НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2020р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

зі спеціальності 122 Комп’ютерні науки та інформаційні технології
за спеціалізацією Геометричне моделювання в інформаційних системах
на тему Прогнозування стану складного технічного об’єкта з допомогою нейронної мережі

Виконав (-ла): студент (-ка) 4 курсу, групи ТР-62

Щур Олександр Любимович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник к.т.н., доцент Лабжинський В.А. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент доцент, к.т.н., с.н.с. Мариненко В.І. _____
(посада, 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 – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

зі спеціальності 122 Комп’ютерні науки та інформаційні технології
за спеціалізацією Геометричне моделювання в інформаційних системах

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020р.

ЗАВДАННЯ
на дипломну роботу студенту

Щуру Олександру Любимовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Прогнозування стану складного технічного об’єкта з допомогою нейронної мережі

керівник роботи к.т.н., доцент Лабжинський В.А.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 201__р. № __

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи Архітектура нейронних мереж, методи навчання нейронних мереж, навчальна вибірка, мова програмування Python.

4.Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Провести аналіз наявних архітектур та методів навчання нейронних мереж, визначити оптимальну модель нейронної мережі, інтегрувати нейронну мережу в систему прогнозування стану складного технічного об’єкта, провести тестування створеної системи.

5. Перелік ілюстративного матеріалу
_Презентація MS PowerPoint_____
_____
_____
_____
_____

6. Консультанти розділів роботи

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

7. Дата видачі завдання ”___”_____201___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	***Затвердження теми роботи***		
2.	Вивчення та аналіз задачі		
3.	Розробка архітектури та загальної структури системи		
4.	Розробка структур окремих підсистем		
5.	Програмна реалізація системи		
6.	Оформлення пояснювальної записки		
7.	***Захист програмного продукту***		
8.	***Передзахист***		
9.	Захист		

Студент _____ _____Щур О.Л._____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ ___Лабжинський В.А._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Дипломна робота присвячена розробці системи прогнозування стану складного технічного об’єкта з допомогою нейронної мережі. У роботі наведено перелік архітектури та методів навчання нейронних мереж.

Розроблено алгоритм та архітектуру програми. Реалізовано систему для прогнозування стану складного технічного об’єкта на основі роботи нейронної мережі з використанням навчальної вибірки згенерованих числових рядів.

Проведено тестування створеної системи, надана покрокова інструкція з експлуатації.

Загальний об’єм роботи 71 сторінок, 38 рисунків, 1 таблиця, 7 бібліографічних найменувань.

Ключові слова: штучна нейронна мережа, нейрон, синапс, функція активації, метод зворотного поширення помилки, градієнтний спуск, критерій Акаіке.

ABSTRACT

Thesis is devoted to the development of a system for predicting the state of a complex technical object using a neural network. The paper presents a list of architecture and methods of learning neural networks.

The algorithm and architecture of the program were developed. The system for predicting the state of a complex technical object based on the operation of a neural network using a training sample of generated numerical series has been implemented.

The created system was tested, step-by-step operating instructions were provided.

Total volume of work 71 pages, 38 figures, 1 table, 7 bibliographic names.

Keywords: artificial neural network, neuron, synapse, activation function, backpropagation error method, gradient descent, Akaike critetion.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ **7**
ВСТУП **8**
1. ПОСТАНОВКА ЗАДАЧІ **10**
2. АРХІТЕКТУРА ТА МЕТОДИ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ . **11**
2.1 Штучний нейрон 11
2.2 Класифікація нейронних мереж 12
2.3 Методи навчання нейронних мереж 16
3. ПРОЕКТУВАННЯ НЕЙРОННОЇ МЕРЕЖІ **20**
3.1 Багатошаровий персептрон 20
3.2 Функції активації 21
3.3 Методи ідентифікації аварій на основі даних, що надходять 22
3.4 Метод зворотного поширення помилки 24
3.5 Градієнтний спуск 25
3.6 Критерій Акаіке 26
3.7 Модель спроектованої нейронної мережі 28
3.8 Конфігурація нейронної мережі 29
4. РОЗРОБКА СИСТЕМИ ПРОГНОЗУВАННЯ СТАНУ СКЛАДНОГО ТЕХНІЧНОГО ОБ'ЄКТА **31**
4.1 Засоби розробки 31
4.2 Об'єктно-орієнтоване представлення 32
4.3 Варіанти використання 33
5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ **35**
6. ТЕСТУВАННЯ СИСТЕМИ **44**
ВИСНОВКИ **49**
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **50**
ДОДАТОК А **51**
ДОДАТОК Б **53**
ДОДАТОК В **60**

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШНМ – штучна нейронна мережа.

МЗПП – метод зворотного поширення помилки.

ФА – функція активації.

ПШ – прихований (проміжний) шар.

ВхШ – вхідний шар.

ВдШ – вихідний шар.

ПЗ – програмний засіб.

ГІК – графічний інтерфейс користувача.

ВСТУП

При інноваційному сценарії розвитку економіки в найближче десятиліття нас очікує якісний стрибок в напрямку інтелектуалізації усіх сфер людської діяльності [1].

В процесі функціонування складного технічного об’єкта відбувається взаємодія його різноманітних компонентів і фізичних процесів, що зумовлює складну поведінку як окремих елементів об’єкта так і всієї системи при нормальній експлуатації і, особливо, в аварійних ситуаціях.

Метою розробки системи прогнозування стану складного технічного об’єкта з допомогою нейронної мережі є інтелектуалізація та автоматизація процесів, які відбуваються під час експлуатації складного технічного об’єкта.

Штучні нейронні мережі – математичні моделі, а також програмні або апаратні реалізації, побудовані по принципу організації і функціонування біологічних нейронних мереж – мереж нервових клітин живого організму. Нейронні мережі не програмуються, а навчаються по вхідному наборі даних. Можливість навчання – одна із головних переваг нейронних мереж перед традиційними алгоритмами. ШНМ застосовуються для вирішення задач класифікації та розпізнавання образів, оптимізації, прогнозування та автоматизації.

1. ПОСТАНОВКА ЗАДАЧІ

Метою дипломної роботи є розробка системи прогнозування стану складного технічного об’єкта з допомогою нейронної мережі.

Система призначена для інтелектуалізації та автоматизації процесів складного технічного об’єкта. Потенційними користувачами програмного комплексу є обслуговуючий персонал технічного об’єкта.

Задачею, яка розв’язується програмним забезпеченням є прогнозування стану складного технічного об’єкта. Цей процес відбувається за допомогою роботи штучної нейронної мережі.

Завдання, що вирішуються в роботі:

1. Проаналізувати наявні архітектури та методи навчання нейронних мереж.
2. Визначити оптимальну модель нейронної мережі.
3. Обрати програмні та апаратні засоби для розробки програмного продукту.
4. Провести тестування створеної системи.

Ефективність обраної архітектури та методу навчання визначаються за допомогою критерію Акаіке для моделей, які використовують метод найменших квадратів.

Визначення оптимальної моделі ШНМ полягає у виборі:

- Кількості вхідних параметрів
- Кількості прихованих шарів
- Кількості нейронів у прихованих шарах
- Кількості вихідних нейронів
- Функції активації
- Кількості ітерацій навчання
- Коефіцієнта швидкості навчання

Для функціонування нейронної мережі на вхід подаються масиви

даних, а саме набір параметрів складного технічного об’єкта та його поточний стан у віжповідний момент часу. Під час процесу навчання штучної нейронної мережі будуються залежності між набором параметрів та вихідним станом об’єкта.

Основним алгоритмом, який будує залежності, є алгоритм зворотного поширення похибки.

Ефективність системи зростає із ростом кількості та різноманітності масивів даних.

2. АРХІТЕКТУРА ТА МЕТОДИ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

Штучна нейронна мережа ґрунтується на сукупності з'єднаних вузлів, що називають штучними нейронами. Кожне з'єднання штучними нейронами може передавати сигнал від одного до іншого. Штучний нейрон, що отримує сигнал, може обробляти його, а потім сигналізувати штучним нейронам, приєднаним до нього.

2.1 Штучний нейрон

Прототипом для створення нейрона став біологічний нейрон головного мозку. Біологічний нейрон має тіло, сукупність відростків – дендритів, за якими в нейрони надходять вхідні сигнали, і аксони, що передають вихідні сигнали нейронів іншим клітинам (рисунок 2.1). Точка з'єднання дендрита і аксона називається синапсом [2].

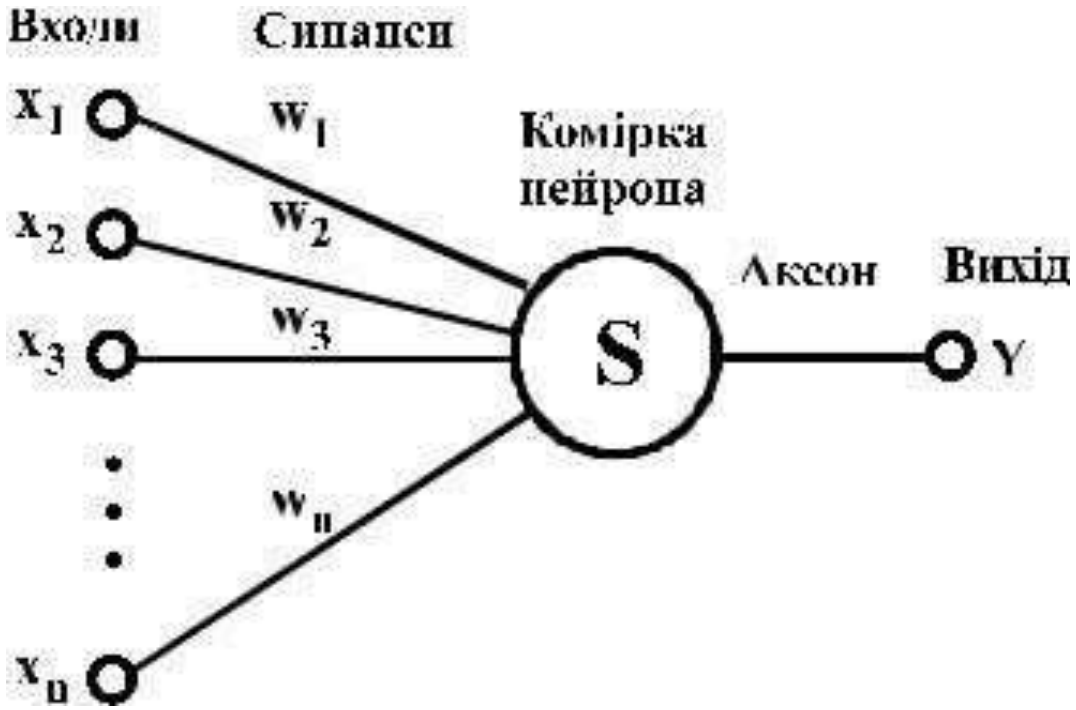

Рисунок 2.1 – Штучний нейрон

Штучний нейрон є вузлом штучної нейронної мережі, він є спрощеною моделлю природнього нейрона. Математично штучний нейрон можна представити як деяку нелінійну функцію від єдиного аргументу – лінійної комбінації всіх вхідних сигналів. Дану функцію називають функцією активації. Отриманий результат надходить на єдиний вихід.

Такі штучні нейрони об'єднують в мережі – з'єднують виходи одних нейронів з входами інших. Штучні нейрони і мережі є основними елементами ідеального нейрокомп'ютера [3].

Зв'язки, по яким вихідні сигнали одних нейронів поступають на входи інших називають синапсами по аналогії до зв'язків між біологічними нейронами. Кожен зв'язок характеризується своєю вагою. Зв'язки з додатною вагою називаються збуджуючими, а з від'ємною – гальмуючими [4]. Нейрон має один вихід, який називають аксоном аналогічно до біологічного прототипу. З єдиного виходу нейрона сигнал може надходити на довільне число входів інших нейронів.

В основному, нейрони класифікують на основі їх положення в топології мережі. Розглядають так види нейронів:

- Вхідні – приймають вхідний вектор, який кодує сигнал. Як правило, ці нейрони не виконують обчислювальних операцій, а просто передають отриманий вхідний сигнал на вихід, можливо, посиливши його або послабивши.
- Вихідні – являють собою виходи мережі. У вихідних нейронах можуть проводитись будь-які обчислювальні операції.
- Проміжні нейрони – виконують основні обчислювальні операції [5].

2.2 Класифікація нейронних мереж

Штучні нейронні мережі надзвичайно різноманітні за конфігураціями. Незважаючи на це, мережеві парадигми мають багато спільного. Нейронні

мережі розрізняють за топологічними типами відповідно до структури зв’язків між нейронами мережі, а також за типом використаних формальних нейронів. ШНМ можуть розглядатися як спрямований граф зі зваженими зв'язками, у якому штучні нейрони є вузлами. По архітектурі зв'язків ШНМ можуть бути згруповані в два класи: мережі прямого поширення, у яких графи не мають петель, і рекурентні мережі, або мережі зі зворотними зв'язками (рисунок 2.2).

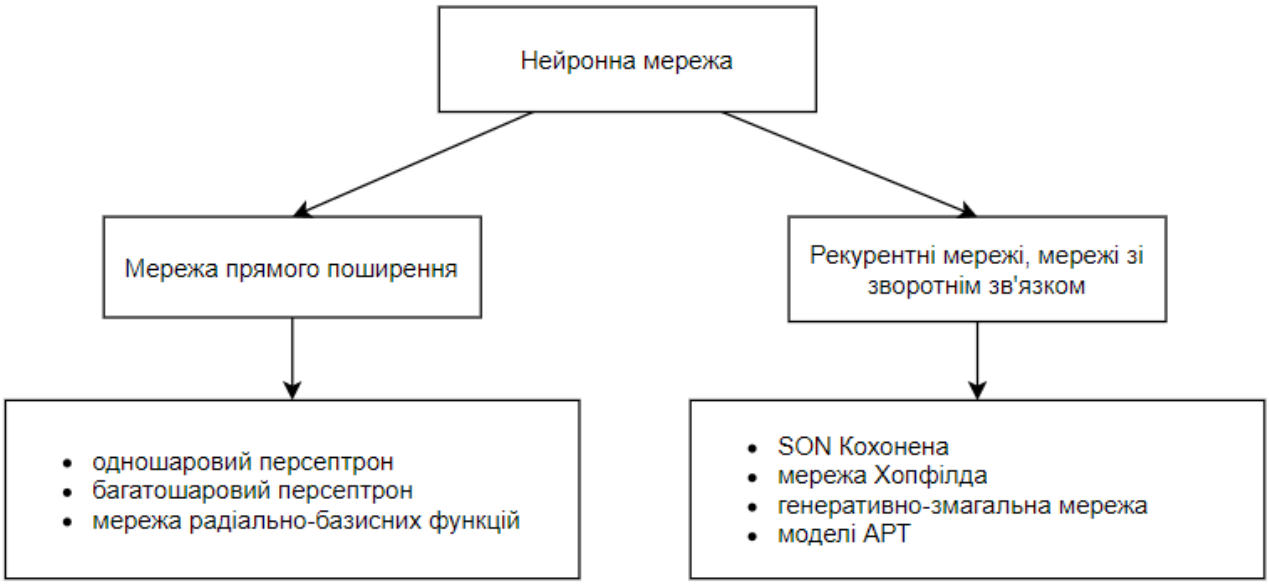

Рисунок 2.2 – Класифікація нейронних мереж

За організацією навчання поділяють навчання нейронних мереж з вчителем та без вчителя.

За способом навчання поділяють навчання за входами і за виходами. При навчанні за входами навчальний приклад є тільки вектором вхідних сигналів, а при навчанні за виходами до нього входить і вектор вихідних сигналів, який відповідає вхідному вектору.

За способом подання прикладів розрізняють подання одиночних прикладів і "сторінки" прикладів. У першому випадку зміна стану нейронної мережі (навчання) відбувається після подання кожного прикладу. У другому – після подання множини прикладів на підставі аналізу одразу усіх їх.

За способом подачі інформації до входів нейронної мережі розрізняють:

- Подачу сигналів на синапси вхідних нейронів.
- Подачу сигналів до виходів вхідних нейронів.
- Подачу сигналів у вигляді ваги синапсів вхідних нейронів.
- Адитивну подачу на синапси вхідних нейронів.

За способом зняття інформації з виходів нейронної мережі розрізняють:

- Зняття з виходів вихідних нейронів.
- Зняття з синапсів вихідних нейронів.
- Зняття у вигляді значень ваги синапсів вихідних нейронів.
- Адитивне зняття із синапсів вихідних нейронів.

За особливостями моделі нейрону розрізняють нейрони з різними нелінійними функціями.

По структурі зв’язків нейронні мережі поділяють:

- Повнозв’язні нейронні мережі, в яких кожен нейрон передає свій вихідний сигнал решті нейронів, в тому числі і самому собі. Всі вхідні сигнали подаються всім нейронам. Вихідними сигналами мережі можуть бути всі або деякі вихідні сигнали нейронів після декількох тактів функціонування мережі.
- Неповнозв’язні нейронні мережі, які також описуються неповнозв’язним орієнтованим графом і зазвичай називаються персептронами, поділяють на одношарові і багатошарові, з прямими, перехресними і зворотними зв’язками. В нейронних мережах з прямими зв’язками нейрони *j*-го шару на входах можуть з’єднуватись лише з нейронами нижніх шарів. В нейронний мережах з перехресними зв’язками допускаються зв’язки всередині одного шару.

В свою чергу, серед багатьох багатошарових ШНМ виділяють наступні типи:

- Монотонні. Це частковий випадок мереж з шарами з додатковими

умовами на зв’язки і нейрони. Кожен шар крім останнього поділений на два блоки: збуджуючий і гальмуючий. Зв’язки між блоками теж розділяють на гальмуючі і збуджуючі. Якщо від нейронів блоку А до нейронів блоку В ведуть лише збуджуючі зв’язки, то це означає що любий вихідний сигнал блоку В є монотонною не спадаючою функцією любого вихідного сигналу з блоку А. Якщо ці зв’язки є гальмуючими, то любий вихідний сигнал блоку В є не зростаючою функцією любого вихідного сигналу блоку А. Для нейронів монотонних мереж необхідна монотонна залежність вихідного сигналу нейрона від параметрів вхідних сигналів.

- Мережі без зворотних зв’язків. В таких ШНМ нейрони вхідного шару отримують вхідні сигнали, перетворюють їх і передають нейронам першого прихованого шару, і так далі до вихідного, який видає сигнали для інтерпретатора і користувача.
- Мережі зі зворотними зв’язками. В таких ШНМ зі зворотними зв’язками інформація з наступних шарів передається на попередні. Серед них, в свою чергу, виділяють наступні:
    - Шарувато-циклічні, які відрізняються тим, що шари замкнуті в кільце: останній шар передає свої вихідні сигнали першому, всі шари рівноправні і можуть як отримувати вхідні сигнали, так і видавати вихідні.
    - Шарувато-повнозв’язні складаються з шарів, кожен з яких представляє з себе повнозв’язну мережу, а сигнали передаються як і від шару до шару, так і всередині шару. В кожному шарі цикл роботи розділяється на три частини: прийом сигналів з попереднього шару, обмін сигналами всередині шару, створення і передача вихідного сигналу до наступного шару.
    - Повнозв’язно-шаруваті, по своїй структурі аналогічно

шарувато-повнозв’язним, але функціонують по-іншому: в них не розділяються фази обміну всередині шару і передача наступному шару. На кожному такті нейрони всіх шарів приймають сигнали від нейронів як і свого шару так, так і наступних шарів.

2.3 Методи навчання нейронних мереж

Розглянемо поняття і принципи процесу навчання нейронної мережі. Процес навчання ШНМ – це процес налаштування архітектури мережі (структури зв’язків між нейронами) і ваги синаптичних зв’язків для ефективного вирішення поставленої задачі.

Навчання нейронної мережі здійснюється на основі навчальної вибірки даних по певному алгоритму.

Виділяють три парадигми навчання ШНМ:

- Навчання з учителем
- Навчання без вчителя
- Змішане навчання

При навчанні мережі з учителем відомі правильні відповіді до кожного вхідного прикладу даних, а ваги підлаштовуються, щоб мінімізувати помилку.

Серед більшості алгоритмів навчання з учителем найбільш успішним є алгоритм зворотного поширення помилки. Основна ідея алгоритму полягає у тому, що зміна ваги синапсів відбувається з урахуванням локального градієнту функції помилки. Різниця між реальними і правильними відповідями нейронної мережі, які визначаються на вихідному шарі ШНМ, поширюється в зворотному напрямку – назустріч потоку сигналів. В результаті кожен нейрон має здатність визначити вклад кожної своєї ваги в сумарну помилку мережі. Найпростіше правило навчання відповідає методу покоординатного спуску, тобто зміні синаптичнх ваг пропорційно їх вкладу в

загальну помилку.

Серед недоліків алгоритму виділяють невизначено довгий процес навчання, який може виникнути у випадках:

- Параліч мережі: в процесі навчання нейронної мережі значення ваг можуть стати дуже високими в результаті корекції. Більшість нейронів будуть функціонувати в області де похідна стискаючої функції має дуже мале значення.
- Локальні мінімуми: мережа може попасти в локальний мінімум, коли поблизу знаходиться більш глибокий мінімум.

Існує також можливість перенавчання мережі. Весь набір образів, наданих для навчання, буде вивчений мережею, проте любі інші образи, навіть які є схожими, можуть бути класифіковані невірно.

Існують сучасні алгоритми другого порядку, такі як метод спряжених градієнтів і метод Левенберга-Марквардта, які на деяких задачах працюють бистріше. Розроблені також евристичні модифікації алгоритму, які добре працюють для певних класів задач:

- Швидке поширення.
- Дельта-дельта з рисою.

Процес навчання з вчителем, як і у випадку з навчанням з вчителем, полягає у налаштуванні ваги синапсів. Деякі алгоритми можуть змінювати структуру мережі, тобто кількість нейронів та їх взаємозв’язки, що називається самоорганізацією.

Налаштування синапсів може проводитися лише на основі інформації, яка доступна в нейроні, тобто його стану і поточних вагових коефіцієнтів. Виходячи з цього побудовані алгоритми навчання Хебба. Алгоритм наступній:

1. На стадії ініціалізації всім ваговим коефіцієнтам присвоюються невеликі випадкові значення
2. На вхід мережі подається вхідний образ і сигнали збудження поширюються по всім шарам згідно принципів класичний мереж

прямого поширення, тобто для кожного нейрона обчислюється зважена сума його входів, до яких потім застосовується активаційна функція нейрона.

3. Відбувається корекція вагових синапсів.
4. Цикл з другого кроку продовжується допоки вихідні значення мережі не стабілізуються із заданою точністю. Застосування цього способу визначення закінчення навчання обумовлено тим, що значення синапсів, які налаштовуються, фактично необмежені.

Варто зазначити, що вид відкликів на кожен клас вхідних образів попередньо невідомий і буде являти собою довільне поєднання станів нейронів вихідного шару, обумовлене випадковим розподілом ваг на стадії ініціалізації. До того ж, мережа знатна узагальнювати подібні образи, відносячи їх до одного класу. Тестування навченої мережі дозволяє визначити топологію класів у вихідному шарі. Для приведення відкликів навченої мережі до зручного представлення можна доповнити мережу одним шаром, який, доприкладу, по алгоритму одношарового персептрона необхідно змусити відображати вихідні реакції мережі у необхідні образи [6].

Іншим алгоритмом навчання ШНМ без вчителя є алгоритм Кохонена. Передбачає налаштування синапсів на основі їх значень від попередніх ітерацій.

Алгоритм навчання має приблизно так ж структуру як і в Хебба, проте на третьому кроці із всього шару обирається нейрон, значення синапсів якого максимально схожі на вхідний образ і налаштування ваг відбувається лише для нього. Цей вибір може супроводжуватись гальмуванням решти нейронів шару і введенням обраного нейрона в насичення. Вибір такого нейрона може здійснюватись, наприклад, розрахунком скалярного добутку вагових коефіцієнтів з вектором вхідних значень. Максимальний добуток дає виграшний нейрон.

При використанні навчання по алгоритму Кохонена існує практика нормалізації вхідних образів, а також, на стадії ініціалізації і нормалізації

початкових значень вагових коефіцієнтів. Ініціалізація вагових коефіцієнтів випадковими значеннями може призвести до того, що різноманітні класи, по яким розподілені вхідні образи, поєднаються або навпаки розділяться на додаткові підкласи у випадку схожих образів одного класу.

Для уникнення такої ситуації використовується метод випуклої комбінації. Суть методу зводиться до того, що вхідні нормалізовані образи піддаються перетворенню. На основі цього методу будуються нейронні мережі особливого типу – так звані структури, які здатні самоорганізовуватися.

3. ПРОЕКТУВАННЯ НЕЙРОННОЇ МЕРЕЖІ

Для реалізації системи прогнозування стану складного технічного об’єкту було обрано нейронну мережу прямого поширення – багатошаровий персептрон.

3.1 Багатошаровий персептрон

Багатошаровий персептрон – нейронна мережа прямого поширення сигналу, в якій сигнал перетворюється у вихідний, проходячи послідовно через декілька шарів.

Перший шар називають вхідним, а останній – вихідним. Окрім цим шарів в багатошаровому персептроні є один чи декілька проміжних шарів, які називають прихованими (рисунок 3.1) [5].

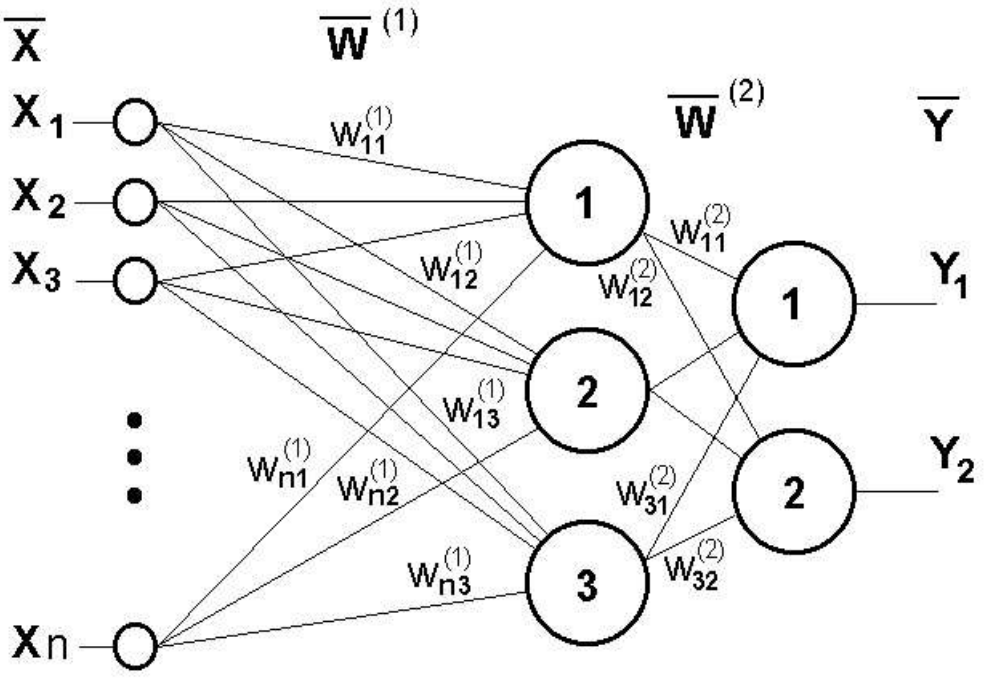

Рисунок 3.1 – Багатошаровий персептрон

Мережа зображена на рисунку 3.1 має *n*-входів. На них поступають

сигнали, які йдуть далі по синапсам на три нейрона першого прихованого шару.

Вихідні сигнали першого шару передаються двом нейронам другого шару. Останні, в свою чергу, дають на виході два вихідних сигнали.

$$u = \sum_{i=1}^{n} x_i w_i \quad (3.1).$$

Вихідний сигнал нейрона формується наступнім чином: обчислюється сума добутків усіх вхідних сигналів на вагові коефіцієнти (3.1), яка далі проходить через функцію активації.

3.2 Функції активації

Існує декілька типів ФА (рисунок 3.2 – 3.5):

- Лінійна:

$$f(x) = cx \quad (3.2).$$

- Сигмоїда (рисунок 3.2):

$$f(u) = \frac{1}{1+e^{-u}} \quad (3.3).$$

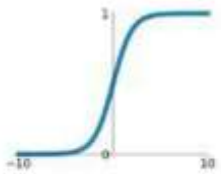

Рисунок 3.2 – Сигмоїда

- Гіперболічний тангенс (рисунок 3.3):

$$f(x) = tg(x) = \frac{2}{1+e^{-2x}} - 1 \quad (3.4).$$

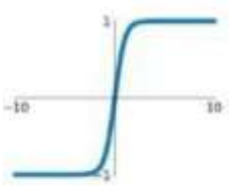

Рисунок 3.3 – Гіперболічний тангенс

- ReLu (рисунок 3.4):

$$f(x) = \max(0, x) \quad (3.5).$$

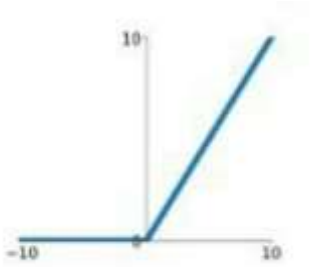

Рисунок 3.4 – ReLu

Для реалізації алгоритму нейронної мережі для системи прогнозування стану складного технічного об’єкту було застосовано функцію активації сигмоїду (3.3), оскільки вона добре підходить під задачі класифікації, так як апроксимацію класифікуючої функції комбінацією сигмоїд провести легше, ніж використовуючи інші види активаційних функцій.

3.3 Методи ідентифікації аварій на основі даних, що надходять

Методи ідентифікації аварій на основі даних, що надходять використовують багатопараметричні статистичні методи та інструментарій машинного навчання для ідентифікації збоїв у роботі технічних об’єктів. Дані методи також опираються на відношення між корельованими вимірами всередині системи, але й використовують їх неявно шляхом аналізу даних для навчання під час нормального функціонування об’єкта. Такі методи також називаються методами на основі історії процесу. Схема методу представлена на рисунку 3.5.

Схема функціонує наступним чином. Припустимо,, що існує матриця даних для навчання *D*, отриманих від системи та емпірична модель *f*, яка

навчається на основі D. Коли надходить набір нових вимірів d, оцінки величини d можуть бути отримані як $d' = f(d)$ і різниці можуть бути згенеровані як $e = d' - d$. Будь-які аварії в системі викличуть зміни у відношеннях між змінними d і d' і по цій причині призводять до статично аномальним змінам в різницях e, що може бути визначено на основі статистичних тестів.

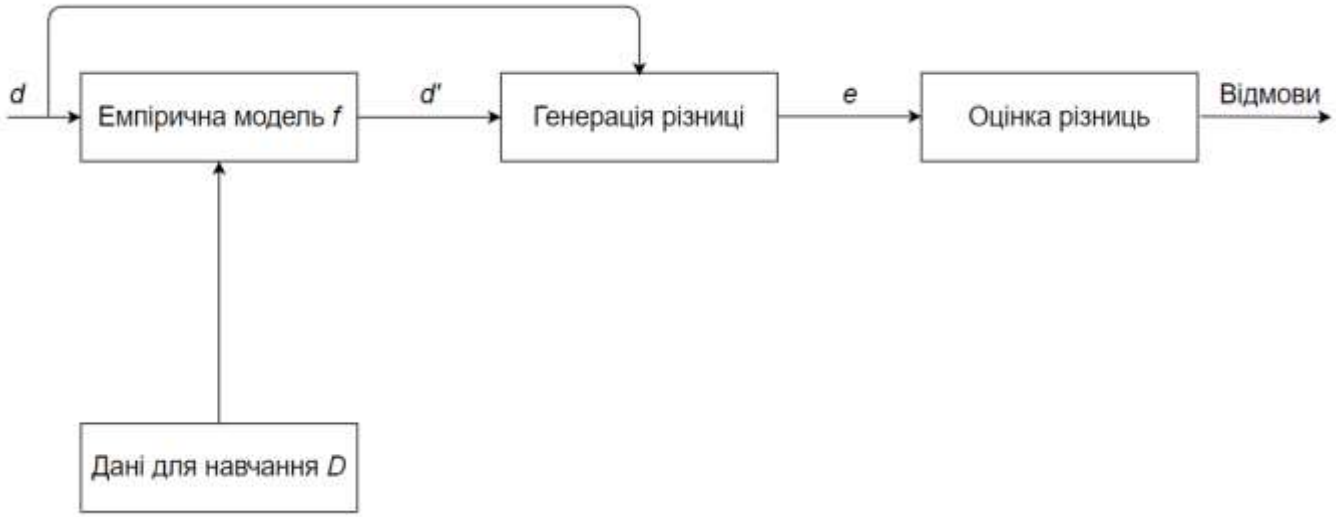

Рисунок 3.5 – Схема метода ідентифікації аварій на основі даних, що надходять

В якості емпіричних моделей або інструментів машинного навчання можуть використовуватись, наприклад, аналіз головних компонент, багатовимірна техніка оцінки станів, в якій визначаються взаємозв'язки між змінними, які використовуються для визначення стану об'єкта, штучна нейронна мережа.

До обмежень методів даного типу можна віднести те, що емпірична модель добре працює лише в межах робочого діапазону об'єкта. При виході за його границі вона може діагностувати сам факт виходу (початку аварії), але визначити тип аварії не може, оскільки дані цього типу ще не використовувались для навчання [7].

3.4 Метод зворотного поширення помилки

Для навчання багатошарового персептрону було використано метод зворотного поширення помилки. Це ітеративний градієнтний алгоритм, який використовується з метою мінімізації помилки роботи багатошарового персептрону та отримання бажаного виходу. 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.

В основі ідеї алгоритму знаходиться використання вихідної помилки ШНМ:

$$E = \frac{1}{2}\sum_{i=1}^{k}(y - y'^{2}) \quad (3.6).$$

Для обчислення величин корекції ваги нейронів в прихованих шарах нейронної мережі, де k – число вихідних нейронів мережі, y – цільове значення, y' – фактичне вихідне значення.

На кожні ітерації відбувається два ходи мережі – прямий і зворотній. На прямому ході вхідний вектор поширюється від входів мережі до її виходів і формує певний вихідний вектор, який відповідає поточному (фактичному) стану ваги нейронів. Далі обчислюється помилка нейронної мережі як різниця між фактичним та цільовим значеннями. На зворотньому ході ця помилка поширюється від виходу мережі до її входів і відбувається корекція ваги нейронів у відповідності до формули (3.7).

$$\Delta w_{j,i} = -\mu \frac{\partial E_{av}}{\partial w_{ij}} \quad (3.7),$$

де $w_{j,i}$ – вага i-го зв’язку та j-го нейрона, μ – параметр швидкості навчання, який дозволяє додатково керувати величиною кроку корекції $\Delta w_{j,i}$, з ціллю більш точного налаштування на мінімум помилки і підбирається експериментально в процесі навчання.

Враховуючи, що вихідна сума j-го нейрона дорівнює (3.1), можна показати, що:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial S_i}\frac{\partial S_j}{\partial w_{ij}} = x_i \frac{\partial E}{\partial S_i} \quad (3.8).$$

З (3.8) виходить, що диференціал ∂S_j активаційної функції нейронів мережі повинен існувати і не бути рівним нулю в любій точці, тобто активаційна функція повинна бути диференційованою на всій числовій осі. Тому для застосування метода зворотнього поширення було обрано сигмоїдальну активаційну функцію.

Таким чином алгоритм використовує так званий стохастичний градієнтний спуск, просуваючись в багатомірному просторі ваги нейронів у напрямку антиградієнту з ціллю досягнути мінімума функції помилки.

3.5 Градієнтний спуск

Градієнтний спуск – метод знаходження мінімума або максимума функції за допомогою руху вздовж градієнту або антиградієнту (рисунок 3.6). Для мінімізації функції у напрямку градієнта використовують методи одномірної оптимізації, наприклад метод золотого перерізу.

Нехай цільова функція має вигляд:

$$f(\vec{x}): X \to R \quad (3.9).$$

Задача оптимізації задана наступним чином:

$$f(\vec{x}) \to \min_{x \in X} f(x) \quad (3.10).$$

Основна ідея метода полягає в тому, щоб рухатись у напрямку найшвидшого спуску, а цей напрямок задається антиградієнтом – ∇F:

$$\overrightarrow{x}^{|j+1|} = \vec{x}^{|j|} - \gamma^{|j|}\nabla F(\vec{x}^{|j|}) \quad (3.11),$$

де $\gamma^{|j|}$ вибирається:

- Постійною, в цьому випадку метод може розходитись.
- Дробовим кроком, тобто довжина кроку в процесі спуску ділиться на деяке число.
- Найшвидшим спуском:

$$\gamma^{|i|} = argmin_{\gamma} F(\vec{x}^{|j|} - \gamma^{|j|} \nabla F(\vec{x}^{|j|})) \quad (3.12).$$

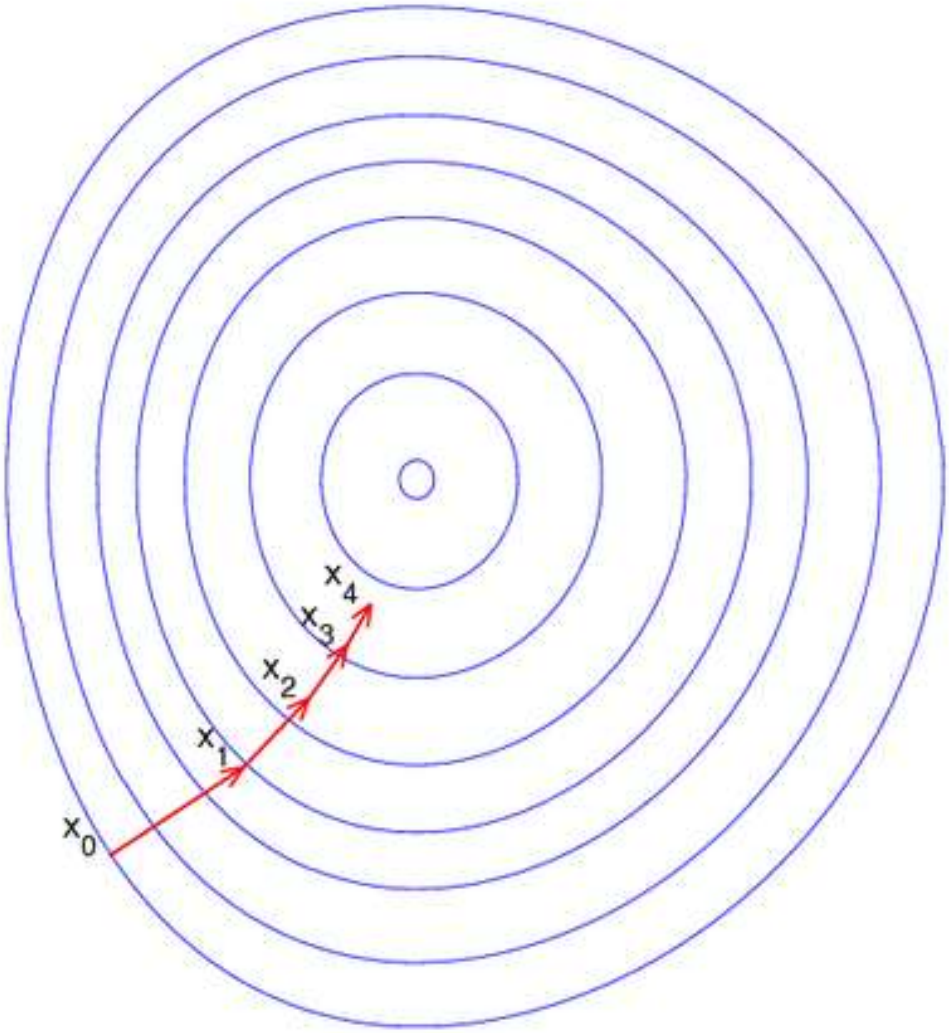

Рисунок 3.6 – Метод градієнтного спуску

Збіжність метода градієнтного спуску залежить від відношення максимального і мінімального власних чисел матриці Гессе в околі мінімума або максимума. Чим більше це відношення, тим гірша збіжність метода.

3.6 Критерій Акаіке

Критерій для вибору найкращої з декількох статистичних моделей, побудованих на однаковому наборі даних. Критерій є не статистичним, а інформаційним, оскільки базується на оцінці втрати інформації при зменшені

числа параметрів моделі. Критерій дозволяти знайти компроміс між складністю моделі (числом параметрів) і її точністю. В загальному випадку критерій Акаіке визначається по формулі:

$$AIC = 2k - 2\ln(L) \quad (3.13),$$

де:

k – число параметрів моделі.

L – максимізоване значення функції правдоподібності моделі.

Найкращою визнається модель для якої значення *AIC* є мінімальним.

AIC тісно пов'язаний з інформаційним критерієм Баєса, але, на відміну від нього, містить у собі штрафну функцію, яка лінійно залежить від числа параметрів.

Якщо модель використовує метод найменших квадратів, то критерій може бути обчисленим наступним чином:

$$AIC = \ln\left(\frac{RSS_p}{n}\right) + 2\frac{p}{n} + 1 + \ln(2\pi) \quad (3.14),$$

де:

RSS_p – сума квадратів залишків моделі, отримана при оцінці коефіцієнтів моделі методом найменших квадратів.

n – об'єм навчальної вибірки.

З формули видно, що при фіксованому розмірі навчальної вибірки ріст критерію обумовлений в більшості ростом кількості параметрів моделі, а не її помилкою. Тобто за збільшення кількості параметрів модель штрафується сильніше, ніж за долю помилки. Таким чином, задача полягає в тому щоб вибрати модель з мінімальним числом параметрів, які пояснюють найбільшу частину дисперсії помилки.

Тобто береться модель початкова (нульова) модель і для неї обчислюється значення критерію. Потім в нульову модель почергово добавляються параметри і кожен раз *AIC* обчислюється знову. В результаті обирається модель для якої значення критерію виявиться мінімальним.

3.7 Модель спроектованої нейронної мережі

Для системи прогнозування стану складного технічного об’єкта було розроблено модель штучної нейронної мережі на основі багатошарового персептрону (рисунок 3.7).

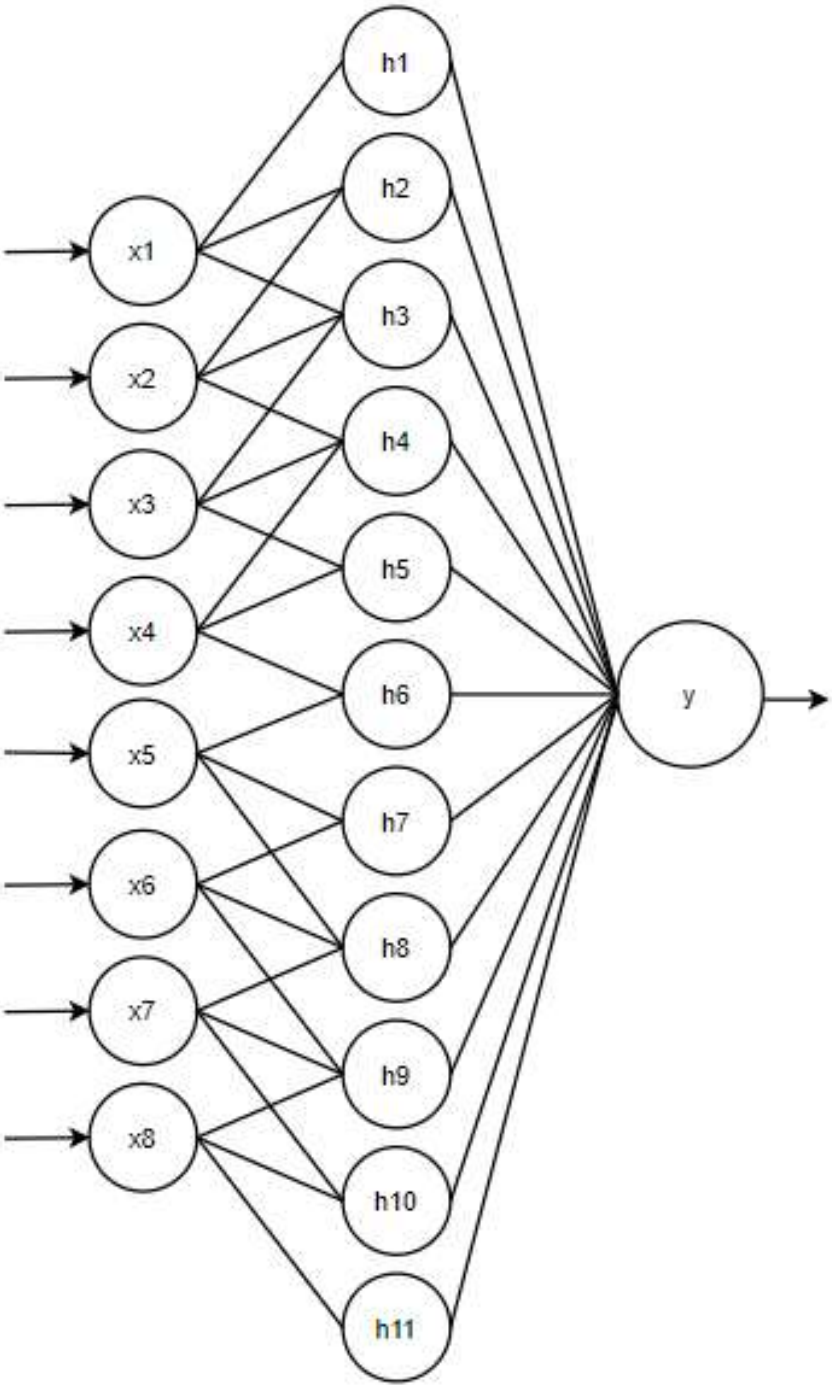

Рисунок 3.7 – ШНМ для системи прогнозування стану складного технічного об’єкта

Вектор вхідних значень (сигналів) $x_{1...8}$ (ВхШ). Один прихований шар. Вихідний шар складається з одного нейрона, який є ідентифікатором стану складного технічного об’єкта.

3.8 Конфігурація нейронної мережі

Оптимальна конфігурація нейронної мережі була визначена використовуючи критерій Акаіке.

Навчальна вибірка містить 8 параметрів. Ітераційним способом було збільшено кількість проміжних шарів та кількість нейронів у ПШ, визначено число критерію для кожної ітерації та обрано мінімальне значення. В результаті отримано оптимальну конфігурацію ШНМ (таблиця 3.1).

Архітектура	Персептрон
Кількість прихованих шарів	1
Кількість вхідних нейронів	8
Кількість нейронів у прихованому шарі	11
Кількість вихідних нейронів	1
Функція активації	Сигмоїда
Кількість ітерацій	890
Коефіцієнт швидкості навчання	0.2

Таблиця 3.1 – Конфігурація нейронної мережі.

В якості функції активації було обрано сигмоїду, яка вдало підходить для задач класифікації.

4. РОЗРОБКА СИСТЕМИ ПРОГНОЗУВАННЯ СТАНУ СКЛАДНОГО ТЕХНІЧНОГО ОБ'ЄКТА

Для початку процесу розробки системи прогнозування стану складного технічного об'єкту було досліджено та обрано найбільш оптимальні засоби розробки програмного забезпечення. Було спроектовано об'єктно-орієнтовано представлення системи.

4.1 Засоби розробки

Для розробки системи прогнозування стану складного технічного об'єкта з допомогою нейронної мережі було обрано інтерпретовану об'єктно-орієнтовану мову програмування високого рівня зі строгою динамічною типізацією Python версії 3.7.

Програмне середовище розробки Pycharm Professional Edition 2020.

Для реалізації алгоритму роботи нейронної мережі було використано бібліотеки sklearn, numpy та pandas.

Для пришвидшення роботи алгоритму було застосовано функціонал бібліотеки multiprocessing.

Графічний інтерфейс було створено використовуючи бібліотеку PyQt5. Дизайн було спроектовано у Qt Designer та конвертовано у код за допомогою застосунку pyuic5.

Візуалізація даних була реалізована використовуючи бібліотеку PyQtGraph.

Застосунок створено утилітою auto-py-to-exe.

У якості апаратного забезпечення було використано персональний комп'ютер Acer Aspire 3 A315-55G, Intel Core i3-8145, 8/256 Гб, GeForce MX230 2 Гб.

4.2 Об’єктно-орієнтоване представлення

На рисунку 4.1 зображене об’єктно-орієнтоване представлення програмної реалізації системи прогнозування стану складного технічного об’єкта.

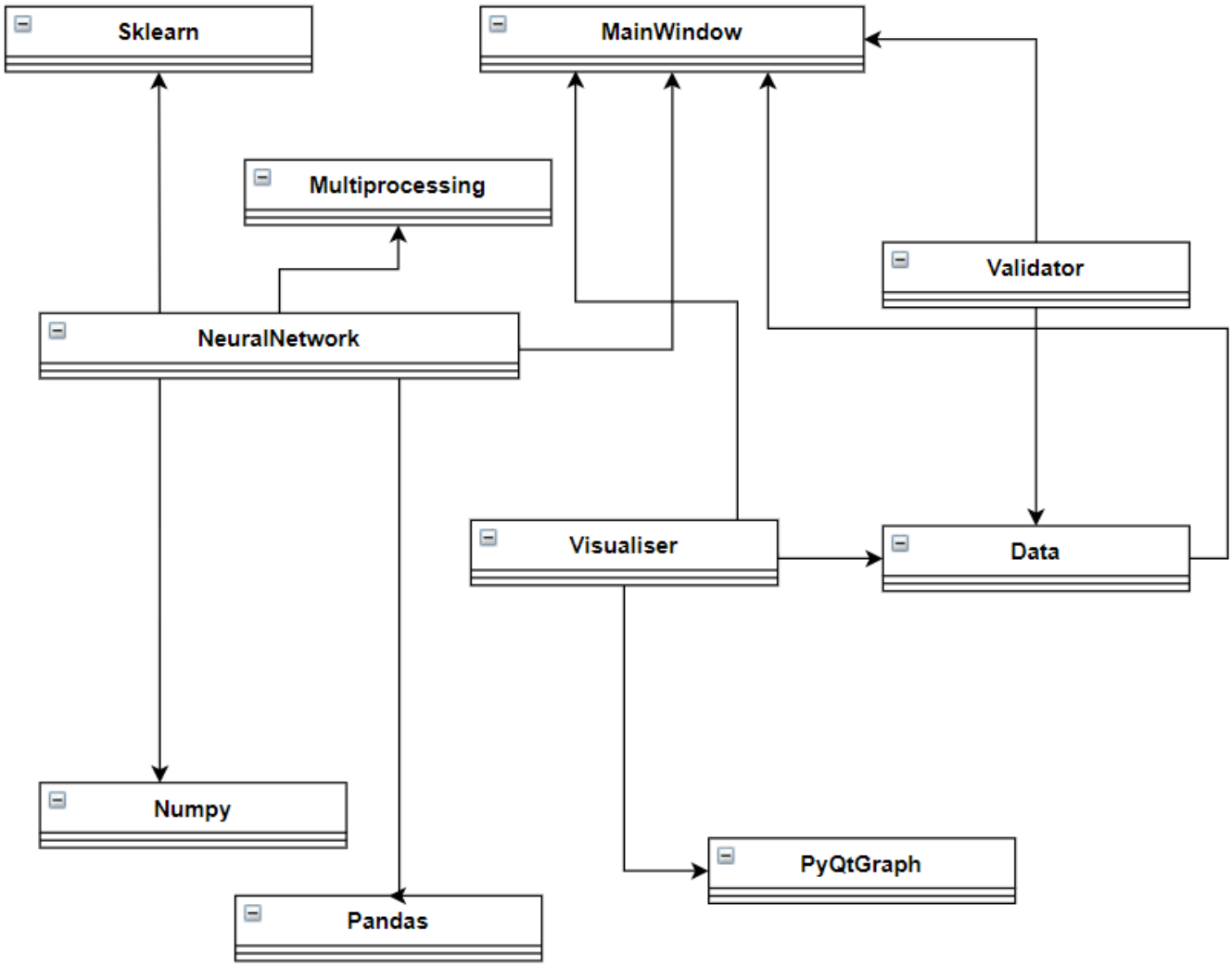

Рисунок 4.1 – Об’єктно-орієнтоване представлення системи

Клас NeuralNetwork містить в собі реалізацію штучної нейронної мережі. Набір атрибутів класу містить:

- Кількість ітерацій для навчання мережі.
- Кількість прихованих шарів.
- Ідентифікатор натренованості мережі.

- Параметр швидкості навчання мережі.

Методи класу:

- Запуск навчання нейронної мережі.
- Зупинка навчання у випадку зростання помилки або невірності вхідних даних.
- Застосування навченої ШНМ до аналізу нових даних, прогнозуючи стан складного технічного об’єкту.

Клас MainWindow є реалізацію графічного інтерфейсу системи, набір атрибутів представляє список віджетів, використаних у розробці користувацького інтерфейсу. Функціонал кожного віджета під’єднано за допомогою сигналів до методів класу:

- Обрати джерела даних.
- Задати параметри для навчання нейронної мережі.
- Очистити консоль.
- Вийти з системи.

Також керуючі віджети графічного інтерфейсу з’єднані з методами класу NeuralNetwork та Data надаючи можливість завантажувати дані, розпочинати навчання тощо.

Клас Validator надає можливість перевірки вхідних даних, обмежуючи дії користувача які можуть призвести до неправильної роботи системи.

Клас Visualiser надає функціонал для візуалізації процесу навчання та процесу аналізу і прогнозування стану складного технічного об’єкта.

4.3 Варіанти використання

На рисунку 4.2 представлена діаграма варіантів використання системи прогнозування стану складного технічного об’єкта з допомогою штучної нейронної мережі.

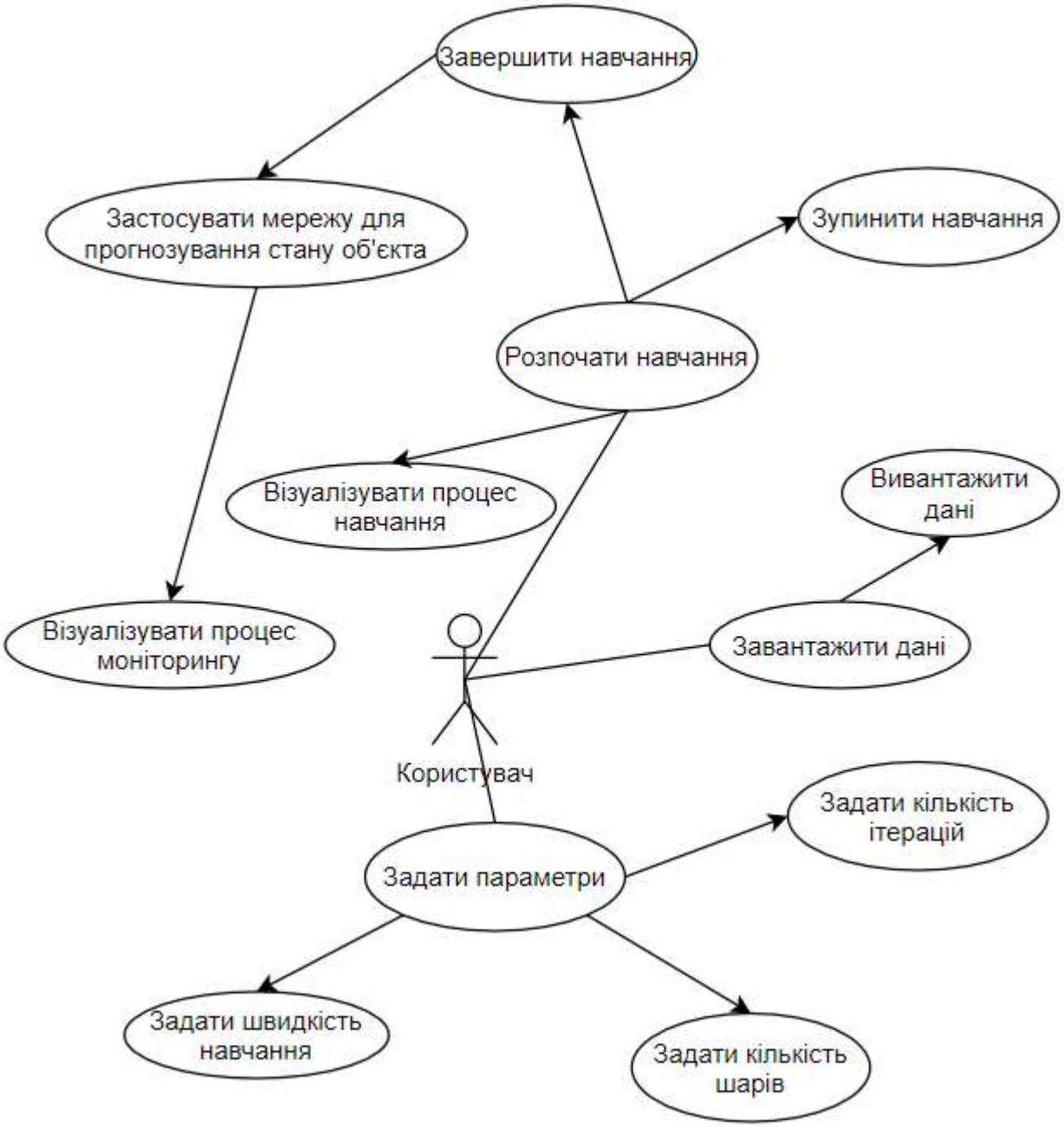

Рисунок 4.2 – Діаграма прецедентів системи

Користувачу надана можливість проводити навчання штучної нейронної мережі, попередньо завантаживши дані та завдавши параметри кількості ітерацій, шарів та швидкості навчання. Опісля процесу навчання користувач може застосувати нейронну мережу для аналізу та прогнозування стану складного технічного об’єкта, візуалізувати отримані результати у вигляді графіків.

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для запуску системи необхідно виконати System of analysis and forecasting of the state of a complex technical object.exe (рисунок 5.1).

Рисунок 5.1 – Застосунок системи

Системні вимоги для запуску:

- Процесор Intel Celeron/Pentium.
- Розмір оперативної пам'яті 4 Гб.
- Вільного дискового простору 1 Гб.
- Операційна система Windows 10.

Вище зазначені системні вимоги розраховані на роботу з невеликими масивами даних, тому зі збільшенням кількості даних, збільшенням кількості прихованих шарів та кількості ітерацій зростають вимоги до апаратного забезпечення. Тому рекомендованими вимогами для оптимальної експлуатації системи прогнозування стану складного технічного об’єкта є:

- Процесор Intel Core i5 8-го покоління.
- 16 Гб оперативної пам’яті.
- Вільного дискового простору 1 Гб.
- Операційна система Windows 10.

Після запуску застосунку на екрані монітору з’явиться головне вікно системи (рисунок 5.2). Розглянемо його основні елементи:

1. Центр навчання.
2. Центр моніторингу, аналізу, прогнозування.
3. Область даних.
4. Консоль.
5. Меню.

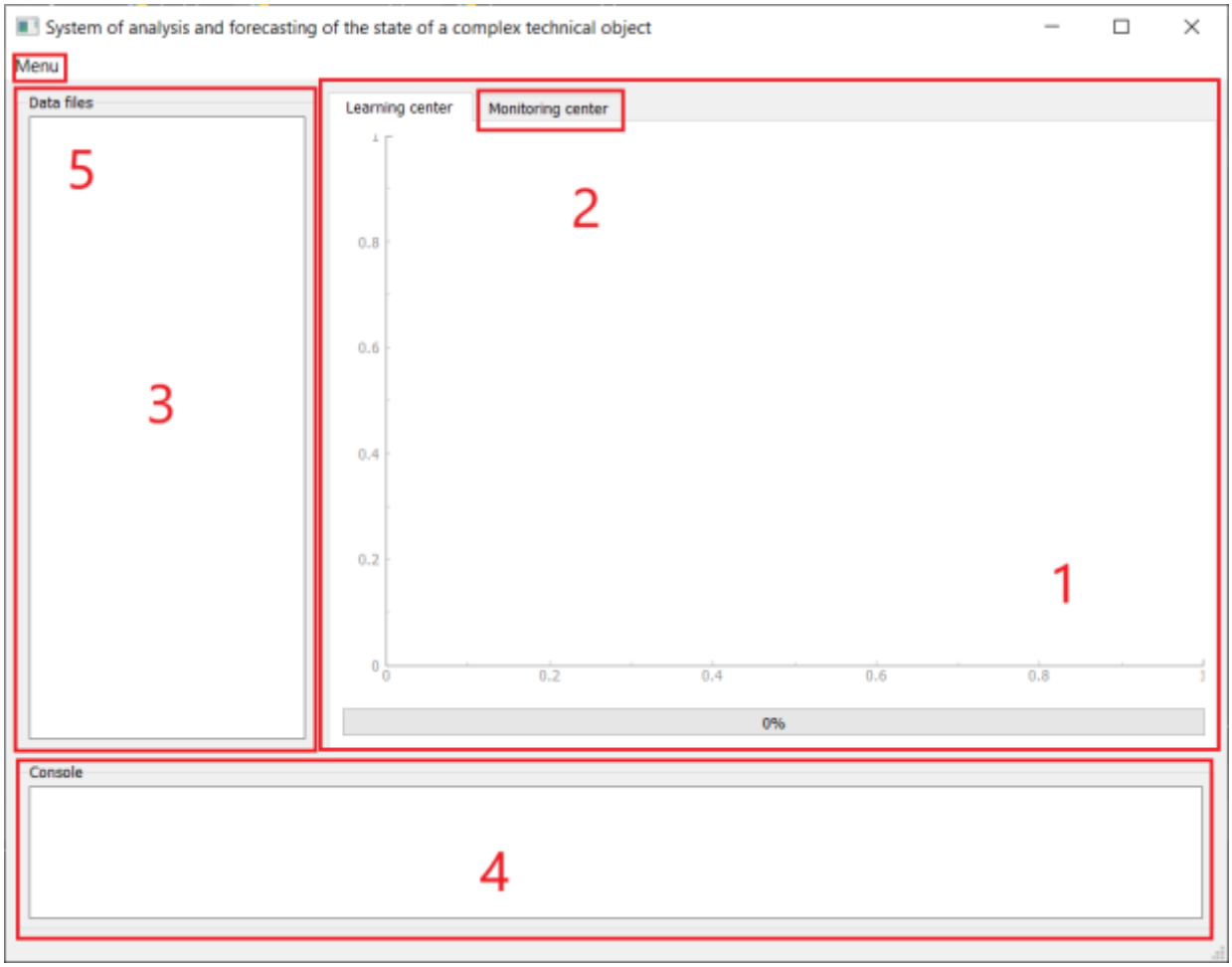

Рисунок 5.2 – Головне вікно системи

Запуск функціональних процесів системи відбувається у меню (рисунок 5.3). Розглянемо пункти меню системи:

1. Розпочати навчання
2. Розпочати моніторинг, аналіз, прогнозування
3. Завантажити дані
4. Задати параметри
5. Звільнити дані

6. Очистити консоль
7. Зупинити навчання
8. Зупинити моніторинг, аналіз, прогнозування
9. Вихід

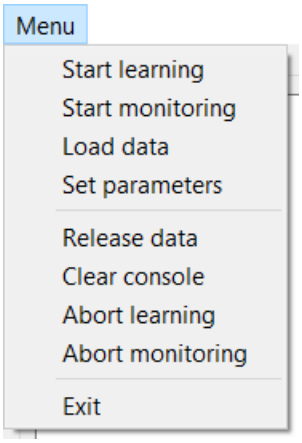

Рисунок 5.3 – Меню системи

Для початку навчання штучної нейронної мережі необхідно:

1. Завантажити дані (рисунок 5.4). Необхідно натиснути пункт меню Load data, обрати необхідні файли та підтвердити вибір.

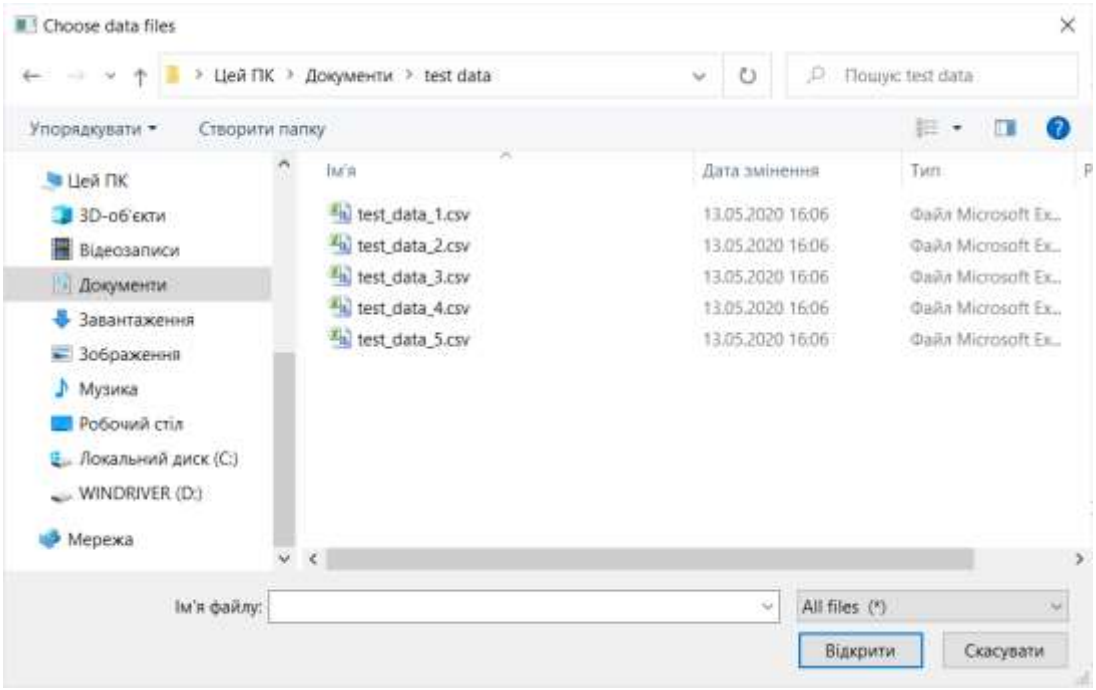

Рисунок 5.4 – Вікно вибору файлів

Після цього в області даних з’явиться список завантажених файлів. У консолі можна спостерігати за процесом завантаження даних (рисунок 5.5).

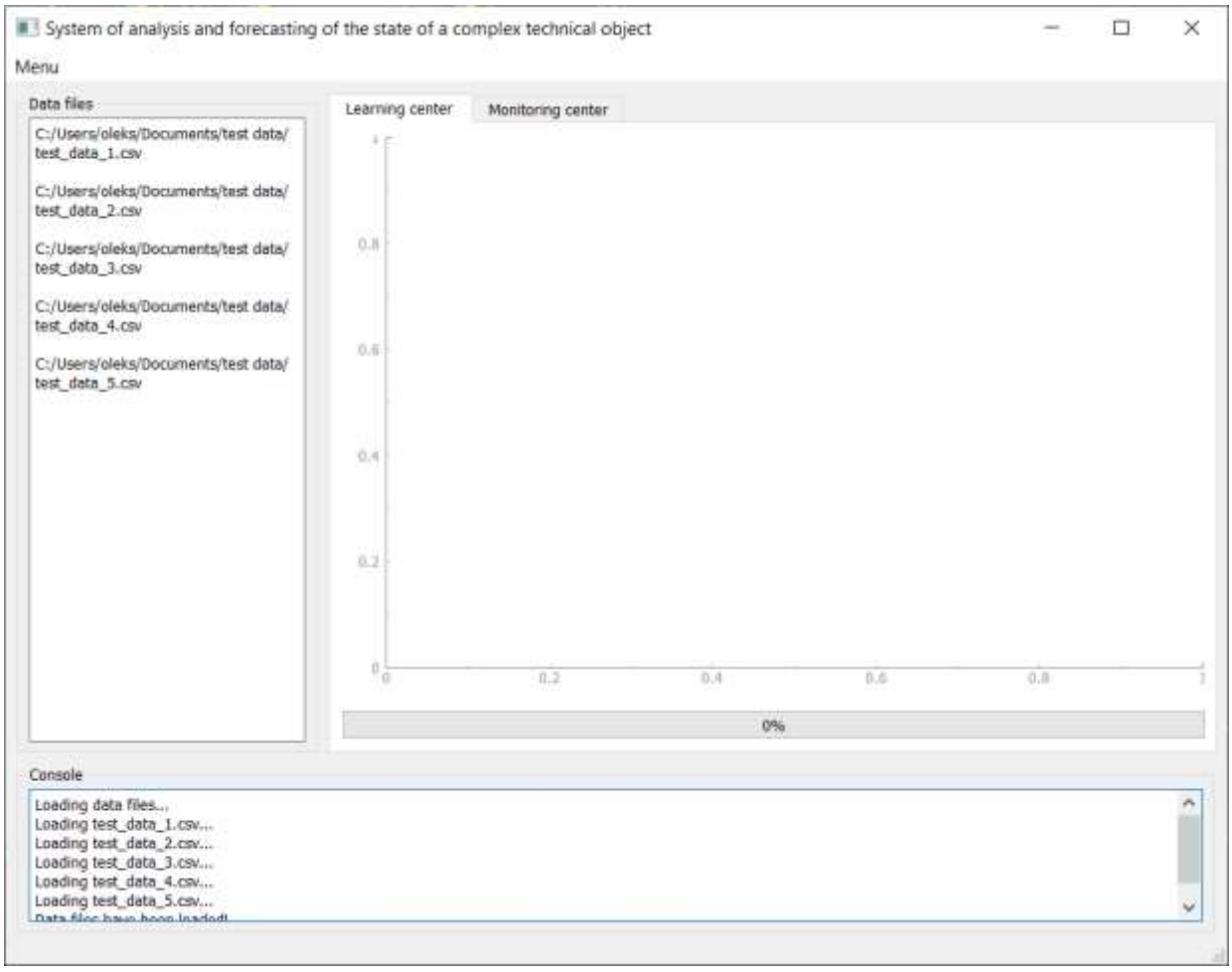

Рисунок 5.5 – Головне вікно після завантаження даних

2. Задати параметри ШНМ натиснувши пункт меню Set parameters:

- Кількість ітерацій навчання (рисунок 5.6)

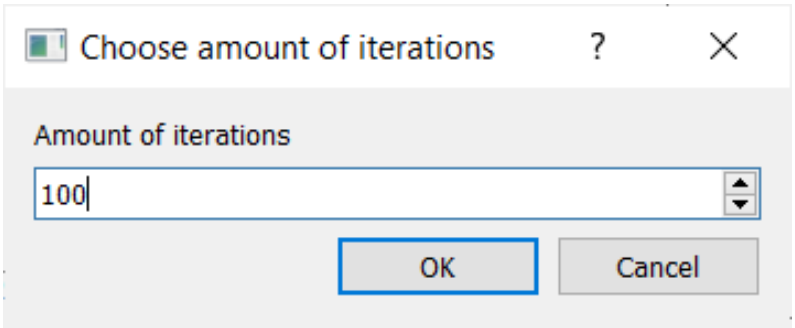

Рисунок 5.6 – Вікно вибору кількості ітерацій навчання

- Кількість прихованих шарів (рисунок 5.7)

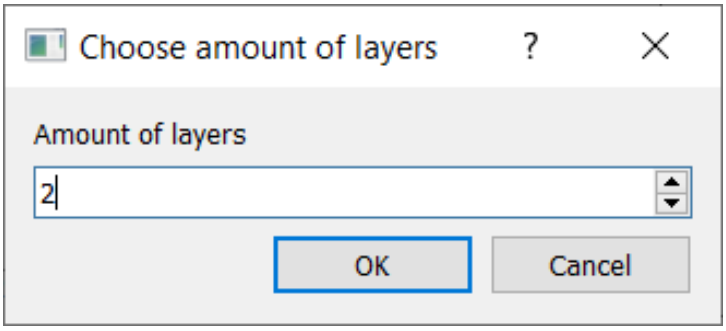

Рисунок 5.7 – Вікно вибору кількості шарів

- Швидкість навчання (рисунок 5.8)

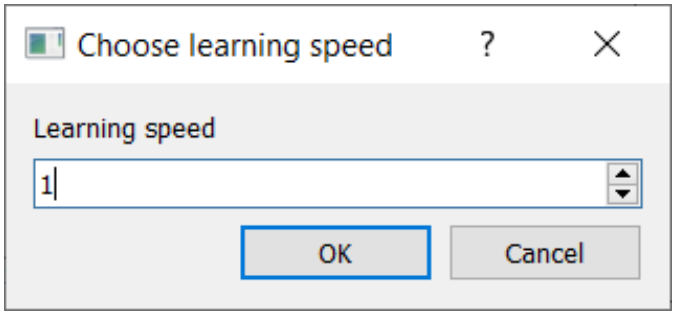

Рисунок 5.8 – Вікно вибору швидкості навчання

Якщо спробувати розпочати навчання не виконавши вище зазначені пункти система виведе на екран попередження про необхідність виконання цих дій (рисунок 5.9, рисунок 5.10, рисунок 5.11).

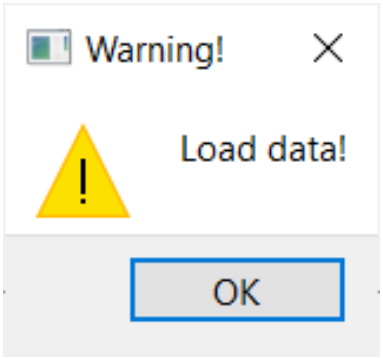

Рисунок 5.9 – Попередження про необхідність завантаженнія даних

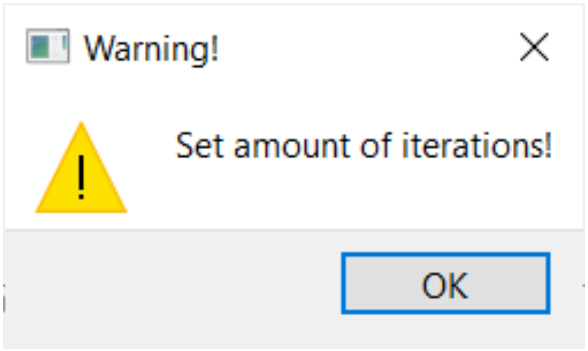

Рисунок 5.10 – Попередження про необхідність вибору кількості ітерацій

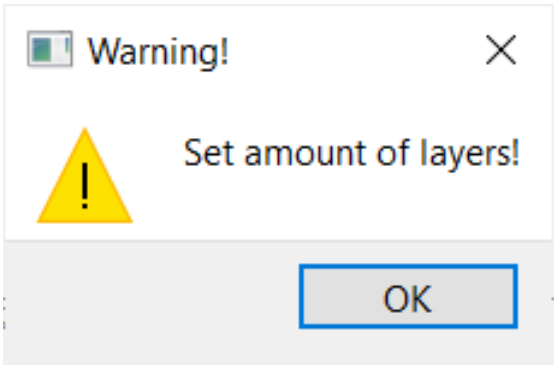

Рисунок 5.11 – Попередження про необхідність вибору кількості прихованих шарів

Після завантаження даних та задання параметрів (рисунок 5.12) можна розпочинати процес навчання нейронної мережі натиснувши пункт меню Start learning (рисунок 5.13).

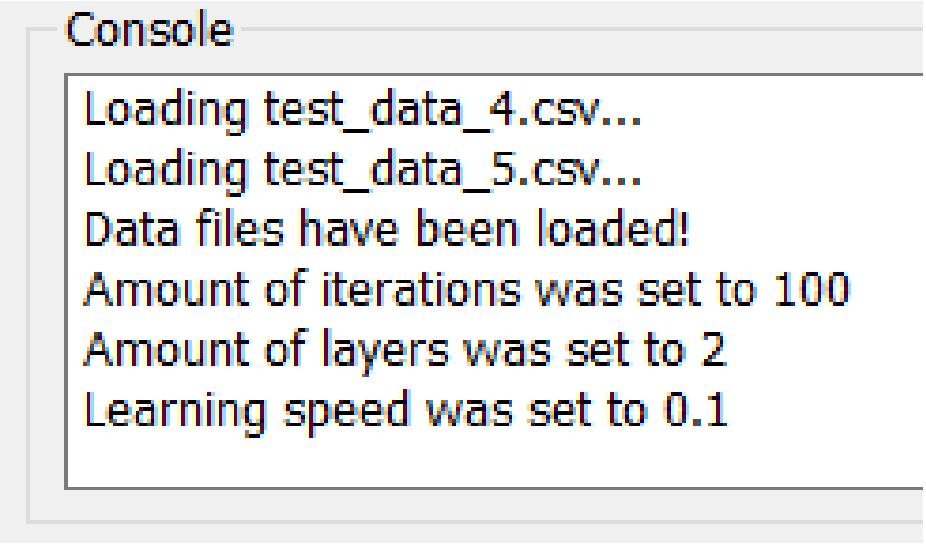

Рисунок 5.12 – Повідомлення у консолі про завантажені дані та задані параметри

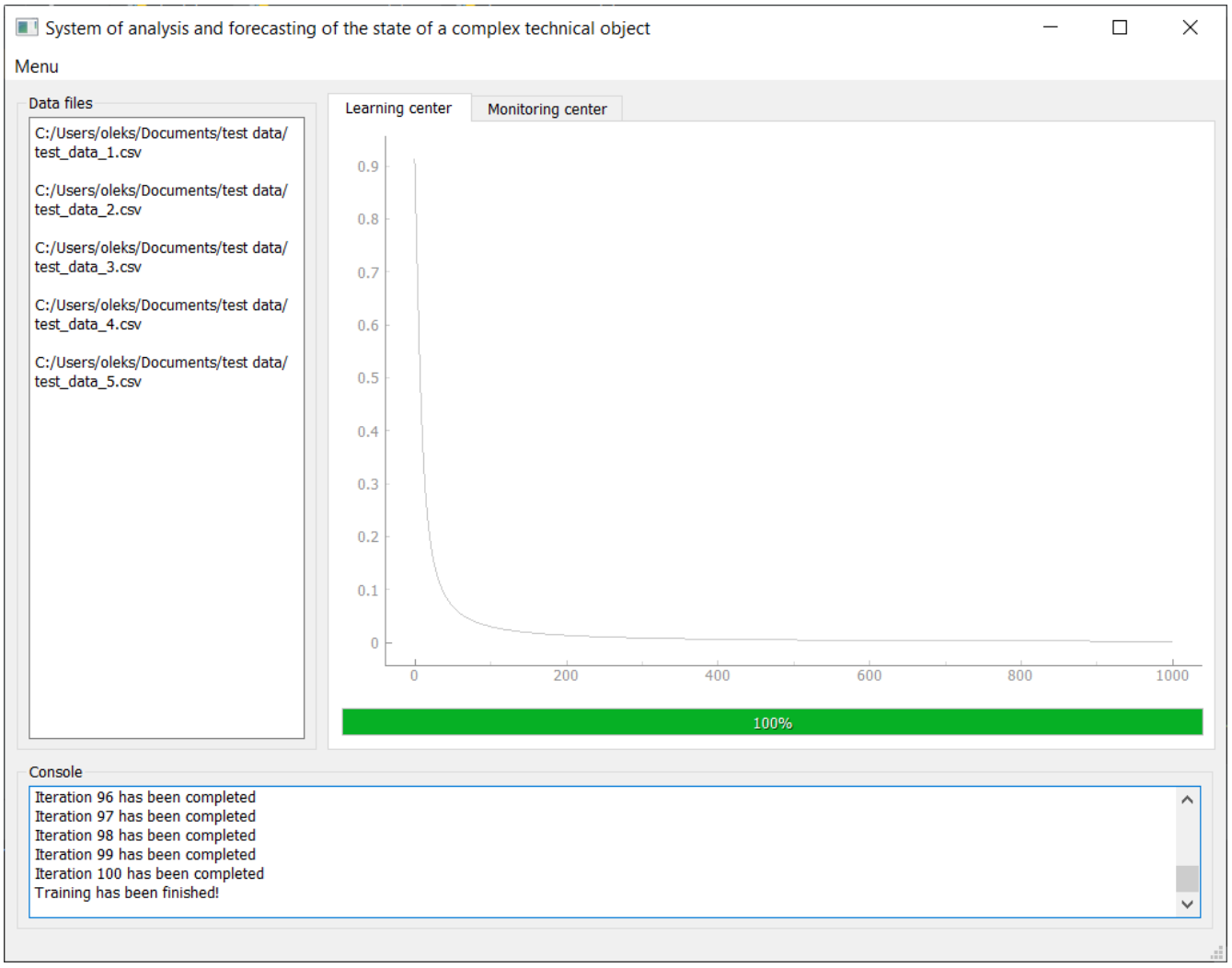

Рисунок 5.13 – Головне вікно. Завершений процес навчання

У консолі можна спостерігати за процесом навчання нейронної мережі. Також прогрес навчання відображається у віджеті індикатора прогресу (0-100%). У центрі навчання в реальному часі будується графік квадратичної похибки. Коли вона наближається до нуля процес навчання можна вважати завершеним.

Після завершення процесу навчання мережі користувач має змогу застосовувати її можливості до аналізу, моніторингу, прогнозування стану складного технічного об’єкта завантаживши дані його параметрів. Для початку процесу аналізу, моніторингу, прогнозування необхідно натиснути на пункт меню Start monitoring (рисунок 5.14).

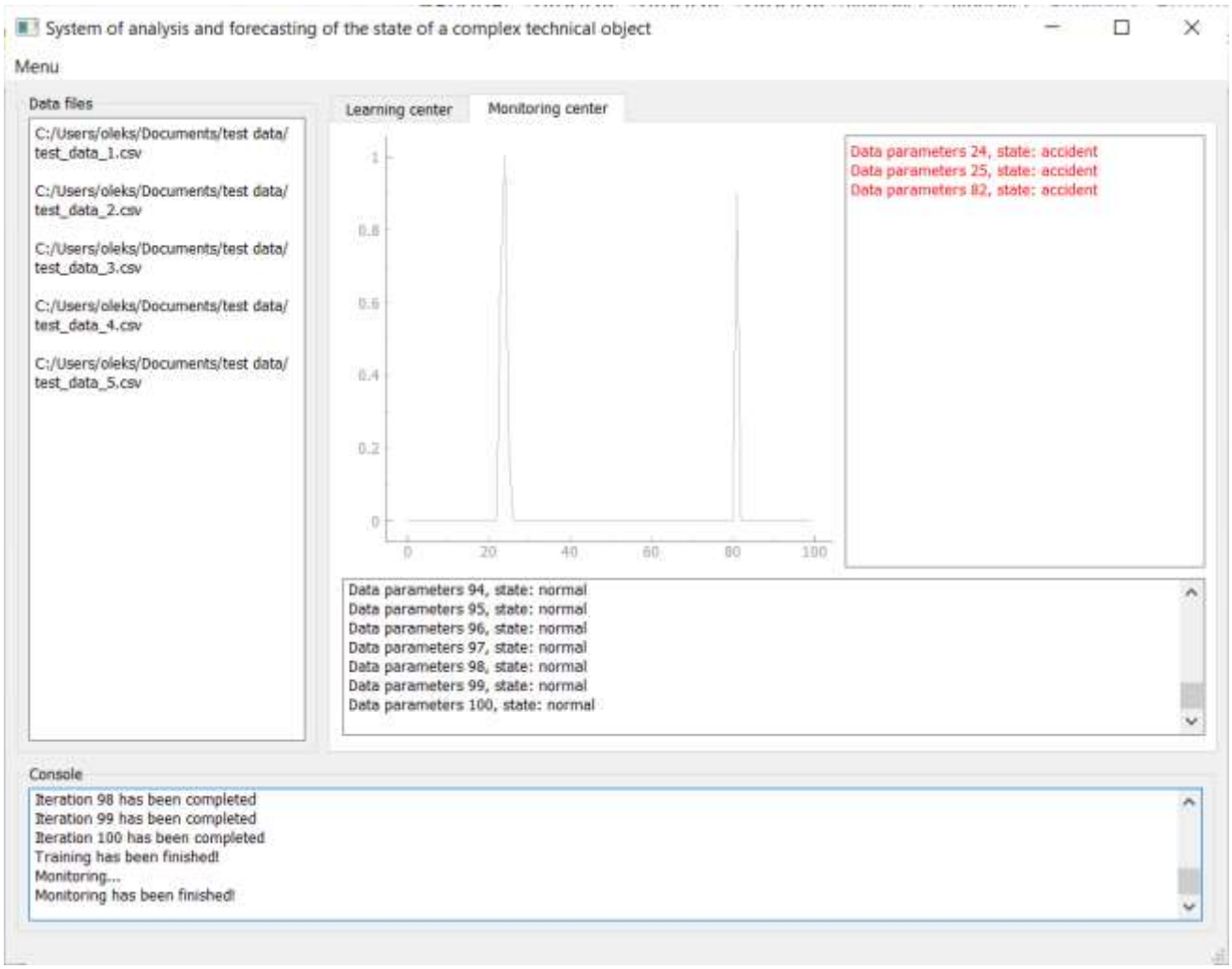

Рисунок – 5.14 Головне вікно. Завершений процес аналізу, моніторингу, прогнозування

Якщо користувач спробує розпочати процес моніторингу попередньо не провівши навчання нейронної мережі система сповістить про це (рисунок 5.15).

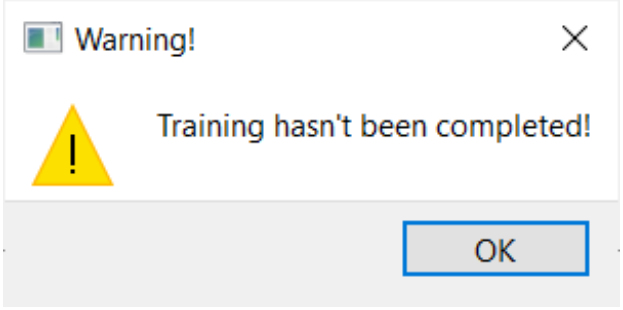

Рисунок 5.15 – Попередження про необхідність попереднього процесу навчання нейронної мережі

Після запуску процесу аналізу, моніторингу, прогнозування у вікні центру моніторингу відображається стан складного технічного об’єкта. Коли стан, згідно параметрів які надходять до системи, знаходиться в межах норми – значення на графіку близьке до нуля, коли відбувається інцидент або аварія – значення на графіку сягає одиниці. У нижньому полі центру моніторингу відображається перебіг процесу аналізу, моніторингу, прогнозування. У правому полі з’являються попередження про збої в роботі об’єкта.

6. ТЕСТУВАННЯ СИСТЕМИ

Перед тестуванням режиму моніторингу за станом складного технічного об’єкту необхідно провести навчання нейронної мережі. Для навчання ШНМ було використано навчальну вибірку, яка містить набори параметрів складного технічного об’єкта (рисунок 6.1).

Температура навколишнього середовища	Температура охолоджуючої рідини	Напруга d-компоненту	Напруга q-компоненту	Швидкість мотору	Момент обертання	Температура поверхні постійного магніта	Температура ярма статора
-0,75214297	-1,1184461	0,3279352	-1,2978575	-1,2224282	-0,2501821	-2,522071	-1,8314217
-0,77126324	-1,1170206	0,3296648	-1,2976865	-1,2224293	-0,2491333	-2,5224178	-1,8309687
-0,78289163	-1,1166813	0,3327715	-1,3018217	-1,2224278	-0,2494311	-2,5226731	-1,8304
-0,78093535	-1,1167642	0,3336999	-1,301852	-1,2224301	-0,2486364	-2,521639	-1,8303328
-0,7740426	-1,116775	0,3352061	-1,303118	-1,2224286	-0,2487008	-2,5219002	-1,8304977
-0,7629362	-1,1169548	0,33490124	-1,3030168	-1,2224286	-0,248197	-2,522203	-1,8319309
-0,74922806	-1,1161705	0,33501354	-1,3020816	-1,2224296	-0,2479142	-2,5225377	-1,8330117
-0,7384499	-1,1139864	0,3362563	-1,3051548	-1,2224321	-0,248321	-2,5228438	-1,8321822
-0,7309097	-1,1118276	0,3349053	-1,3037902	-1,2224315	-0,2477847	-2,5228078	-1,8315759
-0,72712964	-1,1094859	0,3359881	-1,3056333	-1,2224314	-0,2482944	-2,5226767	-1,8314383
-0,72371286	-1,1082836	0,33539978	-1,3045602	-1,2224283	-0,247913	-2,5226302	-1,8314928
-0,71775365	-1,1085879	0,33443072	-1,3043374	-1,2224288	-0,2477212	-2,5226388	-1,8318189
-0,7048766	-1,1099857	0,33624125	-1,30582	-1,2224338	-0,2480017	-2,5222142	-1,8320519
-0,6825343	-1,1113616	0,33556557	-1,3032398	-1,2224305	-0,2475126	-2,5219264	-1,8320305
-0,654583	-1,1126	0,33655748	-1,3062165	-1,2224299	-0,2480666	-2,5221426	-1,8324168
-0,63071096	-1,1136569	0,33535197	-1,3055588	-1,2224302	-0,2475905	-2,5222142	-1,8320504
-0,62009937	-1,1142852	0,33590913	-1,3056751	-1,2224299	-0,2478845	-2,5223105	-1,8317783
-0,6148578	-1,1145984	0,33629027	-1,3066177	-1,2224327	-0,2479104	-2,5216212	-1,8313637
-0,60744816	-1,1150228	0,33301646	-1,2344021	-1,1908675	-0,3074368	-2,5201533	-1,8313625
-0,6037373	-1,1158898	0,32455894	-0,9830486	-1,0761522	-0,3482398	-2,520239	-1,8324342
-0,60323447	-1,1150343	0,316031	-0,63691336	-0,9152268	-0,3762608	-2,5206366	-1,8327438
-0,6004591	-1,1119267	0,3084021	-0,26627913	-0,740405	-0,3997889	-2,521099	-1,8335521
-0,59830505	-1,1096134	0,30417806	0,09185752	-0,5674627	-0,3770513	-2,5225427	-1,8346876
-0,5997665	-1,1090795	0,29940167	0,47535366	-0,3773778	-0,3842986	-2,5232556	-1,8341513
-0,60371727	-1,1082681	0,29658678	0,87474203	-0,1744461	-0,4046426	-2,5230372	-1,8329986
-0,6035687	-1,1073983	0,29612818	1,1744009	-0,0112797	-0,3678239	-2,522864	-1,8318759
-0,6025639	-1,1054107	0,290501	1,3467971	0,1055844	-0,3379655	-2,522989	-1,8307649
-0,59803104	-1,102991	0,27885312	1,4665564	0,2155701	-0,3736125	-2,522904	-1,8299689

Рисунок 6.1 – Набір даних для навчання нейронної мережі

Параметри навчальної вибірки:

- Формат файлу: comma-separated values (*.csv).
- Розмір файлу: 130026 Кб
- Кількість параметрів: 8.
    - Температура навколишнього середовища.

- Температура охолоджуючої рідини
- Напруга d-компоненту
- Напруга q-компоненту
- Швидкість мотору
- Момент обертання
- Температура постійного магніту
- Температура ярма статора

- Кількість наборів параметрів: 998070

На рисунках 6.1 – 6.3 зображено графік залежності квадратичної помилки від кількості ітерацій.

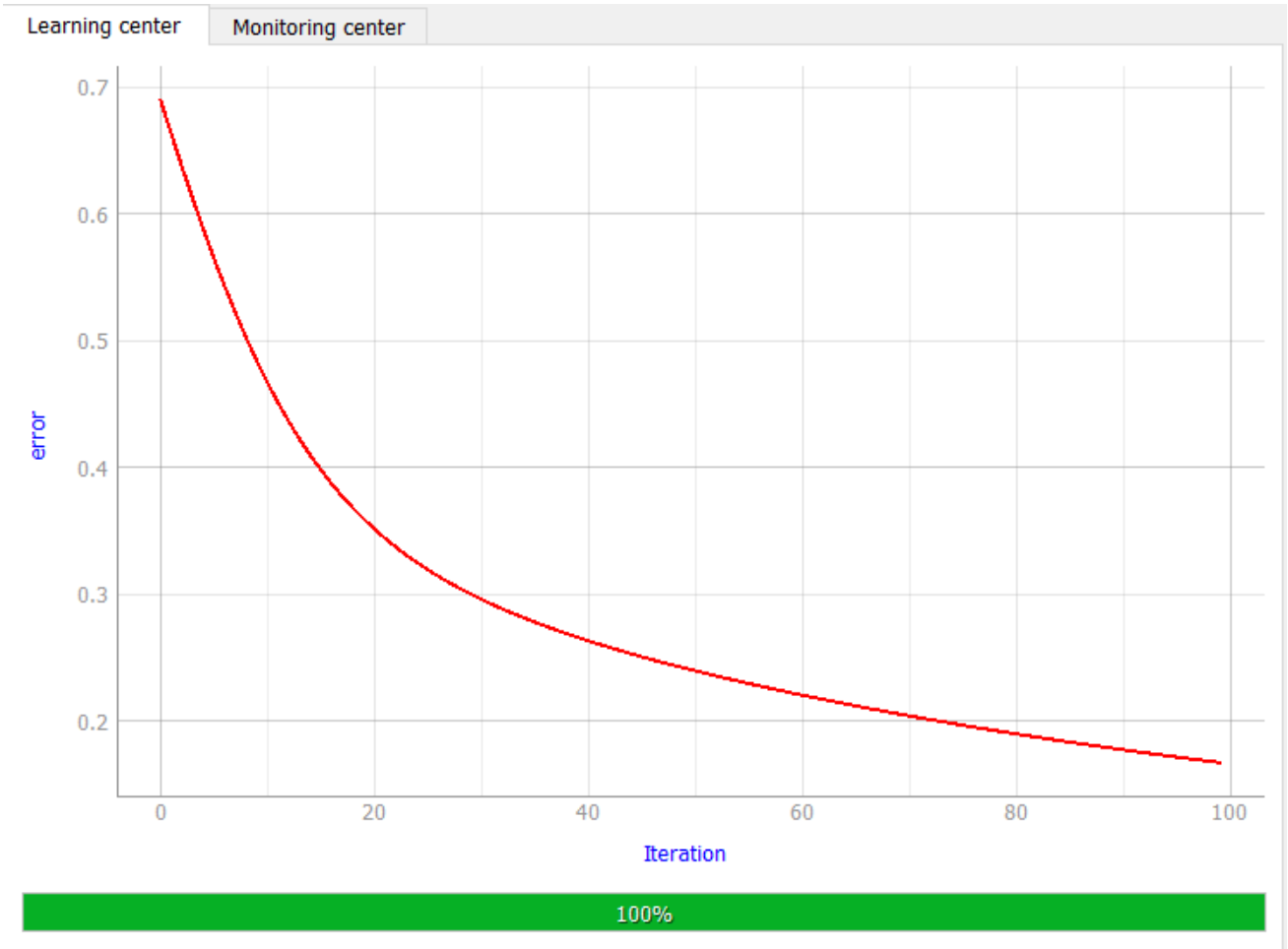

Рисунок 6.2 – Залежність квадратичної помилки від кількості ітерацій (100)

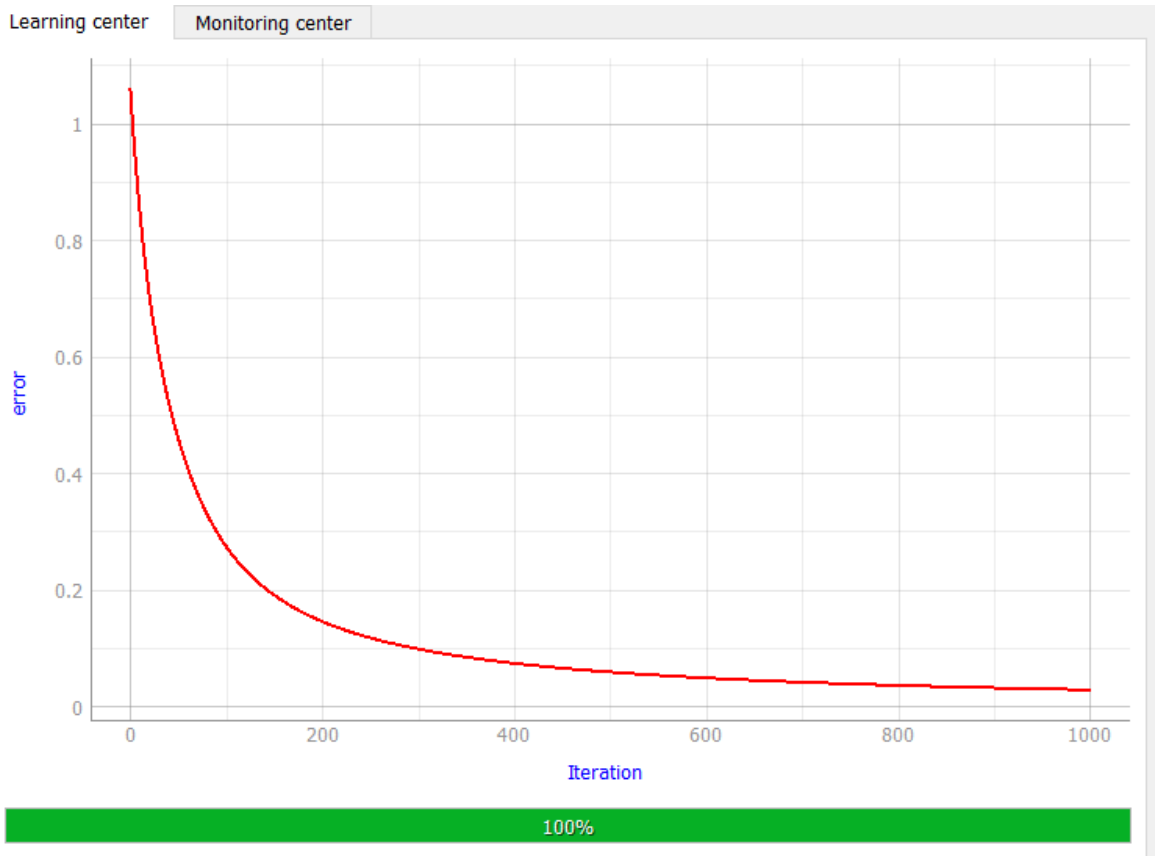

Рисунок 6.3 – Залежність квадратичної помилки від кількості ітерацій (1000)

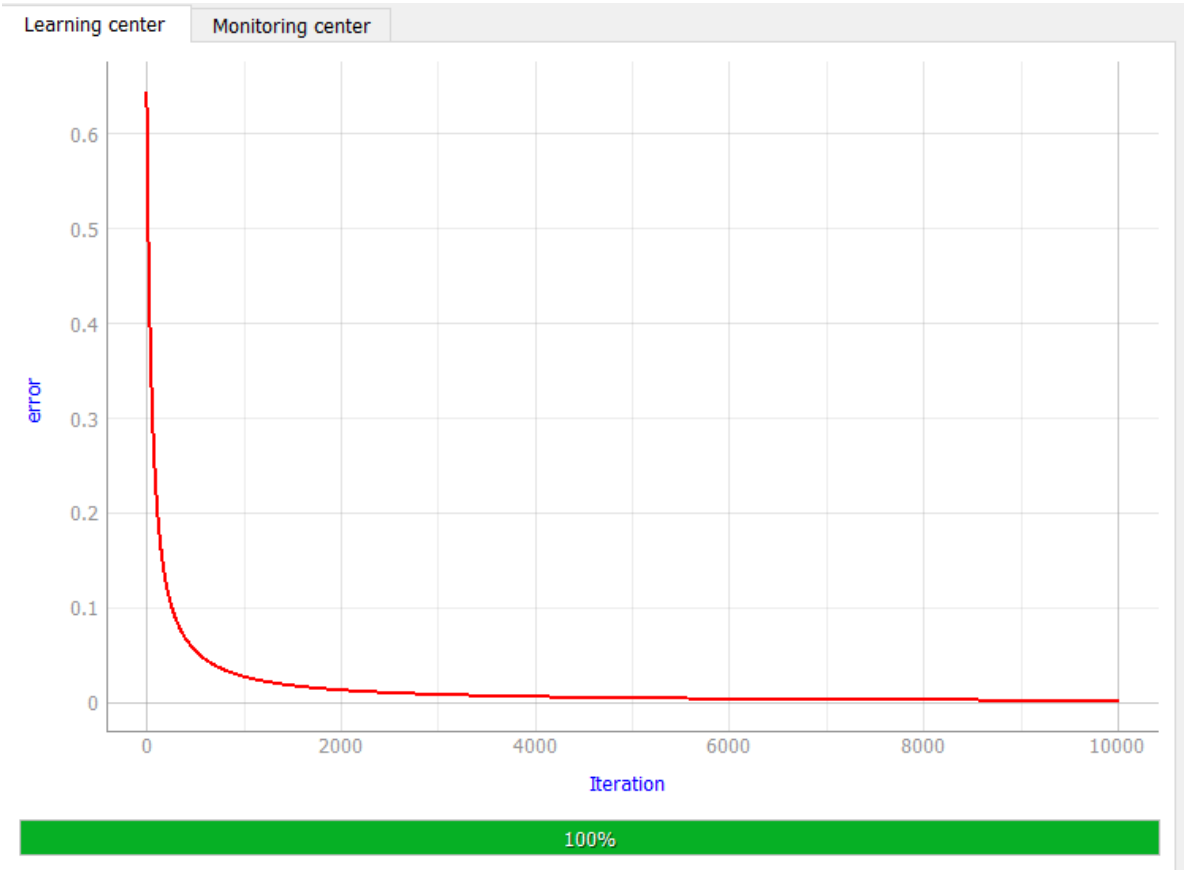

Рисунок 6.4 – Залежність квадратичної помилки від кількості ітерацій (10000)

Для тестування системи прогнозування стану складного технічного об’єкту було розроблено код для генерації скінченних числових рядів для симуляції набору параметрів складного технічного параметру.

Нижче наведені результати моніторингу за станом складного технічного об’єкту на основі різних вхідних даних (рисунок 6.5 – 6.7).

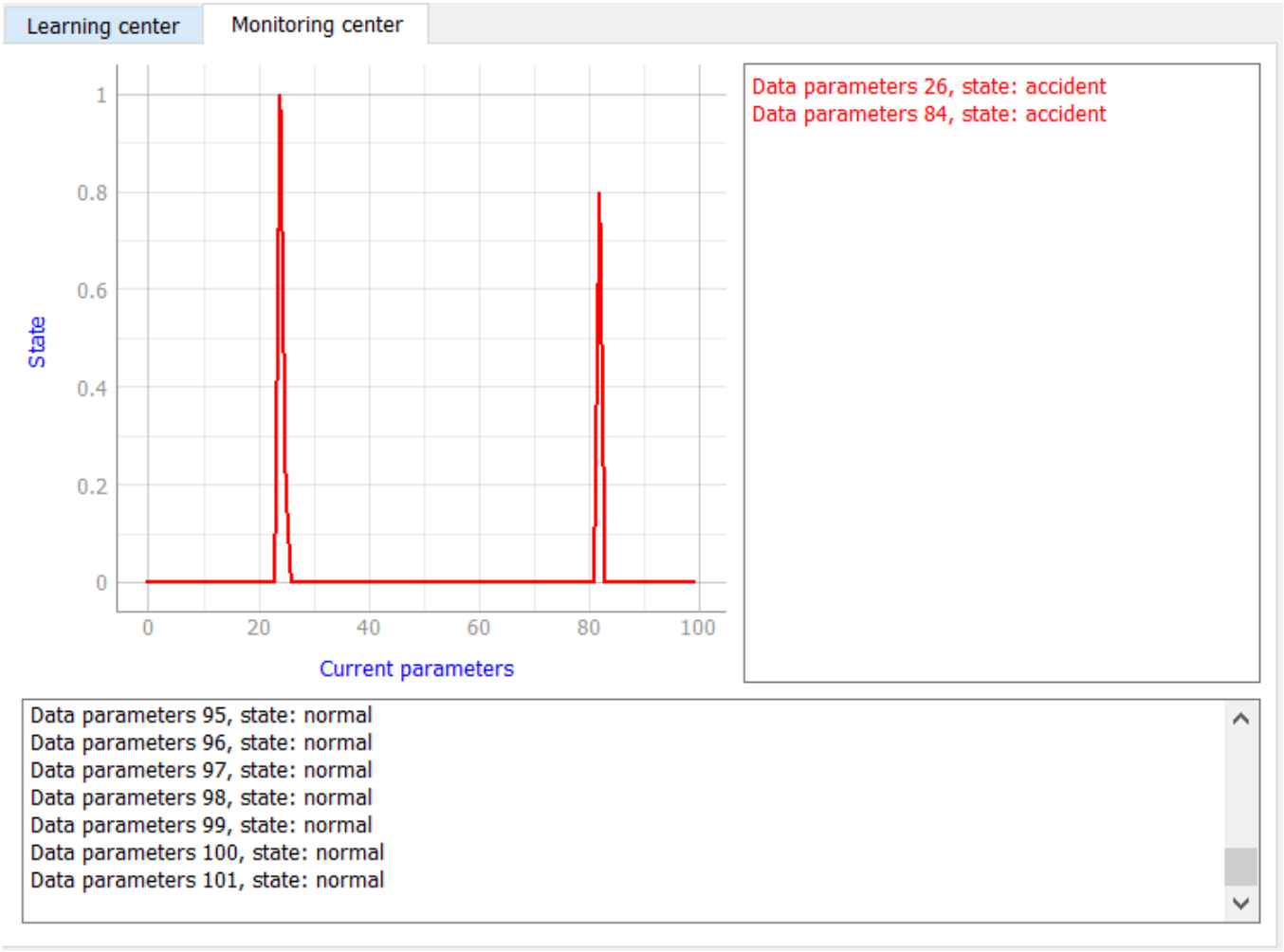

Рисунок 6.5 – Результат роботи. Тестові дані №1

Після завершення обробки нового набору даних система у правому текстовому полі вивела ідентифіковані, як аварійні, ситуації із вказаним номером набору параметрів.

На графіку візуалізована залежність наборів параметрів, що надходять до системи для аналізу від стану об’єкта:

- Аварійні ситуації: значення стану більше 0.
- Нормальний стан об’єкту: значення стану дорівнює 0.

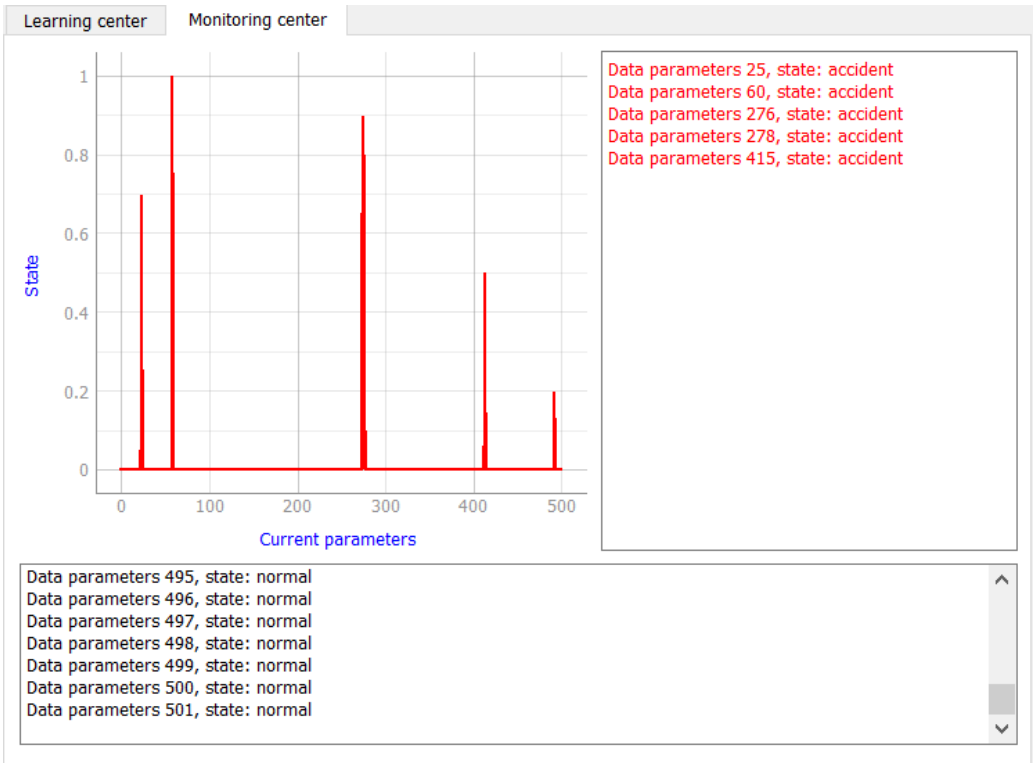

Рисунок 6.6 – Результат роботи. Тестові дані №2

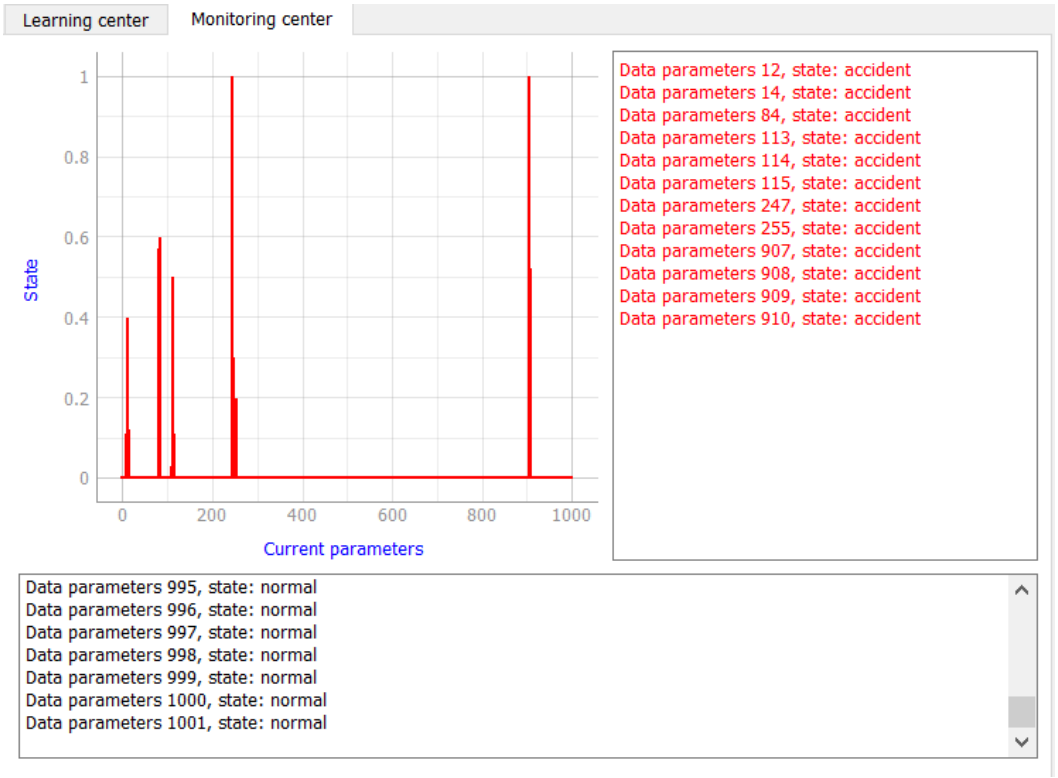

Рисунок 6.7 – Результат роботи. Тестові дані №3

Тестування системи завершено успішно. Створений програмний продукт точно виявляє аварійні ситуації з набору вхідних даних.

ВИСНОВКИ

У ході виконання дипломної роботи було:

1. Проаналізовано наявні архітектури та методи навчання нейронних мереж.
2. Визначено оптимальну модель нейронної мережі.
3. Обрано програмні та апаратні засоби для розробки.
4. Проведено тестування створеної системи.

Розроблено систему прогнозування стану складного технічного об’єкту з використанням нейронної мережі. В процесі функціонування складного технічного об’єкта відбувається взаємодія його різноманітних компонентів і фізичних процесів, що зумовлює складну поведінку як окремих елементів об’єкта так і всієї системи при нормальній експлуатації і, особливо, в аварійних ситуаціях.

Система призначена для інтелектуалізації та автоматизації процесів складного технічного об’єкта. Потенційними користувачами програмного комплексу є обслуговуючий персонал технічного об’єкта. Дана система може бути використана для виявлення аварійних ситуацій на складному технічному об’єкті.

Для розробки системи було проаналізовано відомі архітектури та методи навчання нейронних мереж. Було обрано модель багатошарового персептрону в якості архітектури ШНМ. Було сформовано оптимальну конфігурацію нейронної мережі використовуючи критерій Акаіке, а саме визначено кількість проміжних шарів, кількість нейронів в ПШ, кількість ітерацій навчання та коефіцієнт швидкості навчання. Обрано активаційну функцію сигмо їду, яка вдало підходить для задач класифікації. Для тренування мережі було проаналізовано та запрограмовано алгоритм зворотного поширення помилки.

Для програмної реалізації було обрано інтерпретовану об’єктно-орієнтовану мову програмування високого рівня зі строгою динамічною

типізацією Python версії 3.7, а також її бібліотеки:

- Pandas – для оперування даними.
- Scikitlearn – для побудови моделі нейронної мережі.
- Numpy – для проведення векторних обчислень.
- PyQt5 – для створення графічного інтерфейсу користувача.
- PyQtGraph – для візуалізації даних.

Тестування системи проводилось на основі різних наборів даних та параметрів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Каменев А.С., Королев С.Ю., Сокотущенко В.Н. Нейромоделирование как инструмент интеллектуализации энергоинформационных сетей / Под ред. В.В. Бушуева – М.: ИЦ “Энергия”, 2012. – 5 с.

2. Горбань А.Н., Россиев Д.А. Нейронные сети на персональном компьютере / А.Н. Горбань, Д.А. Россиев – Н.: Наука, 2006. – 276 с.

3. Миркес Е. М., Нейрокомпьютер. Проект стандарта. — Новосибирск: Наука, 1999. — 337 с.

4. В. В. Круглов, В. В. Борисов — Искусственные нейронные сети. Теория и практика — 11 с.

5. В. В. Круглов, В. В. Борисов — Искусственные нейронные сети. Теория и практика — 14 с.

6. Методология формирования профессиональныхнавыков в интеллектуальной компьютерной системе обучения с внешним объектом изучения / С.М. Аванский, С.В. Щербакова // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2009. – №1 (9). – 58 с.

7. Метод идентификации типа и прогнозирование последствий аварии на АЭС, Чыонг Ван Кхань Ньят, Москва – 2016. – 11 с.

ДОДАТОК А
СПЕЦИФІКАЦІЯ

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР62154	Записка.docx	Пояснювальна записка
Комплекс		
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТР62154	System of analysis and forecasting of the state of a complex technical object.exe	Застосунок системи

ДОДАТОК Б
ТЕКСТ ПРОГРАМНОГО МОДУЛЯ

Клас, який реалізує набір віджетів та їх параметри для головного вікна програми:

```
class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(1022, 763)
        self.centralwidget = QtWidgets.QWidget(MainWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
        self.tabWidget.setGeometry(QtCore.QRect(270, 10, 745, 551))
        self.tabWidget.setObjectName("tabWidget")
        self.tab = QtWidgets.QWidget()
        self.tab.setObjectName("tab")
        self.progressBar = QtWidgets.QProgressBar(self.tab)
        self.progressBar.setGeometry(QtCore.QRect(10, 490, 721, 23))
        self.progressBar.setProperty("value", 0)
        self.progressBar.setAlignment(QtCore.Qt.AlignCenter)
        self.progressBar.setTextVisible(True)
        self.progressBar.setOrientation(QtCore.Qt.Horizontal)
        self.progressBar.setInvertedAppearance(False)
        self.progressBar.setTextDirection(QtWidgets.QProgressBar.TopToBottom)
        self.progressBar.setObjectName("progressBar")
        self.tabWidget.addTab(self.tab, "")
        self.tab_2 = QtWidgets.QWidget()
        self.tab_2.setObjectName("tab_2")
        self.text_monitoring = QtWidgets.QPlainTextEdit(self.tab_2)
        self.text_monitoring.setGeometry(QtCore.QRect(10, 380, 721, 131))
        self.text_monitoring.setReadOnly(True)
        self.text_monitoring.setObjectName("text_monitoring")
        self.text_accident = QtWidgets.QPlainTextEdit(self.tab_2)
        self.text_accident.setGeometry(QtCore.QRect(430, 10, 301, 361))
        palette = QtGui.QPalette()
        brush = QtGui.QBrush(QtGui.QColor(255, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active, QtGui.QPalette.Text, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 0, 0, 128))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Active,
QtGui.QPalette.PlaceholderText, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 0, 0))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive, QtGui.QPalette.Text, brush)
        brush = QtGui.QBrush(QtGui.QColor(255, 0, 0, 128))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Inactive,
QtGui.QPalette.PlaceholderText, brush)
        brush = QtGui.QBrush(QtGui.QColor(120, 120, 120))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled, QtGui.QPalette.Text, brush)
        brush = QtGui.QBrush(QtGui.QColor(0, 0, 0, 128))
        brush.setStyle(QtCore.Qt.SolidPattern)
        palette.setBrush(QtGui.QPalette.Disabled,
QtGui.QPalette.PlaceholderText, brush)
        self.text_accident.setPalette(palette)
        self.text_accident.setReadOnly(True)
        self.text_accident.setObjectName("text_accident")
        self.tabWidget.addTab(self.tab_2, "")
        self.groupBox = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox.setGeometry(QtCore.QRect(10, 10, 251, 551))
        self.groupBox.setObjectName("groupBox")
        self.text_loaded_data = QtWidgets.QPlainTextEdit(self.groupBox)
```

```
        self.text_loaded_data.setGeometry(QtCore.QRect(10, 20, 231, 521))

self.text_loaded_data.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarAsNeede
d)
        self.text_loaded_data.setReadOnly(True)
        self.text_loaded_data.setObjectName("text_loaded_data")
        self.groupBox_2 = QtWidgets.QGroupBox(self.centralwidget)
        self.groupBox_2.setGeometry(QtCore.QRect(10, 570, 1001, 141))
        self.groupBox_2.setObjectName("groupBox_2")
        self.text_console = QtWidgets.QPlainTextEdit(self.groupBox_2)
        self.text_console.setGeometry(QtCore.QRect(10, 20, 981, 111))
        self.text_console.setReadOnly(True)
        self.text_console.setObjectName("text_console")
        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1022, 26))
        self.menubar.setObjectName("menubar")
        self.menuMenu = QtWidgets.QMenu(self.menubar)
        self.menuMenu.setObjectName("menuMenu")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)
        self.action_start_learning = QtWidgets.QAction(MainWindow)
        self.action_start_learning.setObjectName("action_start_learning")
        self.action_load_data = QtWidgets.QAction(MainWindow)
        self.action_load_data.setObjectName("action_load_data")
        self.action_set_parameters = QtWidgets.QAction(MainWindow)
        self.action_set_parameters.setObjectName("action_set_parameters")
        self.action_start_monitoring = QtWidgets.QAction(MainWindow)
        self.action_start_monitoring.setObjectName("action_start_monitoring")
        self.action_release_data = QtWidgets.QAction(MainWindow)
        self.action_release_data.setObjectName("action_release_data")
        self.action_clear_console = QtWidgets.QAction(MainWindow)
        self.action_clear_console.setObjectName("action_clear_console")
        self.action_abort_learning = QtWidgets.QAction(MainWindow)
        self.action_abort_learning.setObjectName("action_abort_learning")
        self.action_abort_monitoring = QtWidgets.QAction(MainWindow)
        self.action_abort_monitoring.setObjectName("action_abort_monitoring")
        self.action_exit = QtWidgets.QAction(MainWindow)
        self.action_exit.setObjectName("action_exit")
        self.menuMenu.addAction(self.action_start_learning)
        self.menuMenu.addAction(self.action_start_monitoring)
        self.menuMenu.addAction(self.action_load_data)
        self.menuMenu.addAction(self.action_set_parameters)
        self.menuMenu.addSeparator()
        self.menuMenu.addAction(self.action_release_data)
        self.menuMenu.addAction(self.action_clear_console)
        self.menuMenu.addAction(self.action_abort_learning)
        self.menuMenu.addAction(self.action_abort_monitoring)
        self.menuMenu.addSeparator()
        self.menuMenu.addAction(self.action_exit)
        self.menubar.addAction(self.menuMenu.menuAction())

        self.retranslateUi(MainWindow)
        self.tabWidget.setCurrentIndex(0)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)

    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab),
_translate("MainWindow", "Learning center"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_2),
```

```
_translate("MainWindow", "Monitoring center"))
        self.groupBox.setTitle(_translate("MainWindow", "Data files"))
        self.groupBox_2.setTitle(_translate("MainWindow", "Console"))
        self.menuMenu.setTitle(_translate("MainWindow", "Menu"))
        self.action_start_learning.setText(_translate("MainWindow",     "Start
learning"))
        self.action_load_data.setText(_translate("MainWindow", "Load data"))
        self.action_set_parameters.setText(_translate("MainWindow",       "Set
parameters"))
        self.action_start_monitoring.setText(_translate("MainWindow",   "Start
monitoring"))
        self.action_release_data.setText(_translate("MainWindow",     "Release
data"))
        self.action_clear_console.setText(_translate("MainWindow",      "Clear
console"))
        self.action_abort_learning.setText(_translate("MainWindow",     "Abort
learning"))
        self.action_abort_monitoring.setText(_translate("MainWindow",   "Abort
monitoring"))
        self.action_exit.setText(_translate("MainWindow", "Exit"))
```

Клас, який реалізує функціональні можливості віджетів, а також завантаження даних та задання параметрів:

```
class MainWindow(QtWidgets.QMainWindow):
    def __init__(self):
        super(MainWindow, self).__init__()
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.setWindowTitle("System of analysis and forecasting of the state
of a complex technical object")
        self.files = []
        self.amount_iterations = 0
        self.amount_layers = 0
        self.learning_speed = 0
        self.trained = False
        self.data = ""
        self.w = None
        self.ui.action_load_data.triggered.connect(self.load_data)
        self.ui.action_set_parameters.triggered.connect(self.get_parameters)

self.ui.action_start_learning.triggered.connect(self.learning_process)

self.ui.action_start_monitoring.triggered.connect(self.monitoring_process)
        self.ui.action_release_data.triggered.connect(self.release_data)
        self.ui.action_clear_console.triggered.connect(self.clear_console)
        self.ui.action_abort_learning.triggered.connect(self.abort_learning)

self.ui.action_abort_monitoring.triggered.connect(self.abort_monitoring)
        self.ui.action_exit.triggered.connect(self.exit)

        self.graphWidget = pg.PlotWidget(self.ui.tab)
        self.graphWidget.setGeometry(QtCore.QRect(10, 10, 720, 465))
        self.graphWidget.setBackground('w')

        self.graphWidget_2 = pg.PlotWidget(self.ui.tab_2)
        self.graphWidget_2.setGeometry(QtCore.QRect(10, 10, 410, 360))
        self.graphWidget_2.setBackground('w')
        self.clf = None

    def load_data(self):
        self.data = ""
```

```
        self.files = QFileDialog.getOpenFileNames(self, "Choose data files",
"", "All files (*);;CSV files (*.csv)")[0]

        self.ui.text_console.appendPlainText("Loading data files...")

        for path in self.files:
            file = open(path, "r")
            self.ui.text_console.appendPlainText("Loading
{0}...".format(path.split("/")[-1]))
            self.data += file.read()

        if self.data:
            temp_file = open("temp_file.csv", "w")
            temp_file.write(self.data)
            temp_file.close()

            self.data = pd.read_csv("temp_file.csv")

            self.data = self.data.values.tolist()

        self.ui.text_console.appendPlainText("Data files have been loaded!")
        self.ui.text_loaded_data.setPlainText("\n\n".join([file for file in
self.files]))

    def get_parameters(self):
        self.amount_iterations = QInputDialog.getInt(self, "Choose amount of
iterations", "Amount of iterations",
                                                     min=1, max=50000)[0]
        self.ui.text_console.appendPlainText("Amount of iterations was set to
" + str(self.amount_iterations))

        self.amount_layers = QInputDialog.getInt(self, "Choose amount of
layers", "Amount of layers",
                                                 min=1, max=5)[0]
        self.ui.text_console.appendPlainText("Amount of layers was set to " +
str(self.amount_layers))

        self.learning_speed = QInputDialog.getInt(self, "Choose learning
speed", "Learning speed / 10",
                                                  min=1, max=9)[0] / 10
        self.ui.text_console.appendPlainText("Learning speed was set to " +
str(self.learning_speed))
```

Функції, які реалізують алгоритм навчання та прогнозування стану складного технічного об’єкта:

```
def learning_process(self):
    if not self.files:
        QMessageBox.warning(self, "Warning!", "Load data!")
    elif not self.amount_iterations:
        QMessageBox.warning(self, "Warning!", "Set amount of iterations!")
    elif not self.amount_layers:
        QMessageBox.warning(self, "Warning!", "Set amount of layers!")
    elif not self.learning_speed:
        QMessageBox.warning(self, "Warning!", "Set learning speed!")
    else:
        self.ui.text_console.appendPlainText("Training...")

        for i in range(self.amount_iterations):
            self.ui.text_console.appendPlainText("Iteration {0} has been
completed".format(i + 1))
```

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

        t_in = [i[:-1] for i in self.data]
        t_out = [[i[1] for i in self.data]]
        a = [i[1] for i in self.data]

        training_inputs = np.array(t_in)

        training_outputs = np.array(t_out).T

        weights = 2 * np.random.random((len(self.data[0][:-1]),
self.amount_layers)) - 1

        alpha = self.learning_speed
        outputs = None
        arr = [0 for _ in range(self.amount_iterations)]

        for i in range(self.amount_iterations):
            outputs = sigmoid(np.dot(training_inputs, weights))

            error = training_outputs - outputs

            adjustment = error * outputs * (1 - outputs)

            arr[i] = error[0] ** 2 + error[1] ** 2 + error[2] ** 2

            weights += alpha * np.dot(training_inputs.T, adjustment)

        hidden_lyrs = tuple([len(self.data) for _ in
range(self.amount_layers)])

        self.clf = MLPClassifier(solver='lbfgs', alpha=self.learning_speed,
max_iter=self.amount_iterations,
                                 hidden_layer_sizes=hidden_lyrs,
                                 random_state=1, activation="logistic")
        self.clf.fit(t_in, a)

        self.w = weights
        pen = pg.mkPen(color=(255, 0, 0), width=2)
        self.graphWidget.setLabel('left', 'error', color='blue', size=30)
        self.graphWidget.setLabel('bottom', 'Iteration', color='blue',
size=30)
        self.graphWidget.showGrid(x=True, y=True)
        self.graphWidget.clear()
        self.graphWidget.plot([i for i in range(self.amount_iterations)],
[j[0] for j in arr], pen=pen)

        self.trained = True
        self.ui.progressBar.setValue(100)
        self.ui.text_console.appendPlainText("Training has been finished!")

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

    if not self.trained:
        QMessageBox.warning(self, "Warning!", "Training hasn't been
completed!")
    elif not self.files:
        QMessageBox.warning(self, "Warning!", "Load data!")
    else:
        self.ui.text_console.appendPlainText("Monitoring...")
```

```
        input_data = [i[:-1] for i in self.data]
        y = []

        for i in input_data:
            y.append(sigmoid(np.dot(i, self.w))[0])

        x = [i for i in range(len(self.data))]

        y = self.clf.predict(input_data)

        pen = pg.mkPen(color=(255, 0, 0), width=2)
        self.graphWidget_2.setLabel('left', 'State', color='blue', size=30)
        self.graphWidget_2.setLabel('bottom',      'Current      parameters',
color='blue', size=30)
        self.graphWidget_2.showGrid(x=True, y=True)
        self.graphWidget_2.plot(x, y, pen=pen)

        for i in range(len(y)):
            if y[i] > 0.05:
                self.ui.text_monitoring.appendPlainText("Data  parameters  " +
str(i + 2) +
                                                        ", state: accident")
                self.ui.text_accident.appendPlainText("Data  parameters  "  +
str(i + 2) +
                                                      ", state: accident"
                                                      )
            else:
                self.ui.text_monitoring.appendPlainText("Data  parameters  " +
str(i + 2) + ", state: normal")

        self.ui.text_console.appendPlainText("Monitoring has been finished!")
```

ДОДАТОК В
ОПИС ПРОГРАМНОГО МОДУЛЮ

АНОТАЦІЯ

Додаток присвячений опису програмного модуля розробленої системи прогнозування стану складного технічного об’єкта з допомогою нейронної мережі.

У роботі наведено загальні відомості про систему, її функціональне призначення та опис логічної структури, використовувані технічні засоби, вхідні та вихідні дані необхідні для функціонування системи.

Загальний об’єм роботи 12 сторінок, 5 рисунків.

Ключові слова: нейронна мережа, графічний інтерфейс користувача, метод зворотного поширення помилки, навчальна вибірка.

ЗМІСТ

1. **ЗАГАЛЬНІ ВІДОМОСТІ** **63**
2. **ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ**..... **64**
3. **ОПИС ЛОГІЧНОЇ СТРУКТУРИ** **65**
4. **ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ, ВИКЛИК І ЗАВАНТАЖЕННЯ**..... **67**
5. **ВХІДНІ ДАНІ** **69**
6. **ВИХІДНІ ДАНІ**..... **71**

1. ЗАГАЛЬНІ ВІДОМОСТІ

Розробленою системою є система прогнозування стану складного технічного об’єкта.

Для функціонування системи необхідний персональний комп’ютер зі встановленою операційною системою Windows 10.

Для написання програмного коду було використано об’єктно-орієнтовану мову програмування високого рівня зі строгою динамічною типізацією Python версії 3.7, а також її бібліотеки:

- Pandas – для оперування даними.
- Scikitlearn – для побудови моделі нейронної мережі.
- Numpy – для проведення векторних обчислень.
- PyQt5 – для створення графічного інтерфейсу користувача.
- PyQtGraph – для візуалізації даних.

2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Система призначена для інтелектуалізації та автоматизації процесів складного технічного об’єкта. Потенційними користувачами програмного комплексу є обслуговуючий персонал технічного об’єкта.

В основі системи закладена нейронна мережа, яка виконує задачу класифікації аварій, які можуть виникнути під час експлуатації об’єкта.

3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Для навчання нейронної мережі використовується алгоритм зворотного поширення помилки.

На рисунку В.3.1 зображено об’єктно-орієнтоване представлення програми.

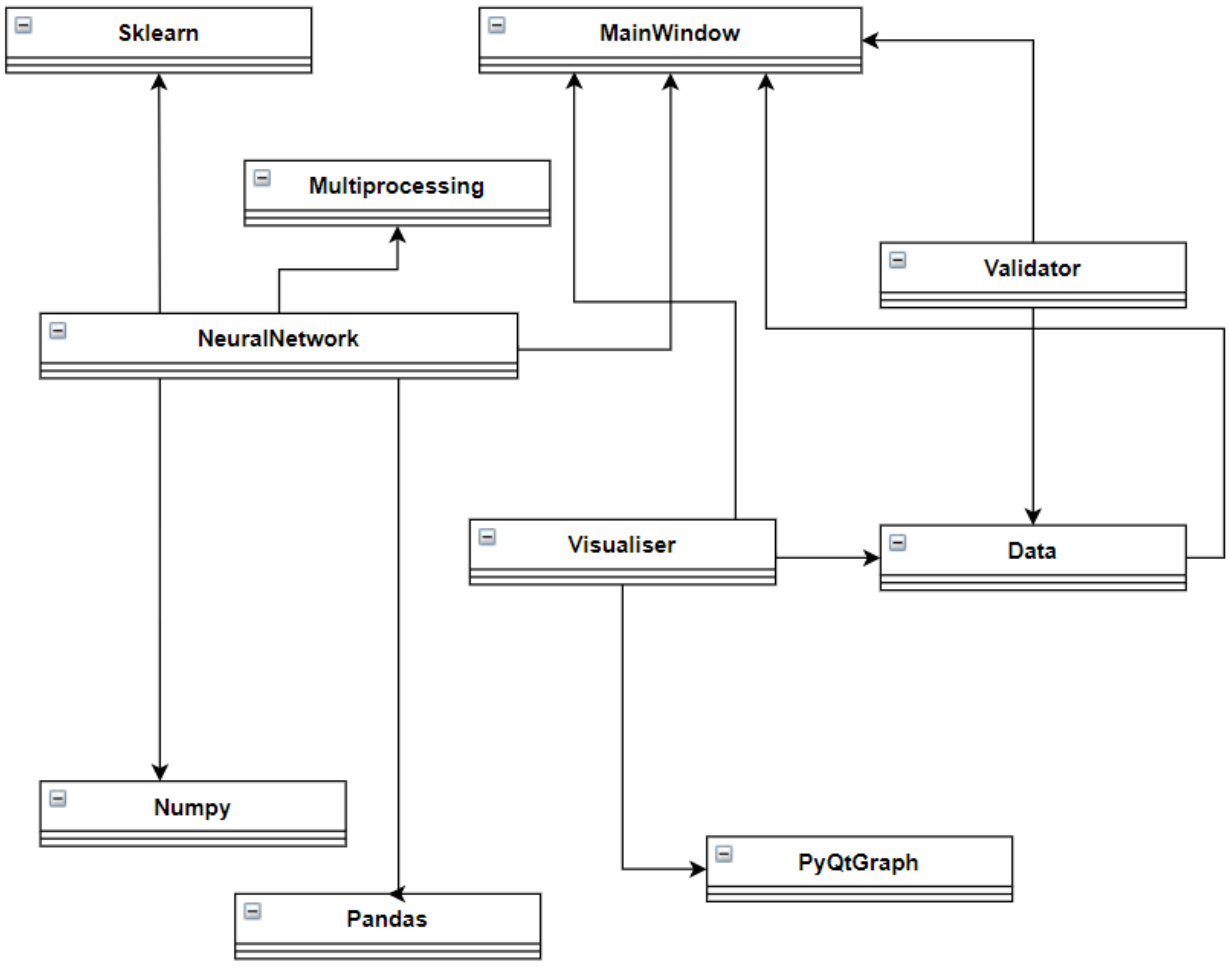

Рисунок В.3.1 – Об’єктно-орієнтоване представлення програми

Клас NeuralNetwork містить в собі реалізацію штучної нейронної мережі. Набір атрибутів класу містить:

- Кількість ітерацій для навчання мережі.
- Кількість прихованих шарів.

- Ідентифікатор натренованості мережі.
- Параметр швидкості навчання мережі.

Методи класу:

- Запуск навчання нейронної мережі.
- Зупинка навчання у випадку зростання помилки або невірності вхідних даних.
- Застосування навченої ШНМ до аналізу нових даних, прогнозуючи стан складного технічного об’єкту.

Клас MainWindow є реалізацію графічного інтерфейсу системи, набір атрибутів представляє список віджетів, використаних у розробці користувацького інтерфейсу. Функціонал кожного віджета під’єднано за допомогою сигналів до методів класу:

- Обрати джерела даних.
- Задати параметри для навчання нейронної мережі.
- Очистити консоль.
- Вийти з системи.

Також керуючі віджети графічного інтерфейсу з’єднані з методами класу NeuralNetwork та Data надаючи можливість завантажувати дані, розпочинати навчання тощо.

Клас Validator надає можливість перевірки вхідних даних, обмежуючи дії користувача які можуть призвести до неправильної роботи системи.

Клас Visualiser надає функціонал для візуалізації процесу навчання та процесу аналізу і прогнозування стану складного технічного об’єкта.

4. ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ, ВИКЛИК І ЗАВАНТАЖЕННЯ

Системні вимоги для запуску:

- Процесор Intel Celeron/Pentium.
- Розмір оперативної пам'яті 4 Гб.
- Вільного дискового простору 1 Гб.
- Операційна система Windows 10.

Вище зазначені системні вимоги розраховані на роботу з невеликими масивами даних, тому зі збільшенням кількості даних, збільшенням кількості прихованих шарів та кількості ітерацій зростають вимоги до апаратного забезпечення. Тому рекомендованими вимогами для оптимальної експлуатації системи прогнозування стану складного технічного об’єкта є:

- Процесор Intel Core i5 8-го покоління.
- 16 Гб оперативної пам’яті.
- Вільного дискового простору 1 Гб.
- Операційна система Windows 10.

Для запуску системи необхідно виконати System of analysis and forecasting of the state of a complex technical object.exe (рисунок В.4.1).

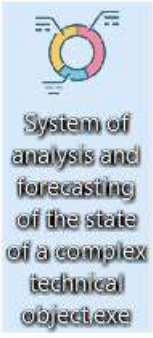

Рисунок В.4.1 – Застосунок системи

Після запуску застосунку на екрані монітору з’явиться головне вікно системи (рисунок В.4.2).

Рисунок В.4.2 – Головне вікно системи

Розглянемо його основні елементи:

6. Центр навчання.
7. Центр моніторингу, аналізу, прогнозування.
8. Область даних.
9. Консоль.
10. Меню.

5. ВХІДНІ ДАНІ

Вхідними даними для системи є набори даних у форматі comma-separated values (*.csv). Для навчання нейронної мережі було використану навчальну вибірку (рисунок В.5.1).

Температура навколишнього середовища	Температура охолоджуючої рідини	Напруга d-компоненту	Напруга q-компоненту	Швидкість мотору	Момент обертання	Температура поверхні постійного магніта	Температура ярма статора
-0,75214297	-1,1184461	0,3279352	-1,2978575	-1,2224282	-0,2501821	-2,522071	-1,8314217
-0,77126324	-1,1170206	0,3296648	-1,2976865	-1,2224293	-0,2491333	-2,5224178	-1,8309687
-0,78289163	-1,1166813	0,3327715	-1,3018217	-1,2224278	-0,2494311	-2,5226731	-1,8304
-0,78093535	-1,1167642	0,3336999	-1,301852	-1,2224301	-0,2486364	-2,521639	-1,8303328
-0,7740426	-1,116775	0,3352061	-1,303118	-1,2224286	-0,2487008	-2,5219002	-1,8304977
-0,7629362	-1,1169548	0,33490124	-1,3030168	-1,2224286	-0,248197	-2,522203	-1,8319309
-0,74922806	-1,1161705	0,33501354	-1,3020816	-1,2224296	-0,2479142	-2,5225377	-1,8330117
-0,7384499	-1,1139864	0,3362563	-1,3051548	-1,2224321	-0,248321	-2,5228438	-1,8321822
-0,7309097	-1,1118276	0,3349053	-1,3037902	-1,2224315	-0,2477847	-2,5228078	-1,8315759
-0,72712964	-1,1094859	0,3359881	-1,3056333	-1,2224314	-0,2482944	-2,5226767	-1,8314383
-0,72371286	-1,1082836	0,33539978	-1,3045602	-1,2224283	-0,247913	-2,5226302	-1,8314928
-0,71775365	-1,1085879	0,33443072	-1,3043374	-1,2224288	-0,2477212	-2,5226388	-1,8318189
-0,7048766	-1,1099857	0,33624125	-1,30582	-1,2224338	-0,2480017	-2,5222142	-1,8320519
-0,6825343	-1,1113616	0,33556557	-1,3032398	-1,2224305	-0,2475126	-2,5219264	-1,8320305
-0,654583	-1,1126	0,33655748	-1,3062165	-1,2224299	-0,2480666	-2,5221426	-1,8324168
-0,63071096	-1,1136569	0,33535197	-1,3055588	-1,2224302	-0,2475905	-2,5222142	-1,8320504
-0,62009937	-1,1142852	0,33590913	-1,3056751	-1,2224299	-0,2478845	-2,5223105	-1,8317783
-0,6148578	-1,1145984	0,33629027	-1,3066177	-1,2224327	-0,2479104	-2,5216212	-1,8313637
-0,60744816	-1,1150228	0,33301646	-1,2344021	-1,1908675	-0,3074368	-2,5201533	-1,8313625
-0,6037373	-1,1158898	0,32455894	-0,9830486	-1,0761522	-0,3482398	-2,520239	-1,8324342
-0,60323447	-1,1150343	0,316031	-0,63691336	-0,9152268	-0,3762608	-2,5206366	-1,8327438
-0,6004591	-1,1119267	0,3084021	-0,26627913	-0,740405	-0,3997889	-2,521099	-1,8335521
-0,59830505	-1,1096134	0,30417806	0,09185752	-0,5674627	-0,3770513	-2,5225427	-1,8346876
-0,5997665	-1,1090795	0,29940167	0,47535366	-0,3773778	-0,3842986	-2,5232556	-1,8341513
-0,60371727	-1,1082681	0,29658678	0,87474203	-0,1744461	-0,4046426	-2,5230372	-1,8329986
-0,6035687	-1,1073983	0,29612818	1,1744009	-0,0112797	-0,3678239	-2,522864	-1,8318759
-0,6025639	-1,1054107	0,290501	1,3467971	0,1055844	-0,3379655	-2,522989	-1,8307649
-0,59803104	-1,102991	0,27885312	1,4665564	0,2155701	-0,3736125	-2,522904	-1,8299689

Рисунок В.5.1 – Набір даних для навчання нейронної мережі

Параметри навчальної вибірки:

- Формат файлу: comma-separated values (*.csv).
- Розмір файлу: 130026 Кб
- Кількість параметрів: 8.
    - Температура навколишнього середовища.
    - Температура охолоджуючої рідини
    - Напруга d-компоненту

  - Напруга q-компоненту
  - Швидкість мотору
  - Момент обертання
  - Температура постійного магніту
  - Температура ярма статора
- Кількість наборів параметрів: 998070

11. ВИХІДНІ ДАНІ

Вихідними даними системами є результати прогнозування стану складного технічного об’єкта (рисунок В.6.1).

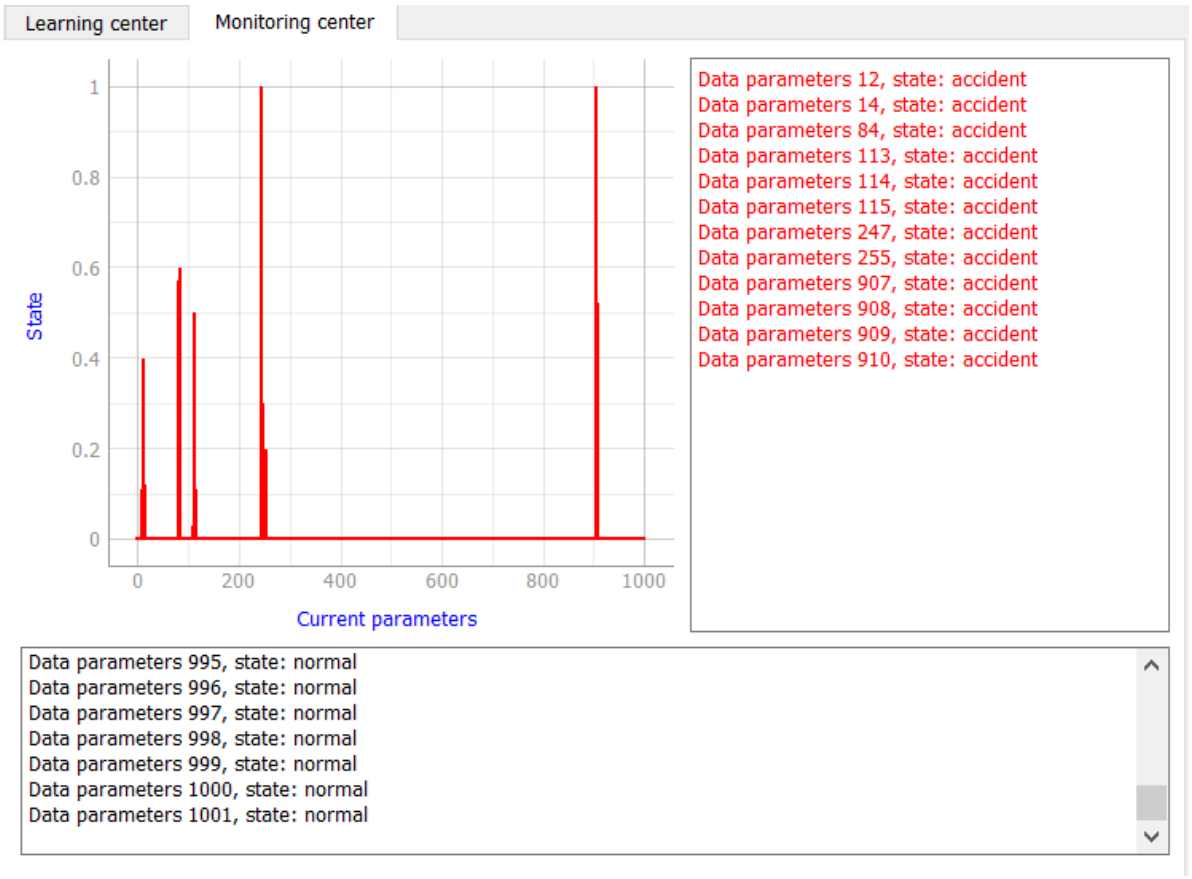

Рисунок В.6.1 – Результат роботи

Після завершення обробки вхідного набору даних система у правому текстовому полі вивела ідентифіковані, як аварійні, ситуації із вказаним номером набору параметрів.

На графіку візуалізована залежність наборів параметрів, що надходять до системи для аналізу від стану об’єкта:

- Аварійні ситуації: значення стану більше 0.
- Нормальний стан об’єкту: значення стану дорівнює 0.