Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

факультет інтелектуальних інформаційних технологій та автоматизації
(повне найменування інституту, назва факультету (відділення))

Кафедра комп`ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія розпізнавання рукописних цифр спайкінговою нейронною мережею»

Виконав: студент 2-го курсу, групи 1КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Савич В. Д.
(прізвище та ініціали)

Керівник: к.т.н., професор каф. КН
_____ Колесницький О.К.
(прізвище та ініціали)
«07» 12 2023 р.

Опонент: к.т.н., професор каф. КСУ
_____ Биков М. М.
(прізвище та ініціали)
«07» 12 2023 р.

Допущено до захисту
Завідувач кафедри КН
д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«09» 12 2023 р.

Вінниця ВНТУ - 2023 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.

29.08. **2023 року**

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Савичу Віталію Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія розпізнавання рукописних цифр спайкінговою нейронною мережею

керівник роботи_ к.т.н., професор кафедри КН Колесницький О.К.
затверджені наказом вищого навчального закладу від "18" 09 2023 року № 247
2. Строк подання студентом роботи 13.11. 202_ року

3. Вихідні дані до роботи:
Вхідні дані – розмірність символу – 28x28 пікселів, кількість вихідних нейронів – не менше 50, обсяг навчальної вибірки – не менше 1000, обсяг тестової вибірки – не менше 100, вид класифікатора – нейронна мережа.

4. Зміст текстової частини:
Вступ, аналіз предметної області розпізнавання рукописних цифр, розробка інформаційної технології розпізнавання рукописних цифр, програмна реалізація інформаційної технології розпізнавання рукописних цифр, економічна частина, висновки, перелік використаних джерел, додатки

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
Схема алгоритму роботи програми
Архітектура спайкінгової нейронної мережі
Діаграми класів та компонентів проекту
Вид вікна програми
Приклад роботи програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціалита посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-4	Колесницький О.К., к.т.н., проф. каф. КН	29.08.23	10.11.23
5	Адлер О. О., к.т.н., доц. каф. ЕПВМ	8.10.23	21.10.23

7. Дата видачі завдання 29.08 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасного рівня інформаційних технологій розпізнавання рукописних цифр. Постановка задач дослідження	01.09.23 - 05.09.23	Розділ 1
2	Побудова моделей розпізнавання рукописних цифр на основі нейромереже та функціонування нейронної мережі	06.09.23 - 16.09.23	Розділ 2
3	Практичне застосування та оцінка ефективності розроблених моделей	17.09.23 - 07.10.23	Розділ 3
4	Підготовка економічної частини	08.10.23 - 23.10.23	Розділ 4,5
5	Апробація та/або впровадження результатів дослідження	24.10.23 - 01.11.23	Тези доповідей
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	02.11.23 - 10.11.23	Пояснювальна записка, графічний матеріал, презентація

Студент _____ (підпис) Савич В. Д.

Керівник роботи _____ (підпис) Колесницький О.К.

АНОТАЦІЯ

УДК 004.8

Савич В. Д. Інформаційна технологія розпізнавання рукописних цифр спайкінговою нейронною мережею. Магістерська кваліфікаційна робота зі спеціальності 122 – «Комп’ютерні науки», освітня програма – «Системи штучного інтелекту». Вінниця: ВНТУ, 2023. 102 с.

На укр. мові. Бібліогр.: 25 назв; рис.: 25; табл. 7.

У даній магістерській кваліфікаційній роботі розроблено інформаційну технологію розпізнавання рукописних цифр спайкінговою нейронною мережею. Було розроблено архітектуру спайкінгової нейромережі для інформаційної технології розпізнавання рукописних цифр, яка має 784 входи та 2 шари спайкінгових нейронів. Програмна реалізація інформаційної технології розпізнавання рукописних цифр виконано на мові Python з використанням спеціалізованої бібліотеки BRIAN. Навчання програми відбувалось з використанням бази даних MNIST. Розроблена програма має достовірність розпізнавання рукописних цифр 91%, а програма-аналог - 84%, тобто розроблена програма має збільшену на 7% достовірність розпізнавання рукописних цифр.

Графічна частина складається із 6 плакатів.

У економічному розділі доведено доцільність проведення науково-дослідної роботи за даною темою. Термін окупності становить 0,79 р., що свідчить про комерційну привабливість науково-технічної розробки.

Ключові слова: розпізнавання, рукописні цифри, спайкінгова нейронна мережа, MNIST.

ABSTRACT

Savych V. D. Information technology for recognition of handwritten digits by a spiking neural network. Master's thesis in the specialty 122 - «Computer sciences», educational program – «Artificial intelligence systems». Vinnytsia: VNTU, 2023. 102 p.

In Ukrainian language. Bibliogr .: 25 titles; fig . 25; table 7.

In this master's qualification work, information technology for recognition of handwritten digits by a spiking neural network was developed. A spiking neural network architecture for handwritten digit recognition information technology has been developed, which has 784 inputs and 2 layers of spiking neurons. The software implementation of the information technology for recognizing handwritten digits is made in Python using the specialized BRIAN library. The program was trained using the MNIST database. The developed program has an accuracy of 91% for recognizing handwritten digits, and the analog program - 84%, that is, the developed program has an increased accuracy of recognizing handwritten digits by 7%.

The graphic part consists of 6 posters.

In the economic section, the expediency of conducting research work on this topic is proven. The payback period is 0.79 years, which indicates the commercial attractiveness of scientific and technical development.

Keywords: recognition, handwritten digits, spiking neural network, MNIST.

ЗМІСТ

ВСТУП 5
1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗПІЗНАВАННЯ ОБРАЗІВ……9
1.1 Постановка задачі 9
1.2 Огляд методів розпізнавання зображень 10
1.2.1 Метод розпізнавання по шаблонах 11
1.2.2 Метод структурного підходу 12
1.2.3 Метод контекстного розпізнавання 12
1.2.4 Метод на основі корекції помилок.....13
1.2.5 Метод на основі нейронних мереж 14
1.3 Обгрунтування вибору аналогу до програми розпізнавання рукописних цифр 15
1.4 Висновок до розділу 1 18
2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР СПАЙКІНГОВОЮ НЕЙРОННОЮ МЕРЕЖЕЮ..... 19
2.1 Обгрунтування вибору спайкінгової нейромережі 19
2.2 Розробка архітектури спайкінгової нейромережі..... 29
2.3 Математична модель та порядок функціонування спайкіногової нейроної мережі для розпізнавання рукописних цифр..... 33
2.3.1 Навчання мережі 34
2.3.2 Стійкість мережі.....36
2.3.3 Вхідне кодування 37
2.3.4 Навчання та класифікація 38

2.4 Особливості реалізації запропонованої архітектури спайкінгової нейромережі 38

2.4.1 Гальмування 38

2.4.2 Навчання на основі спайків для машинного навчання 39

2.4.3 Конкурсне навчання 40

2.4.4 Надійність навчання 41

2.5 Висновок до розділу 2 42

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР СПАЙКІНГОВОЮ НЕЙРОННОЮ МЕРЕЖЕЮ 43

3.1 Обґрунтування вибору мови та середовища програмування 43

3.2 Розробка алгоритму роботи програми розпізнавання рукописних цифр 45

3.3 Програмна реалізація розпізнавання рукописних цифр 47

3.4 Висновок 51

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМИ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР СПАЙКІНГОВОЮ НЕЙРОННОЮ МЕРЕЖЕЮ 52

4.1 Тестування програми розпізнавання рукописних цифр спайкінговою нейронною мережею 52

4.2 Аналіз результатів роботи програми розпізнавання рукописних цифр спайкінговою нейронною мережею 57

4.3 Висновок до розділу 4 60

5 ЕКОНОМІЧНА ЧАСТИНА 61

5.1 Проведення комерційного та технологічного аудиту інформаційної технології розпізнавання рукописних цифр спайкінговою нейронною мережею. 61

5.2 Розрахунок витрат на здійснення науково-дослідної роботи 62

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором..... 68

5.4 Висновок до розділу 5 73

ВИСНОВКИ 74

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ..... 76

Додаток А (обов’язковий) ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ 79

Додаток Б (обов’язковий) Лістинг програми 80

Додаток В (обов’язковий) ІЛЮСТРАТИВНА ЧАСТИНА 93

Додаток Г (довідниковий) Інструкція користувача..... 100

ВСТУП

Актуальність. Розпізнавання - це операція визначення ступеня подібності даного конкретного об'єкта та образів інших об'єктів або до узагальнених образів класів, в результаті якої формується рейтинг об'єктів або класів за спаданням подібності до розпізнаваного об'єкту.

У результаті маємо ситуацію, коли пристрої розпізнавання можуть підвищувати, наприклад, якість систем зв'язку (розпізнаючи сигнали на фоні шумів), допомагають ставити об'єктивний діагноз в медицині, допомагають здійснювати автоматичний контроль у складних технічних системах, вчасно діагностувати їх та проводити ремонтно-відновлювальні роботи і т.д.

Основні здобутки при заміни людини в задачах розпізнавання зводяться до таких:

– звільнення людини від рутинних одноманітних операцій з метою розв'язку інших важливих задач;

– підвищення якості виконуваних робіт;

– зменшення витрат часу на розв'язок задач.

Машинне навчання – це область штучного інтелекту, в якій використовуються статичні методи, щоб дати змогу комп’ютерній програмі навчатися за допомогою конкретних даних, не будучи явно запрограмованою. Машинне навчання розвивається кожного дня та набуває все більшої популярності. Зараз його використовують в дуже багатьох сферах людської діяльності. Наприклад, онлайн-перекладачі, розпізнавання тексту, пошук інформації, фільтрація поштових скриньок, реклама на основі переглядів користувачів, чат-боти, забезпечення для керування автомобілями (система автопілоту) тощо.

Справжнім успіхом в машинному навчанні стали голосові помічники з можливістю розпізнавати мовлення, наприклад, Apple Siri.

Саме з них почалось активне використання машинного навчання в повсякденному житті. Зараз нейронні мережі виконують і складніші завдання, такі як керування автомобілем.

Ця робота має на меті огляд нейронних мереж та більш детально зупинитись на мережі, яка моделює процеси людського сприйняття, а саме спайкінгова нейронна мережа та її використання для розпізнавання графічних об’єктів, та створення застосунку, який продемонструє роботу спайкінгової нейронної мережі на прикладі розпізнавання рукописних цифр.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Розробка прикладних інтелектуальних інформаційних технологій та систем» та плану наукової та навчально-методичної роботи кафедри.

Мета і завдання досліджень. Метою магістерської кваліфікаційної роботи є підвищення достовірності розпізнавання рукописних цифр за рахунок використання спайкінгової нейронної мережі.

Для досягнення мети розробки необхідно виконати такі задачі:

- провести аналіз предметної області розпізнавання рукописних цифр;
- розглянути існуючі методи розпізнавання рукописних цифр та обрати й обґрунтувати вибір методу, який задовольняє мету даної магістерської кваліфікаційної роботи;
- розробити інформаційну технологію нейромережевого медичного діагностування;
- обгрунтувати вибір типу нейромережі та розробити її структуру;
- виконати програмну реалізацію запропонованої інформаційної технології;

− провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об’єкт дослідження – процес комп’ютеризованого розпізнавання рукописних цифр за допомогою спайкінгової нейронної мережі.

Предмет дослідження – інформаційна технологія та програмні засоби розпізнавання рукописних цифр за допомогою спайкінгової нейронної мережі та достовірність їх роботи.

Методи дослідження. У роботі використані наступні методи наукових досліджень: системного аналізу, теорії нейронних мереж для реалізації інформаційної технології, методи математичної статистики для розробки процесу розв'язання задачі розпізнавання рукописних цифр та обрахунків результатів експериментів із програмним засобом, об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів.

1. Набула подальшого розвитку інформаційна технологія розпізнавання рукописних цифр, яка відрізняється використанням спайкінгової нейронної мережі, що дозволило підвищити достовірність програмних засобів розпізнавання рукописних цифр.

Практичне значення одержаних результатів полягає в тому, що на основі проведених досліджень розроблено програмне забезпечення розпізнавання рукописних цифр.

Запропонована інформаційна технологія сприяє підвищенню достовірності програмних засобів розпізнавання рукописних цифр, зокрема:

- розроблено алгоритм роботи програмного забезпечення розпізнавання рукописних цифр;
- розроблено програмні засоби для розпізнавання рукописних цифр.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю використання математичного апарату методів

дослідження, експериментальними дослідженнями тестування програмної реалізації інформаційної технології розпізнавання рукописних цифр. Адекватність розроблених математичних моделей підтверджується результатами експериментальних досліджень.

Особистий внесок здобувача. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У працях, написаних у співавторстві, здобувачу належать: аналіз процесу розпізнавання рукописних цифр та методів підвищення достовірності програмних засобів розпізнавання рукописних цифр [1].

Апробація результатів роботи. Результати роботи були апробовані на Міжнародній науково-практичній Інтернет-конференції студентів, аспірантів та молодих науковців «МОЛОДЬ В НАУЦІ: ДОСЛІДЖЕННЯ, ПРОБЛЕМИ, ПЕРСПЕКТИВИ (МН-2024)» (м. Вінниця, Україна, 2023-2024) [1].

Публікації. За результатами досліджень опубліковано одні тези доповіді на науково-технічній конференції [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗПІЗНАВАННЯ ОБРАЗІВ

1.1 Постановка задачі

Останніми роками зростає інтерес до того, як спайкінгові нейронні мережі (SNN) можна використовувати для виконання складних обчислень або вирішення завдань розпізнавання образів. Однак розробка SNN, яка використовує біологічно подібні механізми (особливо для вивчення нових образів), залишається складним завданням, оскільки більшість таких архітектур SNN покладається на навчання в мережі на основі частоти спайків та подальше перетворення в SNN. Ми представляємо SNN для розпізнавання цифр, яка базується на механізмах з підвищеною біологічною правдоподібністю, тобто: синапси на основі провідності замість синапсів на основі струму; пластичності, що залежить від часу спайку, із залежною від часу зміною ваги; латеральним гальмуванням та адаптивним спайкінговим порогом. На відміну від більшості інших систем, ми не використовуємо навчальний сигнал і не представляємо в мережі жодних міток класів. Використовуючи схему навчання без учителя, розроблена архітектура досягає 95% достовірності за тестом MNIST, що краще, ніж попередні реалізації SNN без учителя. Той факт, що ми не використовували специфічні для предметної області знання, вказує на загальну застосовність цього дизайну мережі. Крім того, продуктивність нашої мережі добре масштабується з кількістю використовуваних нейронів і показує подібну продуктивність для чотирьох різних правил навчання, що вказує на надійність повної комбінації механізмів, що свідчить про застосовність у гетерогенних біологічних нейронних мережах.

Практична задача – розробка методу розпізнавання рукописних цифр за їх зображенням.

Вхідними даними для програми будуть зображення у форматі JPEG, що містить цифру, без зайвих об’єктів, бажано з однорідним фоном.

Вихідними даними для програми є номер класу цифри та ймовірність того, що такий висновок є істиною.

Таким чином, інформаційна технологія для розпізнавання рукописних цифр отримує на вході зображення, і визначає номер класу, до якого відноситься цифра.

1.2 Огляд методів розпізнавання зображень

При реалізації операції розпізнавання ключовим елементом в матмоделі є вибір інтегрального критерію подібності, який на основі знання ознак конкретного об'єкта дозволяв би кількісно визначати ступінь його схожості з узагальненими образами класів.

Вимога стійкості до шуму означає, що результат застосування інтегрального критерію до образу, що складається тільки з білого шуму, повинен бути рівним нулю. Це означає, що як інтегральний критерій може бути використана функція, яка застосовується при визначенні самого поняття "білий шум", тобто згортка, скалярний добуток, кореляція [2].

Основою програми є нейромережа, що моделює штучний інтелект, і яка дозволяє навчати програму і запам'ятовувати образи цифр, для подальшого їх вірного розпізнавання при використанні програми.

Розробка програми розпізнавання цифр є актуальною, що обумовлено популярністю програмних продуктів подібного типу на ринку та їх широкого застосування в КПК, ноутбуках, смартфонах і т.п .

Задачі розпізнавання характеризуються такими рисами [3].

1. Це інформаційні задачі, що вирішуються у два етапи:

а) перетворення вхідних даних до виду, зручного для розпізнавання;

б) власне розпізнавання - визначення приналежності об'єкта до певного класу).

2. Можливість введення поняття аналогії або подібності об'єктів і формулювання правил, на підставі яких об'єкт відносять до того самого класу або в різні класи.

3. Можливість оперувати набором прецедентів - прикладів, класова приналежність яких відома і які у вигляді формалізованих описів можуть пред'являтись алгоритму розпізнавання для налаштування на задачу в процесі навчання.

4. Складність побудови формальних теорій та застосування класичних математичних методів (часто відсутня інформація для точної матмоделі або виграш від використання моделі та математичних методів набагатоменший за витрати).

Розглянемо основні методи розпізнавання образів та оберемо найбільш вигідний для нашої задачі.

1.2.1 Метод розпізнавання по шаблонах

Програмне забезпечення оптичного розпізнавання символів (Optical Character Recognition - OCR) зазвичай працює з великими растровими зображеннями сторінки зі сканера [3]. При цьому багато систем мають шаблони, створені для різних написань. Після кількох розпізнаних слів програмне забезпечення визначає шрифт і шукає подібні пари тільки для цього шрифту. У інших випадках програмне забезпечення застосовує чисельні значення пропорцій символу, щоб визначати новий шрифт. Це може покращити ефективність розпізнавання.

Такий підхід передбачає створення шаблону для кожного шрифту. Наприклад, програма TypeReader може розпізнавати 2100 різних варіантів написань символів.

1.2.2 Метод структурного підходу

Система OCR Caere OmniPage Professional використовує метод, заснований на знаходженні загальних специфічних особливостей написання символів [4].

Ця система OCR містить 100 різних алгоритмів для розпізнавання 100 різних символів: великих і малих літер від «A» до «Z», символи цифр і знаків пунктуації. Кожен з цих алгоритмів шукає «особливості» написання типу «островів», «півостровів», прямих відбитків, точок і дуг. Експертні системи розглядають горизонтальні і вертикальні проекції зображень символу і акцентують увагу на основні особливості у кривих, підраховуючи в них число чорних пікселів.

На жаль, нечіткий текст є специфічною проблемою для цих структурних методів, оскільки відсутній піксель може розділяти довгий штрих або криву, а додаткова пляма може закривати петлю.

1.2.3 Метод контекстного розпізнавання

Людина здатна швидко розрізнити на папері літери "h" та "b" ще й тому, що вона знає контекст слова, в якому знаходяться ці літери. Через цю причину програмне забезпечення OCR має словники на допомогу алгоритмам розпізнавання [4]. Словники допомагають в багатьох випадках, але відмовляють, коли, програма стикається з іменами власними, які не входять до словника. Корпорація Xerox має одну з найбільш складних програм з використанням методу контекстного аналізу.

Таким чином, метод розпізнавання в загальному випадку полягає у послідовному висуненні та перевірці гіпотез, причому порядок їх висування керується закладеними в програму знаннями про досліджуваний об’єкт і результатами перевірки попередніх гіпотез. Основна вимога до попередньої обробки зображень – це не втратити у

вхідному образі суттєву інформацію. Так як для виділення цілого потрібні його частини, а для знаходження частин потрібне ціле, то цілісний процес розпізнавання може відбуватися тільки в рамках гіпотези про сприйняття об'єктів - в цілому. Якщо якість програми має наближатися до якості розпізнавання тексту людиною, то, скоріше за все, вона може використовувати метод, "запозичений" у людини-читача.

Під час читання людина розпізнає літери, сприймає слова, зв'язує їх у синтаксичні конструкції і розуміє зміст прочитаного. Всі ці процеси відбуваються одночасно, впливають один на іншого, а остаточне рішення приймається на основі повного враховування результатів всіх процесів. Цілісний опис класу об'єктів сприйняття повинен задовольняти двом властивостям: по-перше, всі об'єкти певного класу мають задовольняти цьому опису, по-друге, жоден з об'єктів інших класів не повинен задовольняти цьому опису. Звичайно гіпотези висуваються послідовно, об'єднуються в перелік і сортуються в залежності від попередньої оцінки гіпотези. Остаточний вибір гіпотези здійснюється в рамках контексту, із залученням, крім іншого, додаткових джерел знань.

1.2.4 Метод на основі корекції помилок

Підхід у побудові шаблонів в деяких програмах налаштований так, щоб знайти базові пікселі у великій кількості документів. Розірвані символи знаходяться автоматично, тому що програма аналізує пікселі у певному довірчому інтервалі, які можуть зникати у зашумлених зображеннях. Програма лише має знайти місця, в яких відбулося розділення або розмивання символів [4].

Всі програми оптичного розпізнавання символів мають опцію перевірки помилок користувачем, при якій використовується одночасне початкове зображення у вікні екрану, тобто: немає потреби заглядати у паперову версію. Наприклад,, програма TypeReader переглядає рядки

тексту при пошуку помилок клавішею табуляції (програма поміщає відсікання зображення за текстом таким чином, щоб користувач міг порівнювати текст із зображенням без переміщення погляду).

Одна частина програми розпізнавання може функціонувати на основі нейронної мережі, а інша її частина може використовувати переваги клітинних автоматів, і, нарешті, третя частина програми - видавати результат на основі накопиченої статистики. Тому комплекс методів, безперечно, дозволить отримати більш кращий результат, ніж окремо взятий метод.

1.2.5 Метод на основі нейронних мереж

Для розпізнавання символів доволі широко використовують штучні нейронні мережі [5]. Методи, які використовують нейромережі для розпізнавання символів, часто будуються таким чином. Зображення символу (растр), який є вхідним для розпізнавання, масштабується до деякого стандартного розміру, наприклад, використається растр 16x16 пікселів.

Значення яскравості у пікселях нормалізованого растру використовуються, як значення входів нейронної мережі. Кількість вихідних параметрів нейромережі обирається рівним числу розпізнаваних символів. Результатом розпізнавання є шаблонний символ, якому відповідає найбільше значення вихідного вектора нейромережі. Підвищення достовірності таких алгоритмів пов'язано, як правило, або з пошуком більш інформативних ознак символів, або з ускладненням структури самої нейромережі.

В якості вхідних параметрів нейромережі, замість значень яскравості у пікселях нормалізованого растра можуть використовуватися значення перепаду яскравості. Ці вхідні параметри дозволяють краще виділяти межі символу.

Одним із широко використовуваних прийомів підвищення достовірності розпізнавання є одночасне використання кількох різних модулів розпізнавання і наступне об'єднання отриманих результатів (напр., шляхом голосування). При цьому важливим є те, щоб методи, використовувані цими модулями, були якомога більш незалежними. Це може бути досягнуто як шляхом використання розпізнавальних модулів, що працюють за принципово різними алгоритмами розпізнавання, так і шляхом спеціального підбору навчальних даних.

Метод на основі нейронних мереж є найбільш перспективним, оскільки використовує сучасні технології штучного інтелекту і можливість навчання на прикладах. Тому для застосування у даній роботі оберемо саме метод на основі штучних нейронних мереж.

1.3 Обгрунтування вибору аналогу до програми розпізнавання рукописних цифр

Наразі є дуже багато компаній та застосунків, які використовують технології розпізнавання тексту. Напевно, найвідомішою такою компанією є «ABBYY» та їх застосунок «FineReader». На даний момент застосунок розпізнає текст на 192 мовах світу і має вбудовано систему перевірки орфографії для 48 з них. На офіційному сайті «ABBYY FineReader» вказано, що за допомогою їх технологій можна розпізнати текст з файлу або за допомогою мобільного додатку отримати текст з камери смартфону.

ABBYY FineReader – універсальне рішення для роботи з паперовими та PDF файлами будь-якого типу. Поєднання системи оптичного розпізнавання тексту (OCR — Optical Character Recognition) та інструменти для роботи з PDF документами дозволяє ефективно вирішувати різноманітні задачі розпізнавання [6].

Як ABBYY FineReader розпізнає текст? Спочатку він аналізує структуру. Розділяє документ на елементи такі як: таблиці, блоки тексту. Отримані елементи поділяються на слова, які в свою чергу поділяються на символи. Потім символи порівнюються зі зразками програми та будуються гіпотези щодо того який це символ може бути.

Переваги: висока достовірність розпізнавання; велика кількість мов; стійкість до поганої якості зображення.

Недоліки: не розпізнає рукописний текст, безкоштовна пробна версія лише на 15 днів; висока ціна програми.

Другою широко відомою програмою OCR є CuneiForm – безкоштовна програма для розпізнвання текстової інформації із зображень. Достовірнність розпізнавання на порядок нижче, ніж у ABBYY FineReader. Але як для безкоштовної утиліти, функціональні можливості все-таки широкі. CuneiForm розпізнає блоки тексту, текстові зображення і навіть таблиці. Також зчитуванню піддаються і неразліновані таблиці. Програма може розпізнавати і зберігати шрифт і кегль тексту. У базі даних шрифтів міститься більшість використовуваних шрифтів друкованих літер. Підтримується також розпізнавання тексту друкарських машин. Для забезпечення достовірності до процесу розпізнавання залучаються спеціальні словники, які поповнюють свій словниковий запас із вже оброблених документів.

Переваги: безкоштовність; використання словників для перевірки правильності розпізнавання тексту; можливість сканування тексту з копій поганої якості.

Недоліки: не розпізнає рукописний текст, відносно невелика достовірність; мала кількість підтримуваних мов.

Як аналог оберемо програму систему MyScript Stylus [7], яка дозволяє здійснювати оптичне розпізнавання рукописного тексту. Головне вікно цієї програми представлено на рис.1.1.

MyScript Stylus – це програмна система для розпізнавання рукописного тексту. Вводити текст можна за допомогою мишки або планшета. Програма розпізнає текст відповідно до технології, яка застосовується в КПК, і може використовуватися не тільки стандартну клавіатуру (наприклад, якщо комп'ютер використовується як термінал для вводу/виводу інформації або як платіжний термінал). MyScript Stylus можливо закріпити за деякою програмою, і весь текст, який розпізнається, буде передаватися цій програмі, якби текст вводиться стандартним способом. Програма MyScript Stylus підтримує 26 мов.

Рисунок 1.1 – Скріншоти програми MyScript Stylus

MyScript Стилус використовує потужний і інтерактивний метод введення, який заміняє клавіатуру у всіх додатках, які вимагають введення тексту (веб-браузери, календарі, електронна пошта, текстові процесори, таблиці і багато іншого).

MyScript Стилус перетворює рукописні символи в цифровий текст в реальному часі. Крім цього, програма має модуль калькулятора, який

розпізнає математичні формули та перетворювати їх символи і цифри і видає результат в режимі реального часу.

Програма MyScript стилус призначена для спільного використання з такими пристроями введення, як: сенсорний екран ПК, електронні книги, планшетні ПК, нетбуки, інтерактивні дошки, цифрові ручки, мобільні Інтернет-пристрої.

Головним недоліком програми-аналогу є невелика достовірність розпізнавання (приблизно 84%). Ще одним недоліком даного програмного продукту є його порівняно велика ціна, що робить його недоступним для більшості звичайних користувачів. Як і програма, що розробляється, програма-аналог також написана під операційну систему Windows.

1.4 Висновок до розділу 1

У розділі було розглянуто постановку задачі розпізнавання рукописних цифр, проведено огляд відомих методів розпізнавання зображень, які можна використовувати для поставленої задачі. Обґрунтовано доцільність використання нейронних мереж, зокрема спайкінгових нейронних мереж для розпізнавання рукописних цифр. Було проаналізовано різні програмні реалізації розпізнавання рукописних цифр та обрано аналог, головним недоліком якого є невисока достовірність розпізнавання рукописних цифр, що ставить мету дослідження – підвищення достовірності розпізнавання рукописних цифр.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР СПАЙКІНГОВОЮ НЕЙРОННОЮ МЕРЕЖЕЮ

2.1 Обгрунтування вибору спайкінгової нейромережі

Останнє десятиліття стало свідком великого успіху глибоких нейронних мереж у різних областях. Однак глибокі нейронні мережі дуже ресурсомісткі з точки зору споживання енергії, вимог до даних і високих обчислювальних витрат. Оскільки останнім часом зростає потреба в автономності машин у реальному світі, наприклад, безпілотних транспортних засобів, безпілотних літальних апаратів і роботів, що співпрацюють, активно досліджується використання глибоких нейронних мереж у цих програмах. У цих програмах енергоефективність і обчислювальна ефективність є особливо важливими через потребу у відповідях у реальному часі та обмежене енергопостачання. Багатообіцяюче рішення для цих раніше нездійсненних застосувань нещодавно було запропоновано біологічно правдоподібними нейронними мережами. Мета спайкінгових нейронних мереж — подолати розрив між нейронаукою та машинним навчанням, використовуючи біологічно реалістичні моделі нейронів для виконання обчислень.

Останнє десятиліття стало свідком зростання можливостей штучних нейронних мереж (ШНМ) від багатошарового персептрона першого покоління (MLP) до багатьох найсучасніших методів глибоких нейронних мереж другого покоління (ГНМ). Це досягнення значною мірою залежить від великої кількості анотованих даних і широкої доступності високопродуктивних обчислювальних пристроїв, а також графічних процесорів (GPU) загального призначення. Незважаючи на цей великий прогрес, ШНМ все ще відстають від біологічних нейронних мереж з точки

зору енергоефективності та можливостей онлайн-навчання. Було зроблено багато спроб зменшити енергоспоживання традиційних моделей глибокого навчання. Щоб знайти більш компактні мережі, які можуть досягти подібної продуктивності з набагато меншою складністю та меншою кількістю параметрів порівняно з вихідною мережею, було розроблено багато методів, таких як квантування, скорочення та дистиляція знань. Квантування перетворює ваги та вхідні дані мережі в цілі типи, що робить загальні операції легшими, ніж операції з плаваючою комою. Під час скорочення з'єднання мережі періодично видаляються під час або після навчання. Щоб стиснути нейронну мережу без зниження продуктивності, дистиляція знань передає складні знання, отримані важкою мережею під назвою «вчитель», до легкої мережі під назвою «учень».

Незважаючи на те, що історично вивчення мозку надихнуло на створення ШНМ/ГНМ, вони принципово відрізняються за структурою, нейронними обчисленнями та правилами навчання порівняно з біологічною нейронною мережею. Це спостереження веде до спайкінгових нейронних мереж (SNN), які часто називають третім поколінням нейронних мереж, які можуть стати проривом у вузьких місцях ШНМ. Варто згадати використання SNN на нейроморфних апаратних засобах, таких як TrueNorth, Loihi, SpiNNaker, NeuroGrid тощо, і є перспективним підходом до проблеми споживання енергії. У SNN, наприклад, як і у біологічних нейронних мережах, нейрони спілкуються один з одним за допомогою дискретних електричних сигналів, які називаються спайками, і працюють у безперервному часі.

Завдяки своїй функціональній схожості з біологічними нейронними мережами, SNN можуть охоплювати розрідженість, виявлену в біології, і дуже сумісні з часовим кодом. Хоча SNN все ще відстають від DNN з точки зору їх продуктивності, розрив зникає для деяких завдань, тоді як

SNN зазвичай потребують набагато менше енергії для роботи. Однак SNN все ще важко навчити загалом, головним чином через їх складну динаміку нейронів і недиференційований характер спайкових операцій. Порівняння між біологічними нейронними мережами, ШНМ та СМН наведено в таблиці 1. BP – це зворотне поширення.

Таблиця 1 - Порівняння властивостей між біологічними нейронними мережами, ШНМ і СНМ.

Властивості	Біологічні НМ	ШНМ	СНМ
Представлення інформації	Спайки	Скаляри	Спайки
Парадигма навчання	Синаптична пластичність	Зворотне поширення	Пластичність/ Зворотне поширення
Платформа	Мозок	VLSI	Нейроморфний VLSI

Нейрони є основними робочими одиницями нервової системи, які обробляють інформацію, поширюючи електрохімічні сигнали через потенціали дії. Нейрони не є електрично нейтральними або позаклітинною рідиною через присутність у них іонів. Іони постійно входять і виходять з клітини через мембрану, яка може динамічно змінювати свою електричну проникність за допомогою зовнішніх електрохімічних сигналів. Потік іонів, що входять у клітину та виходять із неї, спричиняє віртуальний струм, що протікає через мембрану, переважно пов’язаний з іонами Na^+, K^+ та Cl^-.

На рис. 2.1 показано типову структуру нейрона з чотирма основними компонентами: дендритами, сомою, аксоном і синапсом. Дендрити — це короткі нервові закінчення, які можна розглядати як вхідні дані нейрона.

Вони перетворюють хімічні сигнали, що передаються нейромедіаторами, що вивільняються з пресинаптичного нейрона, в електричні сигнали. Сома — це тіло клітини, де інтегровані мембранні потенціали, що поширюються від синаптичних входів, що в кінцевому рахунку визначає, чи запускає постсинаптична клітина потенціали дії перед передачею до аксона. Така взаємодія впливів називається нейронною інтеграцією. Аксон несе потенціал дії до інших нервових клітин. Для швидкого перенесення потенціалу дії на великі відстані без ослаблення деякі аксони вкриті мієліновою оболонкою.

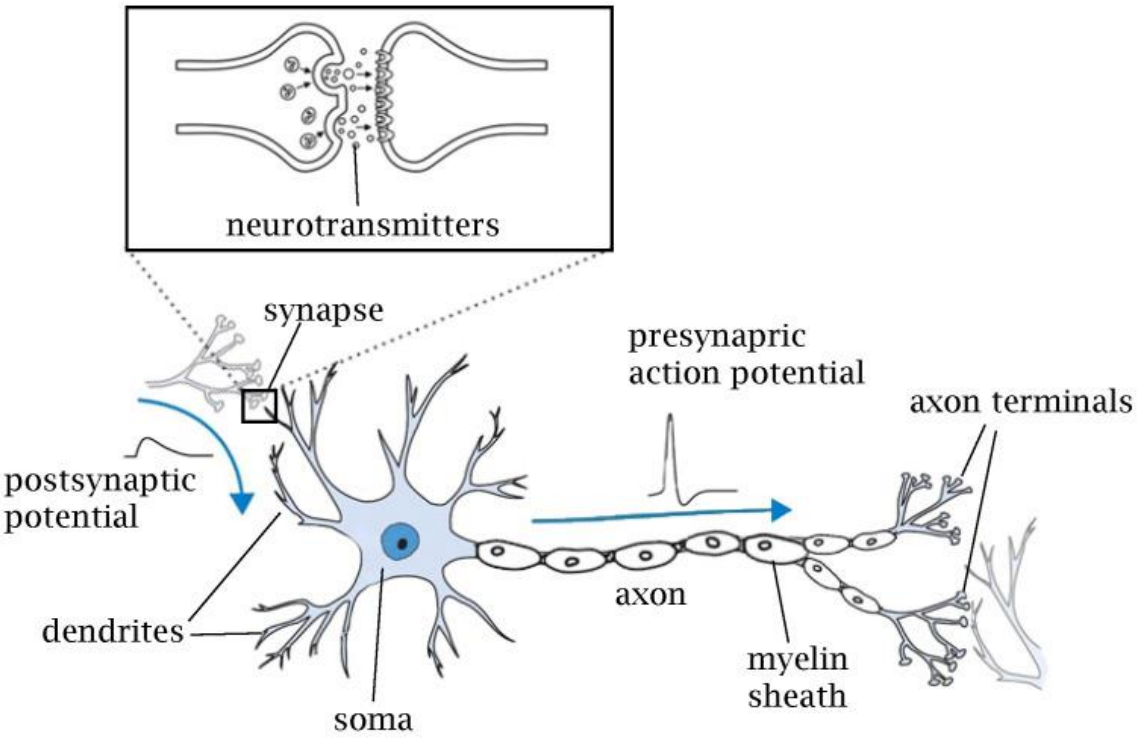

Рисунок 2.1 – Типова будова біологічного нейрона та синапсу

Синапси — це контактна структура для передачі інформації, яка з’єднує нейрони в нейронній мережі. Синапси можна умовно розділити на хімічні та електричні. У хімічних синапсах немає прямого контакту між пре- і постсинаптичними нейронами. Сигнал від пресинаптичного нейрона передається через нейромедіатори, що містяться в синаптичних гранулах,

випущених у синаптичну щілину. Нейромедіатори зв’язуються з рецепторами в постсинаптичній клітині, безпосередньо змінюючи потенціал мембрани або активуючи внутрішньоклітинні вторинні месенджери для передачі інформації. Цей тип передачі повільний, але посилює сигнал і може подовжити вплив вхідного стрибка. Хімічні синапси можна підрозділити на збудливі та гальмівні синапси. Збудливі синапси — це синаптичні зв’язки, які деполяризують постсинаптичні клітини через синаптичну передачу та сприяють запуску потенціалів дії. Гальмівні синапси - це синаптичні зв'язки, які гіперполяризують постсинаптичні клітини шляхом синаптичної передачі та гальмують розвиток потенціалів дії. Глутамат і ГАМК є найбільш поширеними збудливими і гальмівними нейромедіаторами відповідно; іонотропними рецепторами для глутамату є AMPA та NMDA, а для GABA – GABAA та GABAB. Електричні синапси, з іншого боку, є структурами, які передають заряди мембранного потенціалу безпосередньо до наступного нейрона через щілинні з’єднання на контактній мембрані. Цей вид зв’язку є дуже швидким, оскільки в ньому немає хімічних реакцій; однак немає збільшення амплітуди сигналу, як у хімічних синапсах.

Електричний потенціал всередині клітини по відношенню до зовнішньої частини клітини називається мембранним потенціалом. Мембранний потенціал можна отримати за допомогою рівняння Гольдмана–Ходжкіна–Каца, яке враховує відносну проникність плазматичної мембрани для кожного іона.

$$v_m = \frac{RT}{F} \ln \frac{P_K[K^+]_{out} + P_{Na}[Na^+]_{out} + P_{Cl}[Cl^-]_{in}}{P_K[K^+]_{in} + P_{Na}[Na^+]_{in} + P_{Cl}[Cl^-]_{out}} \quad (2.1)$$

де R — універсальна газова стала, T — абсолютна температура 310,15 (K) при температурі тіла людини (37 [°C]), F — стала Фарадея (=96 485 (C·моль−1)), (A)out це позаклітинна концентрація іона A, а (A)in — внутрішньоклітинна концентрація іона A, а PA — проникність мембрани для іона A, а для типового нейрона в стані спокою відомо, що PK:PNa:PCl=1 :0,04:0,45. Навпаки, приблизна відносна проникність на піку типового нейронального потенціалу дії становить PK:PNa:PCl=1:12:0,45.

Потенціал мембрани спокою. Завдяки дії низки білків іони постійно рухаються в клітину та виходять з неї. Хоча приплив іонів не припиняється, перенесення заряду стає, очевидно, нерухомим, коли загальний заряд іонів, що витікають, і загальний заряд іонів, що втікають, за одиницю часу стають однаковими. Мембранний потенціал спокою клітини визначається сумарним потоком іонів через канали «витоку», які відкриті в стані спокою. Базуючись на відносній проникності мембрани для типового нейрона в стані спокою, ми можемо розрахувати мембранний потенціал спокою Em наступним чином:

$$\begin{aligned} E_m &= \frac{RT}{F} \ln \frac{5.5P_K + 135 \times 0.04P_K + 9 \times 0.45P_K}{150P_K + 15 \times 0.04P_K + 125 \times 0.45P_K} \\ &= -70.15\,[mV] \end{aligned} \tag{2.2}$$

Оскільки реверсивний потенціал для іонів Cl− зазвичай близький до потенціалу мембрани спокою, іон Cl− зазвичай ігнорується при обговоренні потенціалу мембрани спокою нейрона.

Потенціал дії. Коли виникає потенціал дії, натрієві канали на аксоні відкриваються, і іони Na+ можуть вільно рухатися в клітинну мембрану та з неї. Мембранний потенціал коливається відповідно до зворотного потенціалу іона Na+. Тоді натрієвий канал інактивується та закривається, а

тепер відкривається калієвий канал, який залежить від потенціалу. Тепер мембранний потенціал опускається назад до потенціалу реверсування іона K+ і виходить за межі мембранного потенціалу спокою Em.

$$\begin{aligned} v_{peak} &= \frac{RT}{F} \ln \frac{5.5P_K + 135 \times 12P_K + 9 \times 0.45P_K}{150P_K + 15 \times 12P_K + 125 \times 0.45P_K 0} \\ &= 38.43\ [mV] \end{aligned} \tag{2.3}$$

Традиційні ШНМ. Нейрон, заснований на частоті, моделює активність біологічного нейрона лише за макроскопічною ознакою, частотою імпульсів r, незалежно від зміни мембранного потенціалу чи часу спайку. Перший штучний нейрон з частотним кодуванням, відомий як формальний нейрон або пороговий логічний блок, був запропонований на початку розвитку ШНМ. На основі формального нейрона було розроблено персептрон, який використовує ступінчасту функцію Хевісайда як функцію активації. Ці нейрони першого покоління дають на виході двійкові сигнали, коли сума вхідних сигналів досягає порогу нейрона. Пізніше ця концепція була розширена для використання неперервних функцій активації, включаючи сигмоїду або функцію гіперболічного тангенса, для роботи з аналоговими входами та виходами. Отже, це дозволило навчати нейронну мережу за допомогою потужного алгоритму зворотного поширення, який використовує градієнтний спуск. Через доведену здатність достатньо великої нейронної мережі штучних нейронів як завгодно добре апроксимувати будь-яку аналогову функцію (універсальна теорема про наближення стверджує, що мережа прямого поширення з одним прихованим шаром із кінцевою кількістю нейронів може апроксимувати безперервні функції за припущень на неполіноміальній функції активації; сигмоїдальна функція активації та ReLU також доведено, що відповідають теоремі універсальної

апроксимації), штучні нейронні мережі широко використовуються як потужний інструмент обробки інформації в машинному навчанні. Загалом, частотна модель нейрона з дискретним часом може бути сформульована як $r = \sigma(\sum_i w_{i,j} x_j)$ і зазвичай групується разом для ефективності обчислень:

$$\mathbf{r} = f(\mathbf{W}\mathbf{u} + \mathbf{b}) \tag{2.4}$$

де $\mathbf{u} \in \mathbb{R}^{N_{pre}}$ — швидкість активації пресинаптичних нейронів, $\mathbf{r} \in \mathbb{R}^{N_{post}}$ — частота активації постсинаптичних нейронів, $\mathbf{W} \in \mathbb{R}^{N_{post} \times N_{pre}}$ — вагова матриця, яка представляє синаптичну силу між пре- та постсинаптичними нейронами , $\mathbf{b} \in \mathbb{R}^{N_{post}}$ – член зміщення, а f(·) – нелінійна функція активації.

Зараз функція активації Rectified Linear Unit (ReLU) та її варіанти зазвичай використовуються як нелінійність, оскільки вони, як правило, демонструють кращу продуктивність збіжності, ніж сигмоподібна функція активації. Таке формування групи частотних нейронів часто називають повністю зв’язаним шаром. Сучасна архітектура нейронних мереж об’єднує варіант цього рівня для створення дуже глибоких мереж нейронів, які часто називають глибокими нейронними мережами (DNN). Нейронні мережі зазвичай називають глибокими, якщо вони мають принаймні два прихованих шари обчислення нелінійних перетворень вхідних даних. Одним із часто використовуваних будівельних блоків DNN є згортковий шар. Згортковий шар — це окремий випадок повністю зв’язаного шару, який реалізує розподіл ваг для обробки даних, що мають відому сіткову топологію, наприклад, зображень. Завдяки такому індуктивному зміщенню згорткові нейронні мережі (CNN) можуть більш розумно використовувати просторову кореляцію сигналу. Репрезентативні властивості ранніх шарів у CNN подібні до властивостей відповіді

нейронів у первинній зоровій корі (V1), яка є першою корковою областю в зоровій ієрархії мозку приматів. CNN володіють двома ключовими властивостями, які роблять їх надзвичайно корисними при застосуванні для зображень: просторово розподілені ваги та просторове об'єднання. Цей тип мережі вивчає функції, які не змінюються, тобто фільтри, які корисні для всього зображення (через те, що статистика зображення є стаціонарною). Рівні об'єднання відповідають за зменшення чутливості вихідного сигналу до незначного вхідного зсуву та спотворень і збільшення поля прийому для наступних шарів. Починаючи з 2012 року, одним із найпомітніших результатів у глибокому навчанні є використання CNN для досягнення значного вдосконалення проблеми класифікації зображень. На основі цього технологічного прориву в класифікації зображень були запропоновані різні вдосконалення для мережевих архітектур у моделях бачення. Хоча ШНМ були надзвичайно успішними в багатьох програмах, включаючи виявлення об'єктів, сегментацію зображення і розпізнавання дій, вони все ще обмежені в тому, як вони мають справу з часовою інформацією.

Моделі штучних нейронних мереж. Здатність одночасно реєструвати активність кількох клітин привела до ідеї, що різниця в часі між спайками в різних нейронах і час самого спайку можуть мати функціональне значення. Оскільки частотна модель не може впоратися з проблемою цієї перспективи, була досліджена модель, що описує час появи спайків і зміну підпорогового мембранного потенціалу. Модель, яка обробляє генерацію таких спайків, відрізняється від частотної моделі та називається спайкінговою моделлю. Такі моделі нейронів зазвичай описуються у формі звичайних диференціальних рівнянь. На рис. 2.2 зображено відмінності між біологічним нейроном, штучним нейроном і спайкінговим нейроном.

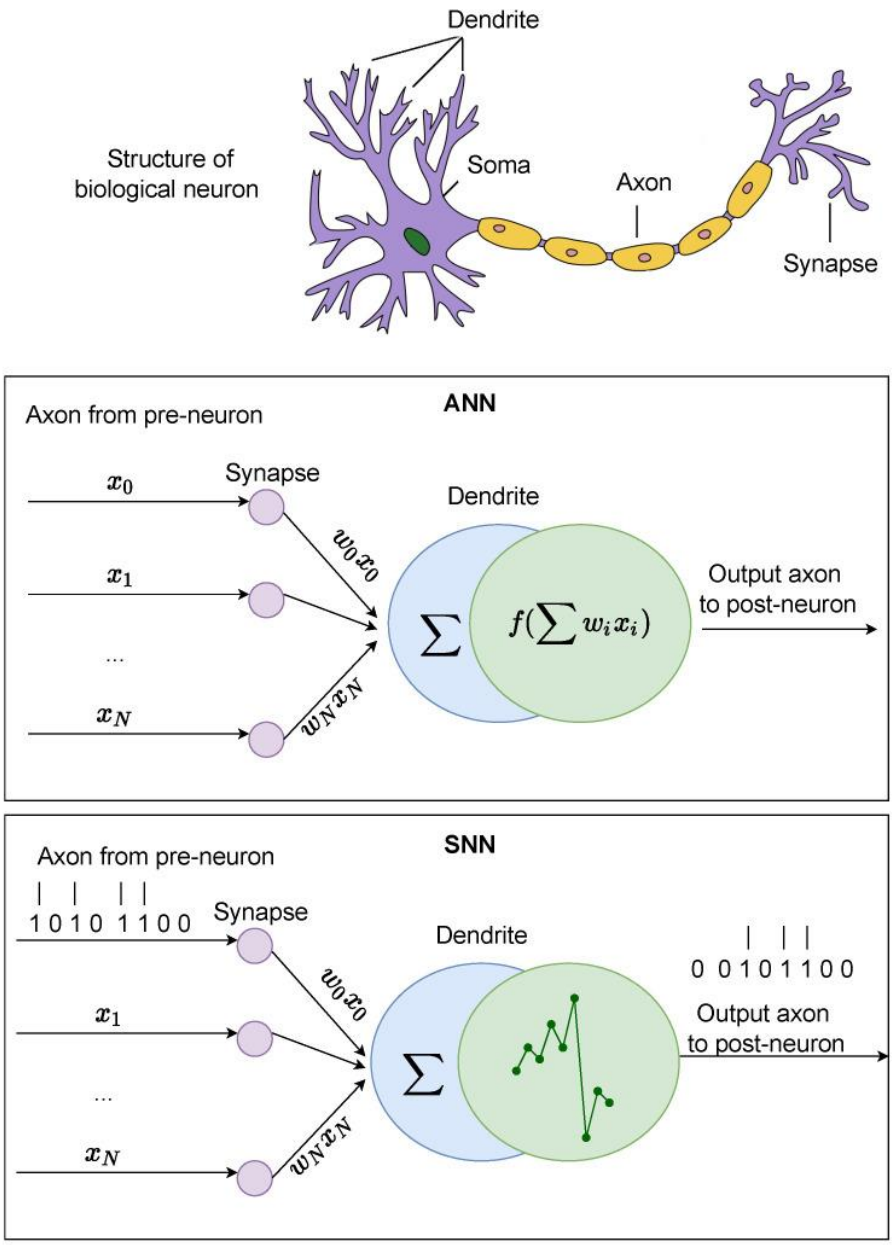

Рисунок 2.2 – Порівняння біологічного нейрона, штучного нейрона та спайкінгового нейрона

Спайкінгові нейромережі були рушійним фактором у розвитку багатьох сучасних систем комп’ютерного зору та інших методів обробки сигналів. Застосування спайкінгових нейромереж поступово стає переважати в комп'ютерному зорі, де обробляються дані, що складаються з часової інформації, або де метою є економія обчислювальних ресурсів. Частіше СНМ використовуються для обробки рухомих зображень з

камери подій або датчика LiDAR, чиї дані мають важливе значення в часовому вимірі. Останній випадок часто зосереджується на перетворенні ШНМ у СНМ, щоб моделі глибоких нейронних мереж могли охоплювати енергоефективні операції нейроморфного обладнання.

Хоча деякі дослідження показали, що СНМ можна використовувати для класифікації зображень у великих наборах даних, таких як ImageNet, більшість застосувань СНМ все ще обмежується менш складними наборами даних, такими як MNIST, N-MNIST і N-Caltech101. Однією з головних причин обмеженої сфери застосування є складна динаміка та недиференційовані операції спайкінгових нейронів. Нещодавно деякі дослідження застосували SNN для завдань виявлення об’єктів [8,9], показуючи порівняльні результати з DNN, вимагаючи набагато менше енергії та обчислень. Після успіху методів перетворення ШНМ у СНМ у задачах класифікації зображень і виявлення об’єктів, розробники використали SiamFC і представило SiamSNN, сіамську спайкінгову мережу для відстеження об’єктів. Нещодавно компанія SNN на базі UNet [8] використала фреймворк Nengo для перетворення спрощеної мережі U-Net у спайкінгову мережу для розгортання на нейроморфному чіпі Intel Loihi. Модель SNN на основі UNet реалізована за допомогою двох фреймворків: TensorFlow і NengoDL.

Все вище сказане свідчить про перспективність застосування спайкінгових нейронних мереж для розпізнавання зображень загалом і для розпізнавання рукописних цифр зокрема.

2.2 Розробка архітектури спайкінгової нейромережі

Щоб змоделювати динаміку нейронів, була обрана модель IF-нейрона [10] («integrate-and-fire»). Напруга мембрани *B* IF-нейрона описується як:

$$\tau \frac{dB}{dT} = (E_{rest} - B)C g_e(E_{exc} - B)C g_i(E_{inh} - B) \quad (2.5)$$

де E_{rest} – мембранний потенціал спокою, E_{exc}та E_{inh} – рівноважні потенціали збудливих та гальмівних синапсів, g_e та g_i– провідності збудливого та гальмівного синапсів відповідно. Як спостерігається в біології, буде використано постійну часу τ, яка для збуджувальних нейронів довша, ніж для гальмівних нейронів. Коли мембранний потенціал нейрона перетинає його мембранний поріг v_{thres} , нейрон спрацьовує, і його мембранний потенціал скидається до v_{reset} . Протягом наступних кількох мілісекунд після скидання нейрон перебуває у своєму рефрактерному періоді і не може знову стрибнути.

Синапси моделюються змінами провідності [11], тобто синапси миттєво збільшують свою провідність на синаптичну вагу *w,* коли пресинаптичний спайк приходить до синапсу, інакше провідність зменшується експоненціально. Якщо пресинаптичний нейрон збудливий, то динаміка провідності g_e є

$$\tau g_e \frac{dg_e}{dt} = \ - g_e \quad (2.6)$$

де τg_e – постійна часу постсинаптичного потенціалу збудження. Аналогічно, якщо пресинаптичний нейрон є гальмівним, провідність g_i оновлюється за допомогою того ж рівняння, але з постійною часу гальмівного постсинаптичного потенціалу τg_i .

Ми використовуємо біологічно подібні діапазони майже для всіх параметрів у нашому моделюванні, включаючи постійні часу мембран, синапсів та вікон навчання [11]; винятком є постійна часу мембранної

напруги збудливих нейронів. Збільшення постійної часу мембранного потенціалу збудливого нейрона до 100 мс (зазвичай спостерігається для біологічних нейронів від 10 до 20 мс) значно підвищило точність класифікації. Причина в тому, що для представлення вхідних даних використовується частотне кодування, і тому довші константи нейронної мембрани дозволяють краще оцінити вхідні частоти спайків. Наприклад, якщо нейрон розпізнавання може інтегрувати тільки вхідні сигнали протягом 20 мс з максимальною швидкістю введення 63,75 Гц, нейрон в середньому інтегруватиме лише 1,275 спайків, що означає, що одиничний спайк шуму матиме великий вплив. Збільшуючи постійну часу мембрани до 100 мс, нейрон може інтегрувати в середньому понад 6,375 спайків, зменшуючи вплив шуму. Проблема занадто малої кількості вхідних спайків існує лише тому, що архітектура використовує набагато меншу кількість вхідних нейронів, ніж біологічно спостерігається, для збільшення швидкості моделювання. Збільшення кількості вхідних нейронів дозволить отримати той самий ефект усереднення.

Мережа складається з двох шарів (Рисунок 2.3) . Перший шар — це вхідний шар, що містить 28 × 28 нейронів (один нейрон на піксель зображення), а другий шар — шар обробки, що містить змінну кількість збуджувальних нейронів і стільки ж гальмівних нейронів. Кожен вхід являє собою пуассонівську спайк-послідовність, яка подається на збудливі нейрони другого шару. Частота генерування кожного нейрона пропорційна інтенсивності відповідного пікселя.

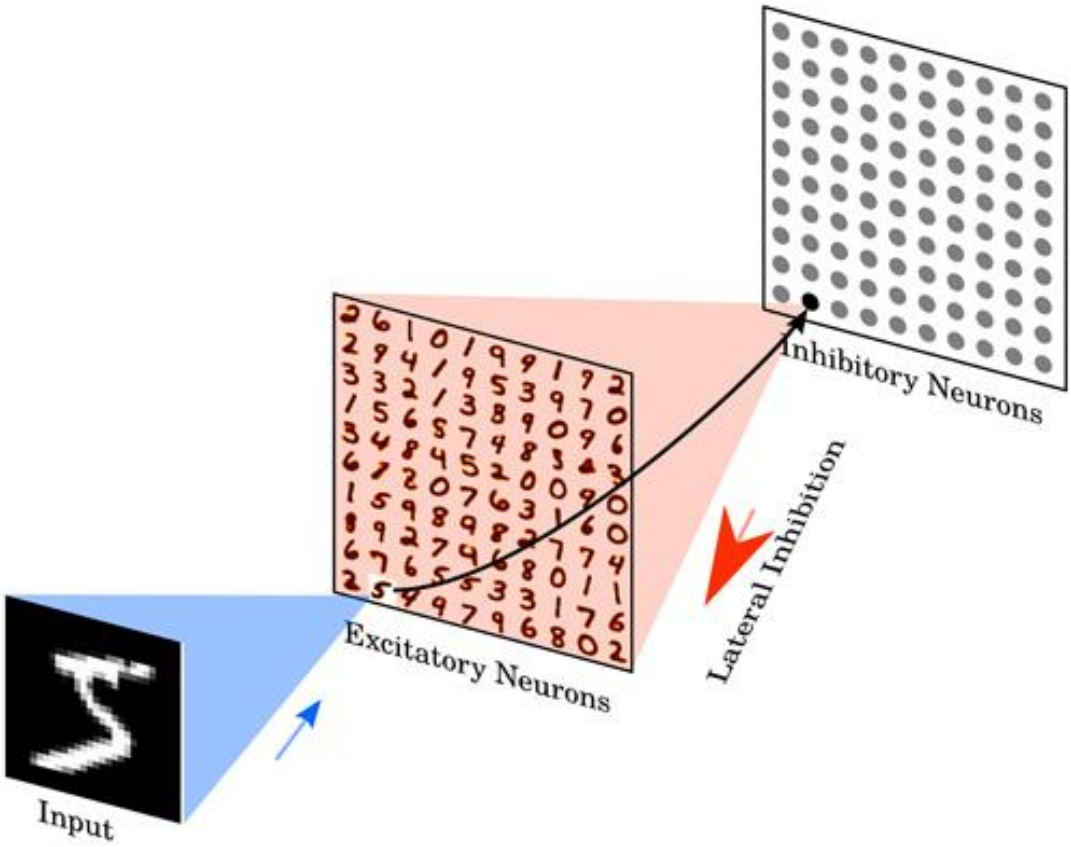

Рисунок 2.3 – Архітектура спайкінгової нейромережі

Нейрони збудження другого шару з’єднані один до одного з гальмівними нейронами, тобто кожен спайк у збудливому нейроні буде викликати спайк у відповідному гальмівному нейроні. Кожен із гальмівних нейронів з'єднаний з усіма збудливими, крім того, від якого він отримує зв'язок. Ця зв’язність забезпечує латеральне гальмування і призводить до конкуренції між збудливими нейронами. Максимальна провідність гальмівного до збуджувального синапсу фіксується на рівні 10 нс. Однак точне значення не мало великого впливу на результати моделювання, натомість співвідношення між гальмівною та збудливою синаптичною провідністю має бути збалансовано, щоб гарантувати, що латеральне гальмування не є або занадто слабким, що означатиме, що воно не має жодних впливів, ані занадто сильним, що означало б, що після

обрання переможця, переможець запобігає спрацьовуванню інших нейронів.

2.3 Математична модель та порядок функціонування спайкіногової нейроної мережі для розпізнавання рукописних цифр

Спайкінгові нейронні мережі (SNN) — це штучні нейронні мережі, які більш точно імітують природні нейронні мережі [12]. На додаток до нейронного та синаптичного стану, SNN включають концепцію часу у свою робочу модель . Ідея полягає в тому, що нейрони в SNN не передають інформацію в кожному циклі поширення (як це відбувається з типовими багатошаровими персептронними мережами), а передають інформацію лише тоді, коли мембранний потенціал – внутрішня якість нейрона, пов’язана з електричним зарядом його мембрани – досягає певного значення, яке називається пороговим. Коли мембранний потенціал досягає порога, нейрон спрацьовує і генерує сигнал, який надходить до інших нейронів, які, у свою чергу, збільшують або зменшують свій потенціал у відповідь на цей сигнал. Модель нейрона, яка спрацьовує в момент перетину порогу, також називається спайкінговою моделлю нейрона [12].

Найпомітнішою моделлю спайкінгових нейронів є LIF-модель (leaky integrate-and-fire) . У LIF-моделі миттєвий рівень активації (змодельований як диференціальне рівняння) зазвичай вважається станом нейрона, при цьому вхідні спайки підсилюють це значення вище або нижче, доки стан зрештою або не загасне, або - якщо спрацьовування досягнуто порогу — нейрон спрацьовує. Після спрацювання змінна стану скидається до нижчого значення.

Існують різні методи декодування для інтерпретації вихідного імпульсу як числа реального значення, що спирається або на

частоту спайків (частотний код), або на час до першого стрибка після стимуляції, або інтервал між спайками.

2.3.1 Навчання мережі

Усі синапси від вхідних нейронів до нейронів збудження навчаються за допомогою правила «spike-timing dependent plasticity» (STDP). Щоб покращити швидкість моделювання, динаміка ваги обчислюється за допомогою синаптичних слідів [13]. Це означає, що, крім синаптичної ваги, кожен синапс відстежує інше значення, а саме пресинаптичний слід x_{pre} , який моделює недавню історію пресинаптичних спайків. Кожного разу, коли пресинаптичний спайк надходить до синапсу, слід збільшується на 1, інакше x_{pre} згасає експоненціально. Коли постсинаптичний спайк приходить до синапсу, зміна ваги Δw розраховується на основі пресинаптичного сліду.

$$\Delta w = \eta\left(x_{pre} - x_{tar}\right)(w_{max} - w)^{\mu} \tag{2.7}$$

де η – швидкість навчання, w_{max}– максимальна вага, а μ визначає залежність оновлення від попередньої ваги. x_{tar} – цільове значення пресинаптичного сліду на момент постсинаптичного спайку. Чим вище цільове значення, тим нижчою буде синаптична вага. Це зміщення гарантує, що пресинаптичні нейрони, які рідко призводять до спрацьовування постсинаптичного нейрона, будуть все більше і більше роз’єднуватися, і це особливо корисно, якщо постсинаптичний нейрон лише рідко активний. Аналогічного ефекту можна досягти, додавши деякий шум до входу та додавши механізм зменшення ваги до правила навчання (як у класичному STDP, [14], щоб відключити невідповідні входи. Однак у наших моделюваннях це відбувається за ціною збільшення

часу моделювання. Правило навчання подібне до того, яке використовується в [15], але тут використано експоненціальну залежність від часу, яка є більш біологічно правдоподібною, ніж незалежна від часу зміна ваги.

Щоб порівняти надійність вибраної архітектури з точною формою правила навчання, було перевірено три інші правила навчання STDP. Друге правило STDP використовує експоненціальну залежність ваги [14] для обчислення зміни ваги

$$\Delta w = \eta_{post}\left(x_{pre}e^{-\beta w} - x_{tar}e^{-\beta(w_{max}-w)}\right) \tag{2.8}$$

де β визначає силу вагової залежності.

Третє правило використовує не тільки пресинаптичний слід, а й постсинаптичний слід, який працює так само, як пресинаптичний слід, але його збільшення викликається постсинаптичним спайком. Крім того, для цього правила навчання зміни ваги відбуваються для пре- і постсинаптичних спайків. Зміна ваги Δw для пресинаптичного спайка дорівнює

$$\Delta w = -\eta_{pre}x_{post}w^{\mu} \tag{2.9}$$

де η_{pre} – швидкість навчання для пресинаптичного спайку, а μ визначає залежність від ваги. Зміна ваги для постсинаптичного спайку становить

$$\Delta w = \eta_{post}\left(x_{pre} - x_{tar}\right)(w_{max} - w)^{\mu} \tag{2.10}$$

де η_{post} — швидкість навчання, w_{max} — максимальна вага, а x_{tar}— цільове середнє значення пресинаптичного сліду в момент постсинаптичного спайку.

Крім того, ваги мережі було знайдено з використанням триплетного правила STDP [13]. Оскільки це правило не використовує вагові залежності для навчання, нам потрібно або включити його в правило, або обмежити вагові показники в іншій формі. Тут було використано роздільну нормалізацію ваги [14], що забезпечує однакове використання нейронів.

Зауважимо, що степеневий закон і правило STDP для експоненційної залежності від ваги мають перевагу, оскільки оновлення ваги запускаються лише тоді, коли постсинаптичний збудливий нейрон запускає спайк. Оскільки швидкість запуску постсинаптичних нейронів досить низька, більш складне оновлення STDP для постсинаптичного спрацьовування не вимагає багато обчислювальних ресурсів. Правило симетричного навчання і правило триплета є дорожчими для моделювання з використанням програмного моделювання (особливо для великих мереж), оскільки для кожної пресинаптичної події зміна ваги має бути розрахована для кожного окремого постсинаптичного нейрона.

2.3.2 Стійкість мережі

Неоднорідність вхідного сигналу призводить до різної швидкості спрацьовування збуджуючих нейронів, а латеральне гальмування ще більше збільшує цю різницю. Однак бажано, щоб усі нейрони мали приблизно однакову швидкість спрацьовування, щоб запобігти тому, щоб окремі нейрони домінували у шаблоні відповіді, і щоб забезпечити диференціацію рецептивних полів нейронів. Щоб досягти цього, було використано адаптивний поріг мембрани, що нагадує внутрішню пластичність. Зокрема, поріг мембрани кожного збуджувального нейрона

визначається не тільки v_{thresh}, але й сумою v_{thresh} + θ, де θ збільшується щоразу, коли нейрон спрацьовує, і експоненціально загасає [15]. Таким чином, чим більше спрацьовує нейрон, тим вищим буде поріг його мембрани, і, у свою чергу, нейрон потребує більшого введення для швидкого зростання в найближчому майбутньому. Використовуючи цей механізм, швидкість спрацьовування нейронів обмежена, оскільки модель синапсу на основі провідності обмежує максимальний потенціал мембрани потенціалом реверсії збудження E_{exc}, тобто, як тільки поріг мембрани нейрона наближається до E_{exc} (або вище), він буде спрацьовувати менш часто (або навіть повністю припинить генерацію), поки θ не зменшиться достатньо.

2.3.3 Вхідне кодування

Вхідні дані в мережу базуються на наборі даних MNIST, який містить 40 000 навчальних прикладів і 1000 тестових прикладів зображень 28 × 28 пікселів з цифрами 0–9 [16]. Вхідні дані подаються в мережу протягом 250 мс у вигляді розподілених по Пуассону ланцюжків спайків з частотою спрацьовування, пропорційною інтенсивності пікселів зображень MNIST. Зокрема, максимальна інтенсивність пікселів 255 ділиться на 4, в результаті чого частота спрацьовування вхідних даних становить від 0 до 63,75 Гц. Крім того, якщо збудливі нейрони в другому шарі генерують менше п’яти спайків протягом 250 мс, максимальна вхідна частота спрацьовування збільшується на 32 Гц, і приклад буде представлено знову протягом 250 мс. Цей процес повторюється до тих пір, поки за весь час представлення конкретного прикладу не буде згенеровано щонайменше п’ять спайків.

2.3.4 Навчання та класифікація

Для навчання мережі в мережу представляються цифри з навчального набору MNIST (40 000 прикладів). Перед представленням нового зображення є фаза 150 мс без будь-якого введення, щоб дозволити всім змінним усіх нейронів знизитися до значень спокою (крім адаптивного порогу). Після завершення навчання встановлюється швидкість навчання на нуль, фіксується порогове значення для кожного нейрона і призначається клас кожному нейрону на основі його найвищої реакції на десять класів цифр за одну презентацію навчального набору. Це єдиний крок, де використовуються мітки, тобто для навчання синаптичних ваг мітки не використовуються.

Відповідь нейронів, призначених для класу, потім використовується для вимірювання достовірності класифікації мережі на тестовому наборі MNIST (1000 прикладів). Прогнозована цифра визначається шляхом усереднення відповідей кожного нейрона на клас, а потім вибору класу з найвищою середньою вихідною частотою.

2.4 Особливості реалізації запропонованої архітектури спайкінгової нейромережі

2.4.1 Гальмування

У поточній реалізації було використано стільки ж гальмівних нейронів, скільки і збуджуючих нейронів, так що кожен спайк збудливого нейрона (опосередковано) призводить до гальмування всіх інших збуджуючих нейронів. Було обрано цю більш пряму реалізацію м’якого механізму «переможець отримує все», щоб зменшити складність обчислень. Це можна змінити на більш біологічно правдоподібну архітектуру, замінивши великий пул гальмівних нейронів меншим, щоб відповідати біологічно спостережуваному співвідношенню збуджуючих

нейронів до гальмівних нейронів 4:1, а також використовуючи зв’язок «один до багатьох» від збудливого до збудливого. гальмівні нейрони. Це призведе до мережі, де спайк збудливого нейрона призводить до неоднорідних гальмівних входів до інших збуджуючих нейронів і, таким чином, може сприяти активації одних нейронів над іншими. Тим не менш, адаптивний поріг може врівноважити цей ефект, а також у великій мережі ці ефекти повинні бути усередненими, а це означає, що продуктивність мережі повинна залишатися приблизно однаковою.

2.4.2 Навчання на основі спайків для машинного навчання

Оскільки споживання енергії є основним фактором витрат для компаній з великою кількістю даних, існує сильна мотивація зменшити енергоспоживання мікросхем. Сучасні реалізації нейронних мереж (SNN) на нейроморфному апаратному забезпеченні [17] використовують лише кілька нДж або навіть пДж для передачі спайка (для деяких установок лише 0,02 пДж на спайк, і споживає лише кілька pW потужності на синапс; деякі з цих нейроморфних систем також пропонують механізми навчання на чіпі [18].

Враховуючи, що енергоспоживання, швидше за все, буде однією з головних причин використання нейроморфного апаратного забезпечення в поєднанні з архітектурами машинного навчання на основі спайків, може бути кращим використовувати навчання на основі спайків замість навчання на основі частоти, оскільки сама процедура навчання має високе енергоспоживання (проте зверніть увагу, що обидва методи засновані на стрибках під час тестування). Зокрема, навчання на основі спайків важливо, коли процедура навчання займає значну частину часу, який буде використовуватися в мережі. Інше застосування, де потрібне навчання на основі спайків, — це для систем, які повинні динамічно адаптуватися до свого середовища, тобто коли недостатньо навчити систему один раз і

запустити її з попередньо натренованими вагами. Можливі приклади включають системи розпізнавання мовлення, які попередньо навчені, але адаптовані до акценту користувача, або процесори зору, які мають бути налаштовані на певний датчик зору. Така адаптивна система обробки зору особливо цікава в поєднанні з датчиком різкого зору, таким як ATIS або DVS, оскільки він забезпечує наскрізну малопотужну систему бачення на основі спайків. Якби наша мережа була реалізована на малопотужному нейроморфному чіпі, вона могла б працювати на дуже низькому рівні. енергетичний бюджет; наприклад, використовуючи чіп TrueNorth від IBM [19], який споживає близько 72 мВт на 1 мільйон нейронів, мережа споживає менше 1 мВт.

2.4.3 Конкурсне навчання

Інтуїтивно функція мережі схожа на процедури змагального навчання, наприклад, самоорганізуючі карти [5], які мають спільні аспекти з k-середніми. Ця аналогія з алгоритмами навчання, подібними до k-середніх, є особливо цікавою, оскільки останнім часом такі підходи виявилися дуже успішними у складних завданнях машинного навчання. Поглиблений аналіз максимізації очікувань у мережі стрибків можна знайти у [20]. Основна ідея полягає в тому, що кожен нейрон навчається і представляє один прототипний вхід або середнє значення деяких подібних входів. Кожного разу, коли вводиться вхід, мережа визначає прототипи, які найбільше подібні до конкретного входу. Ці прототипи-переможці потім використовуються для прогнозування класу вхідних даних, їх ваги адаптуються таким чином, щоб вони стали більш схожими на поточні вхідні дані. У нашій мережі це означає, що щоразу, коли нейрон стрибки, оскільки приклад досить схожий на його рецептивне поле, він зробить своє рецептивне поле більш схожим на приклад. Бокове гальмування не дає прототипам стати занадто схожими один на одного (що означає, що

вони поширюються у вхідному просторі), оскільки лише кілька різних нейронів зможуть реагувати на кожен приклад, і, у свою чергу, лише кілька нейронів можуть адаптувати свої рецептивні поля до нього. Гомеостаз можна розглядати як інструмент для підтримки приблизно постійної кількості прикладів у межах дії прототипу. У крайньому випадку, представляючи стільки різних прикладів введення, скільки нейрони вивчають їх, мережа більше схожа на k-найближчі сусіди (kNN). Це натякає на те, що максимальна продуктивність представленої тут архітектури за рахунок простого збільшення кількості нейронів, ймовірно, становить приблизно 95– 97%, як і для методів kNN без попередньої обробки (мережа більше схожа на k-nearest-neighbors (kNN). Це натякає на те, що максимальна продуктивність представленої тут архітектури за рахунок простого збільшення кількості нейронів, ймовірно, становить приблизно 95–97%, як і для методів kNN без попередньої обробки (мережа більше схожа на k-nearest-neighbors (kNN). Це натякає на те, що максимальна продуктивність представленої тут архітектури за рахунок простого збільшення кількості нейронів, ймовірно, становить приблизно 95–97%, як і для методів kNN без попередньої обробки. Проте, ймовірно, можна ще більше підвищити продуктивність, використовуючи більше шарів тієї ж архітектури.

2.4.4 Надійність навчання

Ми показали, що за допомогою чотирьох різних правил STDP разом з бічним гальмуванням і гомеостазом отримані мережі мають подібну продуктивність і демонструють дуже стабільне навчання з часом. Особливо останньої властивості зазвичай важко досягти, оскільки багато мереж мають тенденцію переповнювати дані або не мають механізмів, щоб запобігти надмірному зростанню ваги. Мережа вже добре працює після представлення 40 000 прикладів, але також вона не демонструє

зниження продуктивності навіть після одного мільйона прикладів. Причина, чому продуктивність мережі є настільки стабільною з часом, швидше за все, полягає в конкуренції між нейронами, яка змушує нейрони вивчати якомога різноманітніші шаблони введення, і залежність від ваги навчання, яка запобігає зростанню ваги, якщо шаблони введення не відображають його. Ця гнучкість щодо кількості прикладів навчання та змін у реалізації є вирішальною в реальних біологічних системах, де ми знаходимо дуже гетерогенні клітини для різних тварин одного виду і навіть різні властивості в однієї тварини для різних клітин.

Іншими перевагами нашої системи є її масштабованість, що дозволяє знайти компроміс між обчислювальною вартістю та продуктивністю, а також гнучкість у правилах неконтрольованого навчання на основі спайків, що дозволяє навчати мережу без міток і використовувати лише кілька міток для призначення нейронів класам.

2.5 Висновок до розділу 2

У розділі було обгрунтовано вибір спайкінгової нейромережі для задачі розпізнавання рукописних цифр, розроблено архітектуру спайкінгової нейромережі для інформаційної технології розпізнавання рукописних цифр, яка має 784 входи та 2 шари спайкінгових нейронів. Описано математичну модель та порядок функціонування спайкіногової нейроної мережі для розпізнавання рукописних цифр. Розроблено процеси навчання, вхідного кодування та класифікації рукописних цифр спайкінговою нейромережею. Також розглянуто особливості реалізації запропонованої архітектури спайкінгової нейромережі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР СПАЙКІНГОВОЮ НЕЙРОННОЮ МЕРЕЖЕЮ

3.1 Обґрунтування вибору мови та середовища програмування

Для реалізації програми розпізнавання рукописних цифр спайкінговою нейронною мережею було обрано мову Python.

Python (найчастіше вживане прочитання — «Пайтон», запозичено назву [21] з британського шоу Монті Пайтон) — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Переваги:

- чистий синтаксис (для виділення блоків слід використовувати відступи);
- переносність програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів (включно з модулем для розробки графічного інтерфейсу);

- можливість використання Python в діалоговому режимі (дуже корисне для експериментування та розв'язання простих задач);
- стандартний дистрибутив має просте, але разом із тим досить потужне середовище розробки, яке зветься IDLE і яке написане мовою Python;
- зручний для розв'язання математичних проблем (має засоби роботи з комплексними числами, може оперувати з цілими числами довільної величини, у діалоговому режимі може використовуватися як потужний калькулятор);
- відкритий код (можливість редагувати його іншими користувачами).

Недоліки:

Низька швидкодія - Python, як і багато інших інтерпретованих мов, які не застосовують, наприклад, JIT-компілятори, мають загальний недолік — порівняно низьку швидкість виконання програм. Однак, у випадку з Python цей недолік компенсується зменшенням часу розробки програми. У середньому, програма, написана на Python, в 2-4 рази компактніша, ніж її аналог на C++ або Java. Збереження байт-коду (файли .pyc і .pyo) дозволяє інтерпретатору не витрачати зайвий час на перекомпіляцію коду модулів при кожному запуску, на відміну, наприклад, від мови Perl. Крім того, існує спеціальна JIT-бібліотека psyco (проте призводить до збільшення споживання оперативної пам'яті). Ефективність psyco значною мірою залежить від архітектури програми.

Існують проєкти реалізацій мови Python, що вводять високопродуктивні віртуальні машини (ВМ) як компілятора заднього плану. Прикладами таких реалізацій може служити PyPy, що базується на LLVM; більш ранньою ініціативою є проєкт Parrot. Очікується, що використання ВМ типу LLVM призведе до тих самих результатів, що й використання аналогічних підходів для реалізацій мови Java, де низька

обчислювальна продуктивність в основному подолана. Низка програм/бібліотек для інтеграції з іншими мовами програмування (див. вище) надають можливість використовувати іншу мову для написання критичних ділянок.

У найпопулярнішій реалізації мови Python інтерпретатор досить великий і більш вимогливий до ресурсів, ніж в аналогічних популярних реалізаціях Tcl, Forth, LISP або Lua, що обмежує його застосування у вбудованих системах. Тим не менше, Python знайшов застосування в КПК і деяких моделях мобільних телефонів.

Єдиним недоліком, на який звертав увагу сам автор є порівняно невисока швидкість виконання Python програми. Однак, це не відграє велику роль у порівнянні з перевагами мови при написанні програм не дуже критичних до швидкості виконання.

Для моделювання SNN було використано Python та симулятор BRIAN [22].

3.2 Розробка алгоритму роботи програми розпізнавання рукописних цифр

Відповідно до дослідженої інформації за темою даної роботи розробки програми розпізнавання рукописних цифр за допомогою спайкінгової нейронної мережі було розроблено алгоритм роботи програми (Рисунок 3.1).

Спочатку вся множина прикладів цифр розбивається на 2 підмножини – навчальна та тестова вибірки. Далі вагам нейронів збуджувальног шару присвоюються початкові значення ваг, близькі до двійкових матриць класів.

Всі зображення символів і з навчальної і з тестової вибірки приводяться до розміру 28x28 пікселів.

Якщо зображення рукописної цифри представлено у форматі RGB, то воно перетворюється у напівтонове (сіре зображення).

Далі проводиться некероване навчання моделі спайкінгової мережі за правилом STDP.

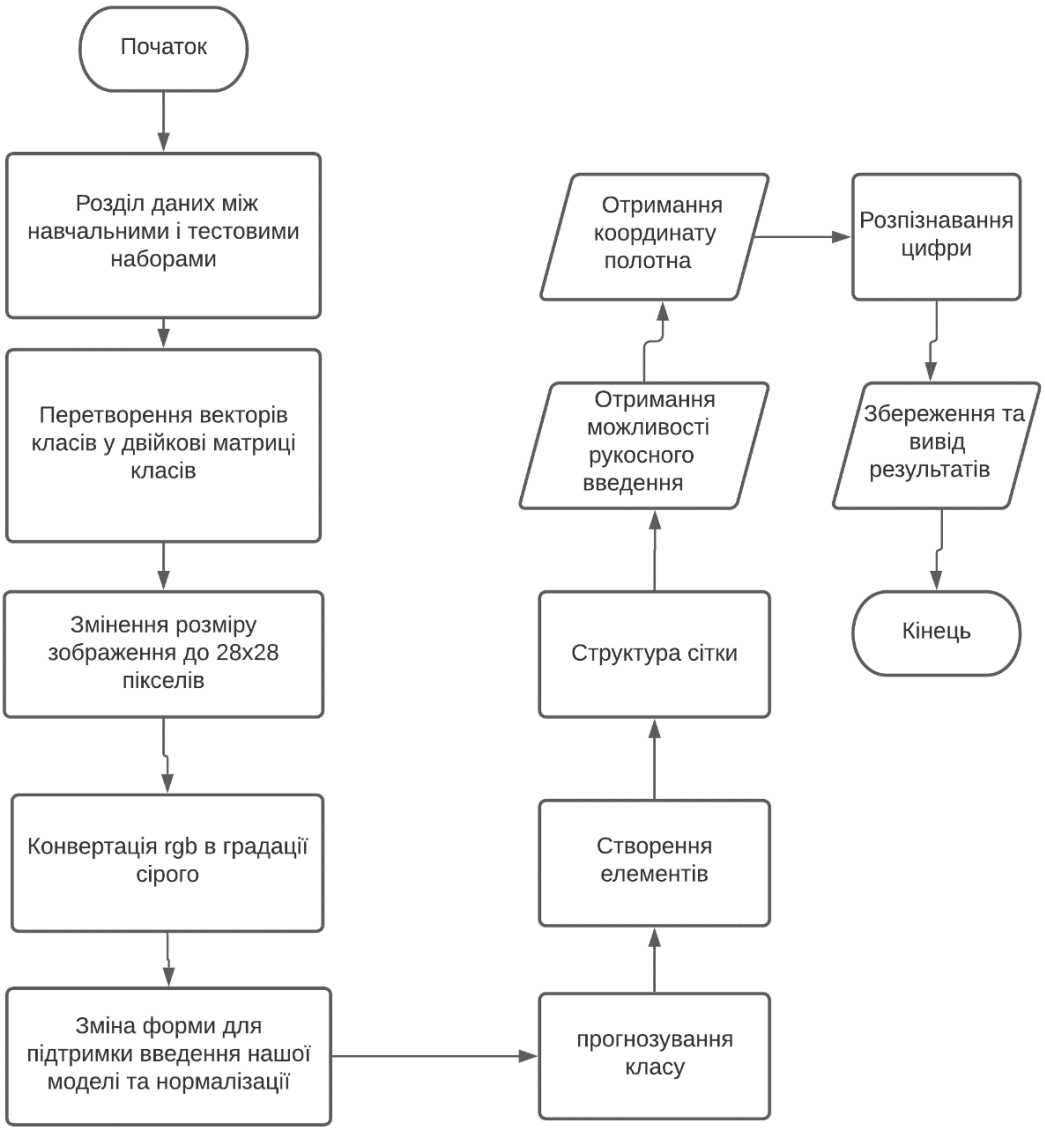

Рисунок 3.1 – Схема алгоритму роботи програми

Після навчання спайкінгова нейромережа готова до розпізнавання рукописних цифр, тобто до прогнозування класу рукописної цифри.

Для розпізнавання створюються програмні нейроелементи, які об’єднуються у структуру спайкінгової нейромережі.

Програма створює у своєму робочому вікні поле, в якому мишкою можна вводити зображення рукописної цифри.

Після введенні цифри формуються координати її пікселів, нормалізуються до розміру 28х28 пікселів.

Далі відбувається процес розпізнавання введеної рукописної цифри спайкінговою нейромережею.

Після цього відбувається виведення результату розпізнавання на екран та збереження у пам’яті результатів розпізнавання.

3.3 Програмна реалізація розпізнавання рукописних цифр

Спочатку потрібно імпортувати всі модулі, які знадобляться для навчання нашої моделі.

```
import warnings
warnings.filterwarnings('ignore')

import matplotlib.cm as cmap
import brian_no_units
import networkx as nx
import cPickle as p
import pandas as pd
import numpy as np
import brian as b
import argparse
import random
import timeit
import time
import math
import os
```

```
from mpl_toolkits.axes_grid1 import make_axes_locatable
from scipy.spatial.distance import euclidean
from sklearn.metrics import confusion_matrix
from struct import unpack
from brian import *

from util import *
import pickle
```

Далі потрібно підключити необхідні нам модулі для навчання нейроної мережі

```
np.set_printoptions(threshold=np.nan, linewidth=200)

# only show log messages of level ERROR or higher
b.log_level_error()

# set these appropriate to your directory structure
top_level_path = os.path.join('..', '..')
MNIST_data_path = os.path.join(top_level_path, 'data')
model_name = 'csnn_pc_mnist'
results_path = os.path.join(top_level_path, 'results', model_name)

performance_dir = os.path.join(top_level_path, 'performance', model_name)
activity_dir = os.path.join(top_level_path, 'activity', model_name)

weights_dir = os.path.join(top_level_path, 'weights', model_name)
best_weights_dir = os.path.join(weights_dir, 'best')
end_weights_dir = os.path.join(weights_dir, 'end')

assignments_dir = os.path.join(top_level_path, 'assignments', model_name)
best_assignments_dir = os.path.join(assignments_dir, 'best')
end_assignments_dir = os.path.join(assignments_dir, 'end')

misc_dir = os.path.join(top_level_path, 'misc', model_name)
best_misc_dir = os.path.join(misc_dir, 'best')
end_misc_dir = os.path.join(misc_dir, 'end')
```

```
ngram_dir = os.path.join(top_level_path, 'ngrams', model_name)

for d in [ performance_dir, activity_dir, weights_dir, misc_dir,
best_misc_dir,
                assignments_dir, best_assignments_dir, MNIST_data_path,
results_path,
            best_weights_dir, end_weights_dir, end_misc_dir,
end_assignments_dir, ngram_dir ]:
    if not os.path.isdir(d):
        os.makedirs(d)
```

Створюємо функції стиснення збуджувальних синаптичних ваг, щоб підсумувати зазделегідь задане число

```
def normalize_weights():

    for conn_name in input_connections:
        connection = input_connections[conn_name][:].todense()
        for feature in xrange(conv_features):
            feature_connection = connection[:, feature * n_e : (feature + 1) *
n_e]
            column_sums = np.sum(np.asarray(feature_connection), axis=0)
            column_factors = weight['ee_input'] / column_sums

            for n in xrange(n_e):
                dense_weights = input_connections[conn_name][:, feature * n_e
+ n].todense()
                dense_weights[convolution_locations[n]] *= column_factors[n]
                input_connections[conn_name][:, feature * n_e + n] =
dense_weights
```

Будуємо графік поточного вхідного прикладу під час процедури навчання

```
def plot_input(rates):
```

```
    fig = plt.figure(fig_num, figsize = (5, 5))
    im = plt.imshow(rates.reshape((28, 28)), interpolation='nearest', vmin=0,
vmax=64, cmap='binary')
    plt.colorbar(im)
    plt.title('Current input example')
    fig.canvas.draw()
    return im, fig
```

Діаграма класів програми розпізнавання рукописних цифр на основі спайкінгової нейромережі представлена на рис. 3.2.

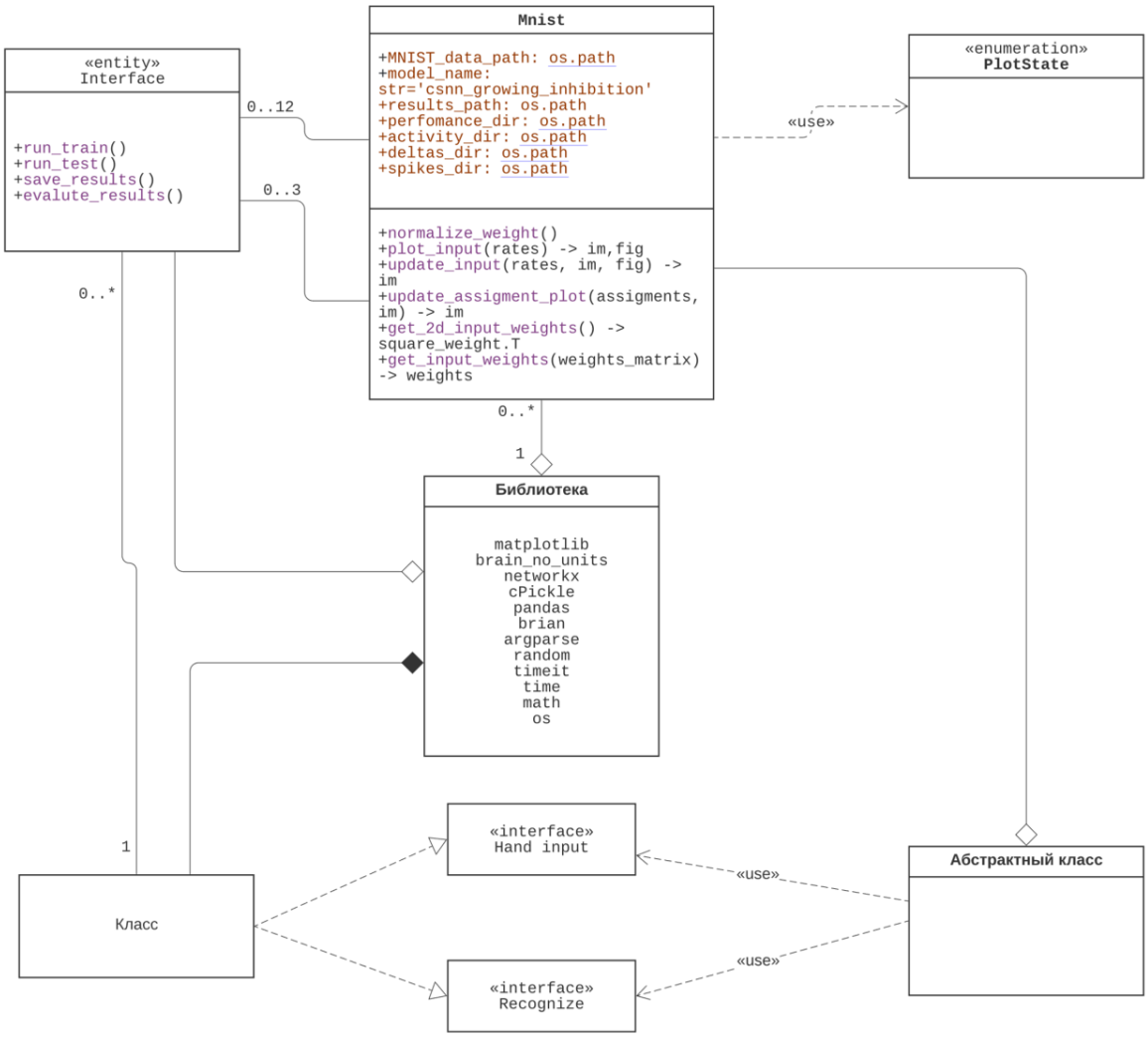

Рисунок 3.2 – Діаграма класів програми розпізнавання рукописних цифр на основі спайкінгової нейромережі

Діаграма дає уявлення про наявні класи програми та їх взаємодію.

3.4 Висновок

У розділі було обґрунтовано вибір мови Python та спеціалізованої бібліотеки BRIAN для програмної реалізації інформаційної технології розпізнавання рукописних цифр. Розроблено алгоритм роботи та UML діаграму класів програми розпізнавання рукописних цифр, Описано основні етапи реалізація та функціонування програми розпізнавання рукописних цифр.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМИ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР СПАЙКІНГОВОЮ НЕЙРОННОЮ МЕРЕЖЕЮ

4.1 Тестування програми розпізнавання рукописних цифр спайкінговою нейронною мережею

В процесі досліджень роботи програми була навчена та протестована мережа зі 100 збудливими нейронами. Навчальна вибірка містила 40 000 прикладів із бази даних MNIST. Отриманий перекомпонований вхід до ваг збуджуючих нейронів показаний на рис. 4.1A. Для кожного нейрона 784-вимірний вхідний вектор перебудовується в матрицю 28 × 28, щоб уявити, що нейрони навчаються прототипним вхідним даним.

На рис. 4.1B кожна точка показує продуктивність для певного розміру мережі як середнє значення за десять презентацій всього набору тестів MNIST. Смужки помилок позначають стандартне відхилення між десятьма представленнями тестового набору. Ефективність кожного з правил навчання позначено кольором лінії: чорним (для правила ступеневої залежності від ваги STDP), червоним (для правила експоненціальної залежності від ваги STDP), зеленим (для правила пре-і-пост STDP) і синім (для правила триплетного STDP) відповідно.

При моделюванні спайкінгової нейронної мережі були використані однакові параметри нейрона, синапсу та STDP (за винятком параметрів адаптивного порогу та сили гальмування, які необхідно було адаптувати для підтримки постійної швидкості реакції). Оскільки зображення інтенсивності тестового набору MNIST перетворюються на розподілені по Пуассону послідовності спайків, точність може відрізнятися для різних реалізацій послідовностей спайків.

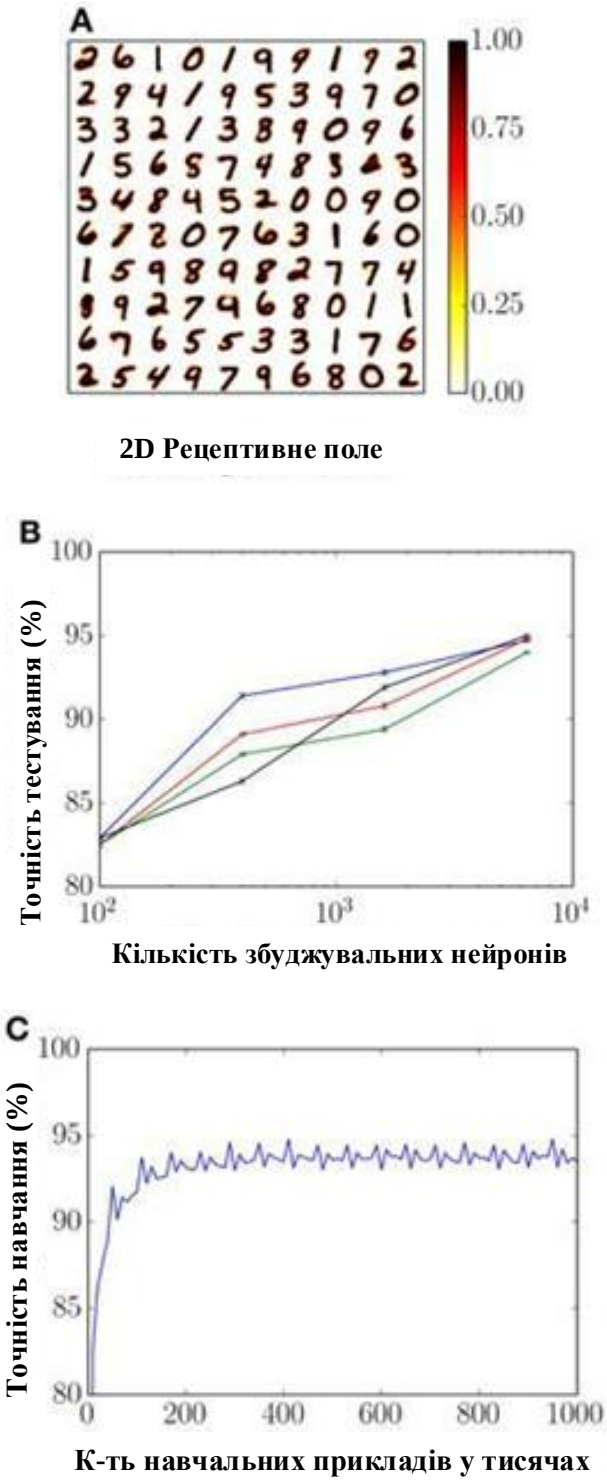

Рисунок 4.1. Результати навчання. (A) Змінені ваги (від 784 до 28 × 28) з’єднань від вхідних до збуджуючих нейронів для мережі зі 100 збудливими нейронами в сітці 10 на 10. (B) Достовірність як функція кількості збуджуючих нейронів. (C) Точність навчання як функція представлених навчальних прикладів.

Однак стандартне відхилення достовірності протягом десяти презентацій усього тестового набору (з використанням тієї ж навченої мережі) невелике (≈0,1%).

Точність навчання для спайкінгової нейронної мережі з 1600 нейронами та з симетричним правилом навчання показано на рис. 4.1C. Після приблизно 200 000 прикладів достовірність близька до конвергенції, і навіть після мільйона прикладів достовірність не знижується, а залишається стабільною. Цікаво, що періодична структура зумовлена повторним представленням навчального набору MNIST. Ця тенденція однакова для всіх розмірів мереж і правил навчання. Однак, більшим мережам потрібно довше тренуватися, поки вони не досягнуть максимальної достовірності.

При запуску програми відкривається вікно, де є поле, на якому можна малювати. Після того як ми написали свою цифру воно її записує в базу та починає розпізнавання (Рисунок 4.2).

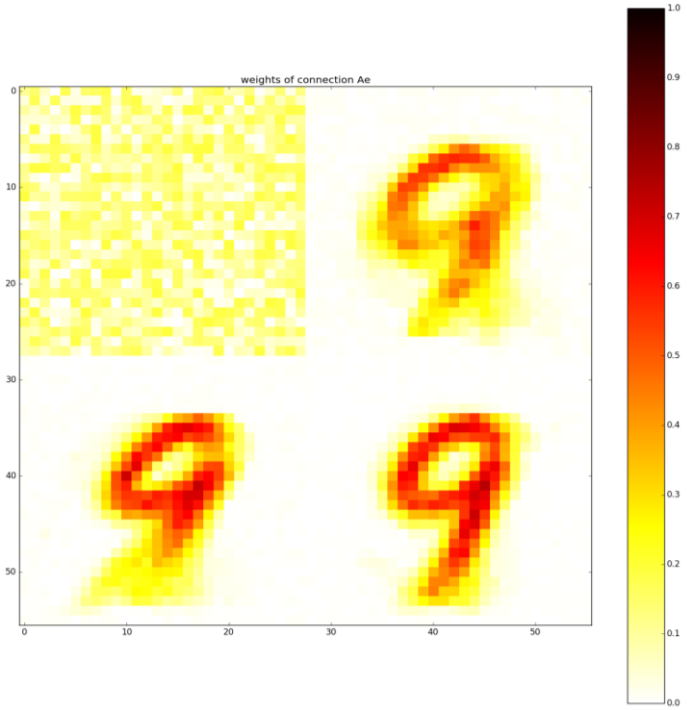

Рисунок 4.2 – Вікно програми для вводу рукописних цифр

На рисунку 4.3 ми бачимо які нейрони у шарі збудливих нейронів найбільше активуються. А це ті нейрони, які навчені на образ найбільш схожий на введену цифру 9, а саме: 3,9,0,8.

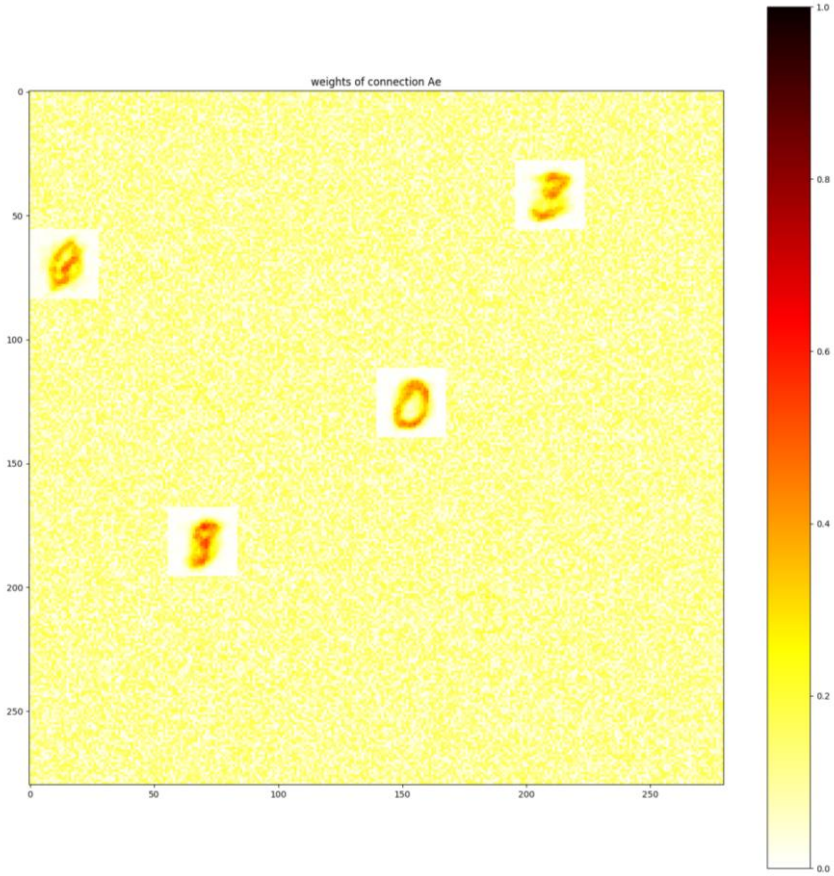

Рисунок 4.3 – Розпізнавання рукописних цифр 100 нейронами

На рисунку 4.4 ми бачимо вихідні імпульсні сигнали (спайки) нейронів шару збудливих нейронів. І там видно, що найбільше активуються (тобто мають найбільшу кількість згенерованих спайків за період моделювання) саме ті нейрони, які підсвічені на рис. 4.3.

На рисунку 4.5 представлено графік підрахунку кількості вихідних імпульсів (спайків) нейронів шару збудливих нейронів. І там видно, що найбільше значення активації (тобто кількість згенерованих спайків за період моделювання) має саме другий зліва максимум, який відповідає нейрону, який на рис. 4.3 підсвічений цифрою 9.

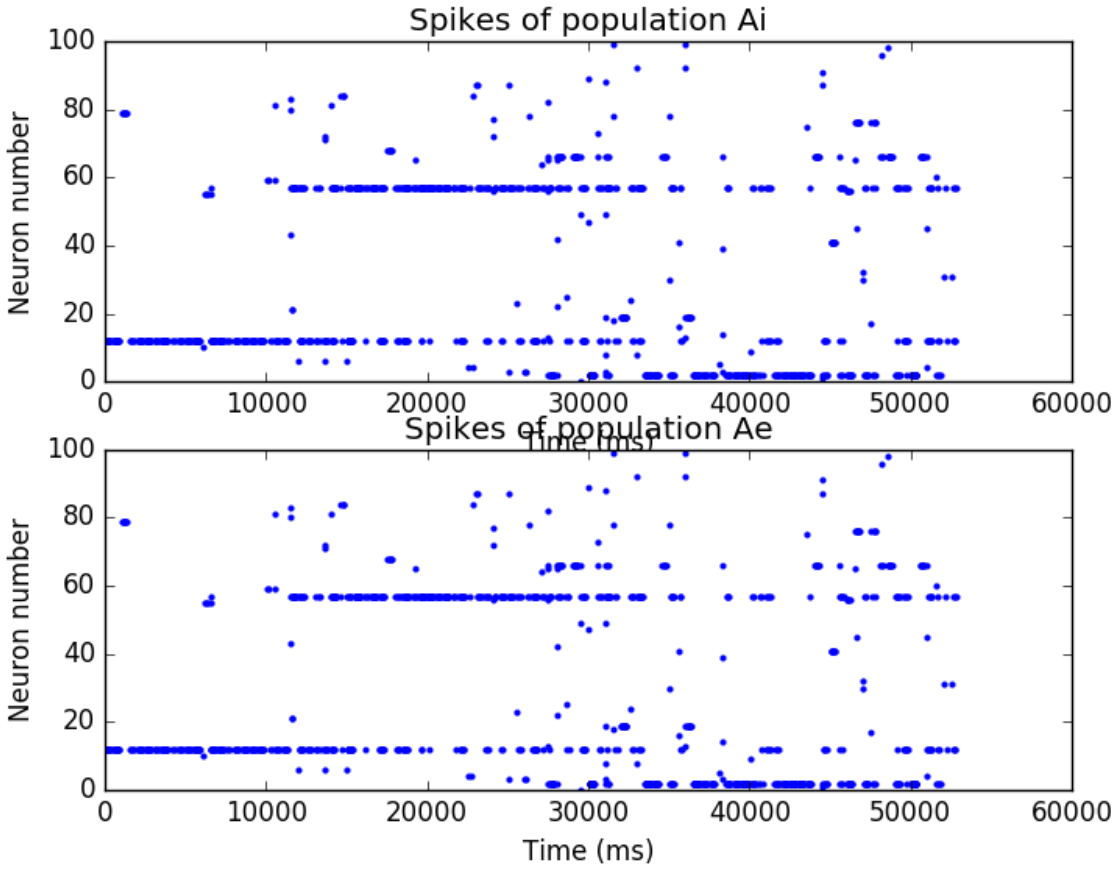

Рисунок 4.4 – Вихідні імпульсні сигнали (спайки) нейронів шару гальмівних Ai та збудливих Ae нейронів

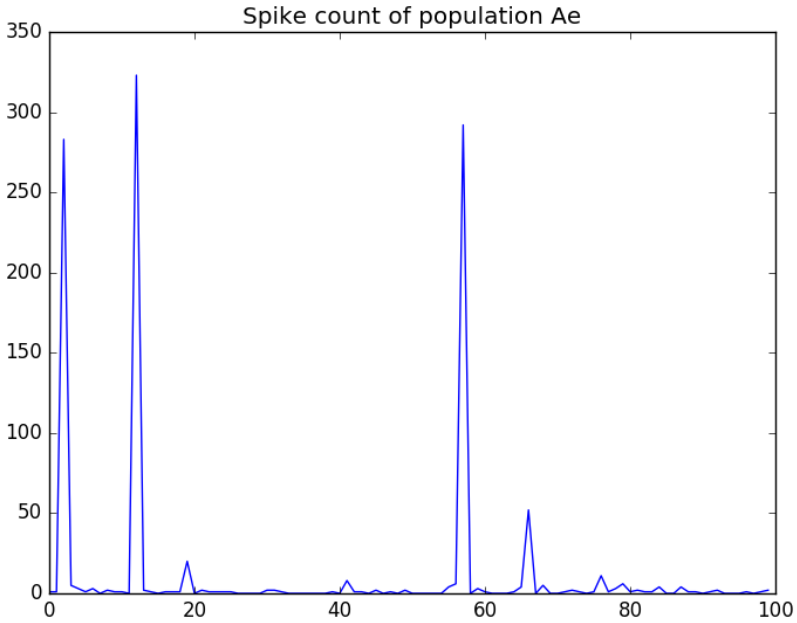

Рисунок 4.5 – Графік підрахунку кількості вихідних імпульсів (спайків) нейронів шару збудливих нейронів

4.2 Аналіз результатів роботи програми розпізнавання рукописних цифр спайкінговою нейронною мережею

Розроблений програма розпізнавання рукописних цифр було навчено з використанням бази даних MNIST [16]. Для навчання мережі в мережу представляються цифри з навчального набору (40 000 прикладів).

Відповідь нейронів, призначених для класу, потім використовується для вимірювання достовірності класифікації мережі на тестовому наборі MNIST (1 000 прикладів).

Аналіз помилок для спайкінгової нейронної мережі із 6400 нейронів із використанням стандартного правила STDP зображено на рис. 4.6. На рис. 4.6A показано усереднену матрицю помилок (confusion matrix) для десяти презентацій тестового набору MNIST, тобто кожна окрема класифікація тестових прикладів належить до одного з 10 на 10 плиток, і його позиція визначається фактичною цифрою та вгаданою цифрою. Високі значення в ідентифікаторі вказують на правильну ідентифікацію, тоді як високі значення будь-де вказують на плутанину (помилки) поміж двох цифр, наприклад, поміж цифрами 4 і 9. Не дивно, що враховуючи достовірність класифікації 95%, більшість прикладів класифікуються правильно. Більш цікавими є неправильно класифіковані приклади. Найпоширеніша помилка полягає в тому, що 4 57 разів ідентифікується як 9, 7 ідентифікується ≈40 разів як 9, а 7 ≈26 разів ідентифікується як 2. Хоча 4 і 9, а також 7 і 2 легко сплутати, це не здається відразу очевидним що 7 можна помилково прийняти за 9. Ймовірне пояснення можна побачити на рис. 4.6B.

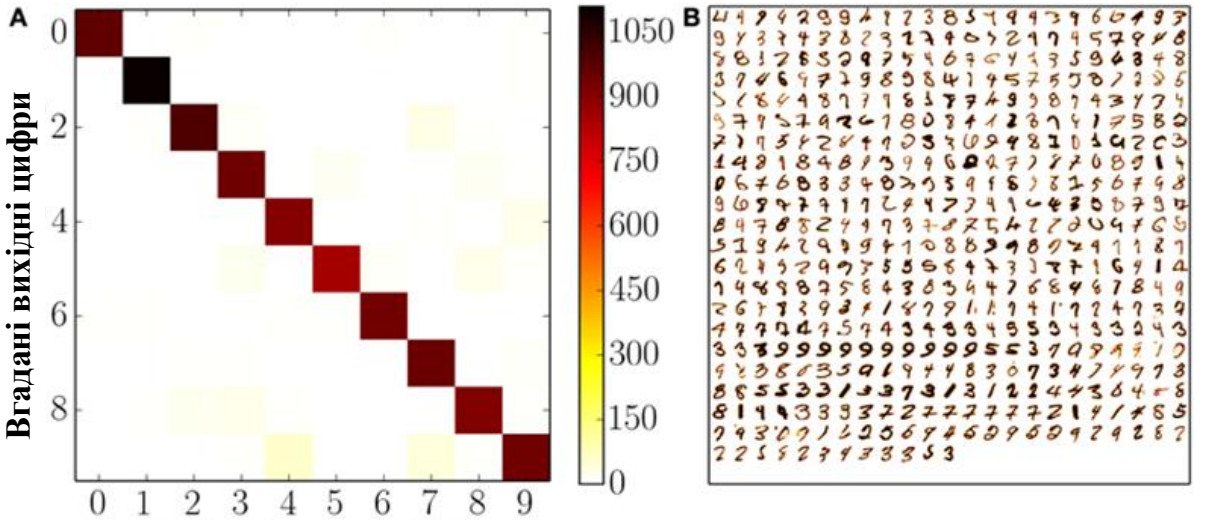

Рисунок 4.6. Аналіз помилок. (A): Середня матриця помилок результатів тестування для десяти представлень 10 000 цифр набору тестів MNIST. (B): Некоректно класифіковані цифри

Усі 495 неправильно класифікованих цифр однієї класифікації проходять через усі 10 000 цифр набору тестів MNIST. Чим темніший піксель цифри, тим вище значення його інтенсивності і, отже, частота вхідних імпульсів. Обидві картинки на рис. 4.6 стосуються спайкінгової нейронної мережі із 6400 збудливими нейронами, що навчалася за стандартним правилом STDP. Часто єдиною відмінною рисою між неправильно класифікованими 7 і типовими 9 є те, що середня горизонтальна лінія в 7 не пов’язана з верхньою рискою, що означає, що нейрони, які мають сприйнятливе поле 9, певною мірою ймовірно також спрацюють.

Оскільки кожен нейрон реагує лише на дуже невелику підмножину вхідних цифр, кількість вихідних імпульсів нейрона дуже невелика, і для однієї вхідної цифри генерується лише кілька імпульсів. Навіть у найбільшій мережі із 6400 збуджувальними нейронами у відповідь на представлення однієї цифри генерується лише ≈17 спайків. Зокрема, для

правильно ідентифікованих прикладів ≈16 спайків генерується від нейронів з того самого класу та ≈1 спайк із нейронів, віднесених до іншого класу, тоді як для неправильно визначених прикладів ≈3 спайки генерується з нейронів правильного класу та ≈12 спайків від нейронів інших класів.

Представлена спайкінгова нейронна мережа досягає хороших показників класифікації за еталонним тестом MNIST, використовуючи неконтрольоване навчання, що складається з біологічно вірогідних компонентів.

Навчальна вибірка складалась із 40 000 прикладів цифр. Тестова вибірка складалась із 1 000 прикладів цифр. Результати тестування запропонованої програми у порівнянні з програмою-аналогом наведено у табл. 4.1.

Таблиця 4.1 – Порівняння параметрів розробленої програми із програмою-аналогом MyScript Стилус

	Кількість зображень у тестовій вибірці	Кількість вірно розпізнанних цифр	Достовірність розпізнавання
MyScript Стилус	1000	842	84 %
Розроблена програма	1000	911	91%

Не дивно, що враховуючи рівень класифікації 91%, більшість прикладів стосується ідентичності, яка відповідає правильній класифікації; цікавішими є неправильно класифіковані приклади. Найпоширеніша плутанина полягає в тому, що 4 57 разів ідентифікується як 9; 7 ідентифікується приблизно 40 разів як 9; а 7 приблизно 26 разів ідентифікується як 2. Хоча 4 і 9, і 7 і 2 легко сплутати, це не здається

відразу очевидним. що 7 можна помилково прийняти за 9. Ймовірне пояснення: часто єдиною відмінною рисою між неправильно класифікованими 7 і типовими 9 є те, що середній горизонтальний штрих у 7 не пов’язаний з верхнім штрихом, а це означає, що нейрони, які мають сприйнятливе поле 9, також мають певну ймовірність спрацьовування.

Із табл. 4.1 видно, що розроблена програма має достовірність розпізнавання рукописних цифр 91%, а програма-аналог MyScript Стилус має достовірність розпізнавання рукописних цифр 84%.

Таким чином, можна зробити висновок, що розроблена програма має порівняно з програмою-аналогом MyScript Стилус збільшену на 7% достовірність розпізнавання рукописних цифр. Тобто мета роботи досягнута – достовірність розпізнавання підвищена.

4.3 Висновок до розділу 4

У четвертому розділі в результаті тестування програми було доведено її повну працездатність та відповідність поставленому завданню. Навчання програми відбувалось з використанням бази даних MNIST. Навчальна вибірка складалась із 40000 зображень цифр. Тестова вибірка складалась із 1000 зображень цифр. Розроблена програма має достовірність розпізнавання рукописних цифр 91%, а програма-аналог MyScript Стилус має достовірність розпізнавання рукописних цифр 84%. Таким чином, розроблена програма має порівняно з програмою-аналогом MyScript Стилус збільшену на 7% достовірність розпізнавання рукописних цифр. Тобто мета роботи досягнута – достовірність розпізнавання підвищена.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення комерційного та технологічного аудиту інформаційної технології розпізнавання рукописних цифр спайкінговою нейронною мережею.

Метою проведення комерційного і технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності [23,24], тобто під час виконання магістерської кваліфікаційної роботи.

Для проведення комерційного та технологічного аудиту залучаємо 3-х незалежних експертів, якими є провідні викладачі випускової або спорідненої кафедри.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснюємо із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, а результати зводимо до таблиці 5.1.

Таблиця 5.1 – Результати оцінювання науково-технічного рівня і комерційного потенціалу

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Технічна здійсненність концепції	2	2	1
Ринкові переваги (наявність аналогів)	3	2	2
Ринкові переваги (ціна продукту)	2	3	3
Ринкові переваги (технічні властивості)	1	2	2
Ринкові переваги (експлуатаційні витрати)	2	2	3
Ринкові перспективи (розмір ринку)	3	3	1
Ринкові перспективи (конкуренція)	4	2	2
Практична здійсненність (наявність фахівців)	3	3	3

Продовження таблиці 5.1

Критерії	Експерти		
	Експерт 1	Експерт 2	Експерт 3
	Бали, виставлені експертами		
Практична здійсненність (наявність фінансів)	2	2	2
Практична здійсненність (необхідність нових матеріалів)	2	4	4
Практична здійсненність (термін реалізації)	2	3	2
Практична здійсненність (розробка документів)	3	2	2
Сума балів	29	30	27
Середньоарифметична сума балів, СБ	29		

За результатами розрахунків, наведених в таблиці 5.1 робимо висновок про те, що науково-технічний рівень та комерційний потенціал інформаційної технології розпізнавання рукописних цифр спайкінговою нейронною мережею – середній [23].

5.2 Розрахунок витрат на здійснення науково-дослідної роботи

Витрати на оплату праці. Належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці, також будь-які види грошових і матеріальних доплат, які належать до елемента «Витрати на оплату праці».

Основна заробітна плата дослідників. Витрати на основну заробітну плату дослідників ($З_о$) розраховують відповідно до посадових окладів працівників [23,24], за формулою:

$$З_о = \sum_{i=1}^{к} \frac{М_{ni} \cdot t_i}{Т_р},$$

де к – кількість посад дослідників, залучених до процесу дослідження; $М_{ni}$ – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.; $Т_р$ – число робочих днів в місяці; приблизно $Т_р$ = (21…23) дні; t_i – число робочих днів роботи розробника (дослідника).

Зроблені розрахунки зводимо до таблиці 5.2.

Таблиця 5.2 – Витрати на заробітну плату дослідників

Посада	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник	41000	1952	10	19520
Розробник	37000	1762	22	38764
Всього:	58284			

Додаткова заробітна плата. Додаткова заробітна плата $З_д$ всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$З_д = 0{,}1 \cdot (З_о + З_р) = 0{,}1 \cdot (38764 + 0) = 3876{,}4 \text{ грн.}$$

Відрахування на соціальні заходи. Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{зп} = \beta \cdot (З_о + З_р + З_д) =$$
$$= 0{,}22 \cdot (38764 + 0 + 3876{,}4) = 9381 \text{ грн.}$$

де $З_о$ – основна заробітна плата розробників, грн.; $З_р$ – основна заробітна плата робітників, грн.; $З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.; β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % (приймаємо для 1-го класу професійності ризику 22%).

Програмне забезпечення. До балансової вартості програмного забезпечення входять витрати на його інсталяцію, тому ці витрати беруться додатково в розмірі 10…12% від вартості програмного забезпечення. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{1}^{к} Ц_{іпрг} \cdot С_{прг.і} \cdot К_і,$$

де $Ц_{іпрг}$ – ціна придбання програмного забезпечення і-го виду, грн.; $С_{прг.і}$ – кількість одиниць програмного забезпечення відповідного виду, шт.; $К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного забезпечення, $К_і$ = (1,1…1,12); к – кількість видів програмного забезпечення.

Таблиця 5.3 – Витрати на придбання програмного забезпечення

Найменування програмного забезпечення	Ціна за одиницю, грн.	Витрачено	Вартість програмного забезпечення, грн.
Visual Studio Code, мова програмування Python	0	1	0
Всього, з врахуванням коефіцієнта інсталяції та налагодження			0

Амортизація обладнання. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час (чи для) виконання даного етапу роботи.

У спрощеному вигляді амортизаційні відрахування А в цілому бути розраховані за формулою:

$$А = \frac{Ц_б}{Т_в} \cdot \frac{t}{12},$$

де $Ц_б$ – загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн.; t – термін використання основного фонду, місяці; $Т_в$ – термін корисного використання основного фонду, роки.

Таблиця 5.4 – Амортизаційні відрахування за видами основних фондів

Найменування	Балансова вартість, грн.	Строк корисного використання, років	Термін використання, місяців	Сума амортизації, грн.
Ноутбук	39000	5	1	650
Всього	650			

Витрати на електроенергію для науково-виробничих цілей. Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою:

Таблиця 5.5 – Витрати на електроенергію

Найменування обладнання	Потужність, кВт	Тривалість годин роботи
Ноутбук		

$$Ве = \sum \frac{W_i \cdot t_i \cdot Ц_е \cdot К_{впі}}{ККД} = \frac{0{,}17 \cdot 150 \cdot 7{,}5 \cdot 0{,}95}{0{,}98} = 185 \text{ грн.},$$

W_i – встановлена потужність обладнання, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год.; Це – вартість 1 кВт електроенергії, грн.; $К_{впі}$ – коефіцієнт використання потужності; ККД – коефіцієнт корисної дії обладнання.

Інші витрати. До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$Ів = \left(З_о + З_р\right) \cdot \frac{Н_{ів}}{100\%} = (38764 + 0) \cdot \frac{90}{100} = 34888 \text{ грн.},$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

Накладні (загальновиробничі) витрати. До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов'язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов'язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов'язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$\text{Внзв} = \left(З_о + З_р\right) \cdot \frac{Н_{нзв}}{100\%} = (38764 + 0) \cdot \frac{150}{100} = 58146 \text{ грн.},$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Витрати на проведення науково-дослідної роботи. Витрати на проведення науково-дослідної роботи розраховуються як сума всіх попередніх статей витрат [23,24] за формулою:

$$В_{заг} = З_о + З_р + З_{дод} + З_н + К_в + В_{спец} + В_{прг} + А_{обл} + В_е +$$
$$+ I_в + В_{нзв} = 38764 + 3876{,}4 + 9381 + 650 + 185 + 34888 + 58146$$
$$= 145890 \text{ грн.}$$

Загальні витрати. Загальні витрати ЗВ на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} = \frac{145890}{0,2} = 729452 \text{ грн.},$$

де η – коефіцієнт, що характеризує етап виконання науково-дослідної роботи. Оскільки, якщо науково-технічна розробка знаходиться на стадії технічного проектування, то η=0,2.

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

В даному випадку відбувається розробка засобу, тому основу майбутнього економічного ефекту буде формувати: ΔN – збільшення кількості споживачів, яким надається відповідна інформаційна послуга в аналізовані періоди часу; N – кількість споживачів, яким надавалась відповідна інформаційна послуга у році до впровадження результатів нової науково-технічної розробки; $Ц_б$ – вартість послуги у році до впровадження інформаційної системи; $\pm\Delta Ц_о$ – зміна вартості послуги (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу.

Можливе збільшення чистого прибутку у потенційного інвестора ΔПі для кожного із років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховується за формулою:

$$\Delta П = (\pm \Delta Ц_о \cdot N + Ц_о \cdot \Delta N_i)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right),$$

де ±ΔЦ – зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай, таким показником може бути зміна ціни реалізації одиниці нової розробки в аналізованому році (відносно року до впровадження цієї розробки); $\pm\Delta Ц_о$ може мати як додатне, так і від'ємне значення (від'ємне – при зниженні ціни відносно року до впровадження цієї розробки, додатне – при зростанні ціни); N – основний кількісний показник, який визначає величину попиту на аналогічні чи подібні розробки у році до впровадження результатів нової науково-технічної розробки; $Ц_о$ – основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році; $Ц_б$ – основний якісний показник, який визначає ціну реалізації існуючої (базової) науково-технічної розробки у році до впровадження результатів; ΔN – зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році. Зазвичай таким показником може бути зростання попиту на науково-технічну розробку в аналізованому році (відносно року до впровадження цієї розробки); λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2023 році ставка податку на додану вартість становить 20%, а коефіцієнт λ =0,8333; ρ – коефіцієнт, який враховує рентабельність інноваційного продукту (послуги). Рекомендується брати ρ =0,2...0,5; ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2023 році ϑ =18%.

Очікуваний термін життєвого циклу розробки 1 рік, тому:

$$\Delta П = ((4800 - 4200) \cdot 10000 - (10000 - 1000) \cdot 4200) \cdot 0{,}8333 \cdot 0{,}3 \cdot \left(1 - \frac{18}{100}\right) = 6716160 \text{ грн.}$$

Далі розраховують приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} = \frac{6716160}{(1+0{,}1)^1} = 6105600 \text{ грн.},$$

де ΔП – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн.; Т – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки (приймаємо Т=1 рік); τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

Далі розраховують величину початкових інвестицій PV , які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} \cdot ЗВ = 5 \cdot 729452 = 3647260 \text{ грн.}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$=2...5, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

Тоді абсолютний економічний ефект $Е_{абс}$ або чистий приведений дохід для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{абс} = ПП - PV = 6105600 - 3647260 = 2458340 \text{ грн.},$$

де ПП – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, грн.; PV – теперішня вартість початкових інвестицій, грн.

Оскільки $Е_{абс}>0$, то можемо припустити про потенційну зацікавленість інвесторів у розробці.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність Ев або показник внутрішньої норми дохідності вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Внутрішня економічна дохідність інвестицій Ев, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки, розраховується за формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Еабс}{PV}} = \sqrt[1]{1 + \frac{2458340}{3647260}} = 1{,}26,$$

де $Т_ж$ – життєвий цикл розробки, роки.

Визначимо бар'єрну ставку дисконтування $\tau_{мін}$, тобто мінімальну внутрішню економічну дохідність інвестицій, нижче якої кошти у впровадження науково-технічної розробки та її комерціалізацію вкладатися не будуть.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$ визначається за формулою:

$$\tau_{мін} = d + f = 0{,}9 + 0{,}05 = 0{,}95,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d =0,9...0,12; f – показник, що характеризує ризикованість вкладення інвестицій; зазвичай величина f =0,05...0,5, але може бути і значно вищою.

Оскільки $E_в$=1,26>$\tau_{мін}$=0,95, то потенційний інвестор може бути зацікавлений у фінансуванні впровадження науково-технічної розробки та виведенні її на ринок, тобто в її комерціалізації.

Далі розраховуємо період окупності інвестицій $T_о$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_о = \frac{1}{E_в} = \frac{1}{1{,}26} = 0{,}79 \text{ року.}$$

Оскільки $T_о$=0,79<1…3-х років, то це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження цієї розробки та виведення її на ринок.

5.4 Висновок до розділу 5

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія розпізнавання рукописних цифр спайкінговою нейронною мережею» становить 29,0 балів, що вказує на комерційну вагомість проведення даних досліджень (рівень комерційного потенціалу розробки середній). Термін окупності становить 0,79 р., що менше 3-х років, що говорить про комерційну привабливість науково-технічної розробки і може спонукати потенційних інвесторів на фінансування впровадження цієї розробки та виведення її на ринок. Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія розпізнавання рукописних цифр спайкінговою нейронною мережею».

ВИСНОВКИ

У роботі було розглянуто задачу розпізнавання рукописних цифр на основі спайкінгової нейронної мережі. В ході аналізу предметної області було розглянуто постановку задачі розпізнавання рукописних цифр, проведено огляд відомих методів розпізнавання зображень, які можна використовувати для поставленої задачі. Обґрунтовано доцільність використання нейронних мереж, зокрема спайкінгових нейронних мереж для розпізнавання рукописних цифр. Було проаналізовано різні програмні реалізації розпізнавання рукописних цифр та обрано аналог, головним недоліком якого є невисока достовірність розпізнавання рукописних цифр, що ставить мету дослідження – підвищення достовірності розпізнавання рукописних цифр.

У другому розділі було обгрунтовано вибір спайкінгової нейромережі для задачі розпізнавання рукописних цифр, розроблено архітектуру спайкінгової нейромережі для інформаційної технології розпізнавання рукописних цифр, яка має 784 входи та 2 шари спайкінгових нейронів. Описано математичну модель та порядок функціонування спайкіногової нейроної мережі для розпізнавання рукописних цифр. Розроблено процеси навчання, вхідного кодування та класифікації рукописних цифр спайкінговою нейромережею. Також розглянуто особливості реалізації запропонованої архітектури спайкінгової нейромережі.

У третьому розділі було обґрунтовано вибір мови Python та спеціалізованої бібліотеки BRIAN для програмної реалізації інформаційної технології розпізнавання рукописних цифр. Розроблено алгоритм роботи та UML діаграму класів програми розпізнавання рукописних цифр, Описано основні етапи реалізація та функціонування програми розпізнавання рукописних цифр.

У четвертому розділі в результаті тестування програми було доведено її повну працездатність та відповідність поставленому завданню. Навчання програми відбувалось з використанням бази даних MNIST. Навчальна вибірка складалась із 40000 зображень цифр. Тестова вибірка складалась із 1000 зображень цифр. Розроблена програма має достовірність розпізнавання рукописних цифр 91%, а програма-аналог MyScript Стилус має достовірність розпізнавання рукописних цифр 84%. Таким чином, розроблена програма має порівняно з програмою-аналогом MyScript Стилус збільшену на 7% достовірність розпізнавання рукописних цифр. Тобто мета роботи досягнута – достовірність розпізнавання підвищена.

У п`ятому розділі було визначено рівень комерційного потенціалу розробки за темою «Інформаційна технологія розпізнавання рукописних цифр спайкінговою нейронною мережею» становить 29,0 балів, що вказує на комерційну вагомість проведення даних досліджень (рівень комерційного потенціалу розробки середній). Термін окупності становить 0,79 р., що менше 3-х років, що говорить про комерційну привабливість науково-технічної розробки і може спонукати потенційних інвесторів на фінансування впровадження цієї розробки та виведення її на ринок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. В. Д. Савич, О. К. Колесницький Інформаційна технологія розпізнавання рукописних цифр спайкінговою нейронною мережею/ в Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», Вінниця, 2023, [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/viewFile/19793/16389

2. Вільям Ґібсон Pattern Recognition/ - Penguin, 2011 ISBN 9780241953532

3. Гудфеллоу Я. Глубокое обучение / пер. с анг. А. А. Слинкина. 2-е изд., испр. М.: ДМК Пресс, 2018. 652 с.: цв. ил.

4. Обробка растрових зображень [Електронний ресурс]. Режим доступу: http://pzs.dstu.dp.ua/ComputerGraphics/raster/index.html

5. Руденко О.В. Штучні нейронні мережі: Навчальний посібник / О.В.Руденко, Є.В.Бодянський. - Харків : ТОВ «Компанія СМІТ», 2006. — 404 с. - ISBN 966-8630-73-X.

6. ABBYY FineReader [Електронний ресурс]: Матеріал з сайту ABBYY – режим доступу: https://www.abbyy.com/en-eu/finereader/ .

7. Обзор программы MyScript Stylus [Електронний ресурс]. Режим доступу: http://www.smartphone.ua/soft_test_515.html

8. Kim S., Park S., Na B., Yoon S. Spiking-YOLO: Spiking Neural Network for Energy-Efficient Object Detection. arXiv. 2019 doi: 10.1609/aaai.v34i07.6787.1903.06530.

9. Zhou S., Chen Y., Li X., Sanyal A. Deep SCNN-Based Real-Time Object Detection for Self-Driving Vehicles Using LiDAR Temporal Data. IEEE Access. 2020;8:76903–76912. doi: 10.1109/ACCESS.2020.2990416.

10. Gerstner, Wulfram. (2002). Spiking neuron models : single neurons, populations, plasticity. Kistler, Werner M., 1969-. Cambridge, U.K.: Cambridge University

11. On Competition and Learning in Cortical Structures. Ph.D. thesis, Diss., Eidgenössische Technische Hochschule ETH Zürich, Nr. 20194, 2012.

12. Maass, Wolfgang (1997). "Networks of spiking neurons: The third generation of neural network models". Neural Networks. 10 (9): 1659–1671.

13. Morrison, A., Aertsen, A., and Diesmann, M. (2007). Spike-timing-dependent plasticity in balanced random networks. Neural Comput. 19, 1437–1467. doi: 10.1162/neco.2007.19.6.1437

14. Bi, G.-Q., and Poo, M.-M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. J. Neurosci. 18, 10464–10472.

15. Querlioz, D., Bichler, O., Dollfus, P., and Gamrat, C. (2013). Immunity to device variations in a spiking neural network with memristive nanodevices. Nanotechnol. IEEE Trans. 12, 288–295. doi: 10.1109/TNANO.2013.2250995

16. LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. Proc. IEEE 86, 2278–2324. doi: 10.1109/5.726791

17. Benjamin, B. V., Gao, P., McQuinn, E., Choudhary, S., Chandrasekaran, A. R., Bussat, J., et al. (2014). Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. Proc. IEEE 102, 699–716. doi: 10.1109/JPROC.2014.2313565

18. Galluppi, F., Lagorce, X., Stromatias, E., Pfeiffer, M., Plana, L. A., Furber, S. B., et al. (2014). A framework for plasticity implementation on the spinnaker neural architecture. Front. Neurosci. 8:429. doi: 10.3389/fnins.2014.00429

19. Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. Science 345, 668–673. doi: 10.1126/science.1254642

20. Habenschuss, S., Bill, J., and Nessler, B. (2012). “Homeostatic plasticity in bayesian spiking networks as expectation maximization with posterior constraints,” in Advances in Neural Information Processing Systems, eds P. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Lake Tahoe, CA: Nips Foundation), 773–781.

21. Python [Електронний ресурс] – режим доступу: https://uk.wikipedia.org/wiki/Python .

22. BRIAN [Електронний ресурс] – режим доступу: https://briansimulator.org/

23. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

24. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

25. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 122 «Комп’ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. К. Колесницький. – Вінниця : ВНТУ, 2023. – (58 с.)

Додаток А (обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: Інформаційна технологія розпізнавання рукописних цифр спайкінговою нейронною мережею

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра комп'ютерних наук, ФІТА
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 87,7% Схожість 12,3%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Савич В.Д.

Керівник роботи Колесницький О.К.

Опис прийнятого рішення

Магістерську кваліфікаційну роботу допущено до захисту

Особа, відповідальна за перевірку Озеранський В.С.

Додаток Б (обов'язковий)

Лістинг програми

```
Convolutional spiking neural network training, testing, and evaluation script.
Evaluation can be done outside of this script; however, it is most
straightforward to call this
script with mode=train, then mode=test on HPC systems, where in the test mode,
the network evaluation is written to disk.
'''

import warnings
warnings.filterwarnings('ignore')

import matplotlib.cm as cmap
import brian_no_units
import networkx as nx
import cPickle as p
import pandas as pd
import numpy as np
import brian as b
import argparse
import random
import timeit
import time
import math
import os

from mpl_toolkits.axes_grid1 import make_axes_locatable
from scipy.spatial.distance import euclidean
from sklearn.metrics import confusion_matrix
from struct import unpack
from brian import *

from util import *
import pickle

np.set_printoptions(threshold=np.nan, linewidth=200)

# only show log messages of level ERROR or higher
b.log_level_error()

# set these appropriate to your directory structure
top_level_path = os.path.join('..', '..')
MNIST_data_path = os.path.join(top_level_path, 'data')
model_name = 'csnn_pc_mnist'
results_path = os.path.join(top_level_path, 'results', model_name)

performance_dir = os.path.join(top_level_path, 'performance', model_name)
activity_dir = os.path.join(top_level_path, 'activity', model_name)
```

```
weights_dir = os.path.join(top_level_path, 'weights', model_name)
best_weights_dir = os.path.join(weights_dir, 'best')
end_weights_dir = os.path.join(weights_dir, 'end')

assignments_dir = os.path.join(top_level_path, 'assignments', model_name)
best_assignments_dir = os.path.join(assignments_dir, 'best')
end_assignments_dir = os.path.join(assignments_dir, 'end')

misc_dir = os.path.join(top_level_path, 'misc', model_name)
best_misc_dir = os.path.join(misc_dir, 'best')
end_misc_dir = os.path.join(misc_dir, 'end')

ngram_dir = os.path.join(top_level_path, 'ngrams', model_name)

for d in [ performance_dir, activity_dir, weights_dir, misc_dir, best_misc_dir,
                assignments_dir, best_assignments_dir, MNIST_data_path,
results_path,
            best_weights_dir, end_weights_dir, end_misc_dir, end_assignments_dir,
ngram_dir ]:
    if not os.path.isdir(d):
        os.makedirs(d)


def normalize_weights():
    '''
    Squash the input to excitatory synaptic weights to sum to a prespecified
number.
    '''
    for conn_name in input_connections:
        connection = input_connections[conn_name][:].todense()
        for feature in xrange(conv_features):
            feature_connection = connection[:, feature * n_e : (feature + 1) *
n_e]
            column_sums = np.sum(np.asarray(feature_connection), axis=0)
            column_factors = weight['ee_input'] / column_sums

            for n in xrange(n_e):
                dense_weights = input_connections[conn_name][:, feature * n_e +
n].todense()
                dense_weights[convolution_locations[n]] *= column_factors[n]
                input_connections[conn_name][:, feature * n_e + n] =
dense_weights


def plot_input(rates):
    '''
    Plot the current input example during the training procedure.
    '''
    fig = plt.figure(fig_num, figsize = (5, 5))
```

```
    im = plt.imshow(rates.reshape((28, 28)), interpolation='nearest', vmin=0,
vmax=64, cmap='binary')
    plt.colorbar(im)
    plt.title('Current input example')
    fig.canvas.draw()
    return im, fig


def update_input(rates, im, fig):
    '''
    Update the input image to use for input plotting.
    '''
    im.set_array(rates.reshape((28, 28)))
    fig.canvas.draw()
    return im


def update_assignments_plot(assignments, im):
    im.set_array(assignments.reshape((int(np.sqrt(n_e_total)),
int(np.sqrt(n_e_total)))).T)
    return im


def get_2d_input_weights():
    '''
    Get the weights from the input to excitatory layer and reshape it to be two
    dimensional and square.
    '''
    # specify the desired shape of the reshaped input -> excitatory weights
    rearranged_weights = np.zeros((conv_features * conv_size, conv_size * n_e))

    # get the input -> excitatory synaptic weights
    connection = input_connections['XeAe'][:]

    for n in xrange(n_e):
        for feature in xrange(conv_features):
            temp = connection[:, feature * n_e + (n // n_e_sqrt) * n_e_sqrt + (n
% n_e_sqrt)].todense()
            rearranged_weights[ feature * conv_size : (feature + 1) * conv_size,
n * conv_size : (n + 1) * conv_size ] = \
                                                        temp[convolution_loca
tions[n]].reshape((conv_size, conv_size))

    if n_e == 1:
        ceil_sqrt = int(math.ceil(math.sqrt(conv_features)))
        square_weights = np.zeros((28 * ceil_sqrt, 28 * ceil_sqrt))

        for n in xrange(conv_features):
            square_weights[(n // ceil_sqrt) * 28 : ((n // ceil_sqrt) + 1) * 28,
                           (n % ceil_sqrt) * 28 : ((n % ceil_sqrt) + 1) * 28] =
rearranged_weights[n * 28 : (n + 1) * 28, :]
```

```
        return square_weights.T
    else:
        square_weights = np.zeros((conv_size * features_sqrt * n_e_sqrt,
conv_size * features_sqrt * n_e_sqrt))

        for n_1 in xrange(n_e_sqrt):
            for n_2 in xrange(n_e_sqrt):
                for f_1 in xrange(features_sqrt):
                    for f_2 in xrange(features_sqrt):
                        square_weights[conv_size * (n_2 * features_sqrt + f_2) :
conv_size * (n_2 * features_sqrt + f_2 + 1), \
                                       conv_size * (n_1 * features_sqrt + f_1) :
conv_size * (n_1 * features_sqrt + f_1 + 1)] = \
                                       rearranged_weights[(f_1 * features_sqrt + f_2) *
conv_size : (f_1 * features_sqrt + f_2 + 1) * conv_size, \
                                                  (n_1 * n_e_sqrt + n_2) * conv_size : (n_1
* n_e_sqrt + n_2 + 1) * conv_size]

        return square_weights.T


def get_input_weights(weight_matrix):
    '''
    Get the weights from the input to excitatory layer and reshape it to be two
    dimensional and square.
    '''
    weights = []

    # for each convolution feature
    for feature in xrange(conv_features):
        # for each excitatory neuron in this convolution feature
        for n in xrange(n_e):
            temp = weight_matrix[:, feature * n_e + (n // n_e_sqrt) * n_e_sqrt +
(n % n_e_sqrt)]
            weights.append(np.ravel(temp[convolution_locations[n]]))

    # return the rearranged weights to display to the user
    return weights


def plot_weights_and_assignments(assignments):
    '''
    Plot the weights from input to excitatory layer to view during training.
    '''
    weights = get_2d_input_weights()

    fig = plt.figure(fig_num, figsize=(18, 9))

    ax1 = plt.subplot(121)
```

```
    image1 = ax1.imshow(weights, interpolation='nearest', vmin=0, vmax=wmax_ee,
cmap=cmap.get_cmap('hot_r'))
    ax1.set_title(ending.replace('_', ' '))
    ax1.set_xticks(()); ax1.set_yticks(())

    ax2 = plt.subplot(122)
    color = plt.get_cmap('RdBu', 11)
    reshaped_assignments = assignments.reshape((int(np.sqrt(n_e_total)),
int(np.sqrt(n_e_total)))).T
    image2 = ax2.matshow(reshaped_assignments, cmap=color, vmin=-1.5, vmax=9.5)
    ax2.set_title('Neuron labels')
    ax2.set_xticks(()); ax2.set_yticks(())

    divider1 = make_axes_locatable(ax1)
    divider2 = make_axes_locatable(ax2)
    cax1 = divider1.append_axes("right", size="5%", pad=0.05)
    cax2 = divider2.append_axes("right", size="5%", pad=0.05)

    plt.colorbar(image1, cax=cax1)
    plt.colorbar(image2, cax=cax2, ticks=np.arange(-1, 10))

    if n_e != 1:
        ax1.set_xticks(xrange(conv_size, conv_size * n_e_sqrt * features_sqrt +
1, conv_size), xrange(1, conv_size * n_e_sqrt * features_sqrt + 1))
        ax1.set_yticks(xrange(conv_size, conv_size * n_e_sqrt * features_sqrt +
1, conv_size), xrange(1, conv_size * n_e_sqrt * features_sqrt + 1))
        for pos in xrange(conv_size * features_sqrt, conv_size * features_sqrt *
n_e_sqrt, conv_size * features_sqrt):
            ax1.axhline(pos)
            ax1.axvline(pos)
    else:
        ax1.set_xticks(xrange(conv_size, conv_size * (int(np.sqrt(conv_features))
+ 1), conv_size), xrange(1, int(np.sqrt(conv_features)) + 1))
        ax1.set_yticks(xrange(conv_size, conv_size * (int(np.sqrt(conv_features))
+ 1), conv_size), xrange(1, int(np.sqrt(conv_features)) + 1))

    fig.canvas.draw()
    plt.tight_layout()

    return fig, ax1, ax2, image1, image2


def update_weights_and_assignments(fig, ax1, ax2, im1, im2, assignments,
spike_counts):
    '''
    Update the plot of the weights from input to excitatory layer to view during
training.
    '''
    weights = get_2d_input_weights()
    im1.set_array(weights)
```

```
    reshaped_assignments = assignments.reshape((int(np.sqrt(n_e_total)),
int(np.sqrt(n_e_total)))).T
    im2.set_array(reshaped_assignments)

    for txt in ax2.texts:
        txt.set_visible(False)

    if n_e == 1:
        spike_counts_reshaped = spike_counts.reshape([features_sqrt,
features_sqrt])
        for x in xrange(features_sqrt):
            for y in xrange(features_sqrt):
                c = spike_counts_reshaped[x, y]
                if c > 0:
                    ax2.text(x, y, str(c), va='center', ha='center',
weight='heavy', fontsize=16)
                else:
                    ax2.text(x, y, '', va='center', ha='center')

    fig.canvas.draw()


def get_current_performance(performances, current_example_num):
    '''
    Evaluate the performance of the network on the past 'update_interval'
training
    examples.
    '''
    global input_numbers

    current_evaluation = int(current_example_num / update_interval)
    if current_example_num == num_examples -1:
        current_evaluation+=1
    start_num = current_example_num - update_interval
    end_num = current_example_num

    wrong_idxs = {}
    wrong_labels = {}

    for scheme in performances.keys():
        difference = output_numbers[scheme][start_num : end_num, 0] -
input_numbers[start_num : end_num]
        correct = len(np.where(difference == 0)[0])

        wrong_idxs[scheme] = np.where(difference != 0)[0]
        wrong_labels[scheme] = output_numbers[scheme][start_num : end_num,
0][np.where(difference != 0)[0]]
        performances[scheme][current_evaluation] = correct /
float(update_interval) * 100
```

```
    return performances, wrong_idxs, wrong_labels


def plot_performance(fig_num, performances, num_evaluations):
    '''
    Set up the performance plot for the beginning of the simulation.
    '''
    time_steps = range(0, num_evaluations)

    fig = plt.figure(fig_num, figsize = (12, 4))
    fig_num += 1

    for performance in performances:
        plt.plot(time_steps[:np.size(np.nonzero(performances[performance]))], \
                                    np.extract(np.nonzero(performances[performanc
e]), performances[performance]), label=performance)

    lines = plt.gca().lines

    plt.ylim(ymax=100)
    plt.xticks(xrange(0, num_evaluations + 10, 10), xrange(0, ((num_evaluations +
10) * update_interval), 10))
    plt.legend()
    plt.grid(True)
    plt.title('Classification performance per update interval')

    fig.canvas.draw()

    return lines, fig_num, fig


def update_performance_plot(lines, performances, current_example_num, fig):
    '''
    Update the plot of the performance based on results thus far.
    '''
    performances, wrong_idxs, wrong_labels =
get_current_performance(performances, current_example_num)

    for line, performance in zip(lines, performances):
        line.set_xdata(range((current_example_num / update_interval) + 1))
        line.set_ydata(performances[performance][:(current_example_num /
update_interval) + 1])

    fig.canvas.draw()

    return lines, performances, wrong_idxs, wrong_labels
```

```
def predict_label(assignments, spike_rates, accumulated_rates,
spike_proportions):
    '''
    Given the label assignments of the excitatory layer and their spike rates
over
    the past 'update_interval', get the ranking of each of the categories of
input.
    '''
    output_numbers = {}
    fire_order_calculated = False

    for scheme in voting_schemes:
        summed_rates = [0] * 10
        num_assignments = [0] * 10

        if scheme == 'all':
            for i in xrange(10):
                num_assignments[i] = len(np.where(assignments == i)[0])
                if num_assignments[i] > 0:
                    summed_rates[i] = np.sum(spike_rates[assignments == i]) /
num_assignments[i]

        elif scheme == 'confidence_weighting':
            for i in xrange(10):
                num_assignments[i] = np.count_nonzero(assignments == i)
                if num_assignments[i] > 0:
                    summed_rates[i] = np.sum(spike_rates[assignments == i] *
spike_proportions[(assignments == i).ravel(), i]) / num_assignments[i]

        output_numbers[scheme] = np.argsort(summed_rates)[::-1]

    return output_numbers

def normalize_probability(v):
    # v is a numpy array
    return v / np.sum(v)

def assign_labels(result_monitor, input_numbers, accumulated_rates,
accumulated_inputs):
    '''
    Based on the results from the previous 'update_interval', assign labels to
the
    excitatory neurons.
    '''
    for j in xrange(10):
        num_assignments = len(np.where(input_numbers == j)[0])
        if num_assignments > 0:
            accumulated_inputs[j] += num_assignments
            accumulated_rates[:, j] = accumulated_rates[:, j] * 0.9 + \
```

```
                    np.ravel(np.sum(result_monitor[input_numbers == j], axis=0) /
num_assignments)

    assignments = np.argmax(accumulated_rates, axis=1).reshape((conv_features,
n_e))

    spike_proportions = np.divide(accumulated_rates, np.sum(accumulated_rates,
axis=0))

    return assignments, accumulated_rates, spike_proportions


def build_network():
    global fig_num, assignments

    neuron_groups['e'] = b.NeuronGroup(n_e_total, neuron_eqs_e,
threshold=v_thresh_e, refractory=refrac_e, reset=scr_e, compile=True,
freeze=True)
    neuron_groups['i'] = b.NeuronGroup(n_e_total, neuron_eqs_i,
threshold=v_thresh_i, refractory=refrac_i, reset=v_reset_i, compile=True,
freeze=True)

    for name in population_names:
        print '...Creating neuron group:', name

        # get a subgroup of size 'n_e' from all exc
        neuron_groups[name + 'e'] = neuron_groups['e'].subgroup(conv_features *
n_e)
        # get a subgroup of size 'n_i' from the inhibitory layer
        neuron_groups[name + 'i'] = neuron_groups['i'].subgroup(conv_features *
n_e)

        # start the membrane potentials of these groups 40mV below their resting
potentials
        neuron_groups[name + 'e'].v = v_rest_e - 40. * b.mV
        neuron_groups[name + 'i'].v = v_rest_i - 40. * b.mV

    print '...Creating recurrent connections'

    for name in population_names:
        # if we're in test mode / using some stored weights
        if test_mode:
            # load up adaptive threshold parameters
            if save_best_model:
                neuron_groups['e'].theta = np.load(os.path.join(best_weights_dir,
'_'.join(['theta_A', ending +'_best.npy'])))
            else:
                neuron_groups['e'].theta = np.load(os.path.join(end_weights_dir,
'_'.join(['theta_A', ending +'_end.npy'])))
        else:
```

```
            # otherwise, set the adaptive additive threshold parameter at 20mV
            neuron_groups['e'].theta = np.ones((n_e_total)) * 20.0 * b.mV

        for conn_type in recurrent_conn_names:
            if conn_type == 'ei':
                # create connection name (composed of population and connection
types)
                conn_name = name + conn_type[0] + name + conn_type[1]
                # create a connection from the first group in conn_name with the
second group
                connections[conn_name] =
b.Connection(neuron_groups[conn_name[0:2]], neuron_groups[conn_name[2:4]],
structure='sparse', state='g' + conn_type[0])

                # instantiate the created connection
                for feature in xrange(conv_features):
                    for n in xrange(n_e):
                        connections[conn_name][feature * n_e + n, feature * n_e +
n] = 10.4

            elif conn_type == 'ie':
                # create connection name (composed of population and connections
types)
                conn_name = name + conn_type[0] + name + conn_type[1] + '_' +
ending
                # create a connection from the first group in conn_name with the
second group
                connections[conn_name] =
b.Connection(neuron_groups[conn_name[0:2]], neuron_groups[conn_name[2:4]],
structure='sparse', state='g' + conn_type[0])
                # instantiate the created connection with the 'weightMatrix'
loaded from file
                for feature in xrange(conv_features):
                    for other_feature in xrange(conv_features):
                        if feature != other_feature:
                            for n in xrange(n_e):
                                connections[conn_name][feature * n_e + n,
other_feature * n_e + n] = 17.4

        print '...Creating monitors for:', name

        # spike rate monitors for excitatory and inhibitory neuron populations
        rate_monitors[name + 'e'] = b.PopulationRateMonitor(neuron_groups[name +
'e'], bin=(single_example_time + resting_time) / b.second)
        rate_monitors[name + 'i'] = b.PopulationRateMonitor(neuron_groups[name +
'i'], bin=(single_example_time + resting_time) / b.second)
        spike_counters[name + 'e'] = b.SpikeCounter(neuron_groups[name + 'e'])

        # record neuron population spikes if specified
        if record_spikes or plot:
```

```
            spike_monitors[name + 'e'] = b.SpikeMonitor(neuron_groups[name +
'e'])
            spike_monitors[name + 'i'] = b.SpikeMonitor(neuron_groups[name +
'i'])

    if record_spikes and plot:
        b.figure(fig_num, figsize=(8, 6))

        fig_num += 1

        b.ion()
        b.subplot(211)
        b.raster_plot(spike_monitors['Ae'], refresh=1000 * b.ms, showlast=1000 *
b.ms, title='Excitatory spikes per neuron')
        b.subplot(212)
        b.raster_plot(spike_monitors['Ai'], refresh=1000 * b.ms, showlast=1000 *
b.ms, title='Inhibitory spikes per neuron')
        b.tight_layout()

    # creating Poission spike train from input image (784 vector, 28x28 image)
    for name in input_population_names:
        input_groups[name + 'e'] = b.PoissonGroup(n_input, 0)
        rate_monitors[name + 'e'] = b.PopulationRateMonitor(input_groups[name +
'e'], bin=(single_example_time + resting_time) / b.second)

    # creating connections from input Poisson spike train to excitatory neuron
population(s)
    for name in input_connection_names:
        print '\n...Creating connections between', name[0], 'and', name[1]

        # for each of the input connection types (in this case, excitatory ->
excitatory)
        for conn_type in input_conn_names:
            # saved connection name
            conn_name = name[0] + conn_type[0] + name[1] + conn_type[1]

            # get weight matrix depending on training or test phase
            if test_mode:
                if save_best_model:
                    weight_matrix = np.load(os.path.join(best_weights_dir,
'_'.join([conn_name, ending + '_best.npy'])))
                else:
                    weight_matrix = np.load(os.path.join(end_weights_dir,
'_'.join([conn_name, ending + '_end.npy'])))

            # create connections from the windows of the input group to the
neuron population
            input_connections[conn_name] = b.Connection(input_groups['Xe'],
neuron_groups[name[1] + conn_type[1]], \
```

```
                            structure='sparse', state='g' + conn_type[0],
delay=True, max_delay=delay[conn_type][1])

            if test_mode:
                for feature in xrange(conv_features):
                    for n in xrange(n_e):
                        for idx in xrange(conv_size ** 2):
                            input_connections[conn_name][convolution_locations[n]
[idx], feature * n_e + n] = \
                                                                weight_matrix[convolu
tion_locations[n][idx], feature * n_e + n]
            else:
                for feature in xrange(conv_features):
                    for n in xrange(n_e):
                        for idx in xrange(conv_size ** 2):
                            input_connections[conn_name][convolution_locations[n]
[idx], feature * n_e + n] = (b.random() + 0.01) * 0.3

            if test_mode:
                if plot:
                    plot_weights_and_assignments(assignments)
                    fig_num += 1

        # if excitatory -> excitatory STDP is specified, add it here (input to
excitatory populations)
        if not test_mode:
            print '...Creating STDP for connection', name

            # STDP connection name
            conn_name = name[0] + conn_type[0] + name[1] + conn_type[1]
            # create the STDP object
            stdp_methods[conn_name] = b.STDP(input_connections[conn_name],
eqs=eqs_stdp_ee, \
                            pre=eqs_stdp_pre_ee, post=eqs_stdp_post_ee, wmin=0.,
wmax=wmax_ee)

    print '\n'


def run_train():
    global fig_num, input_intensity, previous_spike_count, rates, assignments,
clusters, cluster_assignments, \
                simple_clusters, simple_cluster_assignments, index_matrix,
accumulated_rates, \
                accumulated_inputs, spike_proportions

    if plot:
        input_image_monitor, input_image = plot_input(rates)
        fig_num += 1
```

```
        weights_assignments_figure, weights_axes, assignments_axes,
weights_image, \
                        assignments_image =
plot_weights_and_assignments(assignments)
        fig_num += 1

    # set up performance recording and plotting
    num_evaluations = int(num_examples / update_interval) + 1
    performances = { voting_scheme : np.zeros(num_evaluations) for voting_scheme
in voting_schemes }
    num_weight_updates = int(num_examples / weight_update_interval)
```

Додаток В (обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗПІЗНАВАННЯ РУКОПИСНИХ ЦИФР СПАЙКІНГОВОЮ НЕЙРОННОЮ МЕРЕЖЕЮ

Виконав: студент 2-го курсу,
групи 1КН-22м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Савич В. Д.
(прізвище та ініціали)

Керівник: к.т.н., професор каф. КН
_____ Колесницький О.К.
(прізвище та ініціали)
«07» 12 2023 р.

Вінниця ВНТУ - 2023 рік

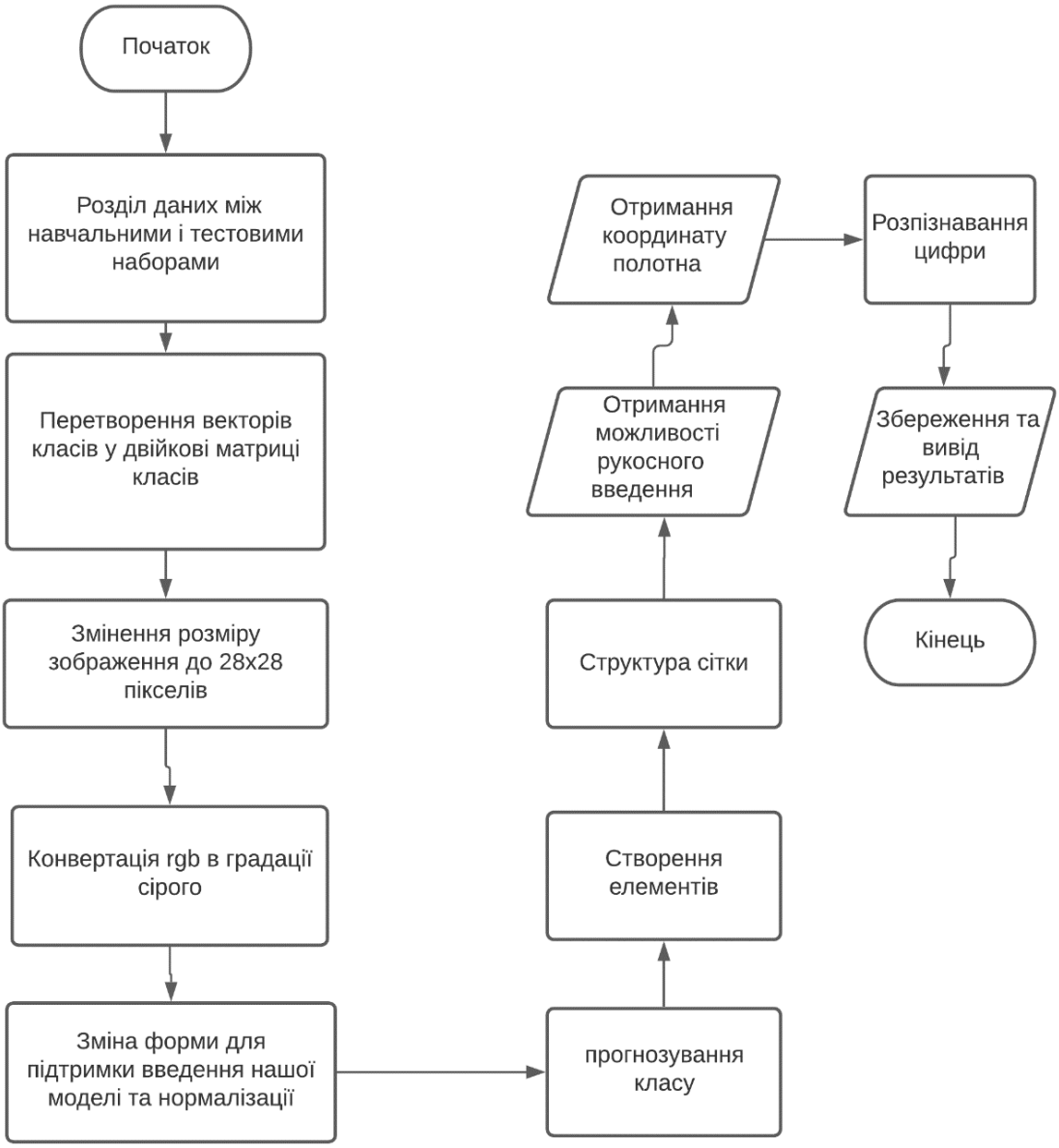

Рисунок В.1 –Загальний алгоритм роботи програми розпізнавання рукописних цифр

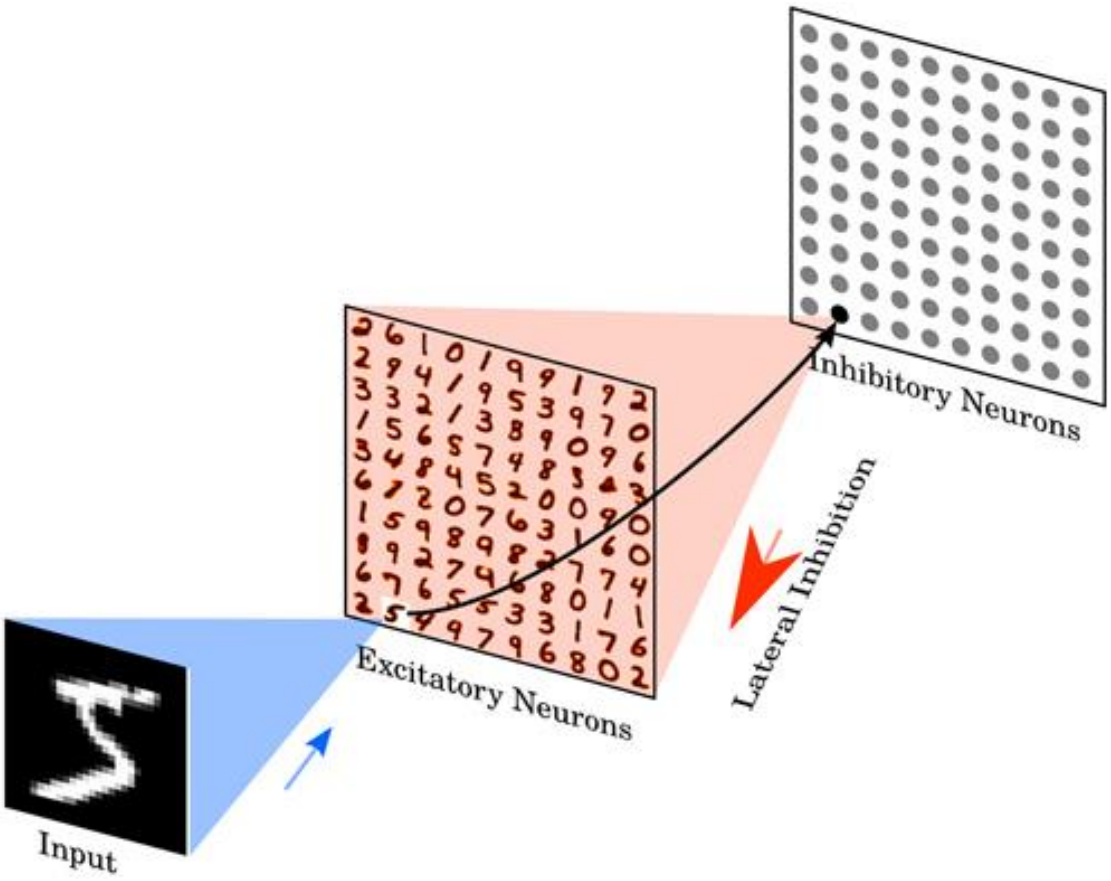

Рисунок В.2 – Архітектура спайкінгової нейромережі

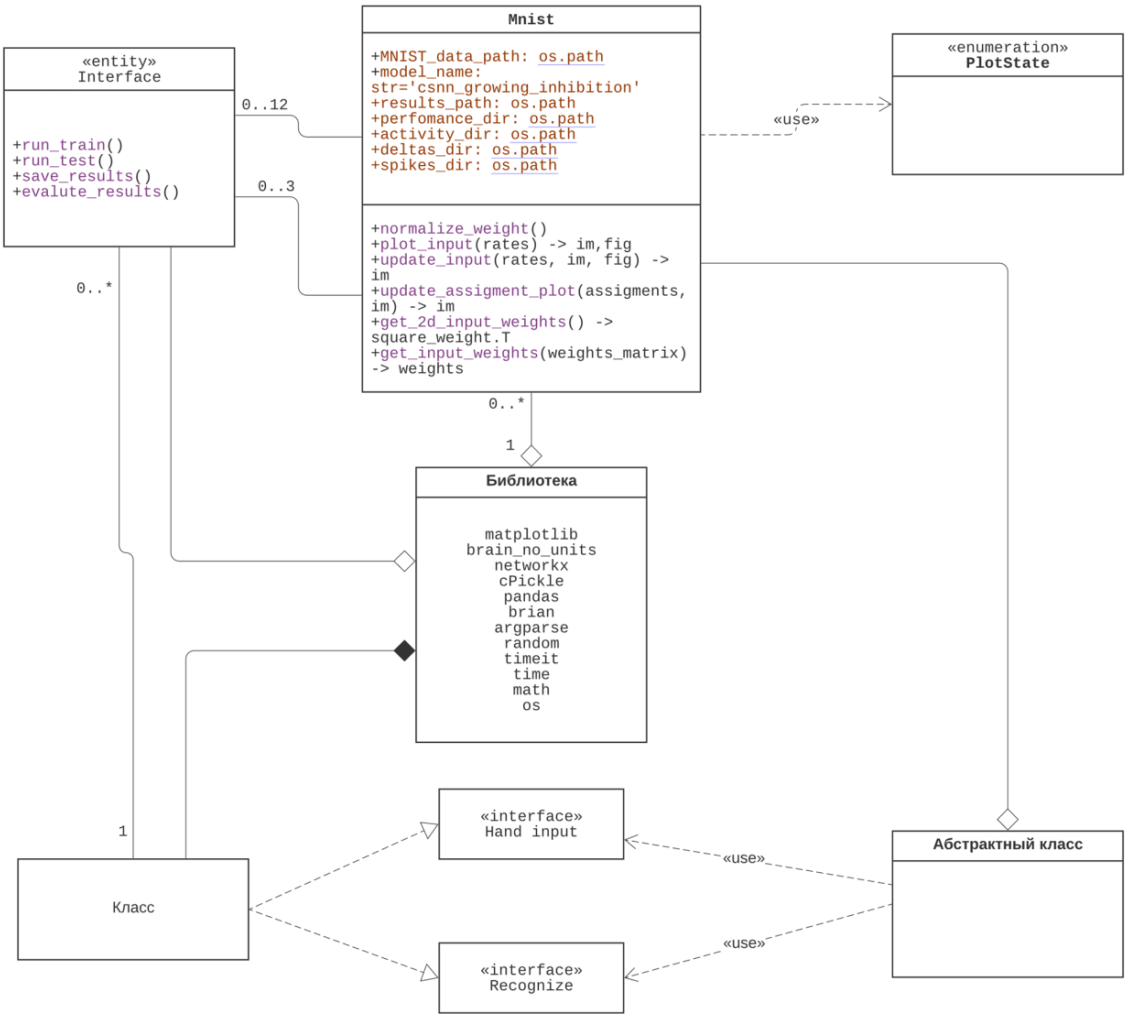

Рисунок В.3 – Діаграма класів програми розпізнавання рукописних цифр на основі спайкінгової нейромережі

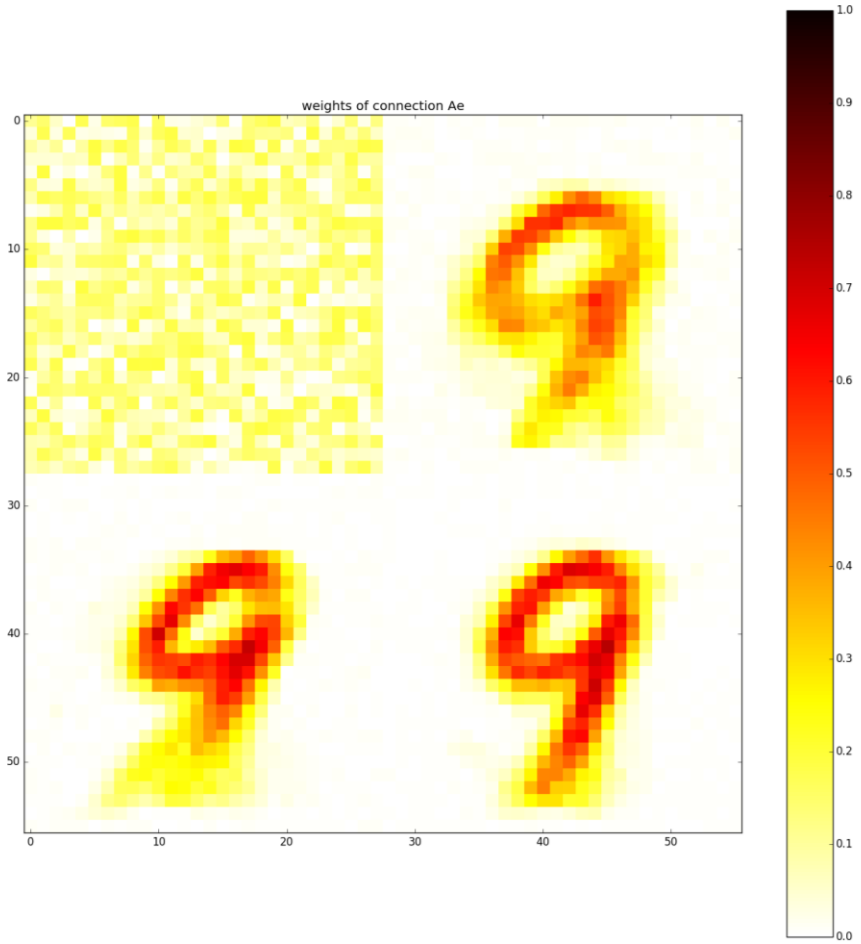

Рисунок В.4 – Вікно програми для вводу рукописних цифр

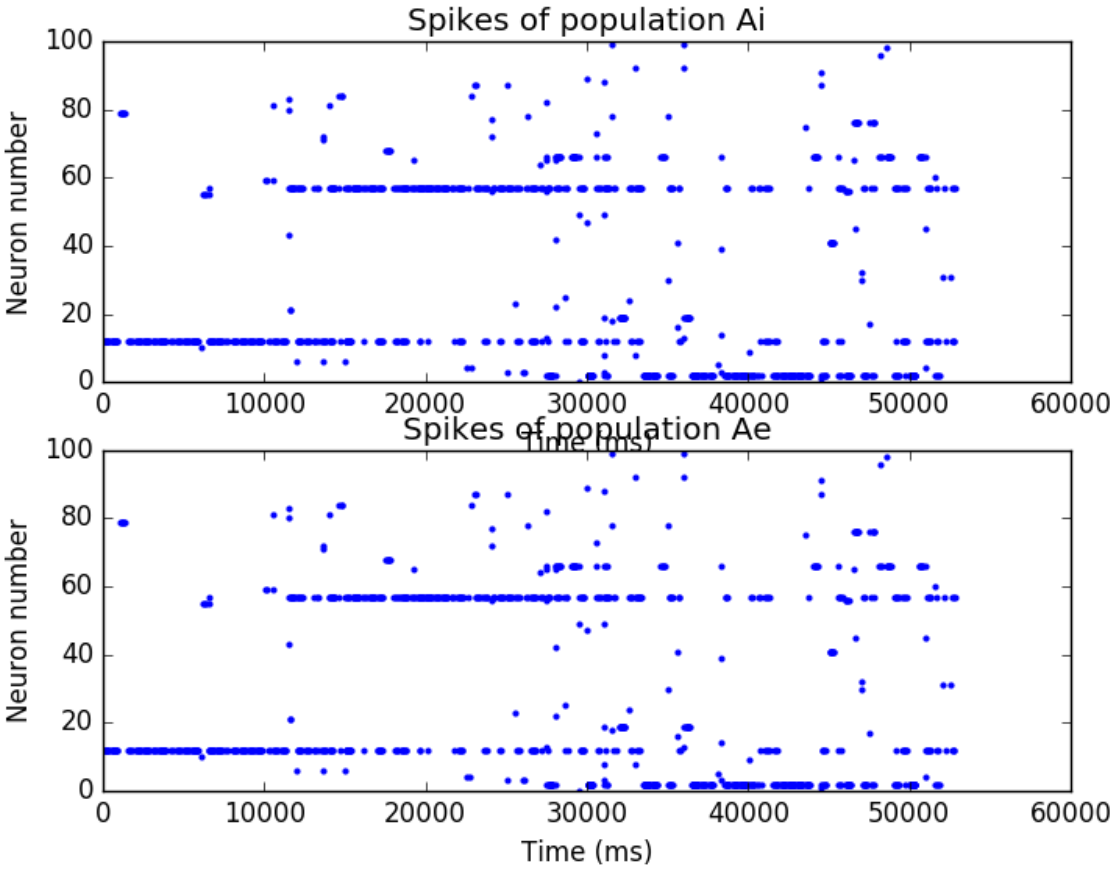

Рисунок В.5 – Вихідні імпульсні сигнали (спайки) нейронів шару гальмівних Ai та збудливих Ae нейронів

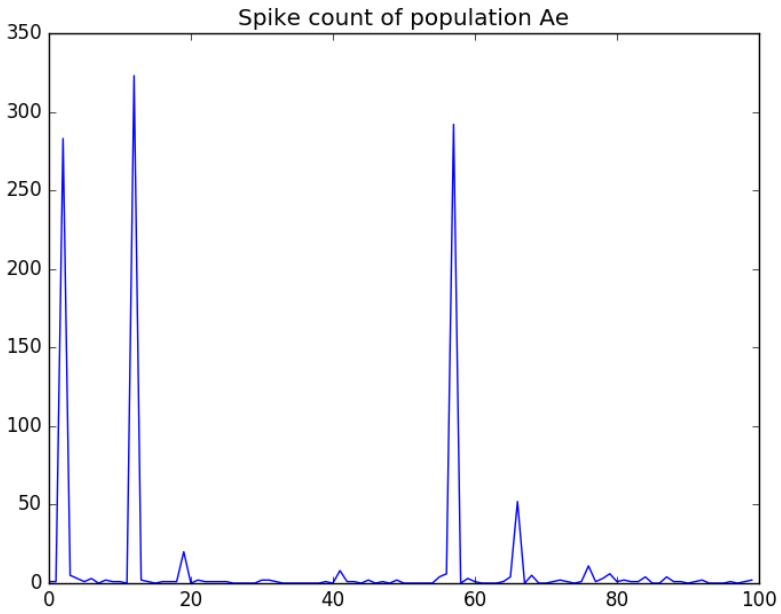

Рисунок В.6 – Графік підрахунку кількості вихідних імпульсів (спайків) нейронів шару збудливих нейронів

	Кількість зображень у тестовій вибірці	Кількість вірно розпізнанних цифр	Достовірність розпізнавання
MyScript Стилус	1000	842	84 %
Розроблена програма	1000	911	91%

Рисунок В.7 – Порівняння параметрів розробленої програми із програмою-аналогом MyScript Стилус

Додаток Г (довідниковий)

Інструкція користувача

Перш за все, завантажте рукописний набір цифр MNIST, запустивши сценарій bash під назвою, `get_MNIST.sh` знайдений у datапапці. Ви можете запустити цей файл, увійшовши `./get_MNIST.sh` в термінал; якщо у вас немає дозволу, виконайте `chmod +x get_MNIST.sh` (рис. Г.1).

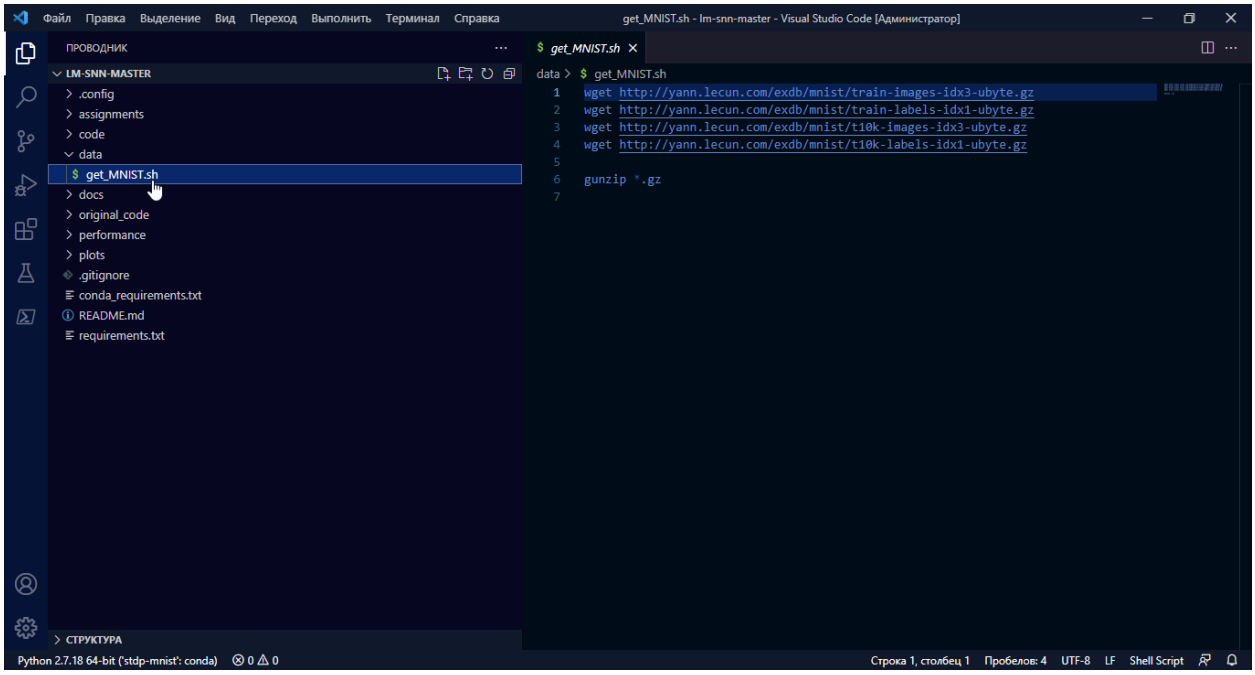

Рисунок Г.1 – Встановлення набору цифр MNIST

Важливо : не забудьте змінити файл matplotlibrc (як правило, розташований у ~/.config/matplotlib/; якщо його там немає, ви можете створити його), щоб додати рядок backend : Qt5Agg. Інакше ви не зможете імпортувати pylabпід час імпорту brian 1.4.3. Базовий сервер за замовчуванням, GKAgg, не працюватиме з цим застарілим модулем!

Клонуйте це сховище в каталог на ваш вибір, позначений <stdp-mnist>. Рекомендується встановити Anaconda2 і зробити його Python за замовчуванням, додавши рядок export PATH=[path to anaconda]/bin:$PATH до ваших .bashrc(для систем *nix), оскільки це спростить встановлення відповідних пакетів. Змініть каталог на <stdp-mnist>, і виконайте pip install -r

requirements.txtв терміналі (у системах *nix) або оболонкі Git Bash (у Windows), а потім виконайте conda install --file conda_requirements.txt.

Щоб навчити мережу, змініть каталог на <stdp-mnist>/code/train, і виберіть один із сценаріїв у ньому для навчання моделі нейронної мережі, що розробляється, на рукописному наборі цифр MNIST. Ми рекомендуємо вам навчити згорткову спайкінгову нейронну мережу з підключенням між патчами (CSNN-PC), яка є найбільш загальною моделлю, оскільки її можна звести до згорткової нейронної мережі без підключення між патчами (CSNN) або просто до згорткової нейронної мережі. Модель нейронної мережі (SNN) в певних особливих випадках. Для цього просто запустіть python csnn_pc_mnist.py для навчання моделі CSNN-PC з параметрами за замовчуванням. Щоб побачити список необов'язкових аргументів, які потрібно вказати в командному рядку, запустіть python csnn_pc_mnist.py –help (рис. Г.2).

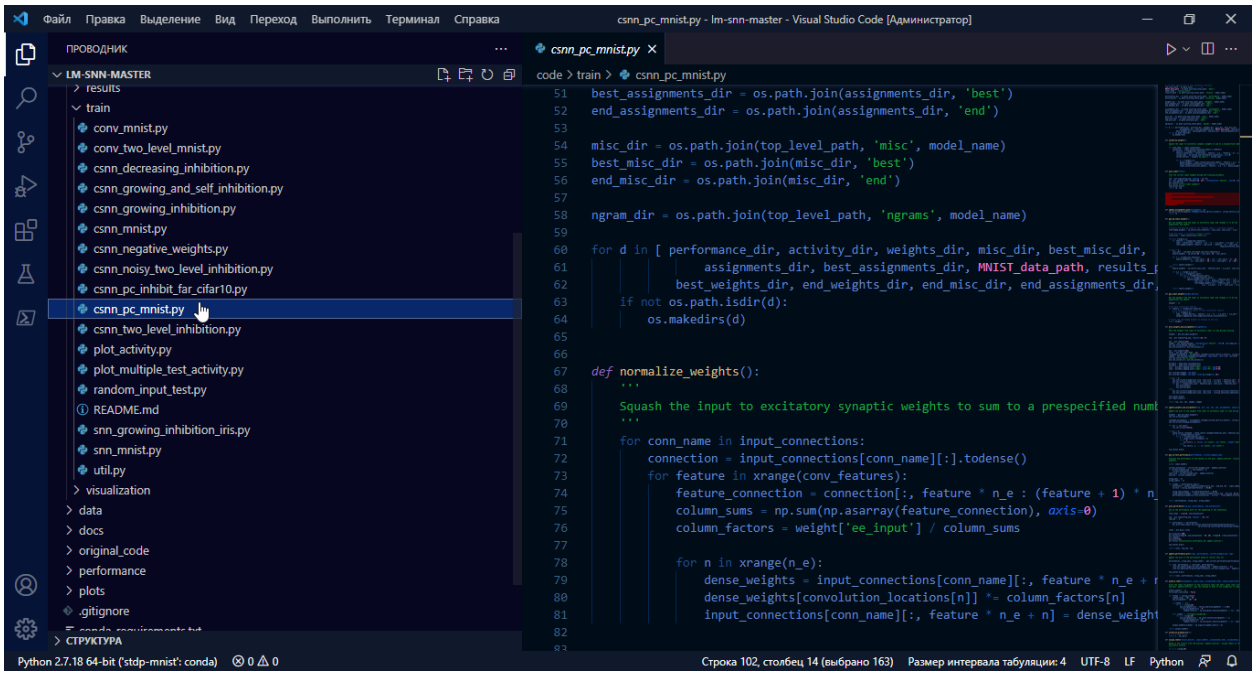

Рисунок А.2 – Запуск навчання мережі

Можливо, найкориснішим аргументом командного рядка є do_plot, який, якщо встановлено як do_plot=True, дозволяє візуалізувати хід навчання мережі, вивчені фільтри згортки та вагові значення підключення між патчами, поточний

вхід до мережі та розподіл «голосів» (тобто, класифікації окремих збуджуючих нейронів) за останній мініпакет (update_interval) даних.

В інших підпапках репозиторію проекту є різні допоміжні сценарії, які дозволяють користувачеві тестувати або візуалізувати параметри навчених моделей, побудовувати криві продуктивності та виконувати завдання на високопродуктивному обчислювальному кластері CICS swarm2.