МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІГІВСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ

СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

Методичні вказівки
до виконання розрахунково-графічної роботи
для студентів за напрямом підготовки (спеціальністю)
6.170103 «Управління інформаційною безпекою»,
125 «Кібербезпека»
денної форми навчання

Обговорено і рекомендовано
на засіданні кафедри
кібербезпеки та математичного моделювання
Протокол № 8
від 19 лютого 2019 р.

Чернігів ЧНТУ 2019

Системи штучного інтелекту. Методичні вказівки до виконання розрахунково-графічної роботи для студентів за напрямом підготовки (спеціальністю) 6.170103 «Управління інформаційною безпекою», 125 «Кібербезпека» денної форми навчання / Укл.: Петренко Т.А. – Чернігів: ЧНТУ, 2019. – 24 с.

Укладач: Петренко Тарас Анатолійович, старший викладач кафедри кібербезпеки та математичного моделювання

Відповідальний за випуск: Ткач Юлія Миколаївна, д.п.н., доц., завідувач кафедри кібербезпеки та математичного моделювання

Рецензент: Ткач Юлія Миколаївна, д.п.н., доц., завідувач кафедри кібербезпеки та математичного моделювання, професор

ЗМІСТ

ПЕРЕДМОВА

Метою викладання навчальної дисципліни “Системи штучного інтелекту” є опанування студентами основ функціонування систем штучного інтелекту, набуття ними навичок їх використання для розв’язання прикладних задач і оволодіння засобами проектування та розробки цих систем. У курсі вивчаються теоретичні та прикладні питання створення інтелектуальних та експертних систем, моделі та методи розв’язання задач, механізми подання знань і виведення нових знань, зокрема, нечітка логіка, дедуктивне виведення, методи навчання, а також механізми обробки невизначеності.

Завдання:

– формування базового уявлення про галузі застосування систем штучного інтелекту;

– набуття вмінь і навичок розв’язання задач з використанням систем штучного інтелекту;

– опанування теоретичних і практичних питань створення та застосування систем штучного інтелекту, експертних систем;

– вивчення механізмів обробки і подання знань в інтелектуальних системах;

– практичне засвоєння механізмів побудови інтелектуальних систем на основі нечіткої логіки;

– вивчення особливостей застосування систем штучного інтелекту в управлінні інформаційною безпекою.

Запропоновані завдання для розрахунково-графічної роботи студентів включають методичні вказівки до виконання, завдання та критерії оцінювання. За допомогою розрахунково-графічної роботи та запропонованого завдання досягається більш глибоке опанування теорії, що здійснюється за допомогою розвитку логічного мислення через вирішення практичної задачі розпізнавання образів за допомогою нейронних мереж та дає змогу студентам закріпити на практиці нові для них поняття.

Завдання для розрахунково-графічної роботи студентів може використовуватися як для аудиторної, так і домашньої роботи. Воно спрямоване на розвиток у студентів організаційних та аналітичних здібностей, а також уміння користуватися теоретичними посиланнями у вирішенні практичних ситуацій та вміння користуватися спеціальною літературою. Завдання для розрахунково-графічної роботи студентів можуть значною мірою полегшити вивчення дисципліни студентами очної форми навчання.

Під час виконання розрахунково-графічної роботи студенти повинні ознайомитися та вивчити лекційний матеріал, запропонований викладачем. Основою для вивчення є літературні джерела, наведені в даній методичній розробці. За наявності незрозумілих питань студентам рекомендується звернутись за консультаціями до викладача з метою отримання всіх необхідних пояснень щодо організації розрахунково-графічної роботи, виконання завдання та пошуку додаткових літературних джерел. Викладачем надаються додаткові роз'яснення та індивідуальні консультації для підвищення компетентності студентів та розширення спектру їх знань з даної дисципліни.

Метою розрахунково-графічної роботи є перевірка рівня засвоєння студентами знань з дисципліни «Системи штучного інтелекту» та вміння самостійно вирішувати поставлені перед ними практичні задачі. Розрахунково-графічна робота виконуються в п'ятому семестрі навчання, після вивчення студентами найважливіших тем з дисципліни.

СТРУКТУРА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Назви змістових модулів і тем	денна форма					
	усього	у тому числі				
		л	п	лаб	інд	с.р.
Змістовий модуль 1. Методи та моделі систем штучного інтелекту						
Тема 1. Загальні відомості про системи штучного інтелекту	7	2	-	-	-	5
Тема 2. Теоретичні основи систем штучного інтелекту	8	-	-	2	-	6
Тема 3. Способи подання інтелектуальних задач і методи пошуку рішень	8	2	-	-	-	6
Тема 4. Нечітка логіка	11	2	-	4	-	5
Тема 5. Генетичні алгоритми	9	2	-	2	-	5
Тема 6. Мурашині алгоритми	5	-	-	-	-	5
Тема 7. Еволюційні моделі	7	2	-	-	-	5
Тема 8. Колективний інтелект	9	2	-	2	-	5
Тема 9. Дерева рішень	6	-	-	-	-	6
Тема 10. Асоціативні правила	6	-	-	-	-	6
Разом за змістовим модулем 1	**76**	**12**	**-**	**10**	**-**	**54**
Змістовий модуль 2. Нейронні мережі в системах штучного інтелекту						
Тема 1. Відомості з біології про будову мозку людини	5	-	-	-	-	5
Тема 2. Штучний нейрон	9	2	-	2	-	5
Тема 3. Класичні нейронні мережі	9	2	-	2	-	5
Тема 4. Прикладне застосування нейронних мереж	14	2	-	6	-	6
Тема 5. Сучасні досягнення в галузі застосування нейронних мереж	7	2	-	-	-	5
Разом за змістовим модулем 2	**44**	**8**	**-**	**10**	**-**	**26**
Усього годин за дисципліну	**120**	**20**	**-**	**20**	**-**	**80**

КРИТЕРІЇ ОЦІНЮВАННЯ РОЗРАХУНКОВО-ГРАФІЧНОЇ РОБОТИ

Розрахунково-графічні робота виконуються за окремим графіком. Студент самостійно готується до такого заняття за індивідуальним завданням. Обсяг розрахунково-графічної роботи визначається навчальним планом з дисципліни.

З даного курсу розрахунково-графічної робота проводиться у формі побудови нейронної мережі за допомогою JavaScript та використання її для розпізнавання образів (реалізація системи розпізнавання рукописних цифр шляхом побудови нейронної мережі на мові програмування JavaScript, використовуючи Git).

Шкала оцінювання знань студентів при виконанні розрахунково-графічної роботи

Рівень виконання розрахункової роботи	**Кількість балів**	
- завдання виконано повністю і правильно, містять пояснення до програмного коду; - здійснено посилання на науково-практичну та технічну базу; - показано вміння самостійно формулювати висновки за результатами виконаного завдання; - присутній творчий підхід та використано новітні інформаційні технології.	9…	10
- завдання виконані повністю, але при розв’язуванні допущені незначні помилки; - не аргументовано викладено матеріал; - у висновках містяться помилки та недоречності	6…	8
- завдання виконано не у повному обсязі та містить грубі помилки; - не сформульовані висновки за результатами розрахунків	3…	5
- завдання виконані частково і неякісно	0…	2

ВИМОГИ ДО ОФОРМЛЕННЯ РОЗРАХУНКОВО-ГРАФІЧНОЇ РОБОТИ

Робота оформляється на листах А4 з однієї сторони, поля: з лівого боку – 20 мм, з правого боку – 10 мм, зверху – 20 мм, знизу – 20 мм. Завдання повинні бути виконані акуратно, розбірливим почерком (або надруковані), з детальними поясненнями та всіма проміжними розрахунками. В кінці розрахункового завдання пишеться висновок (відповідь).

Вимоги до комп'ютерного набору розрахункової роботи:

– текстовий редактор – WORD;

– гарнітура шрифту – Times New Roman;

– кегль шрифту (розмір) – 14;

– міжрядковий інтервал – полуторний;

– абзац – 1,25 см;

– розташування тексту роботи – вирівнювання по ширині;

– міжрядковий інтервал між заголовком (назвою розділу чи підрозділу) і текстом повинна дорівнювати 1 інтервалу.

Приклад оформлення титульної сторінки розрахунково-графічної роботи наведено у Додатку А.

Повністю оформлена і виконана розрахункова робота подається на кафедру в термін, що визначений у плані-графіку виконання розрахункової роботи для перевірки її викладачем. Якщо робота виконана не вчасно без поважних причин, то студенту ставиться 0 балів («незадовільно») і він повинен виконати додатково один з варіантів, який вкаже викладач. Розрахункова робота оцінюється після особистої співбесіди з викладачем. В разі зауважень з боку викладача, робота повинна бути доопрацьована в зазначений термін і подана на перевірку. До підсумкового контролю допускаються лише студенти, що вчасно здали і захистили свою роботу.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Нейронна мережа – це послідовність нейронів, з'єднаних між собою синапсами. Структура нейронної мережі прийшла в світ програмування прямо з біології. Завдяки такій структурі, машина отримує можливість аналізувати і навіть запам'ятовувати різну інформацію. Нейронні мережі також здатні не тільки аналізувати вхідну інформацію, а й відтворювати її зі своєї пам'яті. Нейронні мережі використовуються для вирішення складних завдань, які вимагають аналітичних обчислень подібних тим, що робить людський мозок.
Найпоширенішими застосуваннями нейронних мереж є:

1. Класифікація – розподіл даних по параметрах. Наприклад, на вхід дається набір людей і потрібно вирішити, кому з них давати кредит, а кому ні. Цю роботу може зробити нейронна мережа, аналізуючи таку інформацію як: вік, платоспроможність, кредитна історія і так далі.

2. Передбачення – можливість прогнозувати наступний крок. Наприклад, зростання або падіння акцій, грунтуючись на ситуації на фондовому ринку.

3. Розпізнавання – в даний час, саме широке застосування нейронних мереж. Використовується в Google, коли ви шукаєте фото або в камерах телефонів, коли воно визначає положення вашого обличчя і виділяє його.

Нейрон – це обчислювальна одиниця, яка отримує інформацію, проводить над нею прості обчислення і передає її далі. Вони діляться на три основних типи: вхідний, прихований і вихідний. Також є нейрон зміщення і контекстний нейрон. У тому випадку, коли нейронна мережа складається з великої кількості нейронів, вводять термін шару. Відповідно, є вхідний шар, який отримує інформацію, n прихованих шарів, які її обробляють і вихідний шар, який виводить результат. У кожного з нейронів є 2 параметри: вхідні дані (*input data*) і вихідні дані (*output data*).

В поле *input* потрапляє сумарна інформація всіх нейронів з попереднього шару, після чого, вона нормалізується, за допомогою функції активації *f(x)* і

потрапляє в поле *output*.

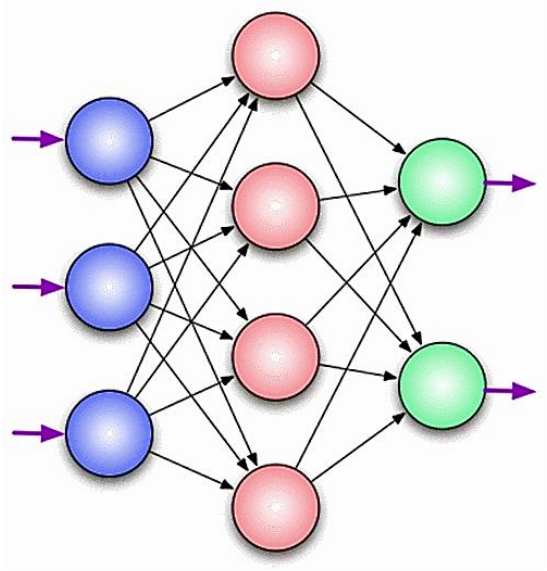


Рисунок 1 – Модель нейронної мережі

Синапс це зв’язок між двома нейронами. У синапсів є 1 параметр – вага. Завдяки йому, вхідна інформація змінюється, коли передається від одного нейрона до іншого. Припустимо, є 3 нейрона, які передають інформацію з наступних підстав. Тоді у нас є 3 ваги, відповідні кожному з цих нейронів. У того нейрона, у якого вага буде більше, та інформація і буде домінуючою в наступному нейроні.

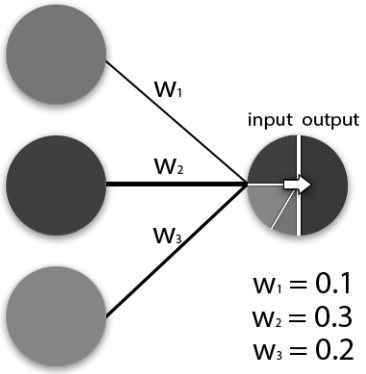


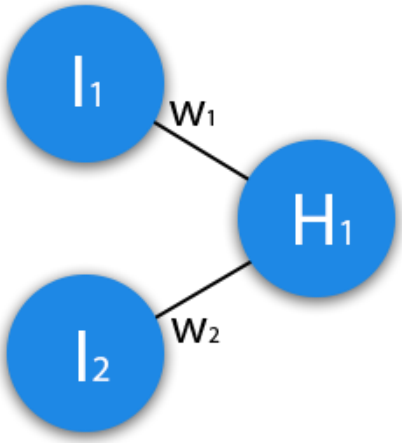
Рисунок 2 – Модель зв’язку між нейронами

Сукупність ваг нейронної мережі або матриця ваг – це своєрідний мозок всієї системи. Саме завдяки цим вагам, вхідна інформація обробляється і перетворюється в результат.

Важливо пам’ятати, що під час ініціалізації нейронної мережі, ваги розставляються в випадковому порядку.

На рисунку 3 зображена частина нейронної мережі, де буквами *I*

позначені вхідні нейрони, буквою *H* – прихований нейрон, а буквою *w* – ваги. З формули видно, що вхідна інформація – це сума всіх вхідних даних, помножених на відповідні їм ваги. Тоді дамо на вхід *1* і *0*. Нехай *w1 = 0.4* і *w2 = 0.7*. Вхідні дані нейрона *H1* будуть наступними: *1 * 0.4 + 0 * 0.7 = 0.4*. Отже, коли у нас є вхідні дані, ми можемо отримати вихідні дані, підставивши вхідне значення в функцію активації. Тепер, коли у нас є вихідні дані, ми передаємо їх далі. І так, ми повторюємо для всіх шарів, поки не дійдемо до вихідного нейрона. Запустивши таку мережу в перший раз ми побачимо, що відповідь далека від правильної, тому що мережа не натренована. Щоб поліпшити результати її необхідно тренувати.



$$1) H_{1_{input}} = (I_1 * w_1) + (I_2 * w_2)$$

$$2) H_{1_{output}} = f_{activation}(H_{1_{input}})$$

Рисунок 3 – Модель роботи нейронної мережі

Функція активації – це спосіб нормалізації вхідних даних. Тобто, якщо на вході буде велике число, пропустивши його через функцію активації, ви отримаєте вихід в потрібному вам діапазоні. Функцій активації досить багато, але найосновніші: лінійна, сигмоід (логістична) і гіперболічний тангенс. Головні їх відмінності – це діапазон значень.

$$f(x) = x$$

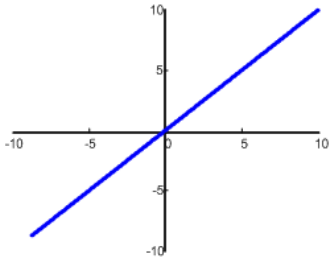


Рисунок 4 – Лінійна функція

Лінійна функція майже ніколи не використовується, за винятком випадків, коли потрібно протестувати нейронну мережу або передати значення без перетворень.

$$f(x) = \frac{1}{1+e^{-x}}$$

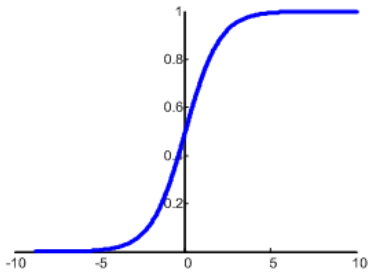


Рисунок 5 – Сигмоід

Сигмоід є найпоширенішою функцією активації, її діапазон значень *[0,1]*. Саме на ній показано більшість прикладів мереж, також її іноді називають логістичною функцією. Але якщо в нашому випадку присутні негативні значення, то знадобиться функція яка їх захоплює.

$$f(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

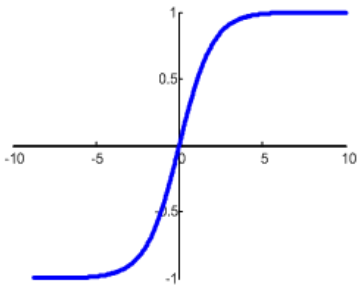


Рисунок 6 – Гіперболічний тангенс

Має сенс використовувати гіперболічний тангенс, тільки тоді, коли значення можуть бути і негативними, і позитивними, так як діапазон функції *[–1,1]*. Використовувати цю функцію тільки з позитивними значеннями недоцільно так як це значно погіршить результати обчислень нейронної мережі.

Тренувальний сет – це послідовність даних, якими оперує нейронна мережа. У випадку виключаючого АБО (*xor*) є всього 4 різних результати. Тобто буде 4 тренувальних сети: *0xor0 = 0*, *0xor1 = 1*, *1xor0 = 1*, *1xor1 = 0*.

Ітерація – це своєрідний лічильник, який збільшується кожного разу, коли нейронна мережа проходить один тренувальний сет. Іншими словами, це загальна кількість тренувальних сетів пройдених нейронною мережею. Повний цикл ітерацій проходить всередині «Епохи». При ініціалізації нейронної мережі ця величина встановлюється в 0 і має найвище значення, що задається вручну. Чим більше епоха, тим краще натренована мережа і відповідно, її результат. Епоха збільшується кожного разу, коли ми проходимо весь набір тренувальних сетів, в випадку виключаючого АБО (*xor*), 4 сетів або 4 ітерацій.

Помилка – це величина у відсотках, що відображає розбіжність між очікуваними і отриманими відповідями. Помилка формується кожну епоху і повинна йти на спад. Якщо цього не відбувається, значить, щось відбувається не так. Помилку можна обчислити різними шляхами, але ми розглянемо три основних способи*: Mean Squared Error* (далі *MSE*), *Root MSE* і *Arctan*.

У випадку рівняння помилки немає якогось обмеження на використання, як у функції активації, й можна вибрати будь–який метод, який буде приносити найкращий результат. Варто лише враховувати, що кожен метод розраховує помилки по різному. У *Arctan*, помилка, майже завжди, буде більше, так як він працює за принципом: чим більша різниця, тим більше помилка. У *Root MSE* буде найменша помилка, тому, найчастіше, використовують *MSE*, яка зберігає баланс в обчисленні помилки.

$$\frac{(i_1-a_1)^2+(i_2-a_2)^2+\ldots+(i_n-a_n)^2}{n}$$

Рисунок 7 – Формула розрахунку помилки *MSE*

$$\sqrt{\frac{(i_1-a_1)^2+(i_2-a_2)^2+\ldots+(i_n-a_n)^2}{n}}$$

Рисунок 8 – Формула розрахунку помилки *Root MSE*

$$\frac{\arctan^2(i_1-a_1)+\ldots+\arctan^2(i_n-a_n)}{n}$$

Рисунок 9 – Формула розрахунку помилки *Arctan*

Принцип розрахунку помилки у всіх випадках однаковий. За кожен сет, ми рахуємо помилку, віднявши від ідеальної відповіді отриману. Далі, зводимо в квадрат, після чого отримане число ділимо на кількість сетів.

ЗАВДАННЯ ДО РОЗРАХУНКОВО-ГРАФІЧНОЇ РОБОТИ

Реалізувати систему розпізнавання образів (рукописних цифр) шляхом побудови нейронної мережі з функцією активації відповідно до варіанту на мові програмування JavaScript, використовуючи Git.

Варіанти функцій активації

1. Лінійна;
2. Напівлінійна;
3. Параметрична лінійна;
4. Рандомізована лінійна;
5. Експоненціальна лінійна;
6. Сигмоїд;
7. Тангенс;
8. Гіпердолічний тангенс;
9. Арктангенс;
10. Обернений квадратний корінь;

МЕТОДИЧНІ ВКАЗІВКИ ДО ВИКОНАННЯ РГР

1. Для початку роботи необхідно завантажити та встановити систему контролю версій *Git*, а саме додаток *Git Bash* для *OS Windows*. Система контролю версій дозволяє завантажувати відкриті (*open source*) проекти з *Github* – співтовариства комерційних та незалежних розробників програмного забезпечення. *Git Bash* розширює можливості командного рядка *Windows*, тому являється незамінним при розробці програмного забезпечення, або веб-проектів. Також необхідна наявність текстового редактору, налаштованого або спеціально розробленого для роботи з мовами програмування.

2. Наступним кроком необхідно завантажити та встановити *NodeJS*, так як до його складу входить власний інсталятор пакетів *npm*, за допомогою якого будемо встановлювати додатки до робочого каталогу (заміною може

слугувати *Bower*).

3. Створити пустий каталог (папку), в якому буде розташований проект. Відкрити всередині каталогу *Git Bash* через контекстне меню *Windows* або перейти до каталогу використовуючи команду *cd* у *Git Bash*.

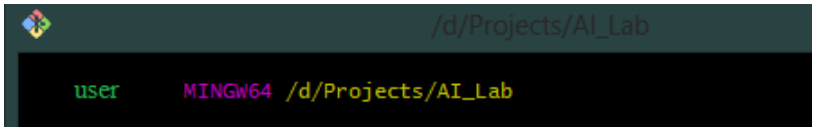


Рисунок 10 – Каталог проекту у Git Bash

4. Для створення нейронної мережі необхідно використовувати бібліотеку *Brain.js*, яка володіє великою швидкістю роботи та можливістю зберігати натреновану модель у файли формату *JSON*. Дана бібліотека написана на мові програмування *JavaScript*, та являє собою фреймворк для створення, навчання та експлуатації нейронних мереж. Встановлюємо її за допомогою команди *npm install brain.js*:

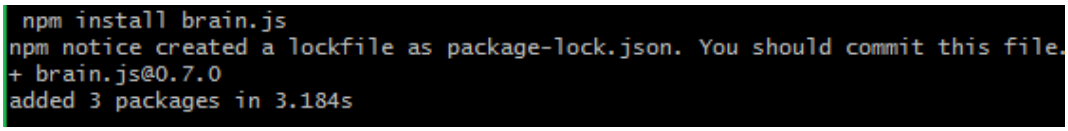


Рисунок 11 – Встановлення *Brain.js*

5. Створити нейронну мережу, за допомогою якої можна буде розпізнати ручне написання цифр від 0 до 9. Для початку необхідно мати зображення рукописних цифр, для цього використати модуль *MNIST digits*, який являє собою набір тисяч бінарних зображень рукописних цифр, розмірами *28*28px*. Оригінальна база даних *MNIST* містить 60 000 прикладів для навчання і 10 000 прикладів для тестування.

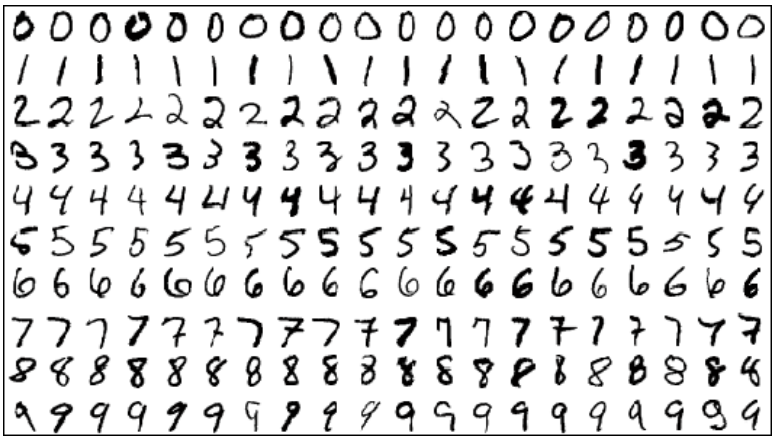


Рисунок 12 – Приклад зображень з *MNIST digits*

Встановити бібліотеку до репозиторію командою *npm install https://github.com/cazala/mnist* .

6. Для навчання мережі необхідно створити у каталозі новий файл з назвою *train.js*, та користуючись можливостями завантажених бібліотек запрограмувати, використовуючи функцію активації відповідно до завдання, навчальний процес. Спочатку створюється мережа, отримується 20000 елементів навчальної вибірки, викликається метод *.train*, який проводить навчання мережі та зберігає все в файл *'./mnistTrain.json'*. Описані операції виконуються кодом, ілюстрованим на рисунку 2.4.

Поріг помилки у прикладі дорівнює 0.001, це означає що навчання нейронної мережі буде завершене, як тільки значення помилки при навчанні стане менше даного порогу. При значенні 0.001 навчання мережі триває приблизно півгодини, при збільшенні порогу помилки, наприклад до 0.01, час навчання буде скорочено до декількох хвилин, але від цього постраждає якість роботи мережі.

```
const brain = require('brain.js');
var net = new brain.NeuralNetwork();
const fs = require('fs');

const mnist = require('mnist');

const set = mnist.set(20000, 0);

const trainingSet = set.training;


net.train(trainingSet,
    {
            errorThresh: 0.001,  // поріг помилки
            iterations: 20000,   // максимальна кількість ітерацій
            log: true,           // запис прогресу до console.log()
            logPeriod: 1,        // кількість ітерацій між протоколюванням
            learningRate: 0.3    // швидкість навчання
    }
);


let wstream = fs.createWriteStream('./mnistTrain.json');
wstream.write(JSON.stringify(net.toJSON(),null,2));
wstream.end();

console.log('MNIST dataset with Brain.js train done.')
```

Рисунок 13 – Код створення та навчання мережі

7. Зберігти код у файлі *train.js* та запустити процес навчання мережі, виконавши команду *node train.js*. Процес тренування відображується у консолі, з виведенням даних по кожній ітерації:

```
 node train.js
iterations: 0 training error: 0.025337777382314196
iterations: 1 training error: 0.01207515331077988
iterations: 2 training error: 0.008433572200507203
iterations: 3 training error: 0.006410056326066398
iterations: 4 training error: 0.0051079992630424014
iterations: 5 training error: 0.0041597024617616 4
iterations: 6 training error: 0.003478800872405162
iterations: 7 training error: 0.00298168402945562
iterations: 8 training error: 0.002552175854978601
iterations: 9 training error: 0.002236822502079419
iterations: 10 training error: 0.001990939762013806
iterations: 11 training error: 0.0017821399990446609
iterations: 12 training error: 0.0016474855866386179
iterations: 13 training error: 0.0015339805417561652
iterations: 14 training error: 0.001408732455024839
iterations: 15 training error: 0.0013109737246929578
iterations: 16 training error: 0.0012486558317570116
iterations: 17 training error: 0.001195887150211833
iterations: 18 training error: 0.0011499372572143505
iterations: 19 training error: 0.0010910918269841346
iterations: 20 training error: 0.0010478372414065545
iterations: 21 training error: 0.0009915064635373173
MNIST dataset with Brain.js train done.
```

Рисунок 14 – Процес тренування мережі

8. Після тренування мережі, необхідно перевірити її роботу. Для наочності, створіть HTML-сторінку з використанням бібліотек *sketchpad.js* та *build.js* (дані бібліотеки являються додатком до роботи). Повний лістинг розмітки сторінки зі стилями CSS наведений у Додатку А.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Тестування мережі</title>
    <!-- script src="./node_modules/mnist/dist/mnist.js"></script-->
    <script src=".sketchpad.js?v1.1"></script>
    <script src=".build.js"></script>
    <script>
        console.log(window["nn"]);
    </script>
    <style>...</style>
</head>
<body>
    <h1>Розпізнавання рукописних цифр з набору даних MNIST з Brain.js. Приклад нейронної мережі.</h1>
    <p>
    Намалюйте цифру у вікні нижче та натисніть кнопку "розпізнати".
    </p>
    <br/>
    <div id="container">
        <canvas id="sketchpad" width="280" height="280">Sorry, your browser is not supported.</canvas>
        <canvas id="thumbnail" width="28" height="28">Sorry, your browser is not supported.</canvas>
        <div id="result"></div>
        <div style="clear: right"></div>
    </div>
    <button type="button" id="sketchClearButton">Очистити</button>
    <button type="button" id="sketchRecogniseButton">Розпізнати</button>

    <br/>

</body>
</html>
```

Рисунок 15 – Розмітка HTML сторінки для перевірки роботи мережі

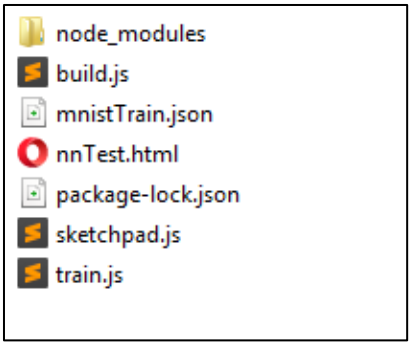


Рисунок 16 – Вміст каталогу в кінці проведеної роботи

9. Після того, як користувач намалює цифру, необхідно натиснути кнопку «Розпізнати», малюнок буди конвертований у розмір *28*28px*, а потім у побітову послідовність, після чого потрапить на вхід мережі. Остаточний результат також відобразиться у вікні браузеру. На рисунках 2.8 та 2.9 наведені приклади роботи з мережею у вікні браузеру.

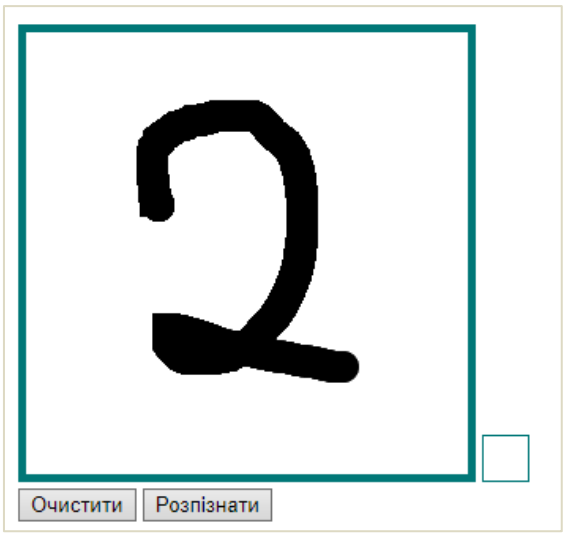


Рисунок 17 – Рукописна цифра до обробки

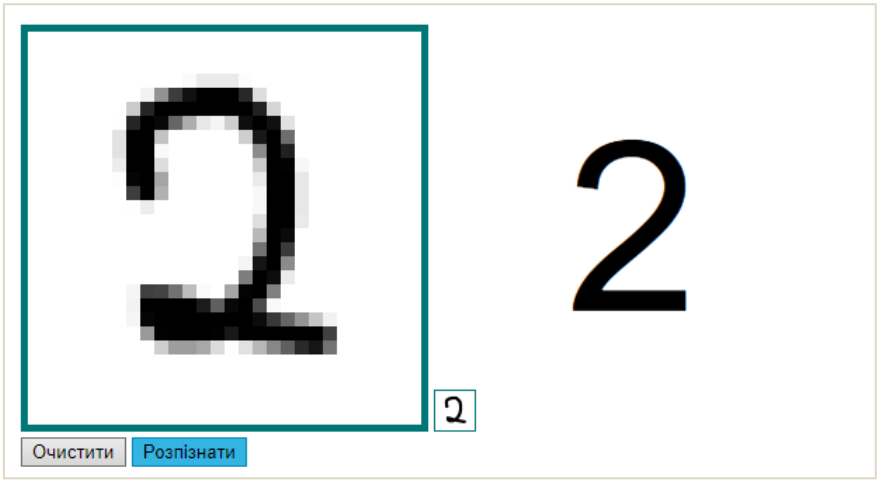


Рисунок 18 – Рукописна цифра після обробки

Мережа повинна розпізнавати цифри досить вдало, все залежить від ітерацій навчання та його часу, для наведених вище в коді реалізації даних, процес навчання тривав приблизно 40 хвилин, для 21 ітерації. Звісно, можна навчати мережу значно довше, що добре відобразиться на кількості помилок, звівши їх до нуля.

10. Після виконання роботи оформити звіт про виконання розрахунково-графічної роботи та здати його викладачу разом з файлами програми в друкованому та електронному вигляді.

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

Базова

1. Negoita M. Computational Intelligence Engineering of Hybrid Systems [Text] / Negoita M., Neagu D., Palade V. – Springer Verlag, 2005. – 213 p.
2. Piszcz A. Engineering Issues for an Adaptive Defense Network. / Piszcz A., Orlans N., Eyler–Walker Z., Moore D. – MITRE Technical Report. 2001. – 70 p.
3. Unsupervised adaptive filtering. V. 1, 2. Edited by S. Haykin. – New York: John Willey & Sons, Inc, 2000. – 1402 p.
4. Бессмертный И.А. Искусственный интеллект / И.А. Бессмертный. – СПб: СПбГУ ИТМО, 2010. – 132 с.
5. Васильев В.Г. Сети Петри, параллельные алгоритмы и модели мультипроцессорных систем [Текст] / В.В. Васильев, В.В. Кузьмук. – Киев: Наук. думка, 1990. – 212 с.
6. Кузьменко Б.В., Чайковська О.А. Системи штучного інтелекту: Навч.посібник.-К.:Альтерпрес, 2006.-140 с.
7. Луценко Е.В. Интеллектуальные информационные системы / Е.В. Луценко. – Краснодар: КубГАУ. – 2006. − 615 с.
8. Месюра В.І., Ваховська Л.М. Основи проектування систем штучного інтелекту. – Вінниця: ВДТУ, 2000.
9. Роберт Каллан. Основные концепции нейронных сетей.: Пер. с англ. – М.: Вильямс, 2001. – 287 с.
10. Спірін О.М. Початки штучного інтелекту: навч. посіб. для студ. фіз.-мат. спец-тей вищих пед. навч. закл-ів / О.М. Спірін. – Житомир:Вид-во ЖДУ, 2004. – 172 с.
11. Стюарт Рассел. Искусственный интеллект: современный подход. 2-е изд. Пер с англ./ С.Рассел, П.Норвиг. – К.: Вильямс, 2006. – 1408 с.

Допоміжна

1. Боровиков В. Statistica. Искусство анализа данных на комьютере / В. Боровиков. – 2003. – 688 с.

3. Глибовець М. М. Штучний інтелект : підручник для студ. вищих навч. закладів / М. М. Глибовець, О.В. Олецький. – К. : КМ Академія, 2002. – 369 с.

4. Достоверный и правдоподобный вывод в интеллектуальных системах / В. Н. Вагин, Е. Ю. Головина, А. А. Загорянский. – М. : Физматлит, 2004. – 704 с.

5. Дюк В. А. Data Mining - обнаружение знаний в базах данных / В. А. Дюк. – СПб. : изд-во «БСК», 2003. – 240 с.

6. Іванченко Г. Ф. Системи штучного інтелекту : навч. посібник / Г. Ф. Іванченко. – К., 2011. – 382 с.

7. Круглов В. В. Искусственные нейронные сети. Теория и практика / В. В. Круглов, В. В. Борисов. – М., 2002. – 382 с.

8. Кузьменко Б. В. Системи штучного інтелекту : Навч.посібник / Б. В. Кузьменко, О. А. Чайковська. – К. :Альтерпрес, 2006. – 140 с.

10. Методы и модели анализа данных: OLAP и Data Mining / А. А Барсегян., М. С. Куприянов, В. В. Степаненко, И. И. Холод. – СПб. : БХВ- Петербург, 2004.

ІНФОРМАЦІЙНІ РЕСУРСИ

1. Штучна нейронна мережа: http://wikipedia.org/wiki/Штучна_нейрона_мережа
2. Система контролю версій Git: https://git-scm.com
3. NodeJS – JavaScript оточення для розробки за межами браузеру: https://nodejs.org/uk/
4. Модуль MNIST digits (для ознайомлення): https://github.com/cazala/mnist
5. brain.js – нейронна мережа на JS: https://github.com/brainjs/ brain.js
6. Текстові редактори: Sublime Text: https://www.sublimetext.com/3, Brackets: http://brackets.io

ДОДАТОК А. ТИТУЛЬНА СТОРІНКА

Міністерство освіти і науки України
Чернігівський національний технологічний університет
Кафедра кібербезпеки та математичного моделювання

Розрахунково-графічна робота

з дисципліни „Системи штучного інтелекту”

варіант №___

виконав(ла)
студент(ка) групи _____
_____
(прізвище, ім’я, по-батькові)
перевірив
_____
оцінка _____ балів

Підпис викладача _____

Чернігів 201_

ДОДАТОК Б. ЛІСТИНГ РОЗМІТКИ СТОРІНКИ ДЛЯ ПЕРЕВІРКИ РОБОТИ НЕЙРОННОЇ МЕРЕЖІ

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Тестування мережі</title>
    <script src="./sketchpad.js?v1.1"></script>
    <script src="./build.js"></script>
    <script>console.log(window["nn"]);</script>
    <style>
        body {
            margin: 10px;
            padding: 10px;
            font-family:Arial;
        }
        #container {
            position:relative;
            width: 500px;
        }
        #sketchpad {
            border: 5px solid #077;
        }
        #thumbnail {
            border: 1px solid #077;
        }
        #result {
            font-size: 128pt; /* Размер шрифта в пунктах */
            float: right;
            width: 120px;
            height: 280px;
            border: 5px solid #fff;
            text-align: center;
            line-height: 280px;
        }</style>
</head>
<body>
    <h1>Розпізнавання рукописних цифр <br> з набору даних MNIST з Brain.js.</h1>
    <p>
    Намалюйте цифру у вікні нижче та натисніть кнопку "розпізнати".
    </p>
    <br/>
    <div id="container">
        <canvas id="sketchpad" width="280" height="280">Sorry, your browser is
not supported.</canvas>
        <canvas id="thumbnail" width="28" height="28">Sorry, your browser is not
supported.</canvas>
        <div id="result"></div>
        <div style="clear: right"></div>
    </div>
    <button type="button" id="sketchClearButton">Очистити</button>
    <button type="button" id="sketchRecogniseButton">Розпізнати</button><br/>
</body>
</html>
```