МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

БОНДАРЧУК ВАДИМ ОЛЕКСАНДРОВИЧ

Допускається до захисту:
В.о. завідувач кафедри
прикладної математики та
кібербезпеки
д-р філос. з математики
_____Луценко А.В
«___»_____20___ р.

ЗАСТОСУВАННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ВИЯВЛЕННЯ АНОМАЛЬНОЇ ПОВЕДІНКИ КОРИСТУВАЧІВ ЦЕНТРІВ ОБРОБКИ ДАНИХ

Спеціальність 113 Прикладна математика

Кваліфікаційна (магістерська) робота

Науковий керівник:
Л.В. Загоруйко, к.т.н. доцент,
доцент кафедри прикладної математики
та кібербезпеки

_____
(підпис)

Оцінка _____ / _____ / _____
(бали/за шкалою ЄКТС/ за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця 2024

АНОТАЦІЯ

Бондарчук В.О. Застосування штучних нейронних мереж для виявлення аномальної поведінки користувачів центрів обробки даних. Спеціальність 113 «Прикладна математика». Освітня програма «Прикладна математика». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній роботі було вдосконалено структуру нейронної мережі для виявлення аномальної поведінки користувачів центрів обробки даних на основі технології *k*-мережі та розроблено програмний продукт для реалізації системи виявлення аномальної поведінки користувачів центрів обробки даних.

Мета дослідження. Вдосконалити структуру нейронної мережі для виявлення аномальної поведінки користувачів центрів обробки даних на основі технології *k*-мережі та розробити програмне забезпечення для виявлення аномальної поведінки користувачів центрів обробки даних, використовуючи відповідні технології штучних нейронних мереж, а також розробити застосунок для демонстрації роботи навчання мережі наочно.

Наукова новизна:

- Дістала подальший розвиток практична пропозиція впровадження технології *k*-карт у сферу захисту інформації у вигляді додатку.

- Розроблений додаток для виявлення аномальної поведінки користувачів та розроблено механізм хибного спрацьовування при виявленні атак у центрі обробки даних.

Ключові слова: *нейроподібна мережі, k-карти, машинне навчання, системи класифікаторів, системи виявлення аномальної поведінки, набір даних, попередня обробка даних.*

Загальний обсяг роботи: 73 сторінки, 16 рисунків, 3 таблиці, 37 джерел.

ANNOTATION

Bondarchuk V.O. Application of artificial neural networks for detecting anomalous behavior of users in data processing centers. Specialty 113 " Applied mathematics ". Educational program "" Applied mathematics ". Vasyl Stus Donetsk National University, Vinnytsia, 2024.
In the qualification work, a software product was designed and developed for the implementation of a system using artificial neural networks to detect anomalous behavior of data center users, based on the k-network technology.
The aim of the study. Develop software to detect anomalous behavior of data center users using appropriate artificial neural network technologies, and develop an application to demonstrate the operation of network learning visually.

Scientific novelty:

• The practical proposal for the introduction of k-card technology in the field of information protection in the form of an application has received further development.

• Developed an application for detecting abnormal user behavior and developed a false alarm mechanism for detecting attacks in the data center.

Keywords: neural networks, k-maps, machine learning, classifier systems, abnormal behavior detection systems, dataset, data pre-processing.
Total volume of work: 74 pages, 16 figures, 3 tables, 37 sources.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ACCS - австралійський центр кібербезпеки

AE - автоенкодер

BMU - найкращий відповідним блок

BP - алгоритм зворотного поширення

CNN - згорткові нейронні мережі

DAE - глибокий автокодер

DARPA - агентство перспективних оборонних досліджень США

DBM - глибока Машина Больцмана

DBN - мережі глибоких переконань

FSM - скінченний автомат

HMM - генеративні глибокі архітектур

IDS - система виявлення вторгнень

LSTM - блоки довгої короткочасної пам’яті

MHMM - приховані моделі Маркова

OCSVM - однокласова опорна векторна машина

PNN - ймовірнісна нейронна мережа

RBF - радіально-базисна функція Гауса

RBM - обмежена машина Больцмана

SVM - машини опорних векторів

WPF - Windows Presentation Foundation

U2R - атака User to Root

СВМА - системи виявлення мережевих аномалій

ШНМ - штучна нейронна мережа

ВСТУП

Системи виявлення мережевих аномалій (СВМА) відіграють важливу роль у кожній системі захисту мережі, оскільки вони виявляють та запобігають зловмисним діям користувачів у центрах обробки даних. Тому ця робота пропонує вичерпний огляд різних аспектів мережевих систем виявлення вторгнень на основі аномалій. Окрім того, обговорюються сучасні шкідливі дії у мережевих системах та важливі властивості систем виявлення вторгнень. У цій роботі пояснюються важливі етапи розробки систем для виявлення аномальної поведінки, такі як попередня обробка, виділення функцій і виявлення та розпізнавання зловмисної поведінки. Окрім того, що стосується фази виявлення та розпізнавання, були всебічно проаналізовані останні підходи до машинного навчання на основі теорії нейромереж, включаючи контрольовані, неконтрольовані, нові методи глибокого та ансамблевого навчання.

Зрештою, визначено потенційні виклики та деякі майбутні напрямки для систем виявлення аномальної поведінки користувачів на основі машинного навчання.

*Актуальність теми***.** Надійний мережевий зв’язок має вирішальне значення у повсякденному функціонуванні сучасного світу. Компанії, навчальні заклади, державні відомства та навіть людина сильно покладається на безперервне спілкування мережеве обслуговування та обчислювальні засоби для проведення повсякденних операцій. Спроби порушити надійний потік мережі та надійне мережеве з’єднання також значно збільшилися. Мережеві комп'ютерні систем все більше стали мішенню для зловмисників, незалежно від того, чи є вони просто хакерами, ворогами чи злочинцями. Багато із цих атак дорого коштують.

Об'єкт дослідження. Сучасна ІТ технологія нейроподібної мережі на основі *k-карт*.

Предмет дослідження. Застосування штучних нейронних мереж для виявлення аномальної поведінки користувачів центрів обробки даних.

Мета дослідження. Вдосконалити структуру нейронної мережі для виявлення аномальної поведінки користувачів центрів обробки даних на основі технології k-мережі та розробити програмне забезпечення для виявлення аномальної поведінки користувачів центрів обробки даних, використовуючи відповідні технології штучних нейронних мереж, а також розробити застосунок для демонстрації роботи навчання мережі наочно.

Завдання:

- Провести літературно-аналітичний огляд існуючих технологій застосування штучних нейронних мереж;
- Проаналізувати існуючі алгоритми застосування штучних нейронних мереж для виявлення аномальної поведінки користувачів центрів обробки даних;
- Розробити архітектуру системи виявлення атак на основі технології *k-карт*;
- Розробити та протестувати додаток на основі побудованої архітектури.

Наукова новизна:

- Дістала подальший розвиток практична пропозиція впровадження технології *k*-карт у сферу захисту інформації у вигляді додатку.
- Розроблений додаток для виявлення аномальної поведінки користувачів та розроблено механізм хибного спрацьовування при виявленні атак у центрі обробки даних.

Практичне значення отриманих результатів. На підставі проведеного дослідження розроблено додаток, який дозволяє виявляти аномальну поведінку користувачів центрів обробки даних.

РОЗДІЛ 1. АНАЛІЗ ШЛЯХІВ ТА МЕТОДІВ ЗАСТОСУВАННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ВИЯВЛЕННЯ АНОМАЛЬНОЇ ПОВЕДІНКИ КОРИСТУВАЧІВ У ЦЕНТРАХ ОБРОБКИ ДАНИХ

1.1 Огляд існуючих технологій нейронних мереж для виявлення аномальної поведінки користувачів у центрах обробки даних

Аномалії - це незвичайна поведінка, спричинена зловмисниками, які залишають сліди у комп'ютерному середовищі. Ці сліди виявляються, щоб ідентифікувати атаки [1].

Більшість сучасних систем використовують методи виявлення неправильного використання або виявлення аномалій. Однак обидва ці методи все ще стикаються із потенційними проблемами. Наприклад, методи виявлення зловживань не в змозі ідентифікувати анонімні вторгнення, тоді як, із іншого боку, підходи до виявлення аномалій зазвичай мають проблеми із високим рівнем помилкових спрацьовувань. Щоб вирішити ці проблеми, пропонуються комплексні підходи з потужним дизайном, що використовує функції багатьох методів виявлення аномалій, а гібридизація кількох моделей підвищує продуктивність систем. Методи *ансамблевого навчання* застосовують кілька методів машинного навчання у одній потужній та гнучкій моделі, щоб зменшити дисперсію (утворення в пакети), зміщення (підвищення) або покращити прогнози (укладання). Головною метою ансамблевого навчання є досягнення загальної точності у порівнянні із кожним класифікатором окремо. Структура ансамблевого навчання складається із пакетування, посилення та узагальнення стеку [2].

Існує багато методів класифікації, таких як дерева рішень, системи на основі правил, нейронні мережі, опорні векторні машини, метод Байєса та метод найближчого *k*-сусіда. Кожен метод використовує метод навчання для побудови моделі класифікації. Однак відповідний підхід до

класифікації має не лише обробляти навчальні дані, але й точно ідентифікувати клас записів, які він ніколи раніше не бачив. Створення моделей класифікації із надійною здатністю до узагальнення є важливим завданням алгоритму навчання мережі [1, 3].

1.1.1 Згорткові автокодери

Автокодери - це нейронні мережі, призначені для вивчення низьковимірного представлення за певних вхідних даних. Вони складаються із двох компонентів: кодера, який перетворює вхідні дані у низьковимірне представлення (так зване вузьке місце) та декодера, який перетворює це низьковимірне представлення назад у вхідні дані. Структуруючи проблему навчання таким чином, мережа кодувальника вивчає ефективну функцію «стиснення», яка перетворює вхідні дані у представлення нижчої розмірності, так що мережа декодера може успішно реконструювати вихідні дані у вхідні. Модель навчається шляхом мінімізації помилки реконструкції, яка є різницею (середня квадратична помилка) між входом та реконструйованим виходом, отриманим декодером. На практиці автокодери застосовувалися як техніка зменшення розмірності, а також у інших випадках, таких як видалення шумів із зображень, розфарбовування зображень, неконтрольоване вилучення функцій та стиснення даних [4].

Застосування автокодера для виявлення аномальної поведінки користувачів відповідає загальному принципу: спочатку моделюється нормальна поведінка, а потім генерується оцінка аномалії для кожного нового зразка даних. Щоб змоделювати нормальну поведінку, використовується напів-контрольований підхід, коли автокодер навчається на звичайних зразках даних. Таким чином, модель вивчає функцію відображення, яка успішно реконструює нормальні зразки даних із дуже

малою помилкою реконструкції. Така поведінка повторюється під час тестування, де помилка реконструкції мала для нормальних зразків даних й велика для аномальних зразків даних. Щоб ідентифікувати аномалії, використовується оцінка помилки реконструкції як оцінку аномалії та позначаються зразки із помилками реконструкції, що перевищують заданий поріг [2, 3-4].

Важливо відзначити, що функція відображення, отримана автокодером, є специфічною для розподілу навчальних даних. Тобто автокодер зазвичай не зможе реконструювати дані, які значно відрізняються від даних, які він бачив під час навчання. Ця властивість вивчення відображення, специфічного для розподілу (на відміну від загального лінійного відображення), особливо корисна для завдання виявлення аномалій [1].

На високому рівні моделі зазвичай складаються із кодера *E*, який генерує приховане подання вхідних маркерів та декодера *D*, який приймає та послідовно генерує набір вихідних маркерів. Традиційно кодер і декодер складаються із блоків довгої та короткочасної пам’яті (*LSTM*), які особливо підходять для моделювання часових зв’язків у маркерах вхідних даних.

Використовуючи *Convolutional Autoencoders*, отримується початкова інформація *Autoencoder* (*AE*). *AE* — це тип штучної нейронної мережі, яка ипопулярна для виявлення аномалій і складається із двох основних модулів: кодера та декодера (рис. 1.1). Кодер відображає вхідні дані у прихований вектор, а декодер намагається відновити вхідні дані із прихованого вектора [6].

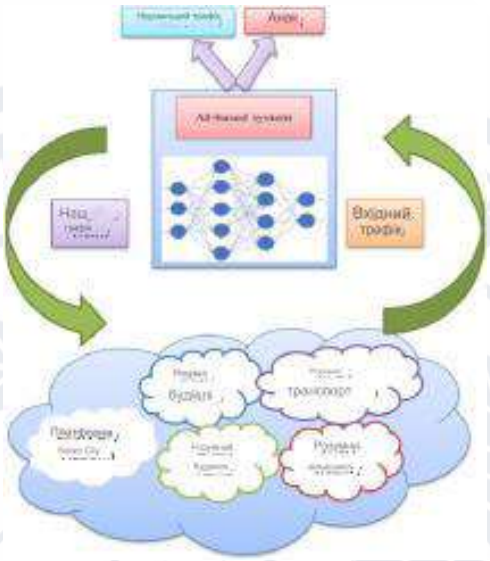

Рис. 1.1 - Система виявлення вторгнень на основі аномалій

Окрім того, можна вивести середнє значення та параметр дисперсії із декодера та обчислити ймовірність того, що нова точка даних належить до розподілу нормальних даних, на яких була навчена модель. Якщо точка даних знаходиться у області низької щільності (нижче деякого порогу), це визначається як аномалія. Це робиться тому, що моделюється розподіл, а не точкова оцінка [2].

Моделі машинного навчання, які вирішують завдання незалежно від даних, чутливі до упереджень та інших проблем [7]. Моделі виявлення аномалій пов’язані із конкретними ризиками та тактиками пом’якшення.

Щоб застосувати *OCSVM* для виявлення аномалій, необхідно навчити модель *OCSVM*, використовуючи нормальні дані або дані, що містять невелику частку аномальних зразків. У більшості реалізацій *OCSVM* модель повертає оцінку того, наскільки подібні дані до зразків даних, які спостерігаються під час навчання. Ця оцінка може бути відстанню від межі прийняття рішення (розділ гіперплощини) або значенням дискретного класу (+1 для подібних даних і -1 для даних, які не

є такими). Будь-який тип оцінки можна використовувати як оцінку аномалії [8].

Мустафа та ін. запропонував підхід до класифікації зловмисної поведінки із використанням методу корентропії-варіації. Автори вважають, що сучасні мережеві атаки можуть імітувати звичайну діяльність та дуже ускладнюють відстеження зловмисних спостережень у мережі. Вони розробили техніку мережевої експертизи для розслідування мережевих атак. На першому етапі дані мережевого трафіку були захоплені; згодом автори витягли важливі ознаки за допомогою статистики хі-квадрат. Нарешті, випадки зловмисників були виявлені за допомогою методу варіації корентропії. Запропонована авторами статистична методика, не була методикою, заснованою на навчанні, а також незастосовною до обчислювальних середовищ *Cloud* і *Fog* - це прогалини, які виявлено у цій роботі [3, 5].

Імовірнісні компоненти Байєса, введені в *VAE*, дають кілька корисних переваг. По-перше, *VAE* дозволяють робити висновок; тобто, тепер можна взяти вибірку із навченого розподілу кодувальника та декодувати зразки, які явно не існують у вихідному наборі даних, але належать до того самого розподілу даних. По-друге, *VAE* вивчають розділене подання даних; тобто окрема одиниця у латентному коді чутлива лише до одного генеративного фактора. Це дозволяє певну інтерпретацію виходу *VAE*, оскільки можна змінювати одиниці у прихованому коді для контрольованої генерації зразків. По-третє, *VAE* надає справжні ймовірнісні показники, які дають принциповий підхід до кількісної оцінки невизначеності при застосуванні на практиці (наприклад, ймовірність того, що нова точка даних належить до розподілу нормальних даних, становить 80%) [8, 9].

1.1.2 Сигнатурні системи виявлення вторгнень

Сигнатурні системи виявлення вторгнень - *SIDS* засновані на методах зіставлення шаблонів для пошуку відомої атаки; вони також відомі як виявлення на основі знань або виявлення неправильного використання (Khraisat et al., 2018). У *SIDS* методі шаблони використовуються для пошуку попереднього вторгнення. Іншими словами, коли сигнатура вторгнення збігається із сигнатурою попереднього вторгнення, яка вже існує у базі даних сигнатур, спрацьовує сигнал тривоги. Для *SIDS* журнали хосту перевіряються для того, щоб знайти послідовності команд або дій, які раніше були визначені як зловмисне ПЗ. В літературі ці системи також називають «виявленням на основі знань» або «виявленням неправильного використання» [5, 10].

1.1.3 Генеративні глибокі архітектури

HMM - це генеративні моделі, що використовуються для характеристики стохастичних процедур. *HMM* є відповідними методами для моделювання динамічної поведінки базових систем і ці моделі популярні у області розпізнавання образів. *HMM* часто застосовуються для побудови моделей часових рядів, а також успішно використовуються у різних областях мережевих систем виявлення аномальної поведінки. Мустафа та ін. запропонував схему аналізу загроз, засновану на суміші прихованих моделей Маркова (*MHMM*) [7].

Техніка *MHMM* використовується моделлю Гауса (*GMM*), коли кількість компонентів відома, а межі сприйнятих даних необмежені (тобто $(-\infty,+\infty)$. Недолік запропонованої системи – вона вимагає велику кількість звичайних та атакуючих випадків для точної оцінки параметрів *BMM* і *HMM* [5], окрім того, система також потребує нових функцій, які

дозволяють запустити алгоритм для налаштування ковзаючого вікна, яке потрібує реалізації.

Генеративна модель обчислює спільні розподіли ймовірностей із спостережуваних даних за допомогою своїх класів. Глибокий автокодер (*DAE*) в основному використовується для навчання ефективному кодуванню без вчителя і зазвичай складається із вхідного рівня, одного (або більше) прихованого(их) шару(ів) і вихідного рівня [10]. Результатом, досягнутим на вихідному рівні, є реконструкція вхідного рівня після того, як вхідні вузли були «відфільтровані» через менший прихований шар. Таким чином, *DAE* працює подібно до методів зменшення розмірності, таких як *PCA*. У разі виявлення аномалії у функціонуванні мережі, які витягуються через прихований прошарок, можуть бути використані для навчання прошарків прямого зв'язку. Загальне навчання мережі відбувається шляхом навчання кожного автокодувальника без учителя, після чого виконується етап тонкого налаштування, за допомогою якого останній рівень навчається контрольованими мережевими даними.

Muna et al. запропонував систему виявлення зловмисних дій у мережі на основі автоматичного кодувальника та нейронної мережі глибокого передавання. Необхідну інформацію та мережеві функції було зібрано із мережевих пакетів *TCP/IP*. Комбінація глибокої прямої нейронної мережі та автоматичного кодувальника створила надійний алгоритм навчання для роботи як із позначеними, так і із непозначеними мережевими функціями, разом із навчанням та тестуванням для процесу оцінки, який використовує два популярні набори даних про аномалії мережі - *NSL- KDD* і *UNSW-NB* 15 [5, 10].

Недоліком цього методу є складність вибору відповідних параметрів для фази навчання під час роботи із даними мережевого трафіку у реальному часі із високою швидкістю передачі. *Ludwing* та ін. запропонували систему виявлення вторгнень на основі автокодувальника

для класифікації різних типів атак. Атаки в наборі даних *NSL-KDD*, такі як *R2L*, *U2R*, *DoS* і *Probing*, були класифіковані із загальною точністю 0,92.

1.1.4 Обмежена машина Больцмана (*RBM*)

Обмежена машина Больцмана (*RBM*) є популярним методом глибокого навчання серед генеративних моделей, які складаються із двох основних архітектур: глибокої Машини Больцмана (*DBM*) і мережі глибоких навчань (*DBN*) [9, 11]. Методи *RBM* у основному застосовуються для зменшення прихованих прошарків у мережі та не використовують внутрішньорівневі зв'язки між прихованими нейронами. Щоб побудувати архітектуру *DBN*, слід навчити *DBM*, використовуючи дані без міток як вхідні дані наступного прошарку та об'єднуючи інші прошарки для розрізнення. *Alom* et al. розробив мережу глибоких навчань для інтерпретації спроб вторгнення у вхідний мережевий трафік. Автори відкрили можливості *DBN* для виявлення вторгнень за допомогою серії експериментів після навчання із набором даних *NSL-KDD*. Навчена *DBN* змогла виявити усі типи невідомих атак у наборі даних та класифікувати їх на п'ять різних категорій; однак у разі невідомих зловмисних атак за межами набору даних (*DoS, U2L R2L, Prode*), запропонована методика не зможе їх ідентифікувати. У таких ситуаціях може знадобитися визначення ненормальних класів, а також нових класів, для яких доступно дуже мало або зовсім немає позначених даних. У подібних випадках ідеальним рішенням є підхід до напівконтрольованої класифікації, який дозволяє виявити як відомі, так і раніше невидимі аномалії [12].

Мережа глибокого навчання (*DBN*) - це генеративна графічна модель, що складається із кількох прихованих прошарків, зі зв'язками між шарами, але не між одиницями усередині кожного шару. Цей тип архітектури глибокого навчання є гібридною моделлю контрольованих та

неконтрольованих навчальних мереж. Неконтрольований розділ був навчений на основі одного пошарового з'єднання за раз, у якому прошарки діють як детектори ознак та виконують очікувану класифікацію, тоді як контрольований розділ - це один або більше прошарків, зв'язаних для класифікації.

1.1.5 Набори даних сценаріїв атак

Набір даних *TUIDS* - це реальний набір даних, створений в університеті Тезпур, Індія. Набір даних містить різні типи сценаріїв атак, а пакети мережевого трафіку збиралися за допомогою таких інструментів, як *nfdump* і *gulp*, щоб отримати репрезентативні характеристики. На основі попередньо оброблених даних та відповідних міток функції класифікуються на базові, часові, віконні, контентні та без підключення.

Набір даних *UNSW-NB15* був створений у 2015 році в *Cyber Range Lab* Австралійського центру кібербезпеки (*ACCS*) в Університеті Нового Південного Уельсу. *UNSW-NB15* було зібрано за допомогою інструменту *IXIA PerfectStorm* та складається із гібриду звичайних та синтетичних сучасних спостережень атак у формі численних шаблонів зі звичайними доказами та дев'ятьма групами атак. *Backdoors, DoS, Analysis, Fuzzers, Generic, Worms, Shellcode, Reconnaissance* та *Exploits* - це типи атак, які змістовно характеризуються 47 характеристиками для кожної атаки. При ближчому розгляді набору даних видно, що 2540044 номери записів (100 ГБ) необроблених спостережень мережевого трафіку були зібрані за допомогою різних пристроїв. Цей набір даних складається з 700000 зразків, включаючи 677789 нормальних і 22211 зловмисних дій відповідно. У наборі даних *UNSW-NB15* розробники трафіку *IXIA* були підключені до трьох різних серверів: сервери 1 і 3 були призначені для створення звичайних екземплярів, а сервер 2 — для шкідливих екземплярів, тоді як усі сервери

підключені до двох маршрутизаторів, а маршрутизатор 1 є основним маршрутизатором. За допомогою маршрутизатора 1 усі файли захоплюються для вилучення векторів функцій [13].

1.1.6 Машини опорних векторів

Машини опорних векторів (*SVM*): *SVM* - це класифікатор, визначений гіперплощиною розщеплення. *SVM* використовують функцію ядра для відображення навчальних даних у просторі із більшою розмірністю для того, щоб вторгнення класифікувалося лінійно. *SVM* добре відомі своєю здатністю до узагальнення і в основному цінні, коли кількість атрибутів велика, а кількість точок даних невелика. Різні типи роздільних гіперплощин можуть бути досягнуті шляхом застосування ядра, наприклад лінійного, поліноміального, радіально-базисної функції Гауса (*RBF*) або гіперболічного тангенса. У наборах даних багато функцій є зайвими або менш впливовими на розділення точок даних на класи. Тому вибір функцій слід враховувати під час навчання *SVM*. *SVM* також можна використовувати для класифікації в кілька кластерів. У роботі Лі та ін. класифікатор *SVM* із ядром *RBF* було застосовано для класифікації набору даних *KDD* 1999 у попередньо визначені класи (Li та ін., 2012). Із загалом 41 атрибута підмножина функцій була ретельно відібрана за допомогою методу вибору ознак [10].

1.1.7 Згорткові нейронні мережі

Згорткові нейронні мережі (*CNN*) досягли значних результатів у різноманітних областях, включаючи медичні дослідження. Хоча глибоке навчання стало домінуючим методом у різноманітних складних завданнях, таких як класифікація зображень та виявлення об’єктів, це не є панацея.

Знайомство із ключовими концепціями та перевагами *CNN*, а також обмеженнями глибокого навчання має важливе значення для того, щоб використовувати його у радіологічних дослідженнях із метою покращення роботи рентгенолога та, зрештою, догляду за пацієнтами. Рівень згортки є фундаментальним компонентом архітектури *CNN*, який виконує виділення ознак, що зазвичай складається з комбінації лінійних і нелінійних операцій, тобто операції згортки та функції активації [3-5, 10].

1.1.8 Скінченний автомат

Скінченний автомат (*FSM*) - це обчислювальна модель, яка використовується для представлення та керування потоком виконання. Цю модель можна застосувати для виявлення аномальної поведінки при створенні моделі системи виявлення вторгнень. Як правило, модель представлена у вигляді станів, переходів та дій. Наприклад, будь-які варіації вхідних даних відзначаються і на основі виявленої варіації і таким чином відбувається перехід (*Walkinshaw et al*., 2016). *FSM* може представляти законну поведінку системи і будь-яке спостережуване відхилення від неї *FSM* розцінюється як атака [10].

1.1.9 Інші методи виявлення сценаріїв аномальної поведінки користувачів

Ігіно та ін. надав опитування про змагальні атаки проти *IDS* у критично важливих для безпеки середовищах. Автори надали загальну таксономію тактики атаки на *IDS* та розділили стратегію виявлення на три різні фази:

-вимірювання;

- класифікація;

-відповіді.

Основною проблемою механізму виявлення у методах *IDS* є впровадження методів, які не ґрунтуються на навчанні, для боротьби зі складністю сучасних вторгнень та шкідливих примірників [5, 10].

Ходо та ін. обговорили глибокі мережі для *IDS*, автори надали огляд загальної класифікації *IDS* та таксономії з останніх робіт. Порівнюючи різні методи, засновані на навчанні, вони підтвердили, що згорточна нейронна мережа (*CNN*) не використовувалася у області виявлення вторгнень, однак вони довели, що це хороший класифікатор. Окрім того, комерційно використовуються методи, засновані на сигнатурах, однак основним недоліком цих методів є те, що вони не можуть виявити усі типи зловмисних атак через відсутність списку сигнатур у базі даних [10].

Абуромман та ін. провели опитування щодо *IDS* із використанням ансамблевих та гібридних методів навчання. Автори виділили дві основні категорії методів із декількома класифікаторами:

- однорідні ансамблі (єдиний класифікаційний підхід);
- гетерогенні ансамблі (два або більше різних класифікаційних підходів).

Вони довели, що неоднорідні підходи, засновані на голосуванні зваженої більшості, рідко реалізуються для *IDS*, а методи метаевристичної оптимізації заслуговують на більшу увагу на основі витягнутих шаблонів у наборі даних *NSL-KDD* [7, 13].

Ахмед та ін. надано огляд методів виявлення аномалій у фінансовій сфері, опитування в основному зосереджено на кластеризації як методі неконтрольованого навчання для виявлення шахрайства та аномальної поведінки порівняно зі звичайними. Різні типи фінансових шахрайств, такі як:

- шахрайство зі зломом;
- шахрайство із виставленням рахунків;
- шахрайство із незаконним перерозподілом;

- невдала авторизація.

Також питання дефіциту реальних даних обговорювалися у даній статті.

Бучак та ін. запропонував огляд методів інтелектуального аналізу даних та машинного навчання для *IDS* кібербезпеки. Методи неправильного використання та виявлення аномалій обговорювалися на основі таких важливих критеріїв:

- точність;

- складність;

- час для класифікації невідомого екземпляра за допомогою навченої моделі;

- зрозумілість остаточного рішення [5].

Найбільшою прогалиною, яку помітили автори, була наявність мічених даних, що є дуже важливою проблемою, коли фаза виявлення та розпізнавання аномальної поведінки базується на методах навчання з учителем.

Ахмед та ін. надав опитування щодо методів виявлення мережевих аномалій, яке було зосереджено на кількох категоріях методів виявлення, включаючи декілька методів класифікації, статистичні методи із методами, що не ґрунтуються на навчанні, та декілька підходів до кластеризації. Окрім того, також обговорювалося кілька наборів даних, які використовуються для виявлення мережевих аномалій. Недоліком у цій оглядовій статті було те, що не надано подробиць про етапи попередньої обробки та вилучення функцій, які дуже важливі у *NADS* [7].

1.2 Моделі штучних нейронних мереж та алгоритми їх навчання

Ідея дизайну штучних нейронних мереж (ШНМ) полягає у імітації роботи людського мозку. ШНМ містить вхідний рівень, кілька прихованих шарів та вихідний рівень. Блоки в суміжних шарах повністю з'єднані. ШНМ містить величезну кількість одиниць та теоретично може апроксимувати довільні функції; отже, він має сильну здатність підгонки, особливо для нелінійних функцій. Через складну структуру моделі навчання ШНМ займають багато часу. Слід зазначити, що моделі ШНМ навчаються за допомогою алгоритму зворотного поширення, який не можна використовувати для навчання глибоких мереж. Таким чином, ШНМ належить до поверхневих моделей та відрізняється від моделей глибокого навчання [12, 14].

Доступні дані зазвичай поділяються на 3 набори:

- навчальний;
- перевіряльний;
- тестовий.

Навчальний набір використовується для навчання нейронної мережі, де значення втрат обчислюються за допомогою прямого розповсюдження, а параметри, що вивчаються, оновлюються за допомогою зворотного розповсюдження. Набір перевірки використовується для моніторингу продуктивності моделі під час процесу навчання, точного налаштування параметрів й виконання вибору моделі. В ідеалі тестовий набір використовується лише один раз у самому кінці проекту, щоб оцінити продуктивність остаточної моделі, яка налаштована та обрана у процесі навчання за допомогою наборів для навчання та перевірки.

1.2.1 Глибоке навчання

У даний час глибоке навчання використовується у поширених технологіях, таких як автоматичні системи розпізнавання обличчя, цифрові помічники та виявлення шахрайства. Глибоке навчання також використовується у нових технологіях [12].

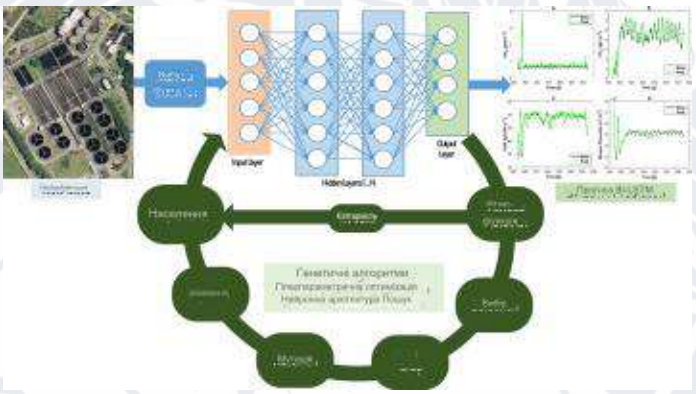

Рис. 1.2 - Оптимізація моделей глибокого навчання для прогнозування продуктивності

1.2.2 Вибір ознак

Вибір ознак або вибір змінних - це ефективна техніка, яка використовується для визначення найбільш значущих змінних, зменшення розмірів набору даних та підвищення ефективності алгоритмів машинного навчання. Зазвичай досліджені змінні зі значенням коефіцієнта кореляції розраховуються за формулою (1.1) [12]:

$$P - value < 0.,2 \tag{1.1}$$

Такі значення коефіцієнта були визначені як ефективні фактори ризику для прогнозування аномальної поведінки та були включені у моделі *ANN* [11, 12].

Регулярна перевірка для виявлення переналаштування полягає у моніторингу втрат на наборах для навчання та перевірки під час ітерації навчання [11]. Якщо модель добре працює на навчальному наборі порівняно із набором перевірки, це означає, що модель перевиконано із навчальними даними. Якщо модель погано працює як на наборах для навчання, так і на перевірці, це означає, що модель є неефективною. Хоча чим довше навчається мережа, тим краще вона працює на навчальному наборі, у певний момент мережа надто добре підходить до навчальних даних та втрачає здатність до узагальнення. Це називається перенавчанням мережі.

За останні 10 років було розроблено низку різних систем: *word2vec*, *GloVe*, *BERT*, *GPT*, тощо. Кожна з цих систем базується на різних нейронних мережах. Але зрештою усі вони беруть слова та характеризують їх списками із сотень до тисяч чисел.

1.2.3 Персептрон, як перша нейромережа

Персептрон, як перша нейромережа, був заснований на концепції простого обчислювального блоку, який приймає один або кілька вхідних даних та виробляє єдиний вихід, змодельований за структурою та функцією нейрона в мозку. Персептрон був розроблений, щоб мати можливість навчатися на прикладах та коригувати свої параметри для підвищення точності класифікації нових прикладів. Типову структуру штучних нейронних мереж представлено на рис. 1.3 [11]:

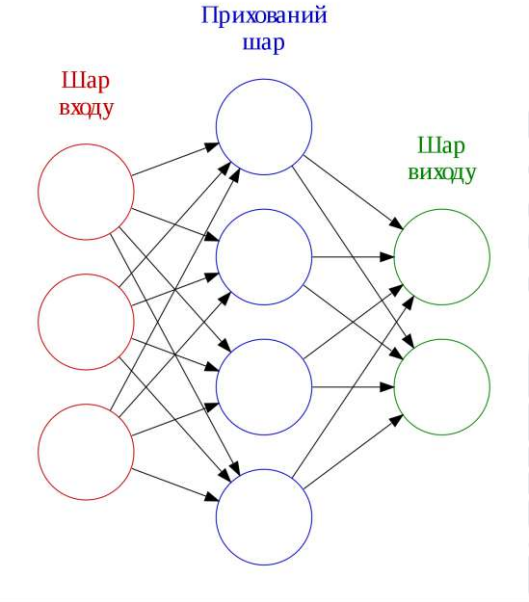

Рис. 1.3 - Типова структура штучних нейронних мереж

Типова нейромережа складається з наступних шарів:

- вхідний шар;
- прихований шар;
- вихідний шар.

Правило навчання *Perceptron* сходиться, якщо два класи можна розділити лінійною гіперплощиною. Однак, якщо класи не можуть бути ідеально розділені за допомогою лінійного класифікатора, це може призвести до помилок [11].

Нейронна мережа *RBF* є прямою нейронною мережею, яка використовує радіальні базисні функції як функції активації. Мережі *RBF* складаються із кількох рівнів, включаючи вхідний рівень, один або більше прихованих рівнів із функціями активації радіального базису та вихідний рівень. Мережі *RBF* відрізняються розпізнаванням образів, апроксимацією функцій та прогнозуванням часових рядів. Однак проблеми у навчанні мереж *RBF* включають вибір відповідних базових функцій, визначення кількості базових функцій і обробку переобладнання [10, 11].

1.2.4 Процес навчання мережі

Машинне навчання починається із даних - чисел, фотографій або тексту, наприклад банківських транзакцій, фотографій людей або навіть хлібобулочних виробів, записів про ремонт, даних часових рядів із датчиків або звітів про продажі. Дані збираються та готуються для використання як навчальні дані або інформація, на основі якої навчатиметься модель машинного навчання [12, 14].

Під час фази навчання алгоритму штучна нейронна мережа (ШНМ) вчиться розпізнавати шаблони із вхідних даних [13]. Потім йде порівняння отриманого результату із бажаним результатом. Різниця між двома результатами коригується за допомогою зворотного робочого процесу, поки така різниця не стане нижчою за попередньо визначений критерій. Тому для навчання нейронної мережі дуже важливим є вибір відповідного алгоритму навчання.

Алгоритми навчання є базовими механізмами для побудови моделей нейронної мережі з метою навчання функцій або шаблонів із вхідних даних, щоб можна було знайти набір внутрішніх параметрів моделі для оптимізації точності моделі. Існує багато типів алгоритмів навчання, але найчастіше використовувані можна перерахувати як градієнтний спуск, сполучений градієнт, метод квазіНьютона та алгоритми Левенберга-Марквардта [10, 11, 12-14].

Є кілька ключових частин. По-перше, це питання про те, яку архітектуру нейронної мережі слід використовувати для конкретного завдання. Окрім того, виникає критичне питання про те, як отримати дані для навчання нейронної мережі. І все частіше ніхто не займається навчанням мережі із нуля: натомість нова мережа може або безпосередньо

включати іншу вже навчену мережу, або, принаймні, може використовувати цю мережу для створення додаткових прикладів для навчання.

У традиційному машинному навчанні процес навчання контролюється і оператор повинен бути конкретним, повідомляючи комп’ютеру, які типи речей він повинен шукати для вирішення задачі: чи містить зображення необхідний об'єкт, чи ні. Це трудомісткий процес, який називається виділенням функцій і рівень успіху комп’ютера повністю залежить від здатності оператора точно визначити набір функцій для об'єкту. Перевагою глибокого навчання є те, що програма створює набір функцій сама без нагляду [14].

Більшість глибоких нейронних мереж є прямими, тобто вони рухаються лише е одному напрямку, від входу до виходу. Однак також можна навчити модель за допомогою зворотного поширення; тобто рухатися у протилежному напрямку від виходу до входу. Зворотне розповсюдження дозволяє обчислити та приписати помилку, пов’язану із кожним нейроном, дозволяючи нам належним чином налаштувати та підігнати параметри моделі(й) [12].

Щоб досягти прийнятного рівня точності, програми глибокого навчання вимагають доступу до величезних обсягів навчальних даних та обчислювальної потужності, жоден з яких не був легкодоступним для програмістів до ери великих даних та хмарних обчислень. Оскільки програмування глибокого навчання може створювати складні статистичні моделі безпосередньо із власного ітераційного результату, воно здатне створювати точні прогнозні моделі із великої кількості немаркованих неструктурованих даних [10, 12].

Переналаштування ваги коефіцієнтів стосується ситуації, коли модель вивчає статистичні закономірності, характерні для навчального набору, тобто запам’ятовує невідповідний шум замість того, щоб вивчати сигнал, і, отже, працює гірше на наступному новому наборі даних. Це одна

із головних проблем у машинному навчанні, оскільки переналаштовану модель неможливо узагальнити на дані, які ніколи раніше не бачили. У цьому сенсі набір даних відіграє ключову роль у належній оцінці продуктивності моделей машинного навчання. Звичайна перевірка для виявлення перепідгонки даних навчання полягає у моніторингу втрати та точності на наборах навчання та перевірки. Якщо модель добре працює на навчальному наборі порівняно із набором перевірки, то модель, ймовірно, перевиконана із навчальними даними [14].

Найкращим рішенням для зменшення переобладнання є отримання більшої кількості навчальних даних [15]. Модель, навчена на більшому наборі даних, зазвичай краще узагальнює, хоча цього не завжди можна досягти у візуалізації. Інші рішення включають регулярізацію із випаданням або спадом ваги коефіцієнта, нормалізацію партії та збільшення даних, а також зменшення складності архітектури.

Вилучення - це техніка регулярізації, де випадково вибрані активації встановлюються на 0 під час навчання, щоб модель стала менш чутливою до певних ваг у мережі. Затухання ваги, яке також називають регуляризацією *L2*, зменшує переналаштування, зменшуючи вагові коефіцієнти моделі, щоб ваги приймали лише малі значення. Пакетна нормалізація - тип додаткового прошарку, який адаптивно нормалізує вхідні значення наступного прошарку, зменшуючи ризик переналаштування, а також покращуючи градієнтний потік через мережу, забезпечуючи вищу швидкість навчання та зменшуючи залежність від ініціалізації. Збільшення даних також є ефективним для зменшення переобладнання, яке є процесом модифікації навчальних даних за допомогою випадкових перетворень, таких як гортання, переклад, обрізання, обертання та випадкове стирання, так що модель не бачитиме абсолютно однакових вхідних даних під час навчальні ітерації [16].

Незважаючи на ці зусилля, все ще існує занепокоєння щодо переобладнання набору перевірки, а не навчального набору через витік інформації під час тонкого налаштування параметрів та процесу вибору моделі. Таким чином, звіт про продуктивність кінцевої моделі в окремому тестовому наборі, а в ідеалі, у зовнішніх наборах даних перевірки, якщо це можливо, має вирішальне значення для перевірки узагальнення моделі [10, 16].

Зі зростаючим повсюдним поширенням машинного навчання кожен бізнесмен, ймовірно, зіткнеться з ним і потребуватиме певних практичних знань у цій галузі. Опитування *Deloitte* у 2020 році показало, що 67% компаній використовують машинне навчання, а 97% використовують або планують використовувати його в наступному році [17].

Але машинне навчання також має недоліки. По-перше, це може бути дорого. Проектами машинного навчання зазвичай керують спеціалісти з обробки даних, які отримують високу зарплату. Ці проекти також потребують програмної інфраструктури, яка може бути дорогою. І бізнес може зіткнутися з багатьма проблемами [14].

1.3 Переваги та недоліки систем виявлення аномальної поведінки користувачів на основі штучних нейронних мереж

Більшість методів не у змозі ідентифікувати невідомі атаки, доки не будуть подані схожі навчальні дані. У випадку підходу до кластеризації, наприклад *k*-середнє, якщо визначено значення *k*, тоді процес, що залишився, простий. Методи кластеризації корисні для створення швидкої відповіді.

У системах виявлення аномалій, заснованих на кластеризації, початковим припущенням є призначення великого кластера для звичайних екземплярів, а менші кластери - для шкідливих екземплярів. За відсутності

цього припущення важко оцінити техніку. У випадку великої навчальної вибірки даних краще розділити її на подібні класи, щоб ефективно виявляти аномальну поведінку користувача, оскільки це зменшує обчислювальну складність. Використання невідповідних заходів близькості часто знижує рівень виявлення. Методи забезпечують надійну продуктивність порівняно із контрольованими чи статистичними підходами. Зазвичай динамічне оновлення профілів займає багато часу. Методи можуть легко ідентифікувати викиди у невеликих наборах даних. Неконтрольовані методи часто використовують як кластеризацію, так і виявлення викидів. Це створює більшу складність порівняно із іншими методами. Методи можуть виявляти бурхливі та ізольовані атаки. Параметри виявлення значною мірою залежать від цих методів [8, 10, 12].

1.3.1 Дерева рішень

Дерево рішень складається з трьох основних компонентів:
- вузол рішень;
- листок.

Перший компонент — вузол рішення, який використовується для ідентифікації тестового атрибута. Друга – це гілка, де кожна гілка представляє можливе рішення на основі значення тестового атрибута. Третій — листок, який містить клас, до якого належить примірник. Існує багато різних алгоритмів дерев рішень, включаючи ID3 (рис. 1.4) (Квінлан, 1986), C4.5 (Квінлан, 2014) і CART (Брейман, 1996) [11].

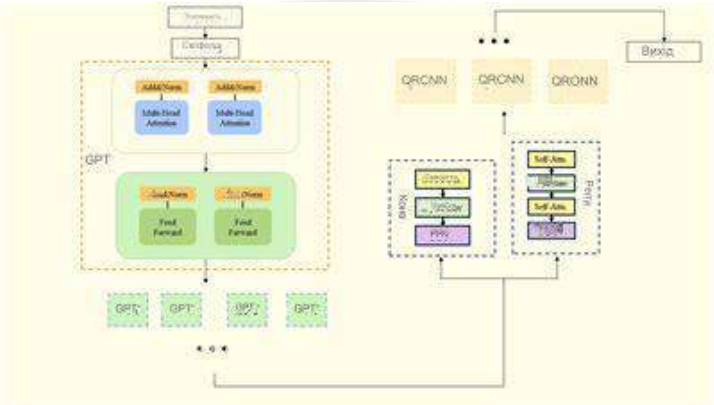

Рис. 1.4 - Дерево рішень

1.3.2 Інформація пакетів даних

Статистична інформація про сеанс включає поля в заголовках пакетів, кількість пакетів, частку пакетів, що надходять з різних напрямків, тощо. Ця статистична інформація використовується для складання векторів ознак, придатних для неглибоких моделей. Сеанси мають семантику високого рівня; таким чином, вони легко описуються правилами. Доречними методами можуть бути дерево рішень або моделі на основі правил. На жаль, методи, засновані на статистичних ознаках, ігнорують інформацію про послідовність і вони мають труднощі із виявленням вторгнень, пов’язаних із вмістом зв’язку [10, 17].

На початку *Staudemeyer* пропонує розглянути характеристики часових рядів відомої аномальної поведінки та мережевого трафіку, що може підвищити точність роботи алгоритмів виявлення атак. Щоб підтвердити це, вони впроваджують LSTM для виявлення вторгнень на основі чудової властивості LSTM моделювати довготривалі залежні відносини. Вони проектують мережу із чотирьох блоків пам'яті, кожен із яких містить дві комірки. Мережа здатна підтримувати баланс між обчислювальними витратами та продуктивністю виявлення. Їх

експериментальні результати вказують на те, що запропонована модель *LSTM* є кращою, ніж раніше опубліковані методи, оскільки *LSTM* може навчитися відстежувати та корелювати безперервні записи з'єднань у змінний час [16, 18].

Було проведено багато випробувань використання *DBN* для виявлення аномальної поведінки користувачів. Проте все ще існує багато невирішених проблем, таких як надлишкова інформація, яку легко вловити в локальний максимум. Щоб вирішити ці проблеми, *Zhao et al*. пропонують виявляти атаки вторгнення за допомогою міцності *DBN* та ймовірнісної нейронної мережі (*PNN*). По-перше, вони перемасштабують початкові вхідні дані до низьковимірних, використовуючи можливість нелінійного опису *DBN*. При цьому *DBN* може підтримувати основні характеристики вихідних даних у представленні. По-друге, алгоритм оптимізації частинок використовується для зменшення розміру прихованих вузлів кожного шару. По-третє, PNN вводиться для класифікації інформації низької розмірності. Їхні експерименти із набором даних *KDDCup* 99 показують, що вони певною мірою вирішили цю проблему.

У *IDS PCA* використовується як метод зменшення розмірності та виявлення. *Elrawy* та ін. використовував підхід *PCA* для створення статистичного IDS і аналізу даних на основі аномалій, який залежить від поділу основних компонентів на найбільш та найменш значущі основні компоненти. У цій системі етап виявлення базується на балах головного основного компонента та балу другорядного головного компонента. Окрім того, *PCA* використовувався у методах виявлення вторгнень на основі моделювання корисного навантаження, статистичного моделювання, аналізу даних машинного навчання [10, 12, 14] .

Поведінка вторгнення може залишати сліди системних викликів та аналіз цих системних викликів за допомогою алгоритмів класифікації може виявити вторгнення. Тран та ін. запропонував метод *CNN* для аналізу

системних викликів. Кожна основна операція, яка включає в себе операційну систему, використовуватиме системні виклики; Таким чином, аналіз шляху системного виклику може відтворити повний процес аномальної поведінки. Вони провели експерименти із наборами даних *NGIDS-DS* і *ADFA-LD*, які включають серію системних викликів. Спочатку вони витягли функції за допомогою розсувного вікна. Потім вони застосували модель *CNN* для виконання класифікації. *CNN* добре знайшов локальні зв'язки та виявляв аномальну поведінку системних викликів [17].

1.3.3 Техніка на основі інтелектуального аналізу даних

Процес виявлення значущих шаблонів та правил із великого набору даних відомий як інтелектуальний аналіз даних. Він використовується для розробки моделей на основі машинного навчання та програм штучного інтелекту, таких як пошукова система. Він також має застосування у сферах виявлення аномальної поведінки, аналізу настроїв, фільтрації спаму, аналізу якісних даних тощо. Застосування інтелектуального аналізу даних у виявленні аномалій включає виявлення дивних або незвичайних подій і включає етапи навчання, кластеризації, класифікації та регресія [16, 19].

Creech та ін. запропонував методологію із застосуванням розривних шаблонів системних викликів із метою підвищення рівня виявлення, одночасно зменшуючи рівень помилкової тривоги (*Creech*, 2014). Основна ідея полягає у використанні семантичної структури для системних викликів на рівні ядра, щоб зрозуміти аномальну поведінку [19].

1.3.4 Нечітка логіка

Ця техніка базується на ступенях невизначеності, а не на типовій істинній чи хибній булевій логіці, на основі якої створено сучасні ПК.

Таким чином, це простий спосіб прийти до остаточного висновку на основі нечітких, неоднозначних, неточних або відсутніх вхідних даних. Нечітка логіка дозволяє екземпляру належати, можливо, частково, до кількох класів одночасно. Таким чином, нечітка логіка є хорошим класифікатором для проблем системи, оскільки сама безпека включає невизначеність, а межа між нормальним та ненормальним станами не ідентифікована. Окрім того, проблема виявлення аномальної поведінки користувачів містить різні числові характеристики у зібраних даних та кілька похідних статистичних показників. Побудова системи на основі числових даних із жорсткими пороговими значеннями створює велику кількість помилкових тривог. Діяльність, яка лише незначно відхиляється від моделі, не може бути розпізнана, або незначна зміна нормальної активності може викликати помилкові тривоги. За допомогою нечіткої логіки можна змоделювати цю незначну аномалію, щоб підтримувати низькі помилкові показники. Elhag та ін. показали, що за допомогою нечіткої логіки частота помилкових тривог при визначенні вторгнення може бути зменшена. Вони окреслили групу нечітких правил для опису нормальних та ненормальних дій у комп’ютерній системі та механізм нечіткого висновку для визначення вторгнень (*Elhag et al*., 2015) [10, 12].

1.3.5 Алгоритм Наївного Байєса

Коли гіпотеза про незалежність атрибутів задовольняється, алгоритм Наївного Байєса досягає оптимального результату. На жаль, цю гіпотезу важко задовольнити насправді; отже, наївний алгоритм Байєса погано працює з даними, пов’язаними з атрибутами [14].

Ці алгоритми зменшують кількість операцій зіставлення, які необхідно виконати. Система залежить від підходу зіставлення шаблонів на основі виявлення сигнатур. Перевагою цієї системи є те, що її можна

застосовувати до розумних об’єктів із обмеженим обсягом пам’яті та часом автономної роботи. Основними особливостями цієї системи є зменшений обсяг пам’яті, необхідний для операцій зіставлення, зменшене навантаження на обробку смарт-об’єктів, підвищена швидкість обробки та її масштабована продуктивність для великої кількості шаблонів [10, 12-14].

РОЗДІЛ 2. ЗАСТОСУВАННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ ДЛЯ ВИЯВЛЕННЯ АНОМАЛЬНОЇ ПОВЕДІНКИ КОРИСТУВАЧІВ ДАТА-ЦЕНТРУ

2.1 Аналіз можливості використання штучних нейронних мереж для виявлення аномальної поведінки користувачів ЦОД

Виявлення аномальної поведінки в основному залежить від контексту. У міру зміни контексту природа об’єкта змінюється. Отже, повинна існувати система, яка вивчає різні типи об’єктів у різних сценаріях, а потім приймає рішення про аномалію об’єкта [1, 19].

Штучна нейронна мережа є одним із найпоширеніших методів машинного навчання, який успішно виявляє різні аномальні явища. Найпоширенішим методом навчання під наглядом є алгоритм зворотного поширення (ВР). Алгоритм *BP* оцінює градієнт помилки мережі щодо її модифікованих ваг. Однак для *IDS* на основі *ANN* точність виявлення, особливо для менш частих атак і точність виявлення все ще потребують покращення. Навчальна вибірка даних для менш частих атак невеликий порівняно із більш частими атаками і це ускладнює для ШНМ правильне вивчення властивостей цих атак. Як наслідок, точність виявлення нижча для менш частих атак. У сфері інформаційної безпеки може бути завдано величезної шкоди, якщо низькочастотні атаки не будуть виявлені. Наприклад, якщо атаки *User to Root* (*U2R*) не будуть виявлені, кіберзлочинець може отримати права авторизації користувача root та таким чином здійснювати зловмисні дії на комп’ютерних системах жертви. Окрім того, менш поширені атаки часто виходять за межі (*Wang et al.*, 2010) [10, 12].

ШНМ часто страждають від локальних мінімумів, тому навчання може зайняти дуже багато часу. Сильна сторона ШНМ полягає у тому, що

із одним або декількома прихованими шарами вона здатна створювати дуже нелінійні моделі, які фіксують складні зв'язки між вхідними атрибутами та мітками класифікації. Із розвитком багатьох варіантів, таких як рекурентні та згорткові *NN, ANN* є потужними інструментами для багатьох завдань класифікації, включаючи *IDS* [11].

Інші дослідження зосереджені на методах класифікації із використанням виділених ознак для розуміння динаміки мережі. Для безперервної організації на поведінку мережі впливають як середовище взаємодії, так і організаційний механізм. Зокрема, *Vigliotti* і *Hankin* класифікують поведінку користувачів у часових наборах даних як нормальну та аномальну. Ці автори спочатку позначають певних індивідів як ненормальні, а потім представляють підхід до висновку про підмножину індивідів, які можуть мати однакову якісну конотацію. Рагаван та ін. кількісно класифікувати групову динаміку як активну та неактивну та використовувати приховану модель Маркова (HMM) для відстеження та прогнозування стану груп [20].

Результати нещодавнього дослідження Агентства перспективних оборонних досліджень США (*DARPA*) підкреслюють сильні та слабкі сторони сучасних дослідницьких підходів до виявлення аномальної поведінки. Наукове дослідження *DARPA* є першим у своєму роді, яке забезпечує незалежну третю сторону оцінку інструментів виявлення вторгнень на основі такого великого масиву даних. Результати цього дослідження вказують на те, що необхідна фундаментальна зміна парадигми досліджень виявлення вторгнень, щоб забезпечити розумні рівні виявлення нових атак і навіть варіацій відомих атак. Центральним для цієї мети є здатність узагальнювати поведінку, що спостерігалася раніше, щоб розпізнавати майбутню непомітну, але подібну поведінку [19, 22].

Після розробки відповідного методу кодування необхідно застосувати відповідну топологію мережі. Нам потрібно було визначити, скільки вхідних і вихідних вузлів необхідно, і якщо буде використовуватися прихований шар, скільки вузлів він повинен містити. Оскільки ми прагнемо визначити, чи є вхідний рядок аномальним чи нормальним, ми використовуємо єдиний вихідний вузол безперервного значення, щоб показати ступінь, до якого мережа вважає вхідні дані нормальними чи аномальними. Чим більш аномальним є вхідний сигнал, тим ближче до 1 мережа обчислює свій вихід. І навпаки, чим ближче до нормального вхід, тим ближче до 0 обчислює вихідний вузол [21].

2.2 Обґрунтування вибору технології на основі штучних нейронних мереж для виявлення аномальної поведінки користувачів

Значна кількість досліджень виконана у області виявлення аномальної поведінки користувачів у центрах обробки даних за останні 20 років. З точки зору архітектури, система виявлення вторгнень (IDS) може бути одним із трьох основних типів [24]:

- *IDS* на основі мережі;

- *IDS* на основі хосту;

- гібридна *IDS*.

IDS на основі мережі моніторить *IP*-пакети, зазвичай заголовки пакетів, що передаються по мережі. *IDS* на основі хосту використовує дані аудиту хост машин. Гібридна *IDS* застосовує обидва вищезазначені методи для виявлення вторгнень як ззовні, так і зсередини. Алгоритмічно, це два різні підходи, які зазвичай використовуються для виявлення вторгнень. Перший підхід, широко відомий як неправильне використання виявлення, це підхід на основі правил, який використовує збережені підписи відомі

випадки вторгнення для виявлення атаки. Цей підхід дуже успішно виявляє відомі раніше атаки. Однак він не може виявити нові типи та варіанти атак, сигнатури яких не зберігаються. Коли відбуваються нові атаки, базу даних підписів потрібно змінювати вручну для майбутнього використання. Другий підхід загальновідомий як підхід до виявлення аномалії. У такому підході зазвичай спочатку встановлюється профіль нормальної поведінки. Потім девіанти від нормального профілю розглядаються як аномалії. У деяких випадках ці аномалії можуть бути звичайними, демонструючи певну поведінку, що відповідає невидимому режиму роботи. У таких випадках аномалії можуть показувати помилкові спрацьовування. Тобто класифікація нормальної поведінки як аномальної. Один із використаних методів для виявлення аномалій - побудова статистичних моделей із використанням метрик, отримані зі спостереження за діями користувача. Однією із метрик, що використовується, є лічильник подій, який представляє кількість подій за встановлений інтервал часу, наприклад, кількість підключень до різних пунктів призначення із тієї самої вихідної IP-адреси через годину. Ще одна метрика зазвичай використовується інтервал часу між двома корельованими подіями. Третім показником, який зазвичай використовується, є використання ресурсів, наприклад, споживання часу ЦП, кількість відкритих файлів тощо [25].

Схеми виявлення аномалій також використовують методи аналізу даних, наприклад, кластеризація, опорні векторні машини тощо, моделі нейронних мереж. Деякі із методів виявлення аномальної поведінки користувачів здатні виявляти новий вид атак, оскільки ці методи здебільшого керуються даними, що не залежать від раніше спостережених шаблонів та збережених підписів. Однак ці методи часто призводять до високої кількості помилкових позитивних результатів [20, 24-25].

Представимо багаторівневу ієрархію мережі Кохонена, або самоорганізуюча карта Кохонена, для реалізації системи виявлення аномальної поведінки користувачів. Навчання та тестування мережі необхідно провести, використовуючи попередньо оброблені вибірки з бази даних *KDD-99*. Спочатку потрібно використовувати *k*-Мар, де переможець отримує все, для впровадження *IDS*. Мережа *k*-Мар є корисною технікою загалом у тому сенсі, що це допомагає групувати подібні вхідні вектори у кластери. Однак це не гарантує оптимального поділу отриманого кластеру (рис. 2.1) [26]:

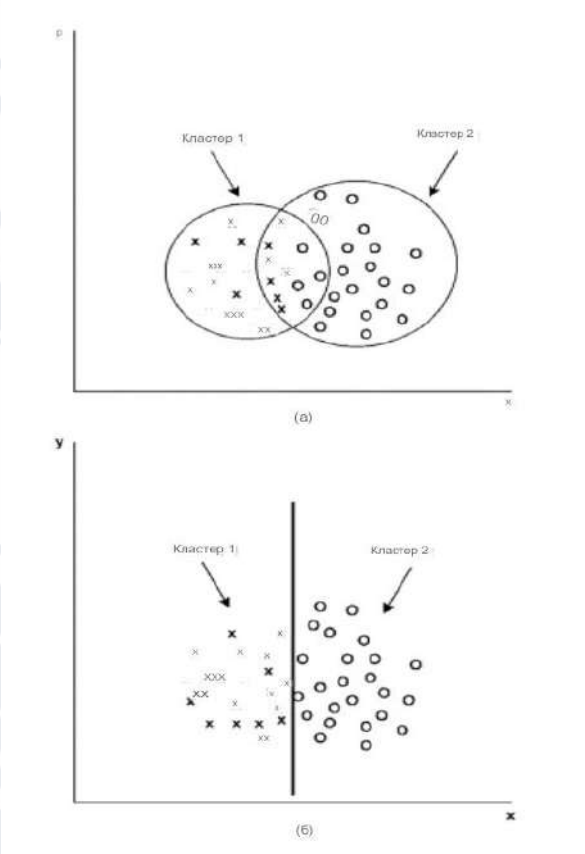

Рис. 2.1 - Вплив розмірів простору ознак на кластеризацію

На рис. 2.1, а показано кластер 1 та кластер 2, які утворилися, коли було використано двовимірне відображення підпростору та рис. 2.1, б ілюструє групування, коли одновимірний підпростір для класифікації

використовувалося відображення. Кластер 1 та 2 не є однорідними на рис. 1, а. Однак кластери 1 і 2 на рис. 2.1, б, отримані Із використанням 1-*D* підпростору відображення, є однорідним. Тобто кластери можуть значно відрізняються залежно від розмірності кластеризації межі [27].

У контексті *IDS* для мережевого трафіку в заголовках пакетів спостерігаються особливості, які є більш значущими ознаками аномальної діяльності. Вхідні вектори використовується для групування пакетів у звичайні. Наприклад, прапор, а також кількість неправильних фрагментів та використана послуга може вказувати на певний тип аномалії. З іншого боку, деякі окремі особливості самі по собі не будуть хорошим індикатором будь-якого роду аномалії. Наприклад, кількість байтів, які були передані із пункт призначення до джерела сам по собі не може бути показником аномальної поведінки, оскільки вона може мати широкий діапазон для нормальної діяльності самостійно. Однак у поєднанні із іншими функціями наприклад, використовуваний протокол, кількість пакетів від джерела до місця призначення, кількість неправильних фрагментів тощо, вони разом будуть кращим показником аномальної поведінки. Це тому, що мережева атака не може відбутися як одна дія. Можуть передувати багато дрібних, здавалося б, нешкідливих дій.

Перед атакою можна виконати кілька невеликих пробних дій рознесених у часі. Кожна варіація атаки буде проявлятися за допомогою певного зразку певного набору ознак. Так здається більше ефективно застосовувати багаторівневий ієрархічний підхід для *k*-мережі із набором функцій, ретельно підібраних для кожного рівня кластеризації. Щоб впоратися із цим проблемами, застосовують показник довіри як міру якості відображення на виходах *k*-Map. Обчислюється фактор для кожного нейрона, який вказуватиме на рівень довіри, із якою можна сказати, що кластер зіставленого нейрону нормальний або має певний тип аномалії.

Якщо група векторів вхідних ознак групуються у кластер нейрона на рівні зі 100% впевненістю, то вважаємо цей набір надійним класифікованим на рівні *i*. Ті вектори вхідних ознак, які були зіставлені із надійно класифікованими кластерами, не потрібно буде подавати на наступні рівні для подальшої обробки [20-24].

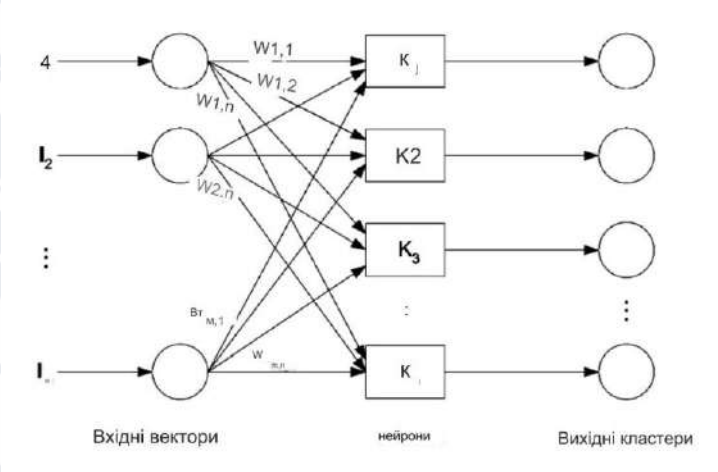

Рис. 2.2 - Карта Кохонена, переможець отримує все

Проста карта Кохонена, де переможець отримує все, складається із вхідних даних шар, шару нейронів та вихідного шару. Оскільки вона використовує лише один шар нейронів, її називають *k*-картою. Шар нейронів складається із набору нейронів, які можуть бути візуалізовані у вигляді стовпця (рис. 2.2). Кожен нейрон має відповідний вектор ваги.

Вхідний рівень слугує для передачі від кожного нейрону із набору вхідних даних векторів до різних нейронів у прошарку нейронів. Вхідний вектор прив'язаний до нейрона за допомогою вектора ваги. Вихідний рівень представляє набір кластерів, по одному для кожного нейрону.

Елементи вихідного кластера представляють групу вхідних векторів, які є найближчими до ваги нейрону *i*. Найкращий відповідний нейрон для вхідного вектору визначається на основі скалярного добутку нормалізованого вхідного вектору та нормалізованого вектору ваги цього

нейрону. Кожен вхідний вектор подається на усі нейрони. Оголошується нейрон, який дає максимальне значення для переможець або найкраща відповідна одиниця. Іншими словами, тільки один нейрон спрацьовує для певного вхідного вектору [29].

Як і будь-яка інша нейронна мережа, *k*-карта потребує навчання. Навчання здійснюється без нагляду за алгоритмом навчання, який описано нижче. У контексті алгоритму представлено коефіцієнт навчання для поточної ітерації.

Крок 1: отримати дані від користувача:

- кількість нейронів;

- розмір вхідного вектора, який визначається із вибірки функцій;

- поріг для визначення чисельності ітерацій навчання;

- вхідний файл, в якому знаходиться навчальна вибірка.

Крок 2: прочитати вхідні дані.

Крок 3: побудувати вхідні вектори за допомогою визначеної підмножини функцій.

Крок 4: створити *k*-карту із вказаною кількістю нейронів та обраною кількістю функцій.

Крок 5: ініціалізувати вагову матрицю за допомогою випадково обраних вхідних векторів та нормалізувати вагову матрицю.

Крок 6: обрання спостереження для тренувальних даних.

Крок 7: для будь-якого вектора у наборі даних знайти прототип, який найбільше на нього схожий. Цей прототип називається «найкращим відповідним блоком» (BMU). Розміщення вхідних векторів із тим самим BMU у одному кластері. Найкращий відповідний нейрон для вхідного

вектора визначається на основі скалярного добутки нормалізованого вхідного вектора та нормалізованого вектору ваги цього нейрона. Це значення позначимо як m_i.

Для кожного вхідного вектора I виконуємо цикл, де для кожного рядка вагової матриці обчислюємо m_i за формулою (2.1):

$$m_i = \sum_{k=0}^{F-1} W_{(i,k)} * I_k, \quad (2.1)$$

де

F – розмір вхідного вектора;

Обираємо переможця як нейрон j, де:

$$m_j = max(m_i), 0 \leqslant i \leqslant z, \quad (2.2)$$

де

z – кількість нейронів.

Налаштування ваги для переможця виконується за формулою (2.3):

$$W_{(\text{переможець},k)} \leftarrow W_{(\text{переможець},k)} + a\left(I_k - W_{(\text{переможець},k)}\right), 0 \leqslant k \leqslant z, \quad (2.3)$$

де

a – коефіцієнт налаштування.

Нормалізувати вагу для переможця. Якщо усі вагові вектори згруповано, то мережа навчена, якщо ж ні — повернутися до *кроку 5*.

Графічна ілюстрація алгоритму представлено на рис. 2.3:

Рис. 2.3 - Алгоритм навчання карти Кохонена

2.3 Розробка моделі штучної нейронної мережі для виявлення аномальної поведінки користувача

Один із аспектів системи виявлення аномальної поведінки користувачів у мережу (СВАП) успішним є його обізнаність про протокол. В контексті набору даних *KDD Cup* 99, кожен запис містить основні характеристики та особливості, які випливають із спостереження останніх підключень, функції, які спостерігалися за останні дві секунди і деякі особливості вмісту. Тестові набори даних перевірені та збережені у

відповідні файли. Бажано побачити ефект кількості використаних нейронів та кількості ітерацій навчання у виконанні *k*-карти. Маючи це на увазі, потрібно розробити графічний інтерфейс користувача, який керував би навчанням та тестував *k*-карту із використанням різних параметрів.

Деякі алгоритми виявлення аномалій для СВАП використовують дані без атак або звичайні дані для навчання. Потім перевіряються дані, які містять атаки, та подаються у систему для виявлення аномальної поведінки користувачів. У тестах використаємо навчальні дані, які містять як атаки, так і нормальні дані. Потім дослідимо склад кластерів на кожному нейроні. Деякі із цих кластерів містять записи певної мітки, наприклад, лише нормальні - однорідні кластери. Інші можуть містити записи двох або більше різних типів атак, таких як *smurf* та *normal*. Такі кластери називаються гетерогенними кластерами. Що стосується різнорідних кластерів, використаємо наступну техніку [30, 31]:

Крок 1: позначимо X_i як кількість записів типу X у кластері i, а загальну кількість записів з надписом X у навчальній вибірці — $X_{загальн.}$. Тоді ймовірність запису P_i про відображення типу X у кластері i обчислюється за формулою (2.4):

$$P_i = \frac{X_i}{X_{загальн.}} \tag{2.4}$$

Крок 2: позначимо загальну кількість записів у кластері i через N_i. Тоді ймовірність запису Q_i, відображеного у кластері i типу X обчислюється за формулою (2.5):

$$Q_i = \frac{X_i}{N_i} \tag{2.5}$$

Крок 3: визначаємо сприятливий фактор для надпису X як стик ймовірностей запису типу із набору вхідних даних зіставлення із кластером i запис, який зіставляється типу X:

$$F_i(x) = P_i Q_i \tag{2.6}$$

Позначимо $A = (a_1, a_2, \dots a_r)$ набір ідентифікаційних міток усіх записів, які зіставляються із нейроном i. Визначаємо надпис i за формулою (2.7):

$$L_i = a, F_i(a) = max_{x \epsilon A} F_i(a) \quad (2.7)$$

Визначаємо a як фактор впевненості, тобто рівень впевненості, із яким обирається тип запису, який домінує у кластері. Фактор впевненості служить основою для багаторівневої побудови мережі k-карт [32]. Він використовується для обрання вибірки даних для навчання. Коефіцієнт впевненості кластера C_i обчислюється за формулою (2.8):

$$C_i = \frac{max F_i(x)}{\sum_{x \epsilon A} F_i(x)}. \quad (2.8)$$

Визначаємо помилковий результат як випадок нормального результату, де
запис визначається як аномалія. Якщо екземпляр аномалії визначається як нормальний, це випадок пропущеного виявлення. Позначимо загальну кількість помилкових спрацьовувань як $N_{\text{помил.}}$, загальну кількість нормальних записів у тестовому наборі - $N_{\text{норм.}}$, загальну кількість атак у тестовому наборі — $N_{\text{аном..}}$, а загальну кількість пропущених випадків — $N_{\text{проп.}}$. Відсоток помилкових спрацьовувань обчислюється за формулою (2.9):

$$PN = 100 \frac{N_{\text{помил}}}{N_{\text{норм}}} \quad (2.9)$$

Відсоток виявлених аномалій обчислюється за формулою (2.10):

$$PA = 100 \frac{N_{\text{аном}} - N_{\text{проп}}}{N_{\text{аном}}} \quad (2.10)$$

Як правило, мережа k-Мар може визначити один тип атаки за інший вид атаки, але у даній роботі нас цікавить саме не тип атаки, а лише те, чи запис нормальний чи містить аномальну поведінку користувача.

Використаний набір контрольних навчальних даних KDD Cup 99 містить 494021 запис, кожен із яких містить 41 ознаку. Є 22 різні типи атак на додаток до звичайних записів. Для навчання мережі необхідно створити кілька файлів у якості навчальних вибірок. Склад 4 навчальних вибірок (вибірка 1, вибірка 2, вибірка 3, вибірка 4, вибірка 5) представлено у таблиці 2.1.

Вибірка 1 сформована та містить записи усіх 22 типів атак із загальною кількістю 40902 записів. У вибірках 2 та 3 є 15 у кожному і 13 типів атак відповідно. Записи навчальних вибірок обираємо випадково із тренувального набору KDD Cup 99 так, щоби виключити дублікати у одній вибірці. Вибірка4 містить 168522 записів, де є усі типи атак.

Таблиця 2.1 - Набори навчальних вибірок та тестувальних даних

Тип атаки	KDD CUP	Вибірка1	Вибірка 2	Вибірка 3	Вибірка 4	Вибірка 5	Тест1	Тест1
відсутня	97278	22000	8500	8800	97278	20492	60000	60000
neptune	107201	5050	26000	9329	45500	21385	57000	57000
smurf	280790	5100	9500	26000	16999	0	163000	9
back	2203	2203	186	203	2203	0	1030	0
satan	1589	1589	113	108	1589	488	1589	1589
ipsweep	1247	1247	200	100	1240	0	300	0
portsweep	1040	1040	103	88	1040	230	360	360
warezclient	1020	1020	77	94	1020		0	
teardrop	979	979	79	979	979		15	
pod	264	264	26	20	264		73	
rmap	231	231	22	21	231		67	

guess_passwd	53	53	13	17	53		49	
buffer_overflow	30	30	9	12	30		29	
land	21	21	3	7	21		16	
waremaster	20	20	1	0	20		11	
imap	12	12	1	0	12		7	
rootkit	10	10	3	0	10		9	
loadmodule	9	9	0	0	9		3	
ftp_write	8	8	1	0	8		2	
multihop	7	7	0	4	7		0	
phf	4	4	2	0	4		2	
perl	3	3	0	3	3		2	
spy	2	2	0	2	2		0	
xlock								9
sendmail								14
saint								750
apache2								800
mscan								1041
updstorm								3
processtable								779
worm								2
mailbomb								4000
Загалом	494021	40902	44839	45787	168522	42595	283564	126356

Під час тестування необхідно побудувати вектор для кожного запису даних *KDD Cup* 99 у тестовому наборі за допомогою набору функцій, який використовувався для навчання. Тестова вибірка містить додаткові типи атак яких немає у жодному із навчальних вибірок.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ВИЯВЛЕННЯ АНОМАЛЬНОЇ ПОВЕДІНКИ КОРИСТУВАЧІВ ЦЕНТРІВ ОБРОБКИ ДАНИХ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ

3.1 Вибір середовища та системи для програмної реалізації

Для розробки даного програмного продукту потрібно обрати мову програмування, яка використовує принципи об'єктно-орієнтованого програмування.

3.1.1 Моделювання нейронних мереж у пакеті *Matlab*

Matlab являє собою математичний пакет, призначений для вирішення задач обчислювальної математики та побудови чисельних моделей складних об’єктів і процесів (рис. 3.1):

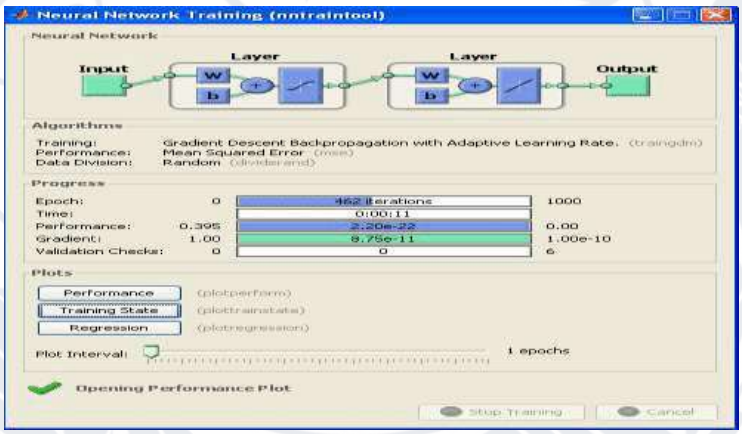

Рис. 3.1 – Головна діалогова форма засобу *nntool*

Пакет *Matlab* складається з інтерпретатора – модельного середовища, що має термінальний інтерфейс, ядра (набору найпростіших стандартних операцій, функцій і процедур для обчислень), а також бібліотек функцій (*Toolbox*).

Перевагами пакета є:

- багаті графічні можливості;
- великий набір математичних функцій;

- простота вбудованої мови *Matlab*;

- можливість автоматичного перетворення текстів програм мовою *Matlab* у тексти програм на мові Сі, а також те, що тексти бібліотечних функцій постачаються у вихідному виді.

До недоліків пакета варто віднести низьку швидкість роботи.

Для моделювання НМ за допомогою пакета *Matlab* необхідно встановити і використовувати бібліотеку *Deep Learning Toolbox* (раніше – *Neural Network Toolbox*). Бібліотека містить як функції для моделювання мілких (*shallow networks*), так і глибоких (*deep networks*) нейромереж [34].

3.1.2 Deep Network Designer

Моделювання глибоких нейромереж можна здійснити шляхом використання засобу *Deep Network Designer* (рис 3.2) [34].

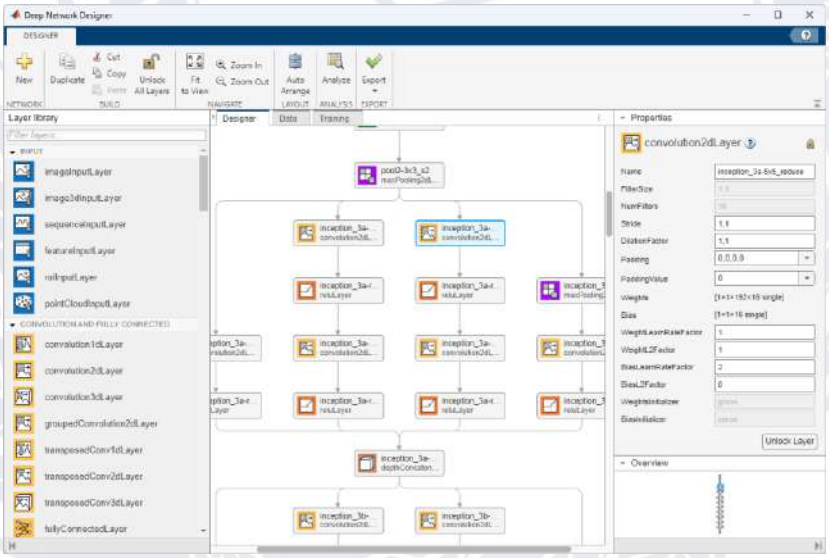

Рис. 3.2 – Головна діалогова форма засобу *Network Designer*

Він забезпечує дружній інтерфейс користувача для конструювання, візуалізації, навчання, редагування глибоких нейромереж. Також засіб дозволяє зберігати та завантажувати нейромоделі, копіювати, видаляти та редагувати їхні складові, а також аналізувати мережу, імпортувати дані,

налаштовувати параметри мереж та методів навчання, слідкувати за точністю. Для запуску засобу потрібно увести команду: *deepNetworkDesigner*.

Deep Network Designer надає вибір варіантів збільшення зображень. Можна ефективно збільшити обсяг навчальних даних, застосувавши до даних рандомізоване збільшення. Якщо треба збільшити дані, *Deep Network Designer* випадковим чином вносить зміни до даних на кожній епосі. Кожна епоха потім використовує дещо інший набір даних. Імпорт тестових даних здійснюють, обравши папку або імпортуючи *imageDatastore* з робочої області [34].

3.1.3 Microsoft Visual Studio

Microsoft Visual Studio – інтегроване середовище розробки, що включає інструментальні засоби для проектування, кодування, транслювання, налагодження та виконання програм [35]. *Visual Studio* дозволяє швидко створювати та впроваджувати різноманітні програми на базі *ОС Windows*, веб-додатки та програми для мобільних пристроїв.

Рис. 3.3 – *Microsoft Visual Studio*

У *Visual Studio* пропонується ціла низка шаблонів додатків, корисних під час створення програм, і кілька мов програмування, якими можна написати ці програми: *Visual Basic*, *Visual C#*, *Visual C++*, *JScript* тощо [35].

Visual Studio підтримує підключення бібліотеки *DevExpress* із широким спектром можливостей.

У додатки створювані за допомогою *Visual Studio* можна впроваджувати різні технології. Нижче наведено опис деяких із них:

- *.NET Framework, .NET Framework 3.5, .NET Framework 3.0, .NET Compact Framework* - це інтегрований компонент *Windows*, який підтримує створення та виконання нового покоління додатків та веб-служб *XML*.

- *Windows Presentation Foundation* (*WPF*) - *WPF* є набір типів *.NET Framework*, який можна використовувати для створення зовнішнього вигляду клієнтських програм *Windows*. *WPF* складається з таких компонентів, як розширювана мова виправлення для програм *XAML*, елементи керування, прив'язка даних, двовимірна та тривимірна графіка,

анімація, стилі, шаблони, документи, мультимедійні дані, текст та типографічні засоби.

- *Silverlight* – це незалежна від оглядача та платформи технологія, що дозволяє проектувати, розробляти та постачати інтерфейси з підтримкою мультимедіа та багатофункціональні програми в Інтернеті.

- *Windows Forms* – дозволяє розробляти прості у розгортанні та оновленні програми з широкими графічними можливостями. Крім того, при доступі до *Windows Forms* до ресурсів на локальному комп'ютері забезпечується більш високий рівень безпеки, ніж при роботі традиційних програм *Windows*.

- Мова *XAML* – це мова розмітки для декларативної розробки програм. *Windows Presentation Foundation* (*WPF*) реалізує завантажувач *XAML* і забезпечує підтримку мови *XAML* для типів *WPF*, тому більшу частину інтерфейсу програми можна створювати за допомогою розмітки *XAML*.

- *ASP.NET* надає платформу, яку можна використовувати для створення веб-застосунків. До її складу входять такі служби, як керування станом, обробники *HTTP*, модулі *HTTP* та маршрутизація *ASP.NET*.

3.1.4 Embarcadero Delphi

Embarcadero Delphi - просунуті програмні інструменти для розробників програм для *Windows, Android, iOS, macOS* та *Linux*. Потужне *RAD*-середовище для швидкої розробки високопродуктивних крос-платформних *native*-додатків із використанням потужних засобів візуального проектування та інтегрованих наборів інструментів, що подобається як незалежним, так і корпоративним розробникам (рис. 3.4).

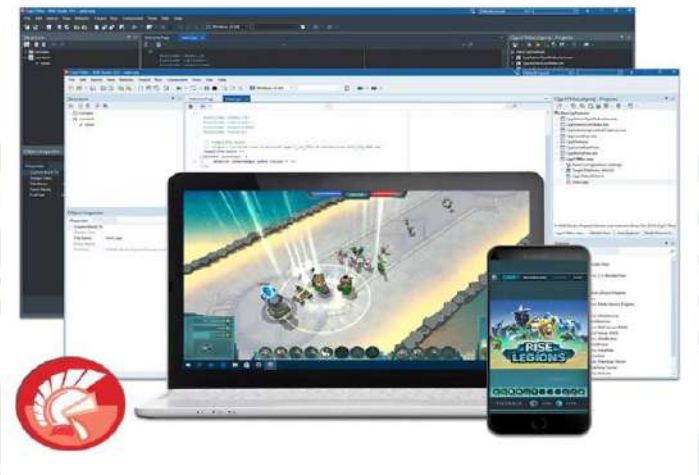

Рис. 3.4 - Embarcadero Delphi

Фреймворк бібліотеки візуальних компонентів для Windows та візуальне середовище *FireMonkey* для міжплатформних інтерфейсів забезпечують основу для створення інтуїтивних і красивих інтерфейсів, що вражають на будь-якій платформі: *Windows*, *macOS*, *iOS*, *Android* та *Linux* [36].

Швидше добирайтеся до причини помилки за рахунок використання інтегрованого міжплатформного налагодження коду для інструментальної платформи. Швидкі цикли розробки не обов'язково призводять до погіршення якості. *Delphi* включає безліч функцій, покликаних впровадити передові методи при написанні коду, знизити дублювання та допомогти стати суперрозробником.

Розглянувши середовища розробки програмного забезпечення, створено порівняльну таблицю 3.1.

Таблиця 3.1 – Порівняльний аналіз середовищ розробки ПЗ

Параметри	MathLab	Visual Studio	Embarcadero Delphi
Редактор користувальницького інтерфейсу	-	+	+
Вбудований профілювальник	-	+	+
Вбудований відладчик	-	-	+
Підтримка C#	+	-	-
Встановлене ПЗ на підприємстві, наявність ліцензії	+	-	+
Опис використання	+	-	+
Сумісність з колишніми проектами	+	-	+

Для розробки програми обрано середовище розробки *Embarcadero Delphi* із безкоштовною версією *Community Edition* 10.4, оскільки воно має ряд переваг, необхідних для розробки функцій, таких як підтримка профілювання коду мовою програмування *Objective Pascal*, налагодження та сумісність із успадкованими проектами.

3.2 Нормалізація вхідних даних

Для моделювання нейроподібної *k*-мережі, яка призначена виявляти аномальну поведінку користувачів на комп'ютерну мережу, потрібно виконати нормалізацію вхідних даних. Проведено аналіз за кількістю еталонів кожного класу атак. Атаки, які мали менше 200 еталонів, виключаємо із розгляду.

Перейдемо до перетворення параметрів, які містили нечислові типи даних, тобто типу String. Отже, необхідна заміна їх на числові коефіцієнти. До параметрів, що потребують конвертації, належить:

- тип протоколу;
- сервіс;
- *flag*;
- *label*.

Параметр "Тип протоколу" має 3 значення, кожному із яких присвоїмо числовий коефіцієнт:

- *Icmp* присвоєно значення 0;
- *Tcp* – значення 1;
- *Udp* – значення 2.

Отже, виявлення аномальної поведінки користувачів центрів обробки даних здійснено за допомогою використання багатошарової нейронної мережі Кохонена із навчанням, на вхід якої подається 41 параметр із метою кластеризації типів аномальної поведінки користувачів. Для моделювання нейромережі використовуються нормалізовані дані, що описано у розділі 2.

3.3 Огляд розробленого програмного продукту та аналіз експериментальних результатів

Головне вікно розробленого програмного середовища представлено на рис. 3.5:

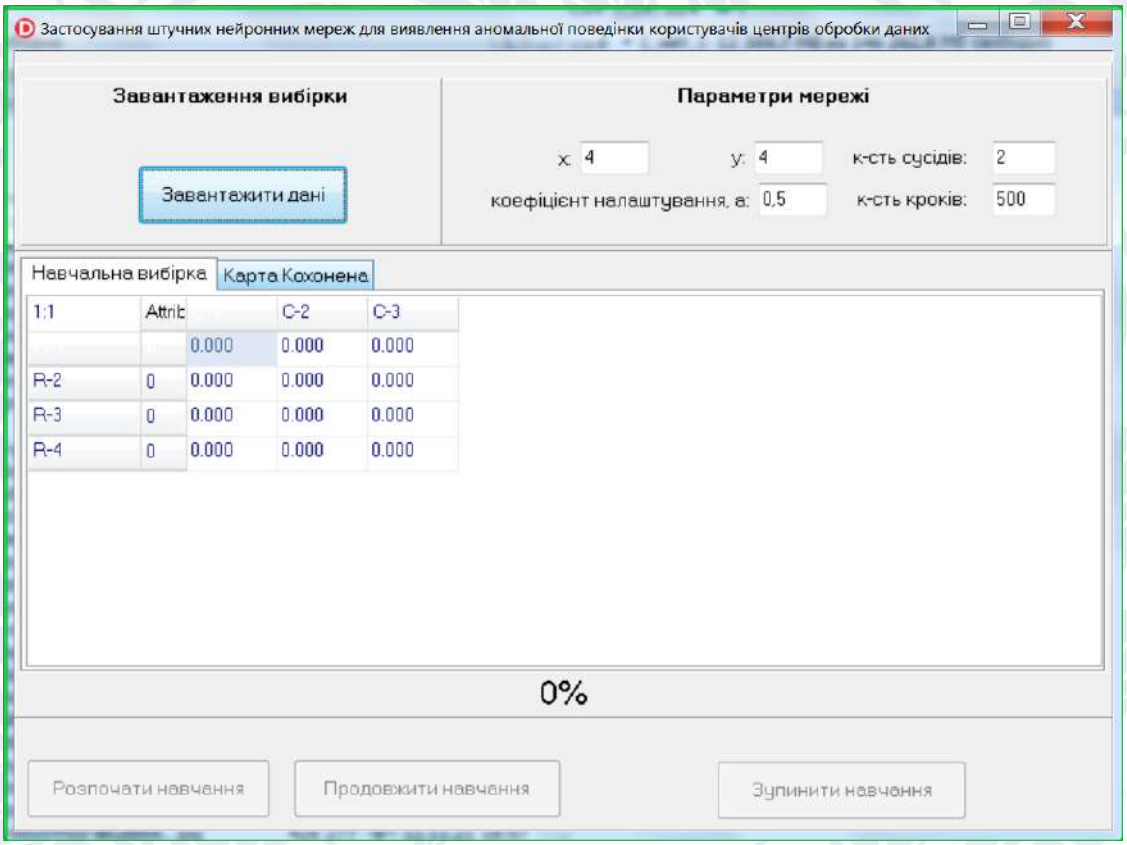

Рис. 3.5 – Головне вікно програми

Розроблений програмний засіб має зручний інтерфейс користувача. Голвне вікно програми логічно розділене на 3 блоки:

- завантаження вибірки;
- параметри мережі;
-навчальна вибірка та карта Кохонена.

Блок «завантаження вибірки» містить кнопку «Завантажити дані» Натиснувши кнопку «Завантажити дані», завантажуємо дані навчальної

вибірки (рис. 3.6). Навчальні вибірки підготовлені у розділі 2, використовуючи базу *KDD99*.

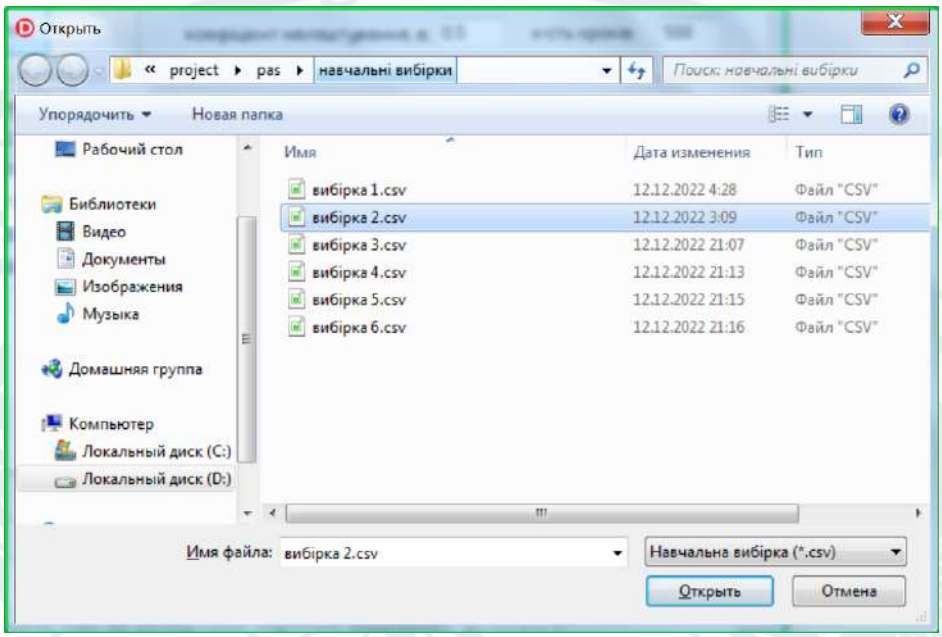

Рис. 3.6 – Завантаження навчальної вибірки

Результат завантаження навчальної вибірки представлено на рисунку 3.7:

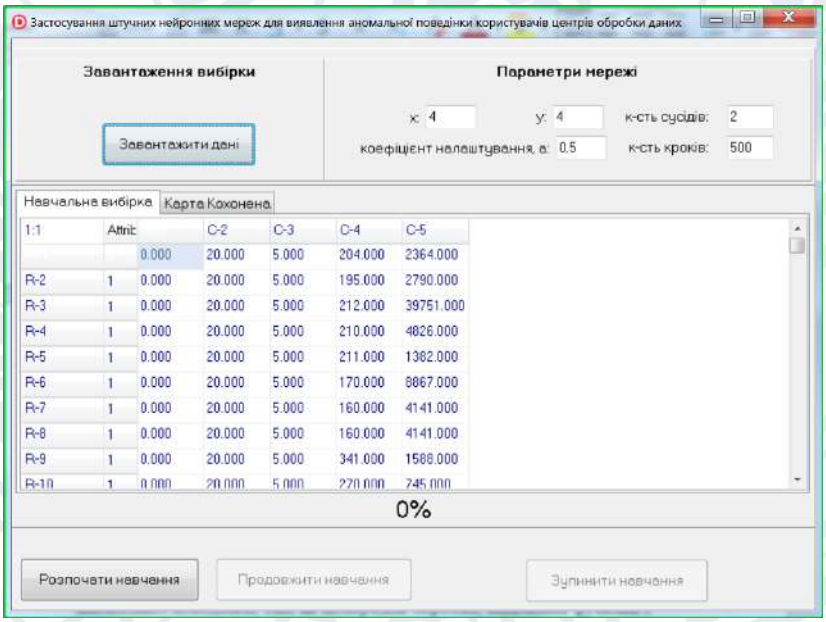

Рис. 3.7 – Результат завантаження навчальної вибірки

Після завантаження навчальних даних, переходимо до блоку «параметри мережі». У даному блоці можна вказати:

- розмір вихідної матриці (*X*, *Y*);
- коефіцієнт налаштування *a,* за замовчанням встановлене значення 0,5;
- кількість сусідів, за замовчанням встановлене значення 2;
- кількість кроків навчання, за замовчанням встановлено значення 500.

Переходимо до процесу навчання мережі, використовуючи попередню нормалізацію вхідних даних відносно наступних параметрів: protocol_type, service, flag. Для цього потрібно натиснути кнопку (рис. 3.8):

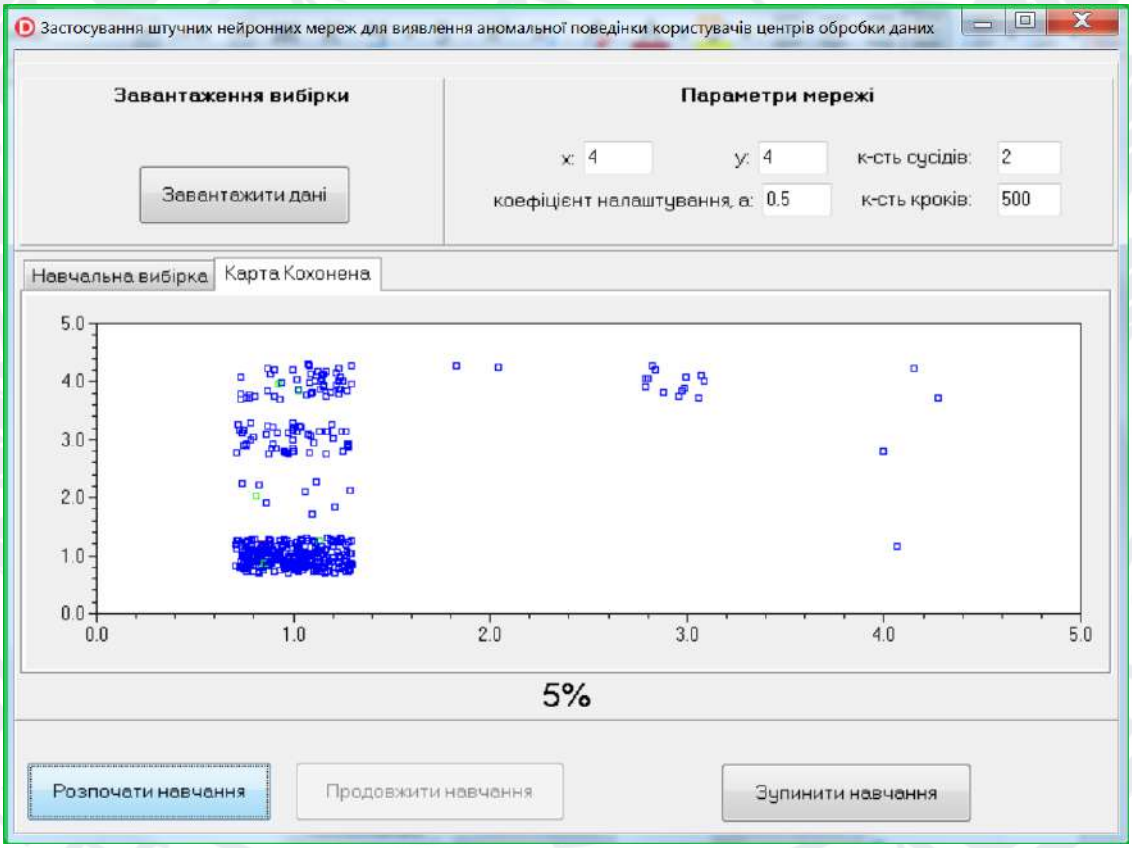

Рис. 3.8 – Процес навчання мережі

Розроблений програмний засіб дозволяє зупиняти процес навчання та при необхідності проводити донавчання мережі. Для цього необхідно

натиснути кнопку «Продовжити навчання». Після успішного навчання у блоці «Карта Кохонена» отримуємо результат навчання (рис. 3.9):

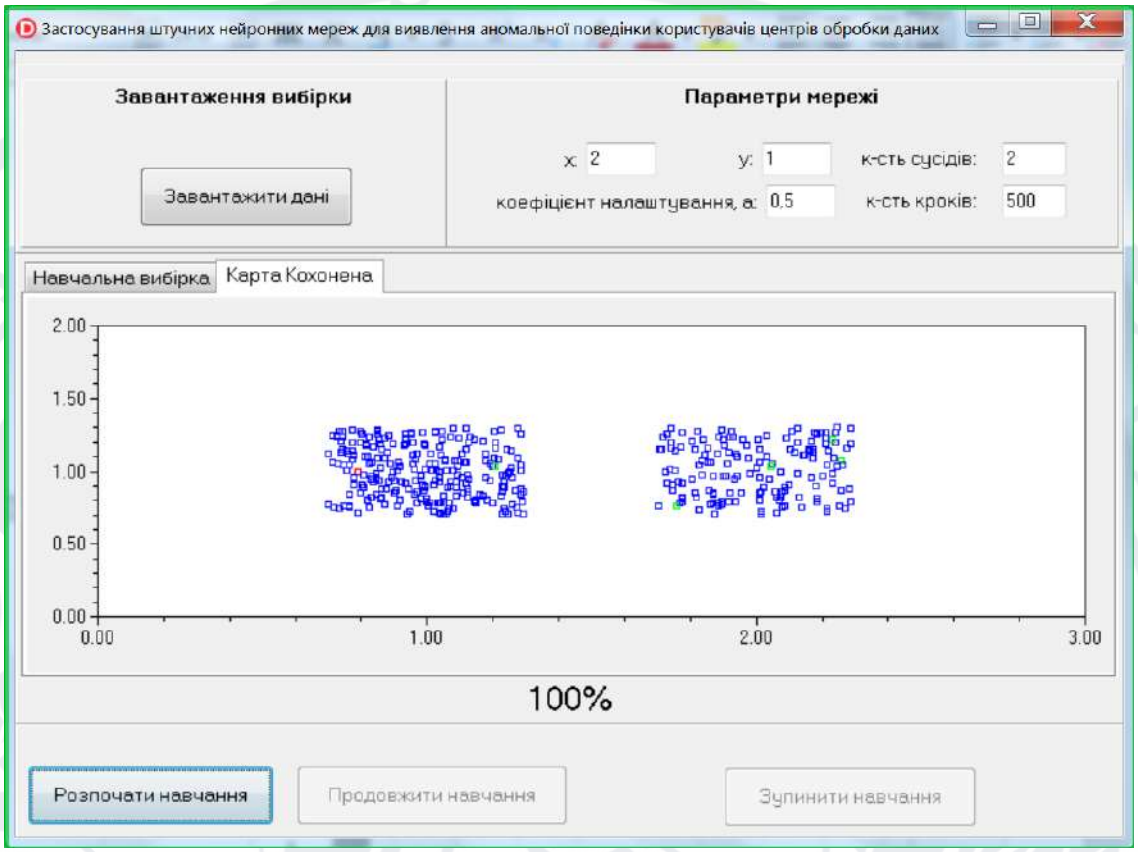

Рис. 3.9 – Результат навчання мережі

Обравши весь набір із 41 функції для *k*-Мар кластеризація може бути непродуктивною. При виборі підмножини простору ознак, корисно візьмемо до уваги деякі предметні знання з області виявлення аномальної поведінки. Наприклад, протокол *ICMP* іноді використовується зловмисниками, які створюють багато запитів із підробленої *IP*-адреси, які потім транслюються через цільову мережу. Якщо всі сигнали на центр обробки даних почнуть надсилатися відповідь відправнику, вона може погіршити продуктивність мережі. Напад атаки *smurf* є прикладом такої аномальної поведінки на відмову в обслуговуванні. Однак нам потрібно зрозуміти, що цілком нормальне з'єднання також може використовуватися [33].

Багато операцій сканування для отримання інформації про активні хости та можливі вразливості у мережі також зроблено із використанням протоколу *ICMP* та служби ехо-запитів. Наприклад, один режим сканування використовує протокол та службу *ICMP*. У цьому випадку кількість байтів, переданих від джерела до адресата зазвичай становить 8. Коли пакет на основі *UDP* надсилається на пристрій, маршрутизатор іноді не може доставити пакет до пункту призначення, або тому, що пункт призначення є тимчасовим недоступний або адреса призначення не існує. У таких випадках мережа недоступна або порт недоступний *ICMP* і повідомлення повертається. Інший режим - через протокол *TCP* із напіввідкритим *SYN*. Окрім сканування діяльності, яка може бути аномальною, існує багато інших типів атак, які викликають атаки на відмову у обслуговуванні. Краплеподібна атака використовує слабкість у повторній збірці фрагментів пакету шляхом створення фрагментів із зміщенням. Спроби зібрати такі неправильні фрагменти призводять до невдач наприклад, збій, зависання або перезавантаження у місці призначення хосту. Інші типи нападів потребують спостереження протягом певного періоду час вказати на аномальну діяльність. Тоді є показання, наприклад, кількість невдалих спроб входу, які вказують на ймовірність вторгнення. Таким чином, різний характер таких функцій, як обслуговування, протокол, кількість байтів від джерела до місця призначення, прапор, кількість неправильних фрагментів є одними із важливих характеристик які можна використати у формулюванні підпростору ознак для *k*-карт [20-24, 30-32].

Для експериментів визначимо різні підмножини функцій, які добре виявляють певні типи атак. Обраний підхід до вибору функцій групує функції у термінах їхніх узгоджених зв’язків (таких як статичний *IP*-демпінг проти динамічного і пов’язані з часом, тимчасові атрибути трафіку

даних, тощо) та взаємні інклюзивність. Результатом такого підходу є те, що у результаті кластери ієрархічної *k*-карти можуть бути більш згуртованими порівняно до випадкового вибору груп ознак. Три з підмножини, які ми використовували для випадків тестування *k*-карти перераховані таким чином:

Набір функцій 1:

- використовуваний протокол, наприклад *ICMP*, *TCP*, *UDP*;

- сервіси, такі як *http, ftp, smtp* тощо;

- кількість байтів, переданих із джерела до місця призначення;

- кількість байтів, переданих із пункту призначення до джерела.

Набір функцій 2:

- використовуваний протокол, наприклад *ICMP, TCP, UDP*;

- сервіси, такі як *http, ftp, smtp* тощо;

- прапор;

- тривалість з'єднання;

- кількість байт, переданих із джерела до місця призначення;

- кількість байтів, переданих із пункту призначення до джерела;

- кількість знайдених неправильних фрагментів при повторному навчанні;

- кількість термінових пакетів;

- кількість підключень, зроблених до того самого хосту у заданий проміжок часу;

- частка підключень із того самого хосту, який мав помилки *SYN* у вказаний проміжок часу;

- частка з'єднань із того самого хосту та тієї самої служби, які мали помилки синхронізації;

- частка підключень із того самого хосту, який мав помилку *REJ*;

- частка з'єднань із того самого хосту і тієї самої служби, які мали помилку *REJ*;

- частка підключень із того самого хосту, який використовував ту саму послугу.

- *sameHstDiffSvcRate*, частка підключень з той самий хост, який використовував різні служби;

- *sameSvcDiffHstRate*.

Набір функцій 3:

- *sameHstSynErrRate*;

- *sameSvcSynErrRate*;

- *sameHstRejErrRate*;

- *sameSvcRejErrRate*;

- *sameHstSameSvcRate*;

- *sameHstDiffSvcRate*;

- *sameSvcDiffHstRate*.

Таблиця 3.1 - Результати тестування набору даних 1 для *k*-карти

Набір функцій	Тестувальний набір	Виявлено, %	Помилковий результат, %
1	1	97,18	0,82
	2	75,82	0,75
	3	77,14	1,15
2	1	99,90	0,57
	2	98,23	0,99
	3	94,17	1,01
3	1	99,94	0,75
	2	99,92	0,87
	3	99,78	1,42

Після вибору описаних навчальних вибірок та підмножин функцій, обрано 40 нейронів і поріг для експериментів. Тобто навчальні ітерації продовжуються поки кількість змінених відображень не стане меншою за 2. Результати, отримані для тестових випадків щодо різних наборів функцій і навчальні набори даних, наведені у таблиці 3.1. Далі використаємо 50 нейронів та повторимо ті самі експерименти. Результати наведено у таблиці 3.2.

Таблиця 3.2 - Результати тестування набору даних 2 для *k*-карти

Набір функцій	Тестувальний набір	Виявлено, %	Помилковий результат, %
1	1	98,41	0,74

	2	83,12	0,68
	3	79,28	1,03
2	1	99,14	0,87
	2	99,15	0,9
	3	92,68	1,02
3	1	99,98	0,84
	2	99,98	0,56
	3	99,45	0,93

Насправді, у деяких випадках кількість помилкових результатів збільшувалася зі збільшенням числа нейронів 50. Це свідчить про різноманітність атак та у той же час низький рівень помилкових спрацьовувань. З іншого боку, також видно, що деякі особливості вибірок мають кращу продуктивність у конкретному тестовому випадку. Більш детальне спостереження у результати експериментів показують, що деякі набори функцій більш чутливі до виявлення певних типів аномалій. Із таблиць 3.1 та 3.2 видно, що незважаючи на виявлений рівень помилкових результатів, він є прийнятним для більшості наведених тестових випадків.

Таким чином, розроблено програмний продукт для реалізації системи із застосуванням штучних нейронних мереж для виявлення аномальної поведінки користувачів центрів обробки даних.

ВИСНОВКИ

Надійний мережевий зв’язок має вирішальне значення у повсякденному функціонуванню сучасного світу. Компанії, навчальні заклади, державні відомства та навіть людина сильно покладається на безперервне спілкування мережеве обслуговування та обчислювальні засоби для проведення повсякденні операції. Спроби порушити надійний потік мережі також значно збільшилися. Мережеві комп'ютерні систем все більше стати мішенню для зловмисників, незалежно від того, чи є вони просто хакерами, ворогами чи злочинцями. Багато із цих атак дорого коштують..

У ході виконання магістерської роботи було досліджено використання технології *k-карт* у сфері захисту інформації та створено алгоритм застосування штучних нейронних мереж для виявлення аномальної поведінки користувачів центрів обробки даних.

В результаті роботи вдосконалено структуру нейронної мережі для виявлення аномальної поведінки користувачів центрів обробки даних на основі технології *k*-мережі та розроблено програмне забезпечення для виявлення аномальної поведінки користувачів центрів обробки даних, використовуючи відповідні технології штучних нейронних мереж, а також розроблено застосунок для демонстрації роботи навчання мережі наочно. ПЗ розроблено у вигляді додатку на мові програмування *Delphi* та з використанням фреймворку *kohMap*.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. A Review on Machine Learning Approaches for Network Malicious: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8145138/, Дата доступу: 20.11.2023.

2. Deep Learning for Anomaly Detection: [Електронний ресурс] // Режим доступу до ресурсу URL : https://ff12.fastforwardlabs.com/, Дата доступу: 20.11.2023.

3. Detection of anomalies in cycling behavior with convolutional neural network: [Електронний ресурс] // Режим доступу до ресурсу URL : https://etrr.springeropen.com/articles/10.1186/s12544-023-00583-4, Дата доступу: 20.11.2023.

4. Survey of intrusion detection systems: techniques, datasets: [Електронний ресурс] // Режим доступу до ресурсу URL : https://cybersecurity.springeropen.com/articles/10.1186/s42400-019-0038-7., Дата доступу: 20.11.2023.

5. Comparing artificial neural network training algorithms to prediction: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9733380/., Дата доступу: 20.11.2023.

6. Convolutional neural networks: an overview and application: [Електронний ресурс] // Режим доступу до ресурсу URL : https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9, Дата доступу: 20.11.2023.

7. Machine learning, explained | MIT Sloan: [Електронний ресурс] // Режим доступу до ресурсу URL : https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained, Дата доступу: 20.11.2023.

8. Top 10 Deep Learning Algorithms You Should Know in 2023: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm, Дата доступу: 20.11.2023.

9. What Is ChatGPT Doing … and Why Does It Work?: [Електронний ресурс] // Режим доступу до ресурсу URL : https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/, Дата доступу: 20.11.2023.

10. What are Neural Networks? | IBM: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.ibm.com/topics/neural-network/, Дата доступу: 20.11.2023.

11. What is Perceptron? A Beginners Guide for 2023: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.simplilearn.com/tutorials/deep-learning-tutorial/perceptron, Дата доступу: 20.11.2023.

12. What is deep learning and how does it work?: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network, Дата доступу: 20.11.2023.

13. Aravindpai Pai. (8489). Analyzing 6 Types of Neural Networks in Deep Learning: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/, Дата доступу: 20.11.2023.

14. On the Training Algorithms for Artificial Neural Network in Predicting: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.hindawi.com/journals/complexity/2021/5548988/, Дата доступу: 20.11.2023.

15. A survey on driving behavior analysis in usage based insurance: [Електронний ресурс] // Режим доступу до ресурсу URL :

https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0249-5, Дата доступу: 20.11.2023.

16. Applied Sciences | Free Full: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.mdpi.com/2076-3417/9/20/4396, Дата доступу: 20.11.2023.

17. Convolutional neural networks: an overview and application: [Електронний ресурс] // Режим доступу до ресурсу URL : https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9, Дата доступу: 20.11.2023.

18. Intrusion detection systems for IoT: [Електронний ресурс] // Режим доступу до ресурсу URL : https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-018-0123-6., Дата доступу: 20.11.2023.

19. Network Attacks Detection Methods Based on Deep Learning: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.hindawi.com/journals/scn/2020/8872923/, Дата доступу: 20.11.2023.

20. Anomaly behavior detection analysis in video surveillance: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.spiedigitallibrary.org/journals/journal-of-electronic-imaging/volume-32/issue-04/042106/Anomaly-behavior-detection-analysis-in-video-surveillance--a-critical/10.1117/1.JEI.32.4.042106.full, Дата доступу: 20.11.2023.

21. A Study in Using Neural Networks for Anomaly and Misuse Detection: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.usenix.org/events/sec99/full_papers/ghosh/ghosh_html/, Дата доступу: 20.11.2023.

22. AI and the Internet of Behaviors: Exploring the Intersection of Data: [Електронний ресурс] // Режим доступу до ресурсу URL :

https://vtmit.vt.edu/academics/student-experience/blog/ai-internet-behaviors.html, Дата доступу: 20.11.2023.

23. Anomaly detection: [Електронний ресурс] // Режим доступу до ресурсу URL : https://en.wikipedia.org/wiki/Anomaly_detection, Дата доступу: 20.11.2023.

24. Anomaly detection: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5708737/, Дата доступу: 20.11.2023.

25. Fortinet Named a Strong Performer in the Forrester Wave: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.fortinet.com/blog/business-and-technology/forrester-nav-report-network-detection-and-respons, Дата доступу: 20.11.2023.

26. Results From Invoking Artificial Neural Networks to Measure Insider: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.osti.gov/pages/servlets/purl/1831170, Дата доступу: 20.11.2023.

27. Using Artificial Intelligence to Address Criminal Justice Needs: [Електронний ресурс] // Режим доступу до ресурсу URL : https://nij.ojp.gov/topics/articles/using-artificial-intelligence-address-criminal-justice-needs, Дата доступу: 20.11.2023.

28. Enhancing anomaly detection in distributed power systems: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10437833/, Дата доступу: 20.11.2023.

29. What is Machine Learning and How Does It Work: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML, Дата доступу: 20.11.2023.

30. Artificial intelligence in disease diagnosis: a systematic literature: [Електронний ресурс] // Режим доступу до ресурсу URL :

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8754556/, Дата доступу: 20.11.2023.

31. Understanding User Behavior: [Електронний ресурс] // Режим доступу до ресурсу URL : https://mhealth.jmir.org/2020/11/e21209, Дата доступу: 20.11.2023.

32. Machine Learning: Trying to detect outliers or unusual behavior: [Електронний ресурс] // Режим доступу до ресурсу URL : https://srnghn.medium.com/machine-learning-trying-to-detect-outliers-or-unusual-behavior-2d9f364334f9., Дата доступу: 20.11.2023.

33. Machine Learning: Trying to detect outliers or unusual behavior: [Електронний ресурс] // Режим доступу до ресурсу URL : https://srnghn.medium.com/machine-learning-trying-to-detect-outliers-or-unusual-behavior-2d9f364334f9., Дата доступу: 20.11.2023.

34. ПРОГРАМНІ ЗАСОБИ ДЛЯ МОДЕЛЮВАННЯ НЕЙРОМЕРЕЖ : [Електронний ресурс] // Режим доступу до ресурсу URL : https://moodle.znu.edu.ua/pluginfile.php/675808/mod_resource/content/1/Subbotin_Neural-100-138.pdf., Дата доступу: 20.11.2023.

35. Visual Studio: IDE and Code Editor for Software Developers and Teams: [Електронний ресурс] // Режим доступу до ресурсу URL : https://visualstudio.microsoft.com/., Дата доступу: 20.11.2023

36. C++Builder: Software Overview - Embarcadero: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.embarcadero.com/products/cbuilder, Дата доступу: 20.11.2023

37. Delphi: IDE Software Overview - Embarcadero: [Електронний ресурс] // Режим доступу до ресурсу URL : https://www.embarcadero.com/ru/products/delphi, Дата доступу: 20.11.2023.

ДОДАТОК А — ПРОГРАМНИЙ ЛІСТИНГ

```
unit kMapMain;
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, DateUtils, Grids, ComCtrls, Tabnotbk, StdCtrls, Buttons,
ExtCtrls,
  SDL_vector, SDL_kohonen, SDL_rchart, SDL_ntabed, SDL_NumIO, SDL_NumLab, SDL_math1, SDL_filesys;

type
  TMainForm = class(TForm)
    Panel1: TPanel;
    NoteBk: TTabbedNotebook;
    RCKohMap: TRChart;
    TEData: TNTabEd;
    OpenDialog1: TOpenDialog;
    NIOysize: TNumIO;
    NIOxsize: TNumIO;
    Label1: TLabel;
    Label2: TLabel;
    NIONeighb: TNumIO;
    Label3: TLabel;
    SaveDialog1: TSaveDialog;
    KohMap1: TKohonen;
    RowVect: TVector;
    Panel2: TPanel;
    Panel3: TPanel;
    BButLoad: TButton;
    BButGenerate: TButton;
    Splitter2: TSplitter;
    Panel4: TPanel;
    Label5: TLabel;
    Label6: TLabel;
    BButTrainIt: TButton;
    BButContTrain: TButton;
    CBShowTraining: TCheckBox;
    BButAbort: TButton;
    Panel5: TPanel;
    Label4: TLabel;
    LabelProgress: TLabel;
    Label7: TLabel;
    NLAlpha: TNumIO;
    NLStepsLabel: TLabel;
    NLSteps: TNumIO;
    ResultsLabel: TLabel;
    procedure KohMap1Feedback(Sender: TObject; PercentDone: Integer);
    procedure BButLoadClick(Sender: TObject);
    procedure BButTrainItClick(Sender: TObject);
    procedure BButContTrainClick(Sender: TObject);
    procedure BButAbortClick(Sender: TObject);
  private
    procedure ShowCurrentNetResponse;
  public
  end;

var
  MainForm: TMainForm;

implementation
{$R *.DFM}

const
  //масив кольорів на графі
  MaxPreDefColors = 10;
  PredefinedColors: array [0..MaxPreDefColors-1] of TColor =
            (clRed, clBlue, clLime, clYellow, clFuchsia,
             clBlack, clMaroon, clGreen, clOlive, clNavy);

//процедура завантаження даних
procedure TMainForm.BButLoadClick(Sender: TObject);
var
  i,j,k,ParamsCount,StringsCount: integer;
  f: TextFile;
  s,val: string;
```

```
  str: TStringList;
  lastval: double;
begin
  //показуємо стартову сторінку
  NoteBk.PageIndex := 0;

  //завантажуємо файл навчальної вибірки у форматі csv
  OpenDialog1.Filter := 'Навчальна вибірка|*.csv';
  if OpenDialog1.Execute then
  begin
    BButGenerate.Enabled := false;

    //читаємо файл даних із навчальною вибіркою
    AssignFile(f, OpenDialog1.FileName);
    Reset(f);
    ParamsCount := 0;
    StringsCount := 0;

    //визначаємо кількість параметрів
    ReadLn(f, s);
    for i := 2 to length(s) do
    if (s[i] <> ' ') and (s[i-1] = ' ') then inc(ParamsCount);
    //задаємо к-сть параметрів
//    TEData.NrOfColumns := ParamsCount;
    TEData.NrOfColumns := 5;

    //визначаємо кількість строк даних
    while not eof(f) do
    begin
      ReadLn(f, s);
      inc(StringsCount);
    end;
    CloseFile(f);

    //задаємо к-сть строк даних
    if (StringsCount > 398) then StringsCount := 398;
    TEData.NrOfRows := StringsCount;

    NoteBk.PageIndex := 0;
    TEData.SuppressPaint := true;

    //зчитуємо навчальну вибірку у змінну f
    AssignFile(f, OpenDialog1.FileName);
    Reset(f);
    i:= 0;
    while not eof(f) do
    begin
      ReadLn(f, s);
      inc(i);

      //якщо не строка заголовку, тобто перша, читаємо дані
      if i > 1 then
      begin
        j := 0;
        //розбиваємо строку
        str := TStringList.create;
        str.text := stringReplace(s, ' ', #13#10, [rfReplaceAll]);
        for j := 0 to str.count-1 do
        begin
          val := str[j];

          //нормалізація даних

          //протокол
          if val = 'tcp' then val := '0';
          if val = 'udp' then val := '1';
          if val = 'icmp' then val := '2';
          //сервіс
          if val = 'auth' then val := '1';
          if val = 'bgp' then val := '2';
          if val = 'courier' then val := '3';
          if val = 'csnet_ns' then val := '4';
          if val = 'ctf' then val := '5';
          if val = 'daytime' then val := '6';
          if val = 'discard' then val := '7';
```

```
if val = 'domain' then val := '8';
if val = 'domain_u' then val := '9';
if val = 'echo' then val := '10';
if val = 'eco_i' then val := '11';
if val = 'ecr_i' then val := '12';
if val = 'efs' then val := '13';
if val = 'exec' then val := '14';
if val = 'finger' then val := '15';
if val = 'ftp' then val := '16';
if val = 'ftp_data' then val := '17';
if val = 'gopher' then val := '18';
if val = 'hostnames' then val := '19';
if val = 'http' then val := '20';
if val = 'http_443' then val := '21';
if val = 'imap4' then val := '22';
if val = 'IRC' then val := '23';
if val = 'iso_tsap' then val := '24';
if val = 'klogin' then val := '25';
if val = 'kshell' then val := '26';
if val = 'ldap' then val := '27';
if val = 'link' then val := '28';
if val = 'login' then val := '29';
if val = 'mtp' then val := '30';
if val = 'name' then val := '31';
if val = 'netbios_dgm' then val := '32';
if val = 'netbios_ns' then val := '33';
if val = 'netbios_ssn' then val := '34';
if val = 'netstat' then val := '35';
if val = 'nnsp' then val := '36';
if val = 'nntp' then val := '37';
if val = 'ntp_u' then val := '38';
if val = 'other' then val := '39';
if val = 'pm_dump' then val := '40';
if val = 'pop_2' then val := '41';
if val = 'pop_3' then val := '42';
if val = 'printer' then val := '43';
if val = 'private' then val := '44';
if val = 'red_i' then val := '45';
if val = 'remote_job' then val := '46';
if val = 'rje' then val := '47';
if val = 'service' then val := '48';
if val = 'shell' then val := '49';
if val = 'smtp' then val := '50';
if val = 'sql_net' then val := '51';
if val = 'ssh' then val := '52';
if val = 'sunrpc' then val := '53';
if val = 'supdup' then val := '54';
if val = 'systat' then val := '55';
if val = 'systat' then val := '56';
if val = 'telnet' then val := '57';
if val = 'tftp_u' then val := '58';
if val = 'tim_i' then val := '59';
if val = 'time' then val := '60';
if val = 'urh_i' then val := '61';
if val = 'urp_i' then val := '62';
if val = 'uucp' then val := '63';
if val = 'uucp_path' then val := '64';
if val = 'vmnet' then val := '65';
if val = 'whois' then val := '66';
if val = 'X11' then val := '67';
if val = 'Z39_50' then val := '68';
if val = 'https' then val := '69';
//прапор
if val = 'OTH' then val := '0';
if val = 'REJ' then val := '1';
if val = 'RSTO' then val := '2';
if val = 'RSTR' then val := '3';
if val = 'S0' then val := '4';
if val = 'SF' then val := '5';
if val = 'SH' then val := '6';
if val = 'S1' then val := '7';
//надпис
if val = 'back' then val := '0';
if val = 'buffer_overflow' then val := '1';
if val = 'ftp_write' then val := '2';
```

```
        if val = 'guess_passwd' then val := '3';
        if val = 'imap' then val := '4';
        if val = 'ipsweep' then val := '5';
        if val = 'neptune' then val := '6';
        if val = 'normal' then val := '7';
        if val = 'phf' then val := '8';
        if val = 'pod' then val := '9';
        if val = 'portsweep' then val := '10';
        if val = 'rootkit' then val := '11';
        if val = 'satan' then val := '12';
        if val = 'smurf' then val := '13';
        if val = 'teardrop' then val := '14';
        if val = 'warezclient' then val := '15';

        for k := 1 to Length(val) do
        begin
          if val[k] = '.' then val[k] := ',';
        end;

        lastval := val.ToDouble;

        //нульові та ненульові за тривалістю дані позначимо різними кольорами
        if str[0] = '0' then TEData.RowAttrib[i] := 1
        else TEData.RowAttrib[i] := 2;

        //пододаємо отриманий вектор даних на вхід мережі
        TEData.Data[j,i-1] := lastval;
      end;
      str.free;
    end;
  end;
  CloseFile(f);

  TEData.SuppressPaint := false;
  BButTrainit.Enabled := true;
  LabelProgress.Caption:= '0%';
 end;
end;

//процедура навчання мережі
procedure TMainForm.BButTrainItClick(Sender: TObject);
begin
 BButContTrain.Enabled := false;
 BButAbort.Enabled := true;
 NoteBk.PageIndex := 1;
 //передаємо дані на вхід мережі
 KohMap1.TrainData := TEData.Data.NumericData;
 //стандартизуємо дані
 KohMap1.StandardizeData;
 //створюємо архітектуру мережі
 KohMap1.SizeX := round(NIOxSize.Value);
 KohMap1.SizeY := round(NIOySize.Value);

 //параметри мережі

 //коефіцієнт налаштування
 KohMap1.InitialAlpha:= strtofloat(NLAlpha.Text);
 //кроки навчання
 KohMap1.NrOfTrnSteps:= strtoint(NLSteps.Text);
 //кількість сусідів
 KohMap1.InitialNeighbors := round(NIONeighb.Value);

 //запускаємо процес навчання мережі
 KohMap1.TrainIt;
 Show;
 //показуємо результат навчання
 ShowCurrentNetResponse;
 BButContTrain.Enabled := false;
 BButAbort.Enabled := false;
end;

//візуалізація результату навчання мережі
procedure TMainForm.ShowCurrentNetResponse;
var
 i,ix,iy: integer;
```

```
  dist: double;
begin
  //очищуємо граф
  RCKohMap.ClearGraf;
  RowVect.NrOfElem := TEData.NrOfColumns;
  //заповнюємо граф
  for i:=1 to TEData.NrOfRows do
    begin
    TEData.Data.NumericData.CopyRowToVec (RowVect,i,1,TEData.NrOfColumns);
    KohMap1.ApplyIt (RowVect, Ix, Iy, Dist);
    RCKohMap.DataColor := PredefinedColors[TEData.RowAttrib[i] mod 16];
    RCKohMap.MarkAt (ix+0.6*random-0.3, iy+0.6*random-0.3, 11);
    end;
  //виводимо граф
  RCKohMap.SetRange (1, 0, 0, KohMap1.SizeX+1,KohMap1.SizeY+1);
  RCKohMap.ShowGraf;
end;

//продовження навчання мережі
procedure TMainForm.BButContTrainClick(Sender: TObject);
begin
  Enabled := false;
  //передаємо дані мережі
  KohMap1.TrainData := TEData.Data.NumericData;
  //стандартизуємо дані
  KohMap1.StandardizeData;
  //продовжуємо навчання
  KohMap1.ContinueTraining;
  Enabled := true;
  //показуємо результат навчання
  ShowCurrentNetResponse;
  Show;
end;

//перервати процес навчання
procedure TMainForm.BButAbortClick(Sender: TObject);
begin
  BButContTrain.Enabled := true;
  KohMap1.AbortTraining;
end;

//процедура обробки зворотного зв'язку від мережі
procedure TMainForm.KohMap1Feedback(Sender: TObject; PercentDone: Integer);
begin
  //показуємо процент прогресу навчання мережі
  LabelProgress.Caption:= PercentDone.ToString+'%';
  //якщо відмічено пропорець "показати процес навчання"
  if CBShowTraining.Checked then
  //візуалізуємо прогрес навчання мережі
  ShowCurrentNetResponse;
  Application.ProcessMessages;
end;

end.
```