

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»**

**ім. Ігоря Сікорського**

Навчально-науковий комплекс «Інститут прикладного системного аналізу»

(повна назва інституту/факультету)

Кафедра Системного проектування

(повна назва кафедри)

«До захисту допущено»

Завідувач кафедри

\_\_\_\_\_ А.І.Петренко

(підпис)

(ініціали, прізвище)

“ \_\_\_\_ ” \_\_\_\_\_ 20\_\_ р.

**Дипломна робота**

**на здобуття ступеня бакалавра**

з напрямку підготовки

6.050101 Комп'ютерні науки

(код і назва)

на тему: Паралельні алгоритми тренування нейронних мереж

Виконав (-ла): студент (-ка) 4 курсу, групи ДА-32

(шифр групи)

Беленок Богдан Іванович

(прізвище, ім'я, по батькові)

(підпис)

Керівник к.т.н, доцент Смаковський Д. С.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Консультант \_\_\_\_\_

(назва розділу)

(посада, вчене звання, науковий ступінь, прізвище, ініціали)

(підпис)

Рецензент \_\_\_\_\_

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

Нормоконтроль старший викладач Бритов О.А.

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент \_\_\_\_\_

(підпис)

Київ – 2017 року

**Національний технічний університет України  
«Київський політехнічний інститут»  
ім. Ігоря Сікорського**

Інститут (факультет) ННК «Інститут прикладного системного аналізу  
(повна назва)

Кафедра Системного проектування  
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрямок підготовки 6.050101 Комп'ютерні науки  
(код і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_ А.І.Петренко  
(підпис) (ініціали, прізвище)

« \_\_\_ » \_\_\_\_\_ 20\_\_ р.

**ЗАВДАННЯ**  
**на дипломну роботу студенту**  
Беленку Богдану Івановичу  
(прізвище, ім'я, по батькові)

1. Тема роботи Паралельні алгоритми тренування нейронних мереж \_\_\_\_\_

керівник роботи Смаковський Денис Сергійович к.т.н, доцент \_\_\_\_\_ ,  
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « \_\_\_ » \_\_\_\_\_ 20\_\_ р. № \_\_\_\_\_

2. Термін подання студентом роботи 10.06.17

3. Вихідні дані до роботи

1. Архітектури та методи навчання нейронних мереж.
2. Методи паралелізації нейронних мереж.
3. Навчальна вибірка на основі образів рукописних цифр.
4. Мова програмування C++.
5. Технологія паралелізації OpenMP
6. Функції активації нейронів в мережі

4. Зміст роботи

1. Проаналізувати існуючі підходи до проектування нейронних мереж

2. Проаналізувати методи паралелізації нейронних мереж
3. Розробити алгоритм роботи та структуру програми
4. Реалізувати програму паралельного тренування нейронної мережі
5. Протестувати програму

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Презентація MS PP
2. Блок схема алгоритму паралельного навчання – плакат
3. Ілюстрації методів паралелізації нейронних мереж - плакат
4. Результати тестування програми - плакат

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний розділ			

7. Дата видачі завдання 01.02.2017

#### Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	01.02.2017	
2	Збір інформації	15.02.2017	
3	Аналіз інструментальних засобів	27.02.2017	
4	Розробка алгоритму вирішення задачі	10.03.2017	
5	Розробка програми	15.03.2017	
6	Тестування програми	25.03.2017	
7	Аналіз отриманих результатів	25.04.2017	
8	Оформлення дипломної роботи	31.05.2017	
9	Отримання допуску до захисту та подача роботи в ДЕК	10.06.2017	

Студент

\_\_\_\_\_

(підпис)

\_\_\_\_\_

(ініціали, прізвище)

Керівник роботи

\_\_\_\_\_

(підпис)

\_\_\_\_\_

(ініціали, прізвище)

## АНОТАЦІЯ

до бакалаврської дипломної роботи Беленка Богдана Івановича  
на тему: «Паралельні алгоритми тренування нейронних мереж»

Дипломна робота присвяча розробці програми для паралельного навчання нейронної мережі. Наведені методи і технології для паралелізації процесу навчання нейронних мереж. Розглянуто реалізації і функції технологій в області паралельних обчислень.

Розроблений алгоритм і архітектура програми, визначені основні переваги і недоліки. Реалізована програма паралельного навчання нейронної мережі з використанням навчальної вибірки рукописних цифр. Проведено тестування для детальної демонстрації роботи програми, а також порівняння ефективності паралельного і лінійного способу навчання нейронної мережі.

Загальний об'єм роботи 77 сторінок, 23 рисунків, 9 таблиць, 18 бібліографічних найменувань.

Ключові слова: Нейронні мережі, алгоритм навчання нейронних мереж сетей, паралельний алгоритм, потоки, OpenMP

## АННОТАЦИЯ

к бакалаврской дипломной работе Беленка Богдана Ивановича  
на тему: «Параллельные алгоритмы обучения нейронных сетей»

Дипломная работа посвящена разработке программы параллельного обучения нейронной сети. Приведен обзор технологий и подходов для параллелизации нейронных сетей.

Исследованы современные методы и средства для параллелизации процесса обучения нейронных сетей. Рассмотрены реализации и функции средств в области параллельных вычислений.

Разработан алгоритм и архитектура программы, определены основные преимущества и недостатки. Реализована программа параллельного обучения нейронных сетей с использованием обучающей выборки состоящей из рукописных цифр. Проведены тестирования для детальной демонстрации работы программы, а также сравнение эффективности параллельного и линейного способа обучения нейронной сети.

Общий объем работы 77 страниц, 23 рисунков, 9 таблиц, 18 библиографических наименований.

Ключевые слова: Нейронные сети, алгоритм обучения сетей, параллельный алгоритм, потоки, OpenMP

## ANNOTATION

of a bachelor`s degree work by Belenok Bohdan Ivanovich  
entitled:« Parallel neural network training algorithms »

This bachelor`s degree work is devoted to developing a program for parallel neural network training. Have been explored methods and technologies for parallelizing of network training. This work reviews the features and functionalities in parallel computing.

The paper develops an algorithm and architecture of the program and provides the main advantages and disadvantages. Implemented a program for parallel neural network training based on handwritten digits data set. Also, the paper conducts testing for a detailed demonstration of the program performance and compared efficiency of parallel and linear algorithms.

The total amount of work: 77 pages, 23 figures, 9 tables, 18 references

Keywords: Neural networks, network training algorithm, parallel algorithm, threads, OpenMP

## ЗМІСТ

ВСТУП .....	9
1 АНАЛІЗ ПІДХОДІВ ДО ПРОЕКТУВАННЯ НЕЙРОННИХ МЕРЕЖ .....	11
1.1 Штучний нейрон .....	13
1.2 Функції активації нейронів .....	15
1.3 Побудова нейронної мережі.....	16
1.4 Навчання штучних нейронних мереж.....	20
1.5 Висновки .....	27
2 МЕТОДИ ПАРАЛЕЛІЗАЦІЇ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ .....	28
2.1 Паралелізація фази навчання .....	29
2.2 Паралелізація навчання вибірки.....	30
2.3 Паралелізація на рівні шарів.....	31
2.4 Паралелізація на рівні нейронів .....	31
2.5 Паралелізація на рівні вагів .....	32
2.6 Ефективність процесу паралелізації .....	33
2.7 Технології паралелізації .....	35
2.7.1 OpenMP .....	36
2.7.2 MPI .....	37
2.8 Висновки .....	40
3 РОЗРОБКА ПРОГРАМИ ПАРАЛЕЛЬНОГО НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ .....	41
3.1 Засоби та платформа реалізації .....	41
3.2 Розробка алгоритму роботи програми .....	42
3.3 Архітектура програми .....	45
3.4 Функціональні аспекти системи.....	46
3.5 Висновки .....	50
4 ТЕСТУВАННЯ ПРОГРАМИ .....	51
4.1 Тест № 1 .....	51
4.2 Тест № 2 .....	52
4.3 Тест № 3 .....	53

4.4 Тест № 4 .....	54
4.5 Висновки .....	55
<b>5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ.</b>	<b>56</b>
5.1 Постановка задачі техніко-економічного аналіз .....	57
5.1.1 Обґрунтування функцій програмного продукту .....	58
5.1.2 Варіанти реалізації основних функцій .....	58
5.2 Обґрунтування системи параметрів ПП .....	60
5.2.1 Опис параметрів.....	60
5.2.2 Кількісна оцінка параметрів.....	60
5.2.3 Аналіз експертного оцінювання параметрів.....	63
5.3 Аналіз рівня якості варіантів реалізації функцій.....	67
5.4 Економічний аналіз варіантів розробки ПП.....	68
5.5 Вибір кращого варіанта ПП техніко-економічного рівня.....	72
5.6 Висновки .....	72
<b>ВИСНОВКИ.....</b>	<b>74</b>
<b>ПЕРЕЛІК ПОСИЛАНЬ.....</b>	<b>76</b>



## ВСТУП

Під час розв'язування практичних задач науки і техніки зазвичай виникає проблема ефективної організації обчислень. Деякі з цих задач характеризуються великою розмірністю вхідних даних і тому потребують оброблення значних обсягів інформації. З іншого, багато задач, пов'язаних, з аналізом ринків, моніторингом роботи великих промислових підприємств, банківських установ, характеризуються високою частотою надходження інформації про стан досліджуваного об'єкта і потребують прийняття оптимальних рішень в режимі реального часу. Зростання розмірності вхідних даних та багатократне використання одного і того ж фрагменту обчислень призводять до збільшення складності розв'язання задач. Тому необхідно удосконалювати вже існуючі і розробляти нові підходи до організації та використання обчислень на обчислювальних системах високої продуктивності.[1]

Одним із способів оптимізації обчислювального процесу є його паралелізація з метою подальшої реалізації на системах паралельної архітектури. Для реалізації паралельних методів та алгоритмів використовується обчислювальні засоби універсально та спеціально призначення. На даний час розвиток універсальних обчислювальних систем здійснюється за чотирма основними напрямками вектороконверні, SMP(Symmetric Multi-Processing), MPP(massively parallel Processing) та кластери. Типовим прикладом SMP-систем є сучасні багатоядерні процесори від компанії Intel(i3,i5, тощо).

Ще в 40-х роках минулого століття досягнення нейробіології дозволили створити першу штучну нейронну мережу, яка імітувала роботу людського мозку. Але тільки через декілька десятиліть, разом з виникненням сучасних комп'ютерів і відповідного програмного забезпечення стала можлива розробка складних додатків в області ШНМ. З цього моменту теорія нейронних мереж стала одним із найбільш перспективних напрямів наукових досліджень. Цьому сприяла сама природа паралельних обчислень і можливість адаптивного навчання нейронних мереж.[2]

Однак, незважаючи на розвиток та впровадження паралельних обчислень у науковій діяльності, використання паралельних методів навчання штучних нейронних мереж (ШНМ) є відносно новою та мало дослідженою задачею.

Мета роботи – реалізувати програму паралельного навчання нейронної мережі. Для цього необхідно розглянути та проаналізувати існуючі засоби та рішення в області паралелізації програмного забезпечення, та підходи до паралелізації нейронних мереж. Розробити власний алгоритм та архітектуру програми.

# 1 АНАЛІЗ ПІДХОДІВ ДО ПРОЕКТУВАННЯ НЕЙРОННИХ МЕРЕЖ

Сьогодні є безперечним значний науковий та практичний інтерес до обчислювальних структур нового типу — штучних нейронних мереж. Він спричинений низкою успішних застосувань цієї нової технології, яка дозволила розробити ефективні підходи до вирішення проблем, що вважалися складними для реалізації на традиційних комп'ютерах. На назву “нейронні мережі” зараз претендують усі обчислювальні структури, які в тій чи іншій мірі моделюють роботу мозку. Але таке моделювання, здебільшого, є дуже фрагментарним, і говорити про створення у найближчому майбутньому штучного мозку або навіть деякої його моделі, яка дублювала б роботу мозку найпримітивніших живих створінь, ще зарано. Такий висновок випливає з надзвичайної складності цього загадкового витвору природи.

При побудові моделі мозку розглядають локальні та глобальні аспекти пізнання його функціонування. Основною глобальною характеристикою, яка істотно утруднює моделювання, є надзвичайно велика кількість базових структурних елементів. Мозок людини містить близько сотні мільярдів нейронів, кожен з яких кількома тисячами зв'язків об'єднується з іншими нейронами. Використання навіть найпростішої моделі нейрона не дозволяє побудувати модель мозку, що наближалася б за своїми глобальними показниками до реального об'єкта моделювання. До локальних характеристик слід віднести власне принципи, за якими будують модель нейрона. Останнім часом нейробіологія досягла значних успіхів у вивченні нейрона як елементарної структурної одиниці мозку. Відкрито велику кількість закономірностей, що описують його функціонування та взаємодію з іншими нейронами. Однак, як і раніше, залишаються без відповіді питання про те, яким чином реалізуються такі властиві мозку основні функції, як пам'ять та свідомість. Отже, залишається до кінця не з'ясованим зв'язок між локальними характеристиками нейрона та

глобальними функціями мозку. Але саме такий зв'язок і є основою побудови штучних нейронних мереж, які моделюють функції мозку. Тому основною проблемою концептуального підходу до нейромережного моделювання є вертикальна стратифікація моделі, тобто з'ясування питання про взаємодію елементів на всіх рівнях знизу вгору. Лише шляхом вдалої координації дій великої кількості структурних елементів можливо досягти вияву якісно нової властивості всієї моделі.

Успішний розвиток теорії нейронних мереж за останнє десятиліття дозволив реалізувати ряд таких глобальних властивостей. Найвідомішими з них є навчання, узагальнення та абстрагування.

Властивість навчання проявляється у здатності нейронної мережі змінювати свою поведінку в залежності від стану навколишнього середовища. Завдяки саме цій властивості нейронні мережі привертають до себе значну увагу. Існує велике розмаїття алгоритмів навчання нейронних мереж, кожен з яких має свої сильні та слабкі сторони, але сьогодні ще не сформовано єдиної думки про те, чому можна навчити нейронну мережу і як таке навчання має проводитись.

Властивість узагальнення дає можливість нейронній мережі знижувати чутливість до незначних флуктуацій вхідних сигналів. Ця властивість дуже важлива для об'єктів, які існують у реальному середовищі. Особливістю нейромережного підходу в даному випадку є те, що узагальнення — це результат властивостей структури, а не роботи спеціальної програми на традиційному комп'ютері.

Властивість абстрагування дозволяє створювати на нейронній мережі нову сутність, виходячи з аналізу вхідної інформації. Особливо ця властивість проявляється для задач розпізнавання образів. Завдяки їй нейромережа може створювати деякий ідеальний образ, керуючись вхідною інформацією, яка має деякі властивості цього образу.

Дослідження штучних нейронних мереж пов'язано з тим, що вони дозволяють наблизитися до можливостей обробки інформації людським мозком, який являє собою надзвичайно складний, нелінійний, паралельний комп'ютер

(систему обробки інформації). Мозок має здатність організувати свої структурні компоненти, звані нейронами, так, щоб вони могли виконувати конкретні задачі (такі як розпізнавання образів, обробку сигналів органів почуттів, моторні функції) в багато разів швидше, ніж можуть дозволити найшвидкодійні сучасні комп'ютери.

На сьогоднішній день існує багато прикладів використання штучних нейронних мереж для прогнозів, класифікації, оптимізації, розпізнавання образів та багато інших [3].

Нейронні мережі – обчислювальні структури, які моделюють прості біологічні процеси, що асоціюються з процесами людського мозку. Вони представляють собою системи, здатні до навчання шляхом аналізу позитивних і негативних впливів. Елементарним перетворювачем в даних мережах є штучний нейрон або просто нейрон, названий так за аналогією з біологічним прототипом.

## 1.1 Штучний нейрон

Прототипом для створення нейрона став біологічний нейрон головного мозку. Біологічний нейрон має тіло, сукупність відростків – дендритів, за якими в нейрон надходять вхідні сигнали, і аксонів, що передають вихідні сигнали нейронів іншим клітинам. Точка з'єднання дендрита і аксона називається синапсом [4]. Спрощено функціонування нейрона можна представити наступним чином:

Нейрон отримує від дендритів набір (вектор) вхідних сигналів;

У тілі нейрона оцінюється сумарне значення вхідних сигналів. Однак входи нейрона нерівнозначні. Кожен вхід характеризується деяким ваговим коефіцієнтом, що визначає важливість інформації переданої ним. Таким чином, нейрон не просто підсумовує значення вхідних сигналів, а обчислює скалярний добуток вектора вхідних сигналів і вектора вагових коефіцієнтів;

Нейрон формує вихідний сигнал, інтенсивність якого залежить від значення обчисленого скалярного перемноження. Якщо воно не перевищує

деякого заданого порогу, то вихідний сигнал не формується зовсім – нейрон «не спрацьовує»;

Вихідний сигнал надходить на аксон і передається дендриту інших нейронів.

Нейрон – це складова частина нейронної мережі. Структуру штучного нейрона зображено на рис. 1

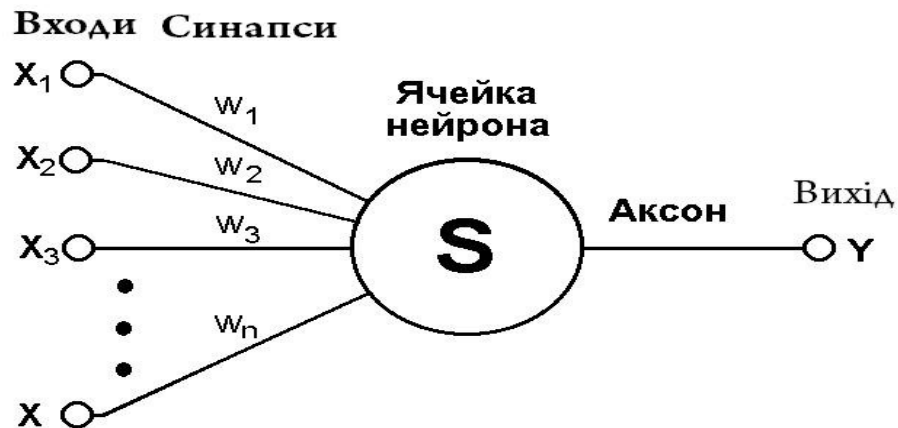


Рисунок 1 – Структура штучного нейрона [16]

На рис. 1 використано позначки:

$W$  – вектор вагових коефіцієнтів;

$S$  – зважена сума.

$X$  – вектор вхідних сигналів;

Поточний стан нейрона визначається, як зважена сума його входів, як в виразі (1):

$$s = \sum_{i=1}^n x_i w_i, \quad (1)$$

де  $n$  – число входів нейрона;

$x_i$  – значення  $i$ -го входу нейрона;

$w_i$  – вага  $i$ -го синапсу

Виходом нейрона є функція його стану (2):

$$Y = f(s), \quad (2)$$

де  $f$  – деяка функція активації.

У загальному випадку вхідний сигнал та вагові коефіцієнт можуть приймати дійсні значення. Вихід  $Y$  визначається видом функції активації і може бути як дійсним, так і цілим. У багатьох практичних задачах входи та їх ваги можуть приймати лише деякі фіксовані значення.

Синоптичні зв'язки з позитивними вагами називають збудливими, а з від'ємними вагами – гальмуючими.

Таким чином, нейрон повністю описується своїми вагами  $W_i$  і передатною функцією  $F(x)$ . Одержавши набір чисел (вектор)  $X_i$  в якості входів, нейрон видає деяке число  $Y$  на виході.

## 1.2 Функції активації нейронів

В основу роботи штучних нейронних мереж [5, 6] покладено такі риси справжніх нейронних мереж, що дозволяють їм добре справлятися з нерегулярними завданнями. Перелік основних функцій активації нейронів приведено в табл. 1

Таблиця 1 – Функції активації нейронів

Назва	Формула	Область значень
Знакова	$f(s) = \begin{cases} 1, & s > 0, \\ -1, & s \leq 0 \end{cases}$	-1,1
Сигмоїдна	$f(s) = \frac{1}{1 + e^{-s}}$	(0,1)
Лінійна	$f(s) = s$	$(-\infty; \infty)$
Радіально-базисна	$f(s) = \exp(-s^2)$	(0,1)

Поведінка штучної нейронної мережі залежить як від значення вагових параметрів, так і від функції збудження нейронів. Відомі три основних види [25] функції збудження: порогова, лінійна і сигмоїдальна. Для порогових елементів вихід встановлюється на одному з двох рівнів залежно від того, більше або

менше сумарний сигнал на вході нейрона деякого порогового значення. Для лінійних елементів вихідна активність пропорційна сумарному зваженому входу нейрона. Для сигмоїдальних елементів залежно від вхідного сигналу, вихід варіюється безперервно, але не лінійно, по мірі зміни входу. Сигмоїдальні елементи мають більше схожості з реальними нейронами, ніж лінійні, але будь-який з цих типів можна розглядати лише як наближення.

Нейронна мережа являє собою сукупність великої кількості порівняно простих елементів – нейронів, топологія з'єднань яких залежить від типу мережі. Щоб створити нейронну мережу для вирішення якої-небудь конкретної задачі, ми повинні вибрати, яким чином слід з'єднувати нейрони, і відповідним чином підібрати значення вагових параметрів на цих зв'язках. Чи може впливати один елемент на інший, залежить від встановлених з'єднань. Вага з'єднання визначає силу впливу.

### 1.3 Побудова нейронної мережі

Не менш важливим є завдання побудова мережі. Це питання вирішується в два етапи:

- вибір типу (архітектури) мережі;
- підбір ваг (навчання) мережі.

На першому етапі слід вирішити наступні питання:

- які нейрони ми хочемо використовувати (число входів, передавальні функції);
- яким чином слід з'єднати їх між собою;
- що взяти в якості входів і виходів мережі.

Це завдання на перший погляд здається складним, але необов'язково придумувати нейромережу – існує кілька десятків різних нейромережових архітектур, причому ефективність багатьох з них доведена математичною статистикою. Найбільш популярні і вивчені архітектури – це багатошаровий



персептрон, нейромережа із загальною регресією, мережі Кохонена, мережі Хопфілда, мережі Хеммінга та інші [7].

На другому етапі слід навчити обрану мережу, тобто підібрати такі значення її ваг, щоб мережа працювала належним чином. У використовуваних на практиці нейромережах кількість ваг може становити кілька десятків тисяч, тому навчання це дійсно складний процес. Для багатьох архітектур розроблені спеціальні алгоритми навчання, які дозволяють налаштувати ваги мережі певним чином.

У залежності від функцій, виконуваних нейронами в мережі, можна виділити три їх типи:

- вхідні нейрони – це нейрони, на які подається вхідний вектор, що кодує вхідний вплив чи образ зовнішнього середовища; в них зазвичай не здійснюється обчислювальні процедури, інформація передається з входу на вихід нейрона шляхом трансформаційних змін його активації;

- вихідні нейрони – це нейрони, вихідні значення яких представляють вихід мережі;

- проміжні нейрони – ці нейрони складають основу штучних нейронних мереж.

У більшості нейронних моделей тип нейрона пов'язаний з його розташуванням у мережі. Якщо нейрон має тільки вихідні зв'язки, то це вхідний нейрон, якщо навпаки – вихідний нейрон. Однак може зустрітися випадок, коли вихід внутрішнього нейрона розглядається як частина виходу мережі. У процесі функціонування мережі здійснюється перетворення вхідного вектора у вихідний.

Сукупність штучних нейронів, суматора та порогового елемента називається штучною нейронною мережею (рис 2).

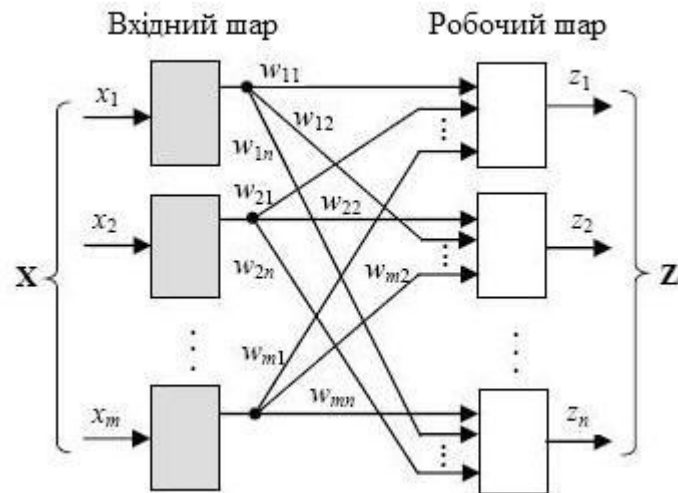


Рисунок 2 – Модель одношарової штучної нейронної мережі [17]

Сигнал надходить на вхідний шар рецепторних нейронів, кожен з яких пов'язаний з усіма елементами вихідного шару з певними значеннями вагових коефіцієнтів  $w_{ij}$ . Їх підсумовування призводить до порушення тих нейронів робочого шару, значення активаційної функції яких перевищила порогове значення.

Кілька основних нейромережевих архітектур, такі, як багатошарові перцептрони, мережі Хопфілда і карти Кохонена, роблять можливим вирішення широкого спектру завдань, найчастіше нерозв'язних класичними статистичними методами обробки даних.

Серед переваг нейронних мереж можна виділити такі:

- навчання на наборі прикладів;
- побудова нелінійної регресійної залежності або нелінійної роздільної поверхні без апіорного завдання виду нелінійної функції з точністю до значень параметрів;
- можливість рішення одночасно кількох завдань прогнозування чи класифікації однією нейронною моделлю з векторним виходом;
- цільова функція, оптимізується при навчанні нейронної мережі, не обмежена звичайним методом найменших квадратів (МНК) і може бути

робастною до викидів в даних, а також може включати в себе додаткові складові, наприклад, регулюючі рішення;

- побудова нелінійних головних компонент нейронною мережею з «вузьким горлом» [8;9];
- при недостатності лінійних головних компонент для опису даних з потрібною точністю для їх подальшої візуалізації в просторі.

В літературі [10] зустрічається значна кількість ознак, які повинна мати задача, щоб можна було ефективно застосувати ШНМ, наприклад: відсутність алгоритму або не відомі принципи вирішення завдань, але накопичено достатню кількість прикладів; проблема характеризується великими обсягами вхідної інформації; дані неповні, або надлишкові чи частково суперечливі.

ШНМ може розглядатися як спрямований граф зі зваженими зв'язками, в якому штучні нейрони є вузлами. За архітектурою зв'язків ШНМ можуть бути згруповані в два класи, в яких графи не мають петель, і рекурентні мережі, або мережі зі зворотними зв'язками.

Систематизація архітектур мереж прямого поширення та рекурентних (зі зворотним зв'язком).

У найбільш поширеному типі мереж першого класу, наприклад, багат шаровому персептроні, нейрони розташовані шарами і мають односпрямовані зв'язку між шарами.

Нейронні мережі можна розподілити на [11]:

а) мережі прямого розповсюдження:

- 1) одношаровий персептрон;
- 2) багат шаровий персептрон;
- 3) мережа радіальних базисних функцій.

б) рекурентні мережі (зі зворотним зв'язком) :

- 1) змагальні мережі;
- 2) мережа Хопфілда;
- 3) мережа Кохонена;
- 4) моделі ART (Adaptive Resonance Theory Network).

Мережі прямого поширення є статичними в тому сенсі, що на вхід вони поставляють одну сукупність вихідних значень, що не залежать від попереднього стану мережі. Рекурентні мережі є динамічними, тому що в силу зворотних зв'язків у них модифікуються входи нейронів, що призводить до зміни стану мережі.

## **1.4 Навчання штучних нейронних мереж.**

У ШНМ процес навчання може розглядатися як налаштування архітектури мережі і ваг зв'язків для ефективного виконання різних типів задач.

Нейронна мережа повинна налаштувати ваги зв'язків по наявній навчальній вибірці. Функціонування мережі поліпшується у міру ітеративного налаштування вагових коефіцієнтів. Властивість мережі навчатися на прикладах робить їх більш привабливими в порівнянні з системами, які слідують певній системі правил функціонування, сформульованих експертами.

Для конструювання процесу навчання, перш за все, необхідно мати модель зовнішнього середовища, в якій функціонує нейронна мережа – знати доступну для мережі інформацію. Також необхідно визначити, як модифікувати вагові параметри мережі – які правила навчання управляють процесом налаштування. Алгоритм навчання означає процедуру, в якій використовуються правила навчання для налаштування ваг.

Найважливішою властивістю нейронних мереж є їх здатність навчатися на основі даних навколишнього середовища і в результаті навчання підвищувати свою продуктивність. Підвищення продуктивності відбувається з часом у відповідності з певними правилами. Навчання нейронної мережі відбувається за допомогою інтерактивного процесу корегування синаптичних ваг і порогів. В ідеальному випадку нейронна мережа отримує знання про навколишнє середовище на кожній ітерації процесу навчання.

З поняттям навчання асоціюється досить багато видів діяльності, тому складно дати цьому процесу однозначне визначення. Більше того, процес

навчання залежить від точки зору на нього. Саме це робить практично неможливим появу будь-якого точного визначення цього поняття. Наприклад, процес навчання з точки зору психолога в корені відрізняється від навчання з точки зору шкільного вчителя. З позицій нейронної мережі, ймовірно, можна використовувати наступне визначення:

Навчання – це процес, у якому вільні параметри нейронної мережі настраюються за допомогою моделювання середовища, у яке ця мережа вбудована.

Тип навчання визначається способом підстроювання цих параметрів.

Це визначення процесу навчання нейронної мережі передбачає наступну послідовність подій:

- 1) у нейронну мережу надходять стимули із зовнішнього середовища;
- 2) у результаті реалізації першого пункту змінюються вільні параметри нейронної мережі;
- 3) після зміни внутрішньої структури нейронна мережа відповідає на штрафи вже іншим чином.

Вищевказаний список чітких правил вирішення проблеми навчання нейронної мережі називається алгоритмом навчання. Не існує універсального алгоритму навчання, відповідного для всіх архітектур нейронних мереж. Існує лише набір засобів, представлений безліччю алгоритмів навчання, кожен з яких має свої переваги. Алгоритми навчання відрізняються один від одного способом налаштування синаптичних ваг нейронів. Ще однією відмінною характеристикою є спосіб зв'язку навченою нейронної мережі із зовнішнім світом. У цьому контексті говорять про парадигму навчання, пов'язану з моделлю навколишнього середовища, в якій функціонує дана нейронна мережа.

Існують три способу навчання: з вчителем; без вчителя; змішана. Навчання нейронної мережі з учителем припускає, що для кожного вхідного вектора з навчальної множини існує необхідне значення вихідного вектора, званого цільовим. Ці вектора утворюють навчальну пару. Ваги мережі змінюють до тих пір, поки для кожного вхідного вектора не буде отриманий прийнятний рівень

відхилення вихідного вектора від цільового. Нейронна мережа має у своєму розпорядженні правильними відповідями (виходами мережі) на кожен вхідний приклад. Ваги налаштовуються так, щоб мережа виробляла відповіді, як можна більш близькі до відомих правильних відповідей. Посилений варіант навчання з учителем припускає, що відома тільки критична оцінка правильності виходу нейронної мережі, але не самі правильні значення виходу.

Навчання нейронної мережі без вчителя є набагато більш правдоподібною моделлю навчання з точки зору біологічних коренів штучних нейронних мереж. Навчальна множина складається лише з вхідних векторів. Алгоритм навчання нейронної мережі підлаштовує ваги мережі так, щоб виходили узгоджені вихідні вектори, тобто щоб пред'явлення досить близьких вхідних векторів давало однакові виходи. Навчання без вчителя не вимагає знання правильних відповідей на кожний приклад навчальної вибірки. У цьому випадку розкривається внутрішня структура даних, або кореляції між зразками в системі даних, що дозволяє розподілити зразки за категоріями. При змішаному навчанні частина ваг визначається за допомогою навчання з учителем, у той час як інша виходить за допомогою самонавчання.

Різні алгоритми навчання та пов'язані з ними архітектури мереж представлені у табл. 2. В останній колонці перераховані задачі, для яких можуть бути застосовані різні алгоритми. Кожен з алгоритмів навчання орієнтований на мережу певної архітектури і призначений для обмеженого класу задач.

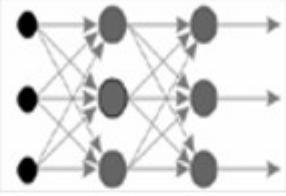

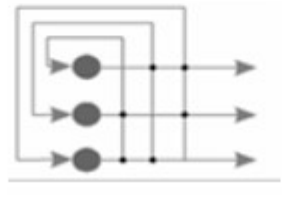
Таблиця 2 — Відомі алгоритми навчання нейронних мереж

Типи навчання	Правило навчання	Архітектура	Алгоритм навчання	Задача	
З вчителем	Корекція помилки	Одношаровий і багатшаровий перцептрон	Алгоритми навчання перцептрона Зворотне поширення	Класифікація образів Аппроксимація функцій Передбачення, управління	
	Больцман	Рекурентна	Алгоритм навчання Больцмана	Класифікація образів	
	Хебб	Багатшарова прямого розповсюдження	Лінійний дискримінантний аналіз	Аналіз даних Класифікація образів	
	Змагальні		Змагальня	Векторне квантування	Категоризація всередині класу Стиснення даних
			Мережа ART	ARTMap	Класифікація образів
Без вчителя	Корекція помилки	Багатшарова прямого розповсюдження	Проекція Саммона	Категоризація всередині класу Аналіз даних	
	Хебб	Прямого поширення або змагання	Аналіз головних компонентів	Аналіз даних Стиснення даних	
		Мережа Хопфілда	Навчання асоціативної пам'яті	Асоціативна пам'ять	
	Змагальні		Змагальня	Векторне квантування	Категоризація Стиснення даних
			Мережа Кохонена	SOM Кохонена	Категоризація Аналіз даних
			Мережа ART	ART1, ART2	Категоризація
Змішані	Коррекція ошибки и соревнование	Мережа RBF	Алгоритм навчання RBF	Класифікація образів Аппроксимація функцій Передбачення, управління	

ШНМ добре підходять для розпізнавання образів і вирішення задач класифікації, оптимізації і прогнозування. Представимо деякі проблеми, які вирішуються з використанням нейронних мереж, які представляють інтерес для користувачів [12].

Зростання обсягів баз даних в техніці, бізнесі, медицині, екології і зростання вимог до точності рішення ставлять нові вимоги перед нейронними мережами. В табл. 3 приведені типи ШНМ та класи задач, які вони вирішують.

Таблиця 3 – Типи штучних мереж

Тип ШНМ	Архітектура ШНМ	Клас задач які вирішує ШНМ
Багатошаровий перцептрон		Апроксимація функцій, класифікація.
Самоорганізаційна карта Кохонена		Стиснення даних, виділення при знаків.
Мережа Хопфілда		Асоціативна пам'ять, кластеризація даних.

Класифікація образів. Завдання полягає у вказівці приналежності вхідного образу (наприклад, мовного сигналу чи рукописного символу), представленого вектором ознак, одному або декільком попередньо визначеним класам. Наприклад розпізнавання букв, розпізнавання мови, класифікація сигналу електрокардіограми, класифікація клітин крові.

Кластеризація (категоризація). При вирішенні задачі кластеризації, яка відома також як класифікація образів «без вчителя», відсутня навчальна вибірка



з мітками класів. Алгоритм кластеризації заснований на подібності образів і розміщує близькі образи в один кластер. Відомі випадки застосування кластеризації для стиснення даних і дослідження властивостей даних.

Апроксимація функцій. Припустимо, що є навчальна вибірка (пари даних вхід-вихід), яка генерується невідомою функцією  $F(x)$ , спотвореної шумом. Завдання апроксимації полягає в знаходженні оцінки невідомої функції  $F(x)$ . Апроксимація функцій необхідна при рішенні численних інженерних і наукових задач моделювання.

Прогнозування. Завдання полягає в передбаченні деякого значення у деякий майбутній момент часу. Передбачення мають значний вплив на прийняття рішень у бізнесі, науці і техніці. Передбачення цін на фондовій біржі і прогноз погоди є типовими програмами прогнозування.

Оптимізація. Численні проблеми в математиці, статистиці, техніці, науці, медицині та економіці можуть розглядатися як проблеми оптимізації. Завданням алгоритму оптимізації є знаходження такого рішення, яке задовольнить обмеженням системи і максимізує чи мінімізує цільову функцію. Задача комівояжера є класичним прикладом задачі оптимізації.

Пам'ять, що адресується за змістом. У моделі обчислень фон Неймана звертання до пам'яті доступно тільки за допомогою адреси, який не залежить від утримання пам'яті. Більш того, якщо допущена помилка в обчисленні адреси, то може бути знайдена зовсім інша інформація. Асоціативна пам'ять, чи пам'ять, що адресується за змістом, доступна за вказівкою заданого змісту. Зміст пам'яті може бути викликаний навіть по частковому входу. Асоціативна пам'ять надзвичайно ефективна при створенні мультимедійних інформаційних баз даних.

Програмні продукти на базі нейронних мереж використовують для контролю якості води та можуть знаходити пластикові бомби в багажі авіапасажирів. Фахівці інвестиційного банку за допомогою програмного нейропакета роблять короткострокові прогнози коливань курсів валют.

Проведення аналізу вирішення завдань на нейронних мережах дозволяє виділити основні переваги їх використання:

Рішення завдань при невідомих закономірностях. Використовуючи здатність навчання на безлічі прикладів, нейронна мережа здатна вирішувати завдання, в яких невідомі закономірності розвитку ситуації і залежності між вхідними і вихідними даними. Традиційні математичні методи та експертні системи в таких випадках не використовують.

Стійкість до шумів у вхідних даних. Можливість роботи при наявності великого числа неінформативних, шумових вхідних сигналів. Немає необхідності робити їх попередній відсів, нейронна мережа сама визначить їх мало придатне для вирішення задачі та відкине їх.

Адаптація до змін навколишнього середовища. Нейронні мережі мають здатність адаптуватися до змін навколишнього середовища. Зокрема, нейронні мережі, навчені діяти в певному середовищі, можуть бути легко перевчені для роботи в умовах незначних коливань параметрів середовища. Більш того, для роботи в нестационарній середовищі (де статистика змінюється з плином часу, наприклад як котирування акцій на біржі) можуть бути створені нейронні мережі, вони перенавчаються в реальному часі. Чим вище адаптивні здатності системи, тим більш стійкою буде її робота в нестационарній середовищі. При цьому слід зауважити, що адаптивність не завжди веде до стійкості; іноді вона призводить до зовсім протилежного результату. Наприклад, адаптивна система з параметрами, що швидко змінюються в часі, може також швидко реагувати на сторонні порушення, що викличе втрату продуктивності. Для того, щоб використовувати всі переваги адаптивності, основні параметри системи повинні бути досить стабільними, щоб можна було не враховувати зовнішні перешкоди, досить гнучкими, щоб забезпечити реакцію на істотні зміни середовища.

Нейронні мережі мають потенційну надвисоку швидкість за рахунок використання масового паралелізму обробки інформації.

Нейронні мережі потенційно відмовостійкі при апаратній реалізації. Це означає, що за несприятливих умов їх продуктивність падає незначно.

Наприклад, якщо пошкоджений якийсь нейрон або його зв'язки, вилучення запам'ятовані інформації ускладнюється. Проте, враховуючи розподілений характер зберігання інформації в нейронній мережі, можна стверджувати, що тільки серйозні пошкодження структури нейронної мережі суттєво вплинуть на її працездатність. Тому зниження якості роботи нейронної мережі відбувається повільно.

## **1.5 Висновки**

В ході аналізу підходів до проектування нейронних мереж, їх архітектури та способів навчання, було вирішено в програмному продукті реалізувати нейронну мережу з архітектурою – багатошаровий перцептрон, з можливістю корегування кількості шарів. В якості правила навчання обрано корекцію помилки з алгоритмом зворотного розповсюдження помилки. Так як обрана конфігурація мережі підходить для розпізнавання образів, то було обрано відповідні навчальні вектори, для даної мережі. Зважаючи на конфігурацію мережі та задачі які вона має вирішувати, в якості функції активації для нейрону обрано гіперболічний тангенс.

## 2 МЕТОДИ ПАРАЛЕЛІЗАЦІЇ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

Паралельна обробка - це об'єднання декількох процесорів для вирішення завдання. Це дозволяє зменшити час, який витрачається на обчислення, якщо порівнювати результати використання паралельних процесів і результати, що надаються одним процесором.

Це дозволяє вирішувати об'ємні завдання і завдання фіксованої розмірності набагато швидше. Якщо говорити про можливість обробки інформації людським мозком, він так само використовує метод паралельної обробки, так як містить в собі величезну кількість інформації та взаємопов'язаних оброблюючих елементів.

Модель штучної нейронної мережі містить в собі кілька паралельних структур. Їх використовують для поліпшення ефективності реалізації на паралельних архітектурах. Існують кілька типів рівнів паралелізації в штучних нейронних мережах.

Так Нордстремом були виділені основні рівні, де проводиться паралелізації при фазі навчання:

- 1) рівень фази навчання
- 2) рівень навчальної вибірки
- 3) рівень шару
- 4) рівень нейрона
- 5) рівень вагів [13]

Всі ці рівні паралелізації знаходяться в нейронній мережі і можуть використовуватися одночасно. Для того, щоб вибрати необхідний рівень необхідно відштовхуватися від кількості нейронів в мережі, процесорів, особливостей комп'ютерної архітектури ІНС і певного завдання, поставленого перед нею.

При навчанні штучної нейронної мережі відбувається повне її навчання по всій мережі. Для того, щоб визначити необхідні параметри для вирішення певної задачі, навчання складається з безлічі сеансів експериментального підбору. Наприклад, підбираються параметри необхідної кількості нейронів у всіх шарах нейронної мережі, швидкості обробки даних (навчання).

## 2.1 Паралелізація фази навчання

Використовуючи метод паралельної обробки даних при фазі навчання в штучних нейронних мережах різні конфігурації мережі можуть досліджуватися в один і той же час. Так, на зображенні представлений паралелізм тренувальної сесії, де різниця між фазами навчання представлена у вигляді швидкості навчання.

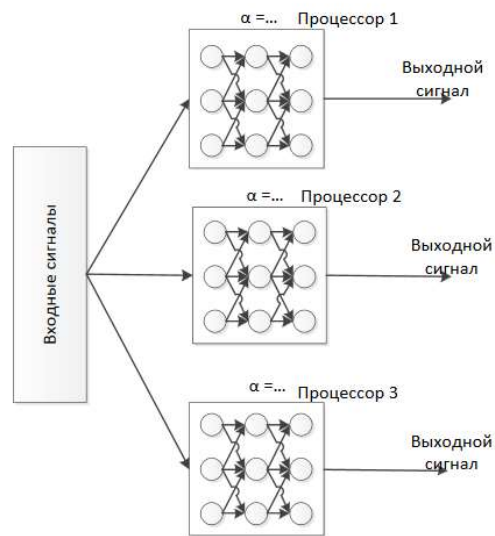


Рисунок 3 — Схема паралелізації фази навчання [18]

При стадії підбору параметрів необхідно враховувати, що результати, отримані під час навчання залежать від початкових значень ваг. Так, при застосуванні алгоритму зворотного поширення помилки з градієнтної пошукової методикою результати навчання дуже чутливі до первинних значень ваг. Аналогічна ситуація зустрічається і в мережах Хопфілда при вирішенні задач оптимізації.

При використанні ж методу паралельної обробки даних при фазі навчання можна проводити дослідження мережі при різних первинних установах. Так само при використанні паралелізації навчання серед плюсів виділяється лінійний приріст швидкодії фази. Це можливо завдяки відсутності необхідності в взаємодії між процесорами.

## 2.2 Паралелізація навчання вибірки

При використанні паралелізації на рівні навчання вибірки використовують навчання одночасно на декількох різних навчальних вибірках. Це важливо, тому що часто потрібна досить велика кількість навчальних векторів в штучній нейронній мережі для вирішення певної задачі великого розміру.

При системі, що використовує один потік векторів, вони будуть направлені в мережу послідовно, по чергово, а при паралельній системі - навчальні вектори поділяються між процесорами, де кожен з процесорів має копією всіх даних штучної нейронної мережі. Тобто кожен процесор навчається на декількох вибірках. На рис. 4 представлена дана схема роботи:

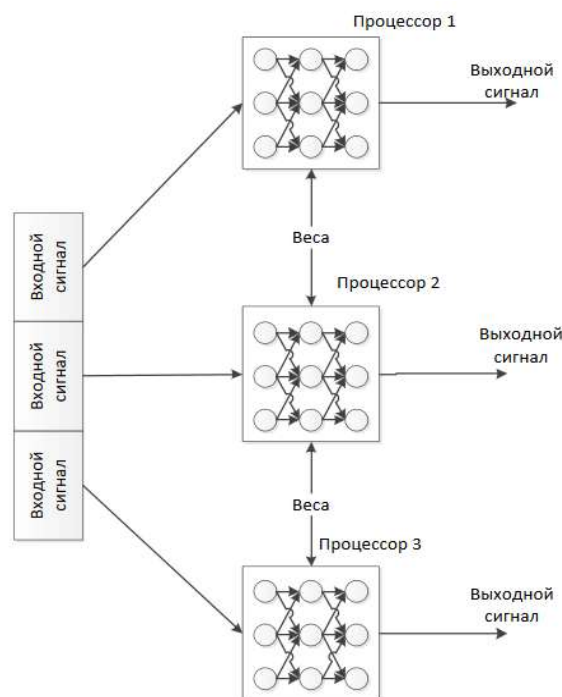


Рисунок 4 – Схема паралелізації навчальної вибірки [18]

## 2.3 Паралелізація на рівні шарів

При паралелізації рівнів шарів, використовуваних в моделях неокогнітрона і в моделях зі зворотним поширенням помилки вектори, що знаходяться на стадії навчання йдуть один за одним по мережі і знаходяться в мережі в один час. У той же час канали деяких мереж нейронної мережі можуть йти по зворотному напрямку. У моделі зі зворотним поширенням помилки, вони можуть переходити і на зворотні шари назад.

Це дозволяє передати операцію іншому процесору. Такий же принцип може бути і в моделях штучних нейронних мереж, які не мають верств, але вектори так само йдуть через процесори за принципом конвеєра. Найточніші вектори, що пройшли через всі процесори, беруться за еталон.

Після їх повторного переходу по процесорам мережі, процесори оновлюють ваги відповідно до цих стандартів. На рис. 5 представлена дана схема роботи:

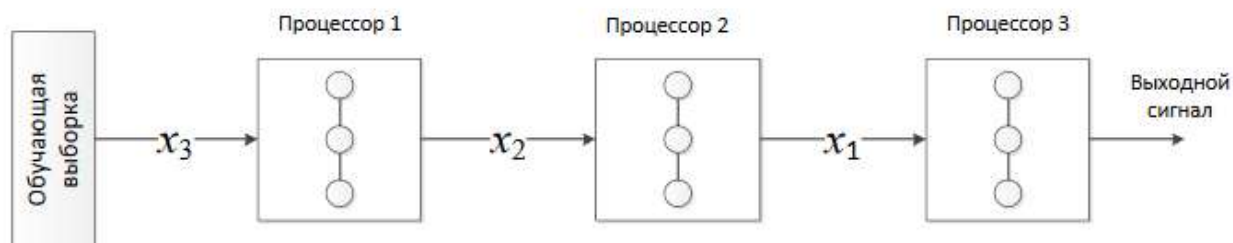


Рисунок 5 – Схема паралелізації на рівні шару [18]

## 2.4 Паралелізація на рівні нейронів

При використанні методу паралелізації на рівні нейронів, нейрон за функціями аналогічний процесору, так як є також обробляючим елементом. Паралельна обробка як процес на даному рівні розділяє нейрони в рамках одного шару по процесорам, а також при паралельних обчисленнях. В рамках даної системи обробки нейрон або деяке їх кількість співвідноситься з кожним

процесором. Така модель паралелізації є в кожній з моделей штучних нейронних мереж і є одним з популярних методів. Схема роботи даного методу на рис.6 :

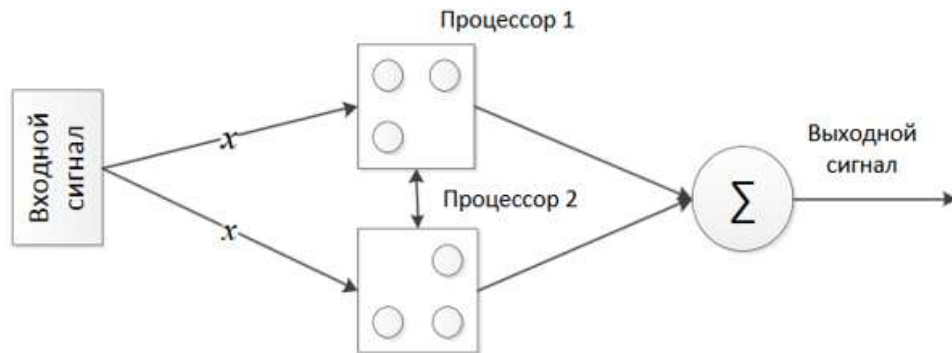


Рисунок 6 – Схема паралелізації на рівні нейрону [18]

## 2.5 Паралелізація на рівні вагів

Рівень вагів при методі паралелізації є дрібно модульним. Найчастіше даний метод використовують в апаратних реалізаціях. У даному методі обчислення в рамках одного нейрона розділені по декількох процесорах.

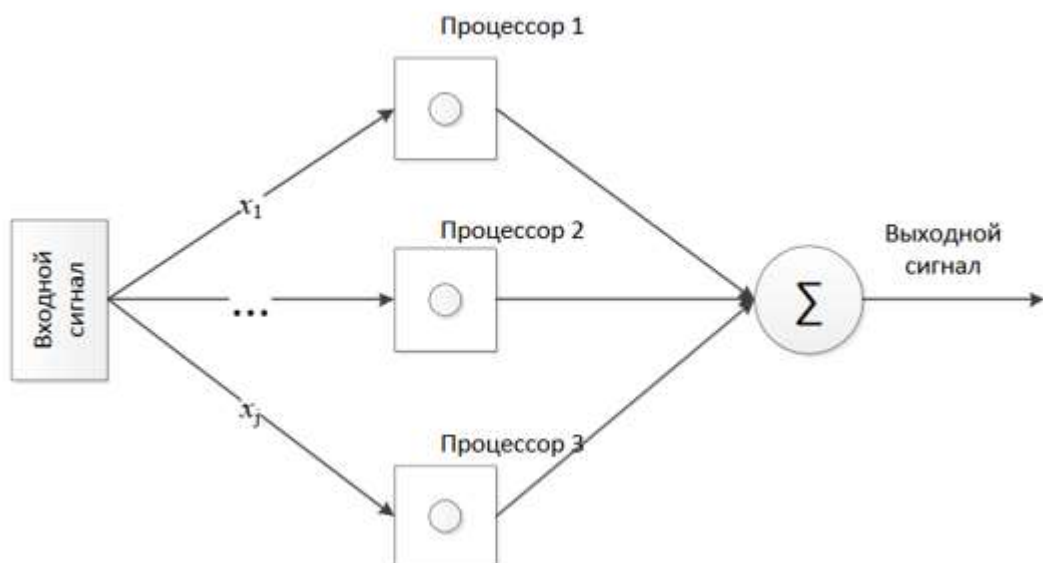


Рисунок 7 – Схема паралелізації на рівні вагів [18]



## 2.6 Ефективність процесу паралелізації

Для визначення ступеня успішності застосування вищевказаних методів паралелізації необхідно виробити систему релевантних показників. Для оцінки успішності паралельного алгоритму використовують такі релевантні показники продуктивності як ефективність, збільшення швидкості (приріст), масштабованість системи. При цьому порівнюється ефективність при використанні методу паралельної обробки і при звичайній послідовній роботі. При порівнянні повинен бути знайдений найбільш швидкий алгоритм для вирішення проблеми при використанні послідовної обробки, в такому випадку будуть помітні переваги використання паралельного методу. Також необхідно мати на увазі, що може бути кілька паралельних алгоритмів, проте не всі найбільш швидкі алгоритми можуть бути придатними для процесу паралелізації.

Для оцінки швидкості роботи алгоритму порівнюємо час найшвидшого з алгоритмів розв'язання певної проблеми при послідовній системі обробки даних і час, витрачений паралельним алгоритмом на певному числі процесорів для вирішення тієї ж проблеми. Якщо швидкість виконання паралельного алгоритму збільшилася в стільки ж разів, скільки процесорів було задіяно при розрахунку, то ми маємо справу з лінійним зростанням. Якщо швидкість виконання збільшилася сильніше, то алгоритм суперлінійний, якщо швидкість зросла менше, ніж при лінійному зростанні, то алгоритм сублінійний. Так само потрібно враховувати, що тут при використанні паралельного алгоритму оброблюючий елемент рівнозначний послідовному алгоритму.

Розглянемо ефективність процесу паралелізації.

Ефективністю в даному випадку є ступінь задіяності процесорів, які використовуються при паралельному алгоритмі обробки.

Час необхідний для навчання на багатопоточній системі можна виразити формулою:

$$\tau = \frac{t}{n} + \varphi, \quad (3)$$

де  $t$  – час навчання на однопоточній системі,  $n$  – кількість потоків. Таким чином, існує деяке  $\phi$  яке відображає різницю між очікуваним часовим виграшем (в  $n$  разів) і реальним.  $\phi$  – складається з часу передачі даних, часу на підготовку даних, часу очікування інших вузлів, побічних часових затрат. Наявність даних величин зумовлено тим, що паралельні алгоритми відрізняються від послідовних. Величина  $\phi$  – показує лише абсолютну різницю між однопотоковим і багатопотоковими часовими затратами, що не дозволяє адекватно оцінити паралельну версію того чи іншого методу навчання, тому необхідно використовувати величину

$$\alpha = \frac{t}{n\tau}, \quad (4)$$

коефіцієнт ефективності паралельного алгоритму. Ця величини показує відносний виграш у часі, та становить (0.7; 1) для ефективних алгоритмів.

При ідеальному випадку з лінійним збільшенням швидкості роботи ефективність буде рівнозначна 1. Але на ділі в більшості випадків при реалізації паралельного алгоритму обробки буває сублінійний приріст швидкості.

Показник масштабованості системи показує здатність системи впливати на прискорення обробки пропорційно кількості процесорів. Чим більше процесорів використовується, тим більше падає ефективність при фіксованому розмірі завдання. За підтримки ефективності системи шляхом збільшення розмірності задачі, система паралелізації буде масштабується.

При збільшенні числа процесорів до конкретного завдання з'являється проблема збалансованості навантаження. При постійному розмірі завдання навантаження на процесори зменшується. Щоб не було проблеми недостачі роботи процесорів, використовують масштабованість завдання.

При збільшенні числа процесорів ефективність падає. Так Амдал встановив верхній поріг приросту продуктивності, якого можливо досягти при використанні методу паралелізації. За його теорією, на практиці обчислення можуть виконуватися послідовно, не паралельно і, власне, паралельно. Так як

кількість процесорів збільшується, час роботи, витрачений на паралельні обчислення зменшується. При цьому послідовна частина роботи над обчисленнями залишається постійною.

Час, витрачений на реалізацію послідовних обчислень, що не масштабується при збільшенні числа процесорів і швидкісний лінійний приріст досягається тільки при відсутності послідовної частини під час вирішення певної задачі. Але так як більшість завдань все ж містять в собі послідовні обчислення, максимальний приріст швидкості обмежується.

## 2.7 Технології паралелізації

Паралельні обчислення — це форма обчислень, в яких кілька дій проводяться одночасно. Ґрунтуються на тому, що великі задачі можна розділити на кілька менших, кожен з яких можна розв'язати незалежно від інших.

Є кілька різних рівнів паралельних обчислень: бітовий, інструкцій, даних та паралелізм задач. Паралельні обчислення застосовуються вже протягом багатьох років, в основному в високопродуктивних обчисленнях, але зацікавлення ним зросло тільки недавно, через фізичні обмеження зростання частоти.. Оскільки споживана потужність (і відповідно виділення тепла) комп'ютерами стало проблемою в останні роки, паралельне програмування стає домінуючою парадигмою в комп'ютерній архітектурі, основному в формі багатоядерних процесорів.[14]

Паралельні комп'ютери можуть бути грубо класифіковані згідно з рівнем, на якому апаратне забезпечення підтримує паралелізм: багатоядерність, багатопроцесорність — комп'ютери, що мають багато обчислювальних елементів в межах одної машини, а також кластери, МРР, та ґрід — системи що використовують багато комп'ютерів для роботи над одним завданням. Спеціалізовані паралельні архітектури іноді використовуються поряд з традиційними процесорами, для прискорення особливих задач.

Програми для паралельних комп'ютерів писати значно складніше, ніж для послідовних, бо паралелізм додає кілька нових класів потенційних помилок, серед яких є найпоширенішою стан гонитви. Комунікація, та синхронізація процесів зазвичай одна з найбільших перешкод для досягнення хорошої продуктивності паралельних програм. Однак існують рішення які дозволяють розпаралелити програму з відносно невеликим збільшенням складності коду, імовірності виникнення помилок.

### 2.7.1 OpenMP

Одним з найбільш популярних засобів програмування для комп'ютерів із загальною пам'яттю, що базуються на традиційних мовах програмування і використання спеціальних коментарів, в даний час являється технологія OpenMP. За основу береться послідовна програма, а для створення її паралельної версії користувачеві надається набір директив, функцій і змінних оточення.

Передбачається, що створювана паралельна програма буде переноситься між різними комп'ютерами з пам'яттю, що підтримують OpenMP API.

Технологія OpenMP націлена на те, щоб користувач мав один варіант програми для паралельного і послідовного виконання. Однак є можливість створювати програми, які працюють коректно лише в паралельному режимі або дають в послідовному режимі інший результат. Більш того, через накопичення помилок округлення результат обчислень із використанням користуванням різної кількості ниток може в деяких випадках відрізнятись.

Інтерфейс OpenMP задуманий як стандарт для програмування на SMP-системах (SSMP, ccNUMA та інших) в моделі загальної пам'яті (shared memory model). У стандарт OpenMP входять специфікації набору директив компілятора, допоміжних функцій і змінних середовища. OpenMP реалізує паралельні обчислення за допомогою багатопоточності, в якій «головний» потік створює набір «підлеглих» потоків, і завдання розподіляється між ними. Передбачається, що потоки виконуються паралельно на машині з декількома процесорами,

причому кількість процесорів не обов'язково має бути більше або дорівнювати кількості потоків.

POSIX-інтерфейс для організації ниток (Pthreads) підтримується практично на всіх UNIX-системах, однак з багатьох причин не підходить для практичного паралельного програмування: в ньому немає підтримки мови Фортран, занадто низький рівень програмування, немає підтримки паралелізму за даними, а сам механізм ниток спочатку розроблявся не для цілей організації паралелізму. OpenMP можна розглядати як високорівневу надбудову над Pthreads (або аналогічними бібліотеками ниток); в OpenMP використовується термінологія і модель програмування, близька до Pthreads, наприклад, динамічно породжувані нитки, загальні і роздільні дані, механізм «замків» для синхронізації.

Відповідно до термінології POSIX threads, будь який UNIX-процес складається з декількох ниток управління, які мають загальний адресний простір, але різні потоки команд і роздільні стеки. У найпростішому випадку процес складається з однієї нитки. Нитки іноді називають також потоками, легковаговими процесами, LWP (light-weight processes).[15]

Важливою перевагою технології OpenMP є можливість реалізації так званого інкрементального програмування, коли програміст поступово знаходить ділянки в програмі, що містять ресурс паралелізму, за допомогою наданих механізмів робить їх паралельними, а потім переходить до аналізу наступних ділянок. Таким чином, в програмі не розпаралелених частин поступово стає все менше. Такий підхід значно полегшує процес адаптації послідовних програм до паралельних комп'ютерів, а також налагодження та оптимізацію.

### 2.7.2 MPI

MPI — це специфікація, що була розроблена в 1993–1994 роках групою MPI Forum, і забезпечує реалізацію моделі обміну повідомленнями між процесами. Остання версія цієї специфікації: MPI-2. У моделі програмування

MPI програма породжує кілька процесів, що взаємодіють між собою за допомогою виклику підпрограм прийому й передачі повідомлень.

Зазвичай, при ініціалізації MPI-програми створюється фіксований набір процесів, причому (що, утім, необов'язково) кожний з них виконується на своєму процесорі. У цих процесах можуть виконуватися різні програми, тому MPI-модель іноді називають MIMD-моделлю (Multiple Instruction, Multiple Data), на відміну від SIMD моделі, де на кожному процесорі виконуються тільки однакові задачі. MPI підтримує двохточкові й глобальні, синхронні й асинхронні, блокуючі й неблокуючі типи комунікацій. Спеціальний механізм — комунікатор — ховає від програміста внутрішні комунікаційні структури. Структура комунікацій може змінюватися протягом часу життя процесу, але кількість задач має залишатися постійною (MPI-2 підтримує динамічну зміну кількості задач).

Специфікація MPI забезпечує переносимість програм на рівні вихідних кодів. Підтримується робота на гетерогенних кластерах і симетричних мультипроцесорних системах. Не підтримується запуск процесів під час виконання MPI-програми. У специфікації відсутні опис паралельного вводу-виводу й налагодження програм — ці можливості можуть бути включені до складу конкретної реалізації MPI у вигляді додаткових пакетів чи утиліт. Сумісність різних реалізацій не гарантується.

Важливою властивістю паралельної програми є детермінізм — програма має завжди давати однаковий результат для однакових наборів вхідних даних. Модель передачі повідомлень загалом такої властивості не має, оскільки не визначено порядок одержання повідомлень від двох процесів третім. Якщо ж один процес послідовно надсилає кілька повідомлень іншому процесу, MPI гарантує, що одержувач отримає їх саме в тому порядку, у якому вони були надіслані. Відповідальність за забезпечення детермінованого виконання програми покладається на програміста.

В обчислювальних системах з розподіленою пам'яттю процесори працюють незалежно один від одного. Для організації паралельних обчислень в таких умовах необхідно мати можливість розподіляти обчислювальне

навантаження і організувати інформаційну взаємодію (передачу даних) між процесорами.

Рішення всіх перерахованих питань і забезпечує інтерфейс передачі даних MPI.

У загальному плані, для розподілу обчислень між процесорами необхідно проаналізувати алгоритм вирішення задачі, виділити інформаційно незалежні фрагменти обчислень, провести їх програмну реалізацію і потім розмістити отримані частини програми на різних процесорах. В рамках MPI прийнятий більш простий підхід - для вирішення поставленого завдання розробляється одна програма і ця єдина програма запускається одночасно на виконання на всіх наявних процесорах! При цьому для того, щоб уникнути ідентичності обчислень на різних процесорах, можна, по-перше, підставляти різні дані для програми на різних процесорах, а по-друге, в MPI є засоби для ідентифікації процесора, на якому виконується програма (і тим самим, надається можливість організувати відмінності в обчисленнях в залежності від використовуваного програмою процесора).

Подібний спосіб організації паралельних обчислень отримав найменування моделі "одна програма безліч процесів" (single program multiple processes or SPMP)).

Для організації інформаційної взаємодії між процесорами в самому мінімальному варіанті досить операції прийому і передачі даних (при цьому, звичайно, повинна існувати технічна можливість комунікації між процесорами - канали або лінії зв'язку) У MPI існує безліч операцій передачі даних. Вони забезпечують різні способи пересилання даних, реалізують практично всі розглянуті в розділі 3 комунікаційні операції. Саме дані можливості є найбільш сильною стороною MPI.

Отже, MPI - це програмні засоби, які забезпечують можливість передачі повідомлень і при цьому відповідають всім вимогам стандарту MPI. Так, за стандартом, ці програмні засоби повинні бути організовані у вигляді бібліотек

програмних модулів (бібліотеки MPI) і повинні бути доступні для найбільш широко використовуваних алгоритмічних мов C і Fortran.

## **2.8 Висновки**

Розглянуті основні підходи та технології паралелізації штучних нейронних мереж. Зважаючи на обмеженість в ресурсах, як технологію паралелізації було обрано OpenMP, адже вона дозволяє легко запускати паралельні версії програм без значних змін вихідного коду, на відміну від інших аналогів. Проаналізувавши підходи до паралелізації, для подальшої реалізації в програмному продукті було обрано метод паралелізації навчання вибірки адже при використанні паралелізації на рівні навчання вибірки використовують навчання одночасно на декількох різних навчальних вибірках. Це важливо, тому що потрібна досить велика кількість навчальних векторів в штучній нейронній мережі для вирішення задачі навчання.



## 3 РОЗРОБКА ПРОГРАМИ ПАРАЛЕЛЬНОГО НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ

### 3.1 Засоби та платформа реалізації

Проаналізувавши задачу яка має бути реалізована, зрозуміло що перевага має надаватися мові програмування високого рівня, яка здатна продуктивно виконувати вихідний код програми. Так було обрано для розробки мову C++ в поєднанні з бібліотекою QT. QT – крос-платформовий інструментарій розробки програмного забезпечення (ПЗ) мовою програмування C++. Дозволяє запускати написане за його допомогою ПЗ на більшості сучасних операційних систем (ОС), просто компілюючи текст програми для кожної операційної системи без зміни сирцевого коду. Містить всі основні класи, які можуть бути потрібні для розробки прикладного програмного забезпечення, починаючи з елементів графічного інтерфейсу й закінчуючи класами для роботи з мережею, базами даних, OpenGL, SVG і XML. Бібліотека дозволяє керувати нитями, працювати з мережею та забезпечує крос-платформовий доступ до файлів.

Qt 5, який вийшов у грудні 2012, примітний модульною структурою та зміщенням акценту в бік використання для написання застосунків засобів декларативного опису інтерфейсу з визначенням логіки взаємодії з користувачем мовою JavaScript, у той час як застосування C++ позиціонується для реалізації критичних до часу виконання або надмірно складних частин програми, а також для створення нових модульних бекендів для Qt Quick. Незважаючи на багато істотних поліпшень і змін, Qt 5 зберігає базову зворотну сумісність із минулими випусками, підтримує повною мірою засоби для створення Qt-програм мовою C++ і містить майже всі компоненти Qt 4 (припинена підтримка давно застарілих елементів), більшість модулів колишнього Qt Mobility й деякі експериментальні елементи з Qt Labs.

В якості засобів паралелізації обрано QTConcurrent – високорівневе API, яке дозволяє запускати багатопоточні програми без використання

низькорівневих потокових примітивів, для розмежування основної функціональності та інтерфейсу користувача. В якості основного засобу паралелізації обрано технологію OpenMP через можливість простої реалізації однопоточного та паралельного виконання програми.

### 3.2 Розробка алгоритму роботи програми

Було досліджено достатньо методів та технологій для реалізації паралельного методу навчання нейронної мережі. Вирішено використовувати метод паралелізації на рівні вибірки – коли навчальні вектори поділяються між процесорами та кожен з процесорів має копію всіх даних штучної нейронної мережі. Тобто кожен процесор навчається на декількох вибірках.

В якості алгоритму навчання було обрано метод навчання з вчителем для багат шарового перцептору – метод зворотного розповсюдження помилки.

Це ітеративний градієнтний алгоритм, який використовується з метою мінімізації помилки роботи багат шарового перцептрону та отримання результату на виході. Основна ідея цього методу полягає в поширенні сигналів помилки від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи.

У мережі є множина входів  $x_1 \dots x_n$ . Перенумеруємо всі вузли (включаючи входи і виходи) числами від 1 до  $N$  (наскрізна нумерація, незалежно від топології шарів). Позначимо через  $w_{ij}$  вагу зв'язку, що з'єднує  $i$ -й і  $j$ -й вузли, а через  $o_i$  – вихід  $i$ -го вузла. Якщо нам відомий навчальний приклад (правильні відповіді мережі  $t_k, k \in Outputs$ ) то функція помилки, отримана за методом найменших квадратів, виглядає так:

$$E = (\{w_{ij}\}) = \frac{1}{2} \sum_{k \in Outputs} (t_k - o_k)^2, \quad (5)$$

Для модифікації вагів нейронів використовується стохастичний градієнтний спуск, тобто будемо корегувати ваги після кожного навчального прикладу  $i$ , таким чином, «рухатися» в багатовимірному просторі ваг. Щоб «дістатися» до мінімуму помилки, нам потрібно «рухатися» в сторону,

протилежну градієнту, тобто, на підставі кожної групи правильних відповідей, додавати до кожної ваги  $w_{ij}$

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}}, \quad (6)$$

Де  $0 < \eta < 1$  - множник, що задає швидкість «руху».

Похідна розраховується таким чином. Нехай спочатку  $j \in Outputs$  тобто вага, яка нас цікавить, входить в нейрон останнього рівня. Спочатку зазначимо, що  $w_{ij}$  впливає на вихід мережі лише як частина суми  $S_j = \sum_i w_{ij} x_i$  де сума береться по входах  $j$ -го вузла. Тому:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial S_j} \frac{\partial S_j}{\partial w_{ij}} = x_i \frac{\partial E}{\partial S_j}, \quad (7)$$

Аналогічно,  $S_j$  впливає на загальну помилку тільки в рамках виходу  $j$ -го вузла  $o_j$ . Тому

$$\frac{\partial E}{\partial S_j} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial S_j} = \left( \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in Outputs} (t_k - o_k)^2 \right) \left( \frac{\partial \sigma(S_j)}{\partial S_j} \right), \quad (8)$$

де  $\sigma()$  — це функція активації, у даному випадку (стосовно обчислення похідної) являє собою Експоненційну сигмоїду. Якщо ж  $j$ -й вузол — не на останньому рівні, то у нього є виходи; позначимо їх через  $Children(j)$ . У цьому випадку

$$\frac{\partial E}{\partial S_j} = \sum_{k \in Children(j)} \frac{\partial E}{\partial o_j} \frac{\partial S_k}{\partial S_j}, \quad (9)$$

Де  $\frac{\partial E}{\partial S_j}$  — це анологічна поправка, але обчислена для вузла наступного рівня (будемо позначати її через  $\delta_k$  від  $\Delta_k$  вона відрізняється відсутністю множника  $(-\eta x_{ij})$

Оскільки проводиться обчислення поправки для вузлів останнього рівня і вираження поправки для вузла нижчого рівня через поправки більш високого, можна вже створювати алгоритм навчання. Саме через цю особливість

обчислення поправок цей алгоритм називається алгоритмом зворотного поширення помилки.

На вхід алгоритму, крім зазначених параметрів, потрібно також подавати форматі структуру мережі. На практиці дуже гарні результати показують мережі досить простої структури, що складаються з двох рівнів нейронів — прихованого рівня (hidden units) і нейронів-виходів (output units), кожен вхід мережі з'єднаний з усіма прихованими нейронами, а результат роботи кожного прихованого нейрона подається на вхід кожному з нейронів-виходів. У такому випадку досить подавати на вхід кількість нейронів прихованого рівня.

Заважаючи на обрану технологію та підхід до паралелізації, загальний алгоритм можна описати наступною блок-схемою:

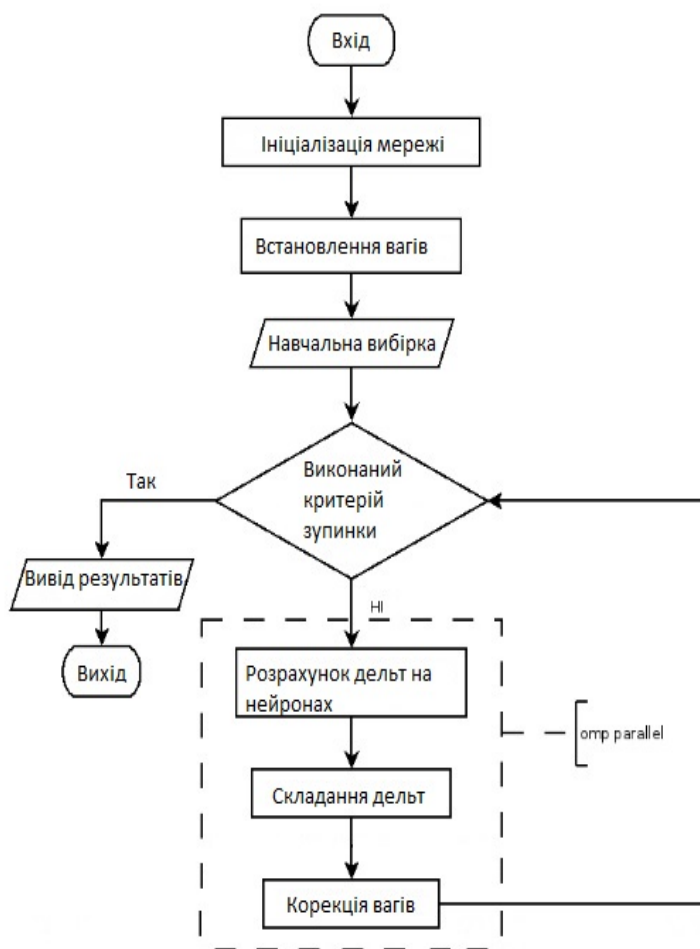


Рисунок 8 – Блок-схема алгоритму паралельного навчання

### 3.3 Архітектура програми

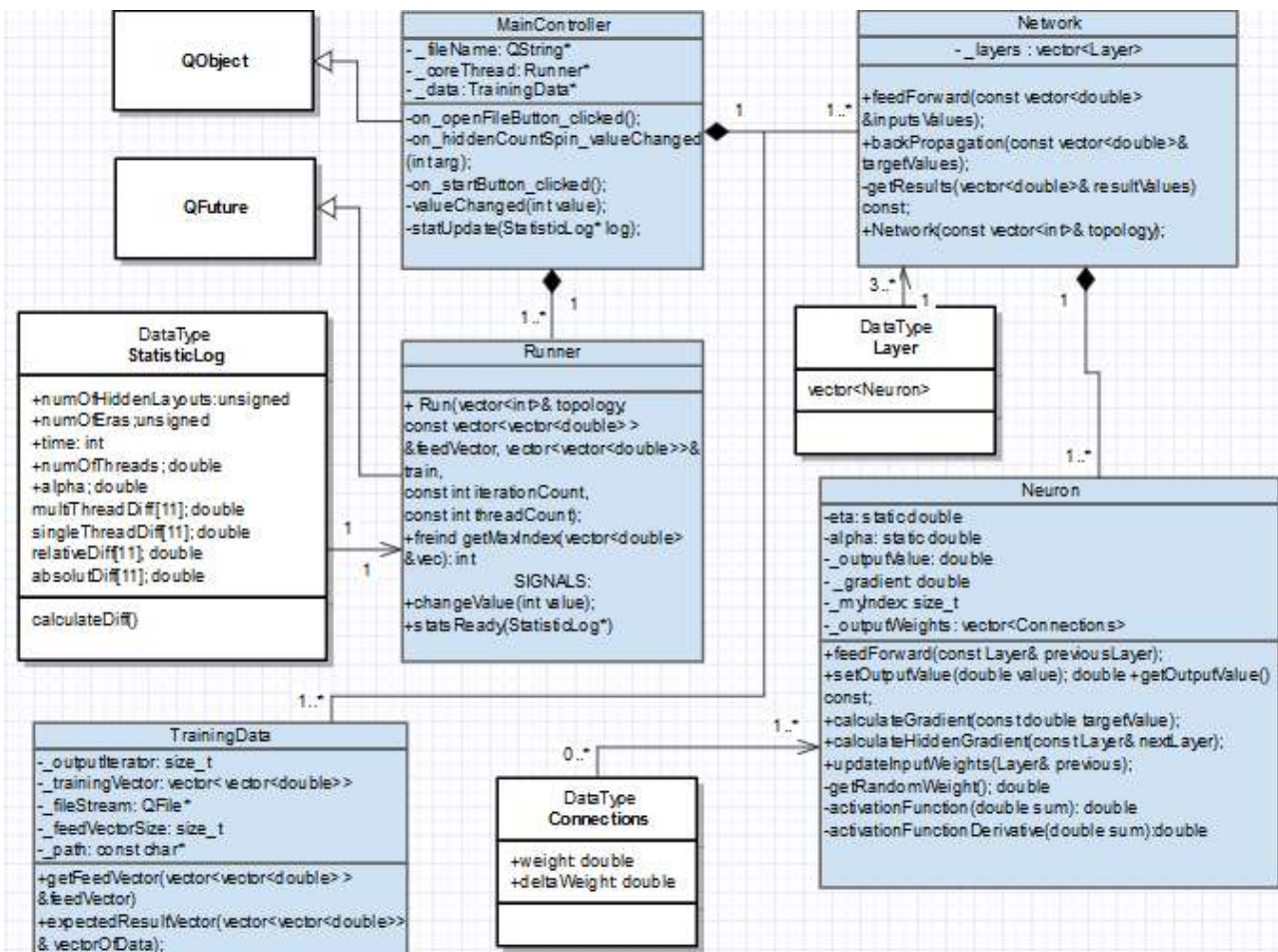


Рисунок 9 – UML діаграма класів програми

Програма розроблена згідно з наданою на рис. 9 діаграмою класів. Розглянемо складові діаграми, щоб детальніше зрозуміти роботу програми.

Клас MainController унаслідований від QObject відповідає за виконання основного функціоналу програми, а також застосовує параметри та функції які викликаються із графічного інтерфейсу користувача. Відповідає за запуск нового потоку в якому будуть проводитися обчислення, не залежно від інтерфейсу, а також приймає та відображає статистичну та графічну статистику після завершення навчання мережі.

TrainingData – реалізує зчитування навчального набору даних, у форматі Semion dataset - бази даних зразків рукописного написання цифр. Дані складаються з заздалегідь підготовлених прикладів зображень, на основі яких

проводиться навчання та тестування систем. Дані представляють собою набір із 1593 рукописних символів написаних 80 різними людьми, які були відскановані, нормалізовані та зменшені до розміру 16x16.

Runner – клас, що реалізує високорівневе API QConcurrent / QFuture, що дозволяє винести основну функціональність в окремий потік, що в свою чергу запобігає зависанню інтерфейсу користувача. Саме в функції Run цього класу і відбувається основна паралелізація навчання. Ця функція виконується двічі- в послідовному та паралельному режимах. Також після виконання функції Run формується StatisticLog – службовий клас для збереження інформації про часові затрати даного раунду навчання, далі він використовується для представлення статистики в текстовому, або графічному режимах.

Клас Network реалізує навчання нейронної мережі методом зворотного розповсюдження помилки. Даний клас забезпечує функціонал зображений на рис. 8, а саме: ініціалізацію мережі та встановлення вагів в конструкторі класу, ініціалізацію входів мережі згідно навчального вектору за допомогою функції feedVector. Сам алгоритм зворотнього розповсюдження помилки реалізований функцією backpropagation.

За розрахунок дельт на нейронах та їх складання відповідає клас Neuron. Також в ньому реалізовані функція активації та її похідна, в нашому випадку в якості функції активації було обрано гіперболічний тангенс, а в якості похідної його апроксимацію  $1 - x^2$ . Також в даному класі зберігаються змінні що відповідають за швидкість навчання мережі.  $\eta(eta)$  – коефіцієнт швидкості навчання мережі та  $\alpha$  – множник вагів попередніх вагів. Експериментально було визначено значення цих змінних 0.35 та 0.05 відповідно.

### 3.4 Функціональні аспекти системи

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів( варіантів використання ) обмежених границею системи

(прямокутник), асоціації між акторами та прецедентами, відношення серед прецедентів та відношень узагальнення між акторами.

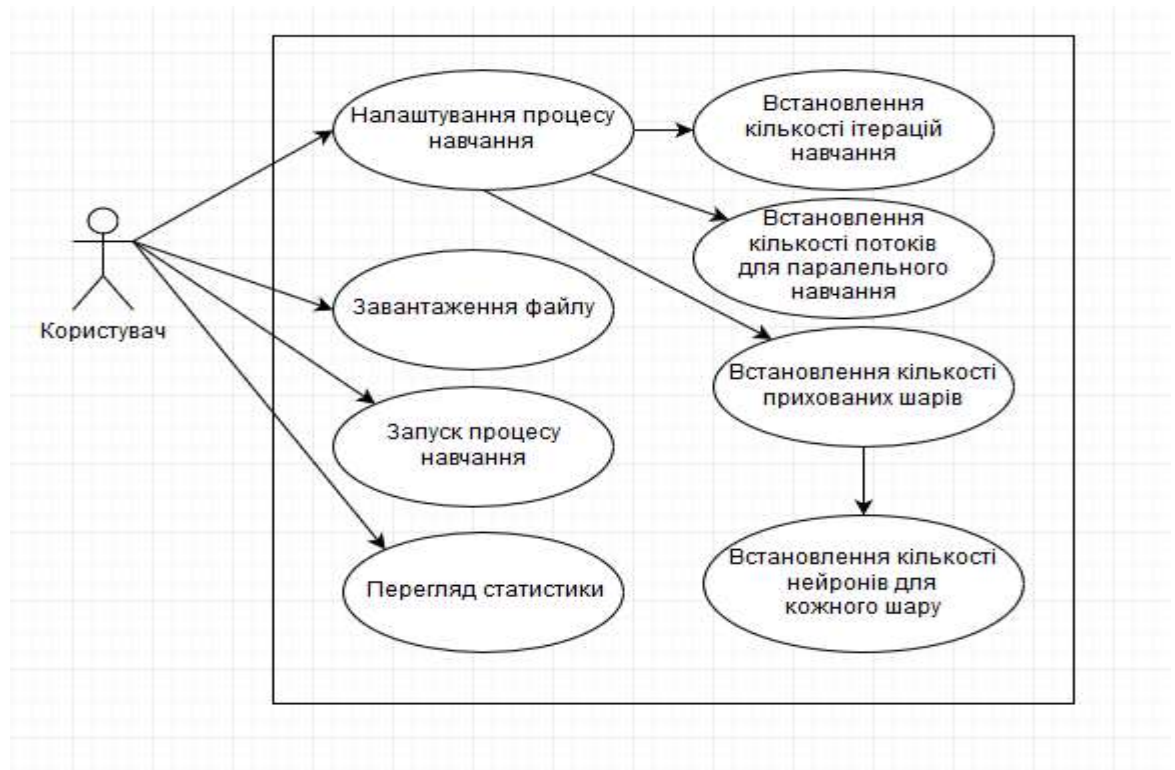


Рисунок 10 – Діаграма прецедентів

У системі представлено одну роль – Користувач. Взаємодія з користувачем відбувається за рахунок графічного інтерфейсу:

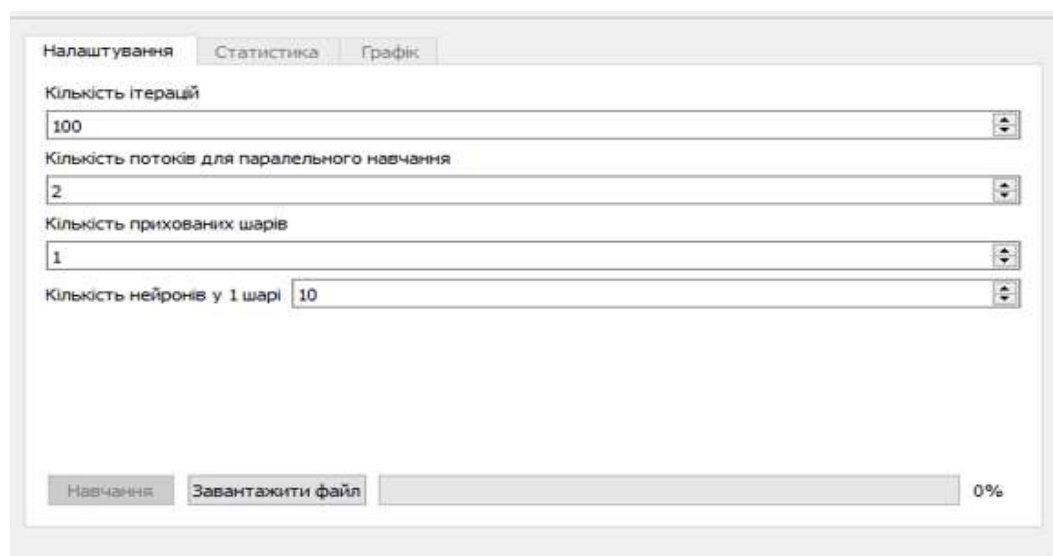


Рисунок 11 – Графічний інтерфейс програми

Інтерфейс представлений у вигляді 3х вкладок Налаштування, Статистика, та Графік. Вкладки Статистика та Графік стають доступними відразу після завершення процесу навчання.

На сторінці налаштувань має доступ налаштувань процесу навчання, а саме, може встановлювати: кількість ітерацій навчання, потоків для паралельного режиму навчання, кількість прихованих шарів нейронної мережі, а також кількість нейронів у кожному шарі. При цьому накладаються наступні обмеження:

- Кількість ітерацій [100 – 10000]
- Кількість потоків [2 – 8]
- Кількість прихованих шарів [1 – 4]
- Кількість нейронів для кожного шару [10 – 512]

Також користувач має можливість завантажувати файли для навчання типу *semeion data set*.

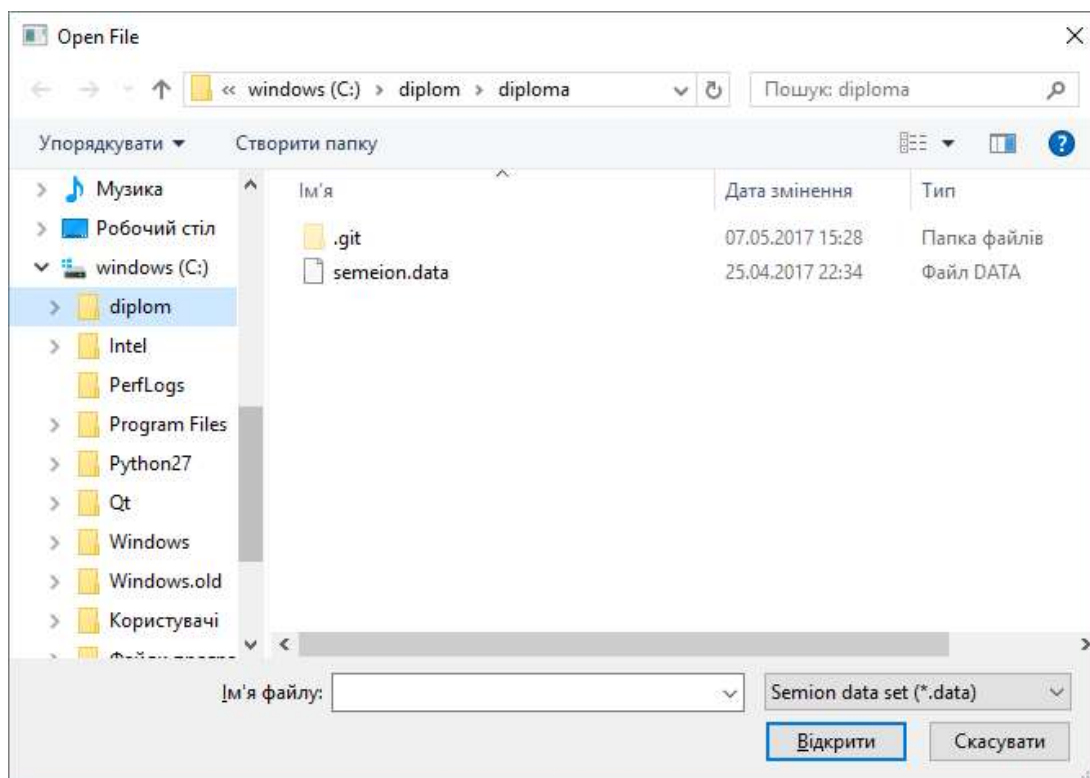


Рисунок 12 – Діалогове вікно завантаження файлу



Після налаштувань, та вибору файлу користувач може почати процес навчання. Навчання відбувається двічі в 1 однопоточному і паралельному режимах відповідно. Поточний статус виконання відображається за допомогою прогрес бару.

На вкладці Статистика користувач може переглянути наступну інформацію:

- Кількість Ітерацій
- Кількість Прихованих шарів
- Кількість ядер системи
- Кількість згенерованих потоків
- Час однопоточного навчання
- Час багатопоточного навчання
- Коефіцієнт ефективності (обрахований по формулі 4)

А також таблицю часових затрат на кожну декаду навчання, для однопоточного та паралельного режимів, абсолютна та відносна різниця між режимами.

		SingleThread	Muti Thread	Differense	Relative Differense	
Кількість ітерацій	1000					
Кількість прихованих шарів	1					
Кількість ядер	2	Час 1 декади	56270	25849	30421	117.687%
Кількість потоків	2	Час 2 декади	57594	27393	30201	110.251%
Час однопоточного навчання(мс)	408061	Час 3 декади	38231	26544	11687	44.0288%
Час багатопоточного навчання	257599	Час 4 декади	35578	24840	10738	43.2287%
Коефіцієнт ефективності	0.792047	Час 5 декади	36286	25678	10608	41.3116%
		Час 6 декади	36009	25061	10948	43.6854%
		Час 7 декади	37224	25174	12050	47.8668%
		Час 8 декади	36059	26154	9905	37.8718%
		Час 9 декади	37820	25462	12358	48.5351%
		Час 10 декади	36990	25444	11546	45.3781%
		Повний час	408061	257599	150462	58.4094%

Рисунок 13 – Вкладка статистики

Також абсолютна різниця у часі виконання між декадами демонструється в якості графіка:

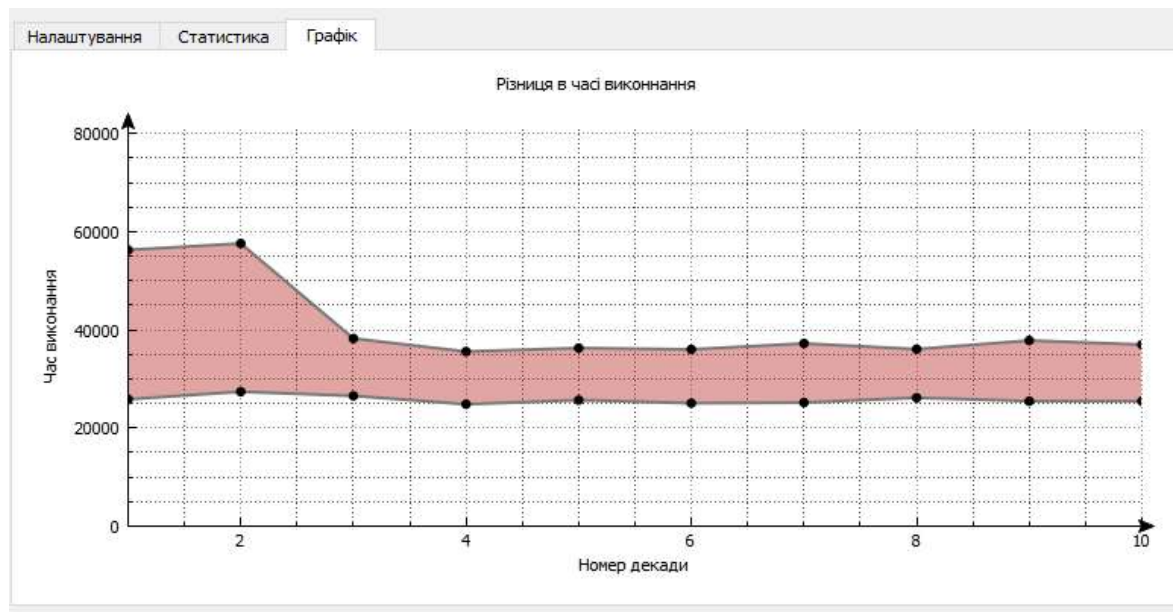


Рисунок 14 – Графічне представлення різниці в часі виконання

### 3.5 Висновки

В розділі наведені деталі алгоритму та архітектури програми. Розробка програми паралельного навчання нейронної мережі проходила відповідно до спроектованої архітектури, та визначених основних функціональних аспектів системи. Розроблено програму з використанням мови високого рівня C++ та бібліотеки Qt. Для паралелізації програми застосовано готові методи та директиви бібліотеки OpenMP.

## 4 ТЕСТУВАННЯ ПРОГРАМИ

Для оцінки роботи будемо розглядати абсолютний та відносний виграш у часі та коефіцієнт ефективності. В тестах перевірено вплив паралелізації на різні моделі навчальної системи, де ступінь навчання залежить від кількості: ітерацій, нейронів, шарів, та комбінацій цих параметрів. Тести виконувались послідовно при однаковій завантаженості системи, та повторювались декілька разів, для усереднення результатів. Б

### 4.1 Тест № 1

Для першого тесту обрано такі параметри

- Кількість Ітерацій - 1000
- Кількість Прихованих шарів - 1
- Кількість потоків – 4
- Кількість нейронів у шарі - 10

Налаштування		Статистика				
Параметр	Значення	SingleThread	Muti Thread	Differense	Relative Differense	
Кількість ітерацій	1000					
Кількість прихованих шарів	1	Час 1 декади	9781	4368	5413	123.924%
Кількість ядер	2	Час 2 декади	9744	3699	6045	163.423%
Кількість потоків	4	Час 3 декади	9951	3662	6289	171.737%
Час однопоточного навчання(мс)	97742	Час 4 декади	9984	3735	6249	167.309%
Час багатопоточного навчання	38096	Час 5 декади	10037	3711	6326	170.466%
Коефіцієнт ефективності	0.88284	Час 6 декади	9875	3733	6142	164.533%
		Час 7 декади	9842	3785	6057	160.026%
		Час 8 декади	9551	3958	5593	141.309%
		Час 9 декади	9492	3735	5757	154.137%
		Час 10 декади	9485	3710	5775	155.66%
		Повний час	97742	38096	59646	156.568%

Рисунок 15 – Результати тесту №1

Абсолютний вигреш у часі становив  $\sim 1$ хв, а ефективність алгоритму паралелізації 0.88284. Отже, можна вважати, що обрана методика паралелізації істотно покращує результати навчання при великій кількості ітерацій навчання.

## 4.2 Тест № 2

Для другого тесту обрано такі параметри

- Кількість Ітерацій - 100
- Кількість Прихованих шарів - 1
- Кількість потоків – 4
- Кількість нейронів у шарі - 512

Налаштування		Статистика		Графік		
Параметр	Значення	SingleThread	Mutli Thread	Differense	Relative Differense	
Кількість ітерацій	100					
Кількість прихованих шарів	1	Час 1 декади	39837	29029	10808	37.2317%
Кількість ядер	2	Час 2 декади	39971	20542	19429	94.5818%
Кількість потоків	4	Час 3 декади	41131	28138	12993	46.176%
Час однопоточного навчання(мс)	378840	Час 4 декади	37736	18954	18782	99.0925%
Час багатопоточного навчання	237840	Час 5 декади	37015	27594	9421	34.1415%
Коефіцієнт ефективності	0.796418	Час 6 декади	37820	18866	18954	100.466%
		Час 7 декади	38492	27569	10923	39.6206%
		Час 8 декади	35536	19342	16194	83.7245%
		Час 9 декади	36587	28452	8135	28.592%
		Час 10 декади	34715	19354	15361	79.3686%
		Повний час	378840	237840	141000	59.2836%

Рисунок 16 – Результати тесту №2

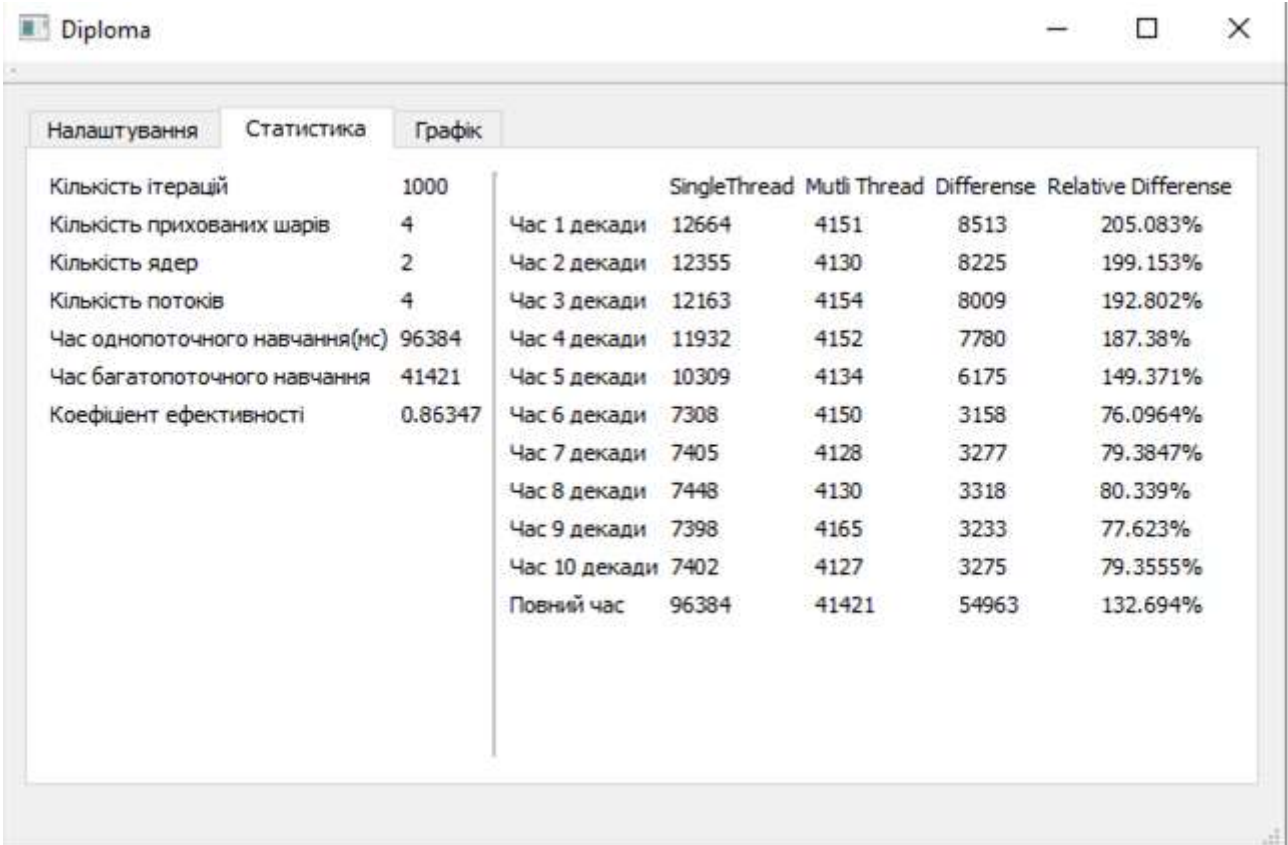
Абсолютний вигреш у часі становив 141000мс, а ефективність алгоритму паралелізації 0.796418. Отже, можна вважати, що обрана методика паралелізації

істотно покращує результати навчання при великій кількості нейронів в шарі нейронної мережі.

### 4.3 Тест № 3

Для даного тесту обрано такі параметри

- Кількість Ітерацій - 1000
- Кількість Прихованих шарів - 4
- Кількість потоків – 4
- Кількість нейронів у шарі - 10 10 10 10



Кількість ітерацій	1000	SingleThread	Mutli Thread	Differense	Relative Differense	
Кількість прихованих шарів	4	Час 1 декади	12664	4151	8513	205.083%
Кількість ядер	2	Час 2 декади	12355	4130	8225	199.153%
Кількість потоків	4	Час 3 декади	12163	4154	8009	192.802%
Час однопоточного навчання(мс)	96384	Час 4 декади	11932	4152	7780	187.38%
Час багатопоточного навчання	41421	Час 5 декади	10309	4134	6175	149.371%
Коефіцієнт ефективності	0.86347	Час 6 декади	7308	4150	3158	76.0964%
		Час 7 декади	7405	4128	3277	79.3847%
		Час 8 декади	7448	4130	3318	80.339%
		Час 9 декади	7398	4165	3233	77.623%
		Час 10 декади	7402	4127	3275	79.3555%
		Повний час	96384	41421	54963	132.694%

Рисунок 17 – Результати тесту №3

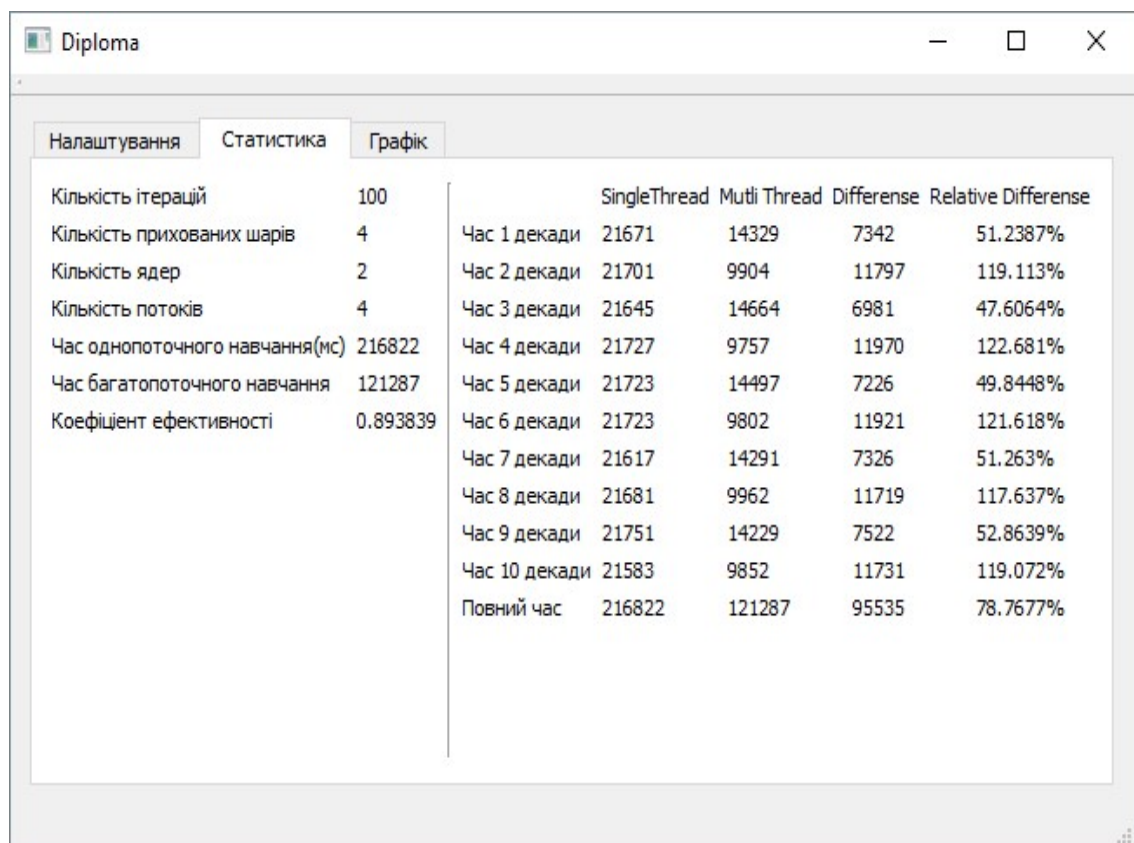
В даному тесті протестована багатошаровість мережі в поєднанні з великою кількістю ітерацій. Коефіцієнт ефективності становив 0.865347, відносний виграш у часі 132%. Також помітно, що незначна кількість нейронів в шарах, майже не впливає на час виконання. Так в порівнянні з Тестом №1 час

лінійного та паралельного навчання майже не змінився, хоча й мережа мала додаткові 3 шари по 10 нейронів у кожному.

#### 4.4 Тест № 4

Для даного тесту обрано такі параметри

- Кількість Ітерацій - 100
- Кількість Прихованих шарів - 4
- Кількість потоків – 4
- Кількість нейронів у шарі - 100 100 100 100



Налаштування		Статистика		Графік			
Кількість ітерацій	100			SingleThread	Multi Thread	Differense	Relative Differense
Кількість прихованих шарів	4	Час 1 декади	21671	14329	7342	51.2387%	
Кількість ядер	2	Час 2 декади	21701	9904	11797	119.113%	
Кількість потоків	4	Час 3 декади	21645	14664	6981	47.6064%	
Час однопоточного навчання(мс)	216822	Час 4 декади	21727	9757	11970	122.681%	
Час багатопоточного навчання	121287	Час 5 декади	21723	14497	7226	49.8448%	
Коефіцієнт ефективності	0.893839	Час 6 декади	21723	9802	11921	121.618%	
		Час 7 декади	21617	14291	7326	51.263%	
		Час 8 декади	21681	9962	11719	117.637%	
		Час 9 декади	21751	14229	7522	52.8639%	
		Час 10 декади	21583	9852	11731	119.072%	
		Повний час	216822	121287	95535	78.7677%	

Рисунок 18 – Результати тесту №4

В даному тесті протестована багатошаровість мережі в поєднанні з великою кількістю нейронів в мережі. Отриманий коефіцієнт ефективності 0.893839 та відносний вигреш у час в 78.77%, що свідчить про ефективність алгоритму на даній конфігурації.

## 4.5 Висновки

Розроблено програмне забезпечення, що відповідає завданню. Алгоритм програми показує високу ефективність у всіх тестових випадках. Середній виграш у часі в ході тестування становив на всіх тестах був більше 80%. Що це раз підтверджує ефективність алгоритму. Також програма використовувала всі можливі ресурси системи для пришвидшення паралельного навчання, про що свідчать показники завантаженості системи під час навчання 23% для лінійного і 94% для паралельного режимів, що свідчить про ефективність обраної технології паралелізації.

## 5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У даному розділі проводиться оцінка основних характеристик програмного продукту, призначеного для аналізу нелінійних нестационарних процесів. Інтерфейс користувача був розроблений за допомогою мови програмування C++ у середовищі розробки QT Creator. Інтерфейс користувача створений за допомогою технології QT.

Програмний продукт призначено для використання на персональних комп'ютерах під управлінням операційної системи Windows.

Нижче наведено аналіз різних варіантів реалізації модулю з метою вибору оптимальної, з огляду при цьому як на економічні фактори, так і на характеристики продукту, що впливають на продуктивність роботи і на його сумісність з апаратним забезпеченням. Для цього було використано апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) – це технологія, яка дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. Як прямі, так і побічні витрати розподіляються по продуктам та послугам у залежності від потрібних на кожному етапі виробництва обсягів ресурсів. Виконані на цих етапах дії у контексті метода ФВА називаються функціями.

Мета ФВА полягає у забезпеченні правильного розподілу ресурсів, виділених на виробництво продукції або надання послуг, на прямі та непрямі витрати. У даному випадку – аналізу функцій програмного продукту й виявлення усіх витрат на реалізацію цих функцій.

Фактично цей метод працює за таким алгоритмом:

– визначається послідовність функцій, необхідних для виробництва продукту. Спочатку – всі можливі, потім вони розподіляються по двом групам: ті, що впливають на вартість продукту і ті, що не впливають. На цьому ж етапі



оптимізується сама послідовність скороченням кроків, що не впливають на цінність і відповідно витрат.

- для кожної функції визначаються повні річні витрати й кількість робочих часів.

- для кожної функції на основі оцінок попереднього пункту визначається кількісна характеристика джерел витрат.

- після того, як для кожної функції будуть визначені їх джерела витрат, проводиться кінцевий розрахунок витрат на виробництво продукту.

## **5.1 Постановка задачі техніко-економічного аналізу**

У роботі застосовується метод ФВА для проведення техніко-економічний аналізу розробки системи аналізу нелінійних нестационарних процесів. Оскільки основні проектні рішення стосуються всієї системи, кожна окрема підсистема має їм задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу гетероскедастичних процесів в економіці та фінансах.

Відповідно цьому варто обирати і систему показників якості програмного продукту.

Технічні вимоги до продукту наступні:

- програмний продукт повинен функціонувати на персональних комп'ютерах із стандартним набором компонент;

- забезпечувати високу швидкість обробки великих об'ємів даних у реальному часі;

- забезпечувати зручність і простоту взаємодії з користувачем або з розробником програмного забезпечення у випадку використання його як модуля;

- передбачати мінімальні витрати на впровадження програмного продукту.

### 5.1.1 Обґрунтування функцій програмного продукту

Головна функція  $F_0$  – розробка програмного продукту, який аналізує процес за вхідними даними та будує його модель для подальшого прогнозування.

Виходячи з конкретної мети, можна виділити наступні основні функції ПП:

$F_1$  – вибір мови програмування;

$F_2$  – алгоритм паралелізації навчання;

$F_3$  – відображення кінцевого результату.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція  $F_1$ :

- а) мова програмування C++;
- б) Пакет Matlab;

Функція  $F_2$ :

- а) створення власного алгоритму
- б) використання готових реалізацій.

Функція  $F_3$ :

- а) відображення тільки кінцевого результату;
- б) виведення всіх етапів навчання.

### 5.1.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 18). На основі цієї карти побудовано позитивно-негативну матрицю варіантів основних функцій (табл. 4).

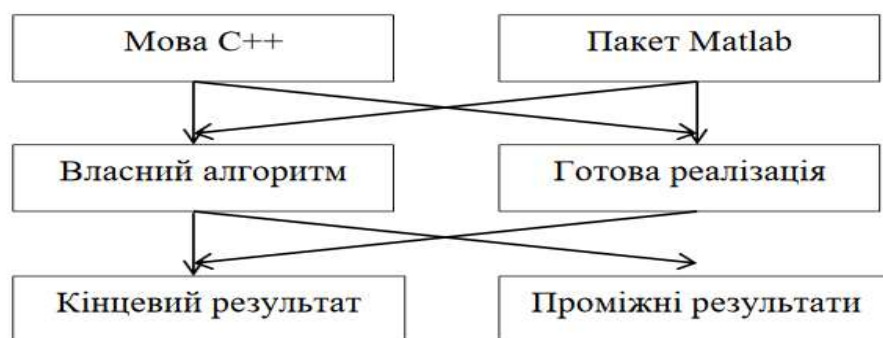


Рисунок 19 – Морфологічна карта

Морфологічна карта відображує всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів ПП.

Таблиця 4 – Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
<i>F1</i>	<i>A</i>	Більша швидкодія Кросплатформений	Займає більше часу при написанні коду
	<i>B</i>	Простота реалізації алгоритмів	Низька швидкодія
<i>F2</i>	<i>A</i>	Унікальність	Більше часу на розробку
	<i>B</i>	Простота реалізації	Не всі методи оптимізовано
<i>F3</i>	<i>A</i>	Оптимізація швидкості відображення Зменшення навантаження на систему	Менша наочність процесу
	<i>B</i>	Більша наочність процесу	Велике навантаження на систему

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

#### Функція *F1*:

Оскільки розрахунки проводяться з великими об'ємами вхідних даних, то час виконання програмного коду є дуже необхідним, тому варіант б) має бути відкинутий.

#### Функція *F2*:

Оскільки планується розглядати широкий спектр різноманітних варіантів даних, та перший є більш зручним в даному випадку, то варіант б) має бути відкинтий

#### Функція F3:

Вважаємо варіанти а) та б) гідними розгляду.

Таким чином, будемо розглядати такі варіанти реалізації ПП:

1. F1a – F2a – F3a
2. F1a – F2a – F3б

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

## **5.2 Обґрунтування системи параметрів ПП**

### **5.2.1 Опис параметрів**

На підставі даних про основні функції, що повинен реалізувати програмний продукт, вимог до нього, визначаються основні параметри виробу, що будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- $X1$  – швидкодія мови програмування;
- $X2$  – об'єм пам'яті для збереження даних;
- $X3$  – час навчання мережі;
- $X4$  – потенційний об'єм програмного коду.

### **5.2.2 Кількісна оцінка параметрів**

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 5.

Таблиця 5 – Основні параметри ПП

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	Оп/мс	19000	11000	2000
Об'єм пам'яті	X2	Мб	32	16	8
Час навчання мережі;	X3	с	800	420	60
Потенційний об'єм програмного коду	X4	кількість строк коду	2000	1500	1000

За даними таблиці 5 будуються графічні характеристики параметрів – рис. 20 – рис. 23.

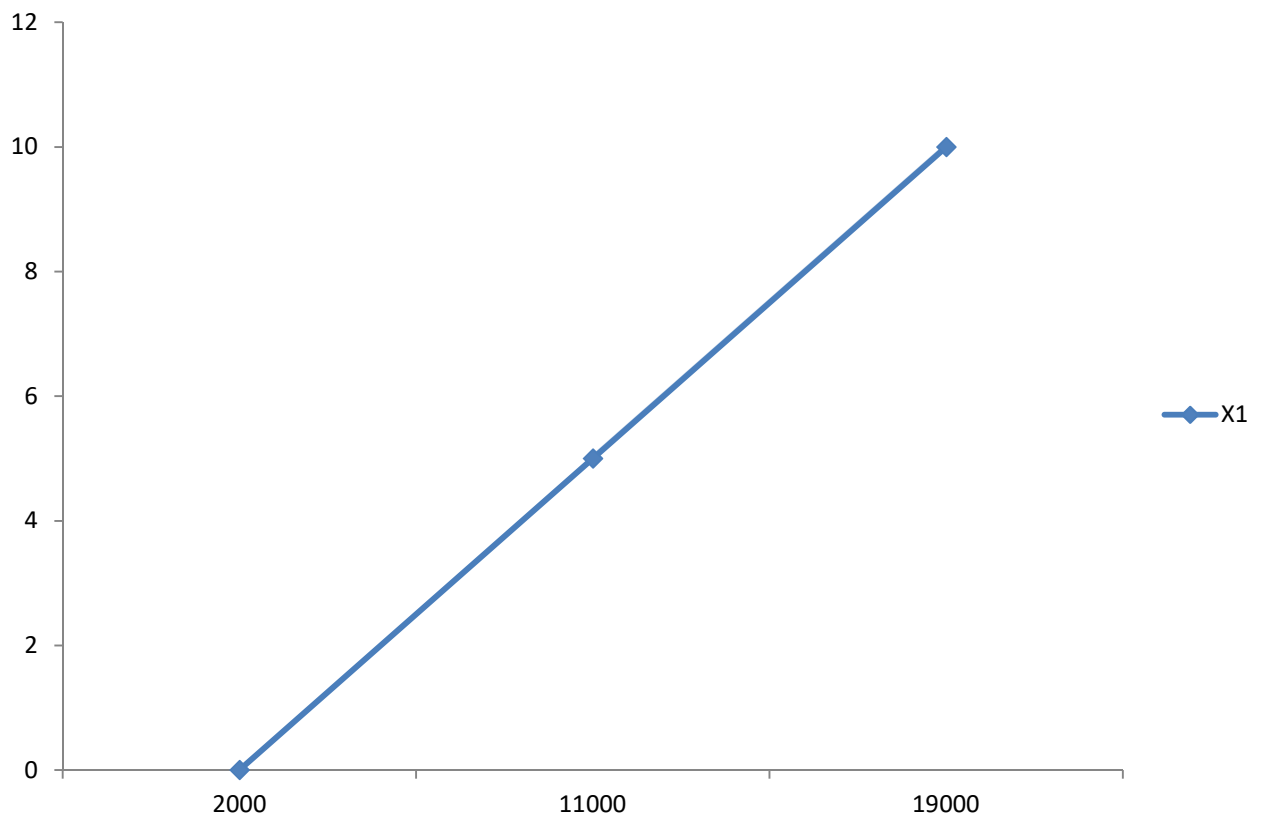


Рисунок 20 – X1, швидкодія мови програмування

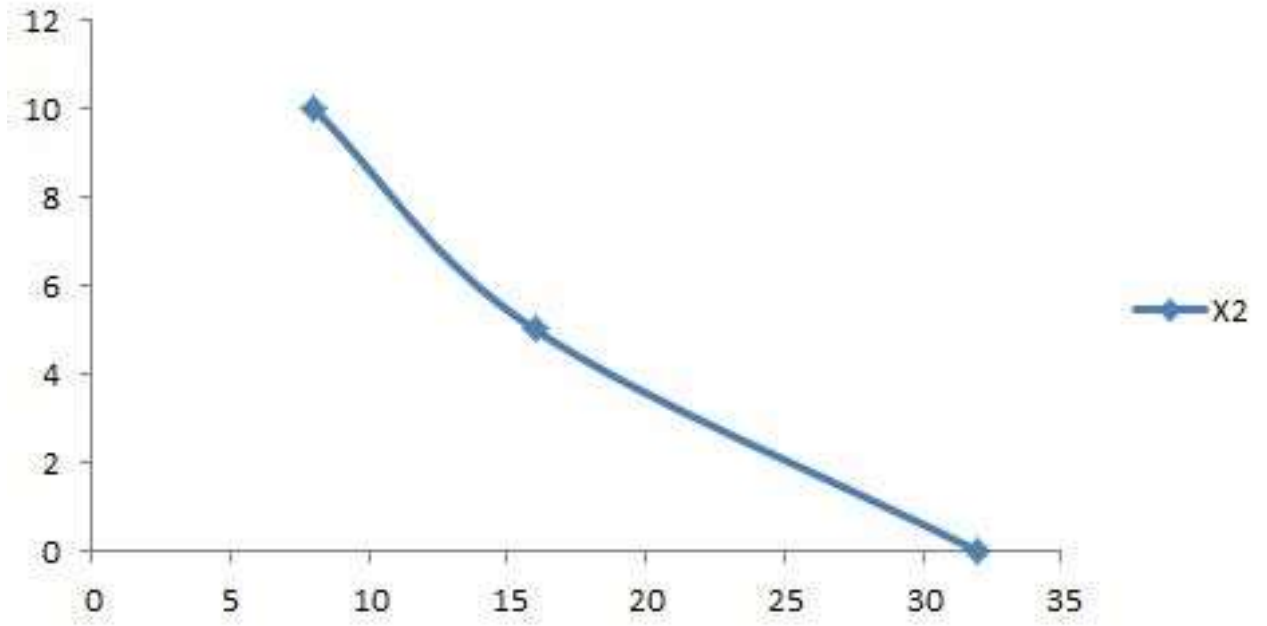


Рисунок 21 – X2, об'єм оперативної пам'яті

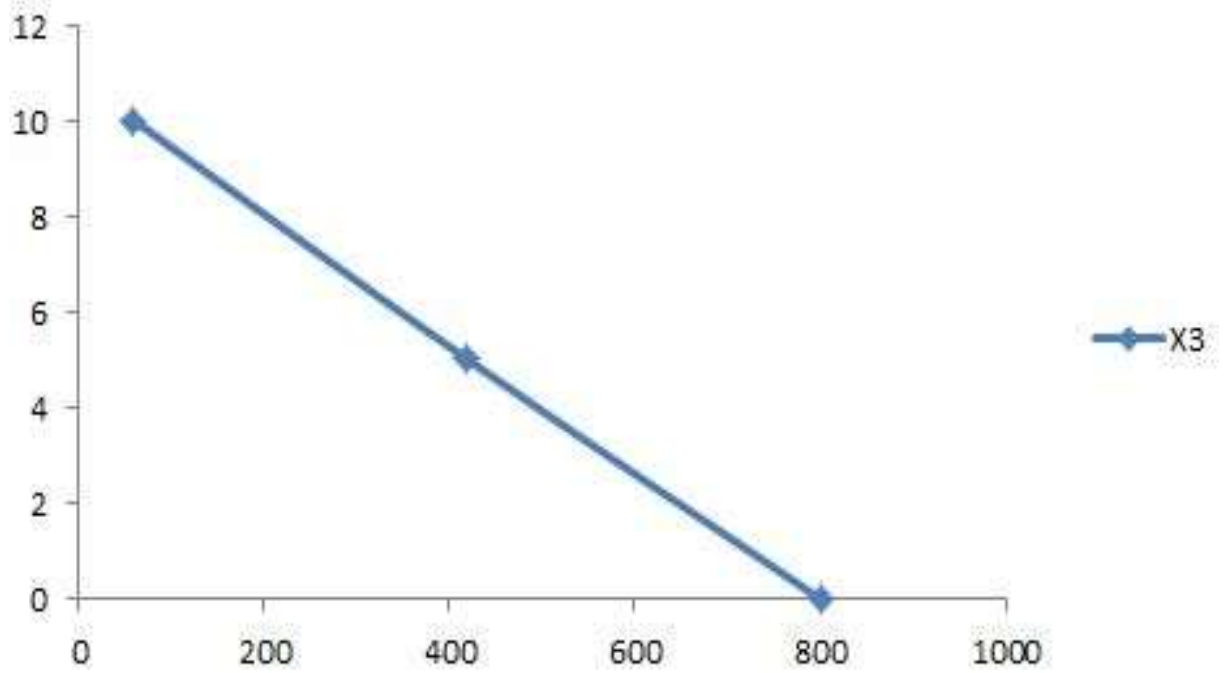


Рисунок 22 – X3, час навчання мережі;

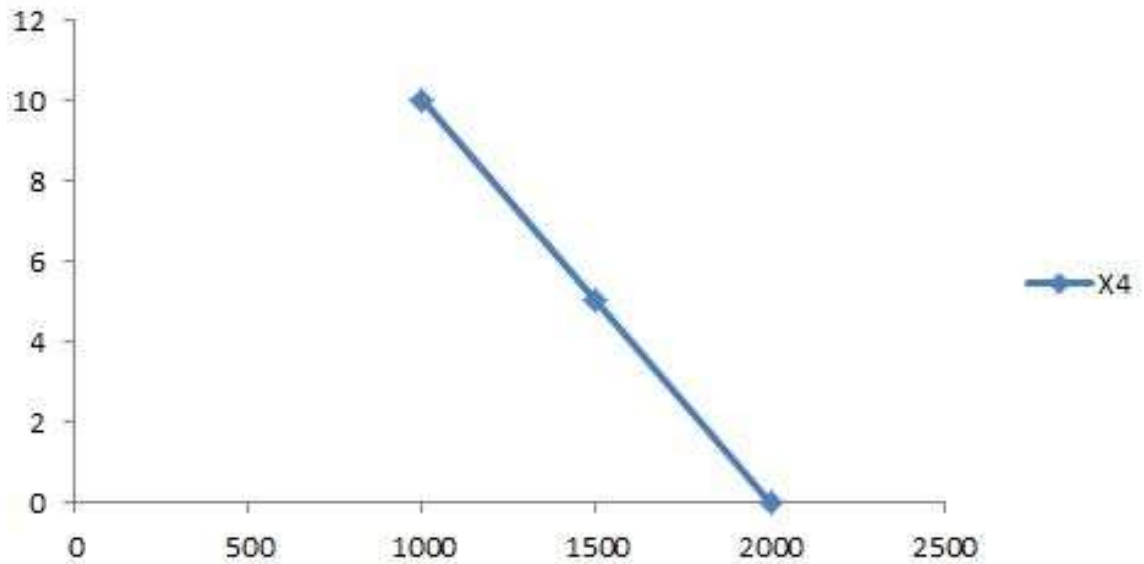


Рисунок 23 – X4, потенційний об'єм програмного коду

### 5.2.3 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 6.

Таблиця 6 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів $R_i$	Відхилення $\Delta_i$	$\Delta_i^2$
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,75	0,56
X2	Об'єм пам'яті	Мб	4	4	4	3	4	3	3	25	-1,25	1,56
X3	Час навчання мережі;	с	2	2	1	2	1	2	2	12	-14,25	203,06
X4	Потенційний об'єм програмного коду	кількість строків коду	5	6	6	6	6	6	6	41	14,75	217,56
	Разом		15	15	15	15	15	15	15	105	0	420,75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105,$$

де  $N$  – число експертів,  $n$  – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26,25.$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T, \quad (10)$$

Сума відхилень по всім параметрам повинна дорівнювати 0;



г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^N \Delta_i^2 = 420,75.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 420,75}{7^2(5^3 - 5)} = 1,03 > W_k = 0,67$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 7.

Таблиця 7 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5
X1 і X4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги  $i$ -го параметра над  $j$ -тим,  $a_{ij}$  визначається по формулі:

$$a_{ij} = \begin{cases} 1,5 & \text{при } X_i > X_j \\ 1,0 & \text{при } X_i = X_j \\ 0,5 & \text{при } X_i < X_j \end{cases} \quad (11)$$

З отриманих числових оцінок переваги складемо матрицю  $A = \| a_{ij} \|$ .

Для кожного параметра зробимо розрахунок вагомості  $K_{ei}$  за наступними формулами:

$$K_{\text{Ві}} = \frac{b_i}{\sum_{i=1}^n b_i}, \quad (12)$$

$$\text{де } b_i = \sum_{i=1}^N a_{ij}.$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятись від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{Ві}} = \frac{b'_i}{\sum_{i=1}^n b'_i} \quad (13)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 8 – Розрахунок вагомості параметрів

Параметри $x_i$	Параметри $x_j$				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	$b_i$	$K_{\text{Ві}}$	$b_i^1$	$K_{\text{Ві}}^1$	$b_i^2$	$K_{\text{Ві}}^2$
X1	1,0	0,5	0,5	1,5	3,5	0,219	22,25	0,216	100	0,215
X2	1,5	1,0	0,5	1,5	4,5	0,281	27,25	0,282	124,25	0,283
X3	1,5	1,5	1,0	1,5	5,5	0,344	34,25	0,347	156	0,348
X4	0,5	0,5	0,5	1,0	2,5	0,156	14,25	0,155	64,75	0,154
Всього:					16	1	98	1	445	1

### 5.3 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів  $X_2$  (об'єм пам'яті для збереження даних) та  $X_1$  (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра  $X_3$  (час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 800 мс або варіанту б) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (табл 9):

$$K_K(j) = \sum_{i=1}^n K_{ei,j} B_{i,j}, \quad (14)$$

де  $n$  – кількість параметрів;

$K_{ei}$  – коефіцієнт вагомості  $i$ -го параметра;

$B_i$  – оцінка  $i$ -го параметра в балах.

Таблиця 9 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	11000	3,6	0,215	0,774
F2(X2)	А	16	3,4	0,283	0,962
F3(X3,X4)	А	800	2,4	0,348	0,835
	Б	80	1	0,154	0,154

За даними з таблиці 9 за формулою

$$K_K = K_{Ty}[F_{1k}] + K_{Ty}[F_{2k}] + \dots + K_{Ty}[F_{zk}], \quad (15)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 0,774 + 0,962 + 0,835 = 2,57$$

$$K_{K2} = 0,774 + 0,962 + 0,154 = 1,89$$

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

#### 5.4 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_0 = T_P \cdot K_{П} \cdot K_{СК} \cdot K_{М} \cdot K_{СТ} \cdot K_{СТ.М}, \quad (16)$$

де  $T_P$  – трудомісткість розробки ПП;

$K_{П}$  – поправочний коефіцієнт;

$K_{СК}$  – коефіцієнт на складність вхідної інформації;

$K_{М}$  – коефіцієнт рівня мови програмування;

$K_{СТ}$  – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$  – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює:  $T_P = 90$  людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання:  $K_{II} = 1.7$ . Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1:  $K_{СК} = 1$ . Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта  $K_{СТ} = 0.8$ . Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 90 \cdot 1.7 \cdot 0.8 = 122.4 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто  $T_P = 27$  людино-днів,  $K_{II} = 0.9$ ,  $K_{СК} = 1$ ,  $K_{СТ} = 0.8$ :

$$T_2 = 27 \cdot 0.9 \cdot 0.8 = 19.44 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (122.4 + 19.44 + 4.8 + 19.44) \cdot 8 = 1328,64 \text{ людино-годин;}$$

$$T_{II} = (122.4 + 19.44 + 6.91 + 19.44) \cdot 8 = 1345.52 \text{ людино-годин;}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 10000 грн., один фінансовий аналітик з окладом 9000 грн. Визначимо зарплату за годину за формулою:

$$C_{ч} = \frac{M}{T_m \cdot t} \text{ грн,}$$

де  $M$  – місячний оклад працівників;

$T_m$  – кількість робочих днів тиждень;

$t$  – кількість робочих годин в день.

(17)

$$C_{\text{ч}} = \frac{20000 + 9000}{3 \cdot 21 \cdot 8} = 57,55 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$C_{\text{зп}} = C_{\text{ч}} \cdot T_i \cdot K_{\text{д}}, \quad (18)$$

де  $C_{\text{ч}}$  – величина погодинної оплати праці програміста;

$T_i$  – трудомісткість відповідного завдання;

$K_{\text{д}}$  – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$\text{I. } C_{\text{зп}} = 57,55 \cdot 1328,64 \cdot 1,2 = 91755,90 \text{ грн.}$$

$$\text{II. } C_{\text{зп}} = 57,55 \cdot 1345,52 \cdot 1,2 = 92921,60 \text{ грн.}$$

Відрахування на єдиний соціальний внесок в залежності від групи професійного ризику (II клас) становить 22%:

$$\text{I. } C_{\text{вд}} = C_{\text{зп}} \cdot 0,22 = 66437,31 \cdot 0,3677 = 20186,30 \text{ грн.}$$

$$\text{II. } C_{\text{вд}} = C_{\text{зп}} \cdot 0,22 = 67281,38 \cdot 0,3677 = 20442,75 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. ( $C_{\text{м}}$ )

Так як одна ЕОМ обслуговує одного програміста з окладом 10000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_{\text{г}} = 12 \cdot M \cdot K_3 = 12 \cdot 10000 \cdot 0,2 = 24000 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{\text{зп}} = C_{\text{г}} \cdot (1 + K_3) = 24000 \cdot (1 + 0,2) = 28800 \text{ грн.}$$

Відрахування на єдиний соціальний внесок:

$$C_{\text{вд}} = C_{\text{зп}} \cdot 0,22 = 28800 \cdot 0,22 = 6336 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$$C_{\text{а}} = K_{\text{тм}} \cdot K_{\text{а}} \cdot C_{\text{пр}} = 1,15 \cdot 0,25 \cdot 10000 = 2875 \text{ грн.,}$$

де  $K_{\text{тм}}$  – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;  $K_{\text{а}}$  – річна норма амортизації;  $C_{\text{пр}}$  – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{TM} \cdot C_{ПР} \cdot K_P = 1.15 \cdot 10000 \cdot 0.05 = 575 \text{ грн.},$$

де  $K_P$  – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$T_{ЕФ} = (D_K - D_B - D_C - D_P) \cdot t_3 \cdot K_B = (365 - 104 - 8 - 16) \cdot 8 \cdot 0.9 = 1706.4 \text{ годин},$$

де  $D_K$  – календарна кількість днів у році;  $D_B$ ,  $D_C$  – відповідно кількість вихідних та святкових днів;  $D_P$  – кількість днів планових ремонтів устаткування;  $t$  – кількість робочих годин в день;  $K_B$  – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{ЕЛ} = T_{ЕФ} \cdot N_C \cdot C_{ЕН} = 1706,4 \cdot 0,156 \cdot 1.94 = 516,42 \text{ грн.},$$

де  $N_C$  – середньо-споживча потужність приладу;  $C_{ЕН}$  – тариф за 1 кВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_H = C_{ПР} \cdot 0.67 = 10000 \cdot 0,67 = 6700 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{ЕКС} = C_{ЗП} + C_{ВІД} + C_A + C_P + C_{ЕЛ} + C_H \quad (19)$$

$$C_{ЕКС} = 28800 + 6336 + 2875 + 575 + 516.42 + 6700 = 45802.42 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{М-Г} = C_{ЕКС} / T_{ЕФ} = 45802.42 / 1706,4 = 26.80 \text{ грн/час.}$$

Оскільки в даному випадку всі роботи, які пов'язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_M = C_{М-Г} \cdot T \quad (20)$$

$$I. \quad C_M = 26.80 \cdot 1328.64 = 35607.55 \text{ грн.};$$

$$II. \quad C_M = 26.80 \cdot 1345.52 = 36059.93 \text{ грн.};$$

Накладні витрати складають 67% від заробітної плати:

$$C_H = C_{ЗП} \cdot 0,67$$

$$I. \quad C_H = 91755.90 \cdot 0.67 = 61476.45 \text{ грн.};$$

$$II. \quad C_H = 92921.60 \cdot 0.67 = 62257.472 \text{ грн.};$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{\text{ПП}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{М}} + C_{\text{Н}} \quad (21)$$

$$\text{I. } C_{\text{ПП}} = 91755.90 + 20186.30 + 35607.55 + 61476.45 = 209026.2 \text{ грн.};$$

$$\text{II. } C_{\text{ПП}} = 92921.60 + 20442.75 + 36059.93 + 62257.472 = 211681.752 \text{ грн.};$$

## 5.5 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР}j} = K_{\text{К}j} / C_{\text{Ф}j}, \quad (22)$$

$$K_{\text{ТЕР}1} = 2.57 / 209026.2 = 0,12 \cdot 10^{-4};$$

$$K_{\text{ТЕР}2} = 1.89 / 211681.75 = 0,89 \cdot 10^{-5};$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня  $K_{\text{ТЕР}1} = 0,12 \cdot 10^{-4}$ .

## 5.6 Висновки

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що



залишилися після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості  $K_{\text{ТЕР}} = 0,12 \cdot 10^{-4}$ .

Цей варіант реалізації програмного продукту має такі параметри:

- мова програмування – C++;
- проектування власного алгоритму;
- результати виводяться тільки після завершення навчання.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, непоганий функціонал і швидкодію.

## ВИСНОВКИ

В результаті виконання роботи була здійснена розробка програми для паралельного навчання нейронної мережі. Дана програма може бути використана для тренування нейронних мереж призначених для класифікації та розпізнавання образів цифр.

Під час розв'язування практичних задач науки і техніки зазвичай виникає проблема ефективної організації обчислень. Деякі з цих задач характеризуються великою розмірністю вхідних даних і тому потребують оброблення значних обсягів інформації. Часто ці задачі вирішуються за рахунок алгоритмів нейронних мереж. Однак головним викликом для дослідників є розробка та оптимізація існуючих алгоритмів найвчання нейронних мереж, адже висока частота зміни інформації та розмірність даних призводять до зростання необхідних для даних обчислень потужностей систем.

Було проаналізовано методи та технології для паралельного навчання нейронних мереж. Розглянуто особливості архітектури мереж та принципи їх тренування. Проаналізовано ефективність архітектур і методів навчання для вирішення конкретних задач.

На основі аналізу розроблено алгоритм та архітектуру програми. Було обрано використання мови високого рівня C++ разом з бібліотекою QT. Для паралелізації алгоритму навчання багатошарового перцептронну на рівні вибірки використовувався відкритий стандарт для розпаралелювання програм на C++ - OpenMP.

Тестування програми відбувалося на 4 тестових прикладах. Алгоритм програми показав високу ефективність у всіх тестових випадках. Середній вигреш у часі в ході тестування становив більше 80%. Що це раз підтверджує ефективність алгоритму. Також програма використовувала всі можливі ресурси системи для пришвидшення паралельного навчання, про що свідчать показники завантаженості системи під час навчання 23% для лінійного і 94% для

паралельного режимів, що також свідчить про ефективність обраної технології паралелізації.

Отже, розвиток і тенденції в області паралельного навчання нейронних мережі не дарма все частіше знаходять застосування і розвиток в багатьох сферах. Так, як призводять до суттєвого скорочення часу процесу навчання, та, в залежності від системи на якій проводиться навчання, можуть показувати близьке до лінійного зростання ефективності в залежності від кількості процесорів, ядер, або потоків системи.

## ПЕРЕЛІК ПОСИЛАНЬ

1. Воеводин В . Параллельные вычисления / Воеводин В .В., Воеводин Вл . В . – СПб .: БХВ – Петер - бург , 2002. – 608 с
2. Simon Haykin . Neural Network a comprehensive foundation(2nd edition) / Simon Haykin - Prentice Hall, 842 pages, 2013
3. Борисов Ю.И. Нейросетевые методы обработки информации и средства их программно-аппаратной поддержки / Борисов Ю.И, Кашкаров В.М., Сорокин С.А. // Открытые системы. – 1997.– № 4. – С. 38 – 40.
4. Горбань А.Н., Россиев Д.А. Нейронные сети на персональном компьютере. / А.Н. Горбань, Д.А. Россиев – Н.: Наука, 2006. – 276с.
5. Горбань А.Н. Нейронные сети на персональном компьютере / А.Н. Горбань, Д.А. Россиев. – Новосибирск: Наука, 2006. – 230 с.
6. Комашинский В.И. Смирнов Д.А. Внедрение в нейро-информационные технологии. / В.И. Комашинский, Д.А. Смирнов - СПб., 1999.
7. Hong S.G., Kim S.W. and Lee J.J., 2015. The Minimum Cost Path Finding Algorithm Using a Hopfield Type Neural Network, Proceedings IEEE International Conference on Fuzzy Systems 4, 719–726.
8. Лисе А.А., Степанов М.В. Нейронные сети и нейрокомпьютеры. / А.А. Лисе, М.В. Степанов // Учеб. пособие. ГЭТУ. - СПб., 2009. 64 с.
9. Тархов Д.А. Нейронные сети. Модели и алгоритмы. – М.: Радиотехника, 2010. – 82 с.
10. Царегородцев В.Г. Перспективы распараллеливания программ нейросетевого анализа и обработки данных / В.Г. Царегородцев // Материалы III Всерос. конф. «Математика, информатика, управление – 2008». – Иркутск, 2014. – С. 110-117.
11. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. - М. Горячая линия-Телеком 2004. – 112 с.

12. Галушкин А.И. Теория нейронных сетей. / А.И. Галушкин - М.: ИПРЖР, 2010. – 348 с.
13. Боголюбов Д. П., Чанкин А. А., Стемиковская К. В. Реализация алгоритма обучения самоорганизующихся карт Кохонена на графических процессорах // Промышленные АСУ и контроллеры. 2012. № 10. С. 30-35
14. Asanovic, Krste et al. (Jan 18, 2016). The Landscape of Parallel Computing Research: A View from Berkeley University of California, Berkeley. Technical Report No. UCB/EECS-2016-183
15. Barbara Chapman, Gabriele Jost, Ruud van der Pas. Using OpenMP: portable shared memory parallel programming (Scientific and Engineering Computation). Cambridge, Massachusetts: The MIT Press., 2008. - 353 pp.
16. Нейромережеве відображення дійсності. – Режим доступу:  
[http://studies.in.ua/mpd\\_seminar/1313-neurmerezh.html](http://studies.in.ua/mpd_seminar/1313-neurmerezh.html) – Дата доступу:  
29.05.2017
17. Моделі нейронних мереж. – Режим доступу:  
<https://studme.com.ua/1246122010028/neural/models.htm> – Дата доступу:  
27.04.2017
18. Моделі нейронних мереж. – Режим доступу:  
<http://techn.sstu.ru/kafedri/подразделения/1/MetMat/Terin/neiro/neiro.htm> –  
Дата доступу: 28.05.2017