Міністерство освіти і науки України
Державний ВНЗ «Національний гірничий університет»

Факультет інформаційних технологій
(факультет)

Кафедра програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
дипломної роботи

магістра
(назва освітньо-кваліфікаційного рівня)

галузь знань *12 Інформаційні технології*
(шифр і назва галузі знань)

спеціальність *122 Комп'ютерні науки*
(код і назва спеціальності)

спеціалізація *Інформаційні управляючі системи та технології*
(назва спеціалізації)

освітній рівень *магістр*
(назва освітнього рівня)

кваліфікація *інженер з комп'ютерних систем*
(назва кваліфікації)

на тему: *Удосконалення застосування нейронних мереж при розпізнаванні нештатних ситуацій на шахтних підйомних машинах*

Виконавець:
студент *2* **курсу , групи** *122М-16-1*

(підпис) *Піменов О.А.* (прізвище та ініціали)

Керівники	Посада, прізвище, ініціали	Оцінка	Підпис
проекту	*проф. Корнієнко В.І.*		
розділів:			
Спеціальний	*проф. Мещеряков Л.І.*		
Економічний	*доц. Касьяненко Л.В.*		

Рецензент	*проф. Зеленцов Д.Г.*		

Нормоконтроль	*доц. Коротенко Л.М.*		

Дніпропетровськ
2018

Міністерство освіти і науки України
Державний вищий навчальний заклад
«Національний гірничий університет»

ЗАТВЕРДЖЕНО:
завідувач кафедри

програмного забезпечення комп'ютерних систем
(повна назва)

_____ І.М. Удовик
(підпис) (прізвище, ініціали)

« » _____ 20 ____ року

ЗАВДАННЯ
на виконання кваліфікаційної роботи магістра

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

студенту *122М-16-1* *Піменов О.А.*
(група) (прізвище та ініціали)

Тема дипломної роботи *Удосконалення застосування нейронних мереж при розпізнаванні нештатних ситуацій на шахтних підйомних машинах*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора Державного ВНЗ «НГУ» від *26.12.2017 р.* № *2127 -л*

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – розпізнавання нештатних станів на шахтних підйомних машинах.

Предмет досліджень – можливість застосування нейронної мережі Кохонена при розпізнаванні нештатних станів на шахтних підйомних машинах

Мета НДР – удосконалення застосування нейронних мереж для розпізнавання нештатних ситуацій на шахтних підйомних машинах використовуючи нейронну мережу Кохонена.

Вихідні дані для проведення роботи – теоретичні й експериментальні дослідження, сигнали шахтної підйомної машини на різних етапах її роботи.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Актуальність даної теми зумовлена наявністю значних недоліків у існуючих системах контролю шахтних підйомних машинах при розпізнаванні нештатних станів: застарілі системи, їх відсутність.

Наукова новизна результатів, що очікуються, полягає у обґрунтуванні застосування нейронної мережі Кохонена, яка допоможе слідкувати за станом шахтної підйомної машини і розпізнавати нештатні стани шахтної підйомної машини.

Практична цінність результатів полягає у розробці алгоритму, який використовуючи результат нейронної мережі Кохонена дозволяє розпізнавати нештатні стани шахтних підйомних машин.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати магістерської роботи повинні відповідати вимогам паспорту наукової спеціальності 05.13.06 – «Інформаційні технології».

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити безпосереднє використання розробленої нейронної мережі. Згідно виробничих функцій та професійних задач магістра, повинна бути розроблена ефективна нейронна мережа для розпізнавання нештатних станів на шахтних підйомних машинах.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок-кінець)**
1	2
Аналіз стану питання застосування нейронних мереж при розпізнаванні нештатних станів на шахтних підйомних машинах	10.09.2017 25.09.2017
Розробка алгоритму який дозволить розпізнавати нештатні стани шахтних підйомних машин використовуючи результати нейронної мережі Кохонена	30.09.2017 18.10.2017
Конфігурація, тестування та проведення експериментів для отримання результатів розробленого алгоритму	25.10.2017 23.11.2017

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивним завдяки скорочення часу на розпізнавання нештатних станів на шахтних підйомних машинах.

Соціальний ефект від реалізації результатів роботи очікується позитивним завдяки удосконаленню застосування нейронних мереж при розпізнаванні нештатних ситуацій на шахтних підйомних машинах.

7 ДОДАТКОВІ ВИМОГИ

Відповідність оформлення ДСТУ 3008-95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.

Завдання видав _____ *Мещеряков Л.І.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Піменов О.А.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *09.09.2017р.*

Термін подання дипломного проекту до ДЕК *23.01.2018*

Реферат

Пояснительная записка: 85 с., 28 рис., 3 приложений., 55 источника.

Объект исследования: распознавания нештатных ситуаций на шахтных подъемных машинах.

Цель магистерской работы: совершенствование применения нейронных сетей для распознавания нештатных ситуаций на шахтных подъемных машинах используя нейронную сеть Кохонена.

Методы исследования: При решении поставленной задачи использовались научные достижения в областях разработки информационных систем и программного обеспечения.

Научная новизна результатов, которые ожидаются, заключается в обосновании применения нейронной сети Кохонена, которая поможет следить за состоянием шахтной подъемной машины и распознавать нештатные состояния шахтной подъемной машины.

Практическая ценность заключается в разработке метода применения нейронной сети Кохонена для распознавания нештатных состояний шахтной подъемной машины.

Область применения. Разработанная информационная система может применяться для решения задач распознавания нештатных состояний на шахтных подъемных машинах.

Значение работы и выводы. Усовершенствованная нейронная сеть позволяет распознавать нештатные состояния на шахтных подъемных машинах со значительным сокращением временных и ресурсных затрат системы контроля, подтверждается разработанным программным продуктом в данной магистерской работе.

Прогнозы по развитию исследований. Разработать универсальные программные модули, которые могут быть использованы для распознавания и предсказания нештатных ситуаций на шахтных подъемных машинах, которые смогут полностью или уменьшить участие человека в контроле шахтной подъемной машиной при возникновении нештатных состояний.

В разделе «Экономика» проведены расчеты трудоемкости разработки программного обеспечения, затрат на создание ПО и длительности его разработки, а также провести маркетинговые исследования рынка сбыта созданного программного продукта.

Список ключевых слов: НЕЙРОННЫЕ СЕТИ, КЛАСТЕРИЗАЦИЯ, САМООРГАНИЗУЮЩАЯСЯ КАРТА КОХОНЕНА, ШАХТНЫЕ ПОДЪЕМНЫЕ МАШИНЫ, НЕШТАТНАЯ СИТУАЦИЯ.

Реферат

Пояснювальна записка: 85 с., 28 рис., 3 додатків., 55 джерела.

Об'єкт дослідження: розпізнавання нештатних ситуацій на шахтних підйомних машинах.

Мета магістерської роботи: удосконалення застосування нейронних мереж для розпізнавання нештатних ситуацій на шахтних підйомних машинах використовуючи нейронну мережу Кохонена.

Методи дослідження: При рішенні поставленої задачі використовувалися наукові досягнення в областях розробки інформаційних систем та програмного забезпечення.

Наукова новизна результатів, що очікуються, полягає у обґрунтуванні застосування нейронної мережі Кохонена, яка допоможе слідкувати за станом шахтної підйомної машини і розпізнавати нештатні стани шахтної підйомної машини.

Практична цінність результатів полягає у розробці метода застосування нейронної мережі Кохонена для розпізнавання нештатних станів шахтної підйомної машини.

Область застосування. Розроблена інформаційна система може застосовуватися для вирішення завдань розпізнавання нештатних станів на шахтних підйомних машинах.

Значення роботи та висновки. Удосконалена нейрона мережа дозволяє розпізнавати нештатні стани на шахтних підйомних машинах зі значним скороченням часових і ресурсних витрат системи контролю, що підтверджується розробленим програмним продуктом в даній магістерській роботі.

Прогнози щодо розвитку досліджень. Розробити універсальні програмні модулі, які можуть бути використані для розпізнавання і передбачення нештатних ситуацій на шахтних підйомних машинах, які зможуть повністю або зменшити участь людини у контролі шахтною підйомною машиною при виникненні нештатних станів.

У розділі «Економіка» проведені розрахунки трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки, а також провести маркетингові дослідження ринку збуту створеного програмного продукту.

Список ключових слів: НЕЙРОННА МЕРЕЖА, КЛАСТЕРІЗАЦІЯ, САМООРГАНІЗАЦІЙНА КАРТА КОХОНЕНА, ШАХТНІ ПІДЙОМНІ МАШИНИ, НЕШТАТНА СИТУАЦІЯ.

The abstract

Explanatory note: 85 pp., 28 fig. 3 applications. 55 sources.

Object of research: detection of emergency situations in mine lifting machines.

The purpose of the degree project: improving the use of neural networks to detect emergency situations in mine lifting machines using Kohonen's neural network.

Research methods: When solving the task used the scientific achievements in the areas of information systems and software.

The scientific novelty expected results, is justifying the use of Kohonen's neural network to help monitor the status of mine lifting machine and detect emergency situations of mine lifting machines.

The practical value of work: Is develop a method for using the Kohonen's neural network result to detect emergency situations in mine lifting machines.

The scope. The developed information system can be used to solve problems of detect emergency situations in mine lifting machines.

The value of the work and conclusions. The advanced neural network allows you to recognize emergency situations on mine lifting machines, with a significant reduction in the time and resource costs of the control system, is confirmed by the developed software product in this master's work.

Projections on development research. Develop universal software modules that can be used to recognize and predict emergency situations in mine lifting machines that can completely or reduce a person's participation in the control of the mine lifting machines in the event of emergency situations.

In section "Economics". Calculations were made of the complexity of software development, the cost of creating software and the duration of its development, as well as to conduct market research of the market for the created software product.

List of keywords: NEURAL NETWORKS, CLUSTERING, SELF-ORGANIZING MAP, MINE LIFTING MACHINE, EMERGENCY SITUATION.

ЗМІСТ

ВСТУП

Актуальність роботи. Актуальність даної роботи зумовлена наявністю недоліків у існуючих системах контролю шахтних підйомних машин при розпізнаванні нештатних станів, застарілість систем, або відсутність сучасних засобів контролю.

Аналізуючи існуючу системи контролю шахтних підйомних машин в Україні стає зрозумілим, що використання сучасних технологій для розпізнавання нештатних ситуацій майже вістуне. Використання нейронних мереж, як правило, обмежується аналізом отриманих даних в результаті роботи шахтних підйомних машин. Виникає необхідність удосконалення застосування нейронних мереж при розпізнаванні нештатних ситуацій на шахтних підйомних машинах.

Для удосконалення використання нейронних мереж при розпізнаванні нештатних ситуацій на шахтних підйомних машинах була обрана нейронна мережа Кохонена.

Штучна нейронна мережа Кохонена або самоорганізована карта ознак (SOM) була запропонована фінським дослідником Тойво Кохоненом на початку 1980-х років.

Нейронні мережі Кохонена типовий приклад нейромережевої архітектури, яка навчається без учителя. Звідси і перелік вирішуваних ними завдань: кластеризація даних або прогнозування властивостей. Крім того, мережі Кохонена можуть використовуватися з метою зменшення розмірності даних з мінімальною втратою інформації.

Більшість нейронних мереж навчаються з учителем на вибірках даних, що включають безліч прикладів, що складаються з відповідних один одному пар вхідних і вихідних векторів. При цьому вихідні значення брали безпосередню участь в налаштуванні вагових коефіцієнтів. У нейронних мережах Кохонена вихідні вектора в навчальній вибірці можуть бути, але можуть бути і відсутніми, і, в будь-якому випадку, вони не беруть участі в процесі навчання. Тобто виходи не використовуються в якості орієнтирів при корекції синапсів. Саме тому даний принцип настройки нейронної мережі називається самонавчанням.

У розглянутій архітектурі сигнал поширюється від входів до виходів в прямому напрямку. Структура нейронної мережі містить єдиний шар нейронів (шар Кохонена) без коефіцієнтів зміщення.

Кількість нейронів дорівнює кількості кластерів, серед яких відбувається початкове розподіл і подальше перерозподіл навчальних прикладів. Кількість вхідних змінних нейронної мережі дорівнює числу ознак, що характеризують об'єкт дослідження і на основі яких відбувається віднесення його до одного з кластерів.

Слід розрізняти власне самонавчання і самоорганізацію нейронної мережі Кохонена. При звичайному самообучении мережу має строго фіксовану структуру, тобто кількість нейронів, що не змінюється протягом усього життєвого циклу. При самоорганізації мережу, навпаки, не має постійної структури. Залежно від знайденого відстані до нейрона-переможця або цей нейрон використовується для кластеризації прикладу, або для поданого на входи прикладу створюється новий кластер з відповідними йому ваговими коефіцієнтами. Крім того, в процесі самоорганізації структури мережі Кохонена окремі нейрони можуть виключатися з неї.

В результаті дослідницької роботи був розроблений алгоритм, який використовуючи результат навчання нейронної мережі Кохонена дозволяє розпізнавати нештатні ситуації на шахтних підйомних машинах.

Метою та задачею дослідження. Метою даної магістерської роботи є удосконалення застосування нейронних мереж для розпізнавання нештатних ситуацій на шахтних підйомних машинах використовуючи нейронну мережу Кохонена.

Для досягнення поставленої мети в роботі сформульовані та вирішені наступні задачі:

1. Проведення аналізу та виявлення методів впровадження досліджень розпізнавання нештатних станів на шахтних підйомних машинах.

2. Розроблено алгоритм, який використовуючи результат навчання нейронної мережі Кохонена дозволяє розпізнати нештатний стан шахтних підйомних машин.

Об'єкт дослідження – розпізнавання нештатних станів на шахтних підйомних машинах.

Предмет дослідження – можливість застосування нейронної мережі Кохонена при розпізнаванні нештатних станів на шахтних підйомних машинах.

Ідея роботи полягає в підвищенні ефективності роботи систем контролю стану шахтних підйомних машин.

Методи дослідження. При вирішенні поставленого завдання використовувалися наукові досягнення в областях розробки інформаційних систем і програмного забезпечення.

Наукові положення, очікувані наукові результати.

1. Проведення аналізу та виявлення методів розпізнавання нештатних ситуацій на шахтних підйомних машинах.
2. Розробка методу розпізнавання нештатних станів шахтних підйомних машин використовуючи нейрону мережу Кохонена.

Обґрунтованість і достовірність наукових положень

Обґрунтованість і достовірність наукових положень, висновків та рекомендацій магістерської роботи обґрунтована коректністю поставлених проблем та прийнятих допущень при аналітичному огляді процесів, обґрунтованістю вихідних посилок, достатнім об'ємом вибірки даних та підтвердженими на модельних об'єктах результатами аналізу.

Наукова новизна отриманих результатів отриманих результатів полягає у обґрунтуванні застосування нейронної мережі Кохонена, яка допоможе слідкувати за станом шахтної підйомної машини і розпізнавати нештатні стани.

Практичне значення отриманих полягає у розробці алгоритму який, використовуючи результат нейронної мережі Кохонена дозволяє розпізнавати нештатні стани шахтних підйомних машин.

Зв’язок роботи з державними програмами, планами науково-дослідницьких робіт.

Результати дипломної роботи можуть бути використані підприємствами в гірській промисловості для контролю стану шахтних підйомних машин.

Особливий вклад магістра полягає в:

- обранні методів досліджень та технологій реалізації;
- розробки метода розпізнавання нештатних станів шахтних підйомних машин використовуючи результат нейронної мережі Кохонена.
- розробці теоретичної частини роботи, в якій досліджено та систематизовано знання про можливості використання нейронної мережі Кохонена для розпізнавання нештатних станів на шахтних підйомних машинах
- оцінці отриманих результатів.

Апробація результатів магістерської роботи.

Основні положення та результати повідомлені та обговорені на студентській науковій конференції.

Структура та об’єм роботи. Робота складається з вступу, трьох розділів та висновків. Складається з 85 сторінок печатного тексту, в тому числі 51 сторінок тексту основної частини з 28 рисунками, переліку використаних посилань з 55 найменуваннями на 5 сторінках, 3 додатки на 16 сторінках.

РОЗДІЛ 1.

АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Шахтні підйомні установки

У практиці розробки родовищ корисних копалин підземним способом застосовуються шахтні підйомні установки з розташуванням підйомних машин на земній поверхні (Рис. 1.1, а) або на баштовому копрі (Рис. 1.1, б). Кожна шахтна підйомна установка складається

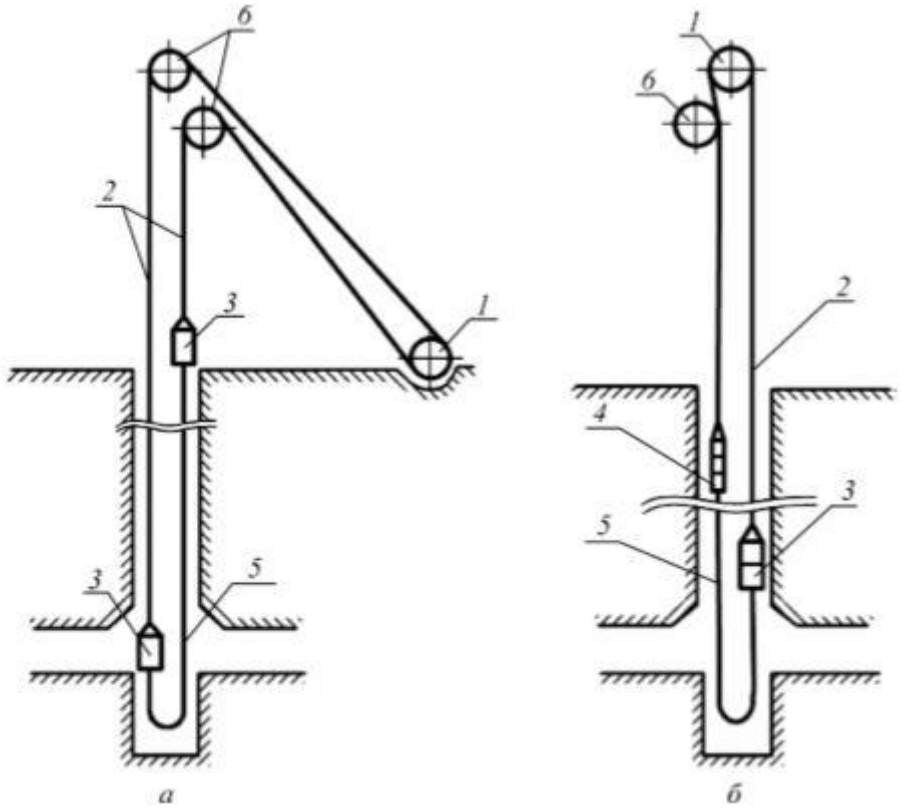

Рис. 1.1. Схеми розміщення обладнання вертикальних шахтних підйомних установок: а - барабанної; б - многоконатной

з шахтної підйомної машини 1, підйомних канатів 2, до яких підвішені підйомні посудини 3 або противагу 4. В підйомних посудинах - скіпах здійснюється підйом корисної копалини. Спуск-підйом людей, матеріалів і обладнання проводять в коморах. На глибоких одноканатних шахтних підйомних установках до судин може бути підвішений хвостовій (врівноважує)

канат 5. Всі багатоканатні підйомні установки оснащені уравновешивающими канатами. Для забезпечення необхідних зазорів між підйомними посудинами, між судинами і кріпленням ствола на копрі встановлені відхиляють шківи 6. багатоканатні підйомні машини можуть бути встановлені як на земній поверхні, так і на баштовому копрі. В останні роки більшість підйомних установок проектується з установкою підйомної машини на земній поверхні.

Шахтні підйомні установки класифікують за такими основними ознаками:

За призначенням:

а) головні - призначені тільки для підйому корисної копалини;

б) допоміжні:

- людські;

- грузолюдскі;

- вантажні.

Ці установки забезпечують виконання операцій по спуску-підйому людей, матеріалів і устаткування;

1.2. Шахтна підйомна машина

Шахтна підйомна машина- складна механічна система, яка складається з ряду зосереджених мас (судини, органи навивки, левередж, двигун, шківи), з'єднаних пружними елементами (канати, валопроводи, пружинні муфти).

Шахтні підйомні машини забезпечують видачу корисних копалин, переміщення людей і вантажів. Від надійності роботи цієї найважливішої ланки технологічного ланцюга залежить безперебійність роботи всього гірничодобувного підприємства. Будь-яка аварійна ситуація на підйомі веде до

зупинки підприємства. Тому питань забезпечення надійності та безпеки експлуатації шахтних підйомних установок завжди приділяли особливу увагу.

За останні роки парк шахтних підйомних машин сильно постарів. Термін служби більшості з них перевищує 25 років. Такий же термін служби мають привід шахтних підйомних машин, система управління цим приводом, обладнання шахтного стовбура, стовбурова сигналізація та інші, життєво важливі елементи шахтних підйомних установок. У зв'язку з тим, що одночасна заміна всіх шахтних підйомних машин та інших елементів шахтних підйомних установок неможлива, вельми актуальним є завдання застосування сучасних технологій для забезпечення надійності та безпеки експлуатації шахтних підйомних установок.

У той же час рівень автоматизації шахтних підйомних установок залишається досить низьким, внаслідок чого знижується якість контролю параметрів і оперативності спрацьовування захистів від неприпустимих режимів роботи, знижується надійність їх роботи, а також ефективність самого технологічного процесу в цілому.

У зв'язку з особливою технологічної важливістю шахтних підйомних установок в шахтному виробництві, великими розмірами збитку від аварій, а також високим рівнем вимог до безпеки підйому людей для управління і контролю такими установками допустимо застосування лише автоматизованих систем, які передбачають обов'язкову участь людини. У міру вдосконалення систем автоматизації шахтних підйомних установок обсяг функцій, виконуваних людиною, неухильно зменшувався, і в даний час за ним залишилися функції, що важко піддаються автоматизації, або більш надійно виконуються людиною в штатної і особливо в нештатної ситуації, а також функції загального контролю ситуації і прийняття екстрених заходів.

Одним з ефективних заходів дозволяють розпізнати майбутню нештатну ситуацію, або неналежний стан ШПУ в автоматичному режимі і прийняти перші заходи попередження - є нейронні мережі.

1. 3. Історія появи нейронних мереж

Термін «нейронна мережа» з'явився в середині XX століття. Перші роботи, в яких були отримані основні результати в даному напрямку, були пророблені Мак-Каллоком і Питтсом. У 1943 році ними була розроблена комп'ютерна модель нейронної мережі на основі математичних алгоритмів і теорії діяльності головного мозку. Вони висунули припущення, що нейрони можна спрощено розглядати як пристрої, які оперують двійковими числами, і назвали цю модель «порогової логікою». Подібно до свого біологічного прототипу нейрони Мак-Каллока-Питтса були здатні навчатися шляхом підстроювання параметрів, що описують синаптичну провідність. Дослідники запропонували конструкцію мережі з електронних нейронів і показали, що подібна мережа може виконувати практично будь-які уявні числові чи логічні операції. Мак-Каллок і Піттс припустили,

Дана модель заклала основи двох різних підходів досліджень нейронних мереж. Один підхід був орієнтований власне на вивчення біологічних процесів в головному мозку, інший - на застосування нейронних мереж як методу штучного інтелекту для вирішення різних прикладних задач.

У 1949 році канадський фізіолог і психолог Хебб висловив ідеї про характер з'єднання нейронів мозку і їх взаємодії. Він першим припустив, що навчання полягає в першу чергу в змінах сили синаптичних зв'язків. Теорія Хебба вважається типовим випадком самонавчання, при якому випробувана система спонтанно навчається виконувати поставлене завдання без втручання з боку експериментатора. У більш пізніх варіантах теорія Хебба лягла в основу опису явища довгострокової потенциации.

У 1954 році в Массачусетському технологічному інституті з використанням комп'ютерів Фарлі і Кларк розробили імітацію мережі Хебба. Також дослідження нейронних мереж за допомогою комп'ютерного моделювання були проведені Рочестером, Холландом, Хебітом і Дудою в 1956 році.

У 1957 році Розенблатта були розроблені математична і комп'ютерна моделі сприйняття інформації мозком на основі двошарової навчається нейронної мережі. При навчанні дана мережа використовувала арифметичні дії додавання і віднімання. Розенблатт описав також схему не тільки основного перцептрону, але і схему логічного складання. У 1958 році їм була запропонована модель електронного пристрою, яке повинно було імітувати процеси людського мислення, а два роки по тому була продемонстрована перша діюча машина, яка могла навчитися розпізнавати деякі з букв, написаних на картках, які підносили до його «очам», що нагадує кінокамери.

Інтерес до дослідження нейронних мереж згас після публікації роботи по машинному навчання Мінського і Пейперта в 1969 році. Ними були виявлені основні обчислювальні проблеми, що виникають при комп'ютерній реалізації штучних нейронних мереж. Перша проблема полягала в тому, що одношарові нейронні мережі не могли здійснювати «складання по модулю 2», тобто реалізувати функцію «Що виключає АБО». Другою важливою проблемою було те, що комп'ютери не мали достатньої обчислювальною потужністю, щоб ефективно обробляти величезний обсяг обчислень, необхідний для великих нейронних мереж.

Дослідження нейронних мереж сповільнилися до того часу, коли комп'ютери досягли великих обчислювальних потужностей. Одним з важливих кроків, які стимулювали подальші дослідження, стала розробка в 1975 році Вербосом методу зворотного поширення помилки, який дозволив ефективно вирішувати завдання навчання багатошарових мереж і вирішити проблему з «складанням по модулю 2».

У 1975 році Фукусімою був розроблений когнітрон, який став однією з перших багатошарових нейронних мереж. Фактична структура мережі і методи, використовувані в когнітроні для настройки відносних ваг зв'язків, варіювалися від однієї стратегії до іншої. Кожна зі стратегій мала свої переваги і недоліки. Мережі могли поширювати інформацію тільки в одному напрямку або перекидати інформацію з одного кінця в інший, поки не активувалися все вузли

і мережа не приходила в кінцевий стан. Досягти двосторонньої передачі інформації між нейронами вдалося лише в мережі Хопфілда (1982), і спеціалізація цих вузлів для конкретних цілей була введена в перших гібридних мережах.

Алгоритм паралельної розподіленої обробки даних в середині 1980 років став популярний під назвою коннектівізма. У 1986 році в роботі Руммельхарта і Мак-Клелланда коннектівізма був використаний для комп'ютерного моделювання нейронних процесів.

Незважаючи на великий інтерес, викликаний в науковому співтоваристві розробкою методу зворотного поширення помилки, це також породило численні суперечки про те, чи може таке навчання бути насправді реалізовано в головному мозку. Почасти це пов'язували з тим, що механізм зворотного проходження сигналу ні очевидним в той час, так як не було явного джерела навчає і цільового сигналів. Проте, в 2006 році було запропоновано кілька неконтрольованих процедур навчання нейронних мереж з одним або декількома шарами з використанням так званих алгоритмів глибокого навчання. Ці алгоритми можуть бути використані для вивчення проміжних уявлень, як з вихідним сигналом, так і без нього, щоб зрозуміти основні особливості розподілу сенсорних сигналів, що надходять на кожен шар нейронної мережі.

Як і в багатьох інших випадках, завдання високої складності вимагають застосування не одного, а декількох методів вирішення або їх синтезу. Не виняток і штучні нейронні мережі. З самого початку нинішнього століття в роботах різних дослідників активно описуються нейро-нечіткі мережі, осередковою-нейромережеві моделі.

1. 4. Що собою являють нейронні мережі і які завдання вони можуть вирішувати

Нейронні мережі - один з напрямків в розробці систем штучного інтелекту. Ідея полягає в тому, щоб максимально близько змоделювати роботу людської

нервової системи - а саме, її здатності до навчання і виправлення помилок. У цьому полягає головна особливість будь-нейронної мережі - вона здатна самостійно навчатися і діяти на підставі попереднього досвіду, з кожним разом роблячи все менше помилок.

Нейросеть імітує не тільки діяльність, а й структуру нервової системи людини. Така мережа складається з великого числа окремих обчислювальних елементів («нейронів»). У більшості випадків кожен «нейрон» відноситься до певного прошарку мережі. Вхідні дані послідовно проходять обробку на всіх шарах мережі. Параметри кожного «нейрона» можуть змінюватися в залежності від результатів, отриманих на попередніх наборах вхідних даних, змінюючи таким чином і порядок роботи всієї сіфвстеми.

Нейронні мережі здатні вирішувати такі ж завдання, як і інші алгоритми машинного навчання, різниця полягає лише в підході до навчання.

Всі завдання, які можуть вирішувати нейронні мережі, так чи інакше пов'язані з навчанням. Серед основних областей застосування нейронних мереж - прогнозування, прийняття рішень, розпізнавання образів, оптимізація, аналіз даних. Також нейронні мережі використовуються, наприклад, для настройки параметрів нечітких систем управління. Загалом, немає ніяких сумнівів і в подальшій інтеграції методів штучного інтелекту між собою і з іншими методами вирішення завдань.

1. 5. Етапи розробки нейронних мереж

1. Постановка завдання і вибір архітектури нейронної мережі

На даному етапі необхідно отримати чітке розуміння про розв'язуваної задачі, причини використання для цього методу нейромережевого моделювання та переваги, які він повинен дати. Під конкретну постановку задачі підбирається архітектура нейронної мережі. Детальніше про можливі завданнях і відповідних їм архитектурах штучних нейронних мереж можна дізнатися з попередньої глави.

2. Визначення кількісного і якісного складів входів і виходів

Якісний склад вхідних і вихідних змінних нейросетевой моделі визначається, головним чином, поставленої на попередньому етапі завданням. Кількісний склад може залежати ще від природи вхідних і вихідних змінних і описуваних ними процесів і властивостей. Так, при необхідності масштабування деяких змінних число фактичних входів і виходів нейронної мережі може збільшитися в порівнянні з кількістю входів і виходів моделі.

3. Формування вихідної вибірки даних

Вихідна вибірка даних може бути сформована на основі практичного або обчислювального експерименту. Чим більше обсяг проведеного експерименту, тим точнішу нейромережевому модель можна отримати і використовувати на практиці. Слід також враховувати і вимоги до вибірки вихідних даних, сформульовані в попередньому розділі.

4. Попередня обробка і нормалізація вихідної вибірки

Даний етап визначається обраною архітектурою нейронної мережі, складом вхідних і вихідних змінних і, власне, фактичними даними, отриманими в результаті експерименту.

5. Поділ вихідної вибірки на навчальну і тестову складові

Наявна вихідна вибірка ділиться на навчальну і тестову підвибірки з урахуванням використовуваної архітектури нейронної мережі. Зазвичай обсяг навчальної підвибірки в кілька разів більше, ніж тестової. Перша використовується строго для настройки вагових коефіцієнтів. Друга - для перевірки коректності налаштованої нейросетевой моделі. У разі обмеженого обсягу вихідної вибірки вона вся може бути задіяна для навчання, а оцінка коректності роботи мережі перевіряється на етапі практичної експлуатації.

6. Визначення структури нейронної мережі

Крім раніше встановленого складу вхідних і вихідних змінних на даному етапі задається кількість прихованих шарів і нейронів у кожному шарі. Для багатьох архітектур нейронних мереж вибір структури залежить, в тому числі, від обсягу навчальних даних.

7. Налаштування параметрів нейронної мережі та алгоритму її навчання

На даному етапі задаються вид і параметри активаційних функцій прихованих і вихідних нейронів. Для ітераційного алгоритму вибираються умови закінчення навчання, коефіцієнт швидкості, порядок пред'явлення прикладів навчальної вибірки.

Для точних алгоритмів зазвичай здійснюється одноразовий розрахунок вагових коефіцієнтів. Для ітераційних - багаторазове повторення епох навчання. Для навчання можуть бути використані тільки приклади навчальної підвибірки.

8. Контрастування нейронної мережі

Якщо алгоритм навчання нейронної мережі на увазі итерационную підстроювання вагових коефіцієнтів на протязі всіх епох навчання, нерідкі випадки, коли деякі з ваг за абсолютним значенням настільки близькі до нуля, що не роблять ніякого практичного впливу на результати розрахунків. Проте, на такі розрахунки витрачається досить багато обчислювальних ресурсів, що помітно уповільнює процес навчання. Для вирішення цієї проблеми такі вагові коефіцієнти повністю обнуляються, тобто відповідна зв'язок між входами і нейронами або між двома нейронами виключається зі структури мережі.

10. Тестування нейронної мережі

Тестування нейронної мережі проводиться з використанням прикладів тестової вибірки. За результатами тестування оцінюється помилка роботи нейромережевої моделі.

Етапи 7-9 можуть виконуватися в циклі неодноразово доти, поки не буде отримана налаштована нейронна мережа, за допомогою якої можна вирішувати завдання з необхідним рівнем помилки.

11. Практичне використання

Навчена і протестована нейронна мережа використовується для практичного вирішення завдання по мірі надходження нових вихідних даних (значень вхідних змінних). Результати по навченої мережі можуть бути отримані практично миттєво і інтерпретовані користувачем для подальшого прийняття рішень.

12. донавчання нейронної мережі

Отримувані в ході практичної експлуатації нейронної мережі пари вхідних і вихідних векторів можуть використовуватися для подальшої підстроювання вагових коефіцієнтів. Це особливо важливо, якщо спочатку обсяг вибірки даних був невеликим. Частина архітектур нейронних мереж - використовують принцип самоорганізації - в ході донавчання можуть не тільки змінювати значення вагових коефіцієнтів, а й змінювати свою структуру.

1. 6. Переваги нейронних мереж

Цілком очевидно, що свою силу нейронні мережі черпають, по-перше, з розпаралелювання обробки інформації і, подруге, з здатності самонавчатися, тобто створювати узагальнення. Під терміном узагальнення розуміється здатність отримувати обґрунтований результат на підставі даних, які не зустрічалися в процесі навчання. Ці властивості дозволяють нейронних мереж вирішувати складні (масштабні) завдання, які на сьогоднішній день вважаються важковирішуваними. Однак на практиці при автономній роботі нейронні мережі не можуть забезпечити готові рішення. Їх необхідно інтегрувати в складні системи. Зокрема, комплексне завдання можна розбити на послідовність щодо простих, частина з яких може вирішуватися нейронними мережами.

Розглянемо основні переваги та гідності нейронних мереж перед традиційними обчислювальними системами.

1. Рішення задач при невідомих закономірності

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

2. 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 малопридатними для вирішення завдання і відкине їх.

3. 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 перевчити для роботи в умовах незначних коливань параметрів середовища. Більш того, для роботи в нестаціонарному середовищі (де статистика змінюється з плином часу) можуть бути створені нейронні мережі, переучувати в реальному часі. Чим вище адаптивні здібності системи, тим більш стійкою буде її робота в нестаціонарному середовищі. При цьому слід зауважити, що адаптивність не завжди веде до стійкості; іноді вона призводить до абсолютно протилежного результату. Наприклад, адаптивна система з параметрами, швидко змінюються в часі, може також швидко реагувати і на сторонні збудження, що викличе втрату продуктивності.

4. Потенційний надвисокий швидкодію

Нейронні мережі мають потенційним надвисоким швидкодією за рахунок використання масового паралелізму обробки інформації.

5. Відмовостійкість при апаратній реалізації нейронної мережі

Нейронні мережі потенційно відмовостійкості. Це означає, що при несприятливих умовах їх продуктивність падає незначно. Наприклад, якщо пошкоджений якийсь нейрон або його зв'язку, витяг запомненной інформації ускладнюється. Однак, беручи до уваги розподілений характер зберігання інформації в нейронної мережі, можна стверджувати, що тільки серйозні пошкодження структури нейронної мережі істотно вплинуть на її працездатність. Тому зниження якості роботи нейронної мережі відбувається повільно.

1. 7. Проблеми розвитку нейронних мереж

Проблеми розвитку ІНС Необхідно зауважити, що в сучасні дослідження нейронних мереж впроваджуються високоефективні математичні методи,

запозичені з статичної фізики, синергетики, математичної кібернетики, теорії ймовірностей, диференціальної геометрії. Але, незважаючи на активність досліджень в області ІНС існує безліч невирішених проблем. Часто вивчаються алгоритми виділяються із загального осмислення роботи нервової системи. Досліджуються ті алгоритми, для яких вдається побудувати хороші моделі, а не найбільш важливі для розуміння властивостей мислення, роботи мозку і для створення систем штучного інтелекту.

Викликає сумнів також надмірна спрощеність розуміння роботи нейронних мереж, при якому нейрони розглядаються як такі, що підсумовують порогові елементи, а навчання мережі відбувається шляхом модифікації синапсів. Все це вказує на необхідність максимально повного розуміння роботи біологічних систем обробки інформації і властивостей організмів, що забезпечуються цими системами. Одним з важливих напрямків досліджень, що сприяють такому розумінню, може бути аналіз того, як в процесі біологічної еволюції виникали "інтелектуальні" властивості біологічних організмів. Штучний інтелект (або теорія нейронних мереж) і теорія управління раніше розглядалися як одна область знань.

Одна з цілей штучного інтелекту полягає в тому, щоб замінити людину машиною при виконанні точних операцій; таким чином, зв'язок між штучним інтелектом і теорією управління очевидна.

Спочатку багатошарові нейронні мережі та алгоритм зворотного поширення були розроблені для задач розпізнавання образів, де навчальні зразки є статичними, процедура навчання і функції помилки - однозначні, і навчання в реальному масштабі часу не потрібно. В управлінні навчальні зразки для нейронної мережі змінюються з часом, можливо кілька алгоритмів навчання і функцій помилки, а навчання в реальному часі необхідно.

Повільна збіжність - основний недолік багатошарових нейронних мереж, серйозно обмежує практичне застосування нейронного управління. Для прискорення збіжності в нейронних управлінні необхідно крім розробки ефективних алгоритмів зворотного поширення, вбудовування в мережу знань

про структуру об'єкта управління і попереднього навчання застосовувати гібридні мережі, в яких ІНС зв'язуються зі структурами управління, отриманими на основі інших технологій. З цієї точки зору застосування нечіткої логіки в теорії управління спільно з нейронними мережами має ряд переваг.

Одне з основних переваг полягає в тому, що нечітко логічний контролер може розроблятися по лінгвістичним правилам, що тісно пов'язано з штучним інтелектом. Нечіткий контролер складається з набору умовних лінгвістичних операторів, або правил (званих нечіткими асоціативними матричними правилами, або НАМ - правилами), які задають конкретні ситуації управління. Управління на основі нечіткої логіки може успішно застосовуватися для багатовимірних, нелінійних процесів, що змінюються в часі.

1. 8. Класифікації нейронних мереж

Нейронні мережі класифікуються за такими видами навчання:

1. Нейронні мережі проходять навчання з учителем;
2. Нейронні мережі проходять навчання без учителя.

Розглянемо ці види трохи докладніше.

Нейронні мережі проходять навчання з учителем.

При навчанні з учителем мається на увазі, що кожен вектор, що входить в існуючий цільової вектор, який представляє з себе необхідний вихід. Спільно вони є навчальною парою. Мережа навчається на кількох навчальних парах.

Надається вихідний вектор, визначається вихід мережі і порівнюється з представленими векторами.

Далі змінюють ваги відповідно до математичним алгоритмом, який прагне зменшити помилку. Вектори безлічі навчальних даних пред'являються послідовно. У міру проходу обчислюються помилки і ваги і підлаштовуються для всіх векторів, поки помилка по навчальних даних не досягне потрібного рівня.

Нейронні мережі, які навчаються без допомоги вчителя.

Навчання без вчителя виглядає набагато більш часто зустрічається моделлю навчання особливо часто зустрічається в біологічних нейронних мережах.

Розвинена Кохоненом і іншими вченими, вона не вимагає цільової вектор для виходів. З цього випливає що, не потрібні і порівняння із заздалегідь підготовленими ідеальними варіантами відповідей. Навчальні дані складаються тільки з вхідних векторів.

Навчальний алгоритм змінює ваги своєї мережі так, щоб утворювалися узгоджені вихідні вектори, тобто щоб надання досить схожих вхідних векторів видавало схожі виходи.

Процес навчання, послідовно, визначає статистичні властивості наданих навчальних даних і групує схожі вектори в класи.

зміна ваг

Нейронні мережі так само діляться на наступні групи. З фіксованими зв'язками - ваги яких вибираються заздалегідь виходячи з завдання і з динамічними зв'язками - які перебудовують свої ваги в процесі навчання.

Тип вхідних даних

Вхідні дані так само діляться на кілька; аналогові вхідні дані представлені в вигляді дійсних чисел і виконавчі інформація яких представляється у вигляді нулів і одиниць.

Моделі нейронної мережі, які найчастіше використовуються на даний момент

Мережі прямого поширення- всі зв'язки цієї мережі мають суворе напрямок від вхідних нейронів до їх виходів. Серед таких мереж хочеться відзначити: найпростіший персептрон автором якого є Френк Розенблат і багатошаровий персептрон.

Нейронні мережі реккурентное типу- дані з вихідних нейронів або з прихованого шару передається частково назад на вхідні нейрони.

Радіально базисні функції- це нейронна мережа, в основі якої є наявність прихованого шару з радіальних елементів і вихідного шару з лінійних елементів. Такі мережі досить компактні і навчаються досить швидко.

Вони були запропоновані в роботах Broomhead and Lowe (1988) і Moody and Darkin (1989).

Радіально базисна мережа користується наступними унікальними властивостями: один прихований шар, нейрони тільки прихованого шару мають нелінійну функцію активації і синаптичні ваги прихованого і вхідного шарів є одиницею.

мережі Кохоненаабо Самоорганізуються карти - це клас мереж зазвичай навчається без допомоги вчителя і часто застосовується в задачах пов'язаних з розпізнаванням зображень.

Такі мережі здатні визначати нові елементи у вхідних даних: якщо пройшовши навчання мережу побачить набір даних, несхожий ні на один зі знайомих зразків, то вона класифікує такий набір і не виявить його новизну.

мережа Кохоненамає всього два шари: вихідний і вхідний, складений з радіальних елементів.

Висновки

У зв'язку з вищевикладеної інформацією про шахтні підйомні установки, шахтні підйомні машини, а також нейронні мережі, були сформульовані такі напрямки завдань подальшого дослідження:

- Розгляд існуючих прикладів використання нейронних мереж на шахтних підйомних машинах;
- Вибір конкретного типу нейронної мережі яка дозволить удосконалити роспізнавання нештатних ситуацій на шахтних підйомних машинах;
- Технічний огляд основних моментів при проектуванні обраного типу нейронної мережі.

РОЗДІЛ 2.
ВИКОРИСТАННЯ НЕЙРОНИХ МЕРЕЖ ДЛЯ РОСПІЗНАВАННЯ НЕШТАТНИХ СИТУАЦІЙ НА ШАХТНИХ ПІДЙОМНИХ МАШИНАХ

2. 1. Використання нейронних мереж на шахтах

За останні роки парк підйомних машин сильно застарів. Так, в м Павлоград (Україна) шахта 2Ц 4х1.8 і в м. Сташкові шахта 2Ц 6х2.4 підйомні установки мають термін служби більше 25 років. Такий же термін служби мають привід підйомних машин, система управління цим приводом, обладнання шахтного стовбура, стовбурова сигналізація та інші життєво важливі елементи підйомних установок. У зв'язку з цим досить актуальною є задача забезпечення безпечної експлуатації підйомних установок.

Впровадження інформаційних технологій в промисловість дозволило розробити систему комп'ютерного моніторингу роботи підйомних установок. Прикладом таких систем є реєстратори параметрів підйомної установки РПУ-03.1, РПУ-03.3, РПУ-03.5, що відрізняються набором виконуваних функцій і виконанням робочої станції. Загальним для типового ряду реєстраторів параметрів найменуванням є «РПУ-03.х». РПУ-03.х дозволяють:

- реєструвати всі цикли роботи підйомної установки;
- контролювати всі режими роботи підйомної установки;
- визначати причини виникнення аварійної ситуації;
- контролювати виконання робіт з технічного обслуговування і ремонту;
- визначати і реєструвати положення, швидкість і напрямок руху підйомних посудин;
- об'єктивно оцінити технічний стан елементів підйомної установки;

В даних і подібних системах нейронні мережі використовуються для визначення причини виникнення аварійної ситуації, після того як вона трапилася. Визначення технічного стану деяких елементів шахтної підйомної машини. Так само є додаткові системи, які пророкують можливість виникнення нештатної ситуації на шахтній підйомної машини шляхом аналізу даних минулих станів шахтних підйомних машин під час роботи.

2. 2. Сигнали шахтної підйомної машини

На Рис. 2.1, 2.2, 2.3 зображені приклади нормального стан сигналів шахтної підйомної машини (ШПМ) у вигляді діаграм циклів руху «Вперед», «Назад» а так само діаграма швидкості ШПМ. Таблиці 2.1, 2.2, 2.3 містять дані для побудови діаграм відповідно.

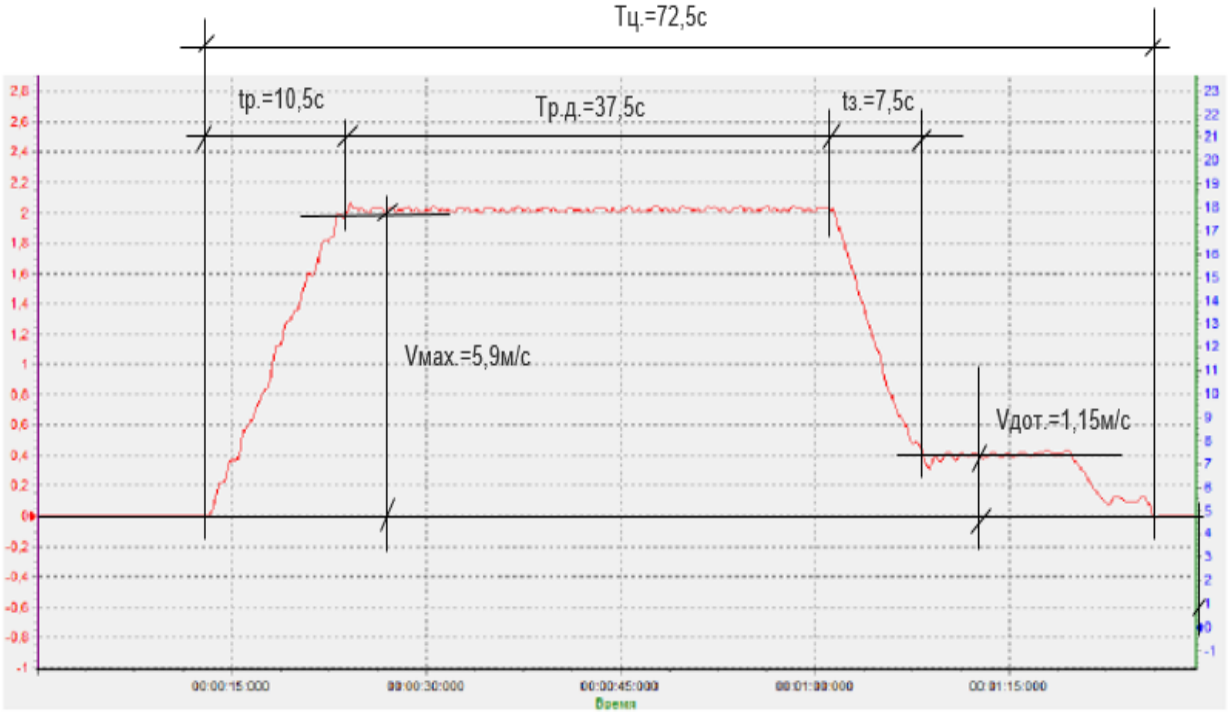

Рис. 2.1 Приклад діаграми руху «Вперед» при нормальному стані ШПМ дані см. Таб. 2.1)

Таблиця 2.1

Дані для побудови діаграми руху «Вперед» при нормальному стані ШПМ.

максимальна швидкість	5,9м / с
швидкість дотягування	1,15м / с
максимальне прискорення	0,56 (А розм. = 5,9: 10,5 = 0,56м / с?)
максимальне уповільнення	0,63 (А спізнюся. = (5,9-1,15): 7,5 = 0,63м / с?)
час циклу	72,5с
час розгону	10,5с
час уповільнення	7,5с
Середній час завантаження і розвантаження	25с

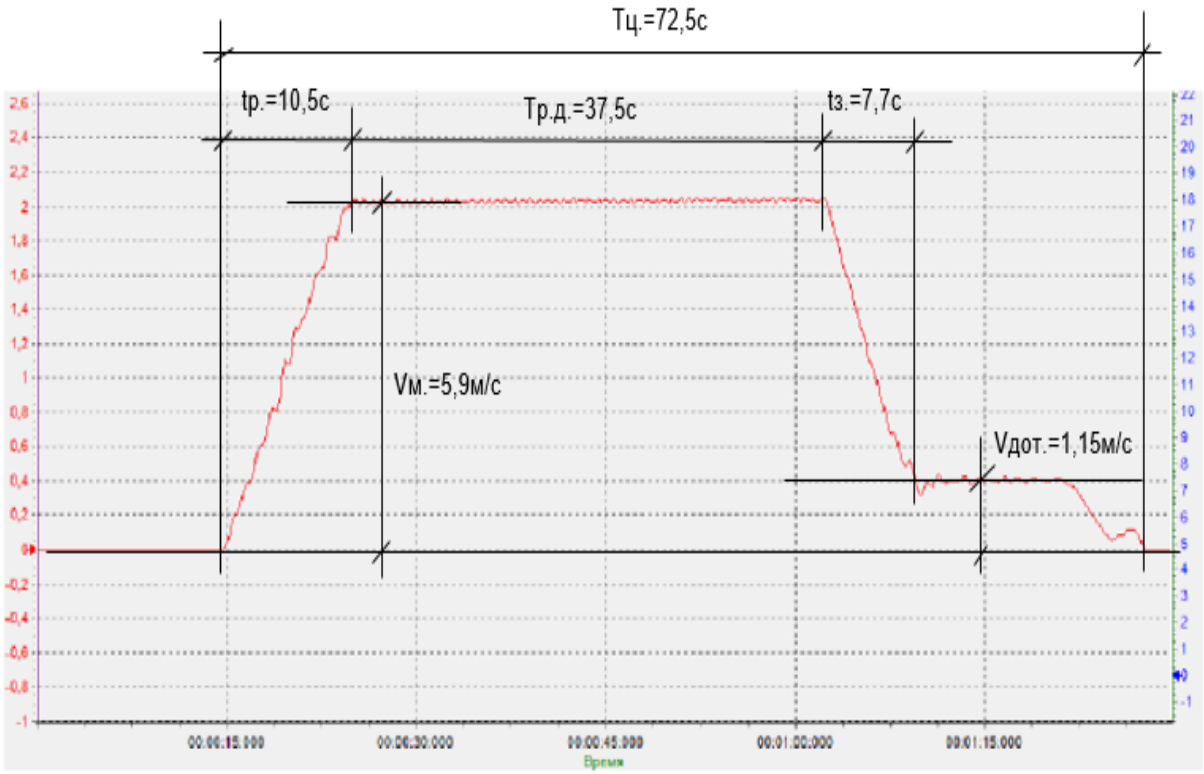

Рис. 2.2 Приклад діаграми руху «Назад» при нормальному стані ШПУ (дані див. Таб. 2.2)

Таблиця 2.2

Дані для побудови діаграми руху «Назад» при нормальному стані ШПУ.

максимальна швидкість	5,9м / с
швидкість дотягування	1,15м / с
максимальне прискорення	0,56 (А розм. = 5,9: 10,5 = 0,56м / с?)
максимальне уповільнення	0,62 (А спізнюся. = (5,9-1,15): 7,7 = 0,62м / с?)
час циклу	72,5с
час розгону	10,5с
час уповільнення	7,7с
Середній час завантаження і розвантаження	25с

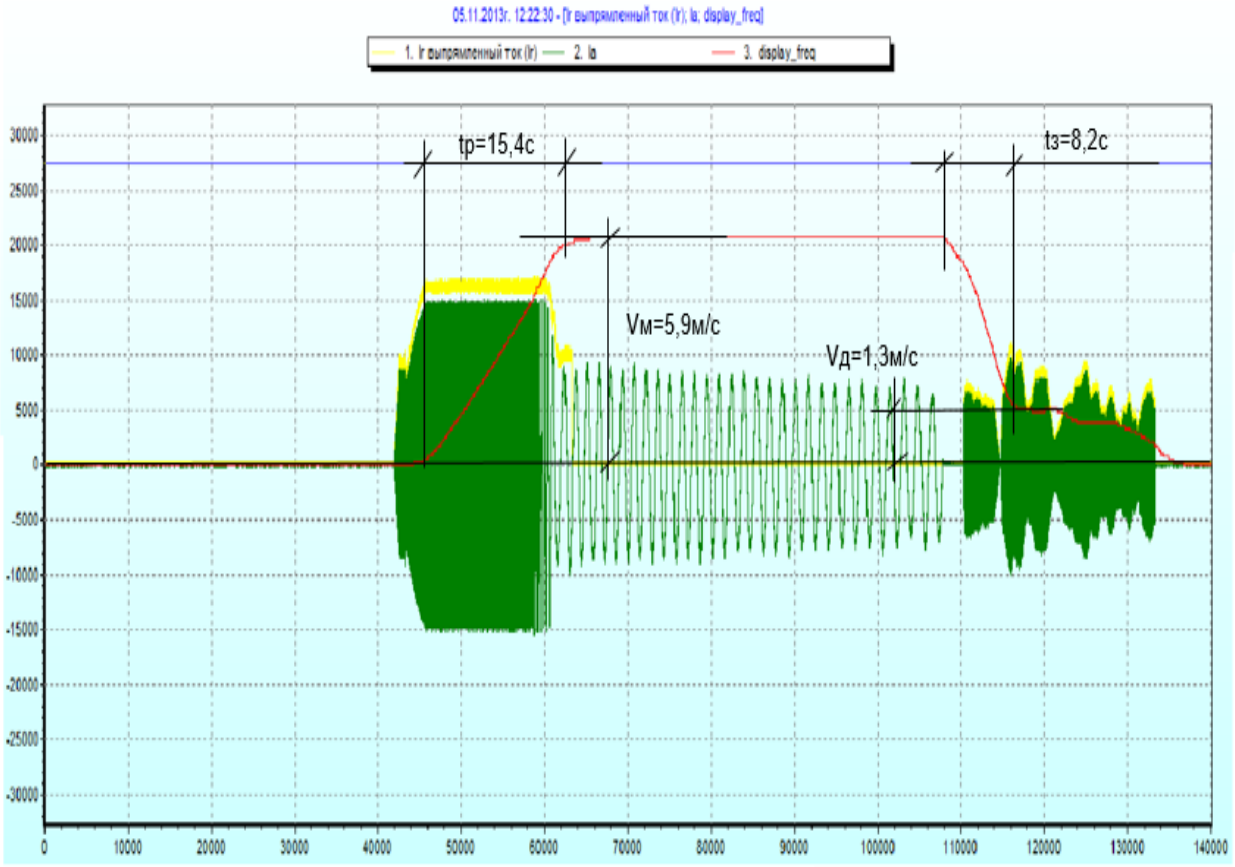

Рис. 2.3 Приклад діаграми швидкості при нормальному стані ШПУ (дані див. Таб. 2.3)

Таблиця 2.3

Дані для побудови діаграми швидкості при нормальному стані ШПУ.

Загальна тривалість циклу Тц	76с
Максимальна швидкість Vмакс.	5,9 м / с
Швидкість підходу Vп	0,5 м / с
Швидкість дотяжки Vд	1,3 м / с
прискорення розгону	0,38м / с?
уповільнення	0.56м / с?

Для визначення нештатної ситуації ШПМ за допомогою нейронної мережі, необхідно навчити мережу на визначення відхилень від нормального стану будь-

якого з вхідних даних, виявити закономірності відхилень призводять до нештатних ситуацій. У результаті мережа буде визначати поточний стан ШПУ і передбачати зміни сигналів ШПМ призводять до нештатних станів (Рис. 2.4).

Рис. 2.4 Приклад діаграми руху «Вперед» нештатного стану ШПУ. (Де жирна смуга, нормальна швидкість ШПУ)

2. 3. Нейронна мережа Кохонена

Кластеризація або природна класифікація - це процес об'єднання в групи об'єктів, що володіють схожими ознаками. На відміну від звичайної класифікації, де кількість груп об'єктів фіксоване і заздалегідь визначено набором ідеалів, тут ні групи і ні їх кількість заздалегідь не визначені і формуються в процесі роботи системи виходячи з певної міри близькості об'єктів.

Кластеризація застосовується для вирішення багатьох прикладних задач: від сегментації зображень до економічного прогнозування та боротьби з електронним шахрайством.

Штучна нейронна мережа Кохонена або самоорганізована карта ознак (SOM) була запропонована фінським дослідником Тойво Кохоненом на початку 1980-х років.

Нейронні мережі Кохонена типовий приклад нейромережевої архітектури, яка навчається без учителя. Звідси і перелік вирішуваних ними завдань: кластеризація даних або прогнозування властивостей. Крім того, мережі Кохонена можуть використовуватися з метою зменшення розмірності даних з мінімальною втратою інформації.

Більшість нейронних мереж навчаються з учителем на вибірках даних, що включають безліч прикладів, що складаються з відповідних один одному пар вхідних і вихідних векторів. При цьому вихідні значення брали безпосередню участь в налаштуванні вагових коефіцієнтів. У нейронних мережах Кохонена вихідні вектора в навчальній вибірці можуть бути, але можуть бути і відсутніми, і, в будь-якому випадку, вони не беруть участі в процесі навчання. Тобто виходи не використовуються в якості орієнтирів при корекції синапсів. Саме тому даний принцип настройки нейронної мережі називається самонавчанням.

У розглянутій архітектурі сигнал поширюється від входів до виходів в прямому напрямку. Структура нейронної мережі містить єдиний шар нейронів (шар Кохонена) без коефіцієнтів зміщення (Рис. 2.5). Загальна кількість вагових коефіцієнтів розраховується як добуток:

$$N_w = MK$$

Кількість нейронів дорівнює кількості кластерів, серед яких відбувається початкове розподіл і подальше перерозподіл навчальних прикладів. Кількість вхідних змінних нейронної мережі дорівнює числу ознак, що характеризують об'єкт дослідження і на основі яких відбувається віднесення його до одного з кластерів.

Слід розрізняти власне самонавчання і самоорганізацію нейронної мережі Кохонена. При звичайному самообучении мережу має строго фіксовану структуру, тобто кількість нейронів, що не змінюється протягом усього життєвого циклу. При самоорганізації мережу, навпаки, не має постійної

структури. Залежно від знайденого відстані до нейрона-переможця або цей нейрон використовується для кластеризації прикладу, або для поданого на входи прикладу створюється новий кластер з відповідними йому ваговими коефіцієнтами. Крім того, в процесі самоорганізації структури мережі Кохонена окремі нейрони можуть виключатися з неї.

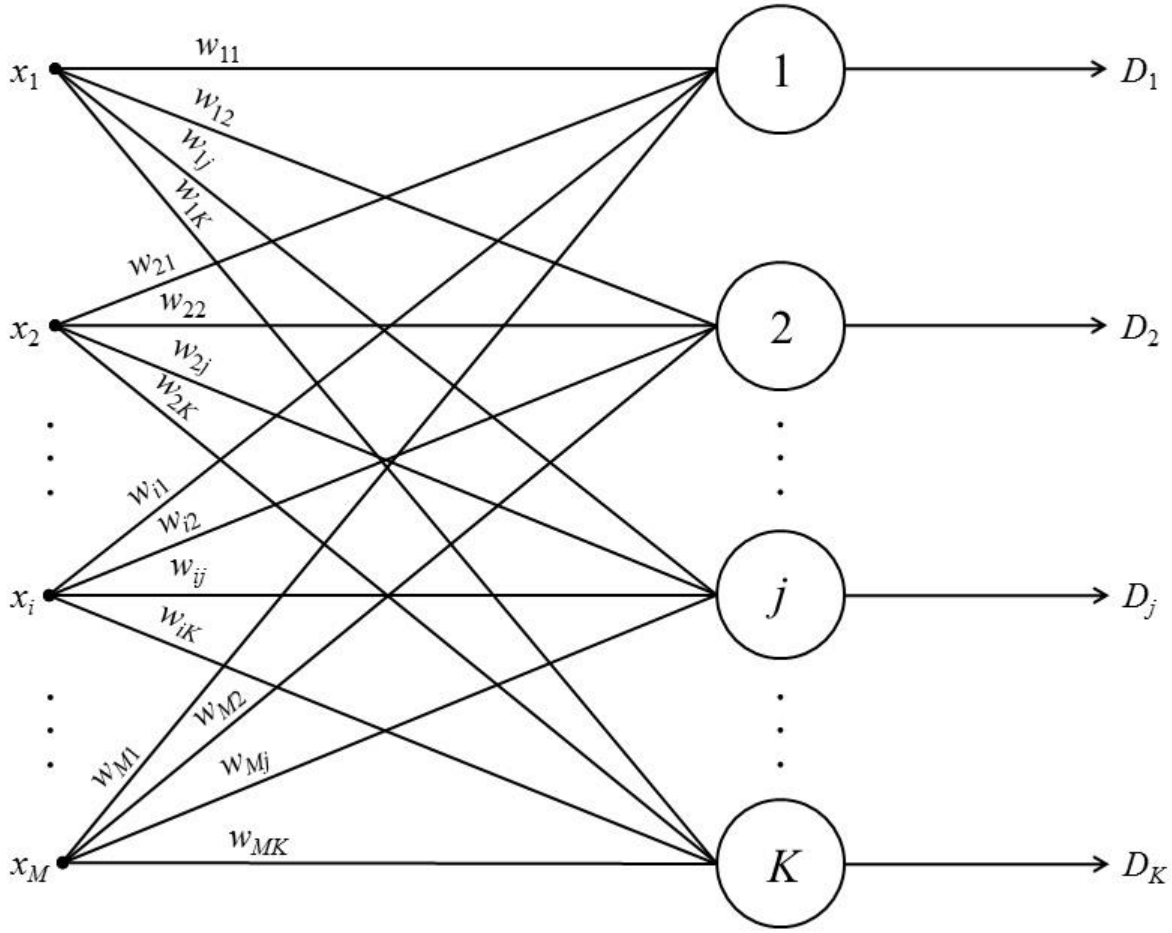

Рис. 2.5 Загальна структура нейронної мережі Кохонена

Нормалізація вхідних змінних виконується в межах [-1, 1] або [0, 1].

Для життєвого циклу нейронних мереж даної архітектури характерні три основні стадії життєвого циклу: навчання, кластерний аналіз та практичне використання.

2. 4. Алгоритм навчання нейронної мережі Кохонена

Алгоритм навчання мережі Кохонена включає етапи, склад яких залежить від типу структури: постійної (самонавчальна мережу) або змінною (самоорганізована мережу). Для самонавчання послідовно виконуються:

1. Завдання структури мережі (кількості нейронів шару Кохонена) (K).

2. Випадкова ініціалізація вагових коефіцієнтів значеннями, що задовольняють одному з наступних обмежень:

- при нормалізації вихідної вибірки в межах [-1, 1]:

$$\left|w_{ij}\right| \le \frac{1}{\sqrt{M}} \tag{1}$$

- при нормалізації вихідної вибірки в межах [0, 1]:

$$0{,}5 - \frac{1}{\sqrt{M}} \le w_{ij} \le 0{,}5 + \frac{1}{\sqrt{M}} \tag{2}$$

де M - кількість вхідних змінних мережі - характеристичних ознак об'єкта дослідження.

3. Подача на входи мережі випадкового навчального прикладу поточної епохи навчання і розрахунок евклідових відстаней від вхідного вектора до центрів всіх кластерів:

$$R_j = \sqrt{\sum_{i=1}^{M} \left(\tilde{x}_i - w_{ij}\right)^2} \tag{3}$$

4. За найменшому з значень Rj вибирається нейрон-переможець j, в найбільшою мірою близький за значеннями з вхідним вектором. Для обраного нейрона (і тільки для нього) виконується корекція вагових коефіцієнтів:

$$w_{ij}^{(q+1)} = w_{ij}^{(q)} + v\left(\tilde{x}_i - w_{ij}^{(q)}\right) \tag{4}$$

де v - коефіцієнт швидкості навчання.

5. Цикл повторюється з кроку 3 до виконання одного або декількох умов закінчення:

- вичерпано заданий гранична кількість епох навчання;

- не відбулося значного зміни вагових коефіцієнтів в межах заданої точності протягом останньої доби навчання;

- вичерпано заданий граничний фізичний час навчання.

Коефіцієнт швидкості навчання може здаватися постійним за межі (0, 1] або змінним значенням, поступово зменшується від епохи до епохи.

У разі самоорганізації мережі Кохонена алгоритм зазнає певних змін:

1. Здається критичну відстань Rкр, відповідне максимально допустимому евклидову відстані між входами прикладу і вагами нейрона-переможця. Початкова структура не містить нейронів. При подачі на входи мережі самого першого прикладу навчальної вибірки створюється перший нейрон з ваговими коефіцієнтами, рівними поданням вхідним значенням.

2. На входи мережі подається новий випадково обраний приклад поточної епохи навчання, розраховуються евклідові відстані від прикладу до центру кожного кластера по співвідношенню (3) і визначається нейрон-переможець з найменшим з них Rmin.

3. Якщо виконується умова Rmin <= Rкр, проводиться корекція вагових коефіцієнтів відповідного нейрона-переможця по співвідношенню (4), в іншому випадку в структуру мережі додається новий нейрон, вагові коефіцієнти якого приймаються чисельно рівними вхідним значенням поданого прикладу.

4. Процедура повторюється з п. 2. Якщо протягом останньої доби навчання будь-які кластери залишилися не задіяними, відповідні нейрони виключаються зі структури мережі Кохонена.

5. Обчислення закінчуються, якщо виконується одна з умов, прописаних в алгоритмі самонавчання мережі фіксованого структури.

Ще одна модифікація алгоритмів самонавчання і самоорганізації передбачає корекцію вагових коефіцієнтів не тільки нейрона-переможця, але і всіх інших нейронів. Для цього слід використовувати коефіцієнт швидкості навчання, регресний зі збільшенням відстані до центру кластера Rj:

$$v_j = v_0\left(1 - \frac{1}{1+e^{-\beta(R_j - R_{кр})}}\right)$$

де Rкр - критичне значення відстані: чим воно менше, тим більш значні будуть коректування ваг найближчих до навчального наприклад кластерів і практично незначущі - ваг більш-менш віддалених від нього; β - параметр, який

встановлює ступінь нелінійності впливу відстані на коефіцієнт швидкості; v0 - базове (максимально можливе) значення коефіцієнта швидкості на поточній епосі навчання.

Як значення Rкр можна розраховувати середнє відстань для кожного кластера при поточному пред'явленні навчального прикладу. Параметр β рекомендується вибирати рівним 3,0 ± 0,5.

Як правило, на практиці при використанні самоорганізації нейронної мережі Кохонена доводиться стикатися ще з однією проблемою. З одного боку, якісь кластери можуть містити занадто маленька кількість прикладів, що призводить до складнощів у наступному узагальненні інформації. З іншого боку, деякі кластери можуть виявитися занадто великими, тобто містити дуже багато прикладів. В цьому випадку для регулювання розміру кластера і вирішення проблеми його переповненості можна задати в якості додаткового параметра граничне число прикладів, які формують кластер (Nпр). Якщо в якийсь момент виявляється, що новий приклад повинен бути віднесений до кластеру, розмір якого вже максимальний, приймається рішення про створення іншого кластера,

До навченої нейронної мережі застосовується процедура кластерного аналізу - процедури опису властивостей кластера на основі аналізу кількісного і якісного складів прикладів, які сформували його. Слід враховувати, що опис кластерів може базуватися не тільки на значеннях вхідних змінних навчальної вибірки, а й на значеннях змінних, які не брали участі у формуванні кластерів. Зокрема, в опис можуть входити дані про середні значення таких змінних серед всіх прикладів, які сформували кластер. Крім того, доцільно для кожного кластера мати дані про середньоквадратичному відхиленні або дисперсії по кожній змінній.

При практичному використанні нейронної мережі Кохонена новий приклад подається на її вхід і відноситься до одного з існуючих кластерів, або робиться висновок про неможливість такого віднесення (при великій відстані до центру найближчого кластера). Якщо вибір кластера відбувся, його опис, отримане в

результаті кластерного аналізу, і відповідні кластеру рішення повинні поширюватися в тому числі на поданий приклад.

Практичне використання мережі Кохонена полегшується за рахунок візуалізації результатів кластеризації. В результаті самонавчання (самоорганізації) мережі виходить набір кластерів, кожен з яких характеризується своїм центром (значеннями вагових коефіцієнтів відповідного нейрона) і кількістю навчальних прикладів, які сформували його. Чи не складає ніяких труднощів визначити евклідова відстань між центрами всіх можливих пар кластерів і графічно зобразити їх на так званій карті Кохонена - двовимірної графічної структурі, що дозволяє судити не тільки про розміри і положення кожного окремо взятого кластера, а й про близькість один до одного і взаємне розташування окремих кластерів.

2. 5. Використання нейронної мережі Кохонена для розпізнавання нештатних станів на шахтних підйомних машинах

Як відомо нейронна мережа Кохонена зазвичай використовується для візуалації даних і подальшому їх аналізу. Після того як мережа навчилася, і визначила набір кластерів (автоматично або задане число кластерів) розбивши данні для навчання в групи, отриману інформацію аналізує людина. Однак якщо вірно навчити мережу, і виконати деякі маніпуляції з виходить матрицею нейронів, можна визначати ставлення до того чи іншого кластеру новий вхідний варіанту не перенавчитися мережу.

Для навчальної вибірки даних шахтних підйомних машин, можна виділити штатний і нештатний стан шахтної підйомної машини, тобто те що не відповідає штатному стану. Для того, щоб правильно навчання нейронної мережі, як правило досить граничних даних нормального і нештатного стану шахтної підйомної установки, причому нормальний стан необхідно вказувати з допусками відхилення від ідеального значення.

Після отримання матриці нейронів навченої мережі, необхідно визначити ставлення кожного вхідного параметра який використовувався для навчання і відображав штатний стан шахтної підйомної машини. (Рис. 2.6). Таким чином можна визначити які елементи вихідний матриці нейронів відносяться до нормального стану об'єкта. До навчання нейронної мережі Кохонена використовуючи випадкові початкові значення ваг нейронів, неможливо визначити які нейрони до якого кластеру будуть ставитися.

1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0
0	0	0	0	1	0	0	1	1
0	0	0	1	1	1	0	0	1
0	0	1	1	0	1	0	0	1
0	1	1	0	0	0	0	0	1
1	1	1	0	0	0	0	1	0
1	1	1	0	1	0	0	0	0

Рис. 2.6 Елементи вихідної матриці нейроні які відносяться до штатного стану. (де 0 – штатний стан)

1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	1	0	1
0	0	1	0	1	0	0	0	0
0	1	0	0	1	0	1	1	1
1	0	0	1	1	1	1	0	1
0	1	1	1	0	0	1	0	1
1	1	1	0	0	0	0	0	1
1	1	1	0	1	1	0	1	0
1	1	1	1	1	0	0	1	0

Рис. 2.7 Елементи вихідної матриці нейроні які відносяться до нештатного стану. (де 1 – нештатний стан)

Для визначення штатного стану шахтної підйомної машини, зазначених дій вище, цілком достатньо. Однак, якщо різниця між навчальними даними нормального і позаштатними станами шахтної підйомної машини не велика, що може бути і причиною неправильного визначення стану шахтної підйомної машини, необхідно провести аналогічну операцію і для позаштатних вхідних даних (Рис. 2.7). Ті нейрони, які ставитися до штатного станом і нештатному, можна розуміти, як стан підозріле на нештатне. (Рис. 2.8)

Далі, маючи розуміння які нейрони до якого стану шахтної підйомної машини ставитися, можна приступати до визначення нештатного стану шахтної підйомної машини. Для цього визначаються параметри необхідно нормалізувати так само як і навчальні дані, потім використовуючи Евклідову відстань визначити нейрон найбільш близький до даних значень. Знайшовши такий нейрон, дивимося, за який стан шахтної підйомної машини він відповідає, і отримавши відповідь, видаємо його.

1	2	1	2	2	2	2	2	2
1	0	1	0	0	0	2	0	2
2	0	1	0	1	0	0	0	0
0	2	0	0	1	0	2	1	1
2	0	0	1	1	1	2	0	1
0	2	1	1	0	2	2	0	1
2	1	1	0	0	0	0	0	1
1	1	1	0	2	2	0	1	0
1	1	1	2	1	0	0	2	0

Рис. 2.8 Елементи вихідної матриці нейроні які відносяться до штатного та нештатного стану. (де 2 – підозрілий стан)

В результаті можна визначити 3 типи стану шахтної підйомної машини завдяки алгоритму описаному вище:

1) Штатний стан

2) Стан підозрілий до нештатного

3) Нештатний стан

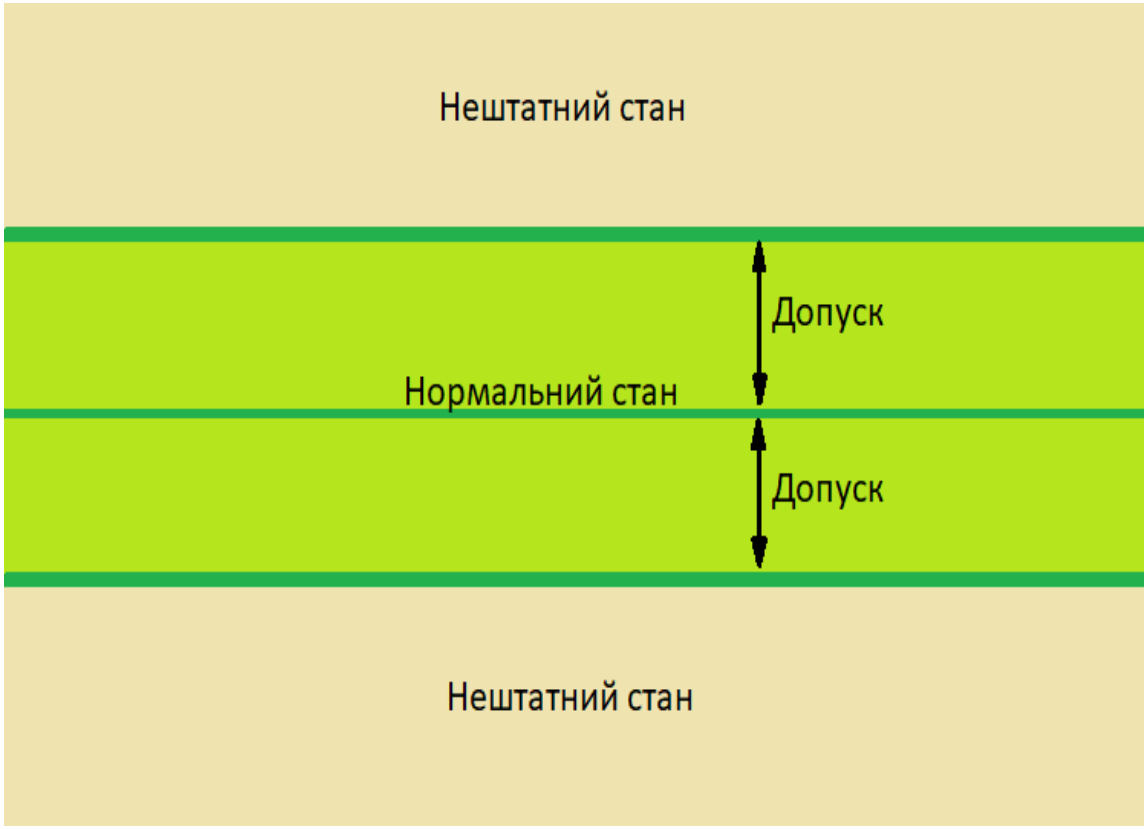

Рис. 2.9 Діапазон вхідних даних

Для навчання нейронної мережі і подальшого використання алгоритму розпізнавання стану шахтної підйомної машини, необхідно навчати нейронну мережу на данні які відповідають за нештатний стан і за нормальний стан в межах допуску (Рис. 2.9).

Під час тестування та експериментів, використовуючи вищезазначений алгоритм розпізнавання нештатних станів на шахтних підйомних машинах, на еталонних даних штатного та нештатних станів, було виявлено, що деяка частина кластерів перетинаються, однак виявлення штатного стану шахтної підйомної машини в більшості ситуацій виконується вірно. З цього можна зробити висновок, що якщо тестові данні ідентифікуються як нормальний стан, то це відповідає дійсному стану шахтної підйомної машини, однак якщо тестові данні попадають до перетину кластерів в яких виявлено перетин штатного та нештатного станів, то це являє собою попередженням технічним службам для необхідності перевірки стану шахтної підйомної машини.

Висновки

Нейронна мережа Кохонена (або самоорганізаційна карта Кохонена) з використанням вищеописаного розробленого алгоритму може використовуватись для розпізнавання нештатних ситуацій на шахтних підйомних машинах. Якщо правильно навчити мережу використовуючи експериментальні, документаційні або теоретичні данні, та використовувати розроблених алгоритм розпізнавання стану шахтної підйомної машини, розроблена нейронна мережа буде ефективно розпізнавати нештатні і підозрілі стани шахтної підйомної машини.

РОЗДІЛ 3.

ВИКОРИСТАННЯ РОЗРОБЛЕННОГО АЛГОРИТМУ РОСПІЗНАВАННЯ НЕШТАТНИХ СИТУАЦІЙ НА ШАХТНИХ ПІДЙОМНИХ МАШИНАХ

3.1 Сбір тестових даних для навчання нейронної мережі Кохонена

Для ефективної роботи мережі Кохонена необхідно зібрати тестові дані на яких мережа буде навчатись. Як правило кількість необхідного об’єму тестових даних визначається експериментальним шляхом.

При тестуванні розробленого алгоритму розпізнавання нештатного стану шахтної підйомної машини, використовувались реальні дані станів деяких систем, та їх показники. Наприклад показники швидкості, прискорення, положення і час шахтної кліті при фазі руху «Вперед» (Рис. 3.1, Таблиця 3.1)

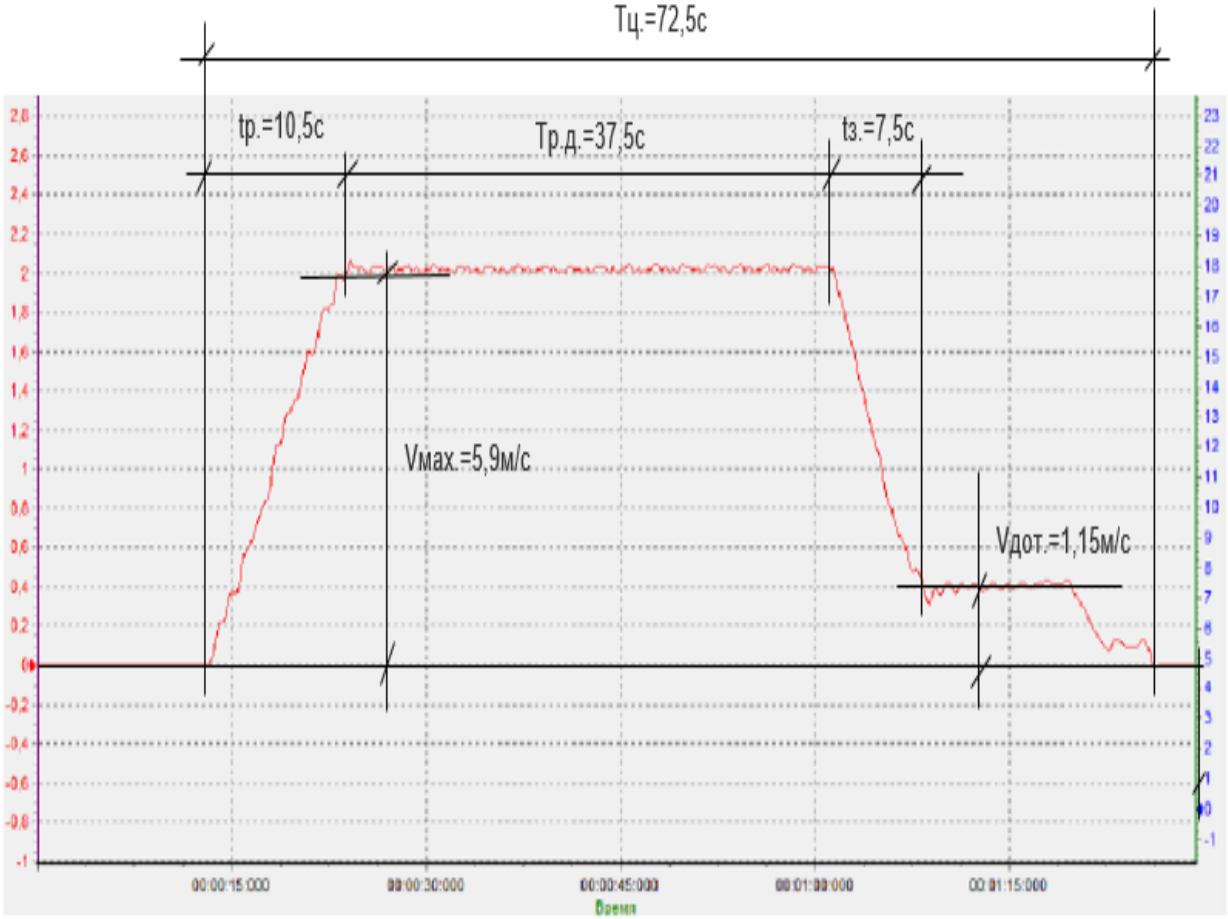

Рис. 3.1 Приклад діаграми руху «Вперед» при нормальному стані ШПМ дані см. Таб. 1)

Таблиця 3.1

Дані для побудови діаграми руху «Вперед» при нормальному стані ШПМ.

максимальна швидкість	5,9м / с
швидкість дотягування	1,15м / с
максимальне прискорення	0,56 (А розм. = 5,9: 10,5 = 0,56м / с2)
максимальне уповільнення	0,63 (А спізнюся. = (5,9-1,15): 7,5 = 0,63м / с2)
час циклу	72,5с
час розгону	10,5с
час уповільнення	7,5с
Середній час завантаження і розвантаження	25с

Ці данні використовувались як еталоні. Були сформовані допуски у 10% від еталонного для даних штатного стану. Загалом по 20 значень штатного стану у межах допуску для кожного етапу роботи. І відповідно до кожнохо стану, було додані стани які являються собою вихід за межі нормальних показників. Важливо щоб дані які повинні представляти собою інший стан (нештатний у нашому випадку) чітко відрізнялися від штатного стану на кожний етап роботи шахтної підйомної машини.

Загалом для тестових даних які відображають нормальний стан с допусками було додано близько 1560 значень. Для нештатного стану близько 390 варіантів показників.

3.2 Аналіз отриманого результату роботи нейронної мережі Кохонена.

Як вже зазначалося, нейронна мережа Кохонена зазвичай використовуються для візуалізації отриманих результатів кластерізації, однак розроблений алгоритм розпізнавання нештатних станів не цього. Для

подальшого використання отриманих результатів, була реалізована мережа Кохонена і до результатів її роботи був використаний розроблений алгоритм, однак спочатку отриманні дані необхідно проаналізувати.

1) Визначення елементів вихідної матриці нейронів які відповідають за штатний стан. Для цього необхідно виділити вхідні данні які відносяться до штатного стану шахтної підйомної машини і знайти нейрони до яких вони відносяться. (Рис. 3.2)
2) Визначення елементів вихідної матриці нейронів які відповідають за нештатний стан. Для цього також виділяймо вхідні данні які відносяться до нештатного стану шахтної підйомної машини і знаходимо нейрони до яких вони відносяться. (Рис. 3.3)
3) Наклавши визначені нейрони які відносяться до штатного і нештатного стану шахтної підйомної машини ми можемо знайти перетини значень, тобто випадок коли визначені кластери перетнулись і деякі данні відносяться до різних кластерів однаково. Такі випадки позначаймо як підозрілі до нештатного стану. Зазначимо, що якщо знайдуться нейрони які не віднеслись ні до штатних, ні до нештатних вхідних даних ми помічаємо як підозрілі. (Рис. 3.4)

1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0
0	0	0	0	1	0	0	1	1
0	0	0	1	1	1	0	0	1
0	0	1	1	0	1	0	0	1
0	1	1	0	0	0	0	0	1
1	1	1	0	0	0	0	1	0
1	1	1	0	1	0	0	0	0

Рис. 3.2 Елементи вихідної матриці нейроні які відносяться до штатного стану. (де 0 – штатний стан)

1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	1	0	1
0	0	1	0	1	0	0	0	0
0	1	0	0	1	0	1	1	1
1	0	0	1	1	1	1	0	1
0	1	1	1	0	0	1	0	1
1	1	1	0	0	0	0	0	1
1	1	1	0	1	1	0	1	0
1	1	1	1	1	0	0	1	0

Рис. 3.3 Елементи вихідної матриці нейроні які відносяться до нештатного стану. (де 1 – нештатний стан)

1	2	1	2	2	2	2	2	2
1	0	1	0	0	0	2	0	2
2	0	1	0	1	0	0	0	0
0	2	0	0	1	0	2	1	1
2	0	0	1	1	1	2	0	1
0	2	1	1	0	2	2	0	1
2	1	1	0	0	0	0	0	1
1	1	1	0	2	2	0	1	0
1	1	1	2	1	0	0	2	0

Рис. 3.4 Елементи вихідної матриці нейроні які відносяться до штатного та нештатного стану. (де 2 – підозрілий стан)

3.3 Визначення теперішнього стану шахтної підйомної машини

Після знаходження перетинів станів вихідної матриці нейронів можна вважати мережу навченою і тепер працювати лише над отриманим результатом, а саме з вихідною матрицею нейронів і матрицею станів до яких кожен елемент матриці нейронів відноситься.

1) Отримуймо теперішні показники стану систем шахтної підйомної машини які відповідають даним які використовувалися для навчання нейронної мережі.

2) Нормалізуємо отримані тестові данні так само як і нормалізувалися данні для навчання мережі.
3) Знаходимо до якого нейрону вихідної матриці відносяться тестові данні, використовуємо Евклідову відстань для знаходження “найближчого” нейрону до нормалізованих даних. Отримуймо його позицію у матриці вихідних нейронів.
4) Визначаймо відповідну позицію знайденого елементу вихідної матриці нейронів у матриці станів. Його позиція відповідає позиції знайденого елемента у вихідній матриці нейронів.
5) Розпізнаємо теперішній стан шахтної підйомної машини.

Під час тестування та експериментів було виявлено що точність відповіді про теперішній стан нейронної мережі відповідала дійсності близько 70% випадках, що є добрим показником при використанні згенерований даних для навчання мережі на основі еталонних. Для того щоб отримати більшу точність необхідно доповняти данні для навчання, проводити подальші експерименти та визначати погрішність навчальних даних для штатних та нештатних станів.

3.4 Опис програми використаної в роботі

Головним завданням при розробці вже відомої нейронної мережі Кохонена, була можливість використання в подальшому отриманих результаті. Для роботи над отриманим результатом нейронної мережі, подальшому використанні розробленого алгоритму а також тестуванні і проведення експериментів програма отримала необхідні доповнення і модифікації.

Розроблена програма являє собою консольний демонстративних додаток, який реалізує нейронну мережу Кохонена, і використовує результат мережі для розпізнавання нештатних станів шахтних підйомних машин використовуючи розроблений алгоритм.

Нижче представлена діаграма класів розробленої програми: (рис 3)

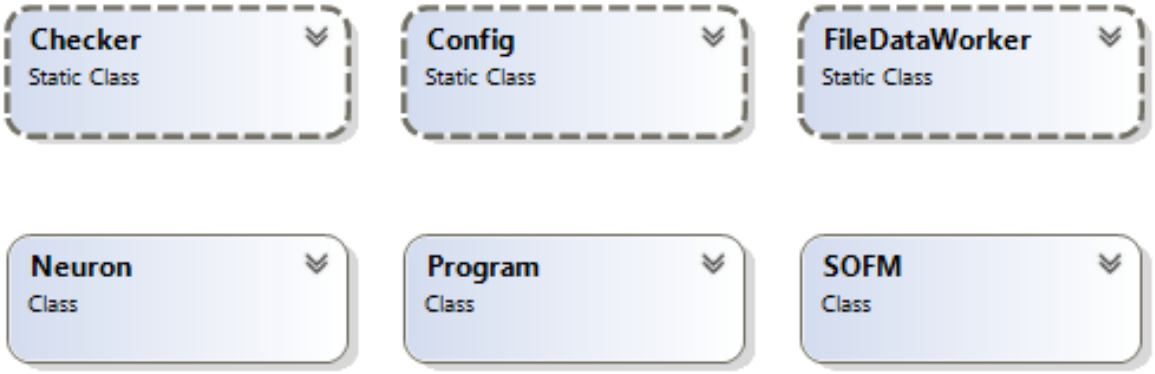

Рис. 3.5 Діаграма класів програми

Клас SOFM

Клас SOFM є реалізацією модифікованої нейронної мережі Кохонена для використання результатів роботи мережі у подальшому.

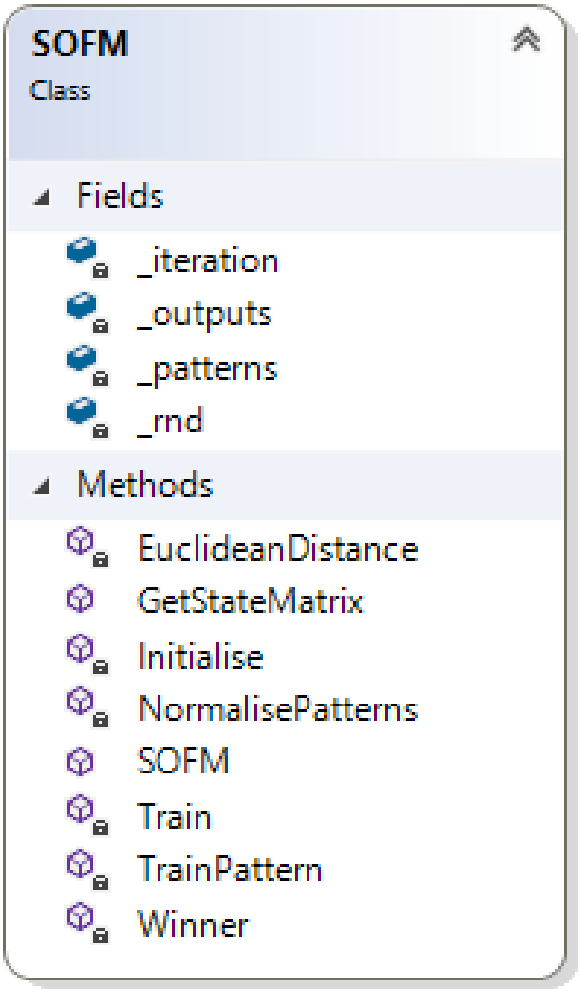

Рис. 3.6 Клас SOFM

Таблиця 3.2 Методи класу SOFM

EuclidianDistance	Метод для розрахування Евклідової відстані між точками
GetStateMatrix	Генерує матрицю відповідності елементів вихідної матриці нейронів до їх відношеня відповідно вхідним даним
Initiialise	Ініціалізація матриці нейронів
NormalisePatterns	Нормалізує вхідні данні
Train	Ініціює навчання мережі
TrainPattern	Обновлює ваги нейронів і вираховує теперішню помилку
Winner	Знаходить нейрона-переможця згідно з алгоритму мережі (нейрон з мінімальною відстанню)
SOFM	Конструктор класу

Клас Neuron

Клас є представленням нейрона в мережі Кохонена.

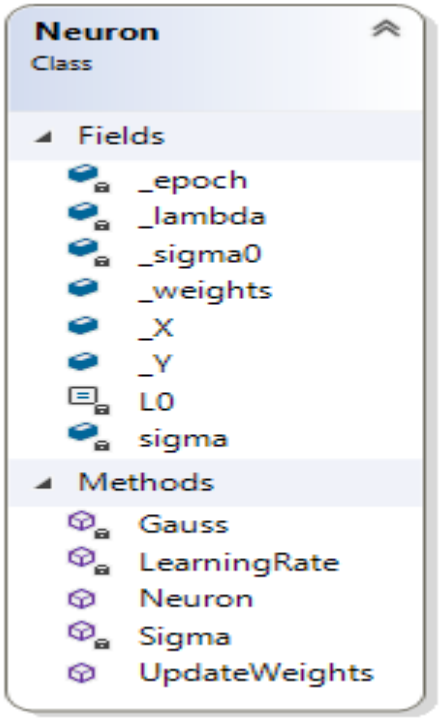

Рис. 3.7 Клас Neuron

Таблиця 3.3 Методи класу Neuron

Neuron	Конструктор класу
LearningRate	Визначає оцінку навчання нейронної мережі
Gauss	Функція околиці, змінює ваги нейронів що поруч з вказаним
Sigma	Визначає значення функції sigma для визначення функції околиці Гауса
UpdateWeights	Обновлення вагів нейрона

Клас FileDataWorker

Допоміжний статичний клас для роботи з файлами, існую для зручності тестування та проведення експериментів.

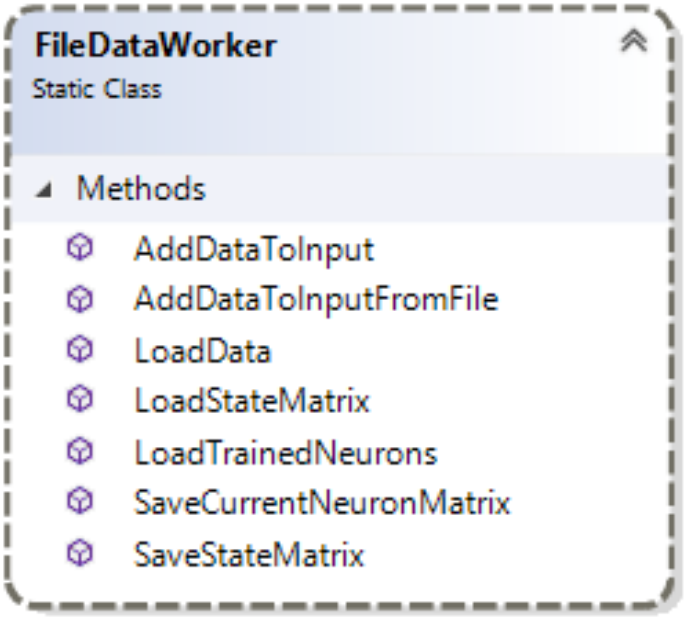

Рис. 3.8 Клас FileDataWorker

Таблиця 3.4 Методи класу FileDataWorker

AddDataToInput	Дозволяє додати до файла з тестовими даними для навчання заданий тестовий зразок
AddDataToInputFromFile	Дозволяє додати до файла з тестовими даними для навчання тестові зразки за заданого файла
LoadData	Завантажує данні для навчання

LoadStateMatrix	Завантажує матрицю станів
LoadTrainedNeurons	Завантажує вихідну матрицю нейронів (результат навчання нейронної мережі)
SaveCurrentNeuronMatrix	Зберігає вихідну матрицю нейронів (результат навчання нейронної мережі)
SaveStateMatrix	Зберігає матрицю станів

Клас Checker

Виконує перевірку стану шахтної підйомної машини на основі вихідної матриці нейронів і матриці стану.

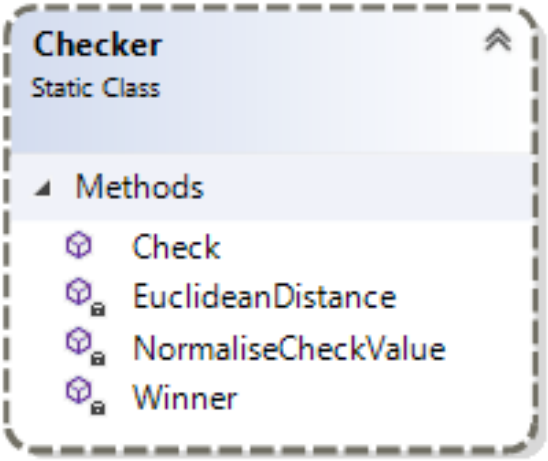

Рис. 3.9 Клас Checker

Таблиця 3.5 Методи класу Checker

Check	Виконує перевірку стану шахтної підйомної машини на основі вихідної матриці нейронів і матриці стану.
EuclidianDistance	Метод для розрахування Евклідової відстані між точками.
Winner	Знаходить нейрона-переможця згідно з алгоритму мережі (нейрон з мінімальною відстанню)
NormileseCheckValue	Нормалізує тестові данні відповідно до данних які використовувались при навчанні

3.5 Опис роботи програми

Використовуючи підготовлені тестові данні, які включають зразки даних штатних та нештатних станів, навчаємо нейрону мережу. В розробленому додатку не передбачено відображення прогресу навчання, але при необхідності це просто зробити.

В результаті отримаємо вихідну матрицю нейронів в збереженому файлі, та матрицю станів в якій виявлено відповідність до штатного, нештатного та підозрілого до нештатного станів.

Використовуючи клас Checker і його метод Check, вказуємо данні для перевірки стану шахтної підйомної машини відповідно їм.

Приклад відповіді (Рис. 3.10, Рис. 3.11, Рис. 3.12), якщо данні відповідають штатному стану і елемент вихідної матриці відповідає лише штатним вхідним тестовим даним:

```
Вхідні тестові данні
5.42
0.02
41

Визначений елемент вихідной матриці нейронів для тестових данних
2,5
Визначеня результата
Штатний стан ШПМ.
0
Виконано.
```

Рис. 3.10 Штатний стан приклад 1

```
Вхідні тестові данні
5.31
-0.056
34

Визначений елемент вихідной матриці нейронів для тестових данних
3,6
Визначеня результата
Штатний стан ШПМ.
0
Виконано.
```

Рис. 3.11 Штатний стан приклад 2

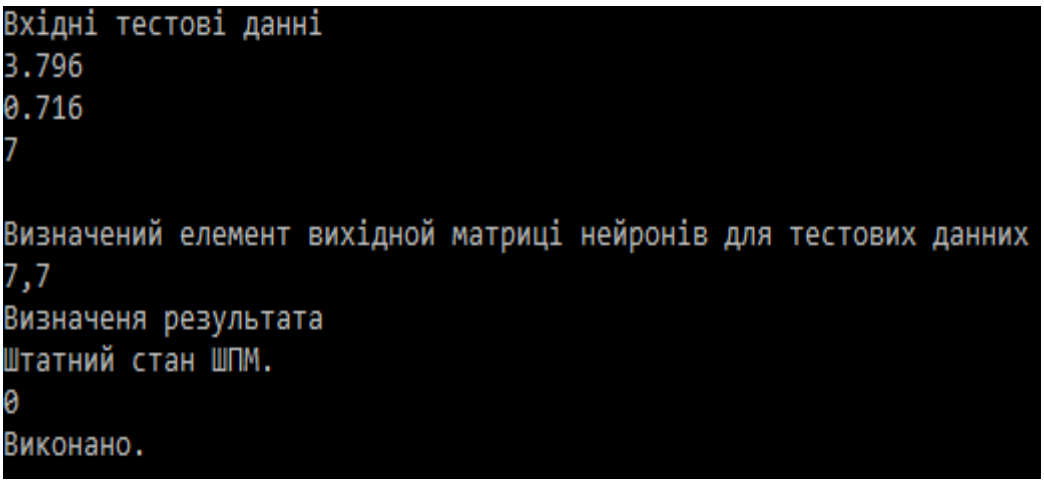

Рис. 3.12 Штатний стан приклад 3

Приклад відповіді (Рис. 3.13, Рис 3.14, Рис 3.15), якщо данні не відповідають штатному стану і елемент вихідної матриці відповідає лише не штатним вхідним тестовим даним:

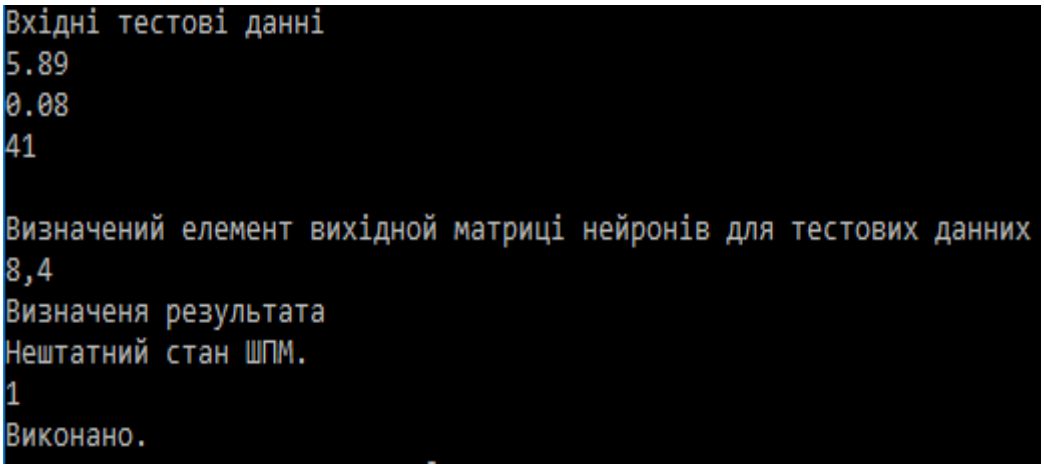

Рис. 3.13 Нештатний стан приклад 1

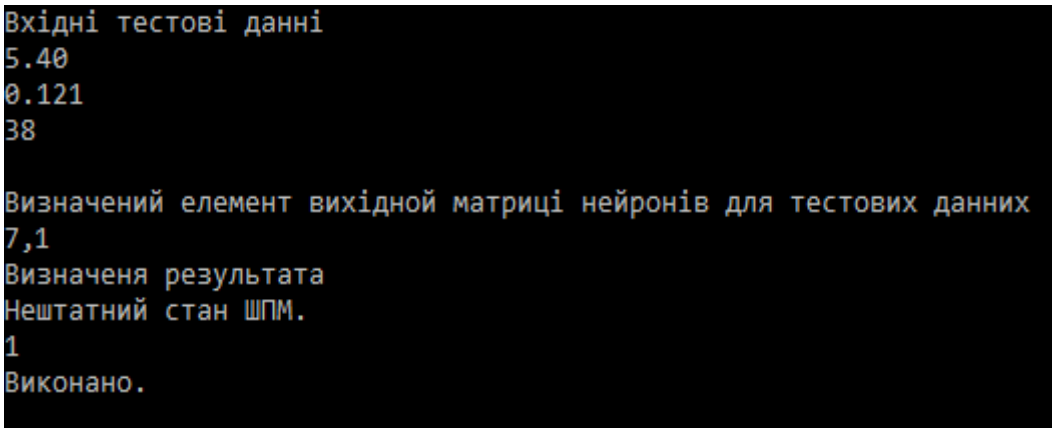

Рис. 3.14 Нештатний стан приклад 2

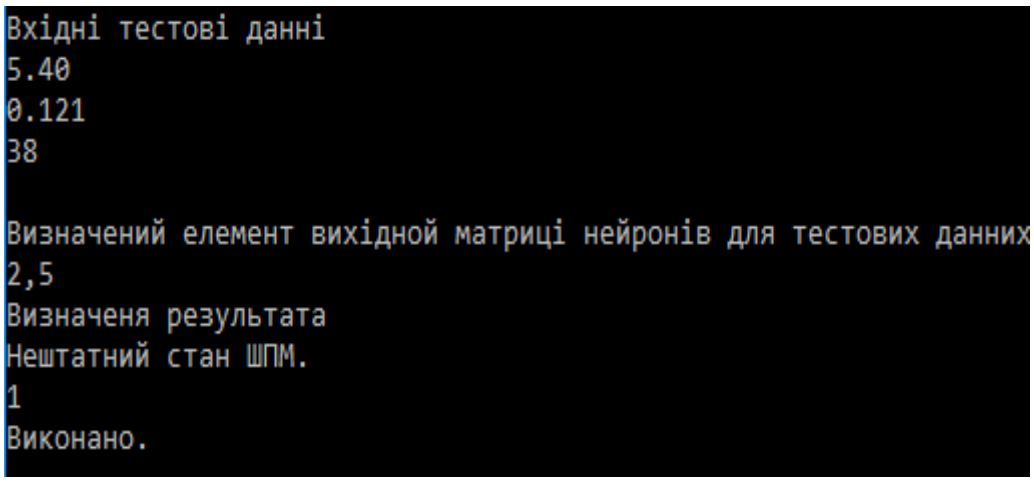

Рис. 3.15 Нештатний стан приклад 3

Приклад відповіді (Рис. 3.16, Рис. 3.17, Рис. 3.18), якщо данні відповідають штатному стану але елемент вихідної матриці відповідає так само як штатним вхідним тестовим даним так і не штатним:

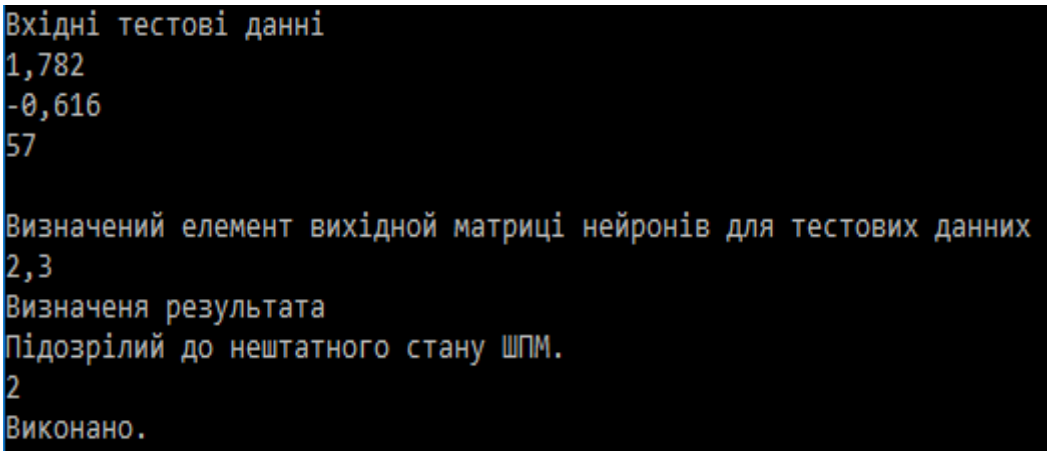

Рис. 3.16 Підозрілий стан приклад 1

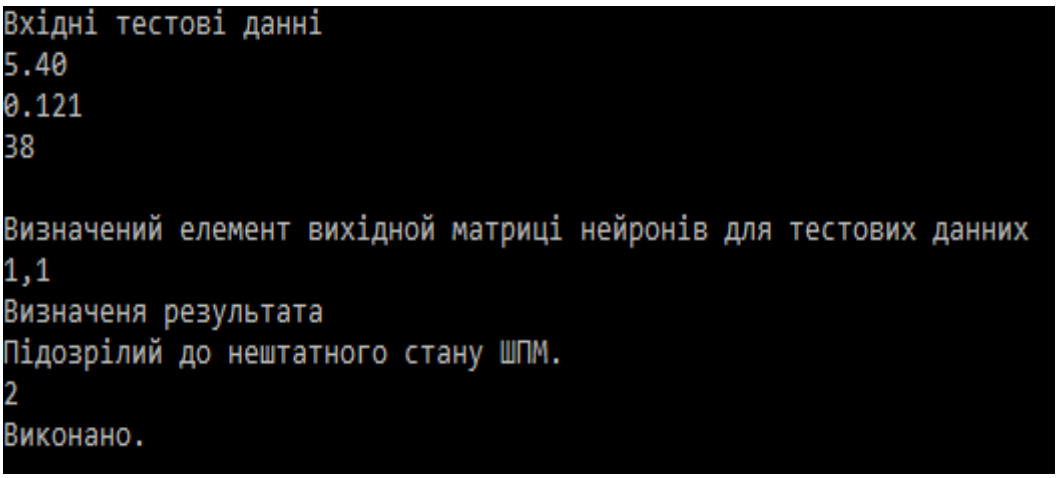

Рис. 3.17 Підозрілий стан приклад 2

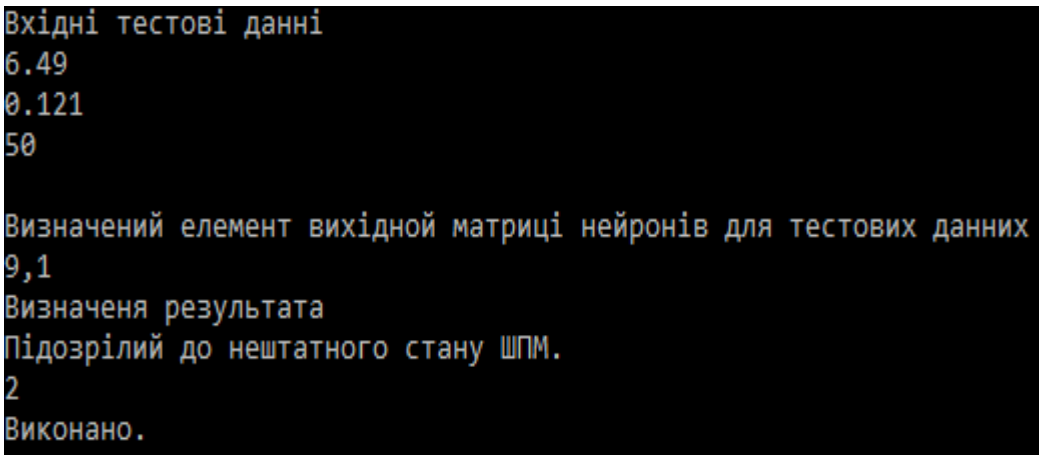

Рис. 3.18 Підозрілий стан приклад 3

Висновки

Розроблений алгоритм є ефективним методом удосконалення використання нейронної мережі Кохонена при розпізнавання нештатних станів на шахтних підйомних машинах. Проведені тести і експерименти показують, що виконане удосконалення є ефективним методом розпізнавання нештатних станів на шахтних підйомних машинах. Використання розробленого алгоритму дозволить полегшити роботу технічним службам для слідкування за станом шахтних підйомних машин.

4. ЕКОНОМІЧНА ЧАСТИНА

В дипломному проекті було розглянуто удосконалення використання нейронних мереж на шахтних підйомних машинах для розпізнавання нештатних станів. Як результат була розроблена, реалізована та відлагоджена нейронна мережа яка допомагає системі управління шахтними підйомними машинами розпізнавати нештатні стани.

1) Передбачуване число операторів – 1145.

2) Коефіцієнт складності програми – 1,4.

3) Коефіцієнт корекції програми в ході її розробки – 0,1.

4) Коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі – 1,2.

5) Коефіцієнт кваліфікації програміста – 0,8.

6) Середня годинна заробітна плата програміста – 35 грн/год.

7) Вартість машино-годин ЕОМ – 10 грн/год.

4.1. Визначення трудомісткості розробки програмного забезпечення

Трудомісткість розробки ПЗ можна розрахувати за формулою:

$$t = t_о + t_u + t_а + t_п + t_{отл} + t_д, \text{людино-годин}$$

де t_o – витрати праці на підготовку й опис поставленої задачі (приймається 50);

t_u – витрати праці на дослідження алгоритму рішення задачі;

t_a – витрати праці на розробку блок-схеми алгоритму;

t_n – витрати праці на програмування по готовій блок-схемі;

$t_{отл}$ – витрати праці на налагодження програми на ЕОМ;

$t_д$ – витрати праці на підготовку документації.

$$t = 50 + 34 + 96 + 96 + 513 + 228 = 1017 \text{ людино-годин}$$

Складові витрати праці визначаються через умовне число операторів у ПЗ, яке розробляється.

Умовне число операторів (підпрограм):

$$Q = q \cdot C \cdot (1+p),$$

де q – передбачуване число операторів;

C – коефіцієнт складності програми (в інтервалі від 1.25 до 2.0);

p – коефіцієнт корекції програми в ході її розробки (в інтервалі від 0.05 до 1).

$$Q = 1145 \cdot 1,4\ (1+0,1) = 1763$$

Витрати праці на вивчення опису задачі t_u визначається з урахуванням уточнення опису і кваліфікації програміста:

$$t_u = \frac{Q \cdot B}{(75...85) \cdot k}, \text{людино-годин},$$

де B – коефіцієнт збільшення витрат праці внаслідок недостатнього опису задачі (в інтервалі від 1.2 до 1.5);

k – коефіцієнт кваліфікації програміста, обумовлений від стажу роботи з даної спеціальності.

Таблиця 4.1

Значення коефіцієнта кваліфікації програміста

Стаж програміста	Значення коефіцієнта k
до 2-х років	0,8
від 2 до 3 років	1,0
від 3 до 5 років	1,1 … 1,2
від 5 до 10 років	1,2 … 1,3
понад 10 років	1,3 … 1,5

$$t_u = \frac{1763 \cdot 1,2}{77 \cdot 0,8} = 34 \text{ людино-годин}$$

Витрати праці на розробку алгоритму рішення задачі:

$$t_a = \frac{Q}{(20...25) \cdot k}, \text{людино-годин}$$

$$t_a = \frac{1763}{23 \cdot 0,8} = 96 \text{ людино-годин}$$

Витрати на складання програми по готовій блок-схемі:

$$t_n = \frac{Q}{(20...25)\cdot k}, \text{людино-годин}$$

$$t_n = \frac{1763}{23\cdot 0{,}8} = 96 \text{ людино-годин}$$

Витрати праці на налагодження програми на ЕОМ:

– за умови автономного налагодження одного завдання:

$$t_{отл} = \frac{Q}{(4...5)\cdot k}, \text{людино-годин}$$

$$t_{отл} = \frac{1763}{4{,}3\cdot 0{,}8} = 513 \text{ людино-годин}$$

– за умови комплексного налагодження завдання:

$$t^k_{отл} = 1{,}5\cdot t_{отл}, \text{ людино-годин}$$

$$t^k_{отл} = 1{,}5\cdot 513 = 770 \text{ людино-годин}$$

Витрати праці на підготовку документації:

$$t_д = t_{др} + t_{до}, \text{людино-годин}$$

$$t_д = 130 + 98 = 228 \text{ людино-годин,}$$

де $t_{др}$ – трудомісткість підготовки матеріалів і рукопису.

$$t_{др} = \frac{Q}{(15...20)\cdot k}, \text{людино-годин}$$

$$t_{др} = \frac{1763}{17\cdot 0{,}8} = 130 \text{ людино-годин,}$$

$t_{до}$ – трудомісткість редагування, печатки й оформлення документації

$$t_{до} = 0{,}75\cdot t_{др}, \text{людино-годин}$$

$$t_{до} = 0{,}75\cdot 130 = 98 \text{ людино-годин}$$

4.2 Витрати на створення програмного забезпечення

Витрати на створення ПЗ $К_{по}$ включать витрати на заробітну плату виконавця програми $З_{зп}$ і витрат машинного часу, необхідного на налагодження

програми на ЕОМ.

$$К_{ПО} = З_{ЗП} + З_{МВ}, \text{грн.}$$

$$К_{ПО} = 35595 + 5130 = 40725 \text{ грн.}$$

Заробітна плата виконавців визначається за формулою:

$$З_{ЗП} = t \cdot С_{пр}, \text{грн.}$$

де t – загальна трудомісткість, людино-годин;

$С_{пр}$ – середня годинна заробітна плата програміста, грн/година

$$З_{ЗП} = 1017 \cdot 35 = 35595 \text{ грн.}$$

Вартість машинного часу, необхідного для налагодження програми на ЕОМ:

$$З_{МВ} = t_{отл} \cdot С_{мч}, \text{грн.},$$

де $t_{отл}$ – трудомісткість налагодження програми на ЕОМ, год.

$С_{мч}$ – вартість машино-годин ЕОМ, грн/год.

$$З_{МВ} = 513 \cdot 10 = 5130 \text{грн.}$$

Визначені в такий спосіб витрати на створення програмного забезпечення є частиною одноразових капітальних витрат на створення АСУП.

Очікуваний період створення ПЗ:

$$T = \frac{t}{B_k \cdot F_p}, \text{міс.},$$

де B_k – число виконавців;

F_p – місячний фонд робочого часу (при 40-годинному робочому тижні F_p=176 годин).

$$T = \frac{1017}{1 \cdot 176} = 6 \text{ міс.}$$

4.3. Маркетингові дослідження ринку збуту розробленого програмного продукту

Своєчасне розпізнавання нештатних станів шахтних підйомних машин – одне и з найбільш важливих завдань для забезпечення безпеки і стабільної роботи на шахтних підйомних машинах.

У дипломній роботі розглядається методи використання і типи нейронних мереж які можна використати для удосконалення розпізнавання нештатних станів на шахтних підйомних машинах.

Актуальність даної роботи диктується умовами постійної небезпеки на шахтах України. Контроль стану механізмів на шахтах, в тому числі шахтних підйомних машин, є пріоритетним завданням для шахтних установ. За останні роки а то і десятиліття, механізми на шахтних підйомних машин застаріли, і потребують постійного контролю стану, попередження і розпізнавання нештатних станів.

Нині для розпізнавання нештатних станів на шахтних підйомних машинах України використовують застарілі аналогові системи, чи дуже примітивне ПО (у порівнянні з ПО шахтних підйомних машин наприклад у США).

Практична значимість проекту полягає в розробці нейронної мережі на основі мереж Кохонена, за допомогою якої можливо удосконалити можливості контролю стану шахтних підйомних машин.

Розвиток української індустрії ПЗ гальмують такі фактори:

– обмежене фінансування;

– недосконале законодавство;

– орієнтація багатьох фірм розробників ПЗ на термінову вигоду;

– орієнтація багатьох державних чиновників на західні технології;

– відсутність серйозних маркетингових досліджень;

– нелегальне використання ПЗ.

Вивчивши ринок на плагіати, можу сказати, що існують приклади використання різних типів нейронних мереж для розпізнавання нештатних станів на шахтних підйомних машинах, але розроблених і удосконалених продуктів які використовують за основу мережу Кохонена немає.

Розроблене мною програмне забезпечення унікальне, бо моделює стан машини на основі нормальних станів машини, і дозволяє перевірити в реальному часі, чи є теперішній стан шахтної підйомної машини задовільним або нештатним.

При впровадженні даної нейронної мережі в роботу шахтного управління, буде значне полегшення для працівників вугільної промисловості, а також допоможе в подальшому удосконаленні роботи шахтних підйомних машин.

Програма вкрай необхідна для виявлення рішень для можливостей запобігання поломок чи аварій.

4.4. Оцінка економічної ефективності впровадження програмного забезпечення

Економічна оцінка ефективності пропонованого впровадження оцінена за системою показників, які використовуються у міжнародній і вітчизняній практиці.

При оцінці економічної ефективності використані наступні показники:

– чиста поточна вартість (NPU);
– строк окупності капітальних вкладень;
– індекс прибутковості;
– коефіцієнт ефективності інвестицій.

При ухваленні рішення стосовно доцільності впровадження проекту необхідно враховувати значення всіх показників. Розрахунок показаний у таблиці 4.2.

Таблиця 4.2

Розрахунок чистих грошових надходжень від розробки ПЗ.

Показники, грн	По місяцям							Всього за 6 місяців	Середнє за 6 місяців
	0	1	2	3	4	5	6		
1. Інвестиції ПЗ	40725	-	-	-	-	-	-	41295	6883
2. Витрати до впровадження ПЗ	-	27000	1885	1896	1905	2376	1888	36950	6158
- на придбання обладнання	-	23900	-	-	-	-	-	23900	3983
- на придбання витратних матеріалів	-	1200	-	-	-	500	-	1700	283
- на маркетинг	-	1800	1800	1800	1800	1800	1800	10800	1800
- на електроенергію	-	100	85	96	105	76	88	550	92
3. Витрати після впровадження ПЗ	-	1680	1690	1680	1679	1680	1675	10084	1680
- на оренду приміщення	-	500	500	500	500	500	500	3000	500
- на електроенергію	-	80	90	80	79	80	75	484	81
- на рекламу	-	1000	1000	1000	1000	1000	1000	6000	1000
- на Інтернет	-	100	100	100	100	100	100	600	100
4. Економія	-	25320	195	216	226	696	213	26866	4478
5. Амортизація	-	6882,5	6882,5	6882,5	6882,5	6882,5	6882,5	41295	6882,5
6. Чисті грошові надходження	-	18437.5	7077,5	7098,5	7108,5	7578,5	7095,5	68161	11360
7. Коефіцієнт дисконтування	-	0,870	0,756	0,658	0,572	0,497	0,432	-	-
8. Дисконтовані грошові находження	-	28016	5351	4671	4066	3767	3065	48936	8156

Коефіцієнти економічної ефективності

Чиста поточна вартість доходів:

$$NPU = 48936 – 40725 = 8211 \text{ грн.} > 0$$

Строк окупності:

$$T = 40725/8156 = 5 \text{ місяців}$$

Індекс прибутковості:

$$ИД = 48936/40725 = 1,2$$

Показник економічної ефективності (NPU – чиста поточна вартість доходів за реалізації впровадження складе 8211 грн., тобто відповідає умовам ефективності, тому що NPU > 0. Середній строк окупності капітальних вкладень складе 5 місяців. Індекс прибутковості за 6 місяців складе 1,2, тобто ИД > 1, проект варто прийняти. Таким чином, показник ефективності свідчить про те, що дане впровадження є економічно вигідним.

Висновки

В дипломному проекті виконано «Удосконалення застосування нейронних мереж при розпізнаванні нештатних ситуацій на шахтних підйомних машинах».

При розробці, тестуванні і проведення експериментів програмного забезпечення для розпізнаванні нештатних ситуацій на шахтних підйомних машинах витрати на його створення склали 40725 грн., очікуваний період створення – 6 місяців та визначено трудомісткість розробленої інформаційної системи (1017 люд-год). Показник економічної ефективності (NPU – чиста поточна вартість доходів за термін реалізації впровадження) становить 8211 грн, тобто відповідає умовам ефективності, т. к. NPU>0. Середній термін окупності капітальних вкладень становить 5 місяців.

ВИСНОВКИ

Актуальність даної роботи є необхідність застосування сучасних технологій для забезпечення надійності та безпеки експлуатації шахтних підйомних машин. Шахтні підйомні установки забезпечують видачу корисних копалин, переміщення людей і вантажів. Від надійності роботи цієї найважливішої ланки технологічного ланцюга залежить безперебійність роботи всього гірничодобувного підприємства. Будь-яка аварійна ситуація на підйомі веде до зупинки підприємства.

Аналізуючи теперішній стан шахтних підйомних машин на шахтах України, можна зробити висновок що за останні роки парк шахтних підйомних машин сильно застарів. Термін служби більшості з них перевищує 25 років. Такий же термін служби мають привід шахтних підйомних машин, система управління цим приводом, обладнання шахтного стовбура, стовбурова сигналізація та інші, життєво важливі елементи шахтних підйомних установок. У зв'язку з тим, що одночасна заміна всіх шахтних підйомних машин та інших елементів шахтних підйомних установок неможлива. Тому питання безпеки виходить на перше місце.

Використання нейронних мереж для розпізнавання нештатних станів на шахтних підйомних машинах дозволяє значно підвищити рівень безпеки, одна в українській гірничий промисловості вони використовуються не часто, тому удосконалення їх використання є актуальною темою для роботи.

Розроблений алгоритм є ефективним методом удосконалення використання нейронної мережі Кохонена при розпізнавання нештатних станів на шахтних підйомних машинах. Використання розробленого алгоритму дозволить полегшити роботу технічним службам для слідкування за станом шахтних підйомних машин.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Беркинблит М. Б. Нейронные сети. — М.: МИРОС и ВЗМШ РАО, 1993. — 96 с. — ISBN 5-7084-0026-9.
2. Вороновский Г. К., Махотило К. В., Петрашев С. Н., Сергеев С. А. Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности. — арьков: Основа, 1997. — 112 с. — ISBN 5-7768-0293-8.
3. Голубев Ю. Ф. Нейросетевые методы в мехатронике. — М.: Изд-во Моск. унта, 2007. — 57 с. — ISBN 978-5-211-05434-9.
4. Горбань А.Н. Обучение нейронных сетей. — М.: СССР-США СП «Параграф», 1990. — 160 с.
5. Горбань А.Н., Россиев Д.А. Нейронные сети на персональном компьютере. — Новосибирск: Наука, 1996. — 276 с. — ISBN 5-02-0311960.
6. Горбань А. Н., Дунин-Барковский В. Л. и др. Нейроинформатика. — Новосибирск: Наука, 1998.
7. Еремин Д. М., Гарцеев И. Б. Искусственные нейронные сети в интеллектуальных системах управления. — М.: МИРЭА, 2004. — 75 с. — ISBN 5-7339-0423-2.
8. Каллан Р. Основные концепции нейронных сетей = The Essence of Neural Networks First Edition. — М.: Вильямс, 2001. — 288 с. — ISBN 5-8459-0210-X.
9. Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика. — М.: Горячая линия - Телеком, 2001. — 382 с. — ISBN5-93517031-0.
10. Миркес Е. М. Нейрокомпьютер. Проект стандарта. — Новосибирск: Наука, 1999. — 337 с. — ISBN 5-02-031409-9. Другие копии онлайн: Нейрокомпьютер. Проект стандарта.
11. Осовский Станислав. Нейронные сети для обработки информации = Sieci neuronowe do przetwarzania informacji (польск.) / Перевод

И. Д. Рудинского. — М.: Финансы и статистика, 2004. — 344 с. — ISBN 5-279-02567-4.

12. Савельев А. В. На пути к общей теории нейросетей. К вопросу о сложности // Нейрокомпьютеры: разработка, применение. — 2006. — № 4—5. — С. 4—14.
13. Сигеру Омату, Марзуки Халид, Рубия Юсоф. Нейроуправление и его приложения = Neuro-Control and its Applications. 2-е изд. — М.: ИПРЖР, 2000. — 272 с. — ISBN 5-93108-006-6.
14. Тадеусевич Рышард, Боровик Барбара, Гончаж Томаш, Леппер Бартош. Элементарное введение в технологию нейронных сетей с примерами программ / Перевод И. Д. Рудинского. — М.: Горячая линия — Телеком, 2011. — 408 с. — ISBN 978-5-9912-0163-6..
15. Терехов В. А., Ефимов Д. В., Тюкин И. Ю. Нейросетевые системы управления. — М.: Высшая школа, 2002. — 184 с. — ISBN 5-06-004094-1.
16. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика = Neural Computing. Theory and Practice. — М.: Мир, 1992. — 240 с. — ISBN 5-03-002115-9.
17. Хайкин С. Нейронные сети: полный курс = Neural Networks: A Comprehensive Foundation. 2-е изд. — М.: Вильямс, 2006. — 1104 с. — ISBN 0-13-273350-1.
18. Ясницкий Л. Н. Введение в искусственный интеллект. — М.: Издат. центр «Академия», 2005. — 176 с. — ISBN 5-7695-1958-4.
19. Корняков М.В. Защита шахтних подъемных установок от динамических нагрузок при зависании подъемных сосудов в стволе : автореф. дис. на сосискание учен. Степени канд. техн. наук: специальность 05.05.06 «Горные машины» / М.В. Корняков - Иркутск, 2000. – 24 с.
20. Ильин С.Р. Механика шахтного подъема / С.Р. Ильин, С.С. Ильина, В.И. Самуся - Днепропетровск : Национальный горный университет, 2014. – 246с.

21. Эксплуатация шахтных подъемных установок / Пермь : Издательство Пермского национального исследовательского политехнического университета, 2015. – 315с.
22. Ткач А.А. Динамика аварийных режимов многоканатной подъемной установки с машиной наземного расположения : автореф. дис. на сосискание учен. степени канд. техн. наук: специальность 05.15.16 «Горные машины» / А.А. Ткач - Днепропетровск, 1996. – 16с.
23. Андрейчиков А.В., Андрейчикова О.Н. Интеллектуальные информационные системы: М. Наука, 2004.
24. Асаи К. и др. Прикладные нечеткие системы / пер. с япон.; под. ред. Тэрано Т., Асаи К. Сугэно М. – М.: Мир, 1993. –368с.
25. Дьяконов В., Круглов В. Математические пакеты расширения MATLAB: специальный справочник. – СПб.: Питер, 2001. – 480 с.
26. База знаний – Википедия [Электронный ресурс] – Режим доступа: http://ru.wikipedia.org/wiki/%C1%E0%E7%E0_%E7%ED%E0%ED%E8%E9
27. Базы данных [Электронный ресурс] – Режим доступа: http://li.romab.ru/intro_db_6.html
28. Муромцев Д.И. Введение в технологию экспертных систем. СПб: СПб ГУ ИТМО, 2005.
29. Нейлор К. Как построить свою экспертную систему: Пер. с англ. – М.: Энергоатомиздат, 1991. – 286 с.
30. Нечеткая логика в системах управления – журнал «Компьютерра» [Электронный ресурс] – Режим доступа: http://offline.computerra.ru/2001/415/13052/
31. Попов Э.В. Экспертные системы: Решение неформализованных задач в диалоге с ЭВМ. - М.: Наука. Гл. ред. физ.- мат. Лит., 1987.
32. Рутковская Д., Пилинский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы: Пер. с польск. И.Д. Рудинского. М.: Горячая линия – Телеком, 2007. – 452 с.

33. Терехов В.А., Нейросетевые системы управления: учебное пособие для вузов. – М.: Высш. шк., 2002. – 183 с.
34. Ясницкий Л.Н. Введение в искусственный интеллект: учебное пособие. М.: Академия, 2008. – 176 с.
35. Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. Пер. с англ. 240 с.
36. Саймон Хайкин Нейронные сети: полный курс Второе издание, 2006-1105с.
37. Rosenblatt F. 1962. Principles of Neurodinamics. New York: Spartan Books. (Русский перевод: Розенблатт Ф. Принципы нейродинамики. - М: Мир. - 1965.)
38. Minsky M. L, Papert S. 1969. Perseptrons. Cambridge, MA: MIT Press. (Русский перевод: Минский М. Л., Пейперт С. Персептроны. - М: Мир. - 1971.)
39. Баженов Р.И.Интеллектуальные информационные технологии. Биробиджан: ПГУ им. Шолом-Алейхема, 2011. 176 с.
40. Баженов Р.И. Информационная безопасность и защита информации: практикум. Биробиджан: Изд-во ГОУВПО «ДВГСГА», 2011. 140 с.
41. Баженов Р.И. Проектирование методики обучения дисциплины «Информационные технологии в менеджменте» // Современная педагогика. 2014. № 8 (21). С. 24-31.
42. Баженов Р.И. Об организации научно-исследовательской практики магистрантов направления «Информационные системы и технологии» // Современные научные исследования и инновации. 2014. № 9-2 (41). С. 62-69.
43. Иванников В., Ланнэ А. Matlab для DSP. Нейронные сети: графический интерфейс пользователя [Электронный ресурс]. URL: http://www.chipinfo.ru/literature/chipnews/200108/1.html#lanne8
44. Медведев В. С., Потемкин В. Г. Нейронные сети. MATLAB 6. М.: ДИАЛОГ-МИФИ, 2001. 630 с.

45. Миронов И.С., Скурлаев С.В. Распознавание образов при помощи нейроннойсети
46. Шеремет А.И., Перепелица В.В., Денисова А.М. Проектирование нейронной сети для распознавания символов в программной среде MATLAB[Электронныйресурс]. URL: http://nauka.zinet.info/13/sheremet.php
47. Principe J.C., Euliano N.R., Lefebvre W.C. Neural and Adaptive Systems. Fundamentals Through Simulations. New York. John Wiley & Sons Inc. 2000.
48. Luo F-L., Unbehauen R. Applied Neural Networks for Signal Processing. Cambridge University Press. 1998.
49. Demuth H., Beale M. Neural Network Toolbox. For Use with MATLAB. The MathWorks Inc. 1992-2000.
50. Awadalla M. H. A., Ismaeil I. I., Sadek M. A. Spiking neural network- based control chart pattern recognition //Journal of Engineering and Technology Research. 2011. Т. 3. №. 1. С. 5-15.
51. Dede G., Sazlэ M. H. Speech recognition with artificial neural networks //Digital Signal Processing. 2010. Т. 20. №. 3. С. 763-768.
52. Методические указания о порядке проведения испытаний стальных канатов на канатно-испытательных станциях (РД-15-12-2007) (утв. приказом Ростехнадзора от 31.07.07 № 522). Сер. 05. Вып. 16 / Ш.М. Тугуз, В.Л. Беляк, В.И. Завгородний, С.Н. Подображин, Г.Д. Трифанов; ОАО «НТЦ по промышленной безопасности». – М., 2007. – 100 с. 50
53. Когаев В.П. Статистические закономерности усталости металлов: автореф. дис. д-ра техн. наук. – М.: Изд-во ИМАШ, 1968. – 55 с.
54. Димашко А.Д., Гершиков И.Я., Кревневич А.А. Шахтные электрические лебедки и подъемные машины: справочник. – 4-е изд., перераб. и доп. – М.: Недра, 1973. – 364 с.
55. Болотин В.В. Прогнозирование ресурса машин и конструкций. – М.: Машиностроение, 1984. – 312 с.

ДОДАТОК А

Клас SOFM
Файл SOFM.cs

```
usingSystem;
usingSystem.Collections.Generic;
usingSystem.IO;
usingSystem.Text;

namespaceSOFM_diploma.SofmNetwork
{
internalclassSOFM
{
privateNeuron[,]_outputs;//Collectionofweights.
privateint_iteration;//Current_iteration.
privateRandom_rnd=newRandom();
privateList<double[]>_patterns=newList<double[]>();

publicSOFM()
{
Initialise();
_patterns=FileDataWorker.LoadData();
NormalisePatterns();
Train(0.000000001);
GetStateMatrix();
FileDataWorker.SaveCurrentNeuronMatrix(_outputs);
}

privatevoidInitialise()
{
_outputs=newNeuron[Config.OutputGridLength,Config.OutputGridLength];
for(inti=0;i<Config.OutputGridLength;i++)
{
for(intj=0;j<Config.OutputGridLength;j++)
{
_outputs[i,j]=newNeuron(i,j,Config.OutputGridLength){_weights=newdouble[Confi
g.InputDataCount]};
for(intk=0;k<Config.InputDataCount;k++)
{
_outputs[i,j]._weights[k]=_rnd.NextDouble();
}
}
}
}
```

```
privatevoidNormalisePatterns()
{
doublemin=_patterns[0][0];
doublemax=_patterns[0][0];
for(intj=0;j<Config.InputDataCount;j++)
{
for(inti=0;i<_patterns.Count;i++)
{
if(min>_patterns[i][j])
{
min=_patterns[i][j];
}
if(max<_patterns[i][j])
{
max=_patterns[i][j];
}
}
}

for(intj=0;j<Config.InputDataCount;j++)
{
for(inti=0;i<_patterns.Count;i++)
{
_patterns[i][j]=(_patterns[i][j]-min)/(max-min);
}
}
}

privatevoidTrain(doublemaxError)
{
doublecurrentError=double.MaxValue;//Firstiterationforwhile

while(currentError>maxError)//CheckHALT
{
currentError=0;
List<double[]>TrainingSet=newList<double[]>();
foreach(double[]patternin_patterns)
{
TrainingSet.Add(pattern);
}
for(inti=0;i<_patterns.Count;i++)
{
double[]pattern=TrainingSet[_rnd.Next(_patterns.Count-i)];
currentError=TrainPattern(pattern);
TrainingSet.Remove(pattern);
```

```
}
}
}

privatedoubleTrainPattern(double[]pattern)
{
doubleerror=0;
Neuronwinner=Winner(pattern);
for(inti=0;i<Config.OutputGridLength;i++)
{
for(intj=0;j<Config.OutputGridLength;j++)
{
error+=_outputs[i,j].UpdateWeights(pattern,winner,_iteration);
}
}
_iteration++;
returnMath.Abs(error/Config.OutputGridLength*Config.OutputGridLength);
}

privateNeuronWinner(double[]pattern)
{
Neuronwinner=null;
doublemin=double.MaxValue;
for(inti=0;i<Config.OutputGridLength;i++)
for(intj=0;j<Config.OutputGridLength;j++)
{
doubledistance=EuclideanDistance(pattern,_outputs[i,j]._weights);
if(distance<min)
{
min=distance;
winner=_outputs[i,j];
}
}
returnwinner;
}

privatedoubleEuclideanDistance(double[]pattern,double[]weights)
{
doublesum=0;
for(inti=0;i<pattern.Length;i++)
{
sum+=Math.Pow(pattern[i]-weights[i],2);
}
returnMath.Sqrt(sum);
}
```

```
publicvoidGetStateMatrix()
{
int[,]normalStateMatrix=newint[Config.OutputGridLength,Config.OutputGridLength
];
int[,]notNormalStateMatrix=newint[Config.OutputGridLength,Config.OutputGridLe
ngth];
int[,]resultStateMatrix=newint[Config.OutputGridLength,Config.OutputGridLength];

for(inti=0;i<Config.OutputGridLength;i++)
{
for(intj=0;j<Config.OutputGridLength;j++)
{
normalStateMatrix[i,j]=1;
notNormalStateMatrix[i,j]=0;
}
}

for(inti=0;i<Config.CountFirstNormalStateElements;i++)
{
Neuronn=Winner(_patterns[i]);
normalStateMatrix[n._X,n._Y]=0;
}

for(inti=Config.CountFirstNormalStateElements-1;i<_patterns.Count;i++)
{
Neuronn=Winner(_patterns[i]);
notNormalStateMatrix[n._X,n._Y]=1;
}

for(inti=0;i<Config.OutputGridLength;i++)
{
for(intj=0;j<Config.OutputGridLength;j++)
{
if(normalStateMatrix[i,j]==0&&notNormalStateMatrix[i,j]==1)
{
resultStateMatrix[i,j]=2;
}
elseif(normalStateMatrix[i,j]==notNormalStateMatrix[i,j])
{
resultStateMatrix[i,j]=normalStateMatrix[i,j];
}
else
{
resultStateMatrix[i,j]=8;
```

```
}
}
}
for(inti=0;i<Config.OutputGridLength;i++)
{
for(intj=0;j<Config.OutputGridLength;j++)
{
Console.Write(normalStateMatrix[i,j].ToString()+");
}
Console.WriteLine();
}
for(inti=0;i<Config.OutputGridLength;i++)
{
for(intj=0;j<Config.OutputGridLength;j++)
{
Console.Write(notNormalStateMatrix[i,j].ToString()+");
}
Console.WriteLine();
}
for(inti=0;i<Config.OutputGridLength;i++)
{
for(intj=0;j<Config.OutputGridLength;j++)
{
Console.Write(resultStateMatrix[i,j].ToString()+");
}
Console.WriteLine();
}

FileDataWorker.SaveStateMatrix(normalStateMatrix,Config.NormalStateMatrixFile)
;
FileDataWorker.SaveStateMatrix(notNormalStateMatrix,Config.UnNormalStateMatr
ixFile);
FileDataWorker.SaveStateMatrix(resultStateMatrix,Config.ResultStateMatrixFile);
}
}
      }
```

КласNeuron

ФайлNeuron.cs

```
usingSystem;
usingSystem.Collections.Generic;
usingSystem.Linq;
usingSystem.Text;
usingSystem.Threading.Tasks;
```

```
namespaceSOFM_diploma.SofmNetwork
{
publicclassNeuron
{
publicdouble[]_weights;
publicint_X;
publicint_Y;
privatereadonlydouble_epoch=1000.0;
privatereadonlydouble_sigma0;
privatereadonlydouble_lambda;
privatereadonlydoublesigma;
privateconstdoubleL0=0.2;

publicNeuron(intx,inty,intlength)
{
_X=x;
_Y=y;
_sigma0=length;//Math.Max(length*x,length*y)/2.0;
_lambda=_epoch/Math.Log(_sigma0);
}

privatedoubleGauss(Neuronwin,intit)
{
doubledistance=Math.Sqrt(Math.Pow(win._X-_X,2)+Math.Pow(win._Y-_Y,2));
returnMath.Exp(-Math.Pow(distance,2)/(Math.Pow(Sigma(it),2)));
}

privatedoubleLearningRate(intit)
{
returnMath.Exp(-it/_lambda)*L0;
}

privatedoubleSigma(intit)//sigma
{
returnMath.Exp(-it/_lambda)*_sigma0;
}

publicdoubleUpdateWeights(double[]pattern,Neuronwinner,intit)
{
doublesum=0;
for(inti=0;i<_weights.Length;i++)
{
doubledelta=Gauss(winner,it)*LearningRate(it)*(pattern[i]-_weights[i]);
_weights[i]+=delta;
```

```
sum+=delta;
}
returnsum/_weights.Length;
}
}
}
```

КласFileDataWorker

ФайлFileDataWorker.cs

```
usingSystem;
usingSystem.Collections.Generic;
usingSystem.IO;
usingSystem.Linq;
usingSystem.Text;
usingSystem.Threading.Tasks;
usingNewtonsoft.Json;

namespaceSOFM_diploma.SofmNetwork
{
publicstaticclassFileDataWorker
{
publicstaticvoidAddDataToInputFromFile()
{
List<double[]>newPatterns=newList<double[]>();
StreamReaderreaderOldData=File.OpenText(Config.InputFile);
while(!readerOldData.EndOfStream)
{
string[]line=readerOldData.ReadLine().Split(';');
double[]inputs=newdouble[Config.InputDataCount];
for(inti=0;i<Config.InputDataCount;i++)
{
inputs[i]=double.Parse(line[i]);
}
newPatterns.Add(inputs);
}
readerOldData.Close();

StreamReaderreaderNewData=File.OpenText(Config.InputAddDataFile);
while(!readerNewData.EndOfStream)
{
string[]line=readerNewData.ReadLine().Split(';');
double[]inputs=newdouble[Config.InputDataCount];
for(inti=0;i<Config.InputDataCount;i++)
{
inputs[i]=double.Parse(line[i]);
```

```
}
newPatterns.Add(inputs);
}
readerNewData.Close();

varcsv=newStringBuilder();
for(inti=0;i<newPatterns.Count;i++)
{
stringlabel="";
for(intj=0;j<Config.InputDataCount;j++)
{
label+=newPatterns[i][j];
if(j<newPatterns[i].Length-1)
{
label+=';';
}
}
csv.AppendLine(string.Format("{0}",label));
}
File.WriteAllText(Config.NewDataFile,csv.ToString());
Console.WriteLine("Datasuccesfulladded");
}

publicstaticvoidAddDataToInput(double[]newData)
{
List<double[]>newPatterns=newList<double[]>();
StreamReaderreaderOldData=File.OpenText(Config.InputFile);
while(!readerOldData.EndOfStream)
{
string[]line=readerOldData.ReadLine().Split(';');
double[]inputs=newdouble[Config.InputDataCount];
for(inti=0;i<Config.InputDataCount;i++)
{
inputs[i]=double.Parse(line[i]);
}
newPatterns.Add(inputs);
}
readerOldData.Close();

newPatterns.Add(newData);

varcsv=newStringBuilder();
for(inti=0;i<newPatterns.Count;i++)
{
stringlabel="";
```

```
for(intj=0;j<Config.InputDataCount;j++)
{
label+=newPatterns[i][j];
if(j<newPatterns[i].Length-1)
{
label+=';';
}
}
csv.AppendLine(string.Format("{0}",label));
}
File.WriteAllText(Config.NewDataFile,csv.ToString());
Console.WriteLine("Datasuccesfulladded");
}

publicstaticList<double[]>LoadData()
{
List<double[]>patterns=newList<double[]>();
StreamReaderreader=File.OpenText(Config.InputFile);
while(!reader.EndOfStream)
{
string[]line=reader.ReadLine().Split(';');
double[]inputs=newdouble[Config.InputDataCount];
for(inti=0;i<Config.InputDataCount;i++)
{
inputs[i]=double.Parse(line[i]);
}
patterns.Add(inputs);
}
reader.Close();
returnpatterns;
}

publicstaticint[][]LoadStateMatrix()
{
List<int[]>patterns=newList<int[]>();
StreamReaderreader=File.OpenText(Config.ResultStateMatrixFile);
while(!reader.EndOfStream)
{
string[]line=reader.ReadLine().Split(';');
int[]inputs=newint[Config.OutputGridLength];
for(inti=0;i<Config.OutputGridLength;i++)
{
inputs[i]=int.Parse(line[i]);
}
patterns.Add(inputs);
```

```
}
reader.Close();
returnpatterns.ToArray();
}

publicstaticvoidSaveCurrentNeuronMatrix(Neuron[,]_outputs)
{
stringneurons=JsonConvert.SerializeObject(_outputs);
varcsv=newStringBuilder();

csv.AppendLine(string.Format("{0}",neurons));

File.WriteAllText(Config.CurrentNeuronMatrixFile,csv.ToString());
}

publicstaticvoidSaveStateMatrix(int[,]matrix,stringfileName)
{
varcsv=newStringBuilder();
for(inti=0;i<Config.OutputGridLength;i++)
{
stringlabel="";
for(intj=0;j<Config.OutputGridLength;j++)
{
label+=matrix[i,j];
if(j<matrix.Length-1)
{
label+=';';
}
}
csv.AppendLine(string.Format("{0}",label));
}
File.WriteAllText(fileName,csv.ToString());
}

publicstaticNeuron[,]LoadTrainedNeurons()
{
stringneuronsString;
StreamReaderreaderData=File.OpenText(Config.CurrentNeuronMatrixFile);
neuronsString=readerData.ReadLine();

Neuron[,]outputs=JsonConvert.DeserializeObject<Neuron[,]>(neuronsString);
readerData.Close();
returnoutputs;
}
}
```

```
}
```

КласChecker

ФайлChecker.cs

```
usingSystem;
usingSystem.Collections.Generic;
usingSystem.IO;
usingSystem.Linq;
usingSystem.Text;
usingSystem.Threading.Tasks;

namespaceSOFM_diploma.SofmNetwork
{
publicstaticclassChecker
{
publicstaticvoidCheck(double[]checkValue)
{
Neuronn=Winner(NormaliseCheckValue(checkValue));
int[][]resultStateMatrix=FileDataWorker.LoadStateMatrix();
intstate=resultStateMatrix[n._X][n._Y];

Console.WriteLine("Визначенийелементвихіднойматрицінейронівдлятестовихда
нних");

Console.WriteLine("{0},{1}",n._X,n._Y);

Console.WriteLine("Визначенярезультата");
if(state==0)
{
Console.WriteLine("ШтатнийстанШПМ.");
}
elseif(state==1)
{
Console.WriteLine("НештатнийстанШПМ.");
}
elseif(state==2)
{
Console.WriteLine("ПідозрілийдонештатногостануШПМ.");
}
Console.WriteLine("{0}",state);
Console.WriteLine("Виконано.");
}

privatestaticdouble[]NormaliseCheckValue(double[]check)//temp
{
```

```
Console.WriteLine("Вхіднітестовіданні");
foreach(doubledincheck)
Console.WriteLine(d.ToString());
Console.WriteLine();

List<double[]>patterns=
FileDataWorker.LoadData();
doublemin=patterns[0][0];
doublemax=patterns[0][0];
for(intj=0;j<Config.InputDataCount;j++)
{
for(inti=0;i<patterns.Count;i++)
{
if(min>patterns[i][j])
{
min=patterns[i][j];
}
if(max<patterns[i][j])
{
max=patterns[i][j];
}
}
}

for(inti=0;i<check.Length;i++)
{
if(min>check[i])
{
min=check[i];
}
if(max<check[i])
{
max=check[i];
}
}

for(inti=0;i<check.Length;i++)
{
check[i]=(check[i]-min)/(max-min);
}
returncheck;
}
privatestaticNeuronWinner(double[]pattern)
{
Neuron[,]outputs=FileDataWorker.LoadTrainedNeurons();
```

```
Neuronwinner=null;
doublemin=double.MaxValue;
for(inti=0;i<Config.OutputGridLength;i++)
for(intj=0;j<Config.OutputGridLength;j++)
{
doubledistance=EuclideanDistance(pattern,outputs[i,j]._weights);
if(distance<min)
{
min=distance;
winner=outputs[i,j];
}
}
returnwinner;
}
privatestaticdoubleEuclideanDistance(double[]pattern,double[]weights)
{
doublesum=0;
for(inti=0;i<pattern.Length;i++)
{
sum+=Math.Pow(pattern[i]-weights[i],2);
}
returnMath.Sqrt(sum);
}
}
}
```

КласConfig

ФайлConfig.cs

```
usingSystem;
usingSystem.Collections.Generic;
usingSystem.Linq;
usingSystem.Text;
usingSystem.Threading.Tasks;
namespaceSOFM_diploma.SofmNetwork
{
publicstaticclassConfig
{
publicstaticstringInputFile=@"fileName.csv";
publicstaticintInputDataCount=3;

publicstaticintOutputGridLength=15;

publicstaticintCountFirstNormalStateElements=178;

publicstaticstringCurrentNeuronMatrixFile=@"fileName.csv";
```

```
publicstaticstringInputAddDataFile=@"fileName.csv";

publicstaticstringNewDataFile=@"fileName.csv";

publicstaticstringNormalStateMatrixFile=@"fileName.csv";

publicstaticstringUnNormalStateMatrixFile=@"fileName.csv";

publicstaticstringResultStateMatrixFile=@"fileName.csv";
}
}
```

ДОДАТОК Б

ВІДГУК

на дипломну роботу магістра на тему:

«Удосконалення застосування нейронних мереж при розпізнаванні нештатних ситуацій на шахтних підйомних машинах»
студента групи 122М-16-1 Піменова Олексій Андрійовича

1. Метою магістерської роботи є удосконалення застосування нейронних мереж для розпізнавання нештатних ситуацій на шахтних підйомних машинах використовуючи результат нейронної мережі Кохонена.
2. Актуальність даної роботи полягає у наявності значних недоліків у існуючих системах контролю шахтних підйомних машинах на шахтах Укрїни при розпізнаванні нештатних станів.
3. Тема дипломної роботи безпосередньо пов'язана з об'єктом діяльності магістра спеціальності 122 «Інформаційні управляючі системи та технології» галузі знань 12 «Інформаційні технології» – створення, дослідження, просування та реалізація комп’ютерних систем і програмних продуктів.
4. Наукова новизна отриманих результатів полягає у обґрунтуванні застосування нейронної мережі Кохонена, розробці алгоритму на основі результатів роботи мережі, який допоможе слідкувати за станом шахтної підйомної машини і розпізнавати нештатні стани на шахтних підйомних машинах.
5. Оригінальність технічних рішень при дослідженні полягає у створенні алгоритму для розпізнаванні нештатних станів на шахтних підйомних машинах використовуючи результат роботи нейронної мережі Кохонена.
6. Практична цінність полягає у розробці алгоритму який використовуючи результат нейронної мережі Кохонена дозволяє розпізнавати нештатні стани шахтних підйомних машин. В результаті розроблений алгоритм можна використовувати на практиці як допоміжний модуль при розпізнаванні нештатних ситуацій на шахтних підйомних машинах.
7. Оформлення дипломної роботи магістра виконано на сучасному рівні і відповідає вимогам, що пред'являються до робіт даної кваліфікації. Ступінь самостійності виконання досить висока.
8. Дипломна робота магістра в цілому заслуговує оцінки «відмінно», а автор - присвоєння кваліфікації «інженер з комп’ютерних систем».

Керівник дипломної
роботи магістра,
д.т.н., професор кафедри ПЗКС /Мещеряков Л.І./