Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ДОСЛІДЖЕННЯ МЕТОДІВ РОЗПІЗНАВАННЯ АРАБСЬКИХ СИМВОЛІВ ЗА ДОПОМОГОЮ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ
(тема)

Виконав:
студент 2 курсу, групи ІНФм-22-2
Помазан В.В.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Творошенко І.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)

Кафедра Інформатики
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Помазану Віктору Вікторовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів розпізнавання арабських символів за допомогою згорткових нейронних мереж

затверджена наказом по університету від 3 листопада 2023 року № 1280Ст

2. Термін подання студентом роботи до екзаменаційної комісії 23 грудня 2023 р.

3. Вихідні дані до роботи математичні моделі згорткових нейронних мереж, теоретичні відомості про методи розробки згорткових нейронних мереж.

4. Перелік питань, що потрібно опрацювати в роботі

1. Огляд методів розпізнавання арабських символів за допомогою згорткових нейронних мереж.

2. Характеристика кожної окремої нейронної мережі, яка була використана в роботі.

3. Реалізація конкретного застосунку для обробки арабських символів за допомогою згорткових нейронних мереж.

4. Аналіз отриманих результатів, порівняння та визначення перспектив подальших досліджень.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) актуальність проблеми, об'єкт та мета дослідження, постановка задачі, аналіз предметної області, вихідні дані для дослідження, етапи реалізації поставленої задачі, аналіз результатів.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	03.11.2023	
2	Аналіз завдання, підбір літератури	03.11.2023	
3	Аналіз літератури з досліджуваної проблеми	03.11.23-06.11.23	
4	Огляд матеріалів для створення датасету	06.11.23-07.11.23	
5	Формування датасету з анотаціями	07.11.23-09.11.23	
6	Програмна реалізація	09.11.23-14.11.23	
7	Оформлення пояснювальної записки	14.11.23-17.11.23	
8	Перевірка на плагіат	17.11.23-23.11.23	
9	Рецензування	05.12.2023	
10	Підготовка презентації та доповіді	09.12.2023	
11	Занесення роботи в електронний архів	23.12.2023	
12	Попередній захист кваліфікаційної роботи	02.01.2024	

Дата видачі завдання 3 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Творошенко І.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 79 с., 1 табл., 27 рис., 44 джерела.

РОЗПІЗНАВАННЯ СИМВОЛІВ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, ОБРОБКА ЗОБРАЖЕНЬ, АНАЛІЗ ДАНИХ, ЕФЕКТИВНІСТЬ РОЗПІЗНАВАННЯ, АРАБСЬКІ СИМВОЛИ.

Об’єктом дослідження є набір арабських символів для розпізнавання.

Метою дослідження є вивчення особливостей різних методів згорткових нейронних мереж на прикладі задачі розпізнавання рукописних арабських символів.

Для досягнення цієї мети використовуються методи машинного навчання, зокрема згорткові нейронні мережі, а також методи обробки зображень і аналізу даних.

У результаті дослідження здійснена програмна методі розпізнавання арабських символів за допомогою згорткових нейронних мереж.

CHARACTER RECOGNITION, CONVOLUTIONAL NEURAL NETWORKS, MACHINE LEARNING, IMAGE PROCESSING, DATA ANALYSIS, RECOGNITION PERFORMANCE, ARABIC CHARACTERS.

The object of research is a set of Arabic characters for recognition.

The purpose of the research is to study the features of different methods of convolutional neural networks on the example of the task of recognizing handwritten Arabic characters.

To achieve this goal, machine learning methods, particularly convolutional neural networks, as well as image processing and data analysis methods, are employed.

As a result of the research, a software method for recognizing Arabic characters using convolutional neural networks has been implemented.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ОАЕ – Об’єднані Арабські Емірати

OCR – Optical Character Recognition (оптичне розпізнавання символів)

CNN – Convolutional Neural Network (згорткова нейронна мережа)

LSTM – Long Short-Term Memory (довга короткострокова пам’ять)

ОРС – оптичне розпізнавання символів

ЗНМ – згорткова нейронна мережа

ДКП – довга короткострокова пам’ять

ШНМ – штучна нейронна мережа

ВСТУП

На Землі існує понад сім тисяч різних мов. Кожна етнічна група має свою власну мову, існує близько 250 різних мовних сімей, і кожна з них є унікальною. Мова – це не лише потужний інструмент спілкування та засіб мислення, але й відображення духу народу, його історії та невід'ємною частиною культури та існування народу [1].

Арабська мова є однією з найпоширеніших у світі та набирає все більшої популярності. Її використовує близько 350 мільйонів людей у 23 країнах світу, де ця мова є офіційною. До цих країн входять, зокрема, Єгипет, Алжир, Ірак, Судан, Саудівська Аравія, ОАЕ, Бахрейн, Палестина та багато інших. Завдання розпізнавання рукописних арабських символів стоїть особливо актуально в сучасному світі. Дисципліни, такі як археологія, палеонтологія та лінгвістика, розвиваються дуже швидко і часто вимагають автоматизації [2].

З урахуванням поширеності арабської мови вчені з усього світу досліджують методи автоматизації процесу розпізнавання арабських символів. Старовинні рукописи, обробка банківських даних, графологічні завдання – це лише декілька прикладів завдань, які потребують постійного вдосконалення процесу розпізнавання символів [3, 4].

Для вирішення цих завдань згорткові нейронні мережі виявили себе найкраще. Перша згорткова нейронна мережа, відома як LeNet, була розроблена саме для завдання розпізнавання рукописних цифр на зображеннях розміром 32×32 пікселя. З тих пір згорткові нейронні мережі пройшли довгий шлях розвитку і зараз використовуються в криптографії, медицині, лінгвістиці, математиці, філології та інших галузях. Нові моделі нейронних мереж постійно розробляються і вдосконалюються. У цій роботі буде проведено дослідження ефективності існуючих моделей згорткових нейронних мереж на прикладі завдання обробки рукописних арабських символів.

Актуальність дослідження полягає в розвитку та вдосконаленні методів розпізнавання арабських символів за допомогою згорткових нейронних мереж. З урахуванням поширеності арабської мови та зростаючої потреби в автоматизації обробки її рукописних символів, це дослідження стає важливим в контексті розвитку і застосування штучного інтелекту. Підходи, які використовуються для розпізнавання арабських символів, можуть знайти застосування в різних галузях, включаючи обробку документів, криптографію, медицину, та інші, сприяючи розвитку інновацій у вивченні нейронних мереж та їх використанні у наукових дослідженнях.

1 АНАЛІЗ ІСНУЮЧИХ ЗАСТОСУНКІВ ДЛЯ РОЗПІЗНАВАННЯ СИМВОЛІВ ЗА ДОПОМОГОЮ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

1.1 Класифікація та аналіз існуючих методів для розпізнавання символів за допомогою згорткових нейронних мереж

Розпізнавання рукописного тексту було однією з найцікавіших і найскладніших областей досліджень у галузі обробки зображень і розпізнавання образів в останні роки. Це робить величезний внесок у вдосконалення процесів автоматизації та покращує взаємодію між людиною та машиною в багатьох програмах. Загалом розпізнавання рукописного тексту поділяється на два типи: офлайн та онлайн методи розпізнавання рукописного тексту. Було показано, що онлайнові методи перевершують їхні офлайнові лічильники у розпізнаванні рукописних символів завдяки часовій інформації, доступній першим. Проте в офлайнових системах, показаних на рисунку 1.1, отримано порівняно високі рівні точності розпізнавання [5].

З урахуванням важливості вирішення задачі розпізнавання рукописних символів було розроблено багато методів та класифікацій вирішення цієї задачі.

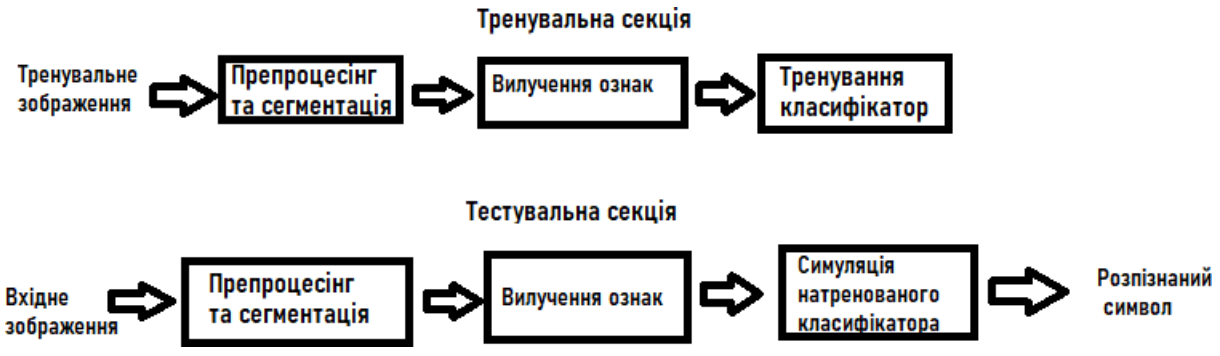

Рисунок 1.1 – Універсальна модель розпізнавання символів

Серед найпопулярніших та найефективніших можна виділити наступні: ОРС – оптичне розпізнавання символів, ЗНН – згорткові нейронні мережі, ДКП – довга короткострокова пам’ять.

1.1.1 Оптичне розпізнавання символів

Оптичне розпізнавання символів дозволяє розпізнавати текстові символи з цифрових зображень, відсканованих документів і відеопотоків. Програмне забезпечення OCR аналізує зображення тексту та перетворює його на машинно-закодований текст, який потім можна редагувати, шукати та індексувати. OCR можна використовувати для широкого діапазону програм, включаючи сканування документів, автоматичне індексування та обробку форм. Крім того, програмне забезпечення OCR можна інтегрувати в різні системи, такі як системи керування документами, системи робочих процесів і мобільні програми. Існують певні труднощі з системами OCR, наприклад стиль написання, розмір тексту та якість документа (написаного від руки, друкованого чи відсканованого) [6].

Оптичне розпізнавання знаків (OCR) відноситься до галузі науки комп’ютерів, яка займається читанням тексту з паперу і перетворенням зображень у форму, з якою комп’ютер може взаємодіяти (наприклад, в ASCII-коди). Система OCR дозволяє брати книгу або статтю з журналу, безпосередньо вводити її в електронний комп’ютерний файл і редагувати файл за допомогою текстового процесора.

Усі системи OCR включають оптичний сканер для читання тексту та програмне забезпечення для аналізу зображень. Більшість систем OCR використовують комбінацію апаратного (спеціалізовані плати) та програмного забезпечення для розпізнавання символів, хоча деякі недорогі системи використовують виключно програмне забезпечення.

Модифіковані системи OCR можуть читати текст в різноманітних шрифтах, але вони все ще мають труднощі з рукописним текстом. OCR представляє механічний або електронний переклад сканованих зображень рукописного, друкованого або машинописного тексту в текст, закодований машинним способом. У широкому спектрі застосувань OCR використовується для перетворення книг та документів у електронні файли, для комп'ютеризації системи обліку документів в офісі або для публікації тексту на вебсайті.

Перші OCR, які були винайдені Джейкобом Рабіновим з початку 1940 року, були примітивними механічними пристроями з досить високою кількістю помилок. З постійним збільшенням обсягу нового оптичного матеріалу з'явилася потреба у радикальному альтернативному рішенні, щоб подолати цю проблему. Для подолання цієї проблеми, Template Matching є одним із рішень, яке було придатне для впровадження у розпізнаванні оптичних символів завдяки простому алгоритму, який використовувався.

Оптичне Розпізнавання Знаків за допомогою Template Matching – це прототип системи, який дозволяє розпізнавати символи або алфавіт шляхом порівняння двох зображень алфавіту [7]. Метою цього прототипу системи є створення прототипу для системи оптичного розпізнавання символів та реалізація алгоритму Template Matching при створенні прототипу системи.

OCR можна впроваджувати як у режимі офлайн, так і онлайн. У офлайн-розпізнаванні письмо зазвичай захоплюється оптично за допомогою сканера, і закінчене письмо доступне у вигляді зображення. Однак у системі онлайн двовимірні координати послідовних точок представлені як функція від часу. Також доступні порядки рухів письмового знака. Виявлено, що онлайн-методи [8] виявляються переважними над офлайн-аналогами у розпізнаванні рукописних знаків завдяки наявності тимчасової інформації.

Попередня обробка покращує якість сирого зображення та виділяє важливі дані. Вона також відома як обробка на рівні пікселів або обробка на низькому рівні, яка застосовується до детального зображення для подальшого аналізу.

У рамках попередньої обробки виконується корекція нахилу зображення для виправлення текстових ліній, порогове перетворення зображення в відтінках сірого або кольорового зображення в бінарне зображення, а також зменшення шуму для зниження зайвих даних (рис. 1.2).

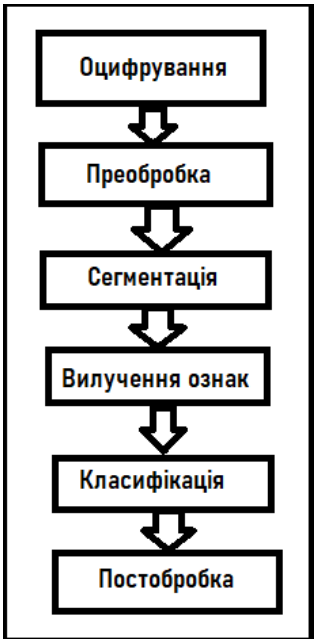

Рисунок 1.2 – Основні етапи оптичного розпізнавання символів

Скановані документи можуть бути забруднені пилом, плямами, точками або лініями, які вважаються шумом, що впливає на результати розпізнавання в значній мірі. Для того, щоб зробити його придатним для подальшої обробки, скановане зображення статті має бути очищене від будь-якого наявного шуму. Методи покращення зображення застосовуються для покращення зображення, яке розглядається машинами або людьми. Для зниження шуму використовуються згладжування та нелінійні операції, такі як морфологічні операції [8].

Корекція нахилу використовується для виправлення текстових ліній на зображеннях сканованих документів, якщо вони не були горизонтально вирівняні під час процесу сканування. Корекція нахилу може застосовуватися на рівні документа або рівні ліній. У попередньому етапі глобальної корекції нахилу застосовується на рівні документа [8].

Фаза сегментації на попередньому етапі означає, що зображення в структурі послідовності алфавітних символів розбивається на під зображенням окремого символу. Вхідне зображення, яке було попередньо оброблене, розділяється на окремі алфавітні символи, присвоюючи кожному символу номер за допомогою процедури маркування. Це маркування надає інформацію про кількість символів на зображенні. Кожен окремий символ однорідно переформатовується в розмір 90×60 пікселів для стадії категоризації та розпізнавання.

Після того як з документованого зображення вилучені рядки, перед вилученням ознак виконується нормалізація. Головною метою фази нормалізації є усунення відмінностей, які інакше ускладнили б категоризацію та знизили швидкість розпізнавання схожого символу або слова з різних авторів. Найбільш загальним джерелом змінності у зображеннях рукописних алфавітних символів є нахил та обсяг тексту [8].

Ознаки відіграють важливу роль у системах розпізнавання рукописного письма. Їх головна мета – підвищення швидкості розпізнавання шляхом ефективного представлення даних. Вилучення ознак пов’язане з видобуванням передусім необхідної інформації з пікселів зображення в залежності від складності, яку потрібно вирішити і використаної інформації.

У системах розпізнавання рукописних алфавітних символів використовуються два основних підходи до вилучення ознак: голістичний і аналітичний. У розпізнаванні кожне слово розглядається як клас і розпізнається як ціле слово. З іншого боку, аналітичний підхід до розпізнавання ґрунтується на розпізнаванні символу без сегментації.

1.1.2 Згорткова нейронна мережа

Згорткова нейронна мережа або Convolutional neural network (CNN) є однією з найважливіших мереж у галузі глибокого навчання.

Оскільки CNN здійснила вражаючі досягнення в багатьох областях, включаючи, але не обмежуючись, комп'ютерне зорове сприйняття та обробку природної мови, вона привернула значну увагу як з боку індустрії, так і академії протягом останніх кількох років [9].

Згорткова нейронна мережа (ЗНН) досягла блискучих результатів і стала однією з найвідоміших нейронних мереж у галузі глибокого навчання. Комп'ютерне зорове сприйняття, засноване на CNN, дозволило людям здійснювати те, що раніше вважалося неможливим протягом останніх століть, таке як розпізнавання обличчя, автономні автомобілі, самообслуговування в супермаркетах та інтелектуальні медичні процедури. Щоб краще зрозуміти сучасні CNN та зробити їх краще служити людству, у цій статті було надано огляд згорток, вводимо класичні моделі та застосування, і пропонуємо деякі перспективи для CNN [9].

Глибоке навчання [10] відрізняється від традиційних систем машинного навчання, оскільки воно автоматично витягує ознаки з сирої інформації через різні рівні представлення, починаючи з низькорівневих ознак і просуваючись до вищих і абстрактних представлень. Це дозволяє глибокому навчанню збільшити свою потужність навчання, фіксуючи значущі патерни та приглушуючи незначні зміни вхідних даних, разом з експоненційною перевагою у представленні складних нелінійних функцій великих обсягів даних, що накопичуються в прихованих шарах глибокої мережі [11]. Кілька технік, використаних у глибокому навчанні, включають згорткові, рекурентні та глибокі нейронні мережі [12], і одним із найбільш використовуваних методів є згорткова нейронна мережа.

Згорткова нейронна мережа [13] вперше представлена у 1960-х роках і продемонструвала вражаючі результати у комп'ютерному зорі [14], ставши

найпрестижнішою нейронною мережею [15] в галузі глибокого навчання. Вона успішно застосовується для розв'язання складних візуальних задач із високими обчислювальними вимогами [16], переважно фокусуючись на класифікації зображень [17–20], сегментації, виявленні об'єктів, обробці відео, обробці природної мови та розпізнаванні мови [18]. Наприклад, згорткову нейронну мережу застосовано для аналізу відео в дослідженні Shri [19] та аналізу зображень у дослідженні Roncancio [20].

Згорткові нейронні мережі – це мережі прямого зв'язку, де потік інформації відбувається лише в одному напрямку, від входів до виходів. Так само, як штучні нейронні мережі (ШНМ), CNN мають біологічне походження.

Архітектури CNN мають декілька варіацій; вони складаються зі згорткових та пулінгових шарів, які групуються в модулі [9–21].

Універсальна модель згорткових нейронних мереж зображена на рисунку 1.3.

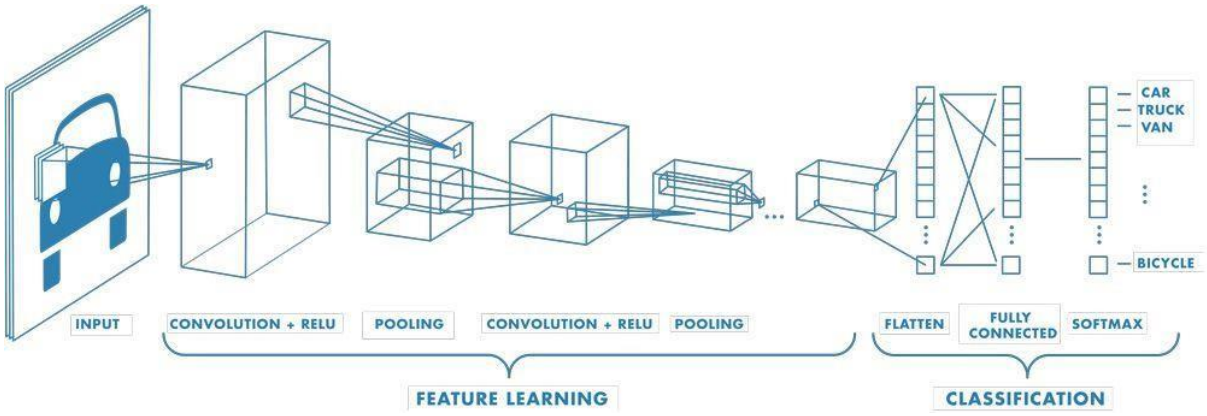

Рисунок 1.3 – Універсальна модель згорткових нейронних мереж [22]

Згортковий шар використовує фільтр-ядро для обчислення згортки вхідних зображень, виділяючи основні особливості. Розмір ядра фільтра має такий самий розмір, але менше постійне значення параметра порівняно зі вхідним зображенням [20].

Наприклад, припустима довжина фільтра-ядра для 2D зображення розміром 35×35×35 дорівнює $f \times f \times 2$, де f = 3, 5, 7 і так далі. Однак розмір

фільтра повинен бути менше розміру вхідного зображення. Маска фільтра ковзає по вхідному зображенню крок за кроком та оцінює добуток між вагою ядра фільтра та значенням пікселя вхідного зображення. Цей процес призводить до отримання 2D активаційної карти.

Згодом ЗНМ навчиться розпізнавати візуальні особливості зображення. Загальне рівняння згорткового шару може бути виражене (1.1). Рисунок 1.4 показує просту ілюстрацію обчислювального процесу в ЗНМ, що призводить до активаційної карти.

$$\text{Activation map} = \text{Input} * \text{Filter} = \\ = \sum_{y=0}^{columns}\left(\sum_{x=0}^{rows} Input(x - p, y - q)Filter\ (x, y)\right). \tag{1.1}$$

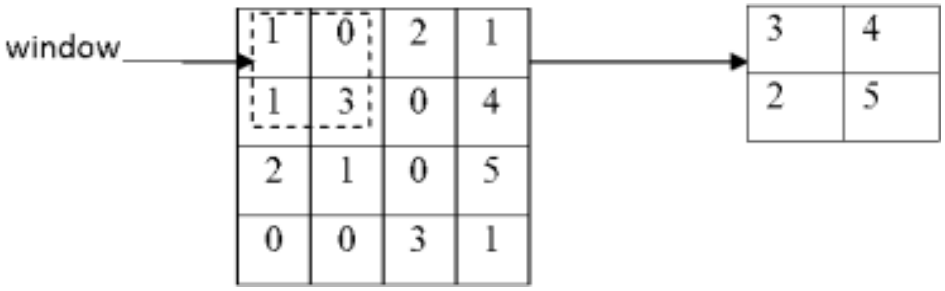

Рисунок 1.4 – Операція пулінгу виконується для вікна розміром 2×2 [6]

Згортковий шар визначається: розміром ядра, довжиною кроку та додатковим заповненням [21]. Розмір ядра – це розмір фільтра-ядра або ковзаючого ядра [22]. Довжина кроку – це кількість ядер, які ковзають, перш ніж робити точки добутку та створювати вихідні пікселі. Додаткове заповнення – це розмір нульового кадру, що оточує вхідну карту ознак.

Точне розташування особливості стає менш важливим після того, як воно було виявлено [16].

Тому за згортковим шаром слідує шар пулінгу [13]. Основною перевагою використання техніки пулінгу є значне зменшення кількості

навчальних параметрів та введення незалежності від зсуву [12–21]. Для виконання операції пулінгу вибирається вікно, а елементи вводу, що лежать у цьому вікні, проходять через функцію пулінгу, як показано на рисунку 1.4.

Функція пулінгу генерує інший вектор виведення. Існують декілька методів пулінгу, таких як середнє пулінг і максимальний пулінг, з яких найбільш поширеним є максимальний пулінг, який дуже суттєво зменшує розмір карті [17]. Під час обчислення помилок помилка не передається до переможної одиниці, оскільки вона не бере участь у прямому потоці.

Повністю підключений рівень подібний до повністю підключеної мережі в звичайних моделях. Вихід першого фаза (включає згортку та повторюване об’єднання) подається на повністю зв’язаний шар, а скалярний добуток ваговий вектор і вхідний вектор обчислюються для отримання кінцевого результату. Градієнтний спуск, також відомий як пакетний режим навчання або офлайн-алгоритм, зменшує функцію витрат шляхом оцінки вартості всього навчання набір даних і оновлює параметри лише після однієї епохи, де епоха відповідає обходу всього набір даних. Це дає глобальні мінімуми, але якщо розмір навчального набору даних великий, час, необхідний для навчання мережі істотно збільшується. Цей підхід зменшення функції витрат був замінений стохастичним градієнтним спуском.

1.1.3 Довга короткострокова пам’ять

Рекурентні або дуже глибокі нейронні мережі складно навчати, оскільки вони часто стикаються з проблемою зникнення / експлодуючого градієнту. Щоб подолати цей недолік при вивченні довгострокових залежностей, була введена архітектура LSTM. Здатність до навчання LSTM вплинула на декілька галузей як з практичної, так і з теоретичної точки зору, що зробило її передовою моделлю. Це призвело до того, що Google використовував цю модель для розпізнавання мови та покращення машинного перекладу на

Google Translate. Amazon використовує цю модель для покращення функцій Alexa, а Facebook використовує її для понад 4 мільярдів перекладів на основі LSTM щодня на 2017 рік [23].

Модель LSTM [23] є потужною рекурентною нейронною системою, спеціально розробленою для подолання проблеми зникнення градієнту, які зазвичай виникають при вивченні довгострокових залежностей, навіть коли мінімальні часові затримки дуже великі [23]. Фактично такі клітини є рекурентними мережами самі по собі, з цікавою архітектурою у тому сенсі, що CEC розширюється додатковими функціями, а саме вхідним та вихідним воротами, утворюючи клітину з пам'яттю. Саморекурентні зв'язки вказують на зворотний зв'язок з затримкою на один часовий крок.

1.2 Особливості арабських символів

Задача розпізнавання арабських символів – не є тривіальною, в першу чергу через те, що розрізняти арабські символи у відриві від нейронної мережі є досить складною задачею для людини, що не є носієм мови. Така складність проявляється, через деякі особливості арабських символів, що дуже різнять їх від звичних західній людині латиниці та кирилиці.

Серед особливостей написання арабських символів є то, що вони написані курсивом. Деякі символи мають різні форми написання, що, потенційно, означає ускладнення навчання для нейронної мережі за рахунок збільшення варіацій форм символів [4].

Також, в різних частинах слова та речення, арабські символи можуть мати різні форми написання, що також ускладнює процес розпізнавання [4].

Точки в арабських символах є досить серйозною проблемою, для задачі розпізнавання арабських символів [4]. Наприклад, розглянемо символи ت [та] і ث [са], вони означають абсолютно різні звуки, вимова в них також абсолютно різна, єдине, що їх відрізняє на письмі – символ ث має на одну точку вгорі

більше, ніж ت, це може ускладнити процес оброблення та розпізнавання символів і нейронна мережа, може видавати некоректний результат розпізнавання цих двох символів. Це лише один приклад, але кожен з символів, має варіації написання, і це є викликом для нейронних мереж, що розпізнають рукописні арабські символи.

1.3 Аналіз літературних джерел щодо апробації результатів розпізнавання символів за допомогою згорткових нейронних мереж

У джерелі [1] описані загальні характеристики світових мов і мовних груп, включаючи арабську мову. Стаття надає фундаментальні знання про структуру та різноманіття світових мов, а також розглядає арабську мову як частину цього різноманіття.

Стаття [2] описує загальні характеристики арабської мови, її історію, та перспективи розвитку. Автори досліджують еволюцію арабської лінгвістики і роль цієї мови в сучасному світі.

У роботі [3] описується один із способів розпізнавання рукописних арабських символів за допомогою згорткових нейронних мереж. Цей метод відкриває нові перспективи для автоматичного розпізнавання арабського письма. Цей метод є досить інноваційним і має великий потенціал у вирішенні завдань автоматичного розпізнавання арабського письма.

У статті [4] автори проводять дослідження, спрямоване на вдосконалення точності розпізнавання рукописних арабських символів. Для досягнення цієї мети вони вивчають та експериментують з різними методами та підходами.

Стаття [5] присвячена докладному опису процесу розпізнавання рукописних арабських символів. Цей процес є важливим в багатьох сферах, таких як оптичне розпізнавання символів (OCR), де текст переводиться з рукописного вигляду в електронний формат.

Робота [6] є оглядом оптичного розпізнавання символів у відношенні до арабської мови. Вона аналізує різні аспекти та техніки OCR, специфічні для арабської мови, і розглядає виклики, пов’язані з цим завданням.

Стаття [7] є дослідженням методів оптичного розпізнавання символів. Вона описує різні методи та підходи, які використовуються в оптичному розпізнаванні символів, і досліджує їх ефективність. Автори статті аналізують, як ці методи були вже застосовані та які результати були досягнуті. Це дозволяє визначити найбільш ефективні та точні підходи до оптичного розпізнавання символів.

Робота [8] містить інформацію про систему оптичного розпізнавання символів (ОРС) і її застосування. Вона докладно описує, як працює така система та які функціональні аспекти враховуються при розпізнаванні символів.

Стаття [9] є оглядом згорткових нейронних мереж та містить аналіз їх застосування та перспектив в різних галузях. Вона надає докладну інформацію про згорткові нейронні мережі та роль цих мереж у сучасних дослідженнях. В статті були описані різні застосування ЗНН у сферах від обробки зображень до медицини та інших галузях.

Робота [10] розглядає роль глибокого навчання (deep learning) в управлінні водними ресурсами в міських умовах. Вона містить критичний огляд сучасних підходів та досліджень у галузі управління водними ресурсами. Робота детально розглядає, як глибоке навчання може бути використане для оптимізації та управління водними системами у міських середовищах.

Стаття [11] є міждисциплінарним оглядом досліджень глибокого навчання та його впливу на водні ресурси. Вона допомагає зрозуміти важливість глибокого навчання для вчених, які працюють у галузі водних ресурсів, і включає в себе аналіз впливу глибокого навчання на управління та збереження водних ресурсів.

Робота [12] є докладним оглядом глибокого навчання та має за мету розглянути всі аспекти цієї галузі. Вона надає комплексний огляд розвитку та застосування глибокого навчання. Робота описує історію розвитку глибокого навчання, різні підходи та архітектури, які використовуються у сучасних дослідженнях та застосуваннях.

Робота [13] описує застосування згорткових нейронних мереж для прогнозу залишкового часу та інтервалу часу для каменів у реакторі з кам'яним ложем. Автори докладно пояснюють методику, яка використовується для прогнозування та визначення залишкового часу для цих реакторів.

Стаття [14] досліджує застосування ітеративної згорткової нейронної мережі для передбачення зараження COVID-19 на основі зображень CT-сканування легень. Вона надає методику та результати прогнозування, описуючи, як дані з CT-сканування використовуються для класифікації зараження.

Робота [15] є оглядом згорткових нейронних мереж і містить докладний аналіз їх застосувань та перспектив в галузі нейронних мереж. Автори досліджують різні застосування CNN у сферах від обробки зображень до розпізнавання образів або інших завдань, які вирішуються за допомогою нейронних мереж.

Робота [16] розглядає класифікацію пацієнтів з COVID-19 за допомогою ефективної глибокої нейронної мережі DenseNet, яка налаштована для досягнення високої точності. Вона описує, як модель DenseNet застосовується до аналізу даних про пацієнтів з COVID-19 та які результати були досягнуті.

У статті [17] описується використання згорткових нейронних мереж для класифікації даних електроенцефалографії. Стаття містить докладний огляд методів та результатів класифікації електроенцефалограм, де дані з електроенцефалограм аналізуються за допомогою ЗНН.

Стаття [18] є оглядом останніх архітектур глибоких згорткових нейронних мереж і надає огляд сучасних досліджень у цій області. Вона

розглядає різні архітектури ЗНН, які застосовуються в сучасних дослідженнях, та їх застосування.

Стаття [19] розглядає застосування згорткових нейронних мереж для класифікації подій на відео з натовпом. Вона докладно описує методи та алгоритми, які використовуються для класифікації подій на відео, а також результати такого аналізу.

Стаття [20] розглядає застосування згорткових нейронних мереж для класифікації зображень турбулентного полум'я, а також містить результати аналізу цього процесу. Автори досліджують, як CNN може бути використана для класифікації турбулентного полум'я на зображеннях, що може мати практичне застосування в різних галузях, включаючи аерокосмічну та енергетичну промисловість.

Стаття [21] є оглядом глибоких згорткових нейронних мереж для класифікації зображень. Вона надає докладний аналіз методів та застосувань глибокого навчання у галузі обробки зображень. Автори розглядають різні аспекти, включаючи роль глибокого навчання в класифікації зображень та техніки, які використовуються для підвищення точності класифікації.

Ресурс [22] представляє вступ до глибокого навчання з використанням середовища MATLAB.

Робота [23] проводить огляд моделі короткострокової пам'яті з довгостроковою пам'яттю (LSTM) та аналіз її застосувань у галузі машинного навчання. Робота надає докладний аналіз методів та застосувань LSTM для різних завдань, що включають в себе обробку послідовностей та прогнозування.

1.4 Постановка задачі дослідження

Актуальність даної роботи полягає у важливості систем розпізнавання арабських символів, їх широкому спектрі застосування в сучасному світі, а

також існуванні ряду пов’язаних із цим проблем та необхідність пошуку оптимальних методів та підходів до розпізнавання.

Об’єктом дослідження є набір арабських символів для розпізнавання.

Метою дослідження є вивчення особливостей різних моделей згорткових нейронних мереж на прикладі задачі розпізнавання рукописних арабських символів.

Враховуючи мету роботи, необхідно вирішити такі завдання:

– дослідити методи розпізнавання рукописних арабських символів за допомогою згорткових нейронних мереж;

– проаналізувати методи розпізнавання емоцій за арабських символів за допомогою згорткових нейронних мереж;

– обрати моделі нейронних мереж для навчання;

– написати універсальний застосунок для розпізнавання арабських символів за допомогою згорткових нейронних мереж;

– дослідити роботу кожної окремої моделі нейронних мереж, при однакових вхідних даних;

– проаналізувати отримані результати;

– визначити перспективи подальшої роботи.

2 ОСОБЛИВОСТІ РОЗРОБЛЕННЯ МЕТОДІВ РОЗПІЗНАВАННЯ АРАБСЬКИХ СИМВОЛІВ ЗА ДОПОМОГОЮ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

2.1 Опис набору даних для навчання згорткових нейронних мереж

Набір даних з арабськими рукописними символами складається з 16800 символів, написаних 60 учасниками, вік яких коливається від 19 до 40 років. Кожен учасник писав кожен символ (від «alef» до «yeh») десять разів на двох формах. Форми сканувалися з роздільною здатністю 300 точок на дюйм. Кожний блок автоматично сегментується. Базу даних розділено на два набори: навчальний набір (13440 символів, 480 зображень на клас) і тестовий набір (3360 символів, 120 зображень на клас) [5]. Приклади символів із набору даних показані на рисунку 2.1.

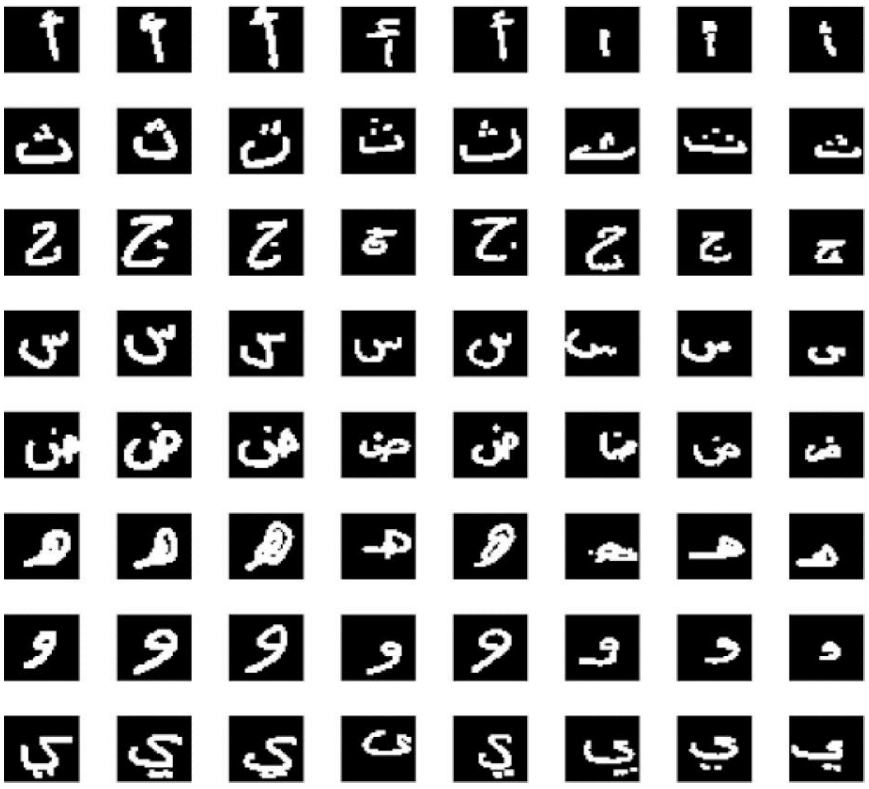

Рисунок 2.1 – Приклад символів у датасеті [5]

Завдання розпізнавання арабських символів є складним завданням, перш за все через те, що відрізняти арабські символи без допомоги нейронної мережі важко навіть для людей, які не є рідними мовцями арабської мови. Ця складність виникає через певні особливості арабських символів, які значно відрізняються від звичайних латинських та кириличних алфавітів для західних осіб. Однією з провідних рис написання арабських символів є їх написання у курсивному стилі. Приклад арабського алфавіту зображено на рисунку 2.2.

Деякі символи мають різні написання, що може ускладнити процес навчання нейронної мережі, збільшуючи різноманітність форм символів [7]. Більше того, в різних частинах слова або речення арабські символи можуть мати різні написання, подвоюючи складність процесу розпізнавання [7]. Наявність крапок у арабських символах становить значну складність у їх розпізнаванні [7]. Наприклад, розглянемо символи ت [та] і ث [са], які представляють абсолютно різні звуки та мають відмінні вимови. Єдина відмінність у їх написанні полягає в тому, що у символу ث є додаткова крапка вище символу ت. Це може ускладнити обробку і розпізнавання символів та призводити до неправильних результатів, коли нейронна мережа намагається розрізняти ці два символи. Це лише один приклад, але кожен арабський символ має різні варіації написання, що ускладнює завдання нейронних мереж у розпізнаванні рукописних арабських символів.

alif	ا	za	ز	qaf	ق
ba	ب	sin	س	kaf	ك
ta	ت	shin	ش	lam	ل
tha	ث	sad	ص	mim	م
jim	ج	dad	ض	nun	ن
ha	ح	ta	ط	ha	ه
kha	خ	dha	ظ	waw	و
dal	د	ain	ع	ya	ي
dhal	ذ	ghain	غ		
ra	ر	fa	ف		

Рисунок 2.2 – Арабський алфавіт [1]

2.2 Опис згорткових нейронних мереж, що будуть використанні в процесі розпізнавання

2.2.1 Модель LeNet

LeNet – це одна з ранніх архітектур згорткових нейронних мереж, яка була розроблена Яном Лекуном в 1998 році. Ця мережа була розроблена для розпізнавання рукописних цифр у зображеннях і була першою, що демонструвала успішність згорткових нейронних мереж у завданнях комп'ютерного зору. LeNet була базовою архітектурою для подальших робіт у галузі глибокого навчання.

Основні риси архітектури LeNet є те, що LeNet використовує згорткові шари для вилучення ознак із зображень. Ці шари використовують невеликі ядра згортки для знаходження локальних ознак у зображеннях. Пулінг (підсумовування) шари: після згорткових шарів в архітектурі LeNet використовуються шари пулінгу для зменшення розмірності зображень і зменшення кількості параметрів. Повні з'єднані (Fully Connected) шари: Після згорткових і пулінгових шарів слідують повні з'єднані шари, які використовуються для класифікації об'єктів.

Нелінійні активації: у LeNet використовуються нелінійні активації, зазвичай сигмоїдальна або гіперболічний тангенс (tanh). Функція втрат: Для завдань класифікації використовується функція втрат кросс-ентропії. Архітектура: зазвичай LeNet має архітектуру, де чергуються згорткові та пулінгові шари, після чого йдуть повні з'єднані шари.

Розмірність зображень: оригінальна LeNet була розроблена для зображень розміром 32×32 пікселів. Хоча архітектура LeNet сьогодні може здаватися простою порівняно з більш сучасними згортковими мережами, вона відіграла важливу роль у початковому розвитку глибокого навчання та розпізнаванні об'єктів на зображеннях. LeNet використовувалася для розпізнавання рукописних цифр у задачі розпізнавання символів і дала імпульс подальшому розвитку концепції згорткових нейронних мереж.

LeNet, як представник ранньої згорткової нейронної мережі, володіє основними компонентами згорткової нейронної мережі, такими як згортковий шар, шар пулінгу та повний зв’язок, закладаючи фундамент для подальшого розвитку згорткових нейронних мереж. Окрім вхідного шару, всі інші шари можуть навчати параметри.

Шар C1 – це згортковий шар із шістьма ядрами згортки розміром 5×5, а розмір карти ознак становить 28×28, що дозволяє уникнути втрати інформації вхідного зображення за межі ядра згортки.

Шар S2 – це шар підсумовування/пулінгу, який видає 6 карт ознак розміром 14×14. Кожна комірка кожної карти ознак пов’язана з 2×2 сусідами в відповідній карті ознак у C1.

Шар C3 – це згортковий шар із 16 згортковими ядрами 5×5. Вхід перших шести карт ознак C3 – це кожний послідовний піднабір з трьох карт ознак в S2, вхід наступних шести карт ознак походить від входу чотирьох послідовних піднаборів, а вхід наступних трьох карт ознак походить від чотирьох розривних піднаборів. Завершується вхід для останньої карти ознак з усіх карт ознак S2.

Шар S4 схожий на S2, має розмір 2×2 та видає 16 карт ознак розміром 5×5.

Шар C5 – це згортковий шар із 120 ядрами згортки розміром 5×5. Кожна комірка пов’язана з 5×5 сусідніми областями на всіх 16 картах ознак S4. Оскільки розмір карт ознак S4 також 5×5, то розмір виходу C5 – 1×1. Тому S4 і C5 є повністю зв’язаними. C5 позначається як згортковий шар, а не шар повного зв’язку, оскільки при збільшенні вхідних даних LeNet-5 та збереженні його структури розмір виходу буде більше 1×1, тобто це не повністю зв’язаний шар.

Шар F6 повністю зв’язаний з C5 і видає 84 карти ознак.

Загальна архітектура згорткової нейронної мережі LeNet зображена на рисунку 2.3.

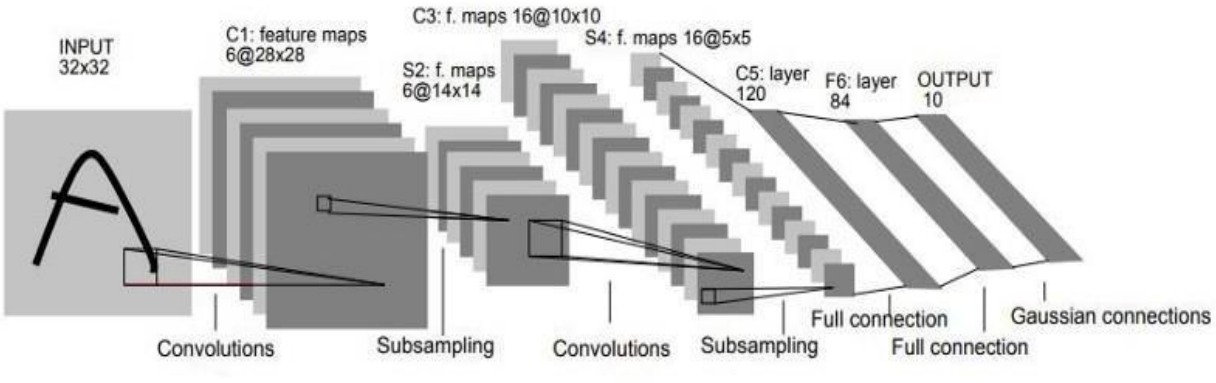

Рисунок 2.3 – Загальна архітектура згорткової нейронної мережі LeNet [6]

LeNet, розроблена Яном ЛеКуном у 1998 році, є історично значущою згортковою нейронною мережею, яка поклала початок розвитку згорткових нейронних мереж і глибокого навчання в цілому [6].

Застосування LeNet для розпізнавання рукописних арабських символів відкрило шлях для подальших досліджень та розробок у галузі комп'ютерного зору та обробки зображень.

Основні особливості LeNet, такі як використання згорткових шарів для виявлення локальних особливостей, змінні зв'язки для ефективної обробки зображень та структура з підсумовуючими шарами, залишаються актуальними у сучасних згорткових мережах.

Завдяки LeNet вдалося розв'язати задачі розпізнавання рукописних символів та розпізнавання арабських символів. Це стало важливим кроком у створенні систем розпізнавання рукописних текстів та машинного навчання, і відкрило нові можливості для автоматизації обробки даних та взаємодії з комп'ютерами. LeNet залишається ключовою точкою в історії глибокого навчання та машинного бачення, і її вплив важко переоцінити [6–10].

2.2.2 Модель AlexNet

AlexNet – це глибока нейронна мережа, створена у 2012 році Алексом Кріжевським та іншими авторами [9]. Вона була створена для класифікації фотографій у конкурсі ImageNet LSFRC-2010 і посіла перше місце [10]. Також вона може працювати на кількох графічних прискорювачах. Після цього були запропоновані більш глибокі та складні нейронні мережі, такі як видатні VGG та GoogleLeNet. Точність її офіційної моделі даних становить 57,11%, а точність у першому п’ятірці – 80,23%. Для типових алгоритмів класифікації машинного навчання це вже вражаюче.

AlexNet – це глибока згорткова мережа, розроблена для обробки великомасштабних кольорових зображен. У ній є понад 62 мільйони навчальних параметрів в сумі. Модель згорткової нейронної мережі AlexNet зображена на рисунку 2.4.

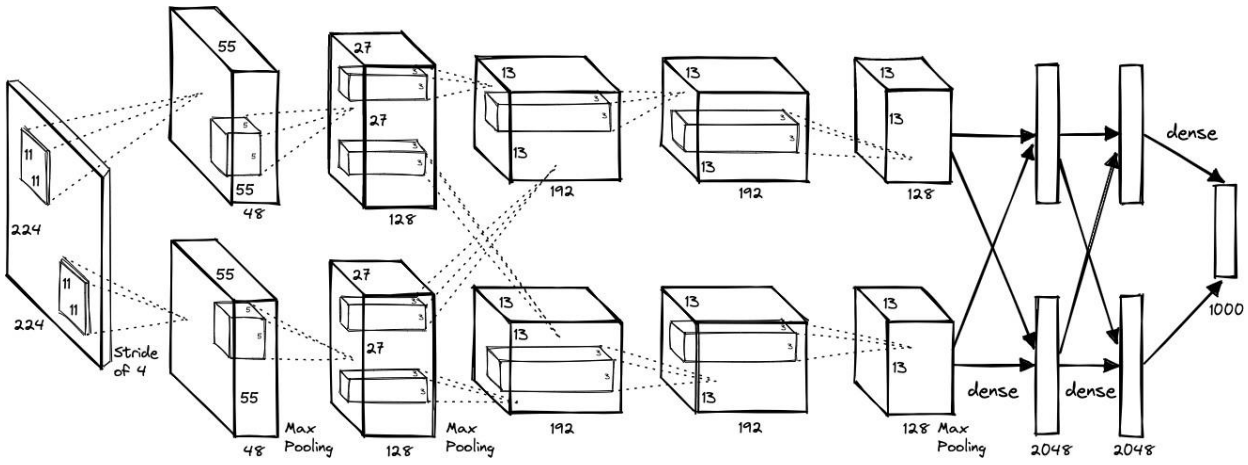

Рисунок 2.4 – Модель згорткової нейронної мережі AlexNet [9]

Архітектура AlexNet включає кілька шарів, які виконують різні функції в обробці зображень.

Перший згортковий шар (Conv1): цей шар має 96 карт ознак і використовує фільтри розміром 11×11 пікселів з кроком 4 пікселі. Він виконує

згортку вхідного зображення та витягує низькорівневі ознаки, такі як краї та текстури.

Другий шар згортки (Conv2): цей шар також містить 256 карт ознак і використовує фільтри розміром 5×5 пікселів. Він покращує вилучення ознак і робить їх абстрактнішими.

Третій згортковий шар (Conv3): складається з 384 карт ознак та використовує фільтри розміром 3×3 пікселя. Цей шар продовжує отримувати більш високорівневі ознаки зображення.

Перший шар пулінгу (Pool1): цей шар використовує max-пулінг з вікном розміром 3×3 пікселя та кроком 2 пікселя. Він зменшує розмір карт ознак та зберігає найбільш значущі ознаки.

Другий шар пулінгу (Pool2): цей шар також застосовує max-пулінг з аналогічними параметрами.

Четвертий згортковий шар (Conv4): містить 384 карти ознак та фільтри розміром 3×3 пікселя. Після цього шару слідує п’ятий шар пулінгу (Pool5).

П’ятий шар згортки (Conv5): містить 256 карт ознак і також використовує фільтри 3×3 пікселя.

Перший повнозв’язковий шар (FC6): містить 4096 нейронів і виконує класифікацію на більш абстрактних ознаках, вилучених із попередніх шарів.

Другий повнозв’язковий шар (FC7): також містить 4096 нейронів та поглиблює абстракцію ознак.

Третій повнозв’язний шар (FC8): цей шар має 1000 нейронів і є вихідним шаром, який передбачає можливість приналежності зображення до 1000 різних класів ImageNet.

Шар класифікації використовує функцію softmax для перетворення значень, отриманих на виході з FC8, у ймовірності приналежності зображення до різних класів.

Ця детальна архітектура дозволяє AlexNet отримувати та аналізувати ознаки із зображень різних рівнів абстракції, що робить його потужною моделлю для класифікації зображень. Після перемоги у змаганні ImageNet

2012 року AlexNet стала відправною точкою для розвитку глибокого навчання в комп’ютерному зорі.

AlexNet – це історична архітектура глибокої нейронної мережі, яка визначила новий етап у розвитку комп’ютерного зору та глибокого навчання. Розроблена Алексом Кріжевським, Іллею Сутськевером та Джеффрі Гінтоном, ця мережа вперше була впроваджена в 2012 році та долучилася до змагання ImageNet Large Scale Visual Recognition Challenge (ILSVRC), здобувши перше місце та вражаючі результати [9].

Архітектура AlexNet має декілька важливих особливостей.

Глибока згорткова структура: AlexNet включає в себе п’ять згорткових шарів, які дозволяють моделі автоматично вивчати важливі ознаки зображень різних рівнів абстракції. Ця структура глибоких шарів сприяє покращенню якості класифікації.

Використання функції активації ReLU: вперше в AlexNet була використана функція активації ReLU (Rectified Linear Unit), яка допомагає уникнути проблем затухання градієнтів та прискорює процес навчання [9].

Високий рівень точності: AlexNet дала вражаючий результат з точністю класифікації на рівні 86,7% для вибору правильного класу та 80,2% для п’ятої найкращої альтернативи на датасеті ImageNet. Ця висока точність встановила новий стандарт в галузі комп’ютерного зору.

Застосування на практиці: AlexNet стала важливим етапом у розвитку глибокого навчання та знаходить своє застосування в різних сферах, включаючи розпізнавання облич, автоматичну класифікацію зображень та інші задачі обробки зображень.

AlexNet є революційною архітектурою глибокої нейронної мережі, яка відкрила шлях до нового рівня точності в комп’ютерному зорі та глибокому навчанні. Її вплив на галузь став критичним у впровадженні нейромереж у різні застосування та покращенні якості автоматичної обробки зображень. AlexNet залишається ключовою моделлю для багатьох дослідників та

інженерів, що працюють у сферах комп'ютерного зору та машинного навчання.

2.2.3 Модель VGG19

VGG19 – це інша визнана архітектура глибокої нейронної мережі, яка розроблена в групі дослідників з Університету Оксфорда. Ця модель стала важливим кроком в розвитку глибокого навчання та знайшла широке використання в обробці зображень.

Основні характеристики архітектури VGG19.

Глибока структура: VGG19 містить 19 шарів, включаючи 16 згорткових та 3 повнозв'язаних шари. Ця глибока архітектура допомагає моделі вивчати високорівневі та абстрактні ознаки зображень [11–15].

Зафіксовані фільтри: усі згорткові шари використовують фільтри розміром 3×3 пікселів та шагом 1 піксель, що сприяє точнішому виділенню ознак на зображеннях.

Повторність структури: основна ідея VGG19 – це повторне використання згорткових шарів з невеликими фільтрами та пулінгових шарів для поетапного зменшення розміру зображення .

Використання функції активації ReLU: архітектура використовує функцію активації ReLU після кожного згорткового та повнозв'язаного шару, що допомагає уникнути затухання градієнтів та покращує швидкість навчання.

Зручність для трансферного навчання: модель VGG19 є досить загальною та здатною до трансферного навчання. Велика кількість параметрів дозволяє використовувати її для різних завдань обробки зображень, включаючи визначення об'єктів, класифікацію та генерацію зображень.

Модель VGG19 зображена на рисунку 2.5.

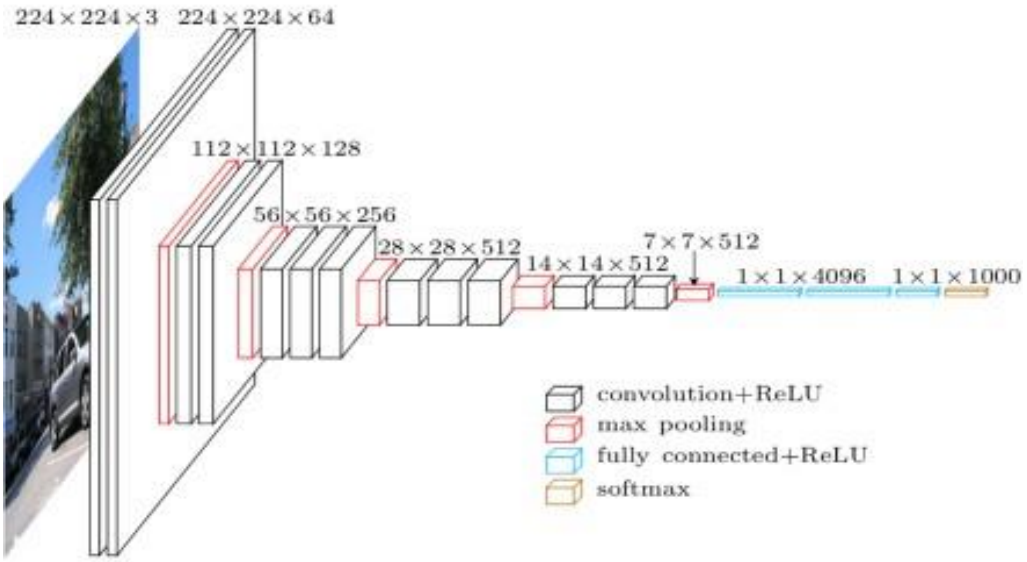

Рисунок 2.5 – Модель нейронної мережі VGG19 [15]

Висока точність класифікації: VGG19 здатна до вражаючих результатів в класифікації зображень. Модель досить надійно розпізнає об’єкти та класи на зображеннях, і це робить її добре підходящою для багатьох завдань в сфері комп’ютерного зору.

Архітектура VGG19 має глибокий структурований дизайн і включає в себе різні типи шарів. Нижче описані всі шари VGG19 без номерів, просто за їхнім типом.

VGG19 має 16 згорткових шарів, де кожен шар використовує фільтри розміром 3×3 пікселів і шагом 1 піксель. Згорткові шари використовують функцію активації ReLU для нелінійної обробки ознак.

Важливим є те, що ці згорткові шари під час навчання взаємодіють із зображеннями різного розміру завдяки використанню нульових падінь (zero-padding), що дозволяє підтримувати розмірність на вході шарів.

Після кожного парного блоку згорткових шарів слідує пулінговий шар, який використовує max-пулінг з вікном розміром 2×2 пікселів і кроком 2 пікселі. Пулінгові шари допомагають зменшити розмірність зображення та відіграють головну роль у підвищенні інваріантності до місця розташування ознак.

VGG19 включає три повнозв’язаних шари, які використовуються для класифікації зображень.

Перші два повнозв’язані шари містять 4096 нейронів кожен, а третій шар, останній, має 1000 нейронів, що відповідають 1000 класам набору даних ImageNet. Останній повнозв’язаний шар використовує функцію активації softmax для побудови ймовірнісного розподілу на класи ImageNet.

Важливо відзначити, що модель VGG19 має велику кількість параметрів, що робить її більш потужною, але при цьому вимагає більше обчислювальних ресурсів для навчання та використання. Вона була успішно використана для багатьох завдань у галузі обробки зображень, включаючи класифікацію, визначення об’єктів та стилізацію зображень, завдяки своїй здатності екстрагувати складні ознаки із зображень.

2.2.4 Модель ResNet

Архітектура ResNet (Residual Network) складається з низки блоків, в яких застосовані залишкові зв’язки. Кожен блок має різні шари, які допомагають зберігати та передавати інформацію в глибоких нейронних мережах.

Розглянемо кожен шар і як він працює.

Вхідний шар (Input Layer): цей шар приймає вхідні зображення та перетворює їх у карту ознак. Зазвичай, це перші згорткові шари, де застосовуються фільтри для виявлення низькорівневих ознак, таких як горизонтальні та вертикальні грані.

Залишковий блок (Residual Block): це ключовий компонент ResNet. В кожному залишковому блоку є два гілки: коротший (skip connection) та глибший. Вхідна карта ознак проходить через послідовність згорткових шарів, і потім коротший зв’язок додає початковий вихід до вихідної карти ознак з

цього блоку. Це дозволяє виразити різницю між поточним і очікуваним виходами блоку. Формально, вираз для залишкового блоку виглядає так:

$$f(x) = H(x) + x, \tag{2.1}$$

де $f(x)$ – вихід блоку;

$H(x)$ – перетворення, виконане внутрішніми шарами блоку;

x – початковий вихід.

Згортковий шар (Convolutional Layer): в блоках ResNet використовуються згорткові шари з фільтрами для витягнення вищерівневих ознак із карт ознак.

Функція активації ReLU: після кожного згорткового шару застосовується функція активації ReLU, щоб вводити нелінійність в мережу.

Пулінговий шар (Pooling Layer): після кількох згорткових шарів застосовується пулінговий шар для зменшення розмірності ознакових карт та підвищення інваріантності до місця розташування об’єктів на зображенні.

Глобальний середній пулінг (Global Average Pooling): замість традиційних повнозв’язаних шарів, в останньому блоку ResNet використовується глобальний середній пулінг. Це означає, що для кожної ознакової карти обчислюється середнє значення всіх її пікселів, і отримується одне значення для кожного каналу.

Повнозв’язаний шар (Fully Connected Layer): у результуючому векторі після глобального середнього пулінгу може бути декілька нейронів, і їх вихід використовується для класифікації.

Функція активації Softmax: на останньому шарі використовується функція активації softmax для перетворення виходів у ймовірності приналежності вхідного зображення до різних класів. Модель згорткової нейронної мережі ResNet зображена на рисунку 2.6.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Рисунок 2.6 – Модель згорткової нейронної мережі ResNet [6]

ResNet надає можливість навчання глибоких нейронних мереж, подолання проблеми затухання градієнту та досягнення високої точності у завданнях класифікації та обробки зображень. Використання залишкових блоків дозволяє навчати глибокі мережі, не втрачаючи інформацію про низькорівневі та високорівневі ознаки.

ResNet (Residual Network) є інноваційною та впливовою архітектурою глибоких нейронних мереж, розробленою для подолання проблеми затухання градієнту та навчання нейромереж з надзвичайною глибиною. Ця архітектура відкрила новий рівень для точності та ефективності в завданнях обробки зображень та машинного навчання.

Залишкові блоки: основним інноваційним елементом ResNet є залишкові блоки, які дозволяють нейронній мережі навчати приріст до наявного ознакового відображення. Це вирішує проблему затухання градієнту та дозволяє побудувати надзвичайно глибокі мережі.

Глибока структура: ResNet складається з великої кількості шарів, що дозволяє моделі виражати високорівневі та абстрактні ознаки, покращуючи точність класифікації та розпізнавання об’єктів.

Ефективність та надійність: ResNet стала однією з найефективніших архітектур для завдань обробки зображень, а також виявилася дуже надійною при навчанні глибоких нейромереж.

Застосування: ResNet знайшла своє застосування в різних сферах, включаючи класифікацію зображень, визначення об’єктів, сегментацію зображень та генерацію зображень. Вона також використовується для розв’язання завдань, де потрібні надзвичайно глибокі мережі.

В цілому, ResNet стала критично важливою архітектурою для глибокого навчання та машинного навчання, а її інноваційність у вирішенні проблеми затухання градієнту революціонізувала галузь обробки зображень і встановила нові стандарти для точності та глибини нейронних мереж.

2.2.5 Модель GoogleNet

GoogleNet, також відома як Inception, є архітектурою глибоких нейронних мереж, розробленою Google для завдань обробки зображень та класифікації. Ця архітектура відзначається своєю ефективністю та здатністю до обробки зображень високої роздільної здатності. Основна ідея захоплення полягає в тому, щоб виявити, як найкраще комбінувати різні типи фільтрів та згорток в одній нейронній мережі.

Основні особливості архітектури GoogleNet.

Змішані конволюції: замість використання тільки згорткових шарів і пулінгу, GoogleNet використовує змішані конволюції різних розмірів (наприклад, 1×1, 3×3, 5×5). Це дозволяє виявити різні види ознак на різних рівнях абстракції.

Inception модуль: центральним елементом GoogleNet є Inception модуль, який включає в себе низку змішаних конволюцій, згорткових шарів та пулінгу, які об’єднуються в одному місці. Це дозволяє мережі вивчати різні ознаки на різних рівнях деталізації.

Глобальний середній пулінг: замість повнозв’язаних шарів на завершальному етапі, GoogleNet використовує глобальний середній пулінг для зменшення розмірності ознакових карт і отримання одновимірного вектора перед фінальною класифікацією.

Побудова мережі: GoogleNet складається з багатьох Inception модулів, розташованих один за одним. Це дозволяє створити дуже глибокі мережі з багатьма параметрами.

Паралельні шляхи: в інцепційних модулях використовуються паралельні шляхи для обробки різних типів ознак. Це сприяє збагаченню інформацією та підвищує точність класифікації.

Використання 1×1 згорток: в архітектурі GoogleNet активно використовуються 1×1 згортки для зменшення розмірності ознакових карт і керування кількістю параметрів.

GoogleNet вражає своєю здатністю до класифікації та обробки зображень високої роздільної здатності та залишає великий вплив на галузі обробки зображень та машинного навчання. Вона стала однією з популярних архітектур для обробки зображень і використовується в різних застосуваннях, включаючи розпізнавання об’єктів та сегментацію зображень.

Основна ідея архітектури Inception базується на пошуку оптимальної локальної розрідженої структури в зображувальній згортковій мережі та спробі наблизити її доступними компонентами з високою щільністю. Важливо враховувати, що в перекладі інваріантності означає, що нашу мережу будуватимуть із згорткових будівельних блоків. Потрібно знайти оптимальну локальну структуру та повторювати її у просторі.

У дослідженні [2] запропоновано побудову на рівень-за-рівнем, в якій слід аналізувати кореляційну статистику останнього шару та групувати її на групи одиниць із високою кореляцією. Ці групи формують одиниці наступного шару та з’єднуються із одиницями попереднього шару. було припущено, що кожна одиниця з попереднього шару відповідає певній області вхідного зображення і ці одиниці групуються в банки фільтрів.

У нижніх шарах (ближче до входу) корельовані одиниці концентруються в локальних областях. Це означає, що можна мати багато груп, що сконцентровані в одній області, і їх можна було б покрити шаром зі згортками розміром 1×1 в наступному шарі, як це запропоновано в [12]. Проте можна припустити, що буде менше груп, розташованих більш розпорошено, які

можна було б покрити згортками над більшими областями, і буде зменшуватися кількість патчів, розміщених над все більшими областями.

З метою уникнення проблем з вирівнюванням патчів, поточні варіанти архітектури Inception обмежуються розмірами фільтрів 1×1, 3×3 і 5×5, проте ця рішення більше базується на зручності, ніж на необхідності. Це також означає, що запропонована архітектура є поєднанням всіх цих шарів, їх вихідні банки фільтрів об'єднуються в один вихідний вектор, який стає входом для наступного етапу. Крім того, оскільки операції пулінгу були важливими для успіху в поточних передових згорткових мережах, це свідчить про те, що додавання альтернативного паралельного шляху пулінгу в кожному такому етапі також має додатковий корисний ефект.

Згорткові нейронні мережі (CNN) є потужним інструментом в області обробки зображень та багатьох інших завдань машинного навчання. Вони революціонізували галузь завдяки своїй здатності ефективно розпізнавати патерни в зображеннях, знижуючи складність та вимоги до обробки даних.

Згорткові мережі можуть виявляти локальні ознаки та шаблони в зображеннях завдяки застосуванню фільтрів та згорток.

Здатність до ієрархічного аналізу: згорткові мережі складаються з різних шарів, що дозволяє їм аналізувати ознаки на різних рівнях абстракції. Вони можуть розпізнавати від простих ознак, таких як границі та кольори, до більш високорівневих об'єктів та концепцій.

Застосування в багатьох областях: згорткові мережі не обмежені лише обробкою зображень. Вони знайшли застосування в розпізнаванні мови, аналізі тексту, медичних дослідженнях, автономних автомобілях та інших галузях.

В цілому, згорткові нейронні мережі стали ключовим інструментом в багатьох галузях та продовжують еволюціонувати, щоб вирішувати складні завдання в галузі машинного навчання та штучного інтелекту [24–44].

3 РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ МЕТОДІВ РОЗПІЗНАВАННЯ АРАБСЬКИХ СИМВОЛІВ ЗА ДОПОМОГОЮ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

3.1 Вибір інструментальних засобів для реалізації методів розпізнавання арабських символів за допомогою згорткових нейронних мереж

Основою для реалізації та методів розпізнавання арабських символів за допомогою згорткових нейронних мереж слугувала мова програмування Python.

Python – це високорівнева мова програмування загального призначення, яка володіє численними перевагами, що роблять її ідеальним вибором для розробки застосунків по розпізнаванню образів з використанням згорткових нейронних мереж. Python славиться своєю читабельністю і простотою синтаксису. Це дозволяє розробникам легко розуміти і модифікувати код, що є критичним для розробки складних систем розпізнавання образів.

Python має одну з найбільших і активних спільнот розробників у світі. Це означає, що завжди є доступ до багатьох бібліотек, модулів і фреймворків для глибокого навчання, включаючи TensorFlow, PyTorch, Keras і багато інших.

Python пропонує різноманітні бібліотеки для обробки зображень і роботи з нейронними мережами, такі як OpenCV для обробки зображень, NumPy для наукових обчислень, і Matplotlib для візуалізації результатів. У спільноті розробників Python знаходяться вже готові нейронні мережі для різних завдань розпізнавання образів. Можна використовувати попередньо навчені моделі, такі як ResNet, Inception, VGG тощо, для своїх завдань.

Python має багато бібліотек для візуалізації даних, такі як Matplotlib, Seaborn та Plotly, що спрощують відображення результатів роботи нейронних мереж та обробки зображень. Python підтримується на багатьох операційних системах, що дає можливість розробляти аплікації на різних платформах,

включаючи Windows, macOS і Linux. Python легко інтегрується з іншими мовами програмування, що дає можливість використовувати оптимізований код на C/C++ для обчисленні важких завдань у реальному часі.

Python є ідеальним вибором для швидкоплинної розробки прототипів, що дозволяє зосередитися на алгоритмічному розв'язанні проблеми розпізнавання образів, не витрачаючи багато часу на писання оптимізованого коду. В Python існують готові бібліотеки та інструменти для навчання нейронних мереж на великих наборах даних, а також для вирішення задач комп'ютерного зору, розпізнавання облич, класифікації об'єктів тощо. Python – це розвинена мова, і вона отримує підтримку від широкої спільноти розробників і академічних установ, що забезпечує її актуальність та удосконалення.

Загалом, Python – це ідеальна мова програмування для розробки застосунків для розпізнавання образів з використанням згорткових нейронних мереж завдяки своїй простоті, багатому екосистемі, активній спільноті та іншим перевагам.

Окрім цього, одним з найважливіших інструментів для роботи зі штучним інтелектом і нейронними мережами в проєкті є бібліотека TensorFlow, яка є фундаментом для всіх вищеописаних згорткових нейронних мереж. TensorFlow – це бібліотека з відкритим вихідним кодом для машинного навчання і глибокого навчання, розроблена командою Google Brain. Вона надає набір інструментів і бібліотек для створення і навчання штучних нейронних мереж. TensorFlow створена для навчання нейронних мереж, які є ключовою технологією в глибокому навчанні. Глибоке навчання дозволяє алгоритмам виявляти складні патерни і ознаки в даних, що робить його корисним для багатьох завдань, включаючи розпізнавання зразків, обробку природної мови та багато інших. Однією з ключових концепцій TensorFlow є граф обчислень. Весь процес навчання нейронної мережі подається у вигляді спрямованого ациклічного графа. У цьому графі визначено операції (вузли), які виконуються над тензорами (багатовимірними масивами даних) і

визначають архітектуру нейронної мережі. Цей граф надає високий рівень модульності і дозволяє легко змінювати і повторно використовувати частини нейронної мережі.

TensorFlow підтримує широкий спектр завдань машинного навчання, включаючи класифікацію, регресію, кластеризацію, обробку природної мови, навчання з підсиленням, генеративні моделі та багато інших. Можна створювати різні типи нейронних мереж, включаючи згорткові, рекурентні та комбіновані моделі. TensorFlow надає високий рівень гнучкості при визначенні моделей і експериментуванні з ними. Він може використовуватися як на CPU, так і на GPU, що забезпечує високу продуктивність обчислень. TensorFlow має велику спільноту розробників і дослідників, що робить його дуже популярним і надає доступ до безлічі ресурсів, бібліотек і прикладів коду. TensorFlow може використовуватися в поєднанні з іншими популярними інструментами для машинного навчання і глибокого навчання, такими як Keras (для більш високорівневого API), TensorBoard (для візуалізації навчання) та іншими.

TensorFlow – потужний інструмент для розробки та навчання нейронних мереж, і його широкі можливості роблять його популярним у спільноті дослідників і розробників машинного навчання.

Також, у парі з TensorFlow іде Keras. Keras – це високорівневий інтерфейс для розробки та навчання нейронних мереж в мові програмування Python. Keras розроблений так, щоб бути простим та інтуїтивно зрозумілим. Він дозволяє швидко створювати складні моделі нейронних мереж з декількома рядками коду. Це робить його ідеальним для початківців та досвідчених розробників.

Keras побудований на принципі модульності. Він включає в себе велику кількість готових модулів (шарів), які можна комбінувати для створення різних архітектур нейронних мереж. Це дозволяє створювати власні моделі швидко і ефективно. Keras був розроблений як вищий рівень абстракції над бібліотекою TensorFlow, але з часом став однією з офіційних частин

TensorFlow. Тобто можна використовувати синтаксис Keras з TensorFlow як вбудовану бібліотеку. Крім інтеграції з TensorFlow, Keras також підтримує інші глибокі нейронні мережі, такі як Theano і Microsoft Cognitive Toolkit (CNTK).

Keras надає доступ до численних готових моделей нейронних мереж (наприклад, VGG16, ResNet, Inception) та навчальних наборів даних. Це допомагає вам почати роботу над вашими завданнями швидше.

Keras має інтегровану підтримку візуалізації навчання нейронних мереж. Keras стежити за процесом навчання, аналізувати метрики та графіки результатів. Keras працює як на центральних процесорах (CPU), так і на графічних процесорах (GPU). Це дозволяє прискорити навчання нейронних мереж на великих обсягах даних.

Можна використовувати моделі, навчені з використанням Keras, на різних платформах, що полегшує їх інтеграцію в різні проекти та застосунки.

Загалом, Keras є потужним і простим у використанні інструментом для розробки нейронних мереж, і він добре підходить для завдань розпізнавання образів завдяки своїй зручності та різноманітним можливостям.

В якості середовища розробки було вибрано Visual Studio Code. Visual Studio Code (VS Code) – це безкоштовна і дуже популярна інтегрована середовища розробки (IDE), розроблена компанією Microsoft. Вона отримала велику популярність серед розробників через свою легкість використання, розширюваність, і підтримку багатьох мов програмування.

VS Code підтримує велику кількість мов програмування, включаючи Python, JavaScript, Java, C++, C#, Ruby, PHP, та інші. Це робить її ідеальним інструментом для багатьох видів розробки.

VS Code має інтуїтивний інтерфейс, який легко розуміти, навіть для початківців. VS Code має велику кількість розширень, які дозволяють розширити її можливості. Це включає в себе розширення для роботи з конкретними мовами програмування, інструментами для роботи з Git, розширеннями для роботи з Docker, і багато інших.

VS Code має вбудовану підтримку Git, що дозволяє вам легко керувати версіями вашого коду та спільно працювати з репозиторіями Git.

VS Code має велику та активну спільноту розробників. VS Code підтримує Windows, macOS та Linux, що робить її доступною для розробників на різних операційних системах. Також Visual Studio Code активно оновлюється і підтримується Microsoft, що забезпечує надійність та безпеку користувачів.

3.2 Етапи програмної реалізації методів розпізнавання арабських символів за допомогою згорткових нейронних мереж

Розробка програмної реалізації методів розпізнавання арабських символів за допомогою згорткових нейронних мереж – це комплексний процес, який можна розділити на шість ключових частин, кожна з яких має важливе значення для успішної реалізації системи розпізнавання.

Перший етап полягає в зборі рукописних символів арабського алфавіту для створення набору даних. Це може включати в себе сканування або фотографування рукописних символів. Важливо збирати різноманітність стилів письма та символів.

Другий етап, а саме підготовка набору даних до навчання потребує обробки перед тим, як його можна використовувати для навчання моделей. Це включає в себе такі дії, як розділення даних на навчальні та тестові набори, збалансування класів (якщо це потрібно), нормалізацію зображень та інші попередні обробки.

Розробка моделей згорткових нейронних мереж одна з ключових частин. На цьому етапі створюються архітектури згорткових нейронних мереж, які будуть відповідальні за розпізнавання символів. Вибір правильної архітектури і гіперпараметрів є критичним для успішного навчання моделей. Наступним етапом є навчання моделей згорткових нейронних мереж. Навчання включає в

себе подачу навчальних даних в модель та налаштування ваг моделі так, щоб вона могла ефективно розпізнавати символи. Цей процес може займати чимало часу та вимагає великої кількості обчислювальних ресурсів.

Після того як модель навчена, важливо виконати постобробку результатів її роботи. Це може включати в себе визначення найкращого варіанту для розпізнання символів, виправлення помилок та інші дії для покращення точності розпізнавання.

На останньому етапі проводиться тестування розробленої системи на тестових даних для оцінки її ефективності та точності. Результати оцінки допомагають покращити та вдосконалити систему. Усі ці етапи взаємозв’язані і вимагають дбайливого та систематичного підходу для досягнення успіху у створенні програмної реалізації методів розпізнавання арабських символів за допомогою згорткових нейронних мереж. Кожна з цих частин є важливою для створення ефективної системи розпізнавання арабського письма.

3.2.1 Збір даних

Задача розпізнавання рукописних арабських символів не є тривіальною, через що, був необхідний особливий підхід до збору даних для роботи. В якості набору даних було обрано набір даних «Arabic Handwritten Characters Dataset», який містить 16800 різних варіацій арабського алфавіту написаного від руки.

«Arabic Handwritten Characters Dataset» – це цінний інструмент для розвитку і вдосконалення технологій в галузі машинного навчання, комп’ютерного зору і обробки тексту на арабській мові. Цей набір даних містить велику колекцію рукописних символів, важливих для арабського алфавіту та письма. Набір даних містить символи, написані різними людьми, що означає різноманітність стилів письма. Це важливо, оскільки почерк може сильно варіюються у залежності від автора.

Кожному символу в наборі даних призначена відповідна мітка класу. Це означає, що алгоритми машинного навчання можуть бути навчені на розпізнавання і класифікацію кожного символу, що дозволяє автоматизувати розпізнавання тексту на арабській мові. Зокрема, він корисний для навчання моделей розпізнавання рукописного тексту на арабській мові. Також його можна застосувати для покращення систем автоматичного перекладу і аналізу тексту на арабській мові. Цей набір даних сприяє розвитку технологій автоматичного розпізнавання рукописного арабського тексту.

Моделі, навчені на цих даних, можуть використовуватися для створення більш точних і ефективних систем розпізнавання та перекладу тексту на арабській мові. У цілому, «Arabic Handwritten Characters Dataset» представляє собою цінний ресурс для дослідників і інженерів, які працюють над розробкою і вдосконаленням систем обробки тексту на арабській мові і надає основу для створення більш точних і надійних застосунків, пов’язаних з арабським письмом.

3.2.2 Підготовка набору даних до навчання

Підготовка набору даних для навчання нейронної мережі – важливий етап в розробці моделей машинного навчання. Спочатку були зібрані необхідні дані, які будуть використовуватися для навчання моделі.

Дані були у форматі зображень png. Після збору даних була проведена очистка та попередню обробка даних, щоб вони були відповідні для навчання. Після очистки дані були розбиті на два набори: навчальний (для навчання моделі) і тестовий (для оцінки ефективності). Було розділено дані у співвідношенні 70 на 30, де 70% навчальний набір і 30% тестовий набір.

Оскільки кількість прикладів у кожному класі нерівномірна, була виконана процедура збалансування даних: після підготовки набору даних його було збережено у форматі, придатному для подальшого використання, а саме

HDF5. Важливо перевірити підготовлені дані на правильність та наявність помилок, перш ніж навчати модель.

Підготовка набору даних – це суттєва частина процесу розробки моделі машинного навчання і може суттєво вплинути на результати та ефективність моделі. На рисунку 3.1 зображено фрагмент коду для підготовки набору даних.

```
train_images_df = pd.read_csv(train_images_path, header=None, delimiter=',')
train_labels_df = pd.read_csv(train_labels_path, header=None)
test_images_df = pd.read_csv(test_images_path, header=None, delimiter=',')
test_labels_df = pd.read_csv(test_labels_path, header=None)

num_classes = len(train_labels_df[0].unique()) + 1
input_shape = (32, 32, 1)

def convert_value(val):
    try:
        return float(val)
    except ValueError:
        return np.nan

train_images = train_images_df.applymap(convert_value).values.astype(np.float32)
train_labels = train_labels_df.values.astype(np.int32)
test_images = test_images_df.applymap(convert_value).values.astype(np.float32)
test_labels = test_labels_df.values.astype(np.int32)

x_train = train_images.reshape(-1, 32, 32, 1) / 255.0
x_test = test_images.reshape(-1, 32, 32, 1) / 255.0

y_train = to_categorical(train_labels, num_classes)
y_test = to_categorical(test_labels, num_classes)
```

Рисунок 3.1 – Фрагмент коду для підготовки набору даних

3.2.3 Навчання моделей згорткових нейронних мереж

Не дивлячись на кількість згорткових нейронних мереж, що описані в даній роботі, процес навчання моделей є тривіальним і повторюється від мережі до мережі, але не дивлячись на це, навчання моделей згорткових нейронних мереж – задача кропітка та потребує особливого підходу до її розв’язання. Процес навчання згорткової нейронної мережі складається з

кількох частин: визначення архітектури моделі, навчання моделі, валідація, тестування, налаштування гіперпараметрів.

Блок-схема алгоритму навчання згорткової нейронної мережі зображена на рисунку 3.2.

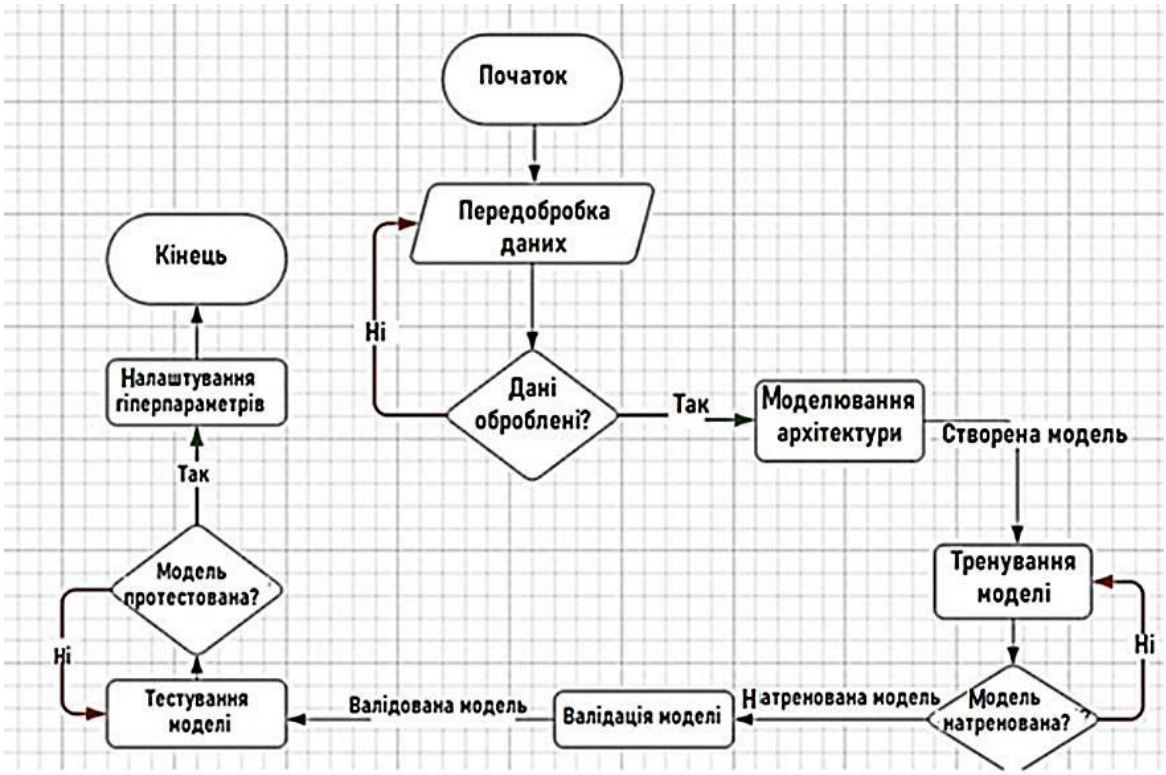

Рисунок 3.2 – Блок-схема алгоритму згорткової нейронної мережі

Підготовка навчальних даних для розпізнавання арабських символів розпочалася з процесу збору даних. Далі проведена очистка та попередня обробка даних, включаючи видалення аномалій, викидів, неправильних даних та пропущених значень, для покращення якості навчальних даних.

Після цього дані були розділені на навчальний і тестовий набори, де навчальний набір використовувався для навчання моделі, а тестовий набір – для оцінки ефективності моделі.

Після розділення дані були піддані перетворенням та нормалізації, включаючи зміну розміру зображень та масштабування значень даних до діапазону від 0 до 1. Оскільки задача розпізнавання арабських символів є

задачею класифікації, то мітки класів були перетворені в формат «one-hot encoding», де кожен клас представлений у вигляді вектора.

Після всіх цих підготовчих кроків набір даних був готовий для використання в процесі навчання та тестування моделі, що стало важливим етапом у розробці системи розпізнавання арабських символів.

Визначення архітектури моделі – це перший та ключовий крок у розробці згорткової нейронної мережі для розпізнавання арабських символів. На цьому етапі обирається структура мережі, яка визначає, як дані будуть оброблятися. Це включає в себе вибір кількості та типів шарів, які входять до моделі, зокрема, згорткові шари для виявлення ознак у зображеннях, пулінгові шари для зменшення розмірності та повністю зв’язані шари для класифікації.

Визначення архітектури моделі було першим та ключовим етапом у розробці згорткової нейронної мережі для розпізнавання арабських символів. На цьому етапі була обрана структура мережі, яка визначила, як дані оброблялися. В цю структуру входили згорткові шари для виявлення ознак у зображеннях, пулінгові шари для зменшення розмірності та повністю зв’язані шари для класифікації. Деякі параметри, такі як розмірність фільтрів для згорткових шарів, були вибрані для того, щоб оптимально виділяти важливі ознаки у зображеннях.

Також було визначено функції активації для кожного шару, такі як ReLU, Sigmoid або Tanh, які впливають на те, як модель навчалася на вхідних даних. Кількість нейронів в кожному шарі також була підібрана з урахуванням обсягу даних та складності завдання.

Важливо було досягти балансу між ефективністю моделі та уникненням перенавчання. Цей процес визначення архітектури був результатом ретельного аналізу даних та завдання, і вже був успішно завершений.

Під час завершення процесу визначення архітектури моделі була надзвичайно важливою ретельна розгляд аспектів, що впливають на її ефективність. Зокрема, було враховано значення вибору розмірності фільтрів у згорткових шарах. Цей параметр має прямий вплив на здатність моделі

розпізнавати важливі ознаки в зображеннях, і його належне налаштування гарантує високу точність.

Крім того, було приділено особливу увагу вибору функцій активації для кожного шару моделі. Такі функції, як ReLU, Sigmoid або Tanh, визначають, як модель реагує на вхідні дані та яким чином навчається. Цей аспект також має велике значення для досягнення високої ефективності моделі. Всі ці налаштування були виконані на підставі аналізу даних і конкретних вимог завдання. Важливо було підкреслити, що цей процес завершився успішно і підготовив основу для подальшого розвитку та навчання моделі.

Спочатку створюється порожня модель CNN з вибраною архітектурою. Ваги нейронів та початкові параметри зазвичай ініціалізувалися випадковим чином. Функція втрат (або функція вартості) є важливою складовою навчання нейронних мереж, включаючи згорткові нейронні мережі (CNN). Її роль полягає в оцінці того, наскільки точно модель здатна робити прогнози та визначати різницю між цими прогнозами та правильними відповідями або мітками, які представляють справжні значення в даних. Функція втрат може бути різного виду, в залежності від завдання машинного навчання.

Наприклад, для задачі класифікації зображень, може використовуватися категоріальна крос-ентропія (categorical cross-entropy) або середньоквадратична помилка (mean squared error) для регресії.

Функція втрати визначає, як буде штрафуватися модель за невірні прогнози, і цей штраф дозволяє нейронній мережі «вчитися» покращувати свої прогнози з кожним проходженням через навчальний набір даних.

Оптимізатор – це алгоритм, який використовується для мінімізації функції втрат. Його завдання полягає в тому, щоб визначити, в якому напрямку слід коригувати параметри моделі, щоб зменшити значення функції втрат і покращити прогнози. Спрощено кажучи, оптимізатор «навчає» модель, оновлюючи її ваги таким чином, якщо вони сприяють зниженню помилки. Після обчислення градієнтів функції втрат відносно внутрішніх параметрів моделі під час зворотного поширення, оптимізатор використовує ці градієнти

для корекції ваг моделі. Ця корекція включає в себе оновлення ваг в напрямку, який допомагає зменшити значення функції втрат і, отже, підвищити точність прогнозів моделі.

Такий процес повторюється на кожному кроці навчання, і модель навчається покращувати свої прогнози на основі отриманих градієнтів та впливу оптимізатора. Загалом, функція втрат і оптимізатор є важливими компонентами навчання нейронних мереж і допомагають моделі адаптуватися до навчальних даних та досягати високої точності у вирішенні різних завдань машинного навчання.

Процес навчання згорткових нейронних мереж (CNN) включає подачу навчальних даних до моделі та мінімізацію функції втрат, і це ключовий етап у розвитку глибокого навчання. На початку визначається набір навчальних даних, який складається з пар вхідних зображень та відповідних міток або класів, і ці дані використовуються для навчання моделі.

Процес навчання включає в себе пряме поширення, під час якого вхідні зображення проходять через всі шари моделі, включаючи згорткові та повнозв’язані шари, щоб згенерувати прогнози моделі для кожного зображення. Потім обчислюється функція втрати, яка вимірює різницю між прогнозами моделі та правильними мітками.

Ця функція втрати служить метрикою для оцінки того, наскільки добре модель виконує завдання, і її мета полягає в мінімізації. Після цього виконується зворотне поширення помилки, в процесі якого обчислюються градієнти функції втрати по відношенню до внутрішніх параметрів моделі. Знайдені градієнти використовуються для оновлення ваг нейронів і параметрів моделі з метою покращення її прогнозів. Цей процес повторюється на протязі кількох епох, де кожна епоха представляє собою повний прохід через весь навчальний набір даних. Під час навчання важливо враховувати контрольні метрики та проводити валідацію моделі для оцінки її здатності до узагальнення на нові дані.

Процес прямого поширення в згорткових нейронних мережах (CNN) представляє собою перший інтегральний етап обробки вхідних даних. Під час цього етапу вхідні дані, які можуть бути зображеннями, проходять через ряд згорткових шарів, де кожен шар має фільтри для витягування особливостей зображення. Операція згортки надає моделі здатність розпізнавати форми, контури та інші важливі характеристики. Після згортки, вхідні дані можуть проходити через шари пулінгу, які допомагають зменшити розмір виходу і зробити модель більш обчислювально ефективною. Це також робить мережу менш чутливою до незначних зміщень у вхідних даних. Потім інформація передається в один або декілька повнозв'язаних шарів, де генеруються прогнози.

Повнозв'язані шари мають багато нейронів і використовуються для об'єднання інформації з попередніх шарів та формування прогнозів. На останньому повнозв'язаному шарі застосовується функція активації, і результатом є вихідні прогнози моделі, які можуть вказувати, наприклад, на класи об'єктів на зображенні. Усі ці кроки у прямому поширенні допомагають моделі вчитися інтерпретувати і аналізувати вхідні дані, створюючи репрезентації, які в подальшому допомагають у здійсненні точних прогнозів або розв'язанні завдань комп'ютерного зору.

Розрахунок функції втрат: порівнюються прогнози з правильними мітками, і розраховується функція втрат.

Зворотне поширення є ключовою частиною навчання згорткових нейронних мереж (CNN). Цей процес дозволяє моделі коригувати свої внутрішні параметри, щоб покращити точність її прогнозів під час навчання. Зворотне поширення включає в себе кілька кроків, які допомагають моделі навчитися знаходити оптимальні параметри.

Спершу, обчислюються градієнти функції втрат відносно внутрішніх параметрів моделі. Градієнти вказують на те, які зміни в параметрах впливають на зміни в функції втрат. Ця інформація дозволяє моделі розуміти, які параметри потрібно коригувати для поліпшення її прогнозів. Потім

градієнти розповсюджуються назад через всі шари мережі, від останнього повнозв’язаного шару до перших згорткових шарів. Цей процес називається «зворотнім поширенням помилки» і дозволяє моделі зрозуміти, які шари внесли найбільший внесок у загальну помилку. Знайдені градієнти використовуються для оновлення ваг нейронів та параметрів моделі. Оптимізатори, такі як стохастичний градієнтний спуск або Adam, керують цим процесом. Вони коригують ваги в напрямку мінімізації функції втрат. Цей процес зворотного поширення повторюється на кожному кроці навчання, коли модель адаптується до окремих навчальних прикладів. Це дозволяє моделі здійснювати поступове вдосконалення своїх прогнозів та розширювати знання, набуте під час навчання.

Зворотне поширення допомагає моделі реагувати на помилки та покращувати її точність прогнозів. Воно важливо для досягнення високої точності в завданнях машинного навчання та для роботи з великими обсягами даних. Цей процес вимагає обчислювальних ресурсів, адже потребує багато обчислень для знаходження градієнтів та оновлення ваг моделі. Оптимізація і ефективні алгоритми градієнтного спуску допомагають прискорити цей процес.

Зворотне поширення дозволяє моделі навчитися розпізнавати складні закономірності в даних, що є ключовим для її успішного використання в різних завданнях, таких як класифікація зображень, розпізнавання об’єктів або обробка природної мови. У цьому процесі важливу роль відіграють правильно підібрані гіперпараметри, такі як швидкість навчання і розмір пакету, які впливають на швидкість та якість навчання моделі. Зворотне поширення є ітеративним процесом, який повторюється на протязі всього процесу навчання, доки модель не досягне високої точності на навчальних даних та вміння узагальнювати на нові, раніше не бачені дані. Оновлення ваг: оптимізатор використовує градієнти для корекції ваг у напрямку мінімізації функції втрат.

У ході виконання роботи було навчено та протестовано п’ять моделей згорткових нейронних мереж, коди яких подані на рисунках 3.3 – 3.7. Кожна з цих моделей представляє собою унікальну архітектуру нейронної мережі і була розроблена для вирішення конкретної задачі машинного навчання. Важливо відзначити, що кожна з цих моделей може мати свої власні параметри та гіперпараметри, які визначають її поведінку та здатність навчатися на конкретній задачі.

```
def build_model(self, input_shape, num_classes):
    inputs = Input(shape=input_shape)
    x = Conv2D(6, kernel_size=(5, 5), activation='relu', padding='valid')(inputs)
    x = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)
    x = Conv2D(16, kernel_size=(5, 5), activation='relu', padding='valid')(x)
    x = MaxPooling2D(pool_size=(2, 2), strides=(2, 2))(x)
    x = Flatten()(x)
    x = Dense(120, activation='relu')(x)
    x = Dense(84, activation='relu')(x)
    output = Dense(num_classes, activation='softmax')(x)

    self.model = Model(inputs=inputs, outputs=output)
    self.model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Рисунок 3.3 – Код розробленої моделі згорткової нейронної мережі LeNet

Рисунок 3.3 представляє першу модель згорткової нейронної мережі LeNet, яка включає в себе згорткові шари, шари пулінгу та повнозв’язані шари.

Для цієї моделі була проведена підготовка даних, навчання та тестування, і були отримані відповідні результати, які могли включати в себе метрики продуктивності, такі як точність класифікації, функції втрат та інші.

Аналогічно, рисунки 3.4 – 3.7 представляють інші чотири моделі згорткових нейронних мереж, кожна з яких має свою власну архітектуру та параметри, а саме AlexNet, VGG19, ResNet, GoogleNet.

```python
def build_model(self, input_shape, num_classes):
    inputs = Input(shape=input_shape)
    x = Conv2D(96, kernel_size=(11, 11), strides=(4, 4), padding='valid', activation='relu')(inputs)
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(3, 3), strides=(1, 1))(x)
    x = Conv2D(256, kernel_size=(5, 5), strides=(1, 1), padding='same', activation='relu')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2))(x)
    x = Conv2D(384, kernel_size=(3, 3), padding='same', activation='relu')(x)
    x = BatchNormalization()(x)
    x = Conv2D(384, kernel_size=(3, 3), padding='same', activation='relu')(x)
    x = BatchNormalization()(x)
    x = Conv2D(256, kernel_size=(3, 3), padding='same', activation='relu')(x)
    x = BatchNormalization()(x)

    x = Flatten()(x)
    x = Dense(4096, activation='relu')(x)
    x = Dropout(0.5)(x)
    x = Dense(4096, activation='relu')(x)
    x = Dropout(0.5)(x)
    output = Dense(num_classes, activation='softmax')(x)

    self.model = Model(inputs=inputs, outputs=output)
    self.model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Рисунок 3.4 – Код розробленої моделі згорткової нейронної мережі AlexNet

```python
    return self.model.predict(x)
def build_model(self, input_shape, num_classes):
    inputs = Input(shape=input_shape)
    # Block 1
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(inputs)
    x = BatchNormalization()(x)
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2, 2), strides=(2, 2))(x)
    # Block 2
    x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2, 2), strides=(2, 2))(x)
    # Block 3
    x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2, 2), strides=(2, 2))(x)
    # Block 4
    x = Conv2D(512, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2, 2), strides=(2, 2))(x)

    # Block 5
    x = Conv2D(512, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = Conv2D(512, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D((2, 2), strides=(2, 2))(x)
    # Classification block
    x = Flatten()(x)
    x = Dense(1024, activation='relu')(x)  # Увеличим размер Dense слоев
    x = Dropout(0.5)(x)  # Увеличим коэффициент Dropout
    x = Dense(512, activation='relu')(x)  # Увеличим размер Dense слоев
    x = Dropout(0.5)(x)  # Увеличим коэффициент Dropout
    x = Dense(num_classes, activation='softmax')(x)

    self.model = Model(inputs=inputs, outputs=x)
    optimizer = Adam(lr=0.0001)
    self.model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
```

Рисунок 3.5 – Код розробленої моделі згорткової нейронної мережі VGG19

```python
def build_model(self, input_shape, num_classes):
    inputs = Input(shape=input_shape)

    # Start block
    x = Conv2D(filters=64, kernel_size=(7, 7), strides=2, padding='same')(inputs)
    x = BatchNormalization()(x)
    x = Activation('relu')(x)
    x = MaxPooling2D(pool_size=(3, 3), strides=2, padding='same')(x)

    # Residual blocks
    x = self.residual_block(x, filters=64, stride=1)
    x = self.residual_block(x, filters=64, stride=1)
    x = self.residual_block(x, filters=64, stride=1)

    x = self.residual_block(x, filters=128, stride=2)
    x = self.residual_block(x, filters=128, stride=1)
    x = self.residual_block(x, filters=128, stride=1)
    x = self.residual_block(x, filters=128, stride=1)

    x = self.residual_block(x, filters=256, stride=2)
    x = self.residual_block(x, filters=256, stride=1)
    x = self.residual_block(x, filters=256, stride=1)
    x = self.residual_block(x, filters=256, stride=1)
    x = self.residual_block(x, filters=256, stride=1)
    x = self.residual_block(x, filters=256, stride=1)

    x = self.residual_block(x, filters=512, stride=2)
    x = self.residual_block(x, filters=512, stride=1)
    x = self.residual_block(x, filters=512, stride=1)

    # Last block
    x = GlobalAveragePooling2D()(x)
    x = Dropout(0.5)(x)
    x = Dense(units=num_classes, activation='softmax')(x)

    self.model = Model(inputs=inputs, outputs=x)
    self.model.compile(loss='categorical_crossentropy', optimizer=('adam'), metrics=['accuracy'])
```

Рисунок 3.6 – Код розробленої моделі згорткової нейронної мережі ResNet

```python
def build_model(self, input_shape, num_classes):
    inputs = Input(shape=input_shape)

    def inception_block(x, filters):
        branch_1 = Conv2D(filters[0], kernel_size=(1, 1), activation='relu')(x)

        branch_3 = Conv2D(filters[1], kernel_size=(1, 1), activation='relu')(x)
        branch_3 = Conv2D(filters[2], kernel_size=(3, 3), padding='same', activation='relu')(branch_3)

        branch_5 = Conv2D(filters[3], kernel_size=(1, 1), activation='relu')(x)
        branch_5 = Conv2D(filters[4], kernel_size=(5, 5), padding='same', activation='relu')(branch_5)

        branch_pool = MaxPooling2D(pool_size=(3, 3), strides=(1, 1), padding='same')(x)
        branch_pool = Conv2D(filters[5], kernel_size=(1, 1), activation='relu')(branch_pool)

        return Concatenate(axis=-1)([branch_1, branch_3, branch_5, branch_pool])

    x = Conv2D(64, kernel_size=(7, 7), strides=(2, 2), padding='same', activation='relu')(inputs)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same')(x)

    x = Conv2D(64, kernel_size=(1, 1), activation='relu')(x)
    x = Conv2D(192, kernel_size=(3, 3), padding='same', activation='relu')(x)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same')(x)

    x = inception_block(x, [64, 96, 128, 16, 32, 32])
    x = inception_block(x, [128, 128, 192, 32, 96, 64])
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='same')(x)

    x = inception_block(x, [192, 96, 208, 16, 48, 64])
    x = inception_block(x, [160, 112, 224, 24, 64, 64])
    x = inception_block(x, [128, 128, 256, 24, 64, 64])
    x = inception_block(x, [112, 144, 288, 32, 64, 64])
    x = inception_block(x, [256, 160, 320, 32, 128, 128])
    x = MaxPooling2D(pool_size=(7, 7), strides=(2, 2), padding='same')(x)

    x = AveragePooling2D(pool_size=(1, 1))(x)
    x = Flatten()(x)
    x = Dense(1024, activation='relu')(x)
    x = Dropout(0.5)(x)
    output = Dense(num_classes, activation='softmax')(x)

    self.model = Model(inputs=inputs, outputs=output)
    self.model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Рисунок 3.7 – Код розробленої моделі згорткової нейронної мережі GoogleNet

Для кожної з цих моделей також були виконані процеси підготовки даних, навчання та тестування.

Отримані результати дозволили порівняти продуктивність кожної моделі та визначити, яка з них найкраще підходить для вирішення конкретної задачі. Навчання та тестування моделей згорткових нейронних мереж є важливою частиною процесу розробки та використання глибокого навчання. Ці моделі можуть використовуватися для різних завдань, таких як класифікація зображень, виявлення об’єктів, аналіз тексту та інші. Результати їх роботи можуть мати важливе значення у вирішенні конкретних практичних завдань та покращенні якості роботи в різних галузях.

У даному підрозділі було надано докладний опис процесу навчання згорткових нейронних мереж (CNN), розпочинаючи з ініціалізації моделі та вибору функції втрат і закінчуючи безпосередньо процесом навчання. Розглянуті основні кроки в навчанні CNN, включаючи пряме та зворотне поширення, оптимізацію, а також поняття епох в контексті навчання.

Навчання згорткових нейронних мереж – це складний, але ефективний процес. Воно розпочинається з ініціалізації моделі та вибору функції втрат. Процес включає в себе пряме та зворотне поширення, де модель робить передбачення та коригує свої ваги. Навчання відбувається через кілька епох для покращення точності моделі. Важливо докладно налаштовувати параметри моделі, вибирати підходящу функцію втрат та оптимізатор, а також слідкувати за метриками для оцінки якості навчання. Результатом успішного навчання CNN може бути модель, здатна до вражаючої продуктивності в завданнях аналізу даних, класифікації зображень та інших областях.

3.3 Тестування розроблених застосунків та аналіз результатів

Тестування застосунку для арабських символів за допомогою згорткових нейронних мереж є комплексним процесом і потребує особливо підходу. Для підготовки до тестування набір даних було підготовлено, та розділено на 2 частини: 1 частина – тренувальна, 2 – частина тестувальна.

Після чого було створено п’ять моделей різних згорткових нейронних мереж. Кожна модель була окремо навчена, натренована, та проаналізована на навчальному наборі даних за допомогою оптимізатора та функції втрат. Продуктивність кожної моделі була оцінена на тестовому наборі за допомогою метрик, таких як точність та матриця помилок. Після цих кроків, була вибрана найкраща модель на основі її продуктивності.

Обрані моделі були протестовані на реальних даних, і, за необхідності, були внесені зміни у гіперпараметри або використані техніки оптимізації, такі як аугментація даних або обрізання ваги. Тренування, оцінка та тестування були повторені декілька разів для досягнення задовільних результатів на нових даних.

Важливо відзначити, що успішна навчання та розпізнавання арабських символів може вимагати кількох ітерацій та налаштувань параметрів для досягнення найкращої продуктивності.

З винятком різниць у структурах, процес навчання всіх моделей нейронних мереж виконувався за однаковим алгоритмом, що зображений на рисунку 3.2.

Першою згортковою нейронною мережею, з якою було проведено тестування була LeNet.

Результати тестування зображені на рисунках 3.8 та 3.9.

Після завершення 20 епох навчання модель досягла високого рівня точності на навчальному наборі даних, приблизно 94,45%. Це свідчить про те, що модель ефективно навчилася розпізнавати арабські символи, використовуючи навчальний набір даних, на якому вона була навчена.

Точність на перевірочному наборі даних також була високою, приблизно 87,22%, що свідчить про здатність моделі ефективно працювати з валідаційним набором даних, навіть якщо він не використовувався для навчання моделі.

Найважливішим показником є точність на тестовому наборі даних, оскільки це відображає, наскільки добре модель узагальнює свої знання на

нові, раніше не бачені дані. Точність тестування складає приблизно 88%, що свідчить про високу ефективність моделі в розпізнаванні рукописних арабських символів з високою точністю.

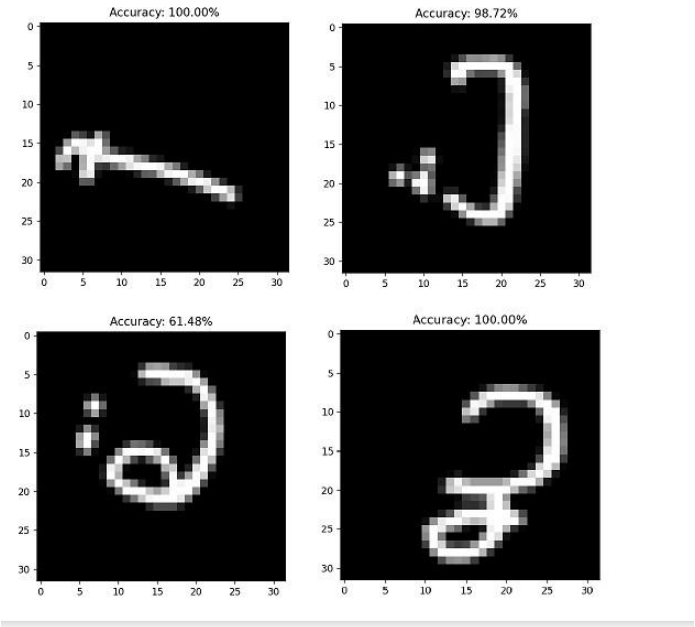

Рисунок 3.8 – Приклад розпізнавання рукописних арабських символів за допомогою LeNet

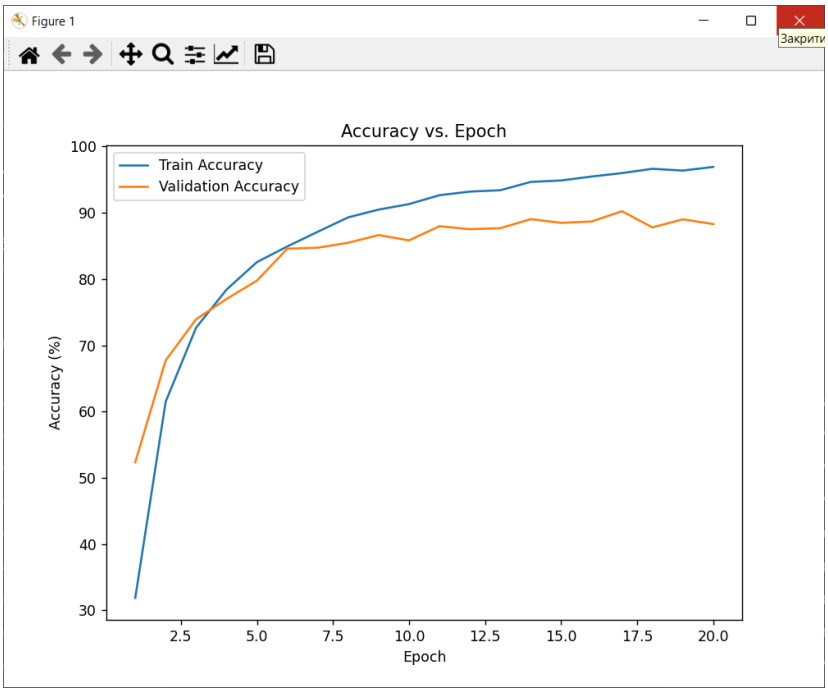

Рисунок 3.9 – Результат розпізнавання арабських рукописних символів зі згортковою нейронною мережею LeNet

Цей показник є критично важливим для практичних застосувань, оскільки він демонструє здатність моделі працювати з реальними даними і виконувати завдання розпізнавання арабських символів з високою ефективністю і точністю. Для досягнення цих результатів було проведено численні ітерації навчання та налаштування параметрів моделі.

Другою згортковою нейронною мережею яку було протестовано є нейронна мережа VGG19, яка показала одні з найкращих результатів. Результати роботи згорткової нейронної мережі VGG19 зображені на рисунках 3.10 та 3.11.

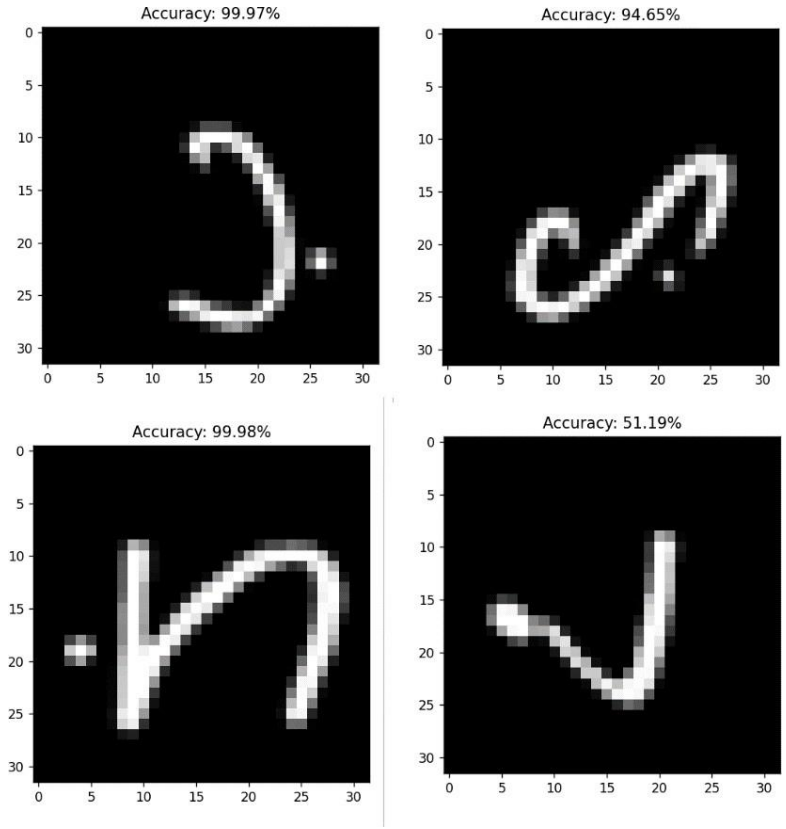

Рисунок 3.10 – Приклад розпізнавання рукописних арабських символів за допомогою VGG19

Під час навчання нейронної мережі було проведено 20 епох. На початку навчання, в першій епосі, точність була низькою, приблизно 10,90% на тренувальному наборі даних та 5,09% на валідаційному наборі даних.

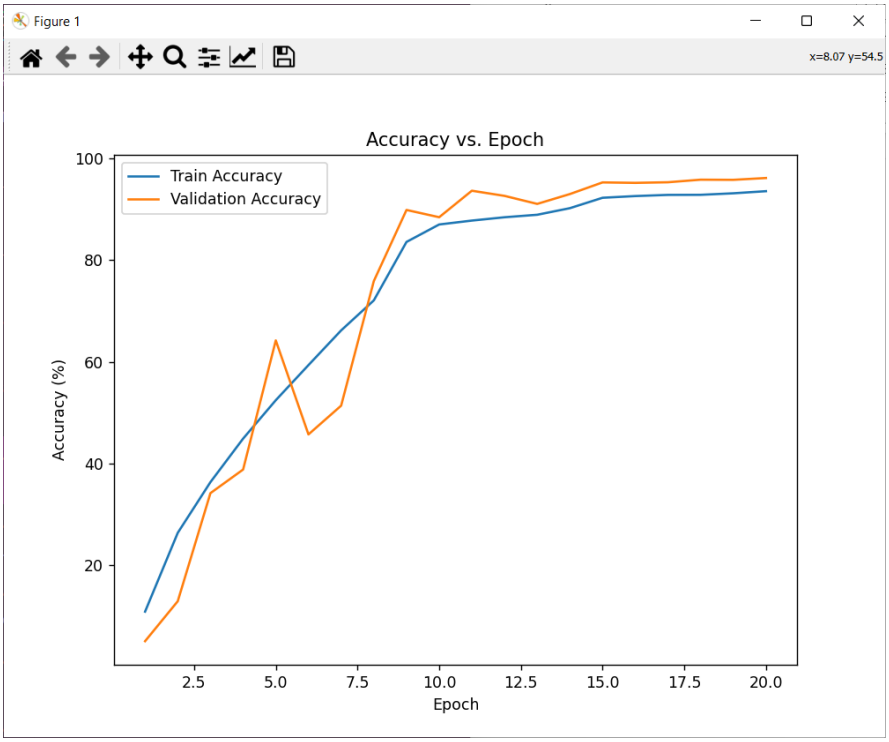

Рисунок 3.11 – Результат розпізнавання арабських рукописних символів зі згортковою нейронною мережею VGG19

Однак з кожною наступною епохою точність зростала, і після 4 епохи вона становила близько 44,92% на тренувальному наборі та 38,84% на валідаційному наборі. Далі відбулася ще одна зростаюча епоха, де точність на тренувальному наборі досягла приблизно 52,47%, а на валідаційному – 64,23%. Однак наступні епохи спостерігалася зменшення точності на валідаційному наборі. З 9 епохи точність на тренувальному наборі стала набагато вищою, приблизно 83,59%, з 10 епохи точність на валідаційному наборі також підвищилася до близько 88,45%. Після цього спостерігалася подальша зростаюча тенденція точності на валідаційному наборі. Загалом, під час навчання нейронної мережі точність поступово зростала, і після завершення навчання на валідаційному наборі точність розпізнавання досягла близько 96,65%.

Такий процес свідчить про те, що модель навчилася добре адаптуватися до даних та ефективно визнавати арабські символи.

Третьою згортковою нейронною мережею є ResNet, яка відрізняється з поміж інших найбільшою кількістю шарів. Результати розпізнавання арабських символів за допомогою згорткової нейронної мережі ResNet зображено на рисунках 3.12 та 3.13.

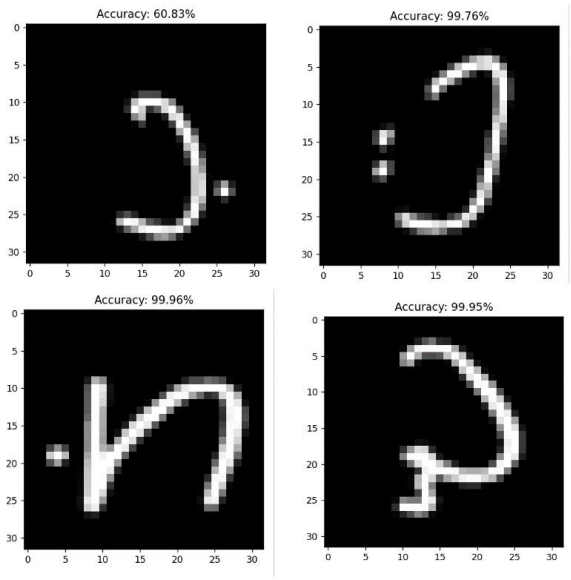

Рисунок 3.12 – Приклад розпізнавання рукописних арабських символів за допомогою ResNet

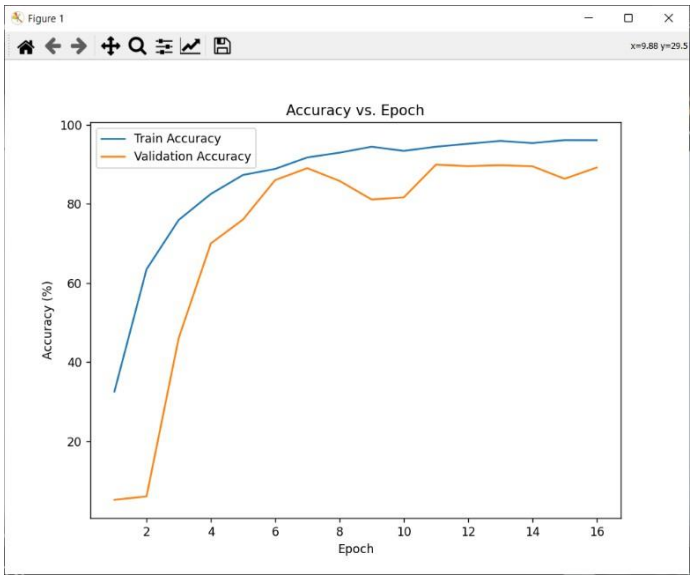

Рисунок 3.13 – Результат розпізнавання арабських рукописних символів зі згортковою нейронною мережею ResNet

Графік навчання моделі показує, що втрати зменшуються з кожною наступною епохою. Це свідчить про успішне навчання моделі та зменшення помилки на навчальному наборі даних.

Початкова величина втрат після першої епохи становила 2,49, і ця величина зменшилася до 0,13 після 16 епохи. Це свідчить про поступове покращення моделі і наближення до оптимального рішення. Також, важливо враховувати втрати на валідаційному наборі даних. Початкова величина цієї метрики була високою – 8,70, що може свідчити про перенавчання моделі. Проте з часом втрати на валідаційних даних почали зменшуватися і досягли 0,41. Це свідчить про поліпшення здатності моделі до узагальнення даних та покращення її якості. Результати також вказують на покращення точності моделі.

Початкова точність на навчальному наборі даних становила 32,57%, а після 20 епох вона зросла до 96,13%. Точність на валідаційних даних також покращилася, піднявшись з 5,24% до 89,20%.

Перенавчання моделі спостерігалося на ранніх стадіях навчання, але зі зменшенням втрат та покращенням точності на валідаційних даних це перенавчання послаблювалося. У підсумку, модель успішно навчилася та продемонструвала значне покращення у якості прогнозів та здатності до узагальнення даних. Однак для оцінки кінцевої якості моделі і прийняття рішення про завершення навчання може знадобитися додатковий аналіз та моніторинг результатів.

Наступна нейронна мережа для навчання та обробки це GoogleNet, також відома як Inception, є однією з ключових згорткових нейронних мереж, розроблених компанією Google DeepMind. Вона представляла собою потужну та вдосконалену архітектуру глибокого навчання, яка стала популярною завдяки своїй ефективності у вирішенні різних завдань комп’ютерного зору, включаючи розпізнавання об’єктів та класифікацію зображень.

Архітектура GoogleNet використовує інноваційні підходи, такі як модуль Inception, що дозволяє об’єднати згорткові шари різних розмірів та

фільтрів для покращення точності та швидкості навчання. Ця нейронна мережа вражає своєю здатністю працювати з великою кількістю параметрів, залишаючись при цьому відносно легкою та ефективною.

Приклади розпізнавання згортковою нейронною мережею зображені на рисунках 3.14 та 3.15.

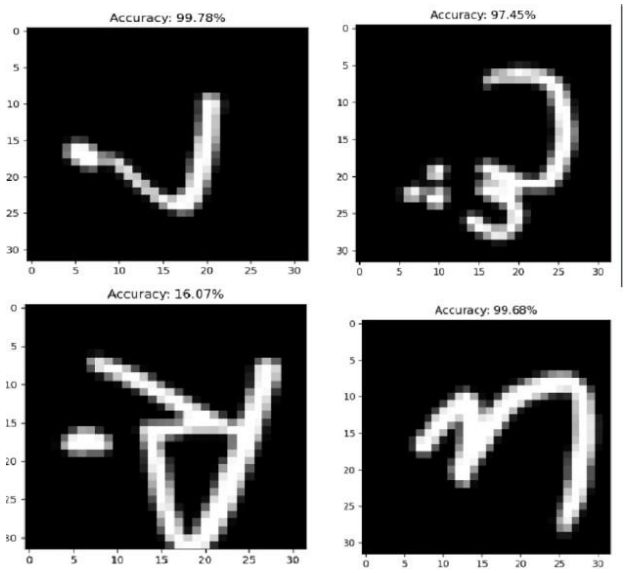

Рисунок 3.14 – Приклад розпізнавання рукописних арабських символів за допомогою GoogleNet

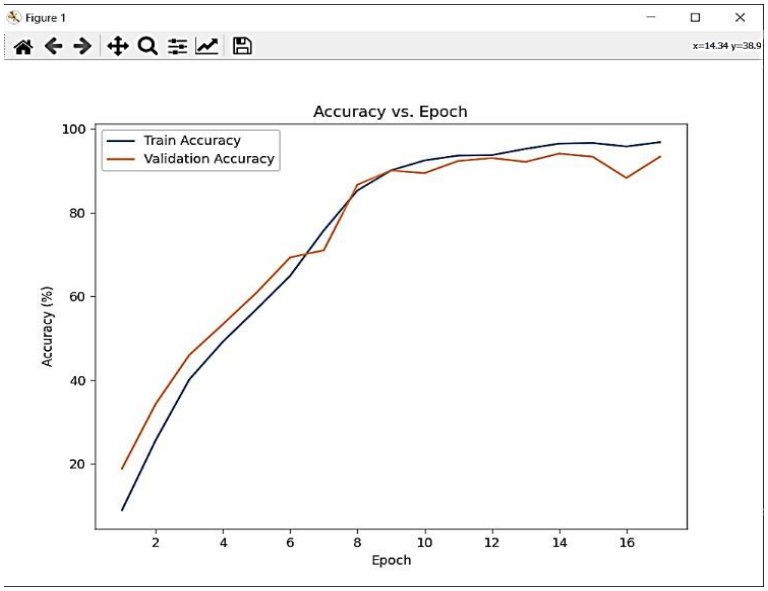

Рисунок 3.15 – Результат розпізнавання арабських рукописних символів зі згортковою нейронною мережею GoogleNet

На графіку навчання моделі видно, що функція втрат постійно зменшується з кожною наступною епохою, починаючи зі значення 28,69% і досягаючи 1,16% після 20 епохи. Це свідчить про те, що модель успішно навчається і зменшує помилку на навчальному наборі даних.

Точність моделі також покращується з 8,97% на початку навчання до 96,83% після 20 епох.

Це означає, що модель стає більш точною у класифікації даних.

Щодо функції втрат на валідаційному наборі, на початку вона мала високе значення 2,22, але з часом зменшилася до 0,29. Це свідчить про поліпшення здатності моделі до узагальнення даних, що є важливим для успішного застосування моделі на нових даних.

Точність на валідаційному наборі також зросла з початкових 18,90% до 93,36% після 20 епох. Це підтверджує, що модель стає більш точною на незалежних даних.

Остання з п’яти нейронних мереж, яка була розглянута – AlexNet, яка є однією з важливих моделей у світі глибокого навчання. Ця нейронна мережа, запропонована Геоффреєм Хінтоном та його командою, відзначилася в конкурсі ImageNet у 2012 році і відкрила нову еру глибокого навчання. Модель складається з глибоких згорткових та повнозв’язаних шарів та використовує методи активації та регуляризації для досягнення вражаючої точності в класифікації зображень.

AlexNet була першою нейронною мережею, яка використовувала декілька графічних процесорів (GPU) для прискорення навчання, і це визначило новий стандарт для розвитку глибокого навчання. Її архітектура стала основою для багатьох інших глибоких мереж і залишається важливим внеском у галузі машинного навчання.

AlexNet дала старт епосі глибокого навчання, і вона продовжує впливати на розробку нейронних мереж та їх застосування в різних сферах, включаючи комп’ютерний бачення, розпізнавання мови та багато інших.

Приклади розпізнавання згортковою нейронною мережею зображені на рисунках 3.16 та 3.17.

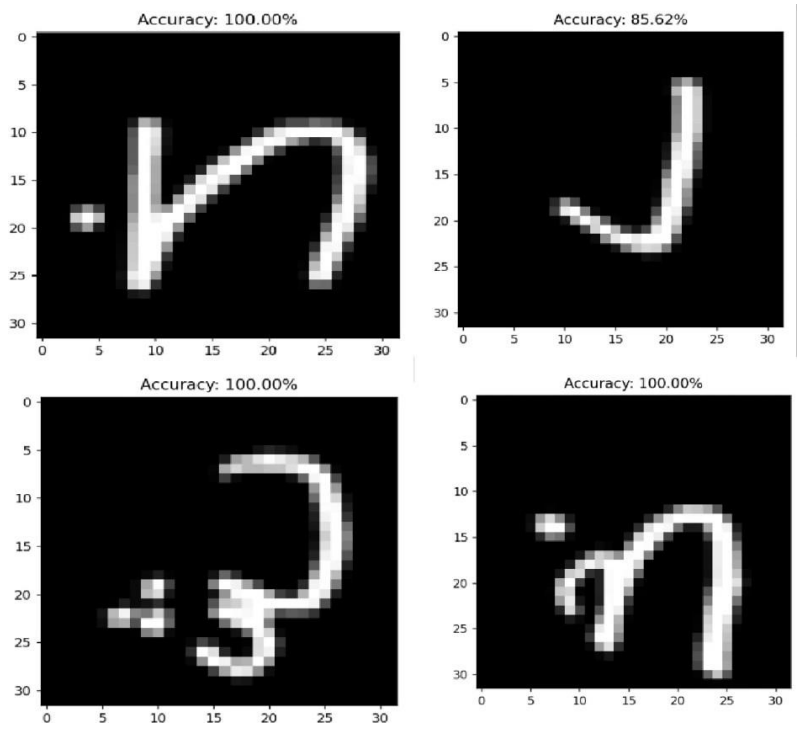

Рисунок 3.16 – Приклад розпізнавання рукописних арабських символів за допомогою AlexNet

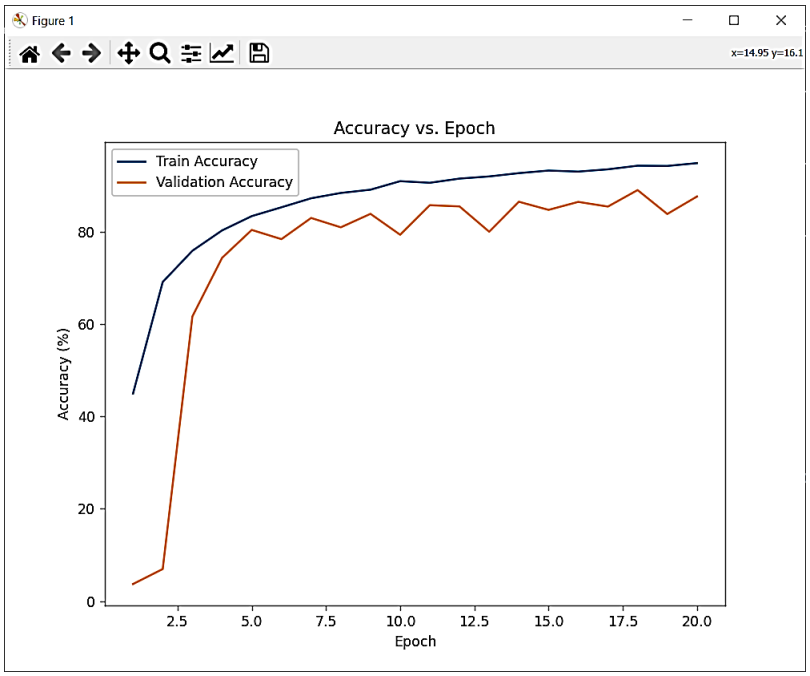

Рисунок 3.17 – Результат розпізнавання арабських рукописних символів зі згортковою нейронною мережею AlexNet

У результаті аналізу 20 епох навчання моделі можна побачити динаміку її покращення.

На початку навчання в першій епосі модель мала високу втрату 17,20% та низьку точність 44,96% на навчальному наборі, а також ще більшу втрату 44,03% та мінімальну точність (3,72%) на валідаційному наборі. Проте з кожною подальшою епохою стан моделі значно покращувався.

У другій епосі втрата зменшилася 9,06%, а точність зросла до 69,11% на навчальному наборі, але на валідаційному наборі ще не відбулося значних змін у втраті та точності.

У третій епосі продовжувалося покращення, з втратою, що зменшилася до 7,05%, і точністю на навчальному наборі, яка досягла 75,88%. На валідаційному наборі втрата зменшилася значно 12,67%, і точність покращилася до 61,67%.

У четвертій і п’ятій епохах модель продовжила покращуватися з ще більш низькими втратами і високою точністю на навчальному наборі. Втрата на валідаційному наборі також зменшилася, і точність стала значно кращою.

У наступних епохах, шостій і сьомій, модель продовжувала покращувати свої результати, з низькими втратами та високою точністю на навчальному наборі. Втрата на валідаційному наборі продовжила зменшуватися, і точність піднялася на новий рівень.

Восьма та дев’ята епохи продовжили тренд покращення точності та зменшення втрат на навчальному та валідаційному наборах.

Десята епоха показала додатковий ріст точності на навчальному наборі. У наступних епохах зберігався загальний тренд поліпшення якості моделі.

Дванадцята епоха відзначалася високою точністю та дуже низькими втратами на навчальному наборі. Втрата на валідаційному наборі зменшилася, і точність також стала високою.

Тринадцята і чотирнадцята епохи продовжували підтверджувати високу точність моделі, а втрата на валідаційному наборі продовжувала зменшуватися.

П’ятнадцята епоха відзначилася ще більш високою точністю на навчальному наборі. У шістнадцятій і сімнадцятій епохах точність моделі підтверджувалася, і втрата на валідаційному наборі лише трохи збільшилася.

Вісімнадцята епоха продовжила тренд покращення, з високою точністю на навчальному наборі та низькими втратами. Втрата на валідаційному наборі також зменшилася.

У дев’ятнадцятій епосі точність на навчальному наборі стала дуже високою, а втрата низькою. На валідаційному наборі втрата підвищилася, але точність залишилася високою.

Двадцята епоха завершила процес навчання зі значною точністю на навчальному наборі та низькими втратами. Втрата на валідаційному наборі залишилася низькою, а точність на високому рівні. Таким чином, модель підтвердила свою здатність до класифікації даних з високою точністю та мінімальною втратою після 20 епох навчання.

Виходячи з аналізу 20 епох навчання, можна зробити висновок про високу ефективність моделі. Початкова модель почала навчання з високою втратою та низькою точністю, але з кожною наступною епохою становилася значно ефективнішою. На навчальному наборі втрата постійно зменшувалася, а точність зростала, досягаючи вражаючих 94,85% на завершальній епосі.

Це свідчить про високу ефективність моделі у класифікації навчальних даних. Найбільше вражає, як модель успішно узагальнювала навчання на валідаційний набір. Втрата на валідаційному наборі також зменшувалася з часом і становила лише 5,53% на останній епосі, що є дуже низьким значенням.

Точність на валідаційному наборі зростала і склала 87,62% на останній епосі, що підтверджує високу ефективність моделі у здатності до класифікації незалежних даних. Загалом, модель AlexNet, навчена протягом 20 епох, продемонструвала вражаючу ефективність в класифікації зображень.

Її здатність до зменшення втрат та підвищення точності на навчальному та валідаційному наборах свідчить про її успішність та потенціал для розпізнавання та класифікації даних в реальних умовах.

Серед розглянутих моделей розпізнавання арабських символів виділяється ResNet завдяки її вражаючій високій точності на багатьох класах. Ця модель здатна ефективно розпізнавати та аналізувати різноманітні образи завдяки своїй складній та добре розробленій архітектурі.

ResNet продемонструвала найкращу продуктивність, забезпечуючи високу точність на більшості класів. Хоча інші моделі також показали себе добре, їхня точність варіювалася в залежності від особливостей класів та архітектурних рішень. Важливо відзначити, що завдання розпізнавання рукописних арабських символів є викликом, і обрана модель має вирішити це завдання настільки точно, наскільки це можливо. Це стосується як метрик навчання, так і підтвердження на валідаційному наборі даних.

ResNet виділяється завдяки вражаючим результатам з точності на більшості класів, що робить її найперспективнішою моделлю для досягнення точних та стійких результатів у розпізнаванні арабських символів. Висока продуктивність у навчанні та валідації підкреслює її ефективність для розв’язання цієї конкретної задачі. Загалом, використання згорткових нейронних мереж у розпізнаванні арабських символів відкриває широкі можливості.

Подальший розвиток технологій та методів у цій галузі може призвести до створення більш точних, гнучких та всебічних систем розпізнавання арабського тексту.

Результати порівняння моделей, показані у таблиці 3.1, підтверджують важливість подальших досліджень та інновацій у цьому напрямку.

Таблиця 3.1 – Результат порівняння моделей згорткових нейронних мереж

	LeNet	**AlexNet**	**VGG19**	**ResNet**	**GoogLeNet**
خ	89,98%	100%	97,65%	99,99%	100%
د	99,99%	99,99%	99,99%	99,99%	99,99%
ذ	65,58%	41,30%	65,58%	98,19%	99,78%
ر	0,18%	100%	91,18%	99,48%	99,98%
ز	99,99%	99,87%	99,99%	99,87%	99,99%
س	0,18%	100%	90,18%	99,48%	99,98%
ش	99,86%	100%	99,99%	99,87%	99,99%
ص	98,80%	99,57%	99,99%	100%	99,99%
ض	99,95%	100%	100%	99,99%	100%
ط	75%	99,99%	99,28%	99,03%	99,28%
ظ	99,35%	99,11%	21,36%	99,96%	16,07%

3.4 Перспективи подальшої роботи

Розроблений застосунок відповідає вимогам поставленої задачі і вміло розпізнає арабські символи на еталонних зображеннях, використовуючи конкретний набір даних.

Важливо відзначити, що цей набір даних є одним із найкращих для навчання нейронних мереж у розпізнаванні арабських символів, завдяки своїй репрезентативності та різноманітності.

Загальна оцінка роботи застосунку свідчить про те, що він показав добрі результати та відповідає цілям розробки. Навіть при виявленні деяких недоліків під час тестування, потенціал для подальшого вдосконалення застосунку виявився більшим за його обмеження.

Серед можливих шляхів поліпшення застосунку важливо відзначити необхідність більшої суб’єктивізації процесу навчання нейронної мережі та подальшого покращення точності розпізнавання арабських символів.

Незважаючи на існуючі обмеження та недоліки, необхідно підкреслити, що головна мета дослідження була досягнута, і застосунок працює відповідно до початкових планів та очікувань. Це свідчить про важливий крок у розвитку системи розпізнавання арабських символів.

Застосунок відкриває широкий спектр можливостей для подальшої роботи у галузі розпізнавання арабських символів за допомогою нейронних мереж. Досягнення в технологіях та методах у цій області можуть призвести до створення більш точних, гнучких та багатофункціональних систем для розпізнавання арабського тексту.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований метод розпізнавання арабських символів за допомогою згорткових нейронних мереж.

Виконано всі поставлені задачі, а саме:

– проаналізовані методи розпізнавання арабських символів, що дало змогу в подальшому обрати найкращі методи для перевірки точності та працездатності застосунку;

– розроблено методику розпізнавання арабських за допомогою згорткових нейронних мереж, яка в подальшому була використана для збору даних та перевірки ефективності застосунку;

– визначено необхідні інструментальні засоби для створення застосунку;

– виконано всі етапи розроблення застосунку для розпізнавання арабських символів, на кожному етапі здійснено різноманітні програмні та алгоритмічні оптимізації, реалізовано гнучкий метод роботи застосунку, його взаємодії з користувачем;

– проведено тестування розробленого застосунку, за результатами якого було визначено особливості роботи реалізованого методу, умови використання та його ефективність, яка виявилася на даному етапі задовільною, але водночас надавала розуміння необхідности ширших випробувань та можливого подальшого доопрацювання. У ході тестування було виявлено, що із зазначеною задачею найкраще впоралася модель згорткової нейронної мереді ResNet із точністю близкь 97,01%, що є вражаючим результатом;

– визначено перспективи подальшої роботи, які виходять з отриманих результатів тестування та теоретичних знань в досліджуваній області, а саме подальший розвиток моделей згорткових нейронних мереж, покращення

набору даних, спроби зменшення суб’єктивації під час навчання згорткових нейронних мереж.

Результати дослідження апробовано у вигляді 2 статей у зарубіжних наукових журналах [24], [27] та тез доповіді під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» [36].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кочерган, М. П. (2001). Вступ до мовознавства: монографія. К. Видавн. центр "Академія", 368 с.

2. Елаіан, А. Б. (2017). *Арабська мова: історія, особливості, перспективи розвитку* (Doctoral dissertation, ВНТУ).

3. El-Sawy, A., Loey, M., & El-Bakry, H. (2017). Arabic handwritten characters recognition using convolutional neural network. *WSEAS Transactions on Computer Research*, *5*(1), 11-19.

4. Al-Salman, A., & Alyahya, H. (2017, October). Arabic online handwriting recognition: a survey. In *proceedings of the 1st international conference on internet of things and machine learning* (pp. 1-4).

5. Sahu, M. K., & Dewangan, D. N. K. (2017). A survey on handwritten character recognition. *International Advanced Research Journal in Science, Engineering and Technology*, *4*(1).

6. Faizullah, S., Ayub, M. S., Hussain, S., & Khan, M. A. (2023). A Survey of OCR in Arabic Language: Applications, Techniques, and Challenges. *Applied Sciences*, *13*(7), 4584.

7. Muda, N., Ismail, N. K. N., Bakar, S. A. A., & Zain, J. M. (2007, August). Optical character recognition by using template matching (alphabet). In *National Conference on Software Engineering & Computer Systems* (pp. 1-6).

8. Isheawy, N. A. M., & Hasan, H. (2015). Optical character recognition (OCR) system. *IOSR Journal of Computer Engineering (IOSR-JCE), e-ISSN*, 2278-0661.

9. Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*.

10. Fu, G., Jin, Y., Sun, S., Yuan, Z., & Butler, D. (2022). The role of deep learning in urban water management: A critical review. *Water Research*, 118973.

11. Shen, C. (2018). A transdisciplinary review of deep learning research and its relevance for water resources scientists. *Water Resources Research*, *54*(11), 8558-8593.

12. Mishra, R. K., Reddy, G. S., & Pathak, H. (2021). The understanding of deep learning: a comprehensive review. *Mathematical Problems in Engineering*, *2021*, 1-15.

13. Wu, M., Liu, X., Gui, N., Yang, X., Tu, J., Jiang, S., & Zhao, Q. (2023). Prediction of the remaining time and time interval of pebbles in pebble bed HTGRs aided by CNN via DEM datasets. *Nuclear Engineering and Technology*, *55*(1), 339-352.

14. DS, P. (2022). COVID-19 infection prediction from CT scan images of lungs using Iterative Convolution Neural Network Model. *Advances in Engineering Software (Barking, London, England: 1992)*, 103214-103214.

15. Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*.

16. Bohmrah, M. K., & Kaur, H. (2021). Classification of Covid-19 patients using efficient fine-tuned deep learning DenseNet model. *Global Transitions Proceedings*, *2*(2), 476-483.

17. Mao, W. L., Fathurrahman, H. I. K., Lee, Y., & Chang, T. W. (2020). EEG dataset classification using CNN method. In *Journal of physics: conference series* (Vol. 1456, No. 1, p. 012017). IOP Publishing.

18. Khan, A., Sohail, A., Zahoora, U., & Qureshi, A. S. (2020). A survey of the recent architectures of deep convolutional neural networks. *Artificial intelligence review*, *53*, 5455-5516.

19. Shri, S. J., & Jothilakshmi, S. J. C. C. (2019). Crowd video event classification using convolutional neural network. *Computer Communications*, *147*, 35-39.

20. Roncancio, R., El Gamal, A., & Gore, J. P. (2022). Turbulent flame image classification using Convolutional Neural Networks. *Energy and AI*, *10*, 100193.

21. Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, *29*(9), 2352-2449.

22. Introducing Deep Learning with MATLAB. URL: https://ch.mathworks.com/campaigns/offers/deep-learning-with-matlab.html (дата звернення 14.10.2023).

23. Van Houdt, G., Mosquera, C., & Nápoles, G. (2020). A review on the long short-term memory model. *Artificial Intelligence Review*, *53*, 5929-5955.

24. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25-36.

25. Tvoroshenko I., and Gorokhovatskyi V. (2022) The application of hybrid intelligence systems for dynamic data analysis, *International Journal of Engineering and Information Systems*, 6(2), pp. 40–48.

26. Tvoroshenko, I., & Kukharchuk, V. (2021). Current state of development of applications for recognition of faces in the image and frames of video captures.

27. Pomazan, V., Tvoroshenko, I., & Gorokhovatskyi, V. (2023). Handwritten character recognition models based on convolutional neural networks.

28. Gorokhovatsky, V., Vlasenko, N., & Rybalka, M. (2021). Застосування засобів хешування даних для прискорення класифікаційних рішень у структурних методах розпізнавання зображень. *Advanced Information Systems*, *5*(2), 13-20.

29. Tvoroshenko, I. S., & Gorokhovatskyi, V. O. (2020). Effective tuning of membership function parameters in fuzzy systems based on multi-valued interval logic. *Telecommunications and Radio Engineering*, *79*(2).

30. Gorokhovatskyi, V.O., Tvoroshenko, I.S., and Peredrii O.O. (2020) Image classification method modification based on model of logic processing of bit description weights vector, *Telecommunications and Radio Engineering*, 79(1), pp. 59-69.

31. Daradkeh, Y.I., Gorokhovatskyi, V., Tvoroshenko, I., Gadetska, S., and Al-Dhaifallah, M. (2021) Methods of Classification of Images on the Basis of the Values of Statistical Distributions for the Composition of Structural Description Components, *IEEE Access*, 9, pp. 92964-92973.

32. Asim, M., Ming, Z., & Javed, M. Y. (2017, June). CNN based spatio-temporal feature extraction for face anti-spoofing. In *2017 2nd International Conference on Image, Vision and Computing (ICIVC)* (pp. 234-238). IEEE.

33. Bai, M., & Goecke, R. (2020, October). Investigating LSTM for micro-expression recognition. In *Companion Publication of the 2020 International Conference on Multimodal Interaction* (pp. 7-11).

34. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., Gadetska S., and Al-Dhaifallah M. (2023) Statistical data analysis models for determining the relevance of structural image descriptions, *IEEE Access*, 11, pp. 126938-126949.

35. ChowdaryM. K., Nguyen, T. N., & Hemanth, D. J. (2021). Deep learning-based facial emotion recognition for human–computer interaction applications. *Neural Computing and Applications*, 1-18.

36. Помазан, В. (2022). Аналіз технологій ідентифікації, розпізнавання та оброблення емоцій людини на зображеннях.

37. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Al-Dhaifallah M. (2022) Classification of Images Based on a System of Hierarchical Features, *Computers, Materials & Continua*, 72(1), pp. 1785–1797.

38. Gorokhovatskyi, V., Tvoroshenko, I., Kobylin, O., & Vlasenko, N. (2023). Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

39. Гороховатський В., Творошенко І., Сидоренко Д. (2021) Класифікація зображень із використанням кластерного подання, *Міжн. наук. симпозіум Інтелектуальні рішення-С. Обчислювальний інтелект. Теорія прийняття рішень: праці міжн. наук. симп. (Вересень 29, 2021)*. Київ-Ужгород, С. 44-45.

40. Gorokhovatskyi V., Tvoroshenko I. (2023) Identification of visual objects by the search request. *International scientific symposium «INTELLIGENT SOLUTIONS-S». Computational intelligence (results, problems and perspectives). Decision making theory: proceedings of the international symposium*, September 28, 2023, Kyiv-Uzhorod, Ukraine, pp. 25-27.

41. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57-70.

42. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2022) Tools for fast metric data search in structural methods for image classification, *IEEE Access*, 10, pp. 124738-124746.

43. Гороховатський В.О., Творошенко І.С., Чмутов Ю.В. (2022) Застосування систем ортогональних функцій для формування простору ознак у методах класифікації зображень, *Сучасні інформаційні системи*, 6(3), С. 5-12.

44. Гороховатський В., Передрій О., Творошенко І., Марков Т. (2023) Матриця відстаней для множини компонентів структурного опису як інструмент для створення класифікатора зображень, *Сучасні інформаційні системи*, 7(1), С. 5-13.