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

Виконала:
Студентка 3 курсу
Навчальної групи ІС-12
Мельникова К.О.

Перевірив:
Професор кафедри ОТ ФІОТ
Писарчук О.О.

Київ 2024

Мета роботи:

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

І рівень складності – максимально 8 балів.

Відповідно до технічних умов, табл.1 додатку.

3	Розробити програмний скрипт, що забезпечує ідентифікацію бінарних зображень 4 спеціальних знаків, заданих матрицею растра. Для ідентифікації синтезувати, навчити та застосувати штучну нейронну мережу в «сирому» вигляді реалізації матричних операцій. Обгрунтувати вибір архітектури та алгоритму навчання нейромережі. Довести працездатність та ефективність синтезованої нейронної мережі.

Програмна реалізація:

```
import numpy as np
import matplotlib.pyplot as plt

def data_x ():
   # знак питання
   question = [0, 0, 1, 0, 0,
               0, 1, 0, 1, 0,
               0, 0, 0, 1, 0,
               0, 0, 1, 0, 0,
               0, 0, 0, 0, 0,
               0, 0, 1, 0, 0]

   # знак оклику
   exclamation = [0, 0, 1, 0, 0,
                  0, 0, 1, 0, 0,
                  0, 0, 1, 0, 0,
                  0, 0, 1, 0, 0,
                  0, 0, 0, 0, 0,
                  0, 0, 1, 0, 0]

   # стрілка вверх
   up = [0, 0, 1, 0, 0,
         0, 1, 1, 1, 0,
         1, 0, 1, 0, 1,
         0, 0, 1, 0, 0,
         0, 0, 1, 0, 0,
```

```
        0, 0, 1, 0, 0]

    # стрілка вниз
    down = [0, 0, 1, 0, 0,
            0, 0, 1, 0, 0,
            0, 0, 1, 0, 0,
            1, 0, 1, 0, 1,
            0, 1, 1, 1, 0,
            0, 0, 1, 0, 0]

    x = [
        np.array(question).reshape(1, 30),
        np.array(exclamation).reshape(1, 30),
        np.array(up).reshape(1, 30),
        np.array(down).reshape(1, 30),
    ]

    plt.subplot(1, 5, 1)
    plt.imshow(np.array(question).reshape(6, 5))
    plt.subplot(1, 5, 2)
    plt.imshow(np.array(exclamation).reshape(6, 5))
    plt.subplot(1, 5, 3)
    plt.imshow(np.array(up).reshape(6, 5))
    plt.subplot(1, 5, 4)
    plt.imshow(np.array(down).reshape(6, 5))
    plt.show()

    return x

def data_y(): # мітки
    out_abcfs = [[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0],  [0, 0, 0, 1]]
    y = np.array(out_abcfs)

    return y

def sigmoid(x):
    return (1 / (1 + np.exp(-x)))

def forward(x, w1, w2):
    # прихований прошарок
    z1 = x.dot(w1)
    a1 = sigmoid(z1)

    # вихідний прошарок
    z2 = a1.dot(w2)
```

```
    a2 = sigmoid(z2)

    return (a2)

def generate(x, y):
    l = []
    for _ in range(x * y):
        l.append(np.random.randn())
    return (np.array(l).reshape(x, y))


def loss(out, Y):
    s = (np.square(out - Y))
    s = np.sum(s) / len(y)
    return (s)


def back_prop(x, y, w1, w2, alpha):
    # прихований прошарок
    z1 = x.dot(w1)
    a1 = sigmoid(z1)

    # вихідний прошарок
    z2 = a1.dot(w2)
    a2 = sigmoid(z2)

    d2 = (a2 - y)
    d1 = np.multiply((w2.dot((d2.transpose()))).transpose(),
                     (np.multiply(a1, 1 - a1)))

    w1_adj = x.transpose().dot(d1)
    w2_adj = a1.transpose().dot(d2)

    w1 = w1 - (alpha * (w1_adj))
    w2 = w2 - (alpha * (w2_adj))

    return (w1, w2)

def train(x, Y, w1, w2, alpha=0.01, epoch=10):
    acc = []
    loss_val = []
    for j in range(epoch):
        l = []
        for i in range(len(x)):
            out = forward(x[i], w1, w2)
```

```
            l.append((loss(out, Y[i])))
            w1, w2 = back_prop(x[i], y[i], w1, w2, alpha)
        print("Епохи: ", j + 1, " Точність: ", (1 - (sum(l) / len(x))) * 100)
        acc.append((1 - (sum(l) / len(x))) * 100)
        loss_val.append(sum(l) / len(x))
    return (acc, loss_val, w1, w2)


def predict(x, w1, w2):
    res = forward(x, w1, w2)
    max_w = 0
    k = 0
    for i in range(len(res[0])):
        if (max_w < res[0][i]):
            max_w = res[0][i]
            k = i
    return k

def print_results(x, w1, w2):
    symbols = ['?', '!', '↑', '↓']
    for i in range(4):
        expected_letter = symbols[i]
        recognized_letter = symbols[predict(x[i], w1, w2)]
        print(f"Очікувано: {expected_letter} | Розпізнано: {recognized_letter} " f"|
{True if expected_letter == recognized_letter else False}")
```

Навчання та результат виконання:

Ваги:

```
w1 = generate(30, 4)
w2 = generate(4, 4)
print('Ініціалізуємо вагові коефіцієнти: ')
print('w1 = ', w1, '\n')
print('w2 = ', w2, '\n')
```

[11] ✓ 0.0s Python

```
Ініціалізуємо вагові коефіцієнти:
w1 =  [[-0.63714911  1.52543487  0.32005264 -1.03788838]
 [-0.66012024 -2.97608139  0.74902071 -0.99224103]
 [ 0.74430756 -0.17061836  0.78475622 -0.55846111]
 [-1.17559769 -0.04553881 -0.48620752 -0.35191312]
 [-0.6348418  -2.15666476 -0.58937361  0.09550053]
 [-1.16074747  0.84275807 -0.78656169  1.3396177 ]
 [ 0.44915159  0.71631728  0.42464862 -1.76574292]
 [ 0.64837951  0.88412104 -0.08884521 -1.02068274]
 [ 0.55622173  0.65402617  0.42008595 -1.48603413]
 [-0.52498653 -0.20805336 -0.21291438 -1.38063212]
 [-0.55054042  2.84617353 -0.53637294 -0.17972896]
 [-0.56808168  0.744125    1.80111487  1.48428293]
 [-1.13422635  1.56794954  1.14862826 -0.01458495]
 [-0.18326485 -0.51215006 -1.09872594  1.70691862]
 [ 0.2169977   0.79064201  0.87727583 -0.5207639 ]
 [ 1.48914897  0.0202928  -1.74414444  0.62205947]
 [-0.23865225  0.3374149  -0.19303674 -0.49185642]
 [ 0.28771754  0.00312851 -0.6644406   1.82998618]
 [ 0.15394766  1.7903301   0.2741812  -1.30335881]
 [ 0.08524512 -0.1889118   0.22791207  0.4412469 ]
 [-0.08024051  0.02049277 -0.83683503  0.37843626]
 [-1.79017048 -0.84649848 -1.01112165 -1.47692603]
 [ 0.43300855 -0.53139724 -0.4295835   1.52827371]
```


```
 [-0.358941    0.40142622 -0.77491362  0.48293419]
 [ 1.45936096  0.3325341   0.52294432 -1.20771769]
 [ 1.52155652 -1.40751268 -0.75016596  0.78285741]
 [-1.20298693 -1.40651916  2.47812291 -1.97819206]
 [-1.65360982  0.19668953 -0.15096577  1.95875742]
 [-0.06330464 -0.92845968 -1.17876908 -0.08224536]
 [ 0.66005479 -1.17760522 -0.1722859   1.48377104]]

w2 =  [[-1.06479125  0.49530106 -1.29542407 -1.1920512 ]
 [ 1.25613826 -1.23734772  0.26125108 -0.70694022]
 [-0.07942815  0.40222605 -0.54835984 -0.14090347]
 [-0.34165643 -1.10712565 -1.21977688 -0.55892103]]
```

```
    print('Тренування мережі')
    acc, loss_value, w1, w2 = train(x, y, w1, w2, 0.1, 300)
```

[12] ✓ 0.0s

```
Тренування мережі
Епохи:  1  Точність:  76.64759800071124
Епохи:  2  Точність:  77.85820303546606
Епохи:  3  Точність:  78.81347904425968
Епохи:  4  Точність:  79.5800890502244
Епохи:  5  Точність:  80.21389824457746
Епохи:  6  Точність:  80.75543447822437
Епохи:  7  Точність:  81.23221398593996
Епохи:  8  Точність:  81.66244308903579
Епохи:  9  Точність:  82.0581618554396
Епохи:  10  Точність:  82.4274732870215
Епохи:  11  Точність:  82.7759975991063
Епохи:  12  Точність:  83.10777749879877
Епохи:  13  Точність:  83.4258275253909
Епохи:  14  Точність:  83.73246562771428
Епохи:  15  Точність:  84.02951632708947
Епохи:  16  Точність:  84.31843900194423
Епохи:  17  Точність:  84.60041172880378
Епохи:  18  Точність:  84.87638766804348
Епохи:  19  Точність:  85.14713374394748
Епохи:  20  Точність:  85.4132576546216
Епохи:  21  Точність:  85.67522735468826
Епохи:  22  Точність:  85.93338612423453
Епохи:  23  Точність:  86.18796566156325
Епохи:  24  Точність:  86.43909904843912
...
Епохи:  297  Точність:  99.57435609478733
Епохи:  298  Точність:  99.57780353024062
Епохи:  299  Точність:  99.5812136950765
Епохи:  300  Точність:  99.58458708974656
```

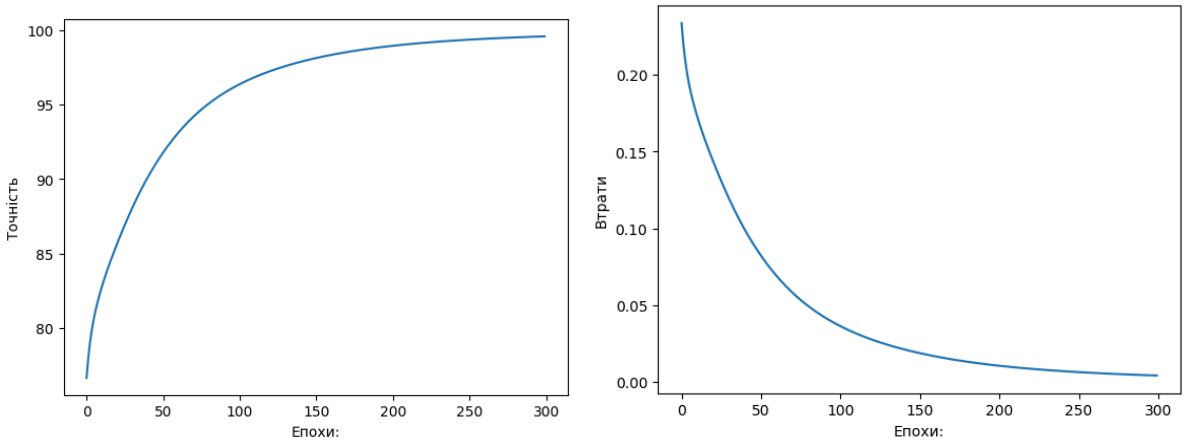
Результати:

```
    print('Вагови коефіцієнти після навчання: ')
    print('w1 = ', w1, '\n')
    print('w2 = ', w2, '\n')

    plt.plot(acc)
    plt.ylabel('Точність')
    plt.xlabel("Епохи:")
    plt.show()

    plt.plot(loss_value)
    plt.ylabel('Втрати')
    plt.xlabel("Епохи:")
    plt.show()

    print_results(x, w1, w2)
```

] ✓ 0.0s

```
Вагови коефіцієнти після навчання:
w1 =  [[-0.63714911  1.52543487  0.32005264 -1.03788838]
 [-0.66012024 -2.97608139  0.74902071 -0.99224103]
 [ 1.19598683 -0.73171975  0.73700607 -0.84150086]
 [-1.17559769 -0.04553881 -0.48620752 -0.35191312]
 [-0.6348418  -2.15666476 -0.58937361  0.09550053]
 [-1.16074747  0.84275807 -0.78656169  1.3396177 ]
 [-0.76469744  2.37127126 -0.48966206 -1.86501936]
 [ 1.68794763 -1.19634497  1.31917843 -1.73011859]
 [-0.6576273   2.30898014 -0.49422474 -1.58531057]
 [-0.52498653 -0.20805336 -0.21291438 -1.38063212]
 [-1.17650061  2.98176288  0.00509018 -0.7054015 ]
 [-0.56808168  0.744125    1.80111487  1.48428293]
 [-0.09465823 -0.51251647  2.5566519  -0.7240208 ]
 [-0.77115369  1.00721456 -2.55449974  2.13331472]
 [-0.40896249  0.92623137 -1.41873894 -1.04643644]
```

```
w2 =  [[-1.64216789  2.5494375  -2.10624553 -2.71093248]
 [ 3.14544934 -5.11524978  1.66997047 -4.43138139]
 [-5.53681056  1.84020936  1.59456974 -2.26926766]
 [-0.83090198 -2.92988097 -5.77044195  2.85719004]]
```




```
Очікувано: ? | Розпізнано: ? | True
Очікувано: ! | Розпізнано: ! | True
Очікувано: ↑ | Розпізнано: ↑ | True
Очікувано: ↓ | Розпізнано: ↓ | True
```

Як можна побвчити, програма коректно ідентифікує 4 зазначені спеціальні символи: знак питання, знак оклику та стрілки вверх та вниз. Кожен символ представлений матрицею розміром 6*5.

Опис моделі

Дана модель навчається за допомогою алгоритму зворотного поширення помилки, який використовується для корекції вагових коефіцієнтів у мережі.

Створена модель нейронної мережі складається з трьох шарів: вхідного, прихованого та вихідного.

Вхідний шар: Вхідними даними для мережі є зображення символів, які подаються у вигляді векторів. Кожен вектор має розмірність 30, так як кожен символ представлено матрицею 6*5.

Прихований шар: Після вхідного шару відбувається передача даних у прихований шар, де використовується сигмоїдальна функція активації.

Вихідний шар: Після прихованого шару вихідні дані передаються у вихідний шар, де вони класифікуються відповідно до визначених чотирьох класів.

Також експерементально було визначено оптимальну кількість епох, що дорівнює 300.

Висновок

У ході виконання даної лабораторної роботи було досліджено принципи та особливості підготовки даних, навчання та застосування штучних нейронних мереж для задач ідентифікації об’єктів на цифрових зображеннях у рамках технологій Computer Vision.

В ході виконання роботи була реалізована модель нейронної мережі, яка складається з трьох шарів: вхідного, прихованого та вихідного. Вхідні дані представляли собою матриці розміром 6x5, що описували спеціальні символи: знак питання, знак оклику та стрілки вверх і вниз. Модель використовувала алгоритм зворотного поширення помилки для корекції вагових коефіцієнтів та сигмоїдальну функцію активації в прихованому шарі.

Результати навчання показали, що створена модель коректно ідентифікує всі чотири зазначені символи. Це підтверджує правильність обраного підходу та ефективність використаних алгоритмів.