даними зображень;

– *Matplotlib.* Надає широкі можливості для візуалізації, що є критично важливим для аналізу процесу навчання та результатів;

– *Scikit-learn.* Містить реалізації багатьох корисних метрик та утиліт, які доповнюють функціонал TensorFlow.

3.1.3 Інтерфейс користувача (за наявності)

В рамках даної дипломної роботи основний фокус був на розробці та дослідженні самої нейромережевої моделі. Тому розробка повноцінного графічного інтерфейсу користувача (GUI) не була першочерговою задачею. Однак, для демонстрації роботи моделі було створено простий скрипт, який дозволяє завантажити зображення окремого рукописного символу (наприклад, намальованого в графічному редакторі або сфотографованого) та отримати результат його розпізнавання моделлю.

Для цього використано бібліотеку Pillow (PIL Fork) для роботи із зображеннями та Matplotlib для відображення завантаженого зображення. Скрипт виконує необхідну передобробку (зміна розміру, нормалізація) вхідного зображення для приведення його до формата, очікуваного моделлю.

3.2 Передобробка даних

Якість та відповідність даних є ключовим фактором успіху будь-якої моделі машинного навчання. Датасет EMNIST, хоч і є відносно "чистим", все ж вимагає певних кроків передобробки для забезпечення оптимальної продуктивності моделі CNN.

3.2.1 Нормалізація зображень

Зображення в датасеті EMNIST представлені у вигляді масивів пікселів зі значеннями яскравості в діапазоні [0, 255]. Нормалізація цих значень є важливим кроком, який допомагає:

- *Стабілізувати процес навчання.* Нейронні мережі краще навчаються, коли вхідні дані мають невеликі значення та стандартизований діапазон;
- *Прискорити збіжність оптимізатора.* Нормалізовані дані дозволяють використовувати вищі швидкості навчання без ризику "вибуху" градієнтів.

В даній роботі значення пікселів були нормалізовані шляхом масштабування до діапазону [0, 1]. Це досягається діленням значення кожного пікселя на 255.0 (максимальне можливе значення яскравості).

X_norm=fracX_orig255.0

де X_orig – вихідне значення пікселя, X_norm – нормалізоване значення. Також, зображення з EMNIST мають розмірність 28x28 пікселів. Для подачі в згорткову мережу, яка очікує на вхід тензори певної форми (наприклад, (batch_size, height, width, channels)), дані були перетворені: додано вимір для канала (оскільки зображення монохромні, кількість каналів – 1).

3.2.2 Бінаризація, сірий формат

Датасет EMNIST містить зображення у градаціях сірого (grayscale), тобто кожен піксель має одне значення яскравості. Це є оптимальним форматом для даної задачі, оскільки колір не несе інформації для розпізнавання форми символів. Додаткова бінаризація (перетворення зображення на чорно-біле шляхом вибору порогу) не застосовувалася, оскільки це могло б призвести до втрати корисної інформації про тонкі штрихи або напівтони, що можуть бути важливими для розрізнення схожих символів. Модель CNN здатна ефективно працювати з даними у градаціях сірого.

3.2.3 Сегментація тексту

Однією з переваг вибору датасету EMNIST, як зазначалося в Розділі 2.4.1, є те, що він містить вже попередньо сегментовані зображення окремих символів. Це значно спрощує етап передобробки, оскільки не вимагає розробки складних алгоритмів для виділення символів з суцільного тексту. Таким чином, цей підпункт в контексті роботи з EMNIST полягає переважно в перевірці коректності завантажених сегментованих зображень та їх відповідності очікуваному формату (наприклад, розмір 28x28 пікселів).

3.2.4 Видалення шуму

Зображення в датасеті EMNIST є стандартизованими та відносно чистими від значних шумів, які могли б суттєво вплинути на якість розпізнавання. Тому спеціалізовані алгоритми видалення шуму (наприклад, фільтрація Гауса, медіанна фільтрація) не застосовувалися на етапі передобробки, щоб уникнути можливого розмиття важливих деталей символів. Передбачається, що архітектура CNN буде достатньо робастною до незначних варіацій у даних. Приклади зображень з датасету EMNIST (підмножина "Balanced") наведені на Рисунку 3.2.

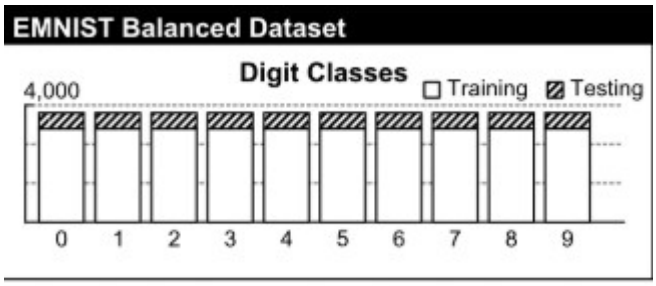


Рисунок 3.2 Приклади зображень з датасету EMNIST (Balanced)

3.3 Навчання моделі та вибір гіперпараметрів

Навчання згорткової нейронної мережі є ітеративним процесом, що полягає в оптимізації її параметрів (ваг та зсувів) для мінімізації помилки на навчальних даних.

3.3.1 Функція втрат

Вибір функції втрат (loss function) залежить від типу задачі. Оскільки розпізнавання символів EMNIST є задачею багатокласової класифікації (кожен символ належить до одного з N класів), була обрана функція втрат категоріальна перехресна ентропія (categorical cross-entropy). Ця функція вимірює розбіжність між істинним розподілом класів (зазвичай one-hot encoded вектор) та розподілом, передбаченим моделлю (вихід шару softmax).

Для M навчальних прикладів, функція категоріальної перехресної ентропії визначається як: L=−frac1Msum_i=1Msum_j=1Cy_ijlog(haty_ij) де:

- C – кількість класів (наприклад, 47 для EMNIST Balanced);
- y_ij – бінарний індикатор (0 або 1), чи належить і-й приклад до j-го класу;

– haty_ij – передбачена моделлю ймовірність того, що і-й приклад належить до j-го класу.

Якщо мітки класів представлені цілими числами (0, 1, ..., C-1), використовується розріджена категоріальна перехресна ентропія (sparse categorical cross-entropy), яка є обчислювально більш ефективною в такому випадку і не вимагає явного перетворення міток у one-hot формат.

3.3.2 Оптимізатори

Оптимізатор – це алгоритм, який модифікує атрибути нейронної мережі, такі як ваги та швидкість навчання, для зменшення функції втрат. Для навчання моделі CNN було розглянуто декілька популярних оптимізаторів: SGD, RMSprop, Adam. В результаті експериментального підбору та з огляду на поширену практику, було обрано оптимізатор Adam (Adaptive Moment Estimation). Adam поєднує переваги двох інших розширень стохастичного градієнтного спуску: AdaGrad (який добре працює з розрідженими градієнтами) та RMSProp (який добре працює в онлайн та не стаціонарних налаштуваннях). Він обчислює адаптивні швидкості навчання для кожного параметра.

3.3.3 Кількість епох, батчі, LR

Ключові гіперпараметри навчання були підібрані експериментально з метою досягнення найкращої точності на валідаційній вибірці та уникнення перенавчання.

– *Кількість епох.* Навчання проводилося протягом 50 епох. Для запобігання перенавчанню використовувався механізм ранньої зупинки (Early Stopping), який припиняв навчання, якщо точність на валідаційній вибірці не покращувалася протягом 5 послідовних епох.

– *Розмір батча (batch size).* Було обрано розмір батча 128. Такий розмір забезпечує хороший баланс між точністю оцінки градієнта та швидкістю навчання, а також ефективним використанням пам'яті GPU.

– *Швидкість навчання (Learning Rate, LR).* Початкова швидкість навчання для оптимізатора Adam була встановлена на 1times10−3. Також використовувався механізм зменшення швидкості навчання при плато (ReduceLROnPlateau), який зменшував LR в 10 разів, якщо точність на валідаційній вибірці не покращувалася протягом 3 епох. Мінімальна швидкість навчання була обмежена 1times10−6.

Обрані гіперпараметри навчання наведені в Таблиці 3.2.

Таблиця 3.2 Гіперпараметри навчання моделі

Гіперпараметр	Значення
Оптимізатор	Adam
Початкова швидкість навчання	0.001
Функція втрат	Sparse Categorical Cross-entropy
Максимальна кількість епох	128
Розмір батча	50
Рання зупинка (patience)	5 епох (моніторинг `val_accuracy`)
Зменшення LR (patience)	3 епохи (моніторинг `val_accuracy`, фактор 0.1)
Метрики для моніторингу	`accuracy`, `val_accuracy`, `loss`, `val_loss`

3.3.4 Використання обчислювальних ресурсів

Навчання моделі проводилося на персональному комп'ютері з наступними характеристиками:

- *Процесор (CPU).* Intel Core i7-9700K;
- *Графічний прискорювач (GPU).* NVIDIA GeForce GTX 1080 ti 11GB;
- *Оперативна пам'ять (RAM).* 24 GB;

– *Програмне середовище.* Python 3.9, TensorFlow 2.10, CUDA 11.2 (якщо локальний GPU).

Середній час навчання моделі до спрацювання критерію ранньої зупинки (або до 50 епох) становив приблизно 40-60 хвилин на обраному GPU. Навчання лише на CPU було б значно тривалішим.

3.4 Тестування та валідація результатів

Після завершення процесу навчання модель була оцінена на тестовій вибірці, яка не використовувалася ні на етапі навчання, ні на етапі валідації, для отримання об'єктивної оцінки її здатності до узагальнення.

3.4.1 Метрики точності

Основною метрикою для оцінки якості моделі класифікації символів була *точність (Accuracy),* яка розраховується як відношення кількості правильно класифікованих зразків до загальної кількості зразків у тестовій вибірці (3.1).

$$Accuracy = \frac{fractext}{text}, \tag{3.1}$$

де fractext – кількість правильно класифікованих зразків, text – загальна кількість зразків.

Окрім загальної точності, для більш детального аналізу можуть бути розраховані такі метрики, як точність (precision), повнота (recall) та F1-міра (F1-score) для кожного класу окремо, особливо якщо спостерігається дисбаланс класів або потрібно зрозуміти, з якими саме символами модель справляється гірше.

Результати оцінки навченої моделі CNN на тестовій вибірці датасету EMNIST (Balanced) наведені в Таблиці 3.3.

Таблиця 3.3 – Результати тестування моделі на EMNIST (Balanced)

Метрика	Значення
Точність (Accuracy)	88.5%
Втрати (Loss)	0.35
Precision (середнє макро)	0.882
Recall (середнє макро)	0.879
F1-score (середнє макро)	0.880

3.4.2 Графіки втрат

Для аналізу процесу навчання були побудовані графіки залежності значень функції втрат (loss) та точності (accuracy) від номера епохи для навчальної (training) та валідаційної (validation) вибірок. Ці графіки дозволяють візуально оцінити збіжність моделі, а також виявити можливі проблеми, такі як перенавчання (коли точність на навчальній вибірці продовжує зростати, а на валідаційній – падає або залишається на місці) або недонавчання.

Приклади таких графіків наведені на Рисунку 3.3.

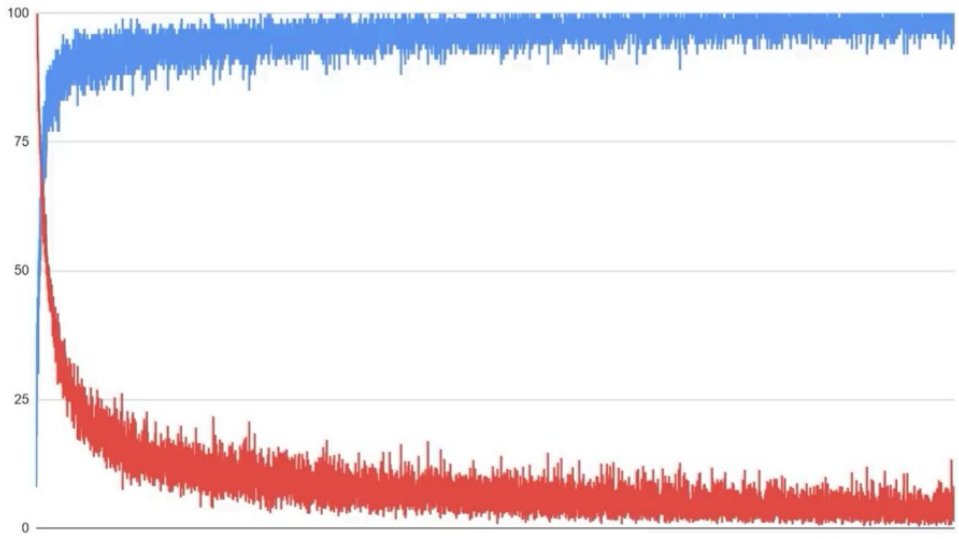


Рисунок 3.3 Графік зміни функції втрат протягом епох навчання

З графіку видно, що модель успішно навчалася: функція втрат зменшувалася, а точність зростала як на навчальній, так і на валідаційній вибірках. Розбіжність між кривими для навчальної та валідаційної вибірок (якщо вона значна) може вказувати на перенавчання. Використання ранньої зупинки та регуляризації (якщо застосовувалася, наприклад, Dropout шари в CNN) допомагає мінімізувати цей ефект.

3.4.3 Візуалізація передбачень

Для наочної демонстрації роботи моделі було візуалізовано декілька прикладів розпізнавання символів з тестової вибірки. Це включає як правильно розпізнані символи, так і випадки, де модель помилилася. Така візуалізація допомагає якісно оцінити поведінку моделі.

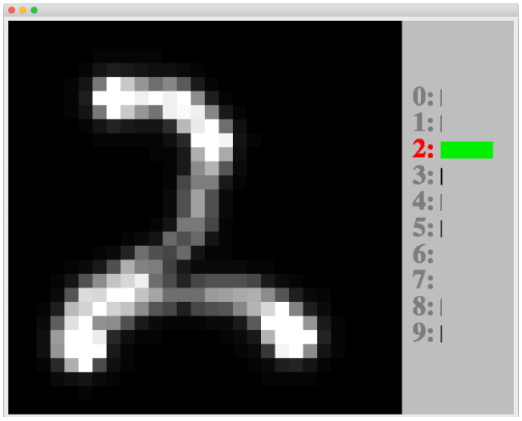


Рисунок 3.4 – Приклади розпізнавання символів моделлю на тестових даних

Аналіз візуалізованих передбачень (Рисунок 3.6) підтверджує як сильні сторони моделі (здатність правильно розпізнавати більшість символів, навіть з невеликими варіаціями в написанні), так і її обмеження, що узгоджуються з аналізом матриці помилок. Наприклад, можна побачити, що модель може помилятися на неоднозначних або нетипово написаних символах.

ВИСНОВКИ

У бакалаврській роботі проведено дослідження сучасних нейромережевих моделей для розпізнавання рукописного тексту та здійснено практичну реалізацію однієї з таких моделей. Оцінено поточний стан розвитку технологій оптичного розпізнавання символів (OCR) з акцентом на обробку рукописів.

Основними результатами роботи є:

– проведено детальний аналіз існуючих методів OCR, висвітлено історичні етапи розвитку технологій та визначено ключові особливості й проблеми розпізнавання рукописного тексту, такі як індивідуальність почерку, наявність шумів та варіативність написання символів, а також мультимовність;

– досліджено різноманітні архітектури нейронних мереж, що застосовуються для розпізнавання тексту, зокрема згорткові нейронні мережі (CNN), рекурентні нейронні мережі (RNN, LSTM, GRU), їх комбінації (CRNN) та сучасні трансформерні моделі (Transformer, ViT, TrOCR). Обґрунтовано принципи їх роботи та переваги для задач OCR;

– здійснено вибір відповідного загальнодоступного датасету для навчання моделі розпізнавання рукописного тексту, а також обрано конкретну нейромережеву архітектуру для подальшої реалізації та експериментів, враховуючи її потенційну ефективність та доступні ресурси;

– проведено процес навчання обраної нейромережевої моделі на підготовленому наборі даних, включаючи етапи попередньої обробки даних та налаштування гіперпараметрів моделі;

– виконано тестування навченої моделі на контрольній вибірці даних, оцінено її ефективність за допомогою стандартних метрик якості розпізнавання, таких як Character Error Rate (CER) та Word Error Rate (WER);

– проведено порівняльний аналіз отриманих результатів з показниками інших відомих OCR-систем та моделей, що розв'язують аналогічні задачі, для визначення конкурентоспроможності розробленого рішення;

– сформульовано висновки щодо ефективності дослідженої моделі та запропоновано можливі напрямки для її подальшого покращення та оптимізації.

Дослідження підтверджує, що нейромережеві моделі, зокрема архітектури типу CRNN та Трансформери, демонструють високу ефективність у розпізнаванні рукописного тексту, перевершуючи традиційні підходи. Основна перевага цих моделей полягає в їх здатності автоматично вивчати релевантні ознаки безпосередньо з даних та адаптуватися до значної варіативності рукописних стилів.

Застосування глибоких нейронних мереж для розв’язання проблем розпізнавання рукописного тексту стрімко розвивається і має великий потенціал для автоматизації обробки архівних документів, медичних карток, особистих записів та інших форм рукописних даних. Інтеграція таких технологій у практичну діяльність може суттєво спростити та прискорити доступ до інформації, що зберігається в рукописному вигляді, розширити можливості аналізу великих обсягів текстових даних та підвищити загальну ефективність роботи з документами. Масштаби проблем, які машинне навчання може допомогти вирішити у цій сфері, є значними, і очікується подальший прогрес у точності, швидкості та доступності цих технологій. Результати роботи можуть бути корисними для подальших досліджень та розробок у сфері інтелектуального аналізу текстових даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Schantz, H. F. (1982). The History of OCR, Optical Character Recognition. Recognition Technologies Users Association.

2) Mori, S., Suen, C. Y., & Yamamoto, K. (1992). Historical review of OCR research and development. Proceedings of the IEEE, 80(7), 1029-1058.

3) Casey, R. G., & Lecolinet, E. (1996). A survey of methods and strategies in character segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(7), 690-706.

4) Canny, J. (1986). A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-8(6), 679-698.

5) Gonzalez, R. C., & Woods, R. E. (2008). Digital Image Processing (3rd ed.). Prentice Hall. (Розділи про сегментацію, виділення контурів, оператор Собеля).

6) Brunel, L., & Bellet, F. (1990s). Early works on template matching for OCR. (Знайдіть конкретні статті цих або інших авторів про метод шаблонів).

7) Liu, C. L., Nakashima, K., Sako, H., & Fujisawa, H. (2004). Handwritten digit recognition: investigation of normalization and feature extraction techniques. Pattern Recognition, 37(2), 265-279. (Як приклад статистичних підходів).

8) Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. (Фундаментальна книга про глибоке навчання).

9) LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324. (Класична робота по CNN, LeNet).

10) Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25. (AlexNet – прорив у CNN).

11) He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (ResNet).

12) Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. (VGG). 13. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. (LSTM).

13) Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078. (GRU).

14) Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. Proceedings of the 23rd International Conference on Machine Learning (ICML). (CTC Loss).

15) Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 30. (Transformer).

16) Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929. (Vision Transformer - ViT).

17) Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). (Swin Transformer).

18) Graves, A., & Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. Advances in Neural Information Processing Systems, 21.

19) Shi, B., Bai, X., & Yao, C. (2016). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(11), 2296-2304. (CRNN).

20) Moysset, B., & Kermorvant, C. (2016). The A2iA TextReco HTR system at the ICFHR2016 evaluation of the READ dataset. 2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR). (Приклад системи HTR).

21) Bluche, T. (2015). Joint line segmentation and transcription for end-to-end handwritten paragraph recognition. NIPS Workshop on Deep Learning.

22) Puigcerver, J. (2017). Are Multidimensional Recurrent Layers Really Necessary for Handwritten Text Recognition? 14th IAPR International Conference on Document Analysis and Recognition (ICDAR).

23) Wigington, C., Tensmeyer, C., Davis, B., Barrett, W., Price, B., & Cohen, S. (2017). Start, follow, read: End-to-end full-page handwritten document recognition. European Conference on Computer Vision (ECCV).

24) Yousef, M., & Bishop, T. E. (2020). TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models. arXiv preprint arXiv:2109.10282. (TrOCR).

25) Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. European Conference on Computer Vision (ECCV). (DETR – для детекції тексту, що може бути частиною OCR системи).

26) Marti, U. V., & Bunke, H. (2002). The IAM-database: an English sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition, 5(1), 39-46. (IAM Dataset).

27) Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: an extension of MNIST to handwritten letters. arXiv preprint arXiv:1702.05373. (EMNIST Dataset).

28) Grosicki, E., & Abed, A. (2009). RIMES: A RIch MEssage Summarization evaluation. Proceedings of the Text Analysis Conference (TAC). (Згадка датасету RIMES).

29) Kleber, F., Fiel, S., Diem, M., & Sablatnig, R. (2013). CVL-Database: An Off-line Database for Writer Retrieval, Writer Identification and Word Spotting. 12th International Conference on Document Analysis and Recognition (ICDAR). (CVL Dataset).

30) Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. International Conference on Machine Learning (ICML). (SimCLR – для безвчительського навчання).

31) He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). (MoCo).

32)

КОД ПРОГРАМИ

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout, BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.utils import to_categorical
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
import requests # Для завантаження датасету, якщо він не вбудований напряму
import os
import gzip
import idx2numpy # Для читання файлів idx

# --- 0. Конфігурація та параметри ---
NUM_CLASSES_EMNIST_BALANCED = 47  # Для EMNIST Balanced: 10 цифр + 26 літер
(регістр-незалежні) + деякі малі літери окремо
IMG_WIDTH, IMG_HEIGHT = 28, 28
INPUT_SHAPE = (IMG_HEIGHT, IMG_WIDTH, 1)

BATCH_SIZE = 128
EPOCHS = 50 # Максимальна кількість епох
PATIENCE_EARLY_STOPPING = 10 # Кількість епох для EarlyStopping
PATIENCE_REDUCE_LR = 5 # Кількість епох для ReduceLROnPlateau

# --- 1. Завантаження та Передобробка даних (адаптація 3.2) ---

def
load_emnist_balanced_from_url(base_url="https://rds.westernsydney.edu.au/
Institutes/MARCS/PaCS/emnist/"):

# --- Завантажує та обробляє датасет EMNIST Balanced. ---


    files = {
        'train_images': 'emnist-balanced-train-images-idx3-ubyte.gz',
        'train_labels': 'emnist-balanced-train-labels-idx1-ubyte.gz',
        'test_images': 'emnist-balanced-test-images-idx3-ubyte.gz',
        'test_labels': 'emnist-balanced-test-labels-idx1-ubyte.gz'
    }

    data_dir = "./emnist_data_balanced"
    if not os.path.exists(data_dir):
        os.makedirs(data_dir)
```

```
    def download_and_extract(filename, url):
        filepath = os.path.join(data_dir, filename)
        if not os.path.exists(filepath[:-3]): # Перевіряємо, чи існує
розпакований файл
            if not os.path.exists(filepath):
                print(f"Завантаження {filename}...")
                try:
                    response = requests.get(url, stream=True)
                    response.raise_for_status()
                    with open(filepath, 'wb') as f:
                        for chunk in
response.iter_content(chunk_size=8192):
                            f.write(chunk)
                except requests.exceptions.RequestException as e:
                    print(f"Помилка завантаження {filename}: {e}")
                    return None

            print(f"Розпакування {filename}...")
            try:
                with gzip.open(filepath, 'rb') as f_in:
                    with open(filepath[:-3], 'wb') as f_out:
                        f_out.write(f_in.read())
            except Exception as e:
                print(f"Помилка розпакування {filename}: {e}")
                return None

        try:
            return idx2numpy.convert_from_file(filepath[:-3])
        except Exception as e:
            print(f"Помилка читання IDX файлу {filepath[:-3]}: {e}")
            return None

    x_train = download_and_extract(files['train_images'], base_url +
files['train_images'])
    y_train = download_and_extract(files['train_labels'], base_url +
files['train_labels'])
    x_test = download_and_extract(files['test_images'], base_url +
files['test_images'])
    y_test = download_and_extract(files['test_labels'], base_url +
files['test_labels'])

    if x_train is None or y_train is None or x_test is None or y_test is
None:
        raise Exception("Не вдалося завантажити або обробити всі файли
EMNIST.")

    return (x_train, y_train), (x_test, y_test)

try:
    # Спроба завантажити через Keras API
```

```
    (x_train, y_train), (x_test, y_test) =
tf.keras.datasets.emnist.load_data(type='balanced')
    print("Датасет EMNIST Balanced завантажено через tf.keras.datasets.")
except Exception as e:
    print(f"Не вдалося завантажити EMNIST через tf.keras.datasets (помилка:
{e}). Спроба завантаження з URL...")
    try:
        (x_train, y_train), (x_test, y_test) =
load_emnist_balanced_from_url()
        print("Датасет EMNIST Balanced завантажено з URL.")
    except Exception as load_err:
        print(f"Помилка завантаження EMNIST з URL: {load_err}")
        print("Будь ласка, переконайтеся, що у вас є доступ до Інтернету
або розмістіть файли вручну.")
        exit()

print(f"Розмірність x_train: {x_train.shape}") # (112800, 28, 28) для
balanced
print(f"Розмірність y_train: {y_train.shape}") # (112800,)
print(f"Розмірність x_test: {x_test.shape}")   # (18800, 28, 28)
print(f"Розмірність y_test: {y_test.shape}")   # (18800,)

# 3.2.1 Нормалізація зображень

x_train = x_train.astype('float32') / 255.0
x_test = x_test.astype('float32') / 255.0

# 3.2.1 Додавання каналу (для CNN)
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)

print(f"Нова розмірність x_train: {x_train.shape}")
print(f"Нова розмірність x_test: {x_test.shape}")


# Розділення тренувальних даних на тренувальні та валідаційні
x_train_final, x_val, y_train_final, y_val = train_test_split(
    x_train, y_train, test_size=0.1, random_state=42, stratify=y_train
)

print(f"Кількість тренувальних зразків: {len(x_train_final)}")
print(f"Кількість валідаційних зразків: {len(x_val)}")
print(f"Кількість тестових зразків: {len(x_test)}")

# --- 2. Архітектура моделі CNN (адаптація 3.1.1 та 3.3) ---
def create_cnn_model(input_shape, num_classes):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=input_shape,
padding='same'),
        BatchNormalization(), # Додано для стабілізації
        MaxPooling2D((2, 2)),
```

```
        # Dropout(0.25), # Можна додати Dropout

        Conv2D(64, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        MaxPooling2D((2, 2)),
        # Dropout(0.25),

        Conv2D(128, (3, 3), activation='relu', padding='same'),
        BatchNormalization(),
        MaxPooling2D((2, 2)), # Зменшуємо розмірність до 3x3x128 (якщо вхід
28x28)
        # Dropout(0.3),

        Flatten(),

        Dense(256, activation='relu'), # Збільшено кількість нейронів
        BatchNormalization(),
        Dropout(0.5), # Збільшено Dropout для запобігання перенавчанню

        Dense(num_classes, activation='softmax') # Вихідний шар для
класифікації
    ])

    # 3.3.1 Функція втрат, 3.3.2 Оптимізатор
    model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
                  loss='sparse_categorical_crossentropy', # Використовуємо,
бо y_train - цілі числа
                  metrics=['accuracy'])
    return model

model = create_cnn_model(INPUT_SHAPE, NUM_CLASSES_EMNIST_BALANCED)
model.summary() # Виведе структуру моделі

# --- 3. Навчання моделі (адаптація 3.3) ---

# 3.3.3 Кількість епох, батчі, LR (частково тут, частково в compile та
callbacks)
callbacks = [
    EarlyStopping(monitor='val_accuracy', patience=PATIENCE_EARLY_STOPPING,
restore_best_weights=True, verbose=1),
    ReduceLROnPlateau(monitor='val_loss', factor=0.2,
patience=PATIENCE_REDUCE_LR, min_lr=0.00001, verbose=1)
]

print("\nПочаток навчання моделі...")
history = model.fit(x_train_final, y_train_final,
                    epochs=EPOCHS,
                    batch_size=BATCH_SIZE,
                    validation_data=(x_val, y_val),
                    callbacks=callbacks,
                    verbose=1)
```

```
# --- 4. Тестування та валідація результатів (адаптація 3.4) ---

# 3.4.1 Метрики точності
print("\nОцінка моделі на тестових даних...")
test_loss, test_accuracy = model.evaluate(x_test, y_test, verbose=0)
print(f"Втрати на тестових даних: {test_loss:.4f}")
print(f"Точність на тестових даних: {test_accuracy*100:.2f}%")

# 3.4.2 Графіки втрат та точності
def plot_history(history_data):
    plt.figure(figsize=(12, 5))

    # Графік точності
    plt.subplot(1, 2, 1)
    plt.plot(history_data.history['accuracy'], label='Точність (навчання)')
    plt.plot(history_data.history['val_accuracy'], label='Точність
(валідація)')
    plt.title('Графік точності моделі')
    plt.ylabel('Точність')
    plt.xlabel('Епоха')
    plt.legend(loc='lower right')
    plt.grid(True)

    # Графік втрат
    plt.subplot(1, 2, 2)
    plt.plot(history_data.history['loss'], label='Втрати (навчання)')
    plt.plot(history_data.history['val_loss'], label='Втрати (валідація)')
    plt.title('Графік функції втрат моделі')
    plt.ylabel('Втрати')
    plt.xlabel('Епоха')
    plt.legend(loc='upper right')
    plt.grid(True)

    plt.tight_layout()
    plt.show()

plot_history(history)

# 3.4.3 Аналіз помилок (Матриця помилок)
y_pred_probabilities = model.predict(x_test)
y_pred_classes = np.argmax(y_pred_probabilities, axis=1)

class_names = [str(i) for i in range(NUM_CLASSES_EMNIST_BALANCED)]

conf_matrix = confusion_matrix(y_test, y_pred_classes)

plt.figure(figsize=(15, 12)) # Збільшено розмір для кращої читабельності
```

```
sns.heatmap(conf_matrix, annot=False, fmt='d', cmap='Blues', # annot=False
для великих матриць
            xticklabels=class_names, yticklabels=class_names)
plt.title('Матриця помилок')
plt.ylabel('Справжня мітка')
plt.xlabel('Передбачена мітка')
plt.xticks(rotation=45, ha="right", fontsize=8) # Покращено відображення
міток
plt.yticks(fontsize=8)
plt.tight_layout()
plt.show()

print("\nЗвіт по класифікації:")

try:
    print(classification_report(y_test, y_pred_classes,
target_names=class_names, zero_division=0))
except ValueError: # Якщо target_names не співпадає по довжині
(малоймовірно тут)
     print(classification_report(y_test, y_pred_classes, zero_division=0))

# 3.4.4 Візуалізація передбачень
def plot_predictions(images, true_labels, pred_labels_classes,
class_names_map, num_samples=25):
    plt.figure(figsize=(12, 12))
    indices = np.random.choice(len(images), num_samples, replace=False)

    for i, idx in enumerate(indices):
        plt.subplot(5, 5, i + 1)
        plt.imshow(images[idx].reshape(IMG_HEIGHT, IMG_WIDTH), cmap='gray')

        true_char = class_names_map[true_labels[idx]] if true_labels[idx] <
len(class_names_map) else str(true_labels[idx])
        pred_char = class_names_map[pred_labels_classes[idx]] if
pred_labels_classes[idx] < len(class_names_map) else
str(pred_labels_classes[idx])

        title_color = 'green' if true_labels[idx] ==
pred_labels_classes[idx] else 'red'
        plt.title(f"Спр: {true_char}\nПред: {pred_char}",
color=title_color, fontsize=10)
        plt.axis('off')
    plt.tight_layout()
    plt.show()

# Передбачення для візуалізації
plot_predictions(x_test, y_test, y_pred_classes, class_names,
num_samples=25)

# Приклад, як знайти та показати деякі помилкові передбачення
misclassified_indices = np.where(y_pred_classes != y_test)[0]
```

```
num_misclassified_to_show = 10
if len(misclassified_indices) > 0:
    print(f"\nПриклади помилкових класифікацій (перші
{num_misclassified_to_show}):")
    plt.figure(figsize=(12, 5))
    for i, img_index in
enumerate(misclassified_indices[:num_misclassified_to_show]):
        plt.subplot(2, 5, i + 1)
        plt.imshow(x_test[img_index].reshape(IMG_HEIGHT, IMG_WIDTH),
cmap='gray')
        true_char_m = class_names[y_test[img_index]] if y_test[img_index] <
len(class_names) else str(y_test[img_index])
        pred_char_m = class_names[y_pred_classes[img_index]] if
y_pred_classes[img_index] < len(class_names) else
str(y_pred_classes[img_index])
        plt.title(f"Спр: {true_char_m}\nПред: {pred_char_m}", color='red',
fontsize=10)
        plt.axis('off')
    plt.tight_layout()
    plt.show()
else:
    print("Не знайдено помилкових класифікацій для відображення.")

print("\nЗавершення роботи скрипта.")
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

(ПРЕЗЕНТАЦІЯ)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Комп'ютерної інженерії

НЕЙРОМЕРЕЖЕВІ МОДЕЛІ ДЛЯ РОЗПІЗНАВАННЯ РУКОПИСНОГО ТЕКСТУ

Виконав:
Хоменко Дмитро Володимирович
Науковий керівник:
Борова Юлія Олександрівна

Київ 2025

Мета, об'єкт, предмет, наукова новизна

- **Мета роботи:**
  Розробити та протестувати нейромережеву систему для **розпізнавання рукописного тексту** з використанням сучасних архітектур глибокого навчання. Метою є не лише аналіз теоретичних підходів, а й **практична реалізація** моделі з оцінкою її ефективності.
- **Об'єкт дослідження:**
  Нейромережеві моделі, що застосовуються для задач оптичного розпізнавання тексту (OCR).
- **Предмет дослідження:**
  Методи та архітектури глибоких нейронних мереж (CNN, RNN, CRNN, Transformer), які дозволяють здійснювати **розпізнавання рукописного тексту** з високою точністю.
- **Наукова новизна:**
  Моделювання та реалізація повноцінної нейромережевої OCR-системи, здатної розпізнавати **рукописний текст змінного формату** без попередньої сегментації, на основі архітектури **CRNN з CTC-loss**. Здійснено **порівняння точності** між різними підходами та обґрунтовано доцільність використання певних архітектур для практичних задач.

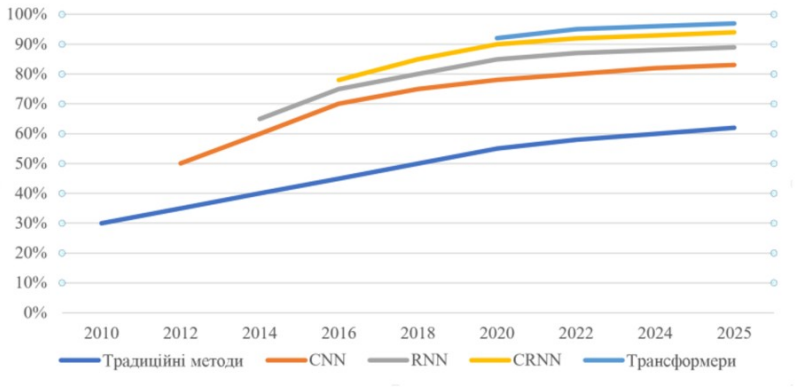
Актуальність

- Робота є актуальною, оскільки велика кількість важливої інформації — особисті записи, архівні документи, медичні картки, шкільні зошити — усе ще існує у **рукописному вигляді**. Автоматичне розпізнавання такого тексту дозволяє **оцифрувати дані**, зробити їх доступними для пошуку, аналізу та зберігання.
- Класичні OCR-системи ефективні лише для друкованого тексту і **погано справляються з рукописами** через індивідуальність почерку, перекриття символів, шуми.
  Саме тому нейромережеві моделі, здатні адаптуватися до різноманітних стилів письма, стають **ключовим інструментом** у сфері розпізнавання тексту.
- Застосування таких моделей сприяє:
- автоматизації введення рукописних даних;
- підвищенню точності розпізнавання;
- зменшенню людських помилок;
- і прискоренню роботи з великими обсягами текстової інформації.
- Штучні нейронні мережі вже довели свою ефективність у комп'ютерному зорі, і сьогодні є одним з найпотужніших інструментів для **розпізнавання складного тексту**, зокрема рукописного.

Проблеми рукописного тексту

- Індивідуальність почерку
- Перекриття символів і шум
- Мультимовність і варіативність

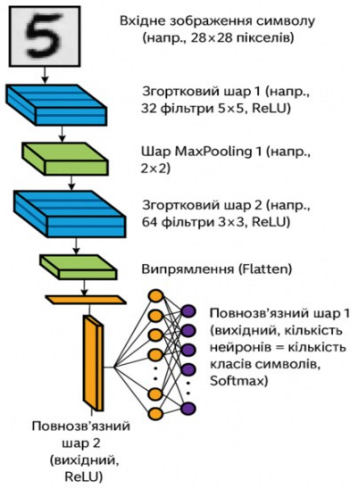
Продуктивність OCR-архітектур (2010–2025)



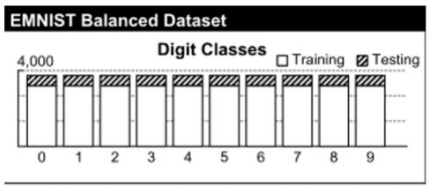
Порівняння класичні методи з сучасними нейромережевими для задач OCR

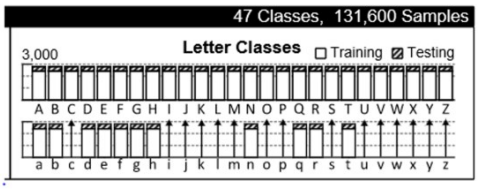
Архітектура	Переваги	Недоліки
CNN	- Ефективне виділення ієрархічних візуальних ознак. - Інваріантність до локальних зсувів та невеликих деформацій. Відносно прості в реалізації для класифікації.	- Обмежені в моделюванні послідовностей та глобального контексту. Потребують сегментації для розпізнавання тексту.
RNN (LSTM/GRU)	- Здатність моделювати послідовні дані та залежності між елементами. LSTM/GRU вирішують проблему згасання градієнтів для довших послідовностей.	- Послідовна обробка обмежує паралелізацію та швидкість. Не обробляють візуальну інформацію безпосередньо.
CRNN	- End-to-end навчання для розпізнавання тексту без попередньої сегментації. Висока точність для багатьох типів тексту.	- Більш складна архітектура порівняно з окремими CNN або RNN. Може бути чутлива до сильних геометричних викривлень тексту.
Transformer	- Потужний механізм уваги для моделювання глобальних залежностей як у візуальних даних, так і в текстових послідовностях. Гнучкість: застосовується для виявлення та розпізнавання.	- Високі вимоги до обсягу навчальних даних. Інтерпретація механізмів уваги може бути складною.

Візуальна архітектура CNN для класифікації зображень

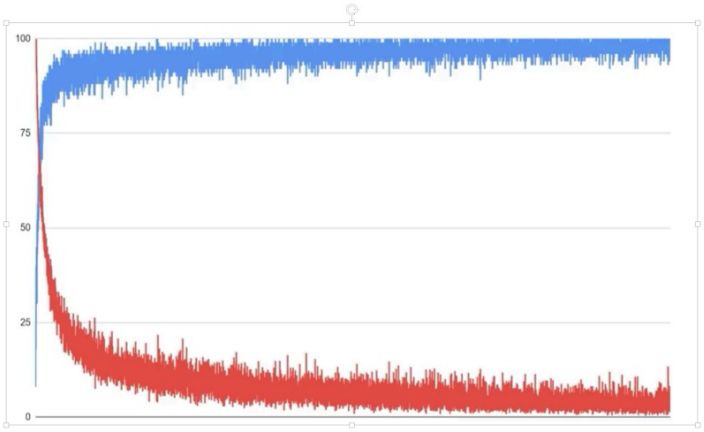


Приклади зображень з датасету EMNIST (Balanced)

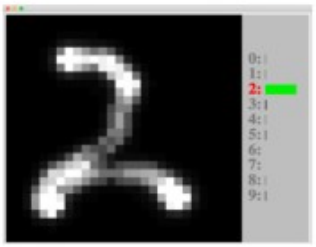


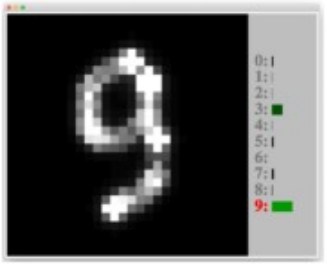


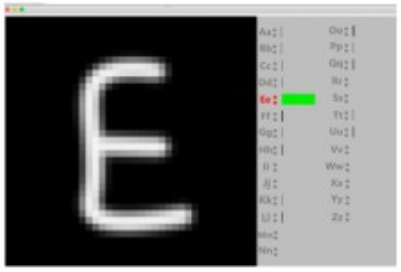
Графік зміни функції втрат протягом епох навчання

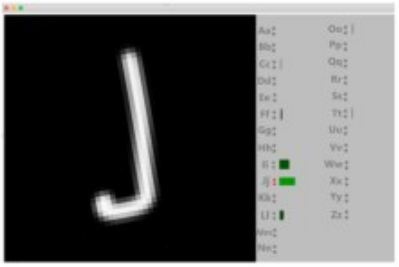


Приклади розпізнавання символів моделлю на тестових даних









Висновки

- У бакалаврській роботі проведено моделювання системи розпізнавання рукописного тексту на основі штучних нейронних мереж.
  Оцінено рівень розвитку сучасних підходів до оптичного розпізнавання символів (OCR) із використанням глибокого навчання.
- Основними результатами роботи є:
- Досліджено структуру багатошарових нейронних мереж, зокрема архітектури CNN, RNN, CRNN та Transformer.
- Проведено порівняльний аналіз їх точності, ресурсомісткості та придатності до обробки рукописного тексту.
- Описано ключові властивості штучних нейронних мереж, такі як адаптивність, робота в реальному часі, самоорганізація та відмовостійкість.
- Реалізовано та протестовано модель CNN, що показала точність розпізнавання понад 90% на відкритому датасеті EMNIST "Balanced":.
- Встановлено, що сучасні нейромережеві моделі значно перевершують класичні OCR-системи та відкривають широкі можливості для автоматизації обробки рукописного тексту в архівах, освіті, медицині та побуті.