МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни

«Основи теорії інтелектуальних систем»

для студентів спеціальності 123 Комп'ютерна інженерія
за освітньою програмою «Комп'ютерні системи та мережі»
денної форми навчання

2024

Методичні вказівки до лабораторних робіт з дисципліни «Основи теорії інтелектуальних систем» для студентів спеціальності 123 Комп'ютерна інженерія за освітньою програмою «Комп'ютерні системи та мережі» денної форми навчання / Укл.: М.Ю. Тягунова – Запоріжжя: НУ «Запорізька політехніка», 2024. – 42 с.

Укладач: М.Ю. Тягунова, к.т.н., доцент

Рецензент: Г.Г. Киричек, к.т.н., доцент

Відповідальний
за випуск: М.Ю. Тягунова, к.т.н., доцент

Затверджено:
На засіданні кафедри
«Комп'ютерні системи та мережі»
Протокол №6
від 01 лютого 2024 р.

Рекомендовано до видання
НМК факультету КНТ
Протокол № 6
від 06 лютого 2024 р.

ЗМІСТ

1 Лабораторна робота №1. Нейронні мережі у середовищі MatLab® 5
1.1 Теоретичні відомості 5
1.2 Хід роботи 7
1.3 Завдання до виконання 20
1.4 Зміст звіту 21
1.5 Контрольні питання 21
2 Лабораторная робота №2. Розпізнавання образів 22
2.1 Теоретичні відомості 22
2.2 Задача на розпізнавання літер 23
2.3 Завдання до виконання 26
2.4 Зміст звіту 26
2.5 Контрольні питання 26
3 Лабораторна робота №3. Розпізнавання обличчя 27
3.1 Теоретичні відомості 27
3.2 Налаштування програмного середовища 28
3.3 Розпізнавання обличчя з відеопотоку 30
3.4 Завдання до виконання 32
3.5 Зміст звіту 32
3.6 Контрольні запитання 33
4 Лабораторна робота №4. Застосування генетичних алгоритмів 34
4.1 Теоретичні відомості 34
4.2 Задача для мінімізації функції 36

4.3 Задача для максимізації функції 39
4.4 Завдання до виконання 41
4.5 Зміст звіту 41
4.6 Контрольні запитання 41
Література 42

1 ЛАБОРАТОРНА РОБОТА №1
НЕЙРОННІ МЕРЕЖІ У СЕРЕДОВИЩІ MATLAB®

Мета роботи: набути навички роботи у програмному середовищі MATLAB при роботі з нейронними мережами.

1.1 Теоретичні відомості

Нейронні мережі широко використовуються для вирішення різноманітних завдань. Основними серед них є завдання класифікації, кластеризації, прогнозування, управління, апроксимації, оптимізації та управління. Середовище MATLAB® надає можливість моделювати роботу нейронних мереж різних типів, використовуючи при цьому зручний інтерфейс.

При використанні нейронів з певними функціями активації (передавальними характеристиками), нейронні мережі є універсальним апроксиматором. Останнє означає, що в заданому діапазоні зміни вхідних змінних нейронні мережі можуть з заданою точністю відтворювати (моделювати) довільну безперервну функцію цих змінних.

Для того, щоб працювати з MATLAB®, необхідно зареєструватися на офіційному сайті розробника цього програмного забезпечення за посиланням https://www.mathworks.com/. Після цього необхідно або завантажити це програмне середовище на свій комп'ютер, або працювати в онлайн версії цього продукту.

Особливі моменти при встановленні програмного забезпечення MATLAB®2022 на своєму комп'ютері зображено на рис. 1.1.

У цій лабораторній роботі буде розглянуто нейронні мережі прямого поширення, тобто без зворотних зв'язків, особливістю яких є їх стійкість.

Після того, як структуру нейронної мережі обрано, задано параметри, які будуть на вході та виході мережі, визначено відсоток даних, які призначаються для навчання, контролю та тестування нейронної мережі, необхідно провести навчання нейронної мережі.

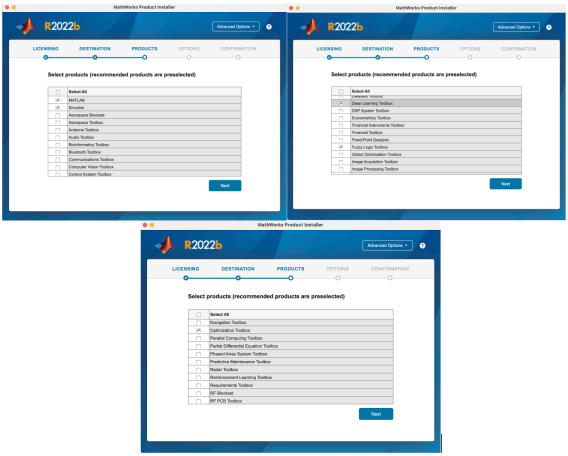

Рисунок 1.1 – Особливості при встановленні MATLAB®2022

Інтерфейс MATLAB® для вирішення завдань нейронними мережами можна запустити декількома способами:

1 спосіб: ввівши команду nnstart у командному вікні, як показано на рис. 1.2;

2 спосіб: перейшовши на вкладку APPS та обравши відповідний тип мережі у MACHINE LEARNING AND DEEP LEARNING, як показано на рис. 1.3.

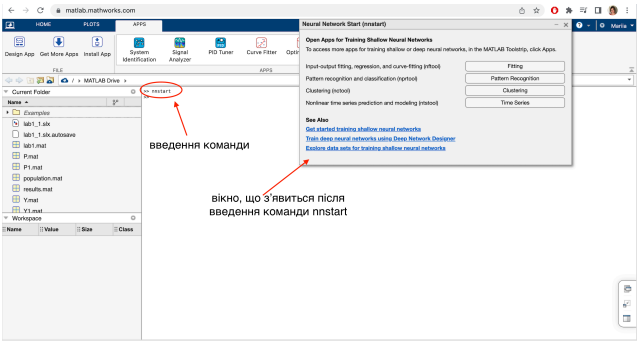

Рисунок 1.2 – Введення команди nnstart

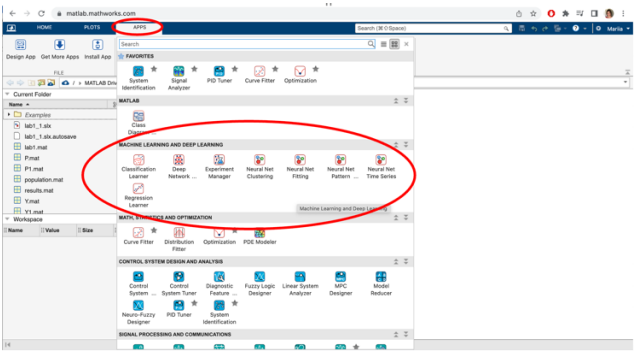

Рисунок 1.3 – Вибір у вкладці APPS

Додаток Neural Net Fitting дозволяє створювати, візуалізувати та навчати двошарову мережу прямого зв'язку для вирішення проблем апроксимаії даних.

Використовуючи цей підрозділ програми, є можливість:

- імпортувати дані з файлу, робочої області MATLAB® або використати один із наведених прикладів наборів даних;
- розділити дані на набори для навчання, перевірки та тестування.
- визначити та навчити нейронну мережу;
- оцінити продуктивність мережі за допомогою середньоквадратичної помилки та регресійного аналізу;
- проналізувати результати за допомогою графіків візуалізації, таких як регресія або гістограма помилок;
- створити функції, придатні для розгортання за допомогою інструментів MATLAB Compiler™ і MATLAB Coder™, і експортування в Simulink® для використання з Simulink Coder.

1.2Хід роботи

У цій лабораторній роботі змоделюємо нейронну мережу, що дозволить визначати кількість населення України в залежності від введеного року.

Необхідні дані для опрацювання візьмемо з таблиці 1.1, що представлена нижче.

Таблиця 1.1 – Населення України у різні роки

Рік	Кількість населення	Рік	Кількість населення	Рік	Кількість населення	Рік	Кількість населення
1950	36588000	1987	51260900	2000	49115000	2010	45782600
1951	37223000	1989	51452000	2001	48663600	2011	45598200
1959	41869000	1992	51708200	2002	48240902	2012	45453300
1960	42468600	1993	51870400	2003	47823100	2013	45372700
1970	47118200	1994	51715400	2004	47442100	2014	45245900
1973	48274400	1995	51300400	2005	47100500	2015	42759661
1976	49151000	1996	50874100	2006	46749200	2016	42590879
1979	49752200	1997	50400000	2007	46465700	2017	42414900
1980	49952500	1998	49973500	2008	46192300	2020	41732779
1984	50678600	1999	49544800	2009	45963300	2022	40997699

Після цього ці дані необхідно ввести у середовище MATLAB®. Для цього необхідно створити масив даних років «years» та масив даних кількості населення «population», як показано на рис. 1.4 ліст. 1.1.

Лістинг 1.1 – Початкові дані

```
years = [1950 1951 1959 1960 1970 1973 1976 1979 1980 1984 1987 1989
1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005
2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2020
2022]

population = [36588000 37223000 41869000 42468600 47118200 48274400
49151000 49752200 49952500 50678600 51260900 51452000 51708200
51870400 51715400 51300400 50874100 50400000 49973500 49544800
49115000 48663600 48240902 47823100 47442100 47100500 46749200
46465700 46192300 45963300 45782600 45598200 45453300 45372700
45245900 42759661 42590879 42414900 41732779 40997699]
```

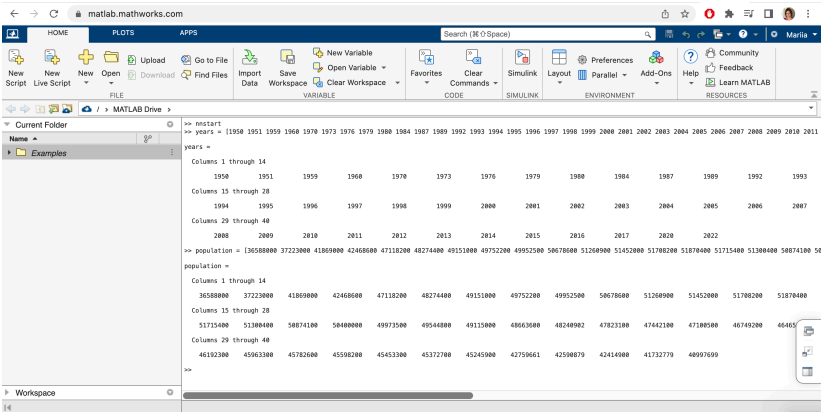

Рисунок 1.4 – Створення масивів даних у робочому просторі MATLAB®

Далі одним із зазначених способів (рис. 1.2 та рис.1.3) переходимо до вікна створення нейронної мережі. Та обираємо Fitting або Neuron Net Fitting в залежності від обраного способу створення нейронної мережі (рис. 1.5 – 1.6).

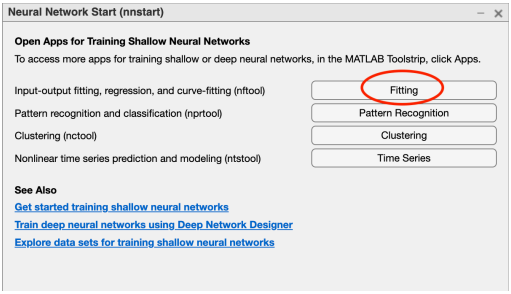

Рисунок 1.5 – Вибір типу нейронної мережі

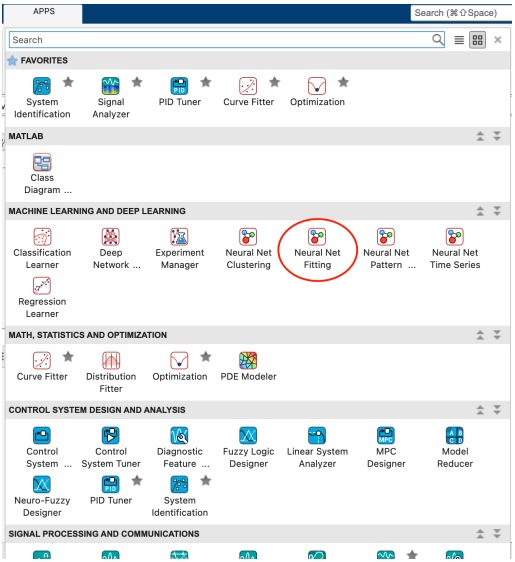

Рисунок 1.6 – Вибір типу нейронної мережі у вкладці APPS

Після вибору відповідної нейронної мережі відкриється вікно з її структурою та налаштуваннями. Обираємо дані для завантаження, як показано на рис. 1.7.

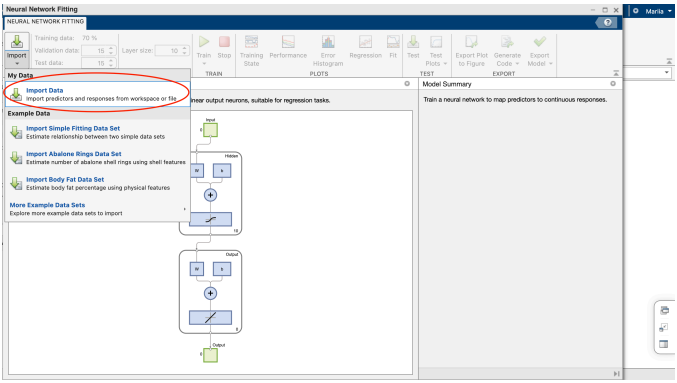

Рисунок 1.7 – Вибір даних для імпортування

Після натиснення на відповідне поле Import Data обираємо необхідні дані років та кількості населення, які ввели до середовища раніше, як показано на рис. 1.8, та натискаємо OK.

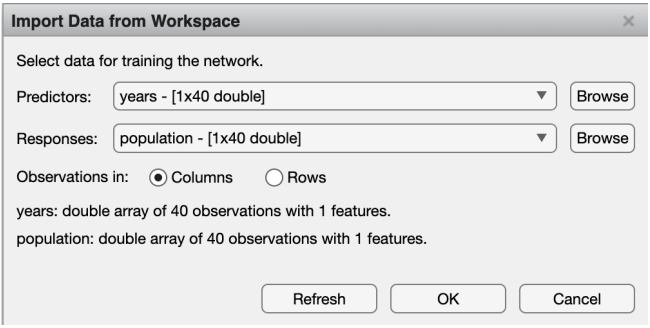

Рисунок 1.8 – Імпортування даних

У вікні, що відкриється можна змінювати відсоток даних для валідації та тестування, а також кількість шарів нейронної мережі Змінювані параметри обведені червоним на рис. 1.9. Для нашого прикладу немає потреби змінювати ці значення, тому залишаємо їх за замовчанням.

Далі необхідно навчити створену нейронну мережу та проаналізувати отримані результати. Для початку процесу навчання натискаємо кнопку Train, як показано на рис. 1.9.

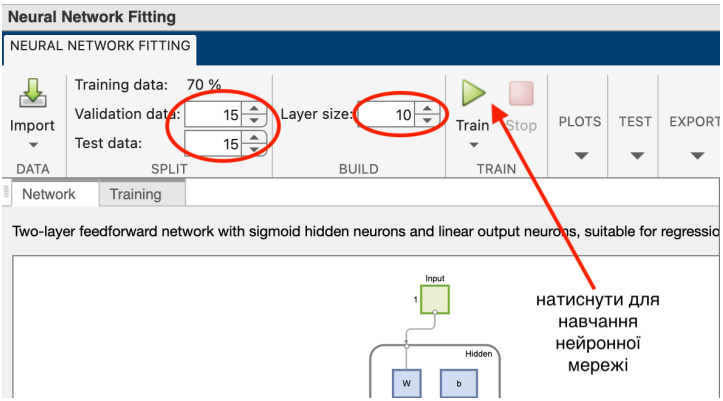

Рисунок 1.9 – Налаштування параметрів нейронної мережі

Після завершення процесу навчання відкриється вікно з отриманими результатами, схоже на рис. 1.10. Проте значення параметрів, які будуть отримані у кожного студента можуть відрізнятися від зображених на скріншоті.

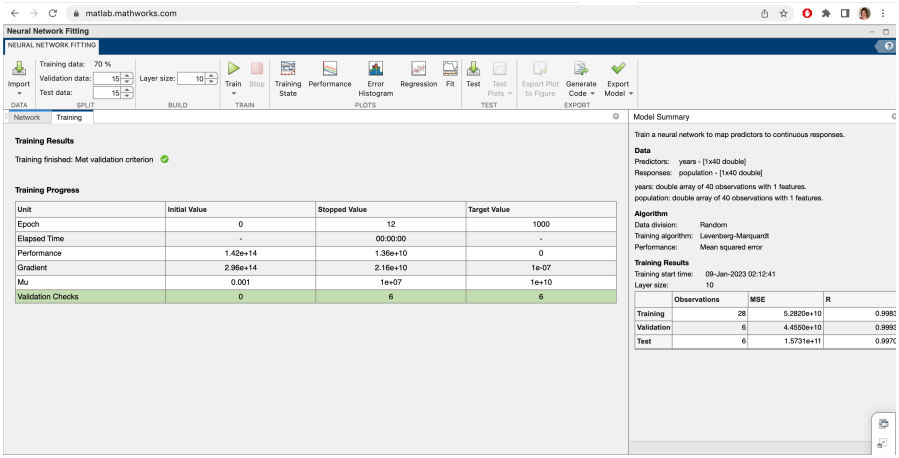

Рисунок 1.10 – Результати навчання нейронної мережі

За замовчанням обрано метод навчання Левенберга-Марквардта. За бажанням можна обирати інші методи для навчання, натиснувши на чорний маленький трикутник під кнопкою Train, як показано на рис. 1.11.

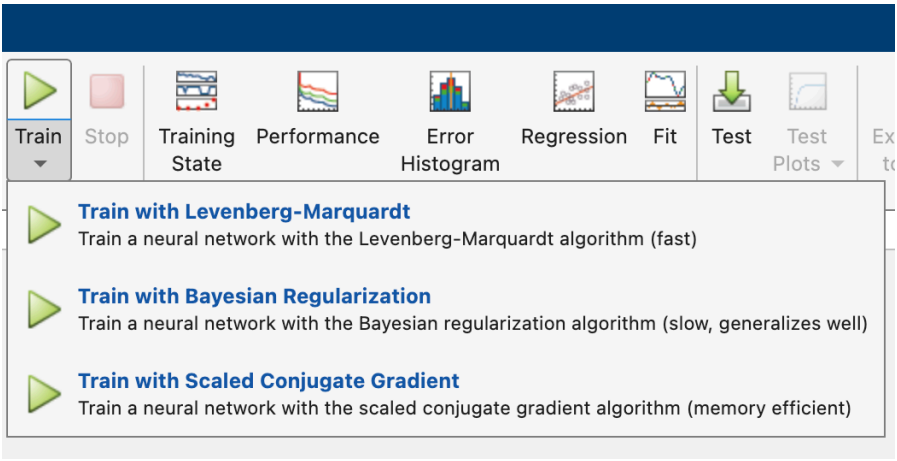

Рисунок 1.11 – Вибір методу навчання нейронної мережі

Далі за кожною із кнопок на панелі розділу PLOTS (обведені червоним на рис. 1.12) можна передивитися та проаналізувати отримані результати навчання нейронної мережі.

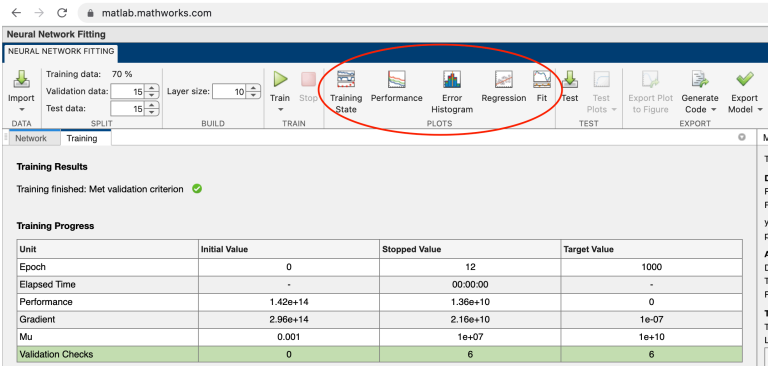

Рисунок 1.12 – Панель наочного аналізу отриманих результатів

Особливо наочним є відображення реального (запланованого згідно табл.1.1) та отриманого нейронною мережею результату за кнопкою Fit (рис. 1.13).

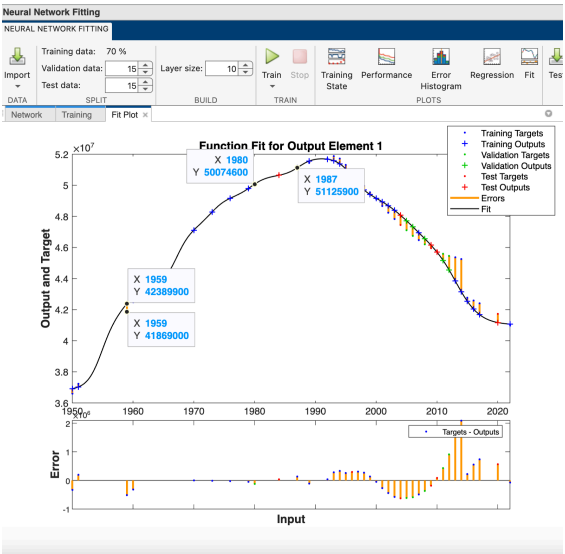

Рисунок 1.13 – Аналіз отриманих результатів за кнопкою Fit

Для того, щоб передивитися, на яке значення на вході системи, яка буде реакція навченої нейронної мережі, необхідно експортувати її до середовища Simulink. Для цього необхідно натиснути на трикутник знизу надпису «Export Model» та обрати із випадаючого меню «Export to Simulink», як це показано на рис. 1.14.

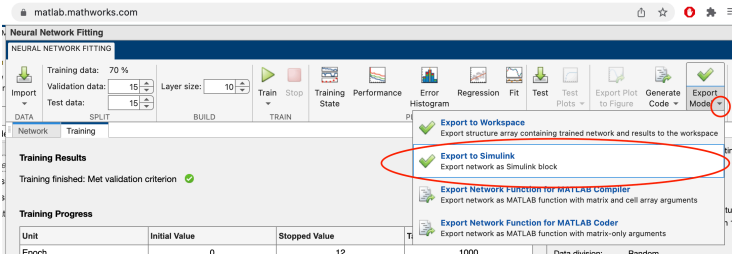

Рисунок 1.14 – Експорт нейронної мережі до середовища Simulink

У результаті буде згенеровано модель у Simulink, як це показано на рис. 1.15. Перемикати необхідні вікна для роботи можна за допомогою навігаційних кнопок у нижньому правому куті (рис.1.16).

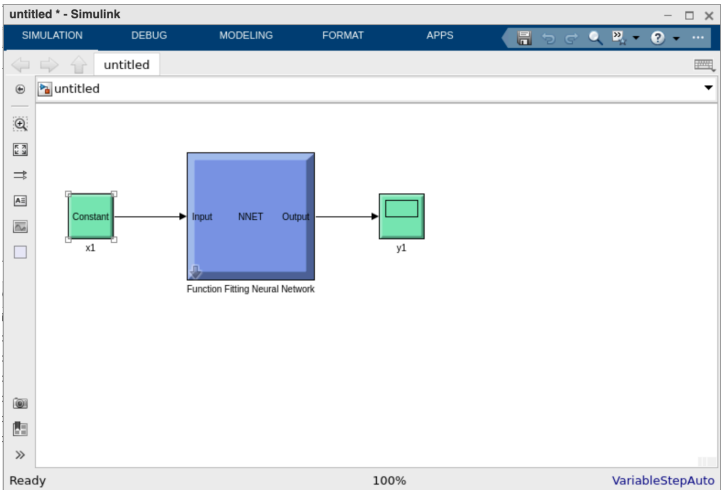

Рисунок 1.15 – Модель нейронної мережі у середовищі Simulink

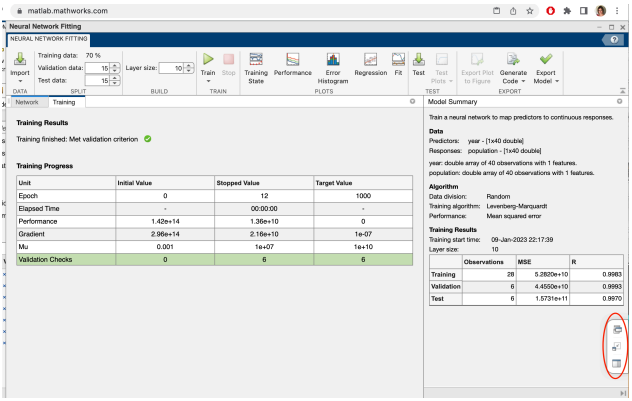

Рисунок 1.16 – Панель навігації

Для зручності відображення результатів у вікні Simulink необхідно змінити існуючий блок виведення «y1» на блок «display». Для цього необхідно обрати у вкладці SIMULATION підрозділ LIBRARY та у ньому обрати Library Browse, як показано на рис. 1.17.

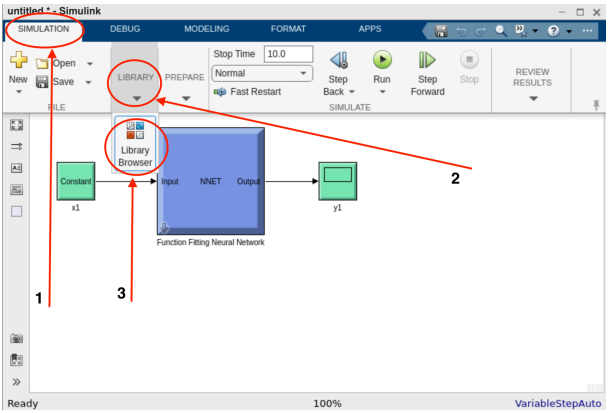

Рисунок 1.17 – Вибір бібліотеки елементної бази

Після цього необхідно у пошуковому полі ввести «display» та натиснути кнопку пошуку. У наданому переліку обрати елемент «Display», як показано на рис. 1.18.

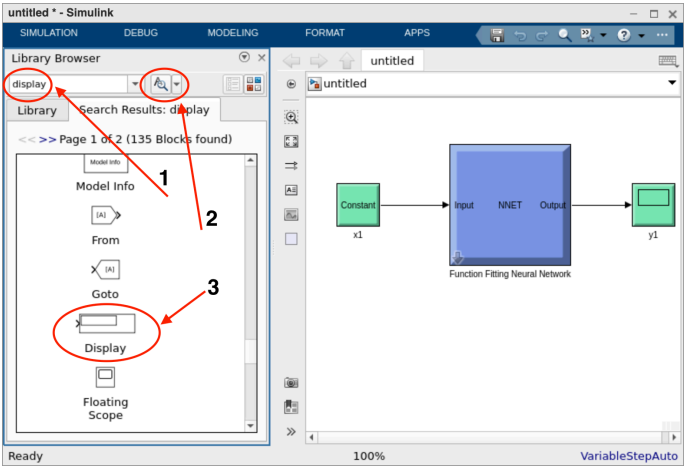

Рисунок 1.18 – Пошук елемента відображення результатів

Далі переносимо знайдений блок «Display» у вікно моделювання та замінюємо ним блок «y1», як показано на рис. 1.19.

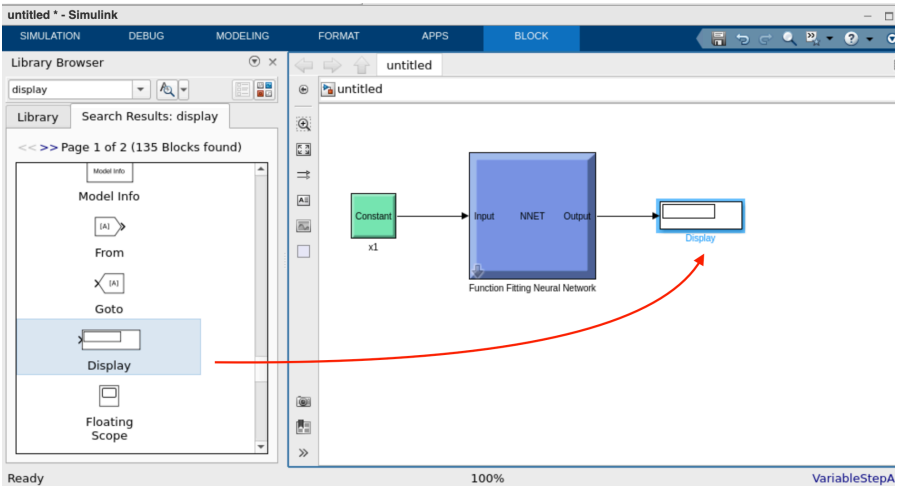

Рисунок 1.19 – Додавання блоку «Display»

Після цього правою клавішею миші натискаємо на блоці «x1», обираємо «Block Parameters» та змінюємо значення у полі «Constant value» на будь-який рік із таблиці 1.1 (рис. 1.20). Натискаємо OK.

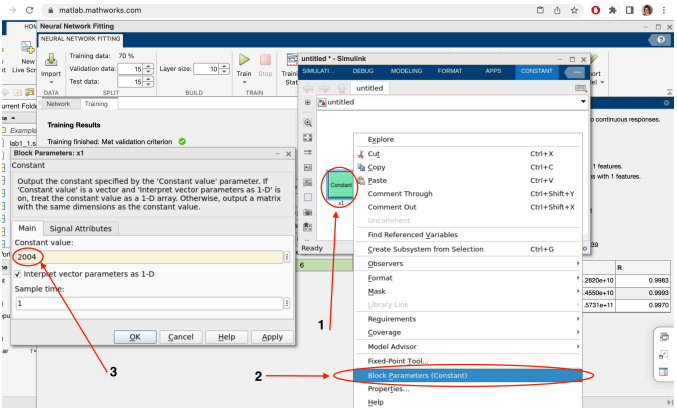

Рисунок 1.20 – Зміна параметрів блоку «x1»

Після цього запускаємо процес моделювання. Для цього у вкладці SIMULATION натискаємо кнопку Run, або клавішу F5. Результати моделювання відображаються у блоці Display (рис. 1.21).

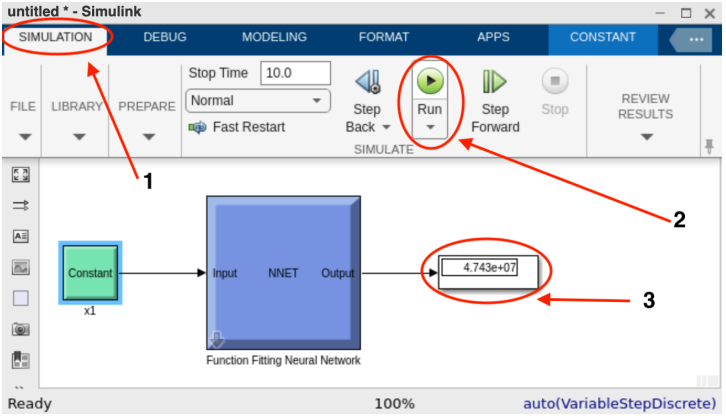

Рисунок 1.21 – Моделювання у Simulink

Для зміни формату відображення вихідних даних можна натиснути правою клавішею миші на блоці Display обираємо «Block Parameters» та обираємо у випадаючому списку формат «bank». Далі натискаємо OK і дані у блоці Display змінять формат відображення на більш звичний. Послідовно все це відображено на рис. 1.22.

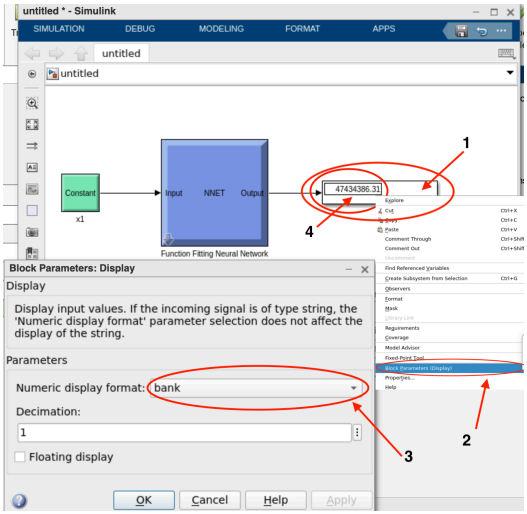

Рисунок 1.22 – Зміна формату відображення результатів

Для того, щоб передивитися деталі змодельованої мережі, натисніть на стрілочку основного блоку моделі (рис.1.23), тоді отримаєте результат, як на рис. 1.24. Для більш детального перегляду окремих блоків двічі натисніть на певному блоці. Результат представлено на рис. 1.25.

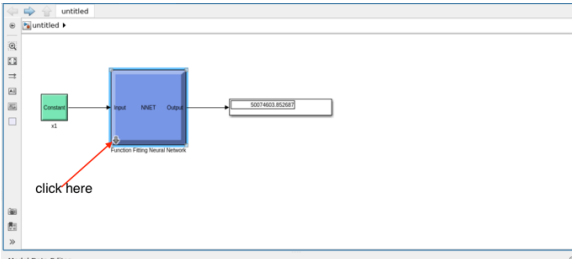

Рисунок 1.23 – Вибір детального перегляду елемента

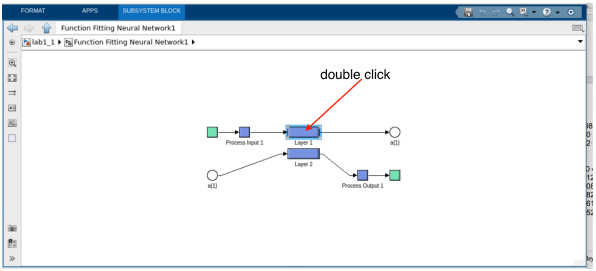

Рисунок 1.24 – Перегляд функції апроксимації

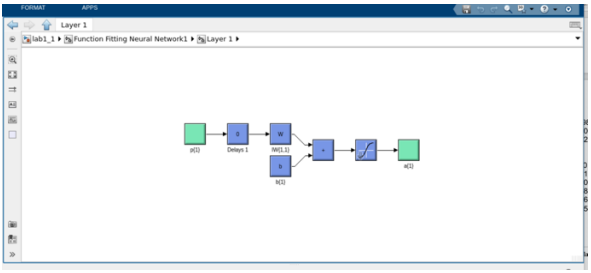

Рисунок 1.25 – Перегляд функції апроксимації певного шару

Далі збережімо отримані результати. Для цього у вкладці SIMULATION натискаємо Save, обираємо місце для зберігання та назву файла (рис. 1.26).

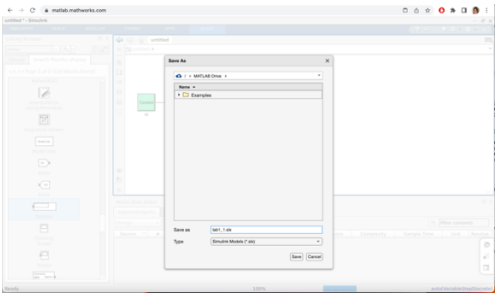

Рисунок 1.26 – Збереження результатів у Simulink

Для того, щоб зберегти введені у робочій простір дані, необхідно їх зберегти в окремий файл з розширенням .mat. Для цього треба натиснути Save Workspace, обрати місце зберігання та задати ім'я, наприклад, lab1_1.mat (рис.1.27).

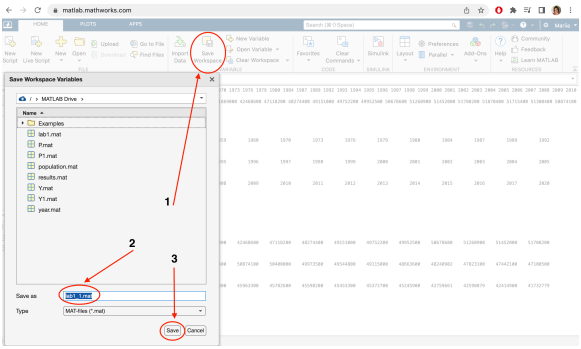

Рисунок 1.27 – Збереження даних з Workspace

Аналізуючи отримані помилки при навчанні та результати у Simulink, робимо висновок, що помилка достатньо велика. Одним із варіантів зменшення помилки може стати збільшення нашого набору початкових даних. Для збільшення даних введемо наступні операції до командного вікна MATLAB, як показано нижче:

```
Y = [years years]
P  = [population population]
Y1 = [Y Y Y]
P1 = [P P P]
```

1.3Завдання до виконання

1. Виконати усі дії описані у пункті 1.2;
2. Навчити нейронну мережу на збільшеному наборі початкових даних Y1 та P1 та проаналізувати отримані результати;
3. Перевірити роботу нейронної мережі, навченої на збільшеному наборі даних, у середовищі Simulink;
4. Розробити нейронну мережу, яка визначатиме кількість населення для певної країни за узгодженням з викладачем згідно таблиці нижче:

Таблиця 1.2 – Індивідуальні завдання

№ студента за списком	**Країна**
1, 9, 18	Великобританія
2, 10, 19	Польща
3, 11, 20	Бразилія
4, 12, 21	США
5, 13, 22	Туреччина
6, 14, 23	Ізраїль
7, 15, 24	Німеччина
8, 16, 25	Франція
усі інші	Швеція

5. Результати показати викладачу та пояснити проведений аналіз;
6. Необхідно виконати апроксимацію функції наступного вигляду:

$$cos\left(\frac{5\pi x}{N} + sin\left(\frac{12\pi x}{N}\right)\right)$$

Для цього введіть у командному рядку цільову функцію target = cos (5 * pi * [1:100] / 100 + sin (12 * pi * [1:100] / 100)) та набір навчальних даних data = [1:100].

7. Проаналізуйте отримані результати.
8. Виконайте апроксимацію функції за варіантом з табл. 1.3 та проаналызуйте отриманы результати.

Таблиця 1.3 – Індивідуальні завдання для апроксимації

№ студента за списком	**Функція**
1, 3, 9, 18	$tg\left(\frac{\pi x}{N} + sin\left(\frac{\pi x}{N}\right)\right)$
2, 10, 19	$cos\left(\frac{3\pi x}{N}\right) + sin\left(\frac{2\pi x}{N}\right)$
4, 12, 21	$cos\left(1 - \frac{5\pi x}{N}\right)$
5, 13, 22	$sin\left(1 - \frac{12\pi x}{N}\right)$
6, 11, 14, 23	$cos\left(\frac{\pi x}{2N}\right) - tg\left(\frac{12\pi x}{N}\right)$
7, 15, 20, 24	$sin\left(1 - \frac{2\pi x}{N}\right) - tg\left(\frac{2\pi x}{N}\right)$
усі інші	$cos\left(\frac{\pi x}{N} - \left(cos\frac{\pi x}{2N}\right)\right)$

1.4Зміст звіту

Звіт повинен мати такі складові:
- титульний лист;
- мета роботи;
- результат виконання завдань;
- відповіді на контрольні питання.

1.5Контрольні питання

1. Які задачі вирішуються нейронними мережами, приклади застосування нейронних мереж.
2. Загальний принцип та призначення навчання нейронних мереж.
3. Особливості використання середовища *MATLAB®* для моделювання нейронних мереж.

2 ЛАБОРАТОРНАЯ РОБОТА №2
РОЗПІЗНАВАННЯ ОБРАЗІВ

Мета роботи: навчитися використовувати апарат нейронних мереж для розпізнавання літер та цифр.

2.1 Теоретичні відомості

Розпізнавання образів - це процес ідентифікації або класифікації об'єктів або паттернів на основі їх властивостей чи характеристик, знятих з вхідних даних, таких як зображення, звуки, текст або інші види інформації. Ця область штучного інтелекту та обробки сигналів знаходить застосування в багатьох сферах, включаючи комп'ютерне зорове сприйняття, розпізнавання мови, біометрію, медицину, автоматизацію виробництва та інші галузі.

Мережа Хопфілда (Hopfield network) - це тип нейронної мережі, яка використовується для розпізнавання шаблонів (паттернів) за асоціаціями. Вона може використовуватися для розпізнавання літер або інших зразків, що представляються у вигляді бінарних матриць.

Основний принцип роботи мережі Хопфілда полягає в тому, що вона зберігає паттерни в зв'язках між нейронами та використовує ітеративний процес для відновлення асоційованого паттерна на вхідних даних.

Для розпізнавання літер за допомогою мережі Хопфілда спочатку мережа повинна бути навчена на паттернах (наприклад, літерах). Кожен паттерн представляється у вигляді бінарної матриці, де кожен піксель може бути 0 або 1.

Коли мережа навчена, вона може бути використана для розпізнавання. Це робиться шляхом подачі вхідних даних (зашумленого чи пошкодженого паттерна) на вхід мережі. Далі мережа починає ітеративно оновлювати стани своїх нейронів на основі правил асоціації, які вона вивчила під час навчання. Це включає в себе оновлення станів нейронів на основі зв'язків між ними.

Процес ітерацій завершується, коли мережа досягає стабільного стану (зближення до асоційованого паттерна). Результатом розпізнавання є паттерн, який мережа "відновила" на основі вхідних даних.

Мережа Хопфілда є досить простою та ефективною для розпізнавання та асоціації зразків, але вона має обмеження щодо кількості та складності зразків, які може зберігати та розпізнавати. У випадках більш складних завдань розпізнавання, можуть бути використані більш потужні нейронні мережі, такі як згорткові нейронні мережі (CNN) або рекурентні нейронні мережі (RNN).

2.2 Задача на розпізнавання літер

Необхідно написати власну програму розпізнавання літер від A до D у середовищі *MATLAB*.

Для цього необхідно у вебверсії *MATLAB* ввести у командному вікні зазначений програмний код у лістингу 2.1.

Лістинг 2.1 – Створення та відображення бази даних літер

```
A1=[1 -1 -1 -1 1;-1 1 1 1 -1;-1 -1 -1 -1 -1;-1 1 1 1 -1;-1 1 1 1 -1];
B1=[-1 -1 -1 -1 1;-1 1 1 1 -1;-1 -1 -1 -1 1;-1 1 1 1 -1;-1 -1 -1 -1 1];
C1=[-1 -1 -1 -1 -1;-1 1 1 1 1;-1 1 1 1 1;-1 1 1 1 1;-1 -1 -1 -1 -1];
D1=[-1 -1 -1 1 1;-1 1 1 -1 1;-1 1 1 -1 1;-1 1 1 -1 1;-1 -1 -1 1 1];
% Display images of stored patterns
hold on;
figure(1);
subplot(221);
imagesc(A1);
title('Stored A');
subplot(222);
imagesc(B1);
title('Stored B');
subplot(223);
imagesc(C1);
title('Stored C');
subplot(224);
imagesc(D1);
title('Stored D');
```

Натискаємо «Enter» та отримуємо результат відображення введених чотирьох літер (рис. 2.1).

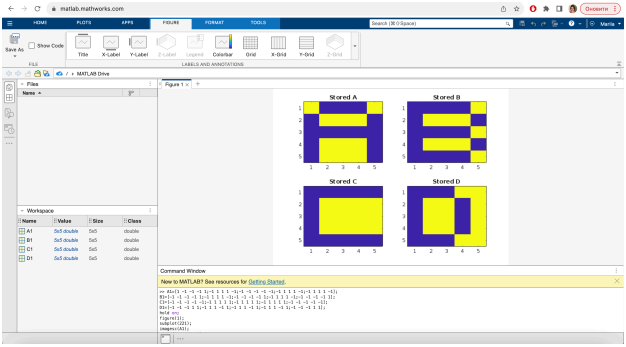

Рисунок 2.1 – Відображення літер

Далі необхідно доповнити скрипт для розпізнавання образу, ввівши код, наведений у лістингу 2.2

Лістинг 2.2 – Розпізнавання образу літери

```
clc
clear

% Ініціалізуємо символи
A1=[1 -1 -1 -1 1;-1 1 1 1 -1;-1 -1 -1 -1 -1;-1 1 1 1 -1;-1 1 1 1 -1];
B1=[-1 -1 -1 -1 1;-1 1 1 1 -1;-1 -1 -1 -1 1;-1 1 1 1 -1;-1 -1 -1 -1 1];
C1=[-1 -1 -1 -1 -1;-1 1 1 1 1;-1 1 1 1 1;-1 1 1 1 1;-1 -1 -1 -1 -1];
D1=[-1 -1 -1 1 1;-1 1 1 -1 1;-1 1 1 -1 1;-1 1 1 -1 1;-1 -1 -1 1 1];

% Виводимо символи
hold on;
figure(1);
subplot(221);
imagesc(A1);
title('Stored A');
subplot(222);
imagesc(B1);
title('Stored B');
subplot(223);
imagesc(C1);
title('Stored C');
subplot(224);
imagesc(D1);
title('Stored D');

% Ініціалізуємо матриці для використання в обчисленнях
```

```
V = [A1(:) B1(:) C1(:) D1(:)];
alpha = 0.7;
iter = 15;

% Розраховуємо вагову матрицю
W = V*V';
W = W - diag(diag(W));

% Обираємо цілочисельну початкову матрицю та змінюємо її на
вектор-стовпець
x = sign(2*rand(5,5)-1);
x = reshape(x,25,1);

% Повторюємо обчислення для оновлення та відображення
матриці
for k = 1:iter
    figure(2)
    x = reshape(x,5,5);
    subplot(3,5,k);
    imagesc(x);
    x = reshape(x,25,1);
    xtemp = alpha*W*x;
    for j = 1:25
        if xtemp(j)>0
            x(j) = 1;
        else
            x(j)=-1;
        end
    end
end
```

Результатом запуску коду буде формування випадкового набору символів розмірністю 5 на 5 елементів. Нейронна мережа, на основі отриманої бази, намагатиметься розпізнати, до якої з літер найбільш схожий сформований набір клітинок (рис. 2.3).

Як видно, уже на другій ітерації образу, нейронна мережа розпізнала в ній літеру С. Ви можете проекспериментувати з новим набором значень, які будуть випадково генеруватись під час кожного запуску.

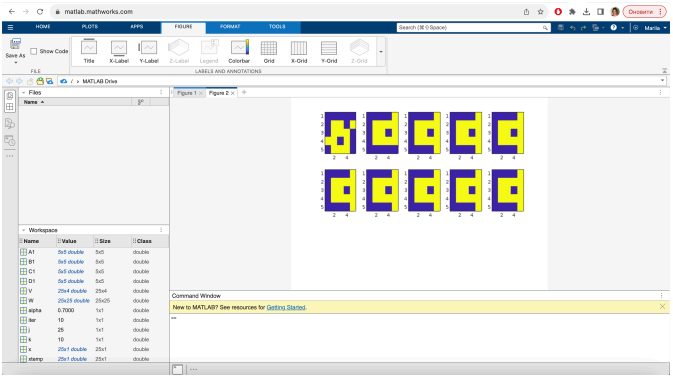

Рисунок 2.3 – Результат розпізнавання образу літери

2.3Завдання до виконання

Виконати приклади 2.2. Додатковим завданням є створення програми для розпізнавання образів цифр від 0 до 9. Продемонструвати результати виконання викладачу.

За результатами виконання завдань слід зробити висновки та навести їх у звіті.

2.4Зміст звіту

Звіт повинен мати такі складові:

– титульний лист;

– мета роботи;

– результат виконання завдань;

– відповіді на контрольні питання.

2.5Контрольні питання

1. Опишіть методи розпізнавання образів.
2. Які галузі використання кожного з методів розпізнавання образів?
3. Які основні напрями в розпізнаванні образів?
4. Які конкретні задачі успішно вирішуються в галузі розпізнавання образів?

3 ЛАБОРАТОРНА РОБОТА №3 РОЗПІЗНАВАННЯ ОБЛИЧЧЯ

Мета роботи: навчитися створювати програми для розпізнавання обличчя мовою Python за допомогою бібліотеки OpenCV.

3.1 Теоретичні відомості

Автоматична ідентифікація особистості людини по його зображенню на фотографії або в відеопотоці має широке комерційне та наукове застосування. Дана тематика з'явилася на початку 80-х років, однак, її бурхливий розвиток почалося в 90-х, після створення нових технологій в сфері обробки зображень та обчислювальних машин. Дана технологія має особливий інтерес у зв'язку з тим, що може відбуватися безконтактно.

В даний час зріс інтерес до різних методів ідентифікації осіб. Для вирішення цього завдання необхідно виконати два етапи: виявити особа на фотографії, виділити і розпізнати його.

При всьому різноманітті різних алгоритмів і методів розпізнавання зображень, типовий метод розпізнавання складається з трьох компонентів:

– перетворення вихідного зображення в початкове уявлення (може включати в себе як попередню обробку, так і математичні перетворення, наприклад обчислення головних компонент);

– виділення ключових характеристик (наприклад береться перші n головних компонент або коефіцієнтів дискретного косинусного перетворення);

– механізм класифікації (моделювання): кластерна модель, метрика, нейронна мережа тощо.

Крім цього, побудова методу розпізнавання спирається на апріорну інформацію про предметну область (в даному випадку - характеристики особи людини), і коригується експериментальної інформацією, що з'являється по ходу розробки методу.

3.2 Налаштування програмного середовища

Для початку необхідно завантажити та інсталювати IDE PyChar. Після чого необхідно створити новий проект. Далі необхідно інсталювати бібліотеку opencv. Для цього, у терміналі програми вводимо команду:

```
pip install opencv-python
```

Далі у вікні програми імпортуємо цю бібліотеку командою:

```
import cv2
```

Враховуючи те, що opencv – безкоштовна бібліотека з відкритим кодом, то багато класифікаторів можна знайти на github. Переходимо за посиланням https://github.com/opencv/opencv та відкриваємо папку data, там же обираємо вже існуючі класифікатори.

Для нашого завдання оберіть з папки haarcascade класифікатор визначення обличчя:

```
haarcascade_frontalface_default.xml
```

Далі необхідно завантажити цей файл та перенести його до поточного проекту за шляхом:

```
Lib\site-packages\cv2\data
```

Після чого, необхідно створити об’єкт у Python (у дужках прописано шлях до обраного файлу):

```
face_cascade_db = cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_frontalface_default.xml")
```

Далі необхідно завантажити до проекту будь-яке фото з обличчям у поточну директорію проекту.

Після чого, необхідно створити об’єкт для цього зображення:

```
img = cv2.imread("img.jpg")
```

Для того, щоб наше зображення правильно розпізнало існуючим класифікатором, необхідно його перетворити у відтінки сірого:

```
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Виведемо це зображення на екран. Для цього введемо команду imshow (назва вікна, назва зображення) та затримку `cv2.waitKey()`:

```
    cv2.imshow('rez',                                    img_gray)
cv2.waitKey()
```

Та запускаємо проект на виконання (правою кнопкою миші - Run). У результаті ми побачимо зображення у сірих відтінках (рис. 3.1).

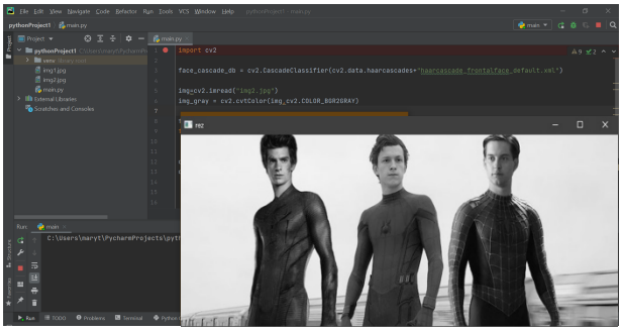

Рисунок 3.1 – Виведення зображення у сірих відтінках

Далі створимо об’єкт faces, у який і запишемо розпізнані обличчя за допомогою класифікатору:

```
faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19)
```

За допомогою об’єкту face ми отримужмо перелік знайдених координат на цьому зображенні.

Далі виведемо квадрат на знайдених обличчях та не чорно-біле, а кольорове, зображення. Повний код програми наведено у лістингу 3.1

Лістинг 3.1 – Код програми для розпізнавання обличь на фотографії

```
import cv2

face_cascade_db =
cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_fronta
lface_default.xml")

img = cv2.imread("family.jpg")
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19)
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow('rez', img)
cv2.waitKey()
```

Після чого запустимо проект та у результаті отримаємо програму, яка розпізнає обличчя на фотографії (рис. 2.2).

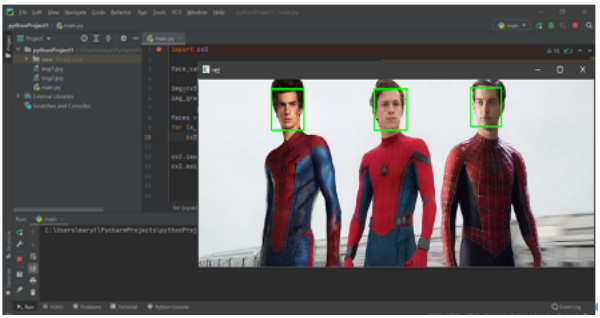

Рисунок 3.2 – Результат розпізнавання обличчя на фотографії

3.3 Розпізнавання обличчя з відеопотоку

Необхідно написати програму, яка здатна розпізнавати обличчя з потоку веб-камери.

Для цього створимо новий файл у проекті. Як джерело інформації введемо поток з веб-камери:

```
cap = cv2.VideoCapture(0)
```

Далі у циклі отримаємо зображення з веб-камери:

```
    while True:
    success, img = cap.read()
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19)
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

Функція для виходу з циклу по натисканню кнопки «q»:

```
    cv2.imshow('rez', img)
if cv2.waitKey() & 0xff == ord('q'):
    break
```

Повний код програми наведено у лістингу 3.2.

Лістинг3.2 – Код програми для розпізнавання обличь з потоку веб-камери

```
    import cv2

    face_cascade_db =
    cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_fronta
    lface_default.xml")

    cap = cv2.VideoCapture(0)
    while True:
        success, img = cap.read()
        img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19)
        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

        cv2.imshow('rez', img)
        if cv2.waitKey() & 0xff == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```

У результаті, почергово будуть відображатись кадри з веб-камери з обведеним обличчям на ньому у разі успіху (рис. 3.3).

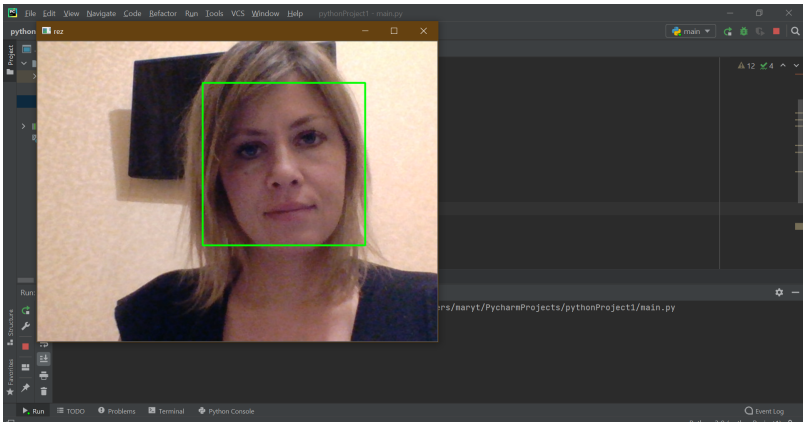

Рисунок 3.3 – Результат розпізнавання обличчя з потоку веб-камери

Як бачимо з результатів, програмою зчитано інформацію з потоку веб-камери, розпізнано на ній обличчя та обведено його зеленою рамкою.

3.4 Завдання до виконання

Виконати приклади 3.2, 3.3. Продемонструвати результати виконання викладачу.

Додатковим (необов'язковим) завданням є :

- створення програми для розпізнавання обличчя із записаного відео.

Продемонструвати результати виконання викладачу.

За результатами виконання завдань слід зробити висновки та навести їх у звіті.

3.5 Зміст звіту

Звіт повинен мати такі складові:

– титульний лист;

– мета роботи;

– результат виконання завдань;

– відповіді на контрольні питання.

3.6 Контрольні запитання

1. Дуже стисло опишіть програмні продукти, які існують для розпізнавання обличь.
2. Стисло опишіть бібліотеки Python, які призначені для розпізнавання.
3. Опишіть бібліотеки Python для роботи з нейронними мережами.
4. Особливості згорткових нейронних мереж.

4 ЛАБОРАТОРНА РОБОТА №4
ЗАСТОСУВАННЯ ГЕНЕТИЧНИХ АЛГОРИТМІВ

Мета роботи: навчитися застосовувати генетичні алгоритми на практиці.

4.1 Теоретичні відомості

Основи теорії генетичних алгоритмів сформульовані Холландом в його основоположній роботі і в подальшому були розвинені рядом дослідників. Найбільш відомою і цитованої в даний час є монографія Д. Голдберга, де систематично викладено основні результати та області практичного застосування генетичних алгоритмів.

Генетичні алгоритми використовують принципи і термінологію, запозичені у біологічної науки – генетики. У генетичних алгоритмах кожна особина представляє потенційне рішення деякої проблеми. У класичному генетичному алгоритмі особина кодується рядком двійкових символів – хромосомою, кожен біт якої називається геном. Множина особин – потенційних рішень становить популяцію. Пошук (суб) оптимального вирішення проблеми виконується в процесі еволюції популяції – послідовного перетворення однієї кінцевої множини рішень у іншу за допомогою генетичних операторів репродукції, кросинговеру та мутації.

Еволюційні обчислення використовують наступні три механізми природної еволюції:

– перший принцип заснований на концепції виживання найсильніших і природного відбору за Дарвіном який був сформульований їм у 1859 році в книзі «Походження видів шляхом природного відбору». Згідно Дарвіну особини, які краще здатні вирішувати завдання в своєму середовищі, виживають і більше розмножуються (репродуцюють). У генетичних алгоритмах кожна особина представляє собою рішення деякої проблеми аналогією з цим принципом особини з кращими значеннями цільової (фітнес) функції мають великі шанси вижити і репродуцювати. Формалізація цього принципу дає оператор репродукції;

– другий принцип обумовлений тим фактом, що хромосома нащадка складається з частин, отриманих з хромосом батьків. Цей принцип був відкритий в 1865 році Менделем. Його формалізація дає основу для оператора схрещування (кросинговеру);

– третій принцип заснований на концепції мутації, відкритої в 1900 році де Вре. Спочатку цей термін використовувався для опису істотних (різких) змін властивостей нащадків і набуття ними властивостей, відсутніх у батьків. За аналогією з цим принципом генетичні алгоритми використовують подібний механізм для різкої зміни властивостей нащадків і тим самим, підвищують різноманіття особин в популяції (множині рішень).

Ці три принципи складають ядро еволюційних обчислень. Використовуючи їх, популяція (множина рішень даної проблеми) еволюціонує від покоління до покоління.

Генетичний алгоритм бере множину параметрів оптимізаційної проблеми та кодує їх послідовностями кінцевої довжини в деякому кінцевому алфавіті (в найпростішому випадку двійковий алфавіт «0» і «1»).

Простий генетичний алгоритм випадковим чином генерує початкову популяцію стрінгів (хромосом). Потім алгоритм генерує наступне покоління (популяцію), за допомогою трьох основних генетичних операторів:

– оператор репродукції (ОР);

– оператор схрещування (кросинговеру, ОК);

– оператор мутації (ОМ).

Генетичні оператори є математичної формалізацією наведених вище трьох основоположних принципів Дарвіна, Менделя і де Вре природної еволюції.

Генетичний алгоритм працює до тих пір, поки не буде виконано задану кількість поколінь (ітерацій) процесу еволюції або на деякій генерації буде отримано задану якість або внаслідок передчасної збіжності при попаданні в деякий локальний оптимум.

У кожному поколінні множина штучних особин створюється з використанням старих і додаванням нових з хорошими властивостями. Генетичні алгоритми – не просто випадковий пошук, вони ефективно використовують інформацію, накопичену в процесі еволюції. У процесі пошуку рішення необхідно дотримувати баланс між «експлуатацією»

отриманих на поточний момент кращих рішень і розширенням простору пошуку. Різні методи пошуку вирішують цю проблему по-різному.

4.2 Задача для мінімізації функції

Необхідно мінімізувати функцію однієї змінної:

$$f(x) = 8x - 16 - 12\sqrt[3]{(x+4)^2}$$

Для цього, у середовищі *MATLAB®* необхідно створити два скрипти `ex2.m` та `ex2_func.m`. У першому міститься описана вище функція (лістинг 4.1), другий містить алгоритм для підрахунку фітнес-функції, пошуку найкращої особини та середньої відстані між особинами (лістинг 4.2).

Для створення скрипта натискаємо у вкладці HOME кнопку New Script. У вікно, що відкриється вводимо необхідний скрипт (ліст. 4.1). Для створення ще одного скрипту достатньо натиснути «+» справа від назви скрипта, як це показано на рис. 4.1.

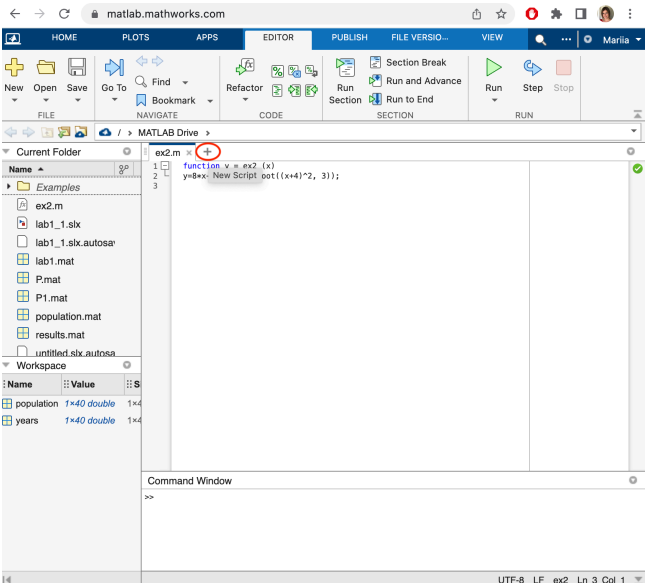

Рисунок 4.1 – Створення нового скрипта

Лістинг 4.1 – Функція мінімізації

```
function y = ex2 (x)
y=8*x-16-12*(nthroot((x+4)^2, 3));
end
```

Лістинг 4.2 – Функція генетичного алгоритму

```
function [X,FVAL,REASON,OUTPUT,POPULATION,SCORES] =
ex2_func
fitnessFunction = @ex2;
nvars = 1;
options = optimoptions(@ga,'PopInitRange',[-4 ; 1]);
options = optimoptions(options,'PopulationSize',10);
options =
optimoptions(options,'MutationFcn',{@mutationgaussian 1
1});
options = optimoptions(options,'Display','off');
options =
optimoptions(options,'PlotFcns',{@gaplotbestf,
@gaplotbestindiv, @gaplotdistance});
[X,FVAL,REASON,OUTPUT,POPULATION,SCORES] =
ga(fitnessFunction,nvars,options);
end
```

Для збереження створених файлів необхідно натиснути Save – Save as…, як показано на рис.4.2, та зберегти скрипти з відповідним ім’ям.

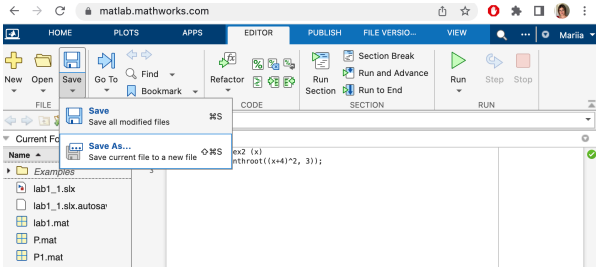

Рисунок 4.2 – Збереження скрипта

Після цього необхідно запустити створені скрипти. Для цього у командному рядку необхідно ввести команду `optimtool('ga')` – рис.4.3.

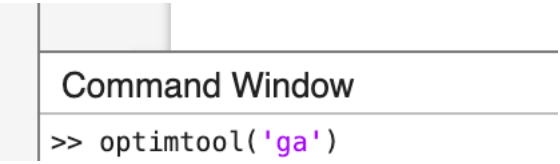

Рисунок 4.3 – Робота у командному вікні

Далі, якщо з’явиться вікно, як на рис. 4.4, обрати «Solver-based».

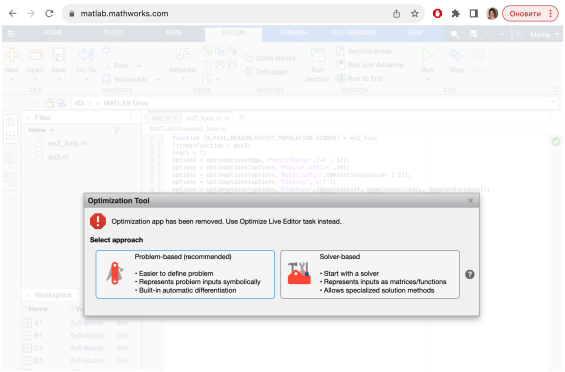

Рисунок 4.4 – Вибір інструмента оптимізації

Далі необхідно запустити створені скрипти, натиснувши Run у вкладці EDITOR (рис.4.5 – обведено червоним), у результаті чого отримаємо результат підрахунку фітнес-функції, пошуку найкращої особини та середньої відстані між особинами (рис. 4.5). Обов’язково необхідно впевнитися, що запускається потрібний проєкт. Для цього або перевірити безпосередньо за кнопкою Run, або атиснути Run Section, а потім – Run.

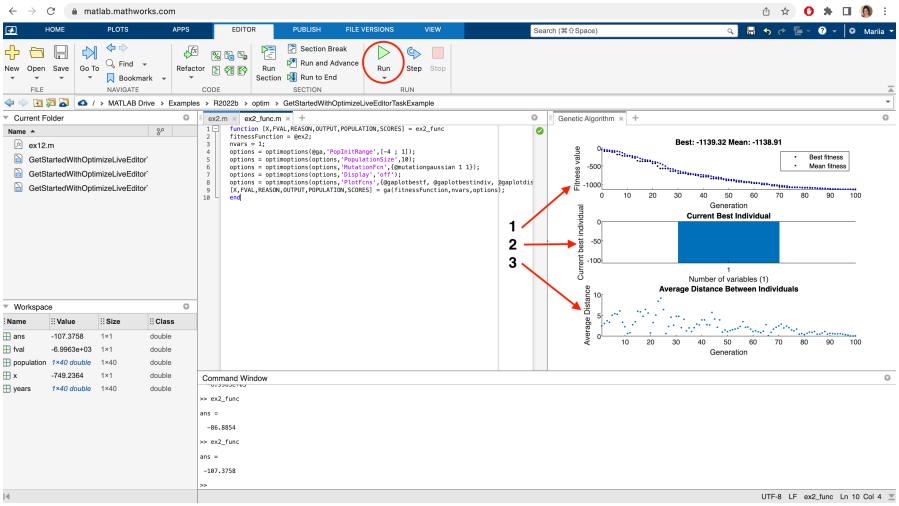

Рисунок 4.5 – Результат роботи генетичного алгоритму

Перший рисунок відображає зміну значення цільової функції. На другому рисунку зображена найкраща особина. Третій рисунок відповідає зсуванням відстані між особинами в поколіннях. Особини стають однакові (хеммінгова відстань = 0) в останніх 20 поколіннях.

Генетичний алгоритм потрібно запустити кілька разів, а потім вибрати оптимальне рішення. Це пов'язано з тим, що початкова популяція формується з використанням генератора випадкових чисел.

4.3 Задача для максимізації функції

Необхідно максимізувати функцію двох змінних:

$$z(x, y) = \exp(-x^2 - y^2) + \sin(x + y)$$

Додаток з генетичних алгоритмів вирішує тільки задачі мінімізації, тому для знаходження максимуму функції f(x) слід мінімізувати функцію -f (x). Це пояснюється тим, що точка мінімуму -f(x) є деякою точкою f(x), в якій досягається максимум.

Для цього, у середовищі *MATLAB*® необхідно знову створити два скрипти. У першому міститься описана вище функція (лістинг 4.3),

другий містить алгоритм для пошуку фітнес функції, найкращої особини та середньої відстані між особинами (лістинг 4.4).

Лістинг 4.3 – Функція максимізації

```
function z = ex13 (x)
z=-(exp(-x(1)^2-x(2)^2)+sin(x(1)+x(2)));
end
```

Лістинг 4.4 – Функція генетичного алгоритму

```
function [X,FVAL,REASON,OUTPUT,POPULATION,SCORES] = ex13_func
fitnessFunction = @ex13;
nvars = 2;
options = optimoptions(@ga,'PopInitRange',[-1 ; 3]);
options = optimoptions(options,'PopulationSize',10);
options =
optimoptions(options,'MutationFcn',{@mutationgaussian 1 1});
options = optimoptions(options,'Display','off');
options = optimoptions(options,'PlotFcns',{@gaplotbestf,
@gaplotbestindiv, @gaplotdistance});
[X,FVAL,REASON,OUTPUT,POPULATION,SCORES] =
ga(fitnessFunction,nvars,options);
end
```

Перевіряємо результат підрахунку фітнес-функції, пошуку двох найкращих особин та середньої відстані між особинами (рис. 4.5).

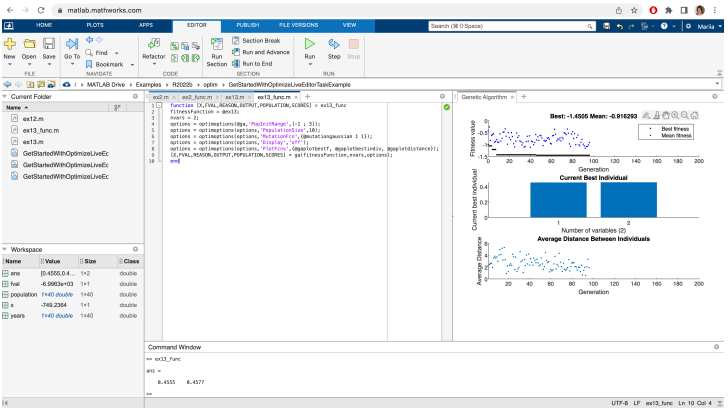

Рисунок 4.5 – Результат роботи генетичного алгоритму для функції з двома змінними

4.4 Завдання до виконання

1. Виконати приклади 4.2, 4.3. Продемонструвати результати виконання викладачу.
2. Додатковими завданнями є :

– знайти максимум функції:

$$y(x) = 3\sqrt[3]{(x+4)^2} - 2x - 8$$

– знайти мінімум функції:

$$z(x,y) = (y-3) * \exp(-x^2 - y^2)$$

Продемонструвати результати виконання викладачу.

За результатами виконання завдань слід зробити висновки та навести їх у звіті.

4.5 Зміст звіту

Звіт повинен мати такі складові:
– титульний лист;
– мета роботи;
– результат виконання завдань;
– відповіді на контрольні питання.

4.6 Контрольні запитання

1. Основі принципи еволюційних обчислень.
2. Що є генетичним алгоритмом?
3. Основні оператори генетичних алгоритмів.
4. Галузі використання генетичних алгоритмів.
5. Розкрийте поняття «паралельні генетичні алгоритми».

ЛІТЕРАТУРА

1. Шаховська Н.Б. Системи штучного інтелекту / Шаховська Н.Б., Р.М. Камінський, О.Б. Вовк - Львів: Львівська політехніка, 2018. - 392 с.

2. Офіційна сторінка підтримки користувачів MATLAB® [Електронний ресурс] / Режим доступу: https://www.mathworks.com/help/matlab/index.html

3. Wilson H. Artificial intelligence. / H Wilson. - Grey House Publishing. 2018.