**Міністерство освіти і науки України**

**Національний технічний університет України «КПІ» імені Ігоря Сікорського**

**Кафедра обчислювальної техніки ФІОТ**

**ЗВІТ**

**з лабораторної роботи №7**

**з навчальної дисципліни «Computer Vision»**

**Тема:**

**ДОСЛІДЖЕННЯ ТЕХНОЛОГІЙ ІДЕНТИФІКАЦІЇ ОБ’ЄКТІВ НА**

**ЦИФРОВИХ ЗОБРАЖЕННЯХ ДЛЯ ЗАДАЧ COMPUTER VISION**

**Виконав:**

Студент 3 курсу кафедри ІПІ ФІОТ,

Навчальної групи ІП-11

Трикош І. В.

**Перевірив:**

Професор кафедри ОТ ФІОТ

Писарчук О.О.

**Київ 2024**

## **І. Мета:**

Дослідити принципи та особливості підготовки даних, синтезу, навчання та застосування штучних нейронних мереж (Artificial Neural Networks) для практичних задач ідентифікації в технологіях Computer Vision.

## **ІІ. Завдання:**

Розробити програмний скрипт мовою Python що реалізує обчислювальний алгоритм ідентифікації об’єктів на цифрових зображеннях за технологіями штучних нейронних мереж (Artificial Neural Networks): підготовка даних; конструювання нейромережі; навчання штучної нейронної мережі; застосування нейромережі.

Розробити програмний скрипт, що забезпечує ідентифікацію бінарних зображень 7 літералів, заданих матрицею растра. Для ідентифікації синтезувати, навчити та застосувати штучну нейронну мережу в «сирому» вигляді реалізації матричних операцій. Обґрунтувати вибір архітектури та алгоритму навчання нейромережі. Довести працездатність та ефективність синтезованої нейронної мережі.

## **ІІІ. Результати виконання лабораторної роботи.**

* 1. **Синтезована математична модель**

Відповідно до умов задачі синтезовано математичну модель для ідентифікації зображень 7 літералів.

Нейрон – це базовий обчислювальний вузол, який приймає вхідні дані, обробляє їх та генерує вихідні дані.

Штучна нейронна мережа – це обчислювальна модель, що натхненна біологічними нейронними мережами, яка призначена для розв’язання задач машинного навчання і штучного інтелекту.

Основні компоненти:

1. Вхідний шар – приймає вхідні дані. Кількість нейронів у цьому шарі відповідає кількості ознак.
2. Приховані шари – можуть складатися з різної кількості нейронів, в залежності від задачі. Вони генерують внутрішнє представлення.
3. Вихідний шар – це шар, що генерує вихідні прогнози на основі оброблений у прихованому шарі даних.
4. Зв’язки або ваги – це з’єднання між нейронами, які впливають на вхідні дані перед їх обробкою нейроном. Вони можуть мати різні значення.
5. Функція активації – вона визначає вихід нейрона на основі зваженої суми вхідних даних.
6. Функція втрат – вона оцінює різницю між прогнозованими вихідними значеннями та фактичними значеннями. Мета ШНМ – мінімізувати цю функцію шляхом оптимізації ваг.
7. Оптимізатор – оновлює ваги ШМН.

Модель реалізує штучну нейронну мережу з трьома шарами:

* Вхідний – 30 нейронів, що відповідає кількості пікселів на бінарному зображенні літерала.
* Прихований – 9 нейронів.
* Вихідний – 7 нейронів, що відповідає кількості літералів, які треба розпізнати.

Початкові ваги генеруються випадковим чином. Як функцію активації обрано сигмоїдальну. Функція втрат – середнє квадратичне відхилення. Рух мережею поступовий – від вхідного до вихідного шару. Ваги оновлюються з допомогою зворотного поширення помилки: спочатку обчислюється вихід на кожному шарі мережі, далі обчислюються дельти для кожного шару, після цього обчислюються градієнти і наостанок оновлюються ваги з допомогою обчислених градієнтів.

Тренування мережі відбувається задану кількість ітерацій. На кожній виконується рух уперед, обчислення похибки та зворотне поширення помилки.

Передбачення мережі – результат руху від вхідного до вихідного шару.

* 1. **Результати архітектурного проектування та їх опис**

На рис. 1 зображена блок-схема до моделі до програмного скрипту:

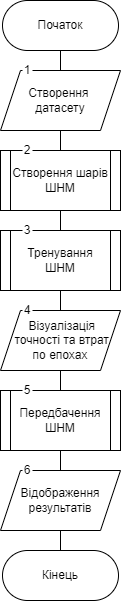


Рисунок 1 – Блок схема алгоритму

Робота алгоритму розпочинається із створення датасету. Сюди входить створення літералів як бінарних зображень та очікуваного результату по літералах. Також літерали відображаються у графічному вікні. Ці дії виконуються у блоці 1 блок-схеми алгоритму рис.1.

У блоці 2 виконується створення шарів нейронної мережі. Початкові ваги генеруються випадковим чином, а кількість нейронів по шарах – 30, 9 та 7.

У блоці 3 виконується тренування нейронної мережі на датасеті. Воно ітеративно 1000 разів.

У блоці 4 виконується візуалізація результатів навчання мережі, а саме таких параметрів, як точність та втрати по епохах.

У блоці 5 відбувається передбачення нейронної мережі. Тут датасет ще раз проходить вже по навченій мережі.

У блоці 6 виконується відображення результатів передбачення по літералах. На цьому виконання алгоритму завершено.

* 1. **Опис структури проекту програми**

Для реалізації розробленого алгоритму мовою програмування Python з використанням можливостей інтегрованого середовища PyCharm сформовано проєкт.

Проєкт базується на лінійній бізнес-логіці функціонального програмування та має таку структуру.

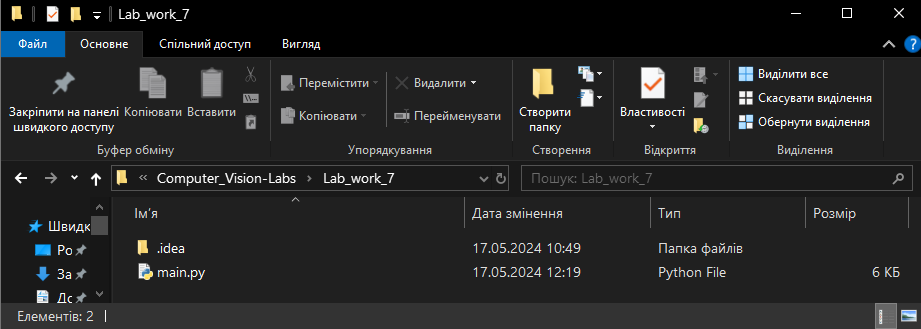


Рисунок 2 – Структура проєкту

Lab\_work\_7 – головний каталог проєкту

main.py – файл програмного коду

* 1. **Результати роботи програми відповідно до завдання (допускається у формі скриншотів)**

Результатом роботи програми є сукупність послідовності графічних вікон, що реалізують умови завдання лабораторної роботи.

1. Датасет:

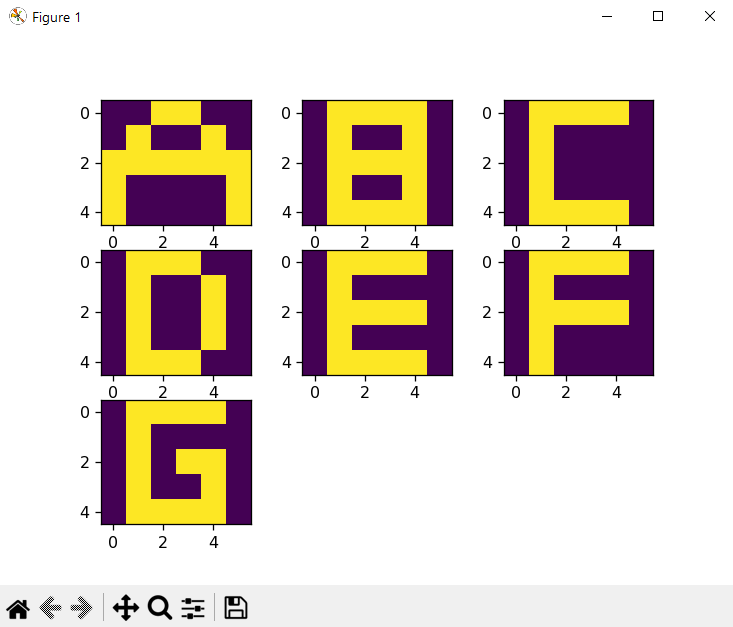


Рисунок 3 – Датасет (літерали)

1. Результати тренування мережі: точність по епохах:

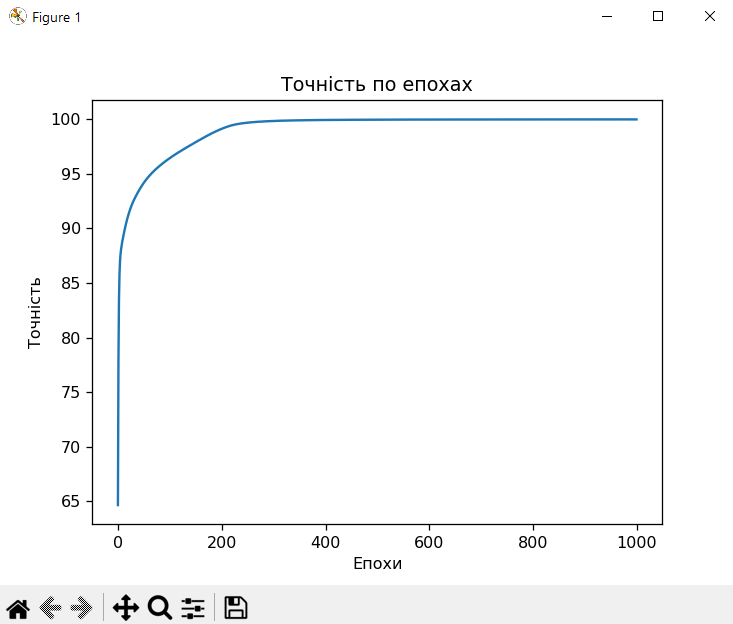


Рисунок 4 – Точність мережі по епохах

1. Результати тренування мережі: втрати по епохах:

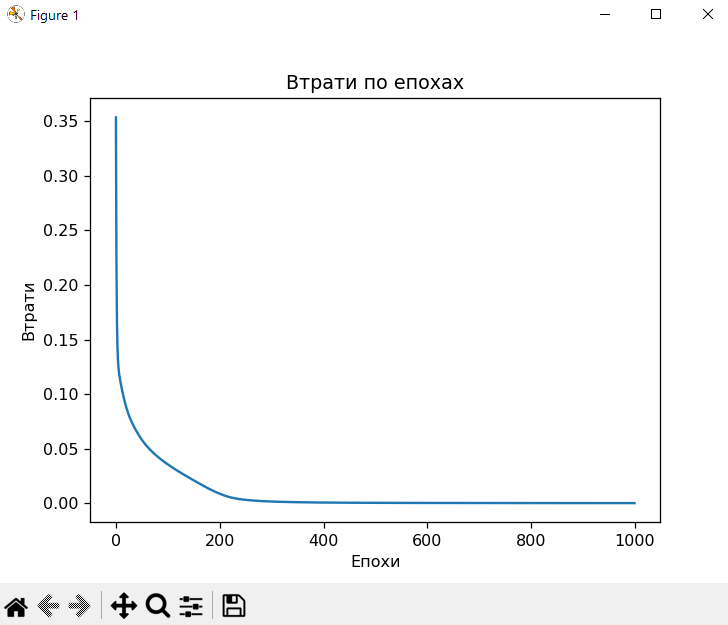


Рисунок 5 – Втрати мережі по епохах

1. Ідентифікація літерала A:

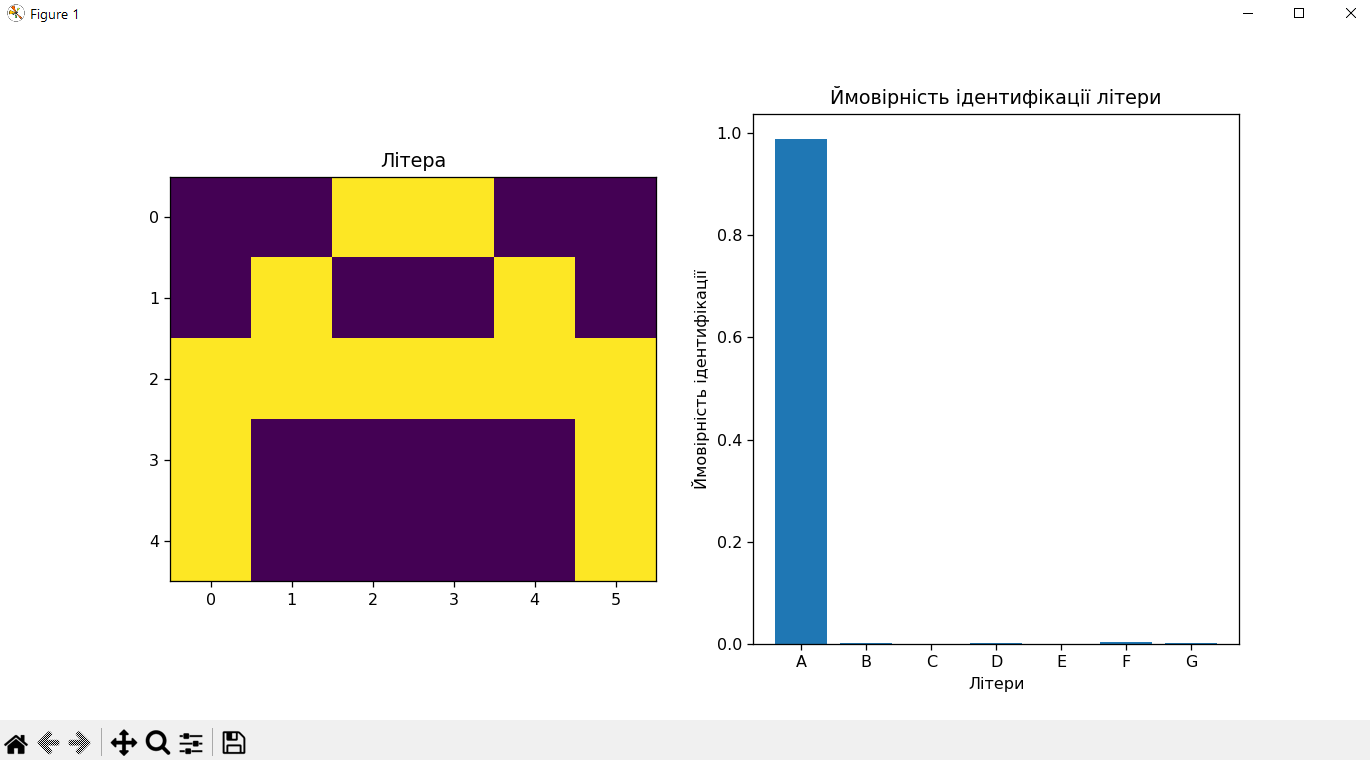


Рисунок 6 – Ідентифікація літерала A

1. Ідентифікація літерала B:

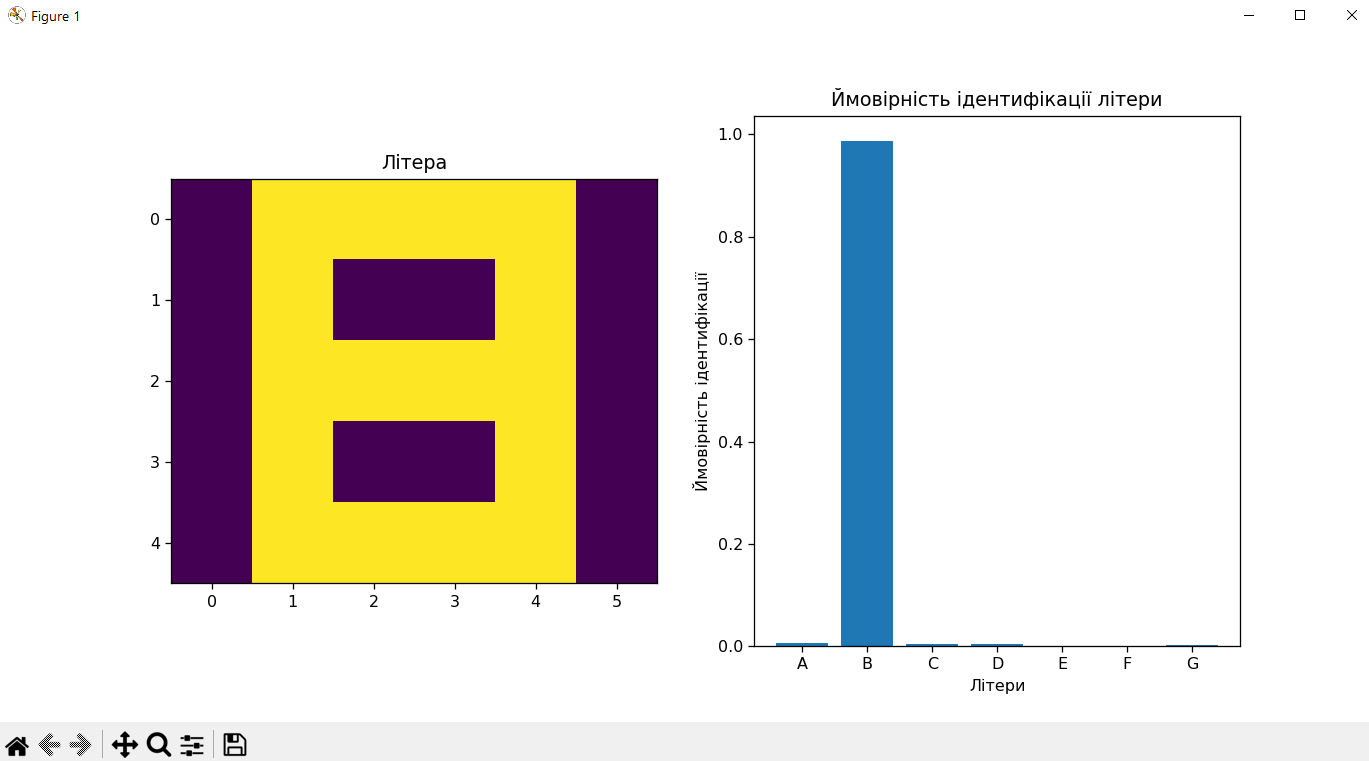


Рисунок 7 – Ідентифікація літерала B

1. Ідентифікація літерала C:

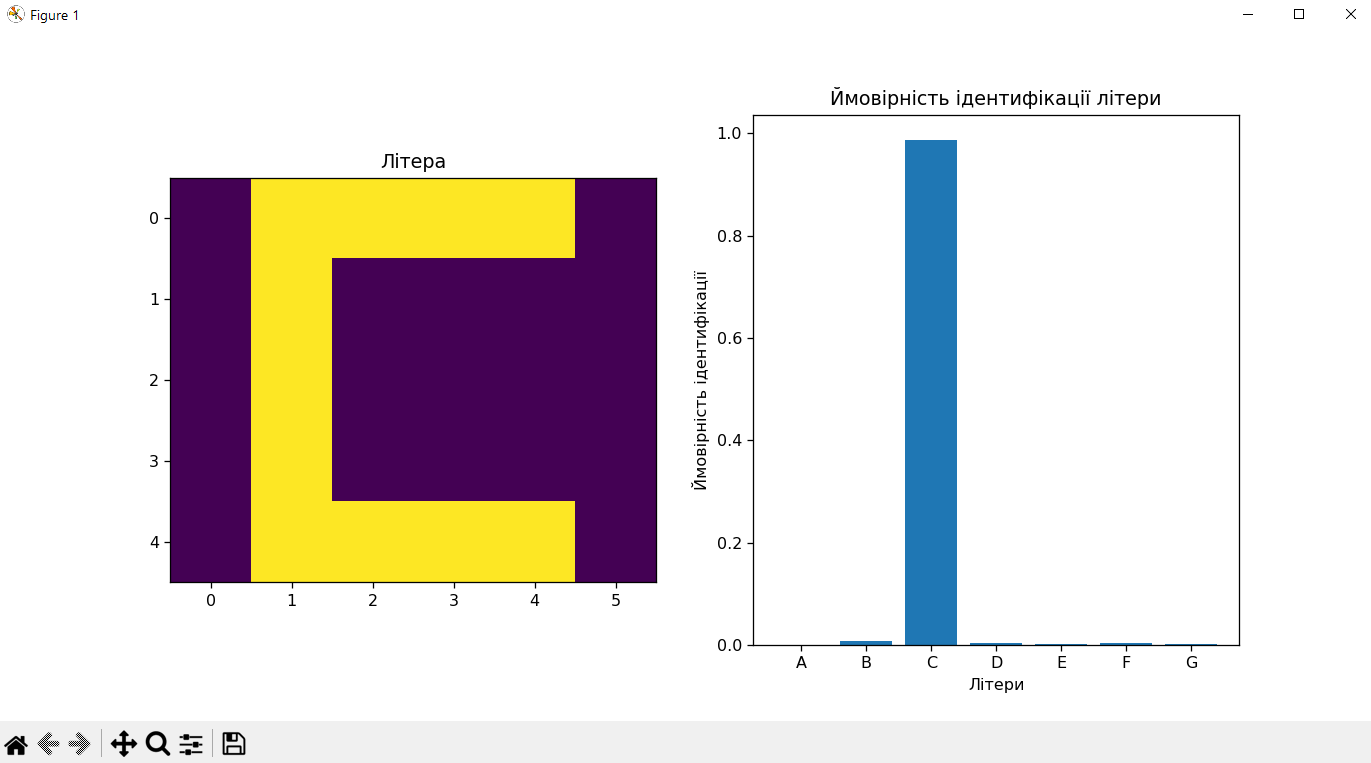


Рисунок 8 – Ідентифікація літерала C

1. Ідентифікація літерала D:

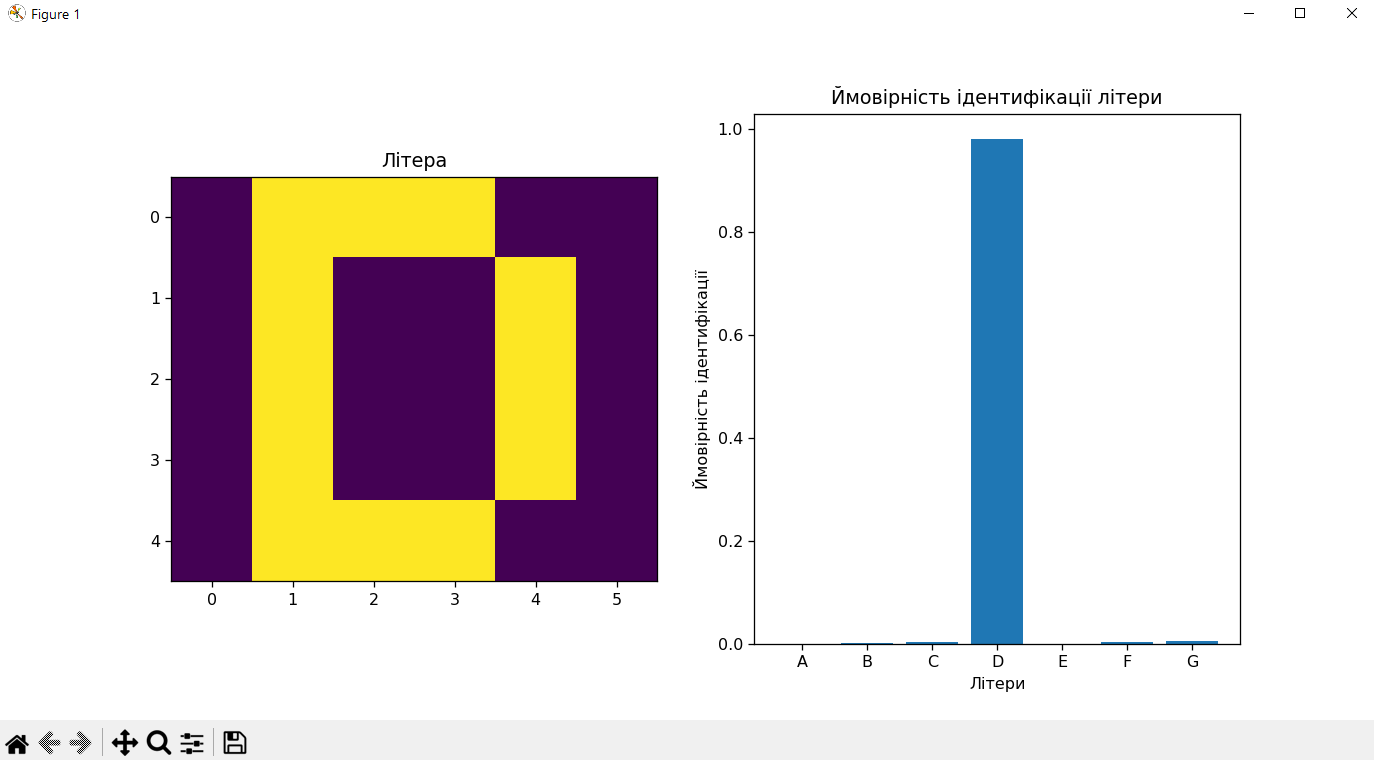


Рисунок 9 – Ідентифікація літерала D

1. Ідентифікація літерала E:

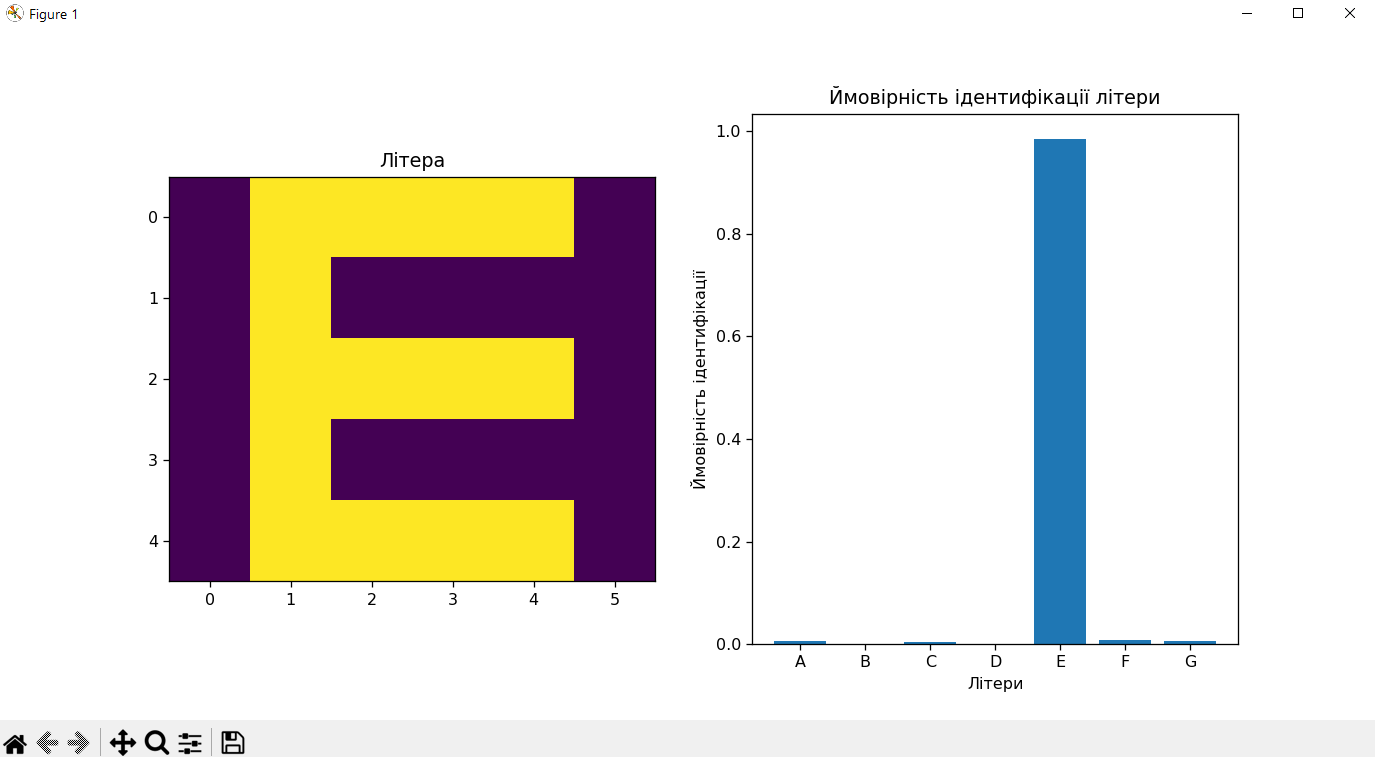


Рисунок 10 – Ідентифікація літерала E

1. Ідентифікація літерала F:

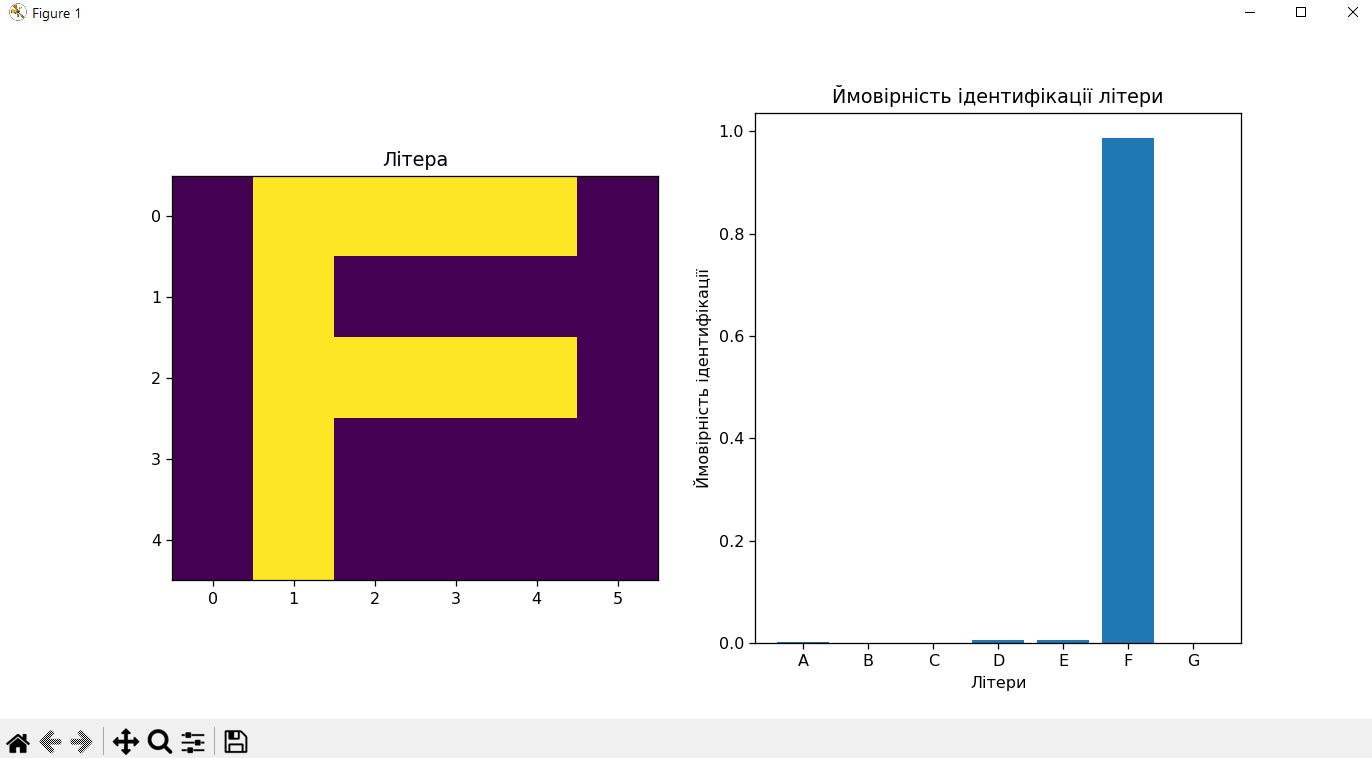


Рисунок 11 – Ідентифікація літерала F

1. Ідентифікація літерала G:

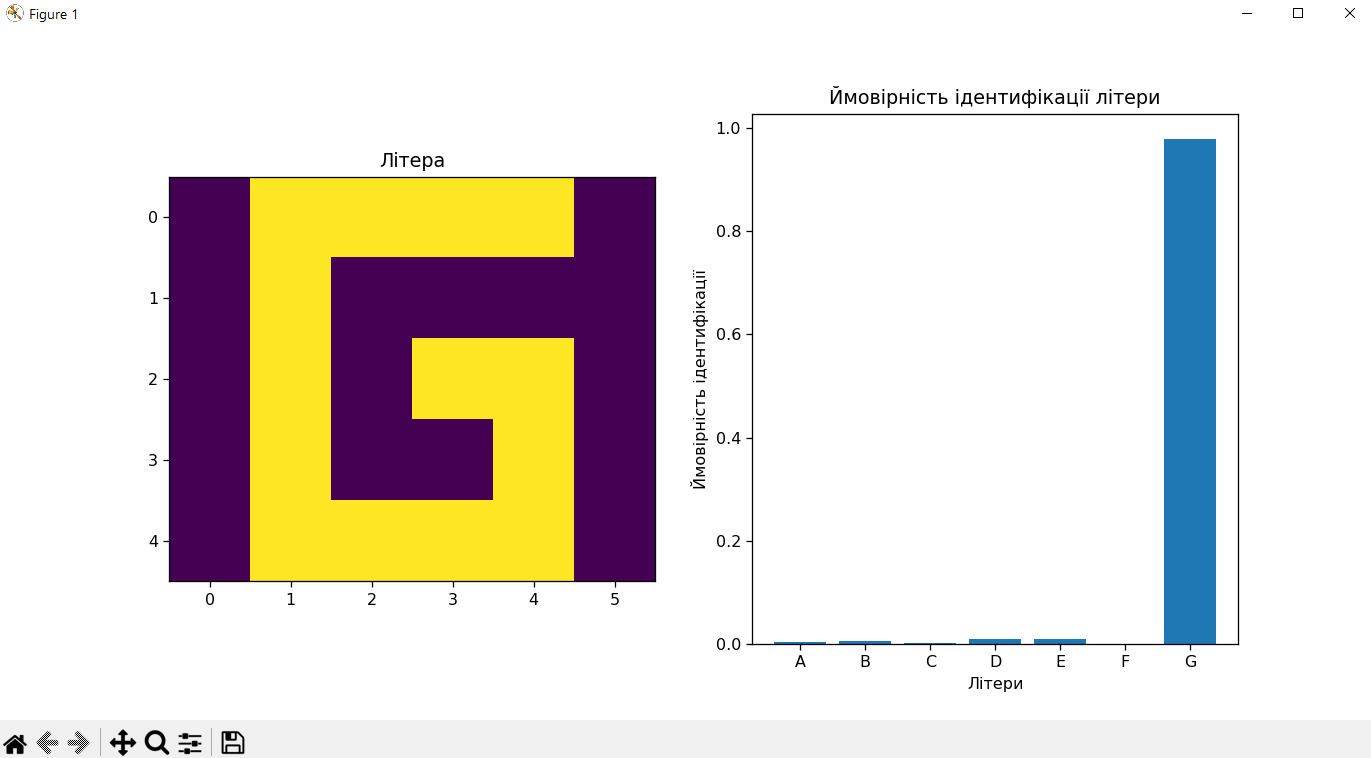


Рисунок 12 – Ідентифікація літерала G

Бачимо, що нейронна мережа правильно ідентифікувала всі літерали з дуже високою точністю, а отже, вона навчена правильно.

* 1. **Програмний код, що забезпечує отримання результату (допускається у формі скриншотів)**

Програмний код послідовно реалізує алгоритм на рис.1 та спрямований на отримання результатів, поданих на рис.3-12.

Для спрощення програмного коду і раціоналізації обчислень застосовано функціональні механізми створення підпрограм.

При цьому використано можливості Python бібліотек: matplotlib та numpy.

Контекстні коментарі пояснюють сутність окремих скриптів наведеного коду програми.

main.py

*"""  
Розпізнавання літералів з допомогою 'сирої' нейромережі  
"""  
  
import* numpy *as* np  
*import* matplotlib.pyplot *as* plt  
  
  
*# Літерали*A = [0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1]  
B = [0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0]  
C = [0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0]  
D = [0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0]  
E = [0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0]  
F = [0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0]  
G = [0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0]  
letters = ['A', 'B', 'C', 'D', 'E', 'F', 'G']  
  
  
*def* generate\_weights(in\_neurons, out\_neurons):  
 *""" Випадкова генерація ваг """  
 return* np.random.randn(in\_neurons, out\_neurons)  
  
  
*def* sigmoid(value):  
 *""" Сигмоїдна функція активації """  
 return* 1 / (1 + np.exp(-value))  
  
  
*def* loss(out, y, out\_number):  
 *""" Обчислення MSE для виводу мережі """* s = (np.square(out - y))  
 s = np.sum(s) / out\_number  
 *return* s  
  
  
*class* NeuralNetwork:  
 *""" Нейронна мережа """  
  
 def* \_\_init\_\_(*self*, neurons\_numbers):  
 *""" Ініціалізація ваг """  
 self*.weights = [generate\_weights(neurons\_numbers[index], neurons\_numbers[index + 1]) *for* index *in* range(len(neurons\_numbers) - 1)]  
  
 *def* feed\_forward(*self*, x):  
 *""" Рух мережею """* a = x  
 *for* weight *in self*.weights:  
 z = a.dot(weight)  
 a = sigmoid(z)  
 *return* a  
  
 *def* back\_propagation(*self*, x, y, alpha):  
 *""" Зворотне поширення помилки """  
  
 # Обчислення виходів мережі на різних шарах* activations = [x]  
 *for* weight *in self*.weights:  
 z = activations[-1].dot(weight)  
 a = sigmoid(z)  
 activations.append(a)  
  
 *# Обчислення похибок* deltas = [(activations[-1] - y)]  
 *for* i *in* range(len(*self*.weights) - 1, 0, -1):  
 d = np.multiply(*self*.weights[i].dot(deltas[0].transpose()).transpose(),  
 np.multiply(activations[i], 1 - activations[i]))  
 deltas.insert(0, d)  
  
 *# Обчислення градієнтів* weight\_adjustments = [activations[i].transpose().dot(d) *for* i, d *in* enumerate(deltas)]  
  
 *# Оновлення ваг  
 for* i *in* range(len(*self*.weights)):  
 *self*.weights[i] -= alpha \* weight\_adjustments[i]  
  
 *return  
  
 def* train(*self*, X, Y, alpha=0.01, epoch=10):  
 *""" Тренування моделі """* acc = []  
 losses = []  
 *for* j *in* range(1, epoch + 1):  
 l = []  
 *for* i *in* range(len(X)):  
 out = *self*.feed\_forward(X[i])  
 l.append(loss(out, Y[i], len(Y)))  
 *self*.back\_propagation(X[i], y[i], alpha)  
 *if* j % 10 == 0:  
 print('epochs:', j, '======== acc:', (1 - (sum(l) / len(X))) \* 100)  
 acc.append((1 - (sum(l) / len(X))) \* 100)  
 losses.append(sum(l) / len(X))  
 *return* acc, losses, *self*.weights  
  
 *def* predict(*self*, x):  
 *""" Виконання передбачення та відображення результатів """* out = *self*.feed\_forward(x)  
  
 fig, axs = plt.subplots(1, 2, figsize=(12, 6))  
  
 axs[0].imshow(np.array(x).reshape(5, 6))  
 axs[0].set\_title('Літера')  
  
 axs[1].bar(range(len(out[0])), out[0])  
 axs[1].set\_xlabel('Літери')  
 axs[1].set\_ylabel('Ймовірність ідентифікації')  
 axs[1].set\_title('Ймовірність ідентифікації літери')  
 axs[1].set\_xticks(range(len(out[0])), letters)  
  
 plt.show()  
 *return* out  
  
  
*def* visualize\_letters(letters):  
 *""" Відображення літер """  
 for* index *in* range(len(letters)):  
 plt.subplot(3, 3, index + 1)  
 plt.imshow(np.array(letters[index]).reshape(5, 6))  
 plt.show()  
  
  
dataset = [A, B, C, D, E, F, G]  
visualize\_letters(dataset)  
x = [np.array(letter).reshape(1, 30) *for* letter *in* dataset]  
y = np.eye(len(dataset), dtype=int)  
  
*# Ініціалізація нейронної мережі  
# Перший (вхідний) шар - 30 нейронів, другий (прихований) шар - 9 нейронів, третій (вихідний) шар - 7 нейронів*neurons = [30, 9, 7]  
neural\_network = NeuralNetwork(neurons)  
  
*# Тренування мережі*acc, losses, weights = neural\_network.train(x, y, 0.1, 1000)  
*# Натреновані ваги*print('Ваги', weights, sep='\n')  
  
*# Візуалізація точності та втрат мережі під час навчання*plt.plot(acc)  
plt.title('Точність по епохах')  
plt.ylabel('Точність')  
plt.xlabel('Епохи')  
plt.show()  
  
plt.plot(losses)  
plt.title('Втрати по епохах')  
plt.ylabel('Втрати')  
plt.xlabel('Епохи')  
plt.show()  
  
*# Ідентифікація літер  
for* index *in* range(len(x)):  
 result = neural\_network.predict(x[index])  
 print(f'Ідентифікація літери {letters[index]}: [{" ".join(["{:.4f}".format(item) *for* item *in* result[0]])}], очікуваний результат: {y[index]}')

## **IV. Висновки.**

У ході виконання лабораторної роботи було проведено дослідження принципів та особливостей підготовки даних, синтезу, навчання та застосування штучних нейронних мереж (Artificial Neural Networks) для практичних задач ідентифікації в технологіях Computer Vision. Для цього було обрано датасет з 7 бінарних зображень літералів, виконано створення штучної нейронної мережі та її навчання і повторна перевірка результатів на датасеті. Нейронна мережа успішно розпізнає усі літерали з високою точністю, тому завдання лабораторної роботи виконано успішно.

Виконав: студент Трикош І. В.