**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”**

**Інститут прикладного системного аналізу**
**Кафедра системного проектування**

# ЗВІТ

З виконання лабораторної роботи №3
з дисципліни “Методи та системи штучного інтелекту”
на тему: “Розпізнавання візуальних образів за допомогою мережі Хопфілда”

Виконав:
студент 4 курсу
групи ДА-82
Муравльов А.Д.
Варіант №18

Київ — 2021

**Мета роботи:** здобути навички програмної реалізації машинного навчання на базі нейронних мереж.

**Завдання:** Дослідити методологію машинного навчання в задачі класифікації візуальних образів за допомогою нейронної мережі Хопфілда на прикладі чорно-білих образів літер абетки.

**Варіант індивідуального завдання:**

Літери для розпізнання: Ї, Т, Р

**Теоретичні відомості**

Нейронна мережа Хопфілда – повнозв’язана нейронна мережа [10] із симетричною матрицею зв'язків. В процесі роботи динаміка таких мереж збігається (конвергує) до одного з положень рівноваги. Ці положення рівноваги є локальними мінімумами функціонала, який називається енергією мережі (в найпростішому випадку - локальними мінімумами негативно певної квадратичної форми на n-вимірному кубі). Така мережа може бути використана як автоасоціативна пам'ять, як фільтр, а також для вирішення деяких завдань оптимізації.

На відміну від багатьох нейронних мереж, що працюють до отримання відповіді через певну кількість тактів, мережі Хопфілда працюють до досягнення рівноваги, коли наступний стан мережі в точності дорівнює попередньому: початковий стан є вхідним чином, а при рівновазі отримують вихідний образ.

Припустимо, у нас є певна кількість еталонних образів - зображень, або ще чого-небудь. Нам дають якийсь спотворений образ, і наше завдання полягає в тому, щоб «розпізнати» в ньому один з еталонних. Таку задачу може вирішити нейронна мережа Хопфілда [8].

Штучна нейронна мережа являє собою систему нейронів , взаємодіючих між собою на зразок справжньої нейронної мережі мозку. Штучна мережа «навчається» рішенням деякої задачі, що, по суті, зводиться до обчислень вагових коефіцієнтів матриці, без будь-якої «магії».

Кожен нейрон мережі отримує і передає сигнали іншим. Те, як нейрони пов'язані між собою, залежить від типу мережі. Мережа Хопфілда є одношарової мережею, тому що в ній використовується лише один шар нейронів. Вона так само є рекурсивної мережею, тому що володіє зворотними зв'язками. Вона функціонує циклічно. Погляньте на приклад мережі Хопфілда з чотирьох нейронів. Кожен з них має виходи сигналу, який подаються на входи всіх інших нейронів, крім себе самого:

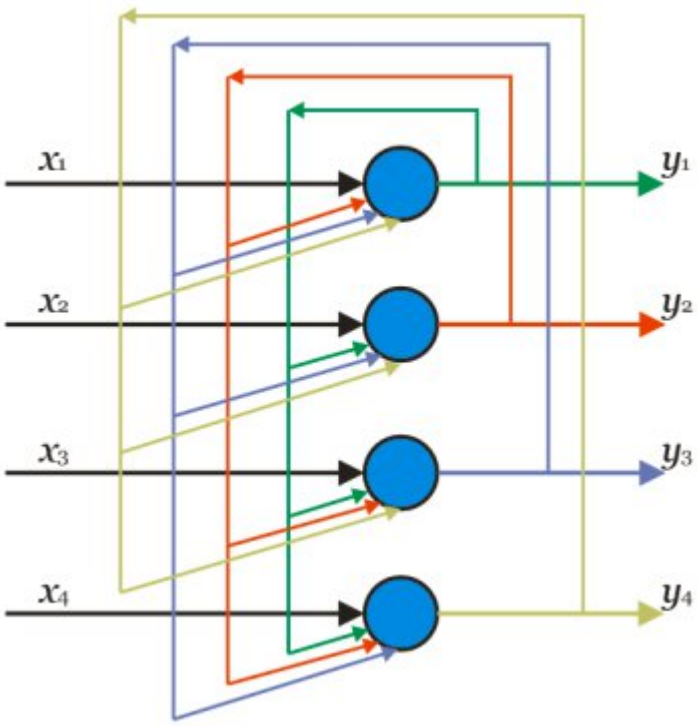



Рис.3.1 Архітектура нейронної мережі Хопфілда

Однак цю мережу можна навчити практично нічому. Нам потрібно набагато більше нейронів. Мережа, що містить N нейронів може запам'ятати не більше ~ 0.15 * N образів. Таким чином реальна мережа повинна містити досить значну кількість нейронів. Це одне з істотних недоліків мережі Хопфілда - невелика ємність. Плюс до всього образи не повинні бути дуже схожі один на одного, інакше в деяких випадках можливе зациклення при розпізнаванні.

Нейронна мережа Гопфілда — це тип рекурентної, повнозв'язної, штучної нейронної мережі з симетричною матрицею зв'язків. У процесі роботи динаміка таких мереж сходиться (конвергує) до одного з положень рівноваги.

Ці положення рівноваги є локальними мінімумами функціоналу, що називається енергія мережі (у найпростішому випадку — локальними мінімумами негативно визначеної квадратичної форми на n-вимірному кубі).

**Хід роботи**

Для виконання лабораторної роботи обрано мову програмування Python 3.10 з задіянням бібліотек PIL, numpy, matploptlib.

Повний лістинг коду також викладено у репозиторій Github за посиланням:

/**/

Програма буде містити у собі навчальну вибірку та тестову вибірку зображень згідно варіанту. У тренувальній вибірці знаходяться файли якими мережа буде тренуватись, у тестовій вибірці знаходяться файли які мережа буде розпізнавати.

Тренувальна вибірка:

ji.png

p.png

t.png

Тестова вибірка:

ji.png

p.png

t.png

**Короткий опис функціоналу програми:**

Програма містить у собі функції для роботи з зображеннями.

Функція imgToHopfieldMatrix приймає в себе зображення та інтерфейс з параметрами зображення та границею чорного. За допомогою бібліотеки PIL зображення зчитується та повертається в форматі матриці зі значеннями кольорів в форматі ЧБ. Далі зображення перетворюється в unit8 (байтовий) масив та байти які мають значення більше границі чорного (білі пікселі) заповнюються значенням 1, чорні ж заповнюються значенням -1. Аналогічно реалізовано зворотню функцію для перетворення розпізнаної матриці мережею у зображення, щоб його можна було вивести.

Лістинг функцій:

```
#image processing functions
#process image from byte representation to [1,-1] matrix
def imgToHopfieldMatrix(image, opts: IImgOpts):
    img = Image.fromImage(image).convert("L").resize(opts.imgSize)
    imgArr = np.asarray(img, dtype=np.unit8)
    hopfieldMatrix = np.zeros(opts.imgSize, dtype=float)
```

```
    hopfieldMatrix[imgArr > opts.blackLimit] = 1
    hopfieldMatrix[imgArr == 0] = -1
    return hopfieldMatrix

#process matrix back to image
def hopfieldMatrixToImg(imgArr):
    convertArray = np.zeros(imgArr.shape, dtype=np.unit8)
    convertArray[imgArr == 1] = 255
    convertArray[imgArr == -1] = 0
    img = Image.fromarray(convertArray, mode = "L")
    return img
```

Навчання мережі заключається у тому щоб знайти ваги матриці так щоб заповнити m векторів (еталонних образів, що складають пам’ять системи):

```
def setWeights(etalonImage):
    weights = np.zeros([len(etalonImage), len(etalonImage)])
    for i in range(len(etalonImage)):
        for j in range(i, len(etalonImage)):
            weights[i, j] = 0 if i == j else etalonImage[i] *
etalonImage[j]
            weights[j, i] = weights[i, j]
    return weights
```

Функція, що відповідає за тренування мережі:

```
def trainNetwork(img, opts):
    weights = None
    for i,path in enumerate(img):
        matrix = imgToHopfieldMatrix(path, opts)
        vector = matrixToVec(matrix)
        plt.imshow(hopfieldMatrixToImg(matrix))
        plt.title("Зображення на якому проводилось тренування")
        plt.show()
        if i == 0:
            weights = setWeights(vector)
        else:
            weights = weights + setWeights(vector)
```

```
    weights = weights / len(img)
    return weights
```

Функція, що відповідає за розпізнавання:

```
def imgRecognition(weights, img, imgOpts, networkOpts):
    matrix = imgToHopfieldMatrix(img, imgOpts)
    vector = matrixToVec(matrix)
    imgBeforeRecognition = hopfieldMatrixToImg(matrix)
    #оновлення вагів для зображення що розпізнається
    recVector, iters, energies = updateWeights(weights, vector,
networkOpts.theta, networkOpts.iters)
    #виведення зображення до розпізнавання
    plt.subplot(2,2,1)
    plt.imshow(imgBeforeRecognition)
    plt.title("Зображення, що розпізнається")
    #виведення зображення після розпізнавання
    plt.subplot(2,2,2)
    plt.imshow(hopfieldMatrixToImg(recVector.reshape(matrix.shape)))
    plt.title('Розпізнане зображення')
    #статистика навчання мережі
    plt.subplot(2,1,2)
    plt.plot(iters, energies)
    plt.ylabel("Локальні мінімуми")
    plt.xlabel("Ітерація")
```

**Результати роботи програми**

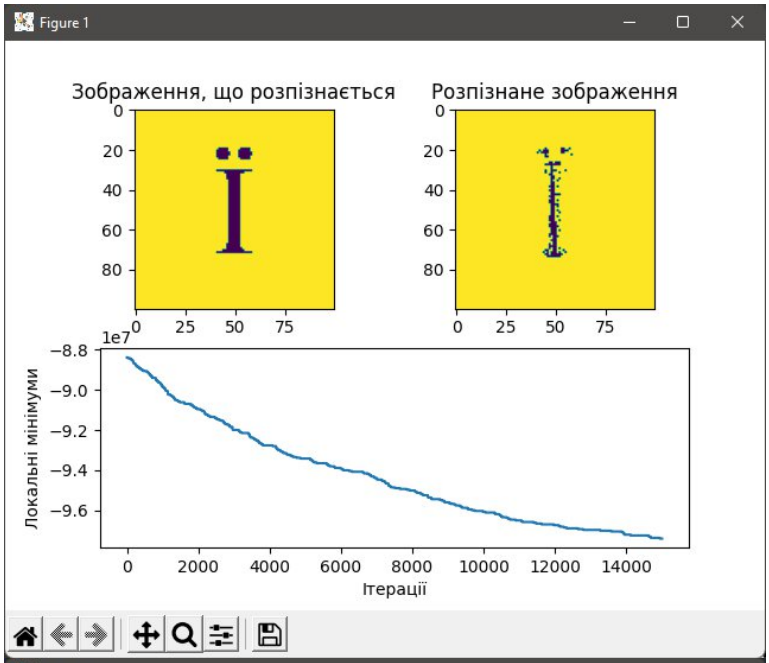


Figure 1
Зображення, що розпізнається
Розпізнане зображення
1e7
Локальні мінімуми
Ітерації


Figure 1
Зображення на якому проводилось тренування

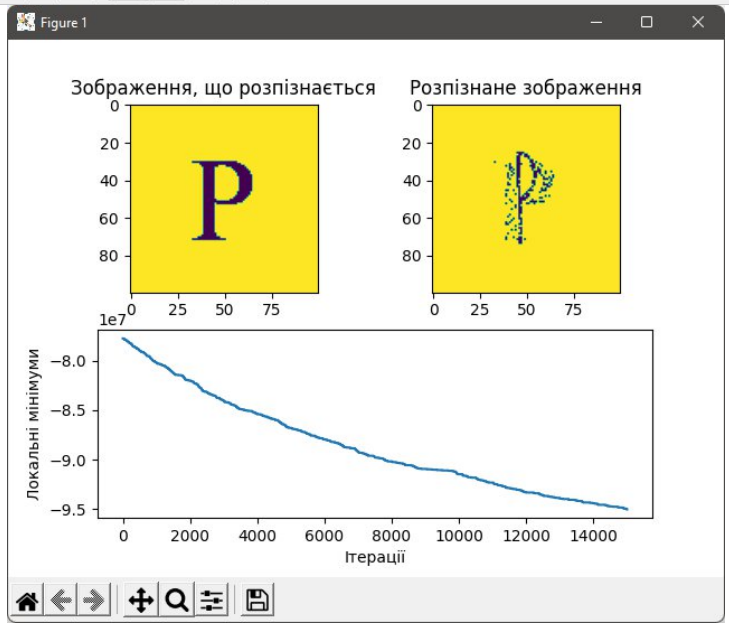
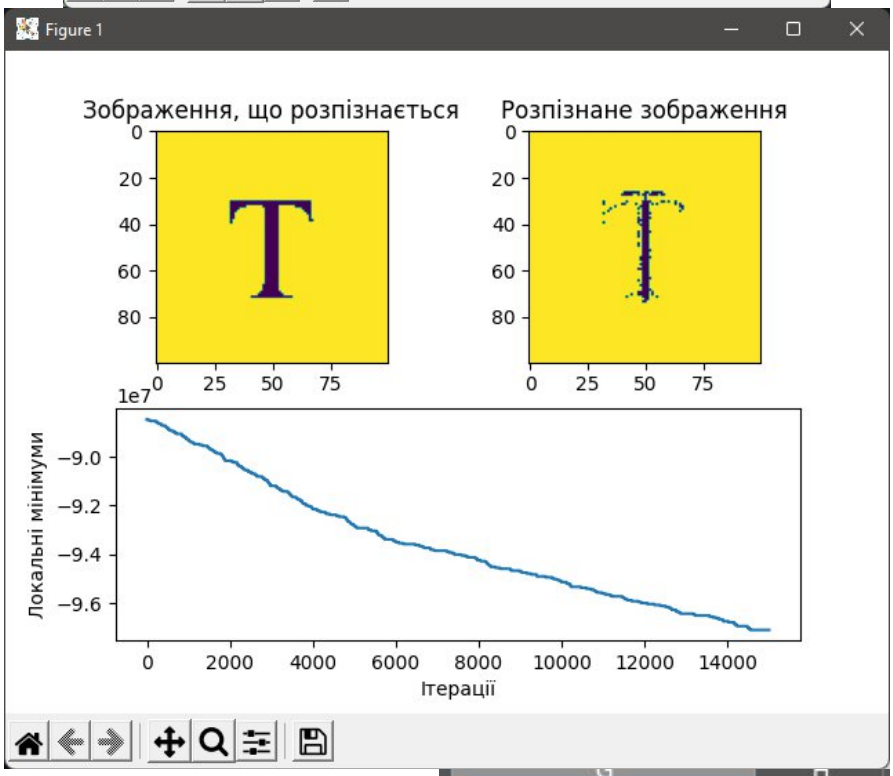


Figure 1
Зображення, що розпізнається
Розпізнане зображення
1e7
Локальні мінімуми
Ітерації


| | |
|---|---|
| Формат зображення | 100x100 |
| Границя чорного | 145 |
| Тета | 0.75 |
| К-ть ітерацій | 15000 |
| Ї | |
| Час | 944сек |
| Р | |
| Час | 798сек |
| Т | |
| Час | 947сек |

| | |
|---|---|
| Формат зображення | 50x50 |
| Границя чорного | 145 |
| Тета | 0.75 |
| К-ть ітерацій | 15000 |
| Ї | |
| Час | 72сек |
| Р | |
| Час | 69сек |
| Т | |
| Час | 111сек |

**Висновок**

Бачимо, що за 15000 ітерацій з коефіцієнтом тета 0.75 вдалося розпізнати літеру в середньому за 850с для зображення 100х100рх та близько 75ск для зображення 50х50рх, з чого можемо зробити висновок що зі збільшенням формату зображення час навчання та розпізнавання зростає експоненційно