Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Метод обробки зображень з використанням штучних нейронних мереж
(тема)

Виконав:
студент II курсу, групи КСМзм-20-1
Лебеденко В.Е.
(прізвище, ініціали)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп’ютерні системи та мережі
(повна назва освітньої програми)

Керівник: доц. Барковська О.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ ______________ Коваленко А.А.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри ____________
(підпис)
“____” ______________ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Лебеденку Владиславу Едуардовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Метод обробки зображень з використанням штучних нейронних мереж

затверджена наказом по університету від “ 25 ” жовтня 2021 р. № 169 Стз

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2021 р.

3. Вхідні дані до роботи
машинне навчання
карти Кохонена
метод рою часток

4. Перелік питань, що потрібно опрацювати у роботі
Аналіз предметної області
Нейромережеве стиснення зображень
Евристичний алгоритм тренування штучної нейронної мережі
Тестування програмних засобів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп’ютерних ілюстрацій (слайдів) 13 слайдів ф.А4

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання. Пошук літератури	27.10.2021–12.11.2021	
2	Порівняльний аналіз сучасних методів	13.11.2021–25.11.2021	
3	Розробка методу	26.11.2021–28.11.2021	
4	Реалізація методу	29.11.2021–05.12.2021	
5	Аналіз результатів	06.12.2021–08.12.2021	
6	Оформлення ПЗ	09.12.2021–12.12.2021	

Дата видачі завдання 27 жовтня 2021 р.

Студент ____________________
(підпис)

Керівник роботи ____________________ проф. Міхаль О.П.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 71 с., 19 рис., 2 табл., 1 дод., 22 джерела.

ЕВРИСТИЧНИЙ АЛГОРИТМ, ШТУЧНА НЕЙРОННА МЕРЕЖА, РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ.

Метою кваліфікаційної роботи є аналіз евристичних методів обробки зображень з використанням машинного навчання, зокрема штучних нейронних мереж

Розроблена тестова архітектура нейронної мережі для розпізнавання рукописних цифр та підібрано дані для тренування мережі. Проведено експериментальні дослідження, які показали, що реалізований алгоритм є конкурентним та може давати прийнятні результати при тренуванні нейронних мереж.

ABSTRACT

Master's thesis: 71 pages, 19 figures, 2 tables, 1 appendices, 22 sources.

HEURISTIC ALGORITHM, ARTIFICIAL NEURAL NETWORK, IMAGE RECOGNITION.

The purpose of the qualification work is the analysis of heuristic methods of image processing using machine learning, in particular artificial neural networks The test architecture of the neural network for handwritten digit recognition was developed and data for network training were selected. Experimental studies have shown that the implemented algorithm is competitive and can give acceptable results in the training of neural networks.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МН – машинне навчання

ПЗ – програмне забезпечення

ШНМ – штучна нейронна мережа

ВСТУП

Враховуючи те, що в наші дні, обсяг даних із зображеннями зростає експоненційно в найширшому розмаїтті додатків - у тому числі дистанційне зондування, земляне спостереження, медичне зображення, цифрові бібліотеки та документи, HDTV, розвага та фільм, відеоконференції, так і потреби для запам’ятовування, відновлення та передачі зображень розширюються, цифрова компресія зображення стає навіть найбільш критично необхідною технологією. Багато з цих прикладних областей - у тому числі «земляне спостереження», відеоконференції та багато програм, взятих на озброєння у військову сферу, використовують технології з послідовностями зображень, що зображають деяку форму руху. Наприклад, послідовності картин, взятих супутником кожного разу, коли це проходить приблизно через таке ж розтягування території, після відповідного повторного приміщення та компенсації, - це наступні зразки такого ж місця дії, що містить зміни за рахунок руху об’єктів (наприклад, транспорт), або за рахунок зміни метеорологічних умов. Тому компресія може мати велику фактичну перевагу в тому, що послідовності зображення потрібно лише утримують трек змін, які походять від однієї споруди до наступної.

У деяких областях (як, наприклад, медицина) зазвичай мають справу з сіро-горизонтальними зображеннями, в інших же областях програми має справу з надзвичайно кольоровими зображеннями (наприклад, в мультимедійному розвагі). Якість оброблюваного або стисливого зображення оцінено зовсім інакше, так різні сфери мають справу з різними колірними гамами або їх відсутністю. У випадку кольору, прийнятна якість зображення буде значною мірою залежати від програми.

В даний час ШНМ знаходять широке застосування в різних сферах, і однієї, важливою, з яких є стиснення зображення.

Поряд з існуючою традиційною технологією компресії зображення, представленої серіями JPEG, MPEG, в даний час розвивається нова технологія, що реалізує елементи обчислювального інтелекту (наприклад, штучні нейронні мережі (ШНМ) та генетичні алгоритми).

В даний час ШНМ знаходять широке застосування в різних сферах, і однієї, важливою, з яких є стиснення зображення.

Метою роботи є аналіз евристичних методів обробки зображень з використанням машинного навчання, зокрема штучних нейронних мереж.

Завдання:

- проведення аналізу існуючих алгоритмів розпізнавання об’єктів на зображеннях за допомогою штучних нейронних мереж;

- вибір алгоритму розпізнавання зображень для реалізації в системі;

- реалізація модифікованого методу тренування штучних нейронних мереж;

- коректна робота розробленого ПЗ з зашумленими зображеннями.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Біологічний нейрон та його моделі

Розвиток штучних нейронних мереж надихається біологією. Тобто, розглядаючи мережеві зміни та алгоритми, дослідники застосовують терміни, запозичені із принципів організації мозкової діяльності. Але на цьому аналогія закінчується. Наші знання про роботу мозку настільки обмежені, що мало знайшлося б точно доведених закономірностей для тих, хто побажав би керуватися ними. Тому розробникам мереж доводиться виходити межі сучасних біологічних знань у пошуках структур, здатних виконувати корисні функції. У багатьох випадках це призводить до необхідності відмови від біологічної правдоподібності, мозок стає просто метафорою, і створюються мережі, неможливі в живій матерії або вимагають неправдоподібно великих припущень про анатомію та функціонування мозку.

Незважаючи на те, що зв’язок з біологією слабкий і часто несуттєвий, штучні нейронні мережі продовжують порівнювати з мозком. Їхнє функціонування часто має зовнішню подібність із людським пізнанням, тому важко уникнути цієї аналогії. На жаль, такі порівняння є неплідними і створюють невиправдані очікування, що неминуче ведуть до розчарування.

Нервова система людини, побудована з елементів, званих нейронами, має приголомшливу складність. Біля нейронів беруть участь у приблизно передавальних зв’язках, що мають довжину метр і більше. Кожен нейрон має багато властивостей, загальними з іншими органами тіла, але йому притаманні абсолютно унікальні здібності: приймати, обробляти і передавати електрохімічні сигнали по нервових шляхах, які утворюють комунікаційну систему мозку.

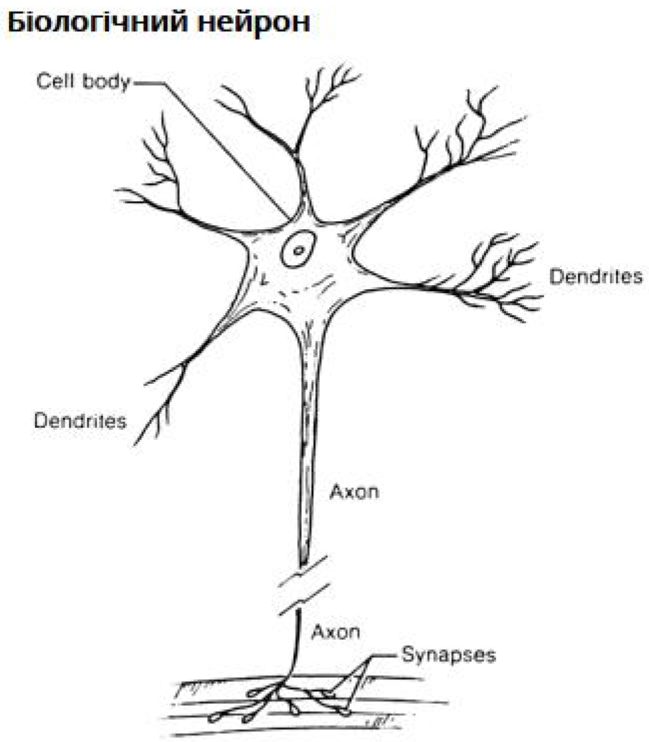

Рисунок 1.1 – Біологічний нейрон

На рисунку 1.1 показано структуру пари типових біологічних нейронів. Дендрити йдуть від тіла нервової клітини до інших нейронів, де вони приймають сигнали в точках з'єднання, які називають синапсами. Прийняті синапс вхідні сигнали передаються до тіла нейрона. Тут вони підсумовуються, причому одні входи прагнуть порушити нейрон, інші перешкодити його збудженню.

Коли сумарне збудження у тілі нейрона перевищує певний поріг, нейрон збуджується, посилаючи на аксон сигнал іншим нейронам. Ця основна функціональна схема має багато ускладнень і винятків, проте більшість штучних нейронних мереж моделюють лише ці прості властивості.

Штучний нейрон імітує у першому наближенні властивості біологічного нейрона. На вхід штучного нейрона надходить кілька сигналів, кожен із яких є виходом іншого нейрона. Кожен вхід множиться на відповідну вагу, аналогічну синаптичній силі, і всі твори сумуються, визначаючи рівень активації нейрона. На рисунку 1.2 представлено модель, що реалізує цю ідею. Хоча мережеві парадигми дуже різноманітні, в основі багатьох їх лежить ця конфігурація. Тут безліч вхідних сигналів, позначених x1, x2, ..., xn, надходить на штучний нейрон. Ці вхідні сигнали, в сукупності, що позначаються вектором X, відповідають сигналам, що надходять синапси біологічного нейрона [1]. Кожен сигнал множиться на відповідну вагу w1, w2, ..., wn, і надходить на сумирний блок, позначений Σ. Кожна вага відповідає силі одного біологічного синаптичного зв’язку. (Багато ваг у сукупності позначається вектором W). Підсумовуючий блок, що відповідає тілу біологічного елемента, складає зважені входи алгебраїчно, створюючи вихід – NET. У векторних позначеннях може бути компактно записано таким чином: NET = XW.

Сигнал NET перетворюється на активаційну функцію F і дає вихідний нейронний сигнал OUT [2]. Активаційна функція може бути звичайною лінійною функцією

$$OUT = K(NET), \quad (1.1)$$

де K - постійна, порогової функції

$$OUT = 1, \text{ якщо } NET > T,$$
$$OUT = 0 \text{ в інших випадках,}$$

де T - деяка постійна порогова величина, або функцією, більш точно моделює нелінійну передатну характеристику біологічного нейрона і представляє нейронної мережі великі можливості.

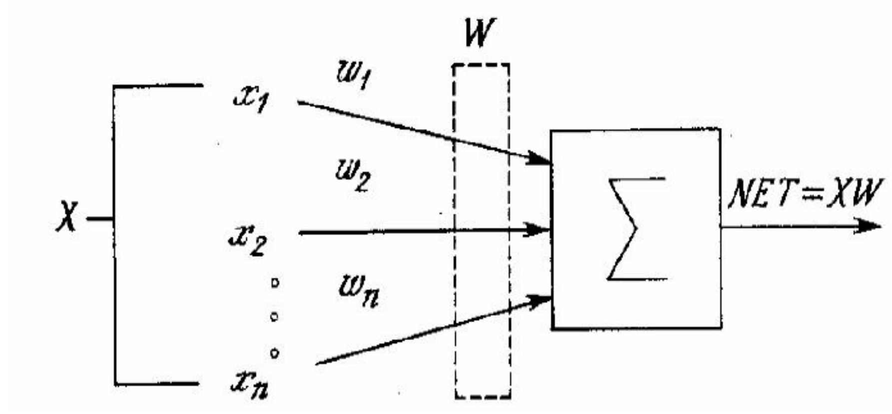

Рисунок 1.2 – Штучний нейрон

На рисунку 1.3 блок, позначений F приймає сигнал NET і видає сигнал OUT. Якщо блок F звужує діапазон зміни величини NET так, що при будь-яких значеннях NET значення OUT належать деякому кінцевому інтервалу, то F називається функцією, що «стискає». Як «стискаючу» функцію часто використовується логістична або «сигмоїдальна» (S-подібна) функція, показана на рис. 1.7. Ця функція математично виражається як F(x) = 1/(1 + e-^{x}). Таким чином,

$$OUT = \frac{1}{1 + e^{-NET}} \qquad (1.2)$$

За аналогією з електронними системами активаційну функцію можна вважати нелінійною підсилювальною характеристикою штучного нейрона. Коефіцієнт посилення обчислюється як відношення збільшення величини OUT до невеликого збільшення величини NET, що викликало його. Він виражається нахилом кривої при певному рівні збудження і змінюється від малих значень при великих негативних збудженнях (крива майже горизонтальна) до максимального значення при нульовому збудженні і знову зменшується, коли збудження стає більшим позитивним. Гроссберг виявив, що така нелінійна характеристика вирішує поставлену їм дилему шумового насичення [3]. Яким чином одна й та сама мережа може обробляти як слабкі, і сильні сигнали? Слабкі сигнали потребують великого мережевого посилення, щоб дати придатний для використання вихідний сигнал. Однак підсилювальні каскади з великими коефіцієнтами посилення можуть призвести до насичення виходу шумами підсилювачів (випадковими флуктуаціями), які є в будь-якій фізично реалізованій мережі. Сильні вхідні сигнали у свою чергу також будуть приводити до насичення підсилювальних каскадів, виключаючи можливість корисного використання виходу. Центральна область логістичної функції, що має великий коефіцієнт посилення, вирішує проблему обробки слабких сигналів, у той час як області з падаючим посиленням на позитивному та негативному кінцях підходять для великих збуджень. Таким чином, нейрон функціонує з великим посиленням u1074 у широкому діапазоні рівня вхідного сигналу.

$$OUT = \frac{1}{1 + e^{-NET}} = F(NET).$$

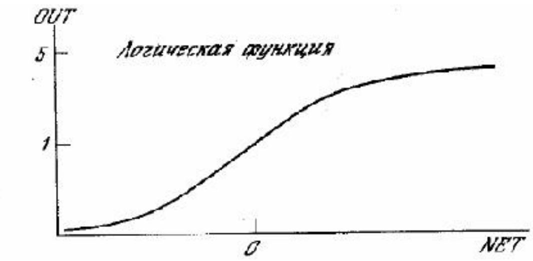

Рисунок 1.3 – Сигмоідальна функція

Іншою широко використовуємою активаційною функцією є гіперболічний тангенс. За формою він подібний до логістичної функції і часто використовується біологами як математична модель активації нервової клітини. Як активаційну функцію штучної нейронної мережі вона записується наступним чином:

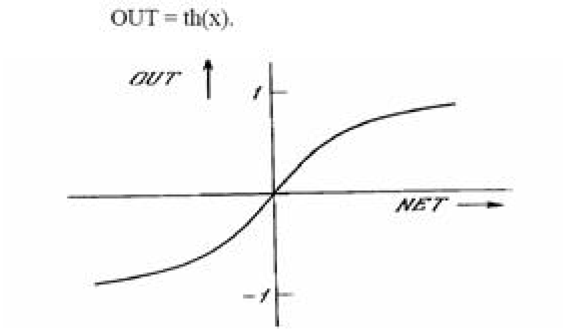

Рисунок 1.4 – Функція гіперболічного тангенсу

Подібно до логістичної функції, гіперболічний тангенс є S-подібною функцією, але він симетричний щодо початку координат, і в точці NET = 0 значення вихідного сигналу OUT дорівнює нулю (дивитися рисунок 1.5). На відміну від логістичної функції, гіперболічний тангенс набуває значень різних знаків, що виявляється вигідним для низки мереж.

Розглянута проста модель штучного нейрона ігнорує багато властивостей свого біологічного двійника. Наприклад, вона не бере до уваги затримки у часі, що впливають на динаміку системи. Вхідні сигнали відразу породжують вихідний сигнал. І, що важливіше, вона не враховує впливу функції частотної модуляції або синхронізуючої функції біологічного нейрона, які низка дослідників вважають вирішальними. Незважаючи на ці обмеження, мережі, побудовані з цих нейронів, виявляють властивості, що сильно нагадують біологічну систему.

1.2 Види штучних нейронних мереж

1.2.1 Одношарові та багатошарові ШНМ прямого поширення

Хоча один нейрон і здатний виконувати найпростіші процедури розпізнавання, сила нейронних обчислень походить від сполук нейронів у мережах. Найпростіша мережа [4] складається з групи нейронів, що утворюють шар, як показано у правій частині рисунок 1.6. Вершини-кола зліва служать лише розподілу вхідних сигналів. Вони не виконують будь-яких обчислень і тому не будуть вважатися шаром. Тому вони позначені колами, щоб відрізняти їх від обчислювальних нейронів, позначених квадратами. Кожен елемент з множини входів X окремою вагою з'єднаний з кожним штучним нейроном, а кожен нейрон видає зважену суму входів у мережу. У штучних та біологічних мережах багато сполук можуть бути відсутніми, всі сполуки показані з метою спільності. Можуть мати місце також з'єднання між виходами та входами елементів у шарі [5].

Ваги всіх елементів матриці можна позначити через W. Матриця має m рядків та n стовпців, де m – число входів, а n – число нейронів. Наприклад, w2,3 – це вага, що пов'язує третій вхід із другим нейроном. Таким чином, обчислення вихідного вектора N, компонентами якого є виходи OUT нейронів, зводиться до матричного множення N = XW, де N та X – вектори рядки.

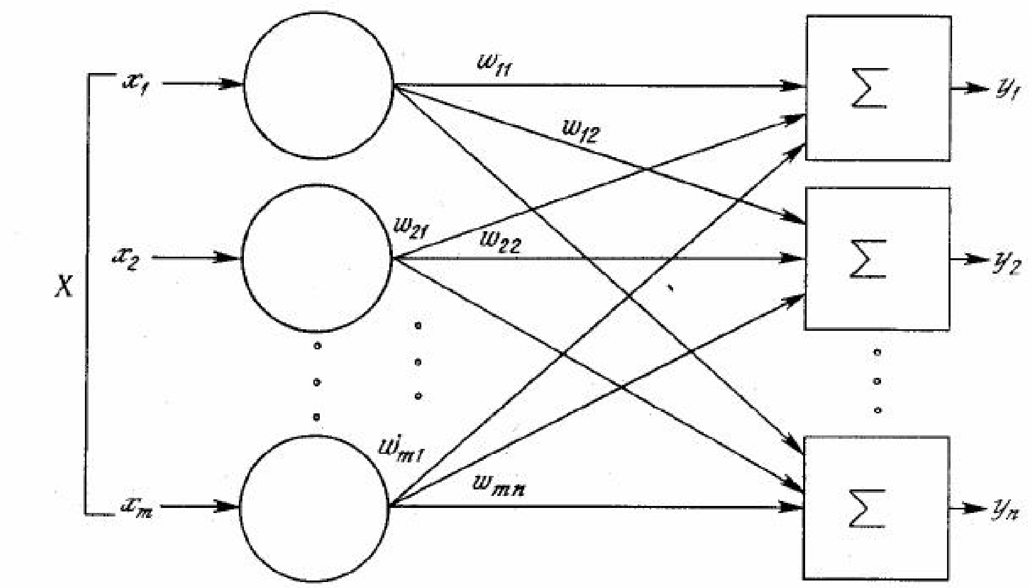

Рисунок 1.5 – Одношарова нейронна мережа

Найбільші та складні нейронні мережі мають, як правило, і великі обчислювальні можливості [5]. Хоча створені мережі всіх змін, які тільки можна собі уявити, пошарова організація нейронів копіює шаруваті структури певних відділів мозку. Такі багатошарові мережі мають більші можливості, ніж одношарові.

Багатошарові мережі можуть утворюватися каскадами шарів. Вихід одного шару є входом наступного шару [6,7]. Подібна мережа показана на рисунку 1.7 і знову зображена з усіма з’єднаннями. Багатошарові мережі що неспроможні призвести до збільшення обчислювальної потужності проти одношаровою мережею лише тому випадку, якщо активаційна функція між шарами буде нелінійної [8]. Обчислення виходу шару полягає у множенні вхідного вектора на першу вагову матрицю з наступним множенням (якщо відсутня нелінійна активаційна функція) результуючого вектора на другу вагову матрицю.

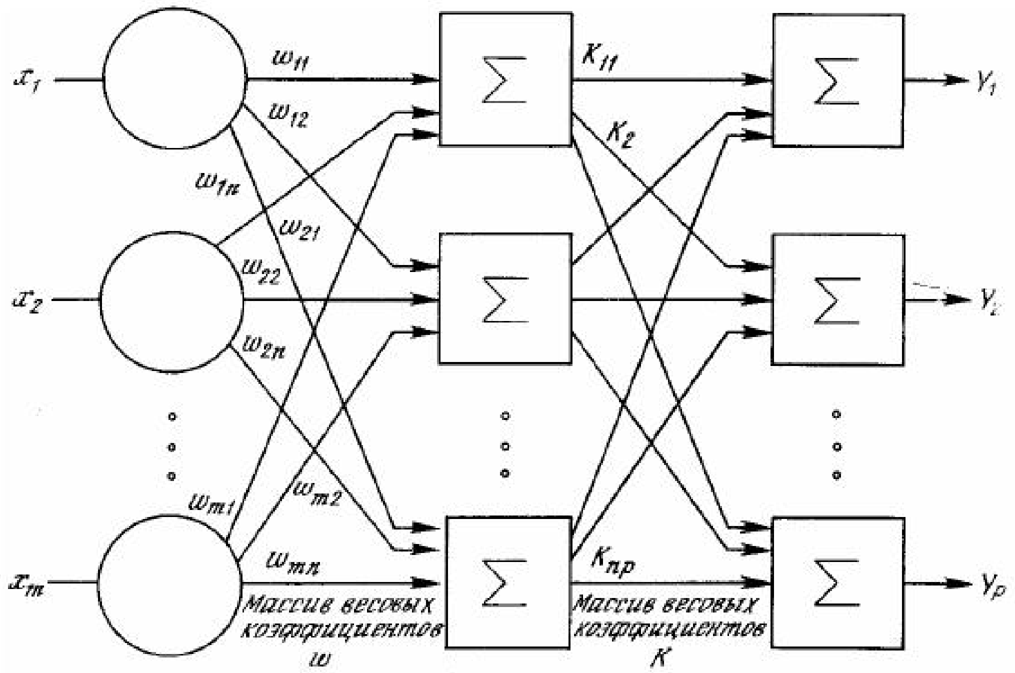

Рисунок 1.6 – Двошарова нейронна мережа

Оскільки множення матриць асоціативно, то X(W1W2). Це показує, що двошарова лінійна мережа еквівалентна одному шару з ваговою матрицею, що дорівнює добутку двох вагових матриць [9]. Отже, будь-яка багатошарова лінійна мережа може бути замінена еквівалентною одношаровою мережею, однак одношарові мережі дуже обмежені за своїми обчислювальними можливостями. Таким чином, для розширення можливостей мереж порівняно з одношаровою мережею потрібна нелінійна активаційна функція.

1.2.2 Нейронна мережа зі зворотним розповсюдженням помилки

Коли в мережі лише один шар, алгоритм її навчання з учителем очевидний, тому що правильні вихідні стани нейронів єдиного шару свідомо відомі, і підстроювання синаптичних зв’язків йде у напрямку, що мінімізує помилку на виході мережі. За цим принципом будується, наприклад, алгоритм навчання одношарового персептрона. У багатошарових мережах оптимальні вихідні значення нейронів всіх шарів, крім останнього, як правило, не відомі, і двох або більше шаровий персептрон вже неможливо навчити, керуючись тільки величинами помилок на виходах ШНМ.

У мереж, розглянутих вище, був зворотних зв’язків, т. е. з’єднань, які від виходів деякого шару до входів цього ж шару чи попередніх шарів. Цей спеціальний клас мереж, які називаються мережами без зворотних зв’язків або мережами прямого розповсюдження. У мереж без зворотних зв’язків немає пам’яті, їх вихід повністю визначається поточними входами та значеннями ваги.

Один з варіантів вирішення цієї проблеми - розробка наборів вихідних сигналів, що відповідають вхідним, для кожного шару ШНМ, що, звичайно, є дуже трудомісткою операцією і не завжди можливо. Другий варіант - динамічне підстроювання вагових коефіцієнтів синапсів, в ході якої вибираються, як правило, найбільш слабкі зв’язки і змінюються на малу величину в той чи інший бік, а зберігаються ті зміни, які спричинили зменшення помилки на виході всієї мережі. Третій варіант – поширення сигналів помилки від виходів ШНМ до її входів, у напрямі, зворотному прямому поширенню сигналів у звичайному режимі роботи. Цей алгоритм навчання ШНМ отримав назву процедури зворотного розповсюдження.

Згідно з методом найменших квадратів [10], цільовою функцією помилки ШНМ, що мінімізується, є величина:

$$E(w)=\frac{1}{2}\sum_{j,p}(y_{j,p}^{(N)}-d_{j,p})^2$$

де $y_{j,p}^{(N)}$ – реальний вихідний стан нейрона j вихідного шару N нейронної мережі при подачі її входи p-го образу; d_{jp} – Ідеальний (бажаний) вихідний стан цього нейрона. Підсумовування ведеться по всіх нейронах вихідного шару і по всіх оброблюваних мережею образах. Мінімізація ведеться методом градієнтного спуску, що означає підстроювання вагових коефіцієнтів таким чином:

$$\Delta w_{ij}^{(n)}=-\eta\cdot\frac{\partial E}{\partial w_{ij}} \tag{1.2}$$

Де w_{ij} – ваговий коефіцієнт синаптичного зв’язку, що з’єднує i-ий нейрон шару n-1 з j-им нейроном шару n, η - коефіцієнт швидкості навчання, $0<\eta<1$.

$$\frac{\partial E}{\partial w_{ij}}=\frac{\partial E}{\partial y_j}\cdot\frac{dy_j}{ds_j}\cdot\frac{\partial s_j}{\partial w_{ij}} \tag{1.3}$$

Тут під yj, мається на увазі вихід нейрона j, а під sj - виважена сума його вхідних сигналів, тобто аргумент активаційної функції. Оскільки множник dyj/dsj є похідною цієї функції за її аргументом, з цього випливає, що похідна активаційна функція має бути визначена на всій осі абсцис. У зв’язку з цим функція одиничного стрибка та інші активаційні функції з неоднорідностями не підходять для ШНМ. Вони застосовуються такі гладкі функції, як гіперболічний тангенс або класичний сигмоїд з експонентою [11]. У разі гіперболічного тангенсу

$$\frac{dy}{ds} = 1 - s^2 \tag{1.4}$$

Третій множник ∂sj/∂wij, очевидно, дорівнює виходу нейрона попереднього шару $y_i^{(n-1)}$. Перший множник (1.3), він легко розкладається наступним чином:

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot \frac{\partial s_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot w_{jk}^{(n+1)} \tag{1.5}$$

Тут підсумовування k виконується серед нейронів шару n+1. Ввівши нову змінну

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j} \tag{1.6}$$

ми отримаємо рекурсивну формулу для розрахунків величин $\delta_j^{(n)}$ шару n із більш старшого шару n+1.

$$\delta_j^{(n)} = \left[\sum_k \delta_k^{(n+1)} \cdot w_{jk}^{(n+1)} \right] \cdot \frac{dy_j}{ds_j} \tag{1.7}$$

Для вихідного ж шару

$$\delta_l^{(N)} = (y_l^{(N)} - d_l) \cdot \frac{dy_l}{ds_l} \tag{1.8}$$

Тепер ми можемо записати (1.2) у розкритому вигляді:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot y_i^{(n-1)} \tag{1.9}$$

Іноді для надання процесу корекції ваги деякої інерційності, що згладжує різкі стрибки при переміщенні по поверхні цільової функції доповнюється значенням зміни ваги на попередній ітерації [2]:

$$\Delta w_{ij}^{(n)}(t) = -\eta \cdot (\mu \cdot \Delta w_{ij}^{(n)}(t-1) + (1-\mu) \cdot \delta_j^{(n)} \cdot y_i^{(n-1)}) \tag{1.10}$$

де μ – коефіцієнт інерційності, t - номер поточної ітерації. Таким чином, повний алгоритм навчання ШНМ за допомогою процедури зворотного розповсюдження будується так [1]:

- подати на входи мережі один із можливих образів та в режимі звичайного функціонування ШНМ, коли сигнали поширюються від входів до виходів, розрахувати значення останніх. Відомо що

$$s_j^{(n)} = \sum_{i=0}^{M} y_i^{(n-1)} \cdot w_{ij}^{(n)} \tag{1.11}$$

де M - число нейронів у шарі n-1 з урахуванням нейрона з постійним вихідним станом +1, що задає зміщення; $y_i^{(n-1)}=x_{ij}^{(n)}$ – i-ий вхід нейрона j шару n.

- розрахувати $\delta^{(N)}$ для вихідного шару за формулою (1.8). Розрахувати за формулою (1.9) або (1.10) зміни ваги $\Delta w^{(N)}$ шару N.

- Розрахувати за формулами (1.7) і (1.10) відповідно $\delta^{(n)}$ та $\Delta w^{(n)}$ решти, n=N-1,...1.

- скоригувати всі ваги в ШНМ:

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t)$$

Якщо помилка мережі суттєва, перейти на крок 1. Інакше – кінець.

Мережі на кроці 1 по черзі у випадковому порядку пред’являються всі тренувальні образи [1], щоб мережа не забувала одні образи в міру запам’ятовування інших. Алгоритм схематично представлений рисунку 1.7.

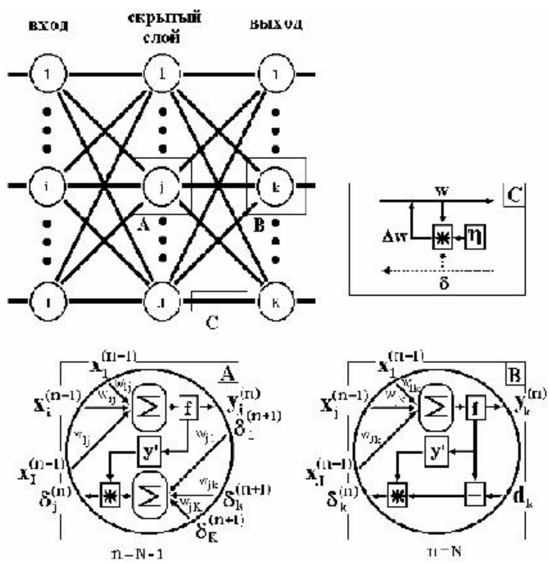

Рисунок 1.7 – Діаграма сигналів у мережі під час навчання за алгоритмом зворотного поширення

З виразу (1.9) випливає, що коли вихідне значення $y_i^{(n-1)}$ прагне нуля, ефективність навчання помітно знижується. При двійкових вхідних векторах в середньому половина вагових коефіцієнтів не коригуватиметься, тому область можливих значень виходів нейронів [0,1] бажано зрушити в межі [-0.5, +0.5], що досягається простими модифікаціями логістичних функцій. Наприклад, сигмоїд з експонентою перетворюється на вид

$$f(x) = -0.5 + \frac{1}{1+e^{-\alpha \cdot x}}.$$

У процесі навчання може виникнути ситуація, коли великі позитивні чи негативні значення вагових коефіцієнтів переміщують робочу точку на сигмоїдах багатьох нейронів у область насичення.

Малі величини похідної від логістичної функції приведуть у відповідність до (1.9) та (1.10) до зупинки навчання, що паралізує ШНМ. По-друге, застосування методу градієнтного спуску не гарантує, що буде знайдено глобальний, а чи не локальний мінімум цільової функції. Ця проблема пов’язана ще з однією, а саме з вибором величини швидкості навчання. Доказ збіжності навчання у процесі зворотного поширення засноване на похідних, тобто збільшення ваги і, отже, швидкість навчання має бути нескінченно малою, проте в цьому випадку навчання відбуватиметься неприйнятно повільно. З іншого боку, занадто великі корекції ваги можуть призвести до постійної нестійкості процесу навчання. Тому як η зазвичай вибирається число менше 1, але дуже маленьке, наприклад, 0.1, і може поступово зменшуватися у процесі навчання. Крім того, для виключення випадкових попадань у локальні мінімуми іноді, після того, як значення вагових коефіцієнтів стабілізуються, η короткочасно сильно збільшують, щоб почати градієнтний спуск з нової точки. Якщо повторення цієї процедури кілька разів приведе алгоритм в той самий стан ШНМ, можна більш-менш впевнено сказати, що знайдено глобальний максимум, а не якийсь інший [2].

Існує і інший спосіб виключення локальних мінімумів, а заодно і паралічу ШНМ, що полягає у застосуванні стохастичних ШНМ [2].

Алгоритм навчання нейронної мережі за допомогою процедури зворотного розповсюдження має на увазі наявність якоїсь зовнішньої ланки, що надає мережі крім вхідних також і цільові вихідні образи. Алгоритми, що користуються подібною концепцією, називають алгоритмами навчання з учителем. Для їхнього успішного функціонування необхідна наявність експертів [2], які створюють на попередньому етапі для кожного вхідного образу еталонний вихідний. Наприклад, навчання людського мозку, на перший погляд, відбувається без вчителя: на зорові, слухові, тактильні та інші рецептори надходить інформація ззовні, і всередині нервової системи відбувається самоорганізація. Однак, не можна заперечувати і того, що в житті людини не мало вчителів – і в буквальному, і в переносному значенні – які координують зовнішні впливи.

Головна риса, що робить навчання без вчителя привабливим, - це його "самостійність". Процес навчання, як і у разі навчання з учителем, полягає у підлаштовуванні ваг синапсів. Деякі алгоритми змінюють структуру мережі, тобто взаємозв’язку нейронів чи навіть їх кількість, такі перетворення називаються самоорганізацією. Очевидно, що підстроювання синапсів може проводитися тільки на підставі інформації, доступної в нейроні, тобто його стану і наявних вагових коефіцієнтів. Виходячи з цього міркування та за аналогією з відомими принципами самоорганізації нервових клітин, побудовані алгоритми навчання Хебба.

Оскільки ця робота присвячена стиску зображення з використанням мережі Кохонена, необхідно розглянути цю мережу докладніше.

1.3 Карти Кохонена

Штучна нейронна мережа Кохонена [1,2] або картка ознак (SOM), що самоорганізується, була запропонована фШНМьким дослідником Тойво Кохоненом на початку 1980-х років.

1.3.1 Структура мережі Кохонена

Вона є двошаровою мережею (рисунок 1.8). Кожен нейрон першого (розподільного) шару з’єднаний з усіма нейронами другого (вихідного) шару, які розташовані у вигляді двомірної ґрати.

Нейрони вихідного шару називають кластерними елементами, їх кількість визначать максимальну кількість груп, на які система може розділити вхідні дані. Збільшуючи кількість нейронів другого шару, можна збільшувати деталізацію результатів процесу кластеризації.

Система працює за принципом змагання [2] - нейрони другого шару змагаються один з одним, перемагає той елемент-нейрон, чий вектор ваг найближче до вхідного вектора сигналів. За мірою близькості двох векторів зазвичай береться евклідова відстань між ними. Таким чином, кожен вхідний вектор відноситься до деякого кластерного елемента.

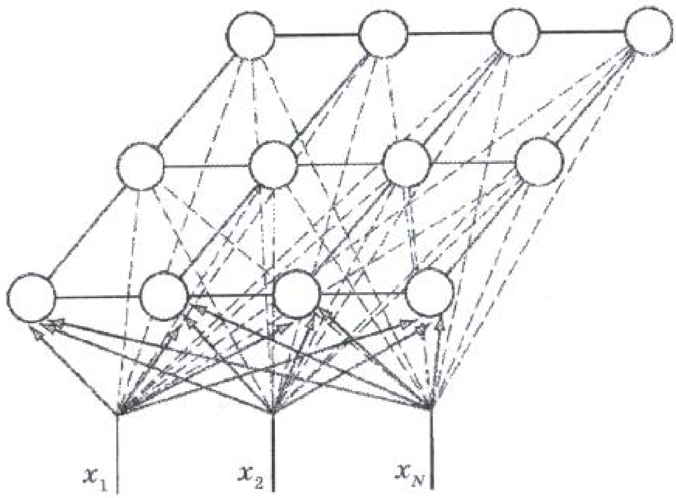

Рисунок 1.8 – Топологія нейронної мережі Кохонена

Алгоритм функціонування самонавчається карт (Self Organizing Maps - SOM) є одним з варіантів кластеризації багатовимірних векторів. Прикладом таких алгоритмів може бути алгоритм k-найближчих середніх (c-means). Важливою відмінністю алгоритму SOM і те, що у ньому все нейрони (вузли, центри класів…) упорядковані у деяку структуру (зазвичай двовимірну сітку). При цьому під час навчання модифікується не тільки нейрон-переможець, а й його сусіди, але меншою мірою. За рахунок цього SOM можна вважати одним із методів проектування багатовимірного простору в простір з нижчою розмірністю. З використанням цього алгоритму вектора, схожі у вихідному просторі, виявляються поруч і отриманої карті.

SOM має на увазі використання впорядкованої структури нейронів. Зазвичай використовуються одне та двомірні сітки. При цьому кожен нейрон є n-мірним вектором-стовпець, де n визначається розмірністю вихідного простору (розмірністю вхідних векторів). Застосування одне і двовимірних сіток пов’язано з тим, що виникають проблеми при відображенні просторових структур більшої розмірності (при цьому знову виникають проблеми зі зниженням розмірності до двовимірної, що представляється на моніторі).

Зазвичай нейрони розташовуються у вузлах двомірної сітки з прямокутними або шестикутними осередками. При цьому, як було зазначено вище, нейрони також взаємодіють один з одним. Величина цієї взаємодії визначається відстанню між нейронами на карті. На рисунку 1.10 дано приклад відстані для шестикутної та чотирикутної сіток.

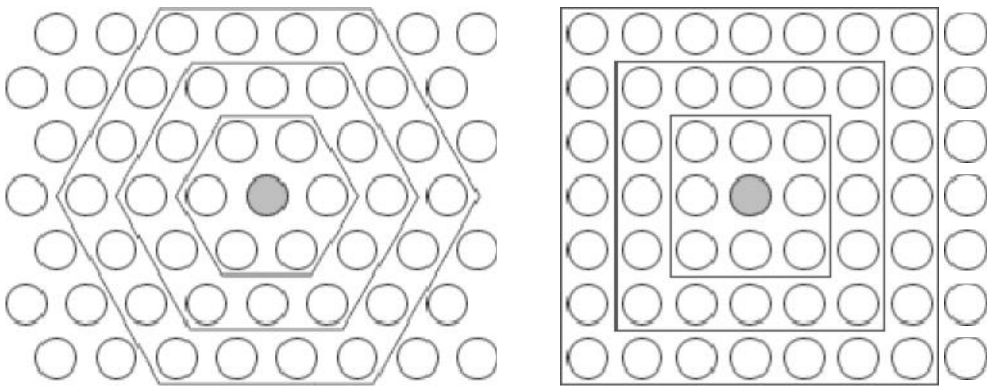

Рисунок 1.9 – Структура мережі Кохонена

Відстань між нейронами на карті для шестикутної (а) та чотирикутної (б) сіток. При цьому легко помітити, що для шестикутної сітки відстань між нейронами більше збігається з евклідовою відстанню, ніж для чотирикутної сітки.

У цьому кількість нейронів у сітці визначає ступінь деталізації результату роботи алгоритму, й у кінцевому підсумку від цього залежить точність узагальнюючої можливості карти.

1.3.2 Навчання карти Кохонена

Мережа Кохонена навчається шляхом послідовних наближень. У процесі навчання таких мереж на входи подаються дані, але мережа у своїй підлаштовується під еталонне значення виходу, а під закономірності у вхідних даних. Починається навчання з обраного випадковим чином вихідного розташування центрів.

У процесі послідовної подачі на вхід мережі навчальних прикладів визначається найбільш схожий нейрон (той, у якого скалярне твір ваги і поданого на вхід вектора мінімально). Цей нейрон оголошується переможцем і є центром підстроювання ваг у сусідніх нейронів. Таке правило навчання передбачає "змагання" навчання з урахуванням відстані нейронів від "нейрона-переможця".

Навчання при цьому полягає не в мінімізації помилки, а в підстроюванні ваг (внутрішніх параметрів нейронної мережі) для найбільшого збігу з вхідними даними.

Основний ітераційний алгоритм Кохонена послідовно проходить ряд епох, кожної з яких обробляється один приклад з навчальної вибірки. Вхідні сигнали послідовно пред’являються мережі, бажані вихідні сигнали не визначаються. Після пред’явлення достатньої кількості вхідних векторів, синаптичні ваги мережі стають здатними визначити кластери. Терези організуються так, що топологічно близькі вузли чутливі до схожих вхідних сигналів.

В результаті роботи алгоритму центр кластера встановлюється у певній позиції, що задовільним чином кластеризує приклади, для яких даний нейрон є "переможцем". Через війну навчання мережі необхідно визначити міру сусідства нейронів, тобто. околиця нейрона-переможця, яка є кілька нейронів, які оточують нейрон-переможець.

Спочатку до околиці належить велика кількість нейронів, далі її розмір поступово зменшується. Мережа формує топологічну структуру, у якій схожі приклади утворюють групи прикладів, що близько перебувають на топологічній карті. Розглянемо це докладніше. Кохонен суттєво спростив вирішення завдання, виділяючи з усіх нейронів шару лише один c-й нейрон, для якого виважена сума вхідних сигналів максимальна:

$$c = \arg\max_i (x^T w_j) \,. \tag{1.12}$$

Зазначимо, що дуже корисною операцією попередньої обробки вхідних векторів є їх нормалізація:

$$\overline{x_i} = \frac{x_i}{\| x \|}, i = \overline{1, N}, \tag{1.13}$$

перетворює вектори вхідних сигналів на одиничні з тим же напрямком.

$$\|x\| = \left(\sum_{i=1}^{N} x_i^2 \right)^{1/2} . \tag{1.14}$$

В цьому випадку внаслідок того, що сума ваг кожного нейрона одного шару $\sum_i w_{ij}$ для всіх нейронів цього шару однакова і умова (1.12) еквівалентна умові:

$$c = \arg\max_i \| x - w_i \| . \tag{1.15}$$

Таким чином, буде активований тільки той нейрон, вектор ваг якого w найбільш близький до вхідного вектора x. Оскільки перед початком навчання невідомо, який саме нейрон буде активуватися при пред’явленні мережі конкретного вхідного вектора, мережа навчається без вчителя, т. е. самонавчається. Вводячи потенційну функцію – функцію відстані f_{ij} («сусідства») між і-м і j-м нейронами з розташуваннями r_i і r_j відповідно монотонно спадаючу зі збільшенням відстані між цими нейронами, Кохонен запропонував наступний алгоритм корекції ваг: де - змінюється у часі коефіцієнт посилення (зазвичай вибирають першої ітерації, поступово зменшуючи у процесі навчання до нуля); - монотонно спадна функція.

Таким образом, будет активирован только тот нейрон, вектор весов которого w наиболее близок к входному вектору x. А так как перед началом обучения неизвестно, какой именно нейрон будет активироваться при предъявлении сети конкретного входного вектора, сеть обучается без учителя, т. е. самообучается. Вводя потенциальную функцию — функцию расстояния f_{ij} («соседства») между і-м и j-м нейронами с местоположениями r_i и r_j соответственно, монотонно убывающую с увеличением расстояния между этими нейронами, Кохонен предложил следующий алгоритм коррекции весов:

$$w_{ij}(k+1) = w_{ij}(k) + \sim(k) f_{ij}(k)(x(k) - w_{ij}(k)) \quad , \qquad (1.16)$$

де $\sim(k) \in (0,1]$ змінюється у часі коефіцієнт посилення (зазвичай вибирають $\sim\ = 1$ першої ітерації, поступово зменшуючи у процесі навчання до нуля); $f_{ij}(k)$ - монотонно спадна функція.

$$f_{ij}(k) = f(\|r_i - r_j\|, k) = f(d, k) = f(d, \pm) , \qquad (1.17)$$

де r_i та r_j - вектори, що визначають положення нейронів і та j у решітці. При прийнятій метриці $d = \|r_i - r_j\|$ функція $f_{ij}(k)$ зі зростанням часу k прямує до нуля. На практиці замість параметра часу k використовують параметр відстані ±, що задає величину області «сусідства» і зменшується з часом до нуля. Вибір функції $f_{ij}(k)$ також впливає на величини ваги всіх нейронів у шарі. Очевидно, що для нейрона-переможця C:

$$f_c(\|r_i - r_j\|) = f_c(0) = 1. \tag{1.18}$$

На рисунку 1.11 показаний приклад зміни двомірних ваг карти $w_j = (w_{j1}, w_{j2})^T$, що утворює ланцюг. З появою вхідного образу найбільш сильно змінюється ваговий вектор нейрона-переможця 5, менш сильно – ваги розташованих поруч із ним нейронів 3, 4, 6, 7. Оскільки нейрони 1, 2, 8, 9 лежать поза області «сусідства», їх вагові коефіцієнти не змінюються.

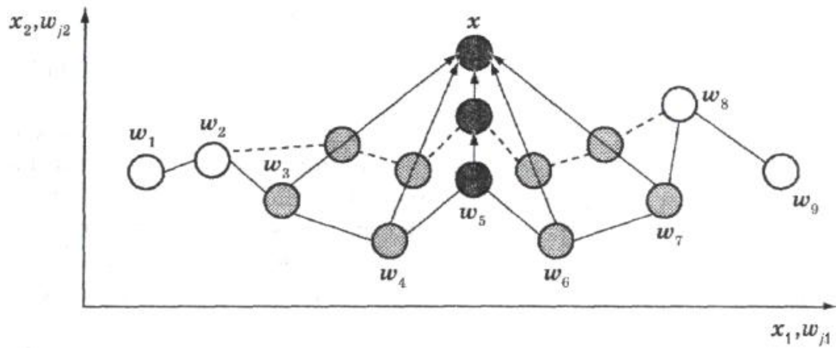

Рисунок 1.11 – Зміна вагів

Таким чином, алгоритм навчання мережі Кохонена може бути описаний так:

ініціалізація. Ваговим коефіцієнтам всіх нейронів надаються малі випадкові значення та здійснюється їх нормалізація. Вибирається відповідна потенційна функція $f_{ij}(d)$ та призначається початкове значення коефіцієнта посилення $\sim_0$.

Вибір навчального сигналу.

З усієї множини векторів навчальних вхідних сигналів відповідно до функції розподілу $P(x)$ вибирається один вектор x, який представляє «сенсорний сигнал», що пред’являється мережі.

Аналіз відгуку (вибір нейрону). За формулою (1.12) визначається активований нейрон.

Процес навчання.

Відповідно до алгоритму (1.16) змінюються вагові коефіцієнти активованого та сусідніх з ним нейронів доти, доки не буде отримано необхідне значення критерію якості навчання або не буде пред’явлено задану кількість навчальних вхідних векторів. Остаточне значення вагових коефіцієнтів збігається із нормалізованими векторами входів. Оскільки мережа Кохонена здійснює проектування N-мірного простору образів на M-мерную мережу, аналіз збіжності алгоритму навчання є досить складне завдання.

Якби з кожним нейроном шару асоціювався один вхідний вектор, то вага будь-якого нейрона шару Кохонена могла б бути навчена за допомогою одного обчислення, так як вага нейрона-переможця коректувалася б з (відповідно до (1.16) для одновимірного випадку вага відразу б потрапляла в центр відрізка [a, b]). Однак зазвичай навчальна множина включає безліч подібних між собою вхідних векторів, і мережа Кохонена повинна бути навчена активувати той самий нейрон для кожного з них. Це досягається усередненням вхідних векторів шляхом зменшення величини, а не за умови пред’явлення кожного наступного вхідного сигналу. Таким чином, ваги, асоційовані з нейроном, усередняться і приймуть значення поблизу "центру" вхідних сигналів, для яких цей нейрон є "переможцем".

2 НЕЙРОМЕРЕЖЕВЕ СТИСНЕННЯ ЗОБРАЖЕНЬ

В даний час ШНМ знаходять широке застосування в різних сферах, і однією з яких і при цьому дуже важливою є стиснення зображень.

2.1 Основні поняття про стиснення зображень

Враховуючи те, що в наші дні обсяг даних із зображеннями збільшується експоненційно у найширшому розмаїтті додатків (наприклад, дистанційне зондування, космічні спостереження, медичні зображення, цифрові бібліотеки та документи, HDTV, розваги та фільми, відеоконференції тощо), а крім того, зростають потреби у запам’ятовуванні, відновленні та передачі зображень, цифрова компресія зображення стає все більш необхідною технологією. Багато з цих прикладних областей - у тому числі «космічні спостереження», відеоконференції та багато програм, взятих на озброєння у військову сферу, використовують технології з послідовностями зображень, що відображають деяку форму руху. Наприклад, послідовності картин, що приймаються супутником кожного разу, коли це проходить приблизно через таке ж розтягування території, після відповідного повторного приміщення та компенсації, - це наступні зразки такого ж місця дії, що містить зміни за рахунок руху об’єктів (наприклад, транспорт) або за рахунок зміни метеорологічних умов. Тому компресія в цих випадках може мати велику перевагу в тому, що послідовності зображення потрібно лише утримувати трек змін, що походять від однієї споруди до наступної.

У деяких областях (як, наприклад, медицина) зазвичай мають справу з сіро-горизонтальними зображеннями, в інших - з кольоровими додатками (приклад, в мультимедійних розвагах). Якість оброблюваного або стисливого зображення оцінюється зовсім по-різному, так різні сфери мають справу з різними кольорами або їх відсутністю. У разі кольору прийнятна якість зображення буде значною мірою залежати від програми.

Стисненням зображення насправді є скорочення обсягу їх цифрового представлення. Під зображенням розуміється прямокутна матриця, елементи якої зазвичай набувають значення (0 … 2^8-1) – для напівтонових, (0… 2^{24}-1) – для кольорових та (0,1) – для бінарних зображень.

Кольорові зображення можуть бути в RGB-форматі (по 8 біт на канал), YCH-форматі (яскрава і дві кольорові складові), а також в інших форматах. Як правило, більшість алгоритмів розробляються для стиснення напівтонових зображень (256 градацій сірого). Це пов’язано з тим, що такі алгоритми легко узагальнити у разі використання кольорового зображення. Існує стиснення зображення з втратами та без втрат.

Стиснення зображення без втрат (lossless compression)- відновлення стисненого зображення повністю ідентичне вихідному («біт у біт»), тобто. при стисканні використовується лише статична надмірність, наявна на зображенні. Така надмірність застосовується у про ентропійних кодерах, які будуть використані нижче. Зрозуміло, що таким чином великого ефекту в стисненні зображення досягти не можна (зазвичай коефіцієнт стиснення становить 1,5-2). Стиснення зображення без втрат знаходить обмежене застосування в деяких областях, таких як астрономія або медицина. Алгоритми стиснення, такі як JPEG, JPEG2000, мають режим роботи без втрат. Іноді розглядається стиск «майже без втрат», тобто. з непомітними людського ока втратами. У цьому досягаються коефіцієнти стиску порядку (6-10).

Для стиснення зображення без втрат знаходить застосування ціле число вейвлет-перетворення. Його перевагою і те, що коефіцієнти перетворення – цілі числа, тому після зворотного перетворення можна відновити зображення «біт у біт». Цілочисленне вейвлет-перетворення виконується для перерозподілу енергії вихідного зображення, тому що в цьому випадку можна побудувати ефективніші ентропійні кодери.

Стиснення з втратами (lossy compression) - відновлення зображення візуально мало відрізняється від вихідного. Ця відмінність називається спотворенням. Кажуть, що стиск із втратами вносить спотворення. Мірою спотворення може бути візуальна відмінність двох зображень. Проте ця характеристика погано описується мовою формул. Тому практично як міри спотворення використовується среднеквадратична помилка. У середовищі дослідників не припиняються суперечки про адекватність цього заходу, проте нічого кращого досі не придумано.

Стиснення із втратами виконується, як правило, у три етапи: лінійне перетворення (DCT, вейвлет), квантування, ентропійне кодування. Цілей першого етапу кілька: перерозподіл енергії між відліками, декореляція відліків та надання коефіцієнтам зручної для квантування структури (наприклад, нульдерева). Квантування буває скалярне (кожен відлік окремо) та векторне. Як ентропійний кодер зазвичай використовується арифметичний (ефективніший, але і більш обчислювально складний) або кодер Хоффмана.

Компресія без втрат є адекватною, коли прийнятні низькі співвідношення компресії. Найістотніші співвідношення компресії можуть бути досягнуті зі збитковими схемами компресії. Ще багато програм будуть використовувати компресію з втратами доти, доки існує результуюча добротність. У деяких критичних додатках - наприклад, медичне зображувальне і військове спостереження - втрата може бути доречна. Однак навіть у цих програмах стислі версії архівних зображень можуть ефективно використовуватися для віддаленої інтеграції та швидкого доступу. Мета компресії зображення - кодувати зображення або послідовність зображення всередину як кілька шматків, з послідовною можливістю використання механізму декодування, який реконструює вихідне зображення з прийнятною візуальною та/або інформаційною якістю. Інша проблема у компресії/декомпресії зображення – це його швидкість, особливо у додатках реального часу або в тих, у яких норма, дані вихідної продукції дуже високі. Таким чином, часто важливою є необхідність здійснювати компресію та декомпресію у реальному часі без додаткової затримки у супроводі зображення.

2.2 Стиснення на основі ШНМ прямого поширення

Один із загальних нейромережевих підходів у компресії зображення – навчити мережу так, щоб кодувати та розшифрувати початкові дані таким чином, щоб результуюча різниця між вхідними та вихідними зображеннями була мінімізована. Мережа складається з вхідного шару та шару виходу рівних розмірів, з проміжним (прихованим) відносно невеликим шаром. Співвідношення розміру вхідного шару розміру проміжного шару є величина, що демонструє співвідношення компресії. Понад те, у випадку може бути кілька проміжних шарів. Мережа зазвичай навчається на одному або більше зображеннях таким чином, що це розробляє відповідність внутрішнього зображення безпосередньо, але швидше до доречних особливостей класу зображень.

У підході, що розглядається нижче, як вхід, проміжна ланка, так і зображення виходу підрозділені на рівно-сортовані за величиною блоки і компресія – це блок, що передається від шару до шару (рисунок 2.1).

Це призводить до скорочення часу навчання мережі. Крім того, досягається хороше узагальнення, так як блоки, що охоплюють єдине тестове зображення, використовуються як навчальний набір. Кількість інформації, що зображує компресію та алгоритм (тобто «ваги») декомпресії, також суттєво зменшуються у цій формі. Нижче використовується випадкова нейронна мережа шифратора/дешифратора управління з прогнозуванням з одним проміжним шаром, як показано на рисунку 2.2.

Ваги між вхідним шаром та проміжним шаром відповідають кодуванню або процесу компресії, у той час як ваги від проміжної ланки до шару виходу відповідають декодуванню або декомпресії процесу.

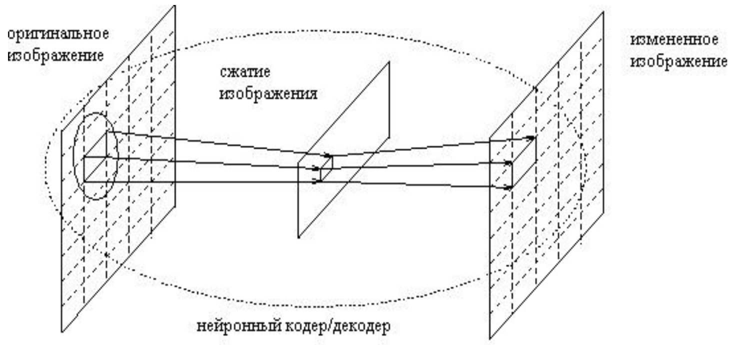

Рисунок 2.1 – Передача блоку

Блоки розміром 8 є результатом, де кожен елемент - це байт. При цьому кодуються 8-розрядні сірі горизонтальні значення як дійсні числа 0 і 1, тобто. будується карта [0,255] інтервалу в інтервалі [0,1] , так як рівень яскравості кожного пікселя зображення перетворений на дійсно оцінене горизонтальне збудження нейрона. Мережа навчається таким чином, щоб мінімізувати квадратичну помилку між цінностями продукції та входу, тому максимізує SNR, за умови, що зображення SNR виміряно для значень квантування в [0,255], у той час як нейронна мережа використовує відповідні дійсно оцінені її параметри.

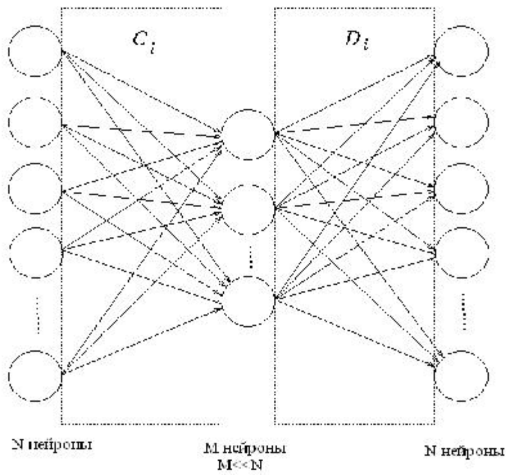

Рисунок 2.2 – Структурна схема нейронної мережі

Приклад якості реконструкції зображення в різних діапазонах від SNR=23dB більш ніж 30dB для 16:1 компресії наведено рисунку 2.3. Він показує результати компресії зі співвідношенням 16:1 для 8-розрядних зображень розмірністю 512512.

2.3 Стиснення на основі мереж зворотного розповсюдження

На відміну від традиційної технології компресії зображення, представленої серіями JPEG, MPEG, та стандартами H.26x, нова технологія, представлена, наприклад, нейронними мережами та генетичними алгоритмами, дозволяє суттєво розширити можливості кодування зображення. Успішні додатки нейронних мереж за векторним квантуванням досить відомі, інші аспекти нейромережевого додатка до вирішення завдань у цій галузі суттєво покращують традиційні методи компресії. На даний момент існує безліч алгоритмів і модифікацій, похідних зі стиснення зображення, наприклад, компресія зображення з використанням зворотного розповсюдження, алгоритм навчання Хебба для стиснення зображення, нейронні мережі векторного квантування, що пророкують нейронні мережі кодування та багато іншого.

а) б)

Рисунок 2.3 – Результат стиснення зображення (а – вихідне, б – стисло (SNR=27,28))

На даний момент існує не так вже й багато топологій штучних нейронних мереж, які використовуються в завданнях стиснення зображення. Одним із варіантів є ШНМ зворотного поширення, яка має кілька варіантів мережевих архітектур, таких як традиційна, ієрархічна, адаптивна. Подібні мережі припускають наявність зв’язків між нейронами різних шарів, так і між нейронами одного шару.

2.3.1 Традиційна ШНМ зворотного розповсюдження

Нейронна структура мережі проілюстрована рисунку 2.4. Розроблено три шари, один вхідний шар, один шар виходу і один прихований шар. Як вхідний шар, так і шар виходу повністю з’єднані з прихованим шаром. Компресія досягається за допомогою проектування значення K - число нейронів у прихованому шарі, менш ніж той із нейронів як у вхідному, так і вихідному шарах. Вхідне зображення розщеплене в блоки або вектори 88, 44 або 1616 пікселів. Коли вхідний вектор звертається як N-мірний, який дорівнює числу пікселів, що входять до кожного блоку, всі зчіпні ваги, з’єднані до кожного нейрона в прихованому шарі, можуть бути зображені $\{W_{ij}$, j = 1, 2, ... K и i = 1, 2, ... N}, який у свою чергу, може також бути описаний матрицею KxN. Від прихованого шару до шару виходу, зв’язку можуть бути зображені $\{W_{ij} : 1 \le i \le N, 1 \le j \le K\}$ який є іншою матрицею ваги N K. Компресія зображення досягається за допомогою навчання мережі таким чином, що зчіпні ваги, $\{W_{ij}\}$, масштабують вхідний вектор N-вимірювання у вузький канал K-вимірювання (K < N) у прихованому шарі і виробляють оптимальну цінність продукції, яка створює квадратичну помилку між входом і мінімумом виходу. Відповідно до нейронної структури мережі, дія може бути описана як зазначено нижче:

$$\text{для кодування } h_i = \sum_{i=1}^{N} w_{ij} x_i \ , 1 \le j \le K, \qquad (2.1)$$

для декодування $\bar{x}_i = \sum_{j=1}^{K} w_{ij}^{/} h_j$, $1 \le i \le N$, (2.2)

де $x_i \in [0, 1]$ означає значення нормалізованого пікселя для зображень шкали яскравості з рівнями яскравості [0, 255]. Причина використання значень нормалізованих пікселів зводиться до того, що нейронні мережі можуть діяти ефективніше, коли їх входи, і виходи обмежені поруч [0, 1].

Згадані вище лінійні мережі можуть також бути перетворені на нелінійні, якщо функція перетворення така, як сигмоїда, додана до прихованого шару і шару виходу, щоб знизити підсумовування в згаданих вище вирівнюваннях. Немає жодного доказу, що нелінійна мережа може забезпечити краще рішення, ніж його лінійний дублікат.

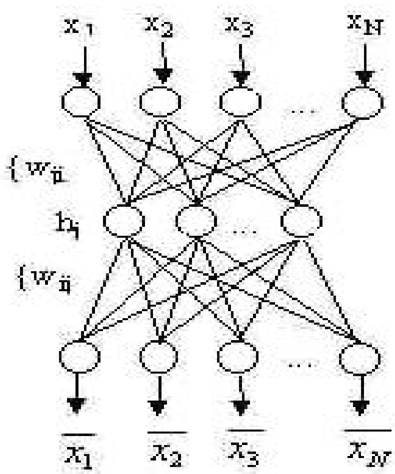

Рисунок 2.4 – Нейронна мережа зворотного розповсюдження

При використанні цієї основної нейронної мережі зворотного розповсюдження компресія проводиться у двох фазах: навчання та кодування. У першій фазі, набір зразків зображення розроблений, щоб навчити мережу через зворотне поширення, що вивчає правило, яке використовує кожен вхідний вектор як бажаний вихід. Це рівноцінно до стиснення входу у вузький канал, представлений прихованим шаром, а потім реконструкція входу від прихованого шару виходу.

Друга фаза просто залучає кодування ентропію вектора в прихованому шарі. У випадку, якщо проводиться адаптивне навчання, кодування ентропії тих зчіпних ваг може також бути потрібним для того, щоб зловити деякі вхідні характеристики, з якими не стикаються у навчальній стадії. Кодування ентропії зазвичай розробляється як просте виправлене двійкове кодування відстані, хоча доступні просунуті кодуючі алгоритми мінливої ентропійної відстані.

Ця нейронна розробка мережі фактично знаходиться в напрямку KL технології перетворення, яка забезпечує оптимальне рішення для всього лінійного каналу вузького виду нейронних мереж при компресії зображення. Коли вирівнювання (1) та (2) представлені у формі матриці, ми маємо

$$[h] = [W]^T [x], \tag{2.3}$$

$$[\bar{x}] = [W'][h] = [W'][W]^T [x] \tag{2.4}$$

для кодування та декодування відповідно.

При такому типі нейронної мережі KL перетворює карту вхідного зображення на новий простір вектора, де всі коефіцієнти співвіднесені. Це означає, що матриця коваріації нових векторів – це діагональна матриця, чиї елементи вздовж діагоналі, – це власні значення матриці коваріації оригінальних вхідних векторів. Припустимо що e_i та λ_i, де i = 1, 2, ... n, є власними (eigen) векторами і власними значеннями , матриця коваріації для вхідного вектора x, і відповідно власні значення розташовані низхідно по порядку, таким чином, що $\lambda_i \geq \lambda_{i+1}$, для i =1, 2, ... n-1. Щоб отримати головні компоненти, власні вектори K, відповідні найбільшим власним значенням K в c_x, зазвичай використовуються, щоб конструювати KL перетворюючу матрицю, $[A_K]$, в якій всі ряди сформовані eigen-векторами c_x. Крім того, всі eigen-вектори в $[A_K]$ замовлені таким чином, що перший ряд $[A_K]$- це eigen-вектор, соответствующий наибольшему eigen-значению, що відповідає найбільшому eigen-значенню, і останній ряд - це eigen-вектор, відповідний найменшому eigen-значенню. Відтепер технологія K-L перетворення або кодування може бути визначена як:

$$[y] = [A_K]([x] - [m_x]), \tag{2.5}$$

та зворотне KL перетворення або декодування може бути визначено як:

$$\left[\bar{x}\right] = [A_K]^T [y] + [m_x], \tag{2.6}$$

де $[m_x]$ – це середнє значення $[x]$ та $[\bar{x}]$, изображающие реконструированные векторы или блоки изображения. що зображають реконструйовані вектори чи блоки зображення. Тому середня квадратична помилка між $[x]$ та $[\bar{x}]$ надана наступним рівнянням:

$$e = E\left\{(x - \bar{x})^2\right\} = \frac{1}{M}\sum_{K-1}^{M}(x_K - \bar{x}_K)^2 = \sum_{j=1}^{k}\lambda_j - \sum_{j=1}^{K}\lambda_j = \sum_{j=K+1}^{k}\lambda_j, \tag{2.7}$$

де статистичне середнє значення E{.} наближається за середнім значенням над усіма зразками вхідного вектора, які, кодування зображення, є всіма частково непокривають блоками 4х4 або 8х8 пікселів.

Таким чином, за допомогою вибору eigen-векторів K, пов’язаних з найбільшими eigen-значеннями, перетворюють пікселі вхідного зображення, результуючі помилки між реконструйованим зображенням та оригіналом можуть бути мінімізовані за рахунок факту, що значення λ монотонно зменшується.

При порівнянні пар вирівнювання (3-4) і (5-6), можна зробити висновок, що лінійна нейронна мережа досягає оптимального рішення щоразу, коли задоволено таку умову:

$$\left[W^{/}\right]\left[W\right]^{T} = \left[A_{K}\right]^{T}\left[A_{K}\right]. \tag{2.8}$$

Під цією обставиною, ваги нейрона від входу до прихованого шару та від прихованого шару до виходу можуть бути описані таким чином:

$$\left[W^{/}\right] = \left[A_{K}\right]\left[U\right]^{-1}; \quad \left[W\right]^{T} = \left[U\right]\left[A_{K}\right]^{T}, \tag{2.9}$$

де [U] – це довільна K×K матриця та $\left[U\right]\left[U\right]^{-1}$ надає матрицю ідентичності K×K. Відтепер, здається, що лінійна нейронна мережа може досягти такого ж результату компресії, так само як ті з K-L перетворення, обов’язково без отримання його матриць ваги, що дорівнюють $\left[A_{K}\right]^{T}$ та $\left[A_{K}\right]$.

2.3.2 Ієрархічна ШНМ зворотного розповсюдження

Основна мережа зворотного розповсюдження може надалі розширюватися, щоб конструювати ієрархічну нейронну мережу за допомогою додавання ще двох прихованих шарів у існуючу мережу, в якій три приховані шари називаються шаром «combiner», шаром компресора і шаром «decombiner». Структура проілюстрована рисунку 2.5. Ідея полягає в тому, щоб експлуатувати кореляцію між пікселями, внутрішнім прихованим шаром і кореляцію між блоками пікселів зовнішніми прихованими шарами. Від вхідного шару до шару комбінування та шару декомбінування до шару виходу, розроблені місцеві зв’язки, що мають той самий ефект, що і повністю з’єднані нейронні підмережі M.

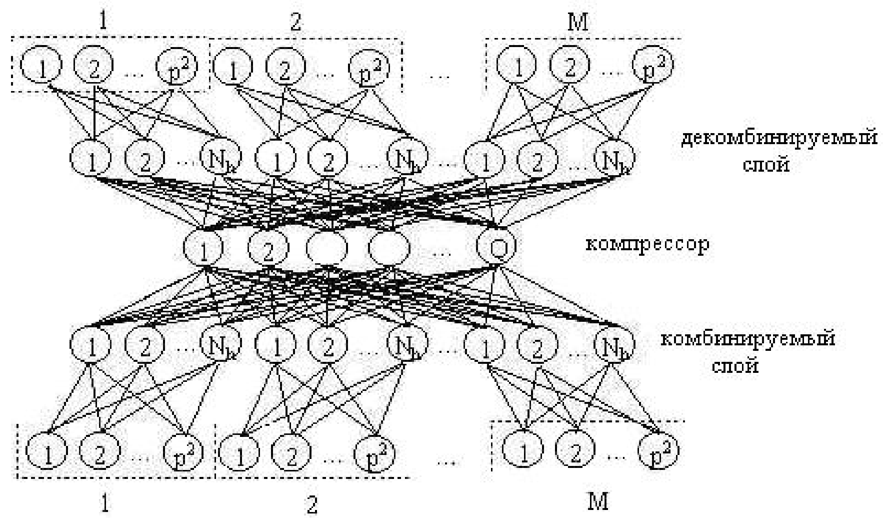

Рисунок 2.5 – Ієрархічна нейронна мережа

Навчання такої нейронної мережі може проводитись у термінах: нейронне навчання мережі (OLNN) зовнішнього циклу; нейронне навчання мережі (ILNN) внутрішнього циклу; та з’єднання розміщення ваги для повної нейронної мережі.

2.3.3 Адаптивна ШНМ зворотного розповсюдження

Адаптивна нейронна мережа зворотного розповсюдження розроблена, щоб зробити нейронну компресію мережі адаптивною («підстроюється») до вмісту вхідного зображення. Загальна структура для типової адаптивної схеми можна переглянути рисунку 2.6 , у якому розроблено групу нейронних мереж зі збільшеним числом прихованих нейронів (hmin, hmax). Основна ідея - класифікувати блоки вхідного зображення в кілька піднаборів з різними особливостями відповідно до їх виміру складності. У такому випадку добре налаштована нейронна мережа стискає кожен підбір.

Навчання такої нейронної мережі може бути розроблене як: (a) паралельне навчання; (b) серійне навчання; та (c) активне базове навчання.

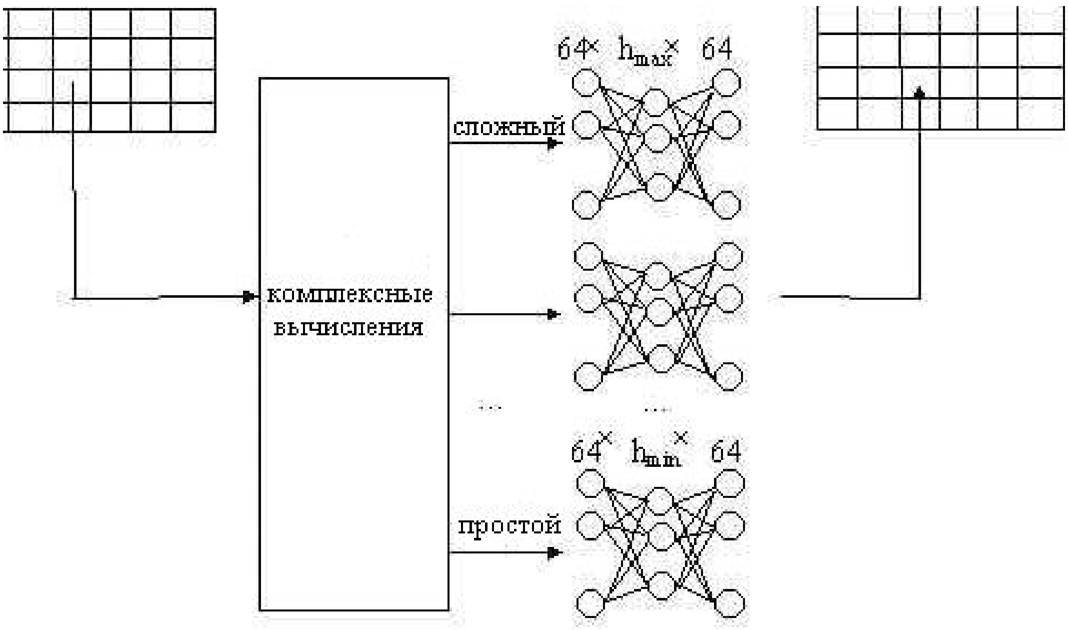

Рисунок 2.6 – Структура адаптивної ШНМ

Паралельна навчальна схема застосовує повне навчання одночасно до всіх нейронних мереж і використовує співвідношення S/N (signal - to-noise), щоб грубо класифікувати блоки зображення в таку кількість піднаборів як та з нейронних мереж. Після того, як ця початкова груба класифікація завершена, кожна нейронна є також мережею, яка навчена його відповідним очищеним піднабором навчальних блоків.

Серійне навчання залучає адаптивний пошуковий процес, необхідний побудови необхідного числа нейронних мереж, з метою пристосувати різні зразки, вбудовані всередині навчальних зображень. Починаючи з нейронної мережі із заздалегідь визначеним мінімальним числом прихованих нейронів, hmin, нейронна мережа "грубо" навчається всіма блоками зображення.

Співвідношення S/N використовується знову, щоб класифікувати всі блоки у два класи залежно від того, чи більший їх S/N, ніж встановлений поріг, чи ні. Для блоків з вищими співвідношеннями S/N, надалі навчання здійснюється до наступної нейронної мережі з кількістю збільшених прихованих нейронів і відповідним порогом, реконструйованим для подальшої класифікації. Цей процес повторюється, поки цілий навчальний набір не класифікований у максимальну кількість піднаборів, що відповідають нумерації встановлених нейронних мереж.

Надалі, діяльність(Pl) та чотири напрямки, визначено, щоб класифікувати навчальний набір. Відтепер зворотне навчання поширення кожної нейронної мережі може бути завершено в одній фазі його підходящим під-набором.

Отже активність викликана 1-м блоком визначена як:

$$A(P_l) = \sum_{i,j} A_p(P_l(i,j)), \tag{2.10}$$

$$A_p(P_l(i,j)) = \sum_{y=-1}^{1} \sum_{s=-1}^{1} (P_l(i,j) - P_l(i+r, j+s))^2, \tag{2.11}$$

де $A_p(P_l(i,j))$ – це дія кожного пікселя, щодо його навколишніх 8 пікселів, оскільки r і s змінюються від -1 до +1.

До навчання, всі блоки зображення класифіковані в чотири класи відповідно до їх значення дії, які ідентифіковані як дуже низько, низько, високо і дуже високо. Відтепер чотири нейронні мережі розроблено зі збільшенням числа прихованих нейронів, щоб стиснути чотири різні піднабори вхідних зображень після завершення навчальної фази.

На довершення високого параметра активності, подальша техніка виділення ознак застосовується з допомогою рахунки чотирьох основних напрямів представленим у деталях зображення, тобто, горизонтальний, вертикальний і два діагональних напрями. Ці пільгові особливості напрямку можуть бути оцінені за допомогою обчислення значень середини квадратної форми відмінностей серед пікселя, що межує, уздовж чотирьох напрямків.

Для цього доданий зразок зображення, класифікований як високоактивний, щоб рафінувати структури та налаштовувати навчальні процеси до пільгових орієнтацій входу для чотирьох нейронних мереж, що відповідають згаданим вище напрямкам. Таким чином, повна нейронна система мережі розроблена, щоб мати шість нейронних мереж, серед яких дві, відповідають низькій діяльності та середнім піднаборам діяльності та інші чотири мережі - високої продуктивності та чотирма класифікаціями напряму.

2.4 Правило навчання Хебба

В той час зворотне поширення засноване нейронна вузька мережу каналу для того , щоб домогтися більш ефективного стиснення K-L перетворення, він розробив серію навчальних правил Hebba необхідна , щоб вирішити цю проблему: так як основні компоненти можуть бути безпосередньо витягнуті з блоків вхідного зображення для досягнення стиснення даних зображення. Загальна структура нейронної мережі складається з одного вхідного шару і одного вихідного шару. Навчання , як правило , Hebba доходи від постулату Hebba, полягає в тому , що якщо два нейрони активні одночасно, про що свідчать високі значення обох виходів , і один з його входів, міцність з’єднання між двома нейронами буде збільшуватися. Відповідно до цим постулатом для значень виробництва, вираженого в = $[w]^{T[x]}$, правило навчання може бути записано таким чином

$$W_i(t+1) = \frac{W(t) + h_i(t)X(t)}{\left\| W_i(t) + h_i(t)X(t) \right\|}, \qquad (2.12)$$

де $W_i(t+1) = \{w_{i1}, w_{i2}, ...w_{iN}\}$ – новий I-й ваговій вектор в наступному циклі (t+1); а $1 \le i \le M$ и M - це кількість вихідних нейронів; α – вивчення норми; $h_i(t)$ – величина г-го продукту ; X(t) – вхідний вектор , відповідний кожен окремий блок зображення; $\|...\|$ – - евклидова норма використовується для нормалізації ваги і поновлення навчального опору.

Виходячи з основного навчального правила, розроблений цілий ряд варіацій, що використовуються в існуючих дослідженнях.

3 ЕВРИСТИЧНИЙ АЛГОРИТМ ТРЕНУВАННЯ ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ

Евристичним називають метод розв‘язання задачі оптимізації, який знаходить приблизне рішення, часто основуючись на випадкових процесах, але робить це шидше ніж відомі точні алгоритми. Часто швидкість досягається за рахунок програшу в оптимальності, повноцінності чи точності.

Основна ціль евристичного алгоритму – знайти розв‘язок задачі оптимізації за прийнятний час. Цей розв‘язок може бути не найкращим, чи він може просто апроксимувати більш точний розв‘язок. Але такий розв‘язок залишається цінним, оскільки на його пошуки витрачається значно менше часу. Результати дослідження NP-складних задач в теоритичних компютерних науках роблять евристичні алгоритми єдиною можливою опцією для розвязання деяких задач оптимізації, які потрібно вирішувати швидко, або навіть в реальному часі. Евристичні алгоритми також знаходять використання у області штучного інтелекту, чи комп‘ютерних симуляціях мислення, а також в галузях, де не існує відомих точних алгоритмів.

Незважаючи на те, що традиційні методи тренування ШНМ (наприклад, метод зворотного поширення помилки чи класичний градієнтний спуск) широко використовуються у багатьох областях, вони мають певні недоліки. Ці недоліки стали основним вузьким місцем, яке обмежує їх подальший розвиток. У більшості випадків метод градієнтного спуску, що використовується в нейронних мережах прямого поширення, має такі основні недоліки:

а) повільне тренування. Для остаточного визначення вагових коефіцієнтів може бути потрібно багато ітерацій в методі градієнтного спуску, особливо якщо функція помилки спадає повільно та має схожість з функцією Розенброка.

б) глобальний мінімум може бути не знайдено, оскільки класичний алгоритм градієнтного спуску легко потрапляє до локального мінімуму.

в) дуже чутливий до початкових значень коефіцієнтів.

Якщо вибір коефіцієнтів некоректний, конвергентна швидкість алгоритму буде дуже малою, і процес тренування займе багато часу. Тому, щоб підвищити ефективність тренування ШНМ, потрібно позбутися наведених проблем, що і стало предметом даного дослідження. Задача тренування ШНМ зазвичай складна та має високу розмірність. Багато дослідників вважають за краще використовувати алгоритми, аналогічні алгоритму зворотного розповсюдження помилки для тренування ШНМ. МЗПП працюють, розраховуючи вихідну похибку та градієнт цієї похибки відносно кооєфіцієнів ШНМ. Після цього відбувається корекція вагових коефіцієнтів і коефіцієнтів зміщення ШНМ в напрямку, протилежному градієнту. Ці градієнтні методи оцінюють помилку в результуючому векторі, що згенерелувала мережа, в порівнянні з учителем і розповсюджують похибку на коефіцієнти по всій мережі. Таким чином, одна з основних перешкод пов’язана з тим, що пошук оптимальних коефіцієнтів сильно залежить від початкових коефіцієнтів. Якщо вони розташовані біля локального мінімуму, алгоритм буде може зійтися до неоптимального розв‘язку. Отже, звичайний метод пошуку градієнта схильний до конвергенції в локальних екстремумах. Запропоновані численні рішення, запропоновані нейромережевими дослідниками для подолання повільної швидкості конвергенції та потрапляння в місцевий мінімум. Існує кілька потужних алгоритмів оптимізації, які базуються на алгоритмі простого градієнтного спуску. Наприклад: кон’югатний градієнтний спуск, масштабований градієнтний спуск, Квазі-Ньютоновські методи. Вони дозволяють підвищити швидкість конвергенції. Оскільки евристичні алгоритми не використовують градієнтну інформацію, це вигідно для проблем, коли така інформація недоступна або якщо її дуже важко отримати. Ці переваги роблять їх більш надійними та привабливими, ніж багато інших алгоритмів пошуку [15]. МРЧ як евристичний алгоритм, простий в реалізації та добре працює над багатьма проблемами оптимізації. Як і інші евристичні методики, МРЧ також може бути використаний у навчанні нейронних мереж.

3.1 Метод рою часток

Метод рою часток, МРЧ (англ. Particle Swarm Optimization, PSO) — метод чисельної оптимізації, для використання якого не потрібно знати точного градієнта оптимізованої функції. МРЧ був доведений Кеннеді, Еберхартом і Ші [2] і спочатку призначався для імітації соціальної поведінки. Алгоритм був спрощений, і було зауважено, що він придатний для виконання оптимізації. Книга Кеннеді й Еберхарта описує багато філософських аспектів МРЧ і так званого ройового інтелекту. Велике дослідження застосувань МРЧ зроблене Полі. МРЧ оптимізує функцію, підтримуючи популяцію можливих розв’язків, називаних частками, і переміщаючи ці частки в просторі розв’язків згідно із простою формулою. Переміщення підпорядковуються принципу найкращого знайденого в цьому просторі положення, що постійно змінюється при знаходженні частками вигідніших положень [4]. Колекти вний інтеле кт (англ. Swarm intelligence) описує комплексну колективну поведінку децентралізованої системи, що самоорганізується. Розглядається в теорії штучного інтелекту як метод оптимізації. Термін був уведений Херардо Бені й Ван Цзином в 1989 році, у контексті системи клітинних роботів. Інколи колективний інтелект ще називають ройовим інтелектом. Із точки зору інформатики, колективний інтелект — це частина комп’ютерних наук, яка проектує та вивчає ефективні числові методи розв‘язання проблем у спосіб, схожий з поведінкою «колективу» живих організмів. Досягнення в цій галузі, а це власне розроблені алгоритми, застосовуються перш за все в задачах комбінаторної оптимізації та для розв’язування задачі комівояжера. Системи колективного інтелекту, як правило, складаються із множини агентів (багатоагентна система), що локально взаємодіють між собою й із навколишнім середовищем. Самі агенти зазвичай досить прості, але всі разом, локально взаємодіючи, створюють так званий колективний інтелект. Прикладом у природі може служити колонія мурах, рій бджіл, зграя птахів, косяк риб. Методи колективного руху агентів у колонії використовуються при проектуванні систем координованої роботи роботів. Розподілена взаємодія між агентами спонукала до створення декількох кластерних алгоритмів та алгоритмів упорядкування. Моделі розподілу праці між агентами колонії були використані для регулювання спільної роботи робототехніки. Дизайнери використовують технології рою як засіб створення складних інтерактивних систем і моделювання натовпу. «Розбиваючи лід» — був перший фільм, що використовував технології колективного інтелекту для візуалізації, реалістичного зображення руху груп риб і птахів із використанням Boids системи. Тім Бертон створив фільм «Бетмен повертається» також з використанням технології колективного інтелекту для демонстрації руху груп кажанів. У фільмі «Володар Кілець» використовували подібну технологію, відому як технологія масивів, під час батальних сцен. Такі технології є особливо привабливими, тому що використання колективного інтелекту — це дешевий, надійний і простий спосіб. Авіакомпанії використовують теорію колективного інтелекту під час моделювання пасажирів перед посадкою в літак. Дослідник Дуглас Лоусон використовував мурах на основі комп’ютерного моделювання та визначив існування лише шести правил взаємодії пасажирів та спромігся оцінити час посадки з використанням різних методів посадки[8]. Колективний інтелект може бути використаний у цілому ряді програм. Збройні сили США використовують методи колективного інтелекту для управління безпілотними транспортними засобами. Європейське космічне агентство думає про «орбітальний рій» для самостійної збірки і інтерферометрії. NASA досліджує використання технології колективного інтелекту для створення планетарних карт. У 1992 році робота M. Anthony Lewis and George A. Bekey довела можливість використання розвідки роєм, за допомогою колективного інтелекту для контролю нанороботів у тілі з метою знищення ракових пухлин. Також колективний інтелект застосовується для інтелектуального аналізу даних. Використання колективного інтелекту в телекомунікаційних мережах також досліджували, у вигляді основ мурашиної маршрутизації. Це було вперше відкрито Dorigo та Hewlett Packard в середині 1990-х років із низкою змін з тих пір. В основному це використання імовірнісних таблиць маршрутизації з використанням «нагородження» — зміцнення успішно пройденого маршруту кожної «мурашки» (невеликий пакет управління), який проходить у мережі. Посилення маршруту вперед, у зворотному напрямку, і обидва одночасно були досліджені. У зворотному напрямку зміцнення вимагає симетричної мережі і пар в обох напрямках разом. Мобільні засоби масової інформації і нові технології можуть змінити поріг для колективних дій у зв’язку з ростом інтелекту систем. Авіакомпанії використовували мурашину маршрутизацію в призначенні воріт для літака, що прибуває в аеропорт. В авіакомпанії Southwest програма використовує колективний інтелект, тобто теорію, що колонія мурашок працює краще, ніж поодинці. Кожен пілот працює як мураха в пошуках найкращих воріт в аеропорт. «Пілот вчиться на власному досвіді, що найкраще для нього, і виявляється, що це найкраще рішення для авіакомпанії» пояснює Дуглас Лоусон. У результаті роботи колонії, кожен пілот завжди прямує до вільних воріт.

3.2 Модифікації методу рою частинок

В оригінальному методі рою частинок, швидкість кожної частинки залежить лише від глобального кращого розв‘язку та історично збереженого кращого розв‘язку конкретної частинки. Таким чином, якщо глобальний карщий розв‘язок виявиться одним з локальних мінімумів, алгориму буває складно вийти з такої ситуації та знайти глобальний мінімум. Щоб вирішити таку проблему в цьому розділі запропоновано три стратегії вдосконалення.

3.2.1 Модифікація МРЧ з вибором кращого сусіднього розв’язку

Припускається, що на швидкість збіжності рою може впливати не тільки краща позіція частинки, та глобальний кращий розв‘язок рою але і краща позиція сусідньої частинки. Імовірно, що в деяких випадках така позиція може позитивніше вплинути на наступну позицію частинки, ніж p_i.

Для визначення підходящих частинок-сусідів можна використати різні підходи. У даній роботі використано перервірку середньої Евклідової відстані між частинками.

Нехай – Евклідова відстань між і-тою та j-тою частинками та md_i – середня Евклідова відстань для і-тої частинки. Тоді:

$$md_i = \frac{\sum_{j=1}^{s} d(i,j)}{s-1} \tag{3.1}$$

Якщо $$d(i,j) < md_i; \tag{3.2}$$

то j-та частинка може бути прийнятою за сусідню до і-тої. Такий підхід можна узагальнити додавши фактор радіусу пошуку сусідніх частинок r. Тоді (3.2) може бути замінено на $d(i,j) \le r * md_i$. Якщо r = 0, то алгоритм працюватиме як звичайний МРЧ, без модифікації. Після вибору сусідньої частинки, у випадку якщо $f(p_{Ni}^{best}) < f(p_i)$, то p_{Ni}^{best} використовується замість p_i у формулі (3.2), тобто

$$v_i(t+1) = \omega\, v_i(t) + \varphi_p r_p \times \left(p_{Ni}^{best} - x_i(t)\right) + \varphi_g r_g \times \left(g - x_i(t)\right) \tag{3.3}$$

інакше використовується формула (3.2). При використанні даної модифікації, на кожній ітерації алгоритму, перед визначенням швидкості частинки відбуається вибір найкращого сусіда, та у випадку, якщо його позиція краще ніж p_i, вона використовується для визначенні нової швидкості (3.3).

3.2.2 Модифікація МРЧ з заміною непривабливих розв’язків

Щоб уникнути передчасної сходимості до локального мінімуму можна запропонувати модифікацію алгоритму, яка заключається у видаленні імовірно поганих розв‘язків, та заміні їх на випадково згенеровані. Нехай L деяке число, що визначає швидкість видалення старих розв‘язків. На початку роботи алгоритму, для кожної частинки ініціалізуємо $l_i(t) = 0$ ($i = 1, \ldots, S$, $t = 0$). На кожній ітерації алгоритму, якщо позиція x_i не може бути змінена в кращу сторону, то модифікуємо:

$$l_i(t+1) = l_i(t) + 1 \tag{3.4}$$

інакше зкинемо значення до нуля: $l_i(t+1) = 0$.

Якщо розв‘язок не може бути покращено разів, тобто якщо $l_i(t) = L$, то позиція x_i буде видалена, та замінена на x_{new}:

$$x_{new} = x_t + sE_t$$

де взято з нормального розподілу з нульовим середнім і єдиним стандартним відхиленням для випадкових блукань, або взяті зі стійкого озподілу для польоту Lévy. Тут розмір кроку s визначає, наскільки далеко ходок може піти за фіксоване число ітерацій. В генерації Леві визначення розміру кроку складне, тому також часто використовують алгоритм Мантени [14]. Якщо s дуже велике, то новий згенерований розв’язок буде занадто далеко від старого (або може навіть не належати до допустимих розв‘язків). Тоді такий крок не буде прийнято. Якщо s занадто мале, зміни занадто малі, щоб бути значними, і, отже, такий пошук не є ефективним. Таким чином, для найбільшої ефективності пошуку розмір кроку має дуже велике значення.

Також розмір кроку значно залежить від області визначення функції та обмежень, накладених на неї для пошуку.

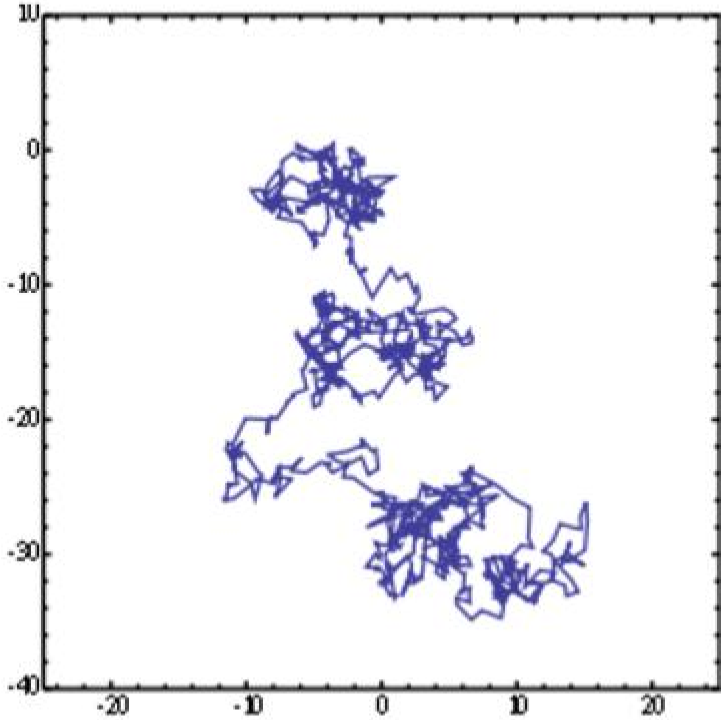

Рисунок 3.1 – Приклад розв’язків, згенерованих розподіленням Lévy у двовимірному просторі

3.2.3 Модифікація МРЧ з додаванням хаотичного оператору пошуку

Щоб покращити глобальну конвергентну швидкість алгоритму ММРЧ, було додано хаотичний оператор пошуку. Для генерації хаотичної складової використано залежність:

$$ch_{i+1} = 4 * ch_i * (1 - ch_i),\ 1 \le i \le k \qquad (3.5)$$

де K –довжина хаотичної послідовності, та ch_0 – випадкова величина.

Таким чином, новий розв‘язок може бути згенеровано за формулою:

$$x_{new} = (1 - \lambda) * g + \lambda * ch, \quad (3.6)$$

де λ – фактор розміру кола пошуку. Розмір кола пошуку будемо лінійно зменшувати, залежно від номеру ітерації алгоритму:

$$\lambda = \frac{maxcycle - t + 1}{maxcycle} \quad (3.7)$$

Таким чином, з плином ітерації алгоритму, хаотична складова буде все менше впливати на новий розв‘язок. Це дозволяє залишити здібність алгоритму знаходити більш точний розв‘язок при наближенні до останньої ітерації, але додасть можливість знайти нові розв‘язки на початкових ітераціях. В свою чергу, така можливість виявляється корисною для запобігання потрапляння алгоритму до локального мінімуму.

Тоді модифікований метод рою частинок (ММРЧ) буде виглядати так:

а) для кожної частки i = 1, …, S зробити:

а.1) згенерувати початкове положення частки за допомогою випадкового вектора $x_i \sim u(b_{lower}, b_{upper})$, що має багатовимірний рівномірний розподіл. b_{lower}, b_{upper} –нижня й верхня границі простору розв‘язків відповідно.

а.2) Присвоїти найкращому відомому положенню частки його початкове значення: $p_i \leftarrow x_i$.

а.3) Якщо $(f(p_i) < f(g))$, то оновити найкращий відомий стан рою: $g \leftarrow p_i$.

а.4) Присвоїти значення швидкості частки:

$$v_i \sim u(-(b_{upper} - b_{lower}), (b_{upper} - b_{lower}))$$

б) Поки не виконаний критерій зупинки (наприклад, досягнення заданого числа ітерацій або необхідного значення цільової функції), повторювати для кожної частки i = 1, …, S:

б.1) Згенерувати коефіцієнт інерції та згенерувати випадкові вектори.

4 ТЕСТУВАННЯ ПРОГРАМНИХ ЗАСОБІВ

Було проведено серію тестових експериментів для порівняння модифікованого методу рою частинок та методу без модифікацій.

Таблиця 4.1 – Список цільових функцій для тестування

Цільова функція	Розмірність	Область визначення	Глобальний мінімум
$f_1 = \sum_{i=1}^{n} x_i^2$	30	[-100; 100]	0
$f_2 = \sum_{i=1}^{n} i x_i^2$	30	[-10; 10]	0
$f_3 = \sum_{i=1}^{h-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	30	[-30; 30]	0
$f_4 = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	30	[-5.12; -5.12]	0
$f_5 = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	[-600; 600]	0
$f_6 = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10; 10]	0
$f_7 = -20\,exp\left(-0.2 * \sqrt{\sum_{i=1}^{n} \frac{z_i^2}{n}}\right)$	30	[-32; 32]	0
$f_8 = \frac{1}{4000}\sum_{i=1}^{n} z_i^2 - \prod_{i=1}^{n} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1$	30	[-600; 600]	0
$f_9 = \sum_{i=1}^{n} z_i^2$	30	[-100; 100]	0
$f_{10} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} z_j\right)^2$	30	[-100; 100]	0

Тестові експерименти були проведені на ноутбуці з процесором Inter® Core(TM) i5-3230M 2.60 GHz. Він має можливість виконувати паралельно тільки чотири потоки.

У якості цільових функцій було використоно десять функцій, наведених в таблиці 4.1. Ці функції відібрано з рекомендованого списку функцій для тестування алгоритмів оптимізації. Деякі з них мають склдну структуру та багато локальних мінімумів. Знаходження мінімуму цих функцій складає певну складність для будь якого алгоритму оптимізації. Наприклад на рисунку 4.2 зображено тривимірний графік однієї з цільових функцій (f_7). На рисунку можна побачити, що функція має дуже багато локальних мінімумів, що значно ускладнює збіжність алгоритму.

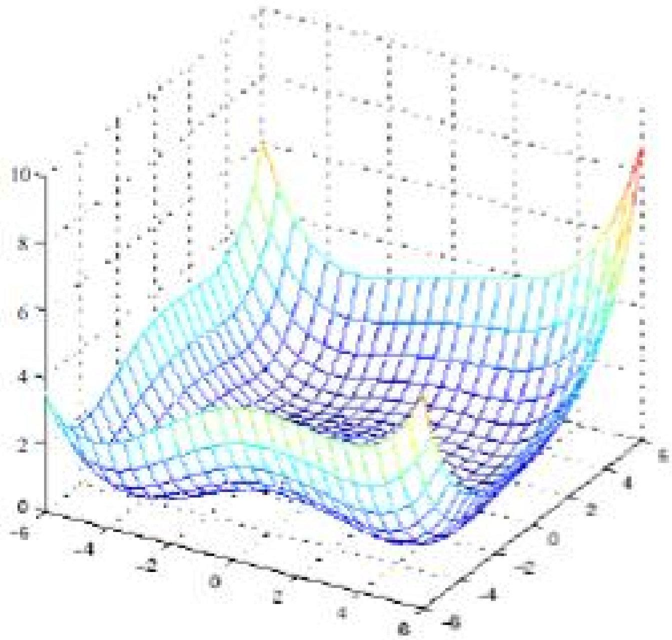

Рисунок 4.1 – Тривимірний графік функції Аклі (Ackley's function)

Спільні парамерти обох алгоритмів:

Розмірність n= 30, кількість частинок S = 30, кількість ітерацій maxcycle = 2000. Параметри, що були підібрані експериментальним шляхом для ММРЧ w_{min} = 0.4, w_{max} = 0.9, L = 5, r = 0.01. Кожен експкримент було повторено 10 разів незележно один від одного.

Таблиця 4.2 – Порівняння результатів роботи МРЧ та ММР

Цільова функція	Алгоритм	Середнє	SD	Мінімальне
f_1	МРЧ	5.07e-03	1.25e-03	4.11e-03
	ММРЧ	5.67e-15	7.34e-16	6.23e-16
f_2	МРЧ	1.81e-01	4.32e-01	8.45e-02
	ММРЧ	7.23e-13	9.28e-14	6.67e-16
f_3	МРЧ	2.21e+01	3.24e-01	1.30e+01
	ММРЧ	2.22e-05	4.93e-05	9.38e-13
f_4	МРЧ	2.90e-01	0	2.90e-01
	ММРЧ	0	0	0
f_5	МРЧ	5.32e-10	3.24e-10	1.30e-10
	ММРЧ	8.32e-15	8.23e-15	9.76e-16
f_6	МРЧ	4.97e-04	4.65e-04	1.91e-04
	ММРЧ	0	0	0
f_7	МРЧ	2.03e+01	1.22e-02	2.02e+01
	ММРЧ	3.29e-07	3.76e-10	1.75e-08
f_8	МРЧ	1.05e+02	9.24e-02	1.04e+02
	ММРЧ	3.68e-09	6.27e-09	6.00-12
f_9	МРЧ	2.32e-04	2.03e-05	2.30e-04
	ММРЧ	6.30e-12	1.04e-11	1.29e-14
f_{10}	МРЧ	7.42e-06	1.03e-06	6.23e-06
	ММРЧ	1.43e-10	2.32e-10	4.65e-12

З проведених експериментів видно, що модифікації алгориму покращують загальну сходимість на різних функціях. Модифікований алгорим знаходить глобальний мінмум функції значно точніше, ніж алгоритм без модифікацій. Різницю в похибці алгоритмів яскраво видно на графіку нижче:

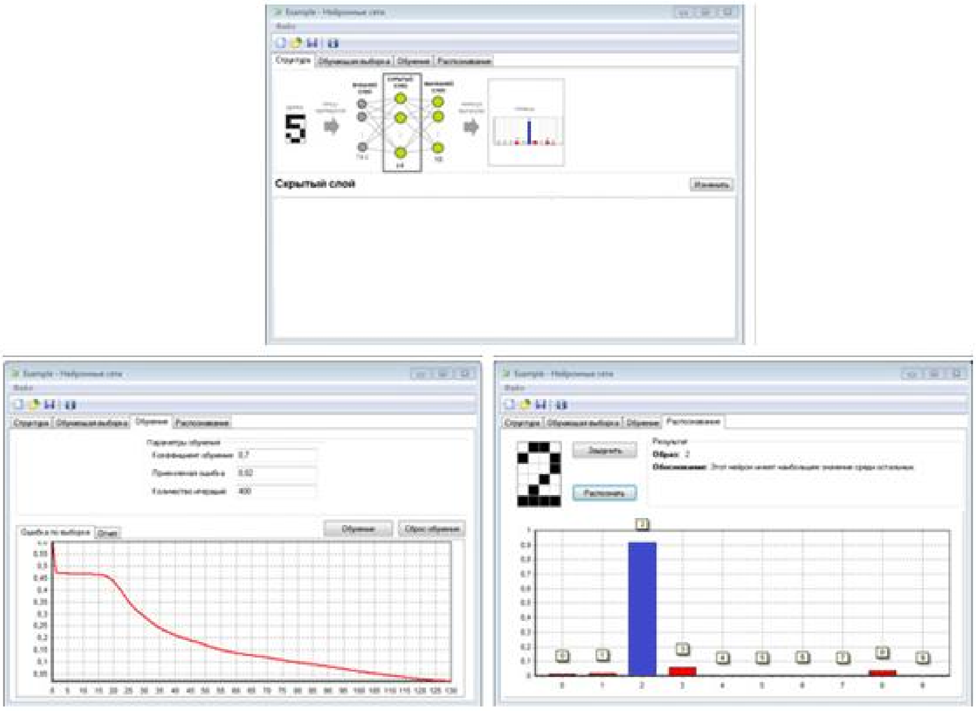

Рисунок 4.3 – Результати роботи

Бачимо, що у випадку f_4 та f_6 алгоритму ММРЧ вдалось знайти мінімум абсолютно точно. Це може бути зв‘язано з відносною простотою функцій, та з меншою областю визначення ніж у інших функцій. В цілому алгоритм ММРЧ показує значно кращі та стабільніші результати, ніж алгоритм без модифікацій, за незмінної кількості ітерацій.

ВИСНОВКИ

Проведено дослідження методів обробки зображень з використанням евристичних алгоритмів штучних нейронних мереж. Розроблені програмні засоби ідентифікації даних на зображенні з використанням нейромережевих технологій. Розроблена тестова архітектура нейронної мережі для розпізнавання рукописних цифр та підібрано дані для тренування мережі. Проведено експериментальні дослідження, які показали, що реалізований алгоритм є конкурентним та може давати прийнятні результати при тренуванні нейронних мереж.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Low, Adrian. Introductory Computer Vision an Image Processing. McGraw Hill. 1991.

2. Форсайт, Девід А., Понс, Жан. Комп’ютерне зір. Сучасний підхід .: Пер. з англ. - М .: Видавничий дім "Вільямс", 2004.: мул, - Парал. Тир. англ.

3. Роберт Хехт-Нільсен. Каліфорнійський університет, Сан-Дієго. Нейрокомпьютинг: історія, стан, перспективи.

4. Kulkami, АШП D. "Computer vision and fuzzy-neural systems." Prentice Hall 2001, ISBN 0-13-570599-1.

5. Р.Гонсалес, Р.Вудс. Цифрова обробка зображень .: Нер. з англ. - М .: Техносфера, 2005. 1072с.

6. Дж. Ту, Р. Гонсалес. Принципи розпізнавання образів. Нер. з англ. - М .: Світ, 1978., мул.

7. Гренандер У. Лекції по теорії образів: Регулярні структури. Нер. з англ. - М .: Світ, 1983. мул.

8. Короткий, Нейронні мережі: алгоритм зворотного поширення. (Www.orc.ru/~stasson/neuroe.html)

9. Медведєв В.С., В.Г. Нотёмкін. Нейронні мережі MATLAB 6 / Нод заг. Ред. к.т.н. В.Г. Нотемкіна. -М .: ДІАЛОГ-МНФН, 2002.-496 с .- (Накети прикладних програм; Кн.4)

10. Anil K. Jain, Jianchang Mao, K.M. Mohiuddin. Artificial Neural Networks: A Tutorial, Computer, Vol.29, No.3, March / 1996.

11. R.P.Lippmann, "An Introduction to Computing with Neural Nets", IEEE ASSP Magazine, Vol.4, No.2, Apr. 1987

12. Ньейн Ей. Штучна нейронна мережа для розпізнавання образів // Наукова сесія МІФІ-2003. Збірник наукових праць. У 14 томах. Т. 13. Конференція «Молодь і наука». Комп’ютерні науки. Інформаційні технології. М .: МІФІ, 2003.

13. Ньейн Ей. Нейронна мережа для розпізнавання зображення // Наукова сесія МІФІ- 2005. Збірник наукових праць. У 15 томах. Т. 14. Конференція «Молодь і наука». Комп'ютерні науки. Інформаційні технології. М .: МІФІ, 2005.

14. Nyein Aye, E. V. Chepin. Car license plate recognition system using artificial neural network Proceedings of the Workshop on Computer Science and Information Technologies (CSIT'2005), Ufa, September 18-21, 2005. Volume 1. Ufa State Aviation Technical University, 2005. ISBN 5-901900- 30-8.

15. В. В. КруглоБ, В. В. Борисов, Штучні нейронні мережі. Теорія та практика. - М: Гаряча лінія - Телеком, 2001..

16. Єжов А.А., Шумський С.А. Нейрокомпьютіг і його додатки в економіці і бізнесі (серія "Підручники економіко-аналітичного інституту МІФІ" під ред. Нроф. В.В. Харитонова). -М .: МІФІ, 1998.-224 с.

17. N. Otsu. "A Threshold Selection Method for Gray Level Histograms". IEEE Transactions on System, Man and Cybernetics. January +1979.

18. V. Turchenko, et al. "Smart Vehicle Screening System Using Artificial Intelligence", IEEE Conference on Homeland Security, May 7-8,2003.

19. Уоссерман Ф ,. Нейкомпьютерная техніка: Теорія і практика; Неревод з англ., М .: Мир, 1992 р

20. Richard P. Lippmann, An Introduction to Computing withNeural Nets, IEEE Acoustics, Speech, and Signal ProcessingMagazine, April, 1987.

21. Підсумки науки і техніки: фізичні та математичні моделі нейронних мереж, том 1, М., изд. ВІНІТІ, 1990.

22. Alain Petrowski, Gerard Dreyfus, Claude Girault, Performance Analysis of a Pipelined Backpropagation Parallel Algorithm // IEEE Transactions on Neural Networks, Vol.4, N6, 1993