МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЮКАЛЬЧУК АНДРІЙ ІВАНОВИЧ

Допускається до захисту:
в. о. завідувача кафедри
прикладної математики та
кібербезпеки,
д – р філософії з математики,
_____ А.В. Луценко
«__»_____20__ р.

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ПОПЕРЕДЖЕННЯ ТА ВИЯВЛЕННЯ КІБЕРАТАК ТИПУ *USER TO ROOT* ТА *REMOTE TO LOCAL* НА ПІДПРИЄМСТВАХ КРИТИЧНОЇ ІНФРАСТРУКТУРИ

Спеціальність 125 Кібербезпека

Кваліфікаційна (бакалаврська) робота

Керівник:
канд. техн. наук, доцент,
доцент кафедри прикладної
математики та кібербезпеки,
Загоруйко Л. В.

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

Вінниця - 2024

АНОТАЦІЯ

Юкальчук А. І. Розробка програмного забезпечення для попередження та виявлення кібератак типу *user to root* та *remote to local* на підприємствах критичної інфраструктури. Спеціальність 125 «Кібербезпека». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній (бакалаврській) роботі досліджується модель нейронної мережі для попередження і виявлення кібератак типу *User to Root* (*U2R*) та *Remote to Local (R2L*) на підприємства критичної інфраструктури, а також її програмна реалізація мовою програмування *Python*.

Ключові слова: нейронна мережа, комп'ютерна атака, U2R, R2L, KDD99, багатошаровий персептрон, навчання нейромережі.

68 с., 2 табл., 16 рис., 1 дод., 48 джерел.

ABSTRACT

Yukalchuk A. I. Development of software for the prevention and detection of user to root and remote to local cyberattacks on critical infrastructure enterprises. Specialty 125 "Cybersecurity". Vasyl' Stus Donetsk National University, Vinnytsia, 2024.

The qualification (bachelor's) thesis investigates a neural network model for preventing and detecting User to Root (U2R) and Remote to Local (R2L) cyberattacks on critical infrastructure enterprises, as well as its implementation in the Python programming language.

Keywords: neural network, computer attack, U2R, R2L, KDD99, multilayer perceptron, neural network training.

68 pp., 2 tables, 16 figures, 1 appendix, 48 sources.

ЗМІСТ

ВСТУП

На сьогоднішній день комп'ютерні мережі мають важливе значення у сучасному житті, оскільки вони застосовуються у майже всіх аспектах людської діяльності. Незалежно від типу підприємства – комерційного, державного, муніципального чи бюджетного – його робота залежить від функціонування комп'ютерної мережі. Отже, забезпечення безпеки цих мереж є критичним, адже наслідки недостатнього захисту можуть бути різноманітними: крадіжка, знищення чи поширення конфіденційної інформації (комерційної таємниці), персональних даних, підміна інформації, блокування доступу до неї, обмеження функціональності або повна зупинка роботи мережі. Останнє може фактично паралізувати бізнес-процеси, що призводить до значних збитків. Одним з методів вирішення цієї проблеми є створення інтелектуальних систем виявлення вторгнень, але їх ефективність можлива лише за наявності якісного набору даних.

У цих умовах особливо важливими є питання оцінки ефективності систем безпеки значущих об'єктів критичної інфраструктури. Тому в ході проектування системи безпеки значущого об'єкта критичної інфраструктури з метою тестування рекомендовано її макетування або створення тестового середовища з використанням засобів та методів моделювання виявлення кібер-вторгнень різних класів та типів.

Актуальність: в умовах постійного зростання ефективності кібератак особливо важливими є питання оцінки ефективності систем безпеки значущих об'єктів критичної інфраструктури.

Мета дослідження: аналіз сучасних моделей нейронних мереж для попередження та виявлення кібератак. Реалізація системи виявлення кібератак типу *User to Root* (*U2R*) та *Remote to Local (R2L*).

Практична значимість: результати роботи можна використовувати при проектуванні та тестуванні систем виявлення вторгнень на об'єкти критичної інфраструктури.

Об’єкт дослідження – нейромережі, атаки типу U2R та R2L, датасет KDD99 .

Предмет дослідженя – методи боротьби з кібератаками, система кібербезпеки для підприємств критичної інфраструктури, спрямована на запобігання та виявлення кібератак.

Апробація результатів дослідження – було представлено на підсумковій науково-практичній конференції ІІ туру всеукраїнського конкурсу студентських наукових робіт зі штучного інтелекту 05.06.2024 на базі КПІ ім. Ігоря Сікорського.

РОЗДІЛ 1. АНАЛІЗ СУЧАСНИХ ДОСЛІДЖЕНЬ СИСТЕМ ВИЯВЛЕННЯ ВТОРГНЕНЬ

1.1. Аналіз сучасних моделей нейронних мереж для попередження та виявлення кібератак

В літературі існує ряд робіт, пов'язаних з виявленням вторгнень [1-8]. Деякі з цих робіт [1,6] стосуються загальної класифікації пакетів на нормальні або атакуючі категорії, в той час як інші стосуються виявлення конкретних категорій атак, таких як атаки типу *Remote to Local User* та *User to Root* [2, 4, 7, 8].

Sagrirogu та ін. запропонували та розробили інтелектуальну систему виявлення вторгнень (*IDS*) [1]. Автори представили різні модефікації *IDS* на наборі даних *KDD'99* з використанням штучних нейронних мереж (*ANN*). Розроблені моделі *IDS* виконували завдання виявлення атак з високою точністю від 81.93% до 97.92%.

Kumar і Selvakumar представили алгоритм під назвою *NFBoost*, що використовує гібридні нейронечіткі системи для виявлення розподілених атак на відмову в обслуговуванні (*DDoS*) [2]. Автори використовували загальнодоступні набори даних, такі як *KDD*, *UCI* тощо, для навчання та тестування. В алгоритмі використовувались лише звичайні записи з'єднань та записи з'єднань під час атак на відмову в обслуговуванні (*DoS*) з набору даних *KDD*. Розроблена на основі алгоритму система досягла високої точності виявлення атак понад 98% як у навчанні, так і в тестуванні на наборі даних *KDD*.

Wu та Yen представили дослідження для порівняння таких оціночних показників, як точність, частота виявлення та частота хибних тривог на наборі даних *KDD'99*, але оскільки ці дані не є нормалізованими для порівняння в навчанні та тестуванні, то автори використовують різні розподіли даних [3].

Дослідження показало кращі результати, ніж *KDD* "Тріумфатора", особливо в порівнянні з атаками "*User to Root*" (*U2R*) та "*R2L*".

Gupta та ін. запропонували багаторівневий метод виявлення вторгнень з використанням умовних випадкових полів (*CRF*) [4]. Метод використовує набір даних *KDD'99* і застосовує послідовне виявлення атак у багаторівневий спосіб. Згідно з експериментальними результатами, запропонований метод, працює краще, ніж дерева рішень та наївний байєс. Він забезпечує значне покращення точності виявлення, особливо для атак *U2R* та *R2L*.

Kuang та ін. запропонували новий підхід, заснований на опорних векторах (*SVM*), який поєднує аналіз головних компонент ядра (*KPCA*) та генетичний алгоритм (*GA*) для виявлення вторгнень [5]. Модель реалізує багатошаровий *SVM*-класифікатор для прогнозування того, чи є дія атакою. В експериментах запропонований *SVM*-класифікатор досягає кращої точності та продуктивності при виявленні вторгнень.

Mukhopadhyay та ін. представили новий метод системи виявлення вторгнень (*IDS*) з використанням нейронної мережі зі зворотним поширенням (*BPN*) [6]. Автори проводили експерименти на наборі даних *KDD'99*, і метод показав 95,6% та 73,9% успішності для двох рівнів тестування відповідно.

Subbulakshmi та Afroze запропонували багаторівневий підхід до виявлення атак з використанням кореляційного вибору ознак (*CFS*) на наборі даних *KDD'99* [7]. Запропонований метод дозволив досягти коефіцієнтів класифікації для *R2L*-атак від 91% до 99%.

Sharma та Mukherjee в своїй роботі представили багаторівневий підхід, що поєднує методи наївного байєсівського класифікатора (*NBC*) та наївного байєсівського дерева (*NBTree*), зокрема, для підвищення точності та швидкості виявлення атак типу *U2R* та *R2L*, не знижуючи при цьому ефективність виявлення інших атак [8]. Автори використовували записи з'єднань з набору даних *KDD'99* для навчання і тестування, який містить 24 різних типи атак.

У роботі M. H. Kamarudin, C. Maple, T. Watson та H. Sofian була проведена розробка і аналіз алгоритму для виявлення атак на мережевий трафік з

використанням пакетних заголовків[12]. Автори використали LbAD модель, яка виявилася дуже перспективною для використання в системах виявлення аномалій. Ця модель використовує статистичний аналіз значень полів пакетних заголовків на трьох рівнях OSI 7 шарів (рівень з'єднання даних, мережевий рівень, транспортний рівень). Був використаний алгоритм бінарної логістичної регресії для виявлення кореляції між різними шарами пакетних заголовків. Це допомогло виявити, які саме шари є найбільш важливими для виявлення двох типів атак - R2L та U2R. Виявлено, що для атак типу R2L найбільш важливими є шари 3 та 4, тоді як для атак типу U2R - шари 2 та 4. Це дозволило авторам розробити модель, яка враховує лише найбільш важливі шари, що скорочує час обробки. Проведені експерименти показали високу ефективність запропонованого підходу. Виявлено, що модель досягла дуже високого рівня виявлення обох типів атак: 84% для R2L та 93.5% для U2R. Порівняно з існуючими алгоритмами, цей підхід показав значні покращення виявлення атак.

В роботі Shanmugavadivu експериментується система виявлення мережевих вторгнень з використанням нечіткої логіки[13]. Ця методика використовує набір нечітких правил, які отримані з чітких правил з використанням частих елементів. Точність класифікації цього підходу перевищує 90% для всіх типів атак.

P.Giffy Jeya, M Ravichandran та C.S. Ravichandran [14] представили новий метод класифікації аномалій, використовуючи кореляційний аналіз для зменшення кількості ознак та застосовуючи лінійний дискримінантний аналіз Фішера (FLDA) для виявлення даних про вторгнення. В експериментах використовувався набір даних KDD Cup'99, і результати експериментів показали, що запропонований метод покращив точність класифікації U2R і R2L атак порівняно з іншими дослідженнями.

У статті Ployphan Sornsuwit використовувався алгоритм Adaboost для створення ансамблю класифікаторів дерева рішень, наївного Байєса, SVM та MLP для виявлення U2R та R2L атак, які важко виявити[15]. Крім того, для зменшення надлишкових ознак використовується кореляційний метод.

Ефективність ансамблевих класифікаторів оцінюється за допомогою наборів даних KDD CUP'99 в репозиторії даних UCI Repository, а результати порівнюються з одиночними класифікаторами. Експерименти проведено на наборі даних з усіма ознаками та деякими вибраними ознаками. Експерименти показують, що ансамблеві класифікатори на основі Adaboost можуть покращити продуктивність усіх класифікаторів. Крім того, ансамбль Naïve Bayes та MLP мають найвищу чутливість, а ансамбль Naïve Bayes має найвищу специфічність, коли вони тестуються на даних з деякими вибраними ознаками. Однак дерево рішень показує найнижчі результати, оскільки обидва типи атак містять невеликий обсяг даних і мають тривалий період часу. Атаки U2R та R2L більш схожі на поведінку звичайних користувачів, а кількість даних дуже мала.

Debojit Boro, Bernard Nongpoh та Dhruba K. Bhattacharyya [16] представили комбінацію декількох моделей для покращення продуктивності систем виявлення вторгнень. Вони використовували чотири шари для об'єднання відповідних класифікаторів за допомогою декількох слабких учнів. Експериментальні результати показали, що запропонований ними метод дозволив покращити швидкість класифікації та частоту помилкових спрацьовувань порівняно з одним класифікатором.

Дослідження Beghdad аналізує продуктивність деякої нейронної мережі коли для навчання використовується весь набір даних KDD з метою класифікації та виявлення атак [18]. П'ять типів нейронних мереж, що досліджуються нейронних мереж, які досліджуються: Багатошарове сприйняття (MLP), самоорганізуюча карта ознак (SOM), нейронні мережі Джордана/Елмана (Jordan/Elman), рекурентні нейронні мережі (RNN) нейронні мережі Джордана/Елмана, рекурентні нейронні мережі та нейронної мережі RBF. Їх результати показали, що відсоток правильної класифікації (PCC) становить 99.16%, 98.28%, 98.36%, 98.44% та 79,23% відповідно.

У статті Kumar і Yadav запропоновано систему виявлення вторгнень на основі штучної нейронної мережі, яка використовує для навчання алгоритм

градієнтного спуску з імпульсним поширенням[17]. Хоча для навчання обрано випадкові патерни, запропонована нейронна мережа протестована на повному "випробувальному" наборі даних KDD cup 99. Результати показують, що точність запропонованої IDS на основі нейронної мережі для бінарної класифікації (атака або нормальний стан) є високою, а рівень виявлення атак зондування, R2L та U2R є високим у порівнянні з іншими методами.

У роботі Mehmet Ali було розроблено та реалізовано систему виявлення R2L-атак на основі штучного інтелекту у вигляді бінарного класифікатора, який вирішує, чи є запис про з'єднання частиною R2L-атаки, чи ні. Система використовує набір даних KDD'99, зібраний для оцінки моделей IDS з точки зору частоти виявлення та частоти хибних спрацьовувань [19]. Модуль обробки даних системи видаляє надлишкові записи та налаштовує розподіл записів у навчальному наборі даних, який зазвичай є незбалансованим. Експериментальні результати показують, що MLPNN моделі з добре налаштованими параметрами можуть досягти дуже високого рівня виявлення R2L-атак вище %96 за рахунок усунення надлишкових записів та збалансування розподілу даних на навчальних наборах даних KDD'99.

1.2. Аналіз систем виявлення вторгнень (IDS)

Процес вивчення подій [35], [37]-[39], що відбулися в комп'ютерній системі або мережевих ресурсах, з подальшим їх аналізом на предмет виявлення ознак вторгнення та ймовірних інцидентів, які можуть спричинити загрози заходам безпеки, називається Виявленням вторгнень (Intrusion Detection). Вторгнення зазвичай спричиняються зловмисниками, які хочуть отримати несанкціоновані та додаткові привілеї до певної системи або мережі для власних цілей [35].

Система виявлення вторгнень (Intrusion Detection System, IDS) визначається як програмний або апаратний продукт, який фокусує та ідентифікує ймовірні інциденти, спричинені зловмисниками, відстежує

інформацію про ці вторгнення, намагається їх припинити та створює звіт для адміністраторів безпеки [36] в режимі реального часу. Таким чином, систему виявлення вторгнень можна розглядати як операцію безпеки, що доповнює захист, наприклад, брандмауери [41]. Вона також допомагає забезпечити безпеку та запобігти різноманітним вторгненням, спричиненим зловмисниками.

Функції виявлення вторгнень такі:

- Спостереження та моніторинг: Використовується для спостереження та моніторингу активності користувачів, мережі та системи на предмет будь-яких підозрілих подій [41].

- Розпізнавання шаблонів: Має здатність розпізнавати шаблони атак [41].

- Звіти про вторгнення: Підготувати детальний звіт про захоплені події. Потім системні адміністратори використовують ці звіти для аналізу шаблонів аномальної активності, конфігурацій системи та налаштувань безпеки для визначення вразливостей [41].

- Відстеження порушень користувацької політики: Використовується для відстеження порушень політики користувача, оцінки цілісності системи та файлів [41].

- Логування подій: При виявленні підозрілої активності IDS записує інформацію, пов'язану з цією активністю [41].

- Сповіщення системних адміністраторів: IDS надсилає сповіщення системному адміністратору через веб-сторінки, електронні листи, повідомлення тощо, коли відбувається будь-яка підозріла подія в базі даних.

1.3. Підходи системи виявлення вторгнень

Виявлення зловживань

Виявлення зловживань також називають виявленням на основі сигнатур або правил [43]. У цьому підході до виявлення дії користувача порівнюються з відомою поведінкою зловмисників для проникнення в систему або мережу. При виявленні зловживань зібрана інформація аналізується і порівнюється з великими базами даних для пошуку сигнатур атак. Виявлення зловживань або виявлення на основі сигнатур є корисним, оскільки його рівень виявлення є високим, а рівень хибних тривог низьким для відомих атак [43].

Виявлення аномалій

У підході виявлення аномалій ідентифікуються дії, які відрізняються від вже встановлених шаблонів для користувачів або груп користувачів [45]. У цьому підході можуть бути встановлені профілі для нормальної поведінки користувачів, які випливають зі статистики даних користувачів. При виявленні, профіль порівнюється з фактичними даними користувачів. Якщо порогове значення вище або більше, ніж зміщення, поведінка користувача вважається нормальною, і вважається, що він не має наміру атакувати. Якщо ж порогове значення менше, ніж зміщення, поведінка користувача вважається ненормальною, і атака може відбутися. Це означає, що він будує базову лінію того, що є нормальним. Нормальна поведінка мережі повинна бути відома до її реалізації. Виявлення аномалій може легко виявити невідомі атаки, але рівень помилкових оцінок є високим [43]. Він також може виявляти попередні невідомі загрози [41].

1.4. Класифікація систем виявлення вторгнень

Мережеві IDS (NIDS)

Мережеві IDS - це автономні апаратні пристрої, які включають в себе функції виявлення мережевих вторгнень [46]. Вони здебільшого розгортаються

в стратегічних точках мережевої інфраструктури [41], наприклад, на кордоні між мережами, серверами віртуальних приватних мереж, серверами віддаленого доступу та бездротовими мережами [47]. NIDS відстежує мережевий трафік, що проходить через певні сегменти мережі або пристрої. Він може перехоплювати та аналізувати дані для виявлення відомих атак або незаконної діяльності або аналізувати активність мережевих та прикладних протоколів для виявлення аномальної та підозрілої активності шляхом сканування трафіку [47]. NIDS також можна назвати "пакетними перехоплювачами", оскільки вони перехоплюють і збирають дані у вигляді інтернет-пакетів, що проходять через середовища зв'язку [43].

IDS на основі хосту (HIDS)

У IDS на основі хосту відстежуються характеристики одного хоста, а також події, що відбуваються на цьому хості, на предмет виявлення будь-якої зловмисної активності. Вони можуть відстежувати мережевий трафік, журнали, процеси, операції, що виконуються додатками, доступ до файлів і їх модифікацію, а також будь-які зміни конфігурації системи [43-47]. Розгортання HIDS зазвичай відбувається на критичних хостах. До критичних хостів відносяться сервери або системи, які є загальнодоступними і містять певну конфіденційну інформацію. Вони розміщуються на одному сервері або робочій станції, де збираються дані з різних ресурсів і проводиться машинний аналіз даних локально. У гібридних IDS одночасно використовуються як на основі хосту, так і мережеві IDS [41,47].

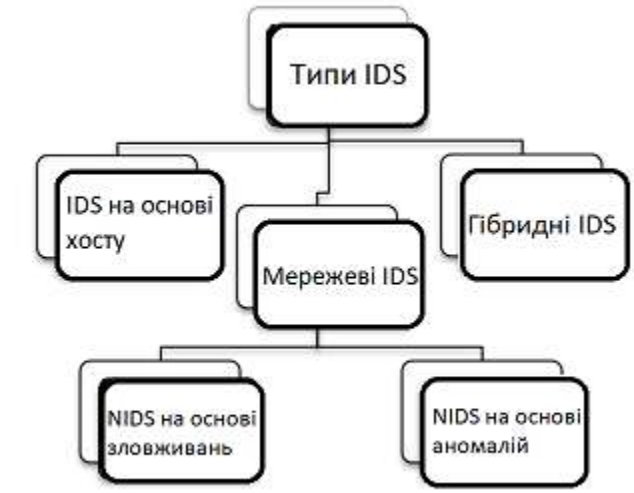

Рисунок - 1.1 Класифікація IDS[48]

1.5. Методи виявлення вторгнень

Штучні нейронні мережі

Штучні нейронні мережі забезпечують гнучкі можливості розпізнавання образів. У ШНМ система проходить спеціальне навчання, щоб вона могла розпізнавати різні довільні шаблони, які надаються їй як вхідні дані. Коли система повністю розпізнає ці шаблони, її просять зіставити ці шаблони з вихідними даними. Шляхом зіставлення різних вхідних і вихідних довільних шаблонів визначається, чи відбулося вторгнення, чи ні [40,47].

Таблиці переходів станів

У таблиці переходів станів послідовність дій, виконаних зловмисником, описується у вигляді діаграми переходів станів і спостерігається поведінка системи. Коли вона збігається з ідентифікованим скомпрометованим станом і станом проникнення, виявляється вторгнення [42].

Генетичні алгоритми (GAs)

Функція генетичних алгоритмів (GAs) полягає в імітації або відтворенні природної системи розмноження в природі. Після рекомбінації та різних випадкових змін у наступних поколіннях відтворюється лише найбільш пристосована особина. У 1995 році в дослідженнях IDS з'явилося застосування

ГА. Воно передбачає еволюцію сигнатури, яка вказує на вторгнення. Система класифікаторів, що навчаються (Learning Classifier System, LCS) є спорідненою технікою, в якій розробляються бінарні правила, що розпізнають шаблони вторгнення [47].

Байєсівська мережа

У мережі Байєса були введені графічні моделі. Ці графічні моделі визначаються набором правил переходів, представлених у вигляді ймовірнісних залежностей. У цій моделі в кожному вузлі описується таблиця умовних ймовірностей і стан випадкових величин. Таблиця умовних ймовірностей визначає ймовірності перебування вузла в тому чи іншому стані, враховуючи стан його батька. Цей підхід може працювати з неповними даними [47].

Нечітка логіка

Нечітка логіка призначена для обробки нечітких і неточних даних. Щоб вказати на втручання, зв'язок між вхідними та вихідними змінними визначається шляхом створення різних наборів правил. Для дослідження інтенсивності правдивості використовуються функції належності [47]. Усі вищезгадані методи узагальнено в таблиці 1.1

Таблиця 1.1- Методи IDS

Методи	**Функції**
Штучні нейромережі	Система навчається шляхом введення відповідних вхідних/вихідних даних. Це навчання потім використовується для розпізнавання довільних шаблонів, що подаються на вхід системи.
Таблиці переходу станів	Вторгнення виявляється шляхом порівняння поведінки системи з діаграмою переходу станів зловмисника.
Генетичні алгоритми (GAs)	Імітація природної системи відтворення в природі, де після певних змін у наступних поколіннях відтворюються лише найбільш пристосовані

	особини в поколінні.
Байєсівська мережа	Вводяться графічні моделі, які мають справу з неповними даними.
Нечітка логіка	Працюють з нечіткістю та неточністю.

1.6. Висновки до розділу 1

Здійснений аналіз сучасних моделей нейронних мереж та систем виявлення вторгнень (IDS) виклав основні підходи, методи та класифікації, які застосовуються для виявлення втручань у комп'ютерні системи. В ході дослідження було виявлено, що сучасні моделі IDS на основі нейронних мереж виявляються досить ефективними у виявленні та прогнозуванні кібератак. Вони можуть виявляти відхилення від звичайного зловмисної активності, що робить їх корисним інструментом для захисту мереж. Щодо систем виявлення вторгнень, було проведено детальний аналіз їх підходів, класифікації та методів виявлення. Це дозволило зрозуміти, як системи виявлення вторгнень працюють на практиці та які основні принципи лежать в їхній основі.

РОЗДІЛ 2. ОСНОВІНІ ПОНЯТТЯ ПРО НЕЙРОННІ МЕРЕЖІ. КЛАСИФІКАЦІЯ ТА ОСНОВНІ ВИДИ

2.1. Поняття нейронної мережі

Штучні нейронні мережі - це відносно грубі електронні моделі, засновані на нейронній структурі мозку. Мозок в основному вчиться на власному досвіді. Коли ми говоримо про нейронну мережу, ми повинні говорити "штучна нейронна мережа". Нейронна мережа - це комп'ютери, архітектура яких змодельована за зразком мозку. Зазвичай вони складаються з сотень простих процесорів, які об'єднані в складну комунікаційну мережу. Кожен блок або вузол є спрощеною моделлю реального нейрона, який посилає новий сигнал або спрацьовує, якщо отримує достатньо сильний вхідний сигнал від інших вузлів, до яких він підключений. Традиційно нейронною мережею називали мережу або ланцюг біологічних нейронів, але сучасне використання цього терміну часто відноситься до штучних нейронних мереж. Це математична модель або обчислювальна модель, парадигма обробки інформації, тобто натхненна тим, як працює біологічна нервова система, наприклад, інформаційна система мозку[22].

Нейронна мережа складається зі з'єднаних між собою штучних нейронів, які запрограмовані таким чином, щоб імітувати властивості одного або декількох біологічних нейронів. Ці нейрони працюють в унісон для вирішення конкретних завдань. Нейронна мережа налаштована на вирішення завдань штучного інтелекту без створення моделі реальної біологічної системи. Вони використовується для розпізнавання мови, аналізу зображень, адаптивного управління тощо. Ці програми виконуються через процес навчання, подібно до навчання в біологічній системі, який включає в себе налагодження зв'язків між нейронами через синаптичний зв'язок. Те саме відбувається і в штучній нейронній мережі. На рис. 2.1 показано просту нейронну мережу.

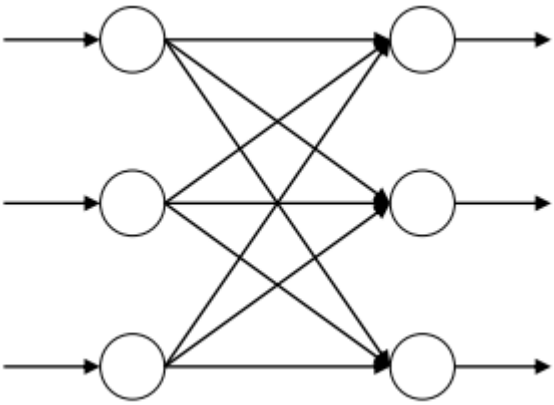

Рисунок 2.1- Приклад нейронної мережі [28]

2.2. Історія нейронних мереж

Перший крок до штучних нейронних мереж був зроблений у 1943 році, коли нейрофізіолог Уоррен Маккалох та молодий математик Волтер Піттс розробили перші моделі нейронних мереж. Вони опублікували статтю з назвою "Логічна обробка ідей, властивих нервовій діяльності" про те, як можуть працювати нейрони. Їхні мережі базувалися на простих елементах, які вважалися бінарними пристроями з фіксованими порогами. Результатом їхньої моделі були прості логічні функції з бінарним характером нервової активності. У 1944 році Йозеф Ерлангер разом з Гербертом Спенсером Гассером ідентифікували кілька різновидів нервових волокон і встановили зв'язок між швидкістю потенціалу дії та діаметром волокна. У 1949 році психолог Гебб написав " Управлінням поведінкою", роботу, У висвітленні цього факту було зазначено, що нейронні зв'язки зміцнюються з кожним повторенням, коли вони використовуються, і ця концепція є фундаментально важливою для того, як людина навчається [23].

У 1958 році психолог Розенблат (Rosenblatt) провів ранню роботу над перцептронами. Перцептрон був електронним пристроєм, який був сконструйований відповідно до біологічних принципів і демонстрував здатність до навчання. Він також написав ранню книгу про нейрокомп'ютери "Основи нейродинаміки". Іншою системою була ADALINE (ADAptive LInear Element),

яка була розроблена в 1960 році двома інженерами-електриками Відроу і Хоффом. Метод, який використовувався для навчання, відрізнявся від методу Персептрона, він використовував правило навчання за методом найменших середніх квадратів (Least-MeanSquares). У 1962 році Відроу та Хофф розробили метод навчання, який оцінює вагу до того, як вона буде виправлена.

Після початкового періоду ентузіазму ця галузь пережила період розчарування і ганьби. У 1969 році Мінський і Паперт випустили книгу "Перцептрони: Вступ до обчислювальної геометрії", що була частиною ініціативи, спрямованої на критику досліджень нейронних мереж через їхні фундаментальні проблеми. У книзі вони узагальнили обмеження одношарового перцептрона, незважаючи на те, що було відомо про існування потужних перцептронів з кількома шарами, а також базових перцептронів Розенблата з трьома шарами. Вони визначили перцептрон як двошарову систему, яка може розв'язувати лише лінійно роздільні задачі і, наприклад, не може впоратися з проблемою виключного АБО. У зв'язку з тим, що зацікавленість суспільства та фінансування досліджень стали невеликими, лише декілька науковців продовжили роботу над темами, такими як розпізнавання образів. Проте в цей період з'явилося кілька парадигм, які дослідники активно розвивають й досліджують у сучасних роботах.

У 1972 році Клопф розробив основи для навчання в штучних нейронах, які були засновані на біологічних принципах. У 1974 році Пол Вербос створив метод навчання зі зворотнім поширенням, проте його важливість не була повністю зрозумілою до 1986 року. Фукусіма розробив багатошарову нейронну мережу з поетапним навчанням для інтерпретації рукописних символів. Оригінальна робота "Cognitron: Багатошарова нейронна мережа, що самоорганізується", була опублікована в 1975 році. У 1976 році Гроссберг у статті "Adaptive pattern classification and universal recoding" представив адаптивний резонанс як теорію когнітивної обробки інформації людиною.

У 1980-х роках сталося кілька подій, які викликали новий інтерес у галузі штучних нейронних мереж. Однією з них був внесок Кохонена, який

представив штучну нейронну мережу, часто відому як "карта Кохонена" або "мережа Кохонена". У той же період Хопфілд з Каліфорнійського технологічного інституту представив свою роботу "Нейронні мережі та фізичні системи з емерджентними колективними обчислювальними можливостями". У своїй книзі Хопфілд описав рекурентну штучну нейронну мережу, яка функціонує як система пам'яті з адресованим вмістом. Його роботи переконали сотні висококваліфікованих вчених, математиків і технологів приєднатися до нової галузі нейронних мереж.

У 1986 році алгоритм зворотного поширення, який спочатку був відкритий Вербосом у 1974 році, був відновлений та активно застосовувався в книзі "Навчання внутрішньої репрезентації шляхом поширення помилки" Румельхарта та його колег. Цей метод, що базується на градієнтному спуску, став ефективним інструментом для корекції помилок у штучних нейронних мережах. У 1985 році Американський інститут фізики ініціював подію, яка згодом стала регулярною - конференцію "Нейронні мережі для обчислень". У 1987 році в Сан-Дієго відбулася перша відкрита конференція з нейронних мереж - IEEE International Conference on Neural Networks, і було засновано Міжнародне товариство нейронних мереж (INNS). Також у 1988 році був створений журнал INNS "Нейронні мережі", який став важливим форумом для обміну новими дослідженнями в цій області, у 1989 році - "Нейронні обчислення", а в 1990 році - "IEEE Transactions on Neural Networks". У 1987 році Карпентер і Гроссберг у статті A massively parallel architecture for a selforganizing neural pattern recognition machine описали ART1 - модель неконтрольованого навчання, спеціально розроблену для розпізнавання бінарних образів.

У галузі нейронних мереж досягнуто значного прогресу - достатнього для того, щоб привернути велику увагу та профінансувати подальші дослідження. Сьогодні нейромережі обговорюються повсюдно. Здається, що прогрес за межами поточних комерційних застосувань можливий, і дослідження просувають цю сферу на багатьох фронтах. З'являються мікросхеми, засновані

на нейронній теорії, і розвиваються додатки для вирішення складних проблем. Очевидно, що сьогодні для нейромережевих технологій настав перехідний період. Між 2009 і 2012 роками в дослідницькій групі Шмідхубера були розроблені рекурентні нейронні мережі та нейронні мережі з глибоким прямим поширенням (deep feedforward).

У 2014 році вчені з IBM представили процесор (TrueNorth), архітектура якого подібна до тієї, що існує в мозку. IBM представила інтегральну схему розміром з поштову марку, здатну імітувати роботу мільйонів нейронів і 256 мільйонів синапсів у режимі реального часу. Система здатна виконувати від 46 до 400 мільярдів синаптичних операцій в секунду.

2.3. Класифікація нейромереж

Загалом, нейронна мережа — це сукупність нейронів, з’єднаних певним чином. Основні відмінності між моделями нейронних мереж полягають у способах спілкування нейронів між собою, функціях нейроелементів, а також механізмах і напрямках поширення сигналу в мережі. Характеристики нейроноподібних елементів та архітектура мережі визначають конкретний тип перетворення інформації, який здійснюється штучними нейронними мережами. Це включає в себе такі аспекти, як топологія міжнейронних зв'язків, виділення певних підмножин нейроноподібних елементів для введення та виведення інформації, методи навчання мережі, наявність або відсутність конкуренції між нейронами, а також напрямок і способи контролю та синхронізації передачі інформації між нейронами [24].

Нейронні мережі можна класифікувати за такими параметрами [24 - 28]:

Залежно від типу вхідної інформації, можна виділити два основних типи штучних нейронних мереж: аналогові та двійкові. Аналогові мережі працюють з інформацією у вигляді дійсних чисел, тоді як двійкові (виконавчі) мережі оперують інформацією, представленою у двійковій формі;

Щодо типу функції активації нейронів, можна виділити три основні типи мереж: безперервні, дискретні і дискретно-неперервні. Безперервні мережі, також відомі як аналогові, реальні або диференційовані мережі, мають кожен елемент, що реалізує безперервну диференційовану функцію. Дискретні мережі, відомі як бінарні, порогові або недиференційовані, мають кожен елемент, що реалізує недиференційовану функцію. Дискретно-неперервні мережі містять елементи з обома видами функцій активації - диференційованою та недиференційованою;

В залежності від типу структур нейронів, можна виділити два основних типи штучних нейронних мереж: гомогенні і гетерогенні. Гомогенні мережі, також відомі як однорідні мережі, складаються з нейронів одного типу з єдиною функцією активації. Гетерогенні мережі містять нейрони з різними функціями активації;

Залежно від типу графа міжнейронних зв'язків, можна виділити два основних типи мереж: мережі без циклів, такі як ациклічні мережі, і мережі з циклами. Мережі з циклами поділяються на збалансовані мережі з циклами та мережі з обмеженими циклами;

В залежності від кількості шарів нейронів, виділяють два основних типи штучних нейронних мереж: одношарові та багатошарові. Одношарові нейронні мережі містять лише один шар нейронів. Багатошарові нейронні мережі (БНМ) мають більше одного шару взаємопов'язаних нейронів;

Залежно від принципу синтезу існують два основних типи нейронних мереж: навчальні мережі і побудовані мережі. Навчальні мережі формуються за допомогою методів навчання, де граф міжнейронних зв'язків і ваги вхідних нейронів змінюються під час процесу навчання. У побудованих мережах кількість і тип нейронів, граф міжнейронних зв'язків і ваги нейронних входів визначаються під час створення мережі відповідно до конкретної проблеми, яку потрібно вирішити;

Існують кілька типів нейронних мереж, розділених за топологією зв'язків. Повнозв'язані (інтерактивні) мережі. Кожен нейрон в таких мережах пов'язаний

з кожним іншим нейроном "один до одного". Вони можуть мати всі або деякі вихідні сигнали нейронів як вхідні сигнали після кількох циклів роботи. Слабозв'язані (шарові, багатошарові, ієрархічні) мережі. Такі мережі зазвичай організовані у шари, кожен з яких може мати свою структуру зв'язків. Ці мережі можуть мати латеральні (бічні) зв'язки між нейронами одного шару та проекційні (аферентні) з'єднувальні шари. Вони використовуються для більш високорівневої обробки інформації. Ієрархічно інтерактивні мережі. Шари цих мереж з'єднані двосторонніми зв'язками, і вони мають інтерактивну структуру, але з різною кількістю зв'язків між елементами одного шару. Слабозв'язані (з локальними зв'язками) мережі**.** Нейрони у таких мережах розташовані у вузлах прямокутної або гексагональної решітки, і кожен нейрон зв'язаний лише з обмеженою кількістю своїх найближчих сусідів;

Є дві основні категорії мереж в залежності від характеру зв'язків: немонотонні мережі, де важко передбачити, як зміна будь-якого внутрішнього параметра вплине на вихідний сигнал, і монотонні мережі, де кожен шар розділяється на збудливий і гальмівний блоки, а зв'язки влаштовані таким чином, що збудливі елементи впливають на інші збудливі елементи того ж шару, а гальмівні — на гальмівні наступного шару. У монотонних мережах важлива монотонна залежність вихідного сигналу від вхідних параметрів. Також, для мереж з сигмоподібними елементами важливо, щоб ваги всіх зв'язків були невід'ємними;

Існують два типи мереж залежно від характеру зв'язків: мережі прямого розповсюдження, де сигнал проходить лише в одному напрямку, і рекурентні мережі, які мають зворотній зв'язок і характеризуються як прямим, так і зворотнім поширенням інформації між шарами. Серед рекурентних мереж можна виділити релаксаційні, де інформація циркулює до стану рівноваги, та шаруваті мережі, де відсутній процес релаксації. Також, серед багаторівневих мереж існують багаторівневі циклічні мережі, де шари замкнуті в кільце, багатошарові повнозв’язані мережі, де кожен рівень має повнозв’язану структуру, і повністю з’єднані-шарові мережі, що не розділяють фази обміну

всередині шару та передачі наступного шару. Також варто відзначити рециркуляційні мережі, які самоорганізуються без викладача, та мережі з бічним зворотним зв'язком;

Залежно від типу методу навчання, мережі можуть бути розділені на кілька категорій. Перша категорія - мережі з динамічними зв'язками та ітераційним навчанням, які базуються на принципі корекції помилок. Ці мережі використовують метод зворотного розповсюдження помилки, який має перевагу асимптотичної збіжності навчання, але може вимагати багаторазового повторення ітерацій навчання на всьому об'ємі даних, що запам'ятовуються, що може вимагати значних часових та обчислювальних ресурсів. Друга категорія - мережі з фіксованими з'єднаннями. Ці мережі не використовують ітераційного навчання, а замість цього вони використовують методи прямого обчислення значень матриці зв'язків, що сприяє швидкості навчання. Однак вони мають недоліки, такі як надмірна кількість нейронів і низький обсяг запам'ятовуваної інформації, тому їх застосування досі обмежено, переважно в дослідницьких проектах;

Існують різні типи навчання у мережах залежно від характеру його контролю. Серед них можна виділити мережі з супервізованим навчанням, коли наявний викладач або супервізор, який порівнює отримані значення з відомими, і мережі з неконтрольованим навчанням, де модель навчається без точно відомих вихідних даних, але може групувати подібні вхідні вектори для формування однакового виходу. Також існують мережі зі змішаним навчанням, які поєднують в собі як спостережувані, так і неспостережувані методи, використовуючи при цьому приклади з наборів вхідних даних разом з відповідними результатами;

Залежно від методів навчання, мережі можуть бути класифіковані як ті, що використовують безітераційний підхід, метод зворотного поширення помилки, конкурентне навчання, правило Хебба або гібридні методи, які комбінують різні підходи до навчання;

Залежно від типу часу функціонування, мережі можуть бути класифіковані як мережі з безперервним часом (також відомі як аналогові мережі), мережі з дискретним асинхронним часом (де стан може змінюватись лише одного нейрона в кожний момент часу), та мережі з дискретним синхронним часом, де зміна стану відбувається одночасно для всієї групи нейронів;

В залежності від урахування попереднього стану мережі можна виділити кілька типів мереж. Спочатку, це статичні мережі, які не мають контурів зворотного зв'язку або динамічних елементів. Вихід цих мереж залежить лише від введеного набору і не залежить від попередніх станів мережі. Далі, є нечіткі мережі, які поєднують нейронні мережі і нечіткі системи. Перший шар нейронів виконує фазу фазифікації, другий рівень містить нечіткі правила, а третій рівень відповідає за дефазифікацію, тобто введення ясності. Також можуть існувати нетрадиційні нейронні мережі, які використовують нетрадиційні підходи до обробки інформації або структури;

Мережі відповідно до завдань можуть бути розроблені для наступних цілей: обробки і фільтрації даних, категоризації і класифікації даних, виявлення закономірностей у наборах даних, заповнення прогалин у таблицях даних, візуалізації і представлення даних, розпізнавання (класифікації) зображень, непараметрична апроксимація зв'язків на основі дискретних даних, створення великої бази даних зі швидким асоціативним пошуком інформації за неповними вхідними даними, розробки пристроїв з асоціативною пам'яттю, експертних систем, які навчаються на основі прикладів, отримання знань з даних, адаптивного керування складними об'єктами і процесами, вирішення трудомістких задач оптимізації (наприклад, проблеми комівояжера тощо), шифрування, дешифрування, стиснення інформації та переклад тексту;

В залежності від області застосування, мережі можуть бути використані для наступних цілей: розпізнавання сигналів, мови, зображень і тексту; технічної та біомедичної діагностики; моделювання зв'язків у природничих науках і техніці; соціально-економічного прогнозування; управління в техніці та економіці; побудови інформаційно-пошукових систем; криптографії;

2.3.1. Перші одношарові нейронні мережі

Першою штучною нейронною мережею, класифікованою відповідно до сучасної термінології як одношарову, був персептрон Розенблата[24], показаний на малюнку 2.2.

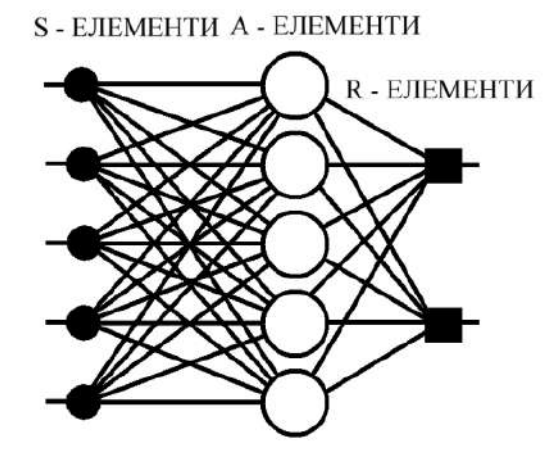

Рисунок - 2.2 Персептрон Розенбланта[24]

Оскільки автор розглядав перцептрон як модель мозку, елементи його структури відповідають елементам простої рефлекторної мережі. Елемент S імітує роботу сенсорних клітин, збираючи інформацію про навколишнє середовище та передаючи її елементу A, який фактично є формальним нейроном із функцією порогової активації. R елементів із фіксованими вхідними вагами, які використовуються для визначення внеску кожного елемента A, призначаються для формування реакції персептрона. Сучасним прототипом перцептрона Розенблата є одношарова нейронна мережа прямого розповсюдження, показана на малюнку 2.3.

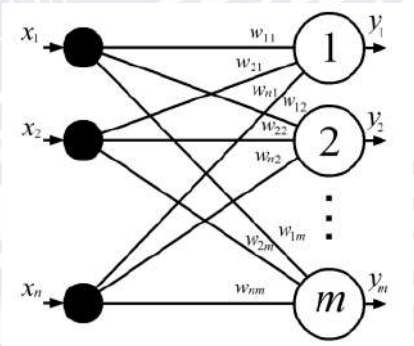

Рисунок - 2.3 Одношарова нейронна мережа[24]

Він складається з m нейронів, які здатні приймати вхідні вектори сигналів $X = (x_1, \dots, x_i, \dots, x_n)$ одночасно. Для відтворення елементів x_i цього вектора використовуються спеціальні пристрої, які зображені зліва від нейрона. Ці пристрої не виконують обробки інформації, тому вони не вважаються шаром нейронної мережі. Відповідно до формальної моделі нейрона, кожен його вхідний сигнал множиться на ваговий коефіцієнт w_{ij}, де i – поточний номер елемента вектора X, а j – поточний номер нейрона[24].

На рис. 2.3 показаний загальний випадок одношарової нейронної мережі, де кожен нейрон бере участь в обробці всіх елементів вектора вхідних даних. Такий підхід не завжди економічно і технічно виправданий. Тому для вирішення конкретної задачі можна використовувати архітектуру зі зв'язною структурою, яка є підмножиною повнозв'язаної.

2.3.2. Багатошаровий перцептрон (MLP)

Багатошаровий персептрон - найвідоміший і найчастіше використовуваний тип нейронної мережі[31]. У більшості випадків сигнали передаються в мережі в одному напрямку: від входу до виходу. Немає ніякої петлі, вихід кожного нейрона не впливає на сам нейрон. Така архітектура називається прямим поширенням (рис. 2.4).

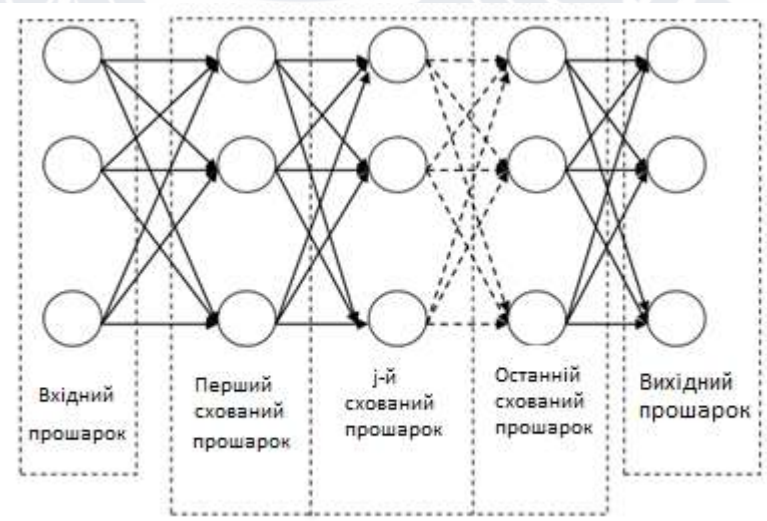

Рисунок - 2.4 Структура багатошарового персептрону[28]

Багатошаровий персептрон (MLP) складається з трьох типів прошарків - вхідного, вихідного та прихованого, як показано на рис. 2.4. Вхідний прошарок отримує вхідний сигнал для обробки. Необхідне завдання, таке як прогнозування та класифікація, виконує вихідний прошарок. Довільна кількість прихованих прошарків, які розміщуються між вхідним і вихідним прошарками, є справжнім обчислювальним двигуном MLP. Подібно до мережі прямого поширення, в MLP потоки даних рухаються в прямому напрямку від вхідного до вихідного прошарку. Нейрони в MLP навчаються за допомогою алгоритму навчання зі зворотним поширенням. Багатошарові песептрони призначені для апроксимації будь-якої неперервної функції і можуть розв'язувати задачі, які не є лінійно роздільними. Основними сферами застосування цієї нейронної мережі є класифікація образів, розпізнавання, прогнозування та апроксимація.

Всі нейрони одного прошарку повністю з'єднані з нейронами в сусідніх шарах[30]. Ці зв'язки представлені у вигляді ваг (інтенсивність зв'язку) в обчислювальному процесі. Ваги відіграють важливу роль у поширенні сигналу в мережі. Вони містять знання нейронної мережі про зв'язок між проблемою і рішенням. Кількість нейронів у вхідному прошарку залежить від кількості незалежних змінних у моделі, тоді як кількість нейронів у вихідному прошарку дорівнює кількості залежних змінних. Кількість вихідних нейронів може бути одинарною або множинною. В екологічному моделюванні екологічні змінні зазвичай подаються на вхідний прошарок як незалежні змінні для прогнозування біологічних змінних, які подаються на вихідний прошарок як цільові значення, що відповідають заданим вхідним значенням. Крім того, кількість прихованих прошарків та їх нейронів залежить від складності моделі і є важливими параметрами при розробці MLP-моделі.

2.3.3. Рекурентна нейронна мережа (RNN)

Рекурентні нейронні мережі відрізняються тим, що вхідна інформація передається між нейронами не лише у напрямку введення-виведення, а також у

зворотному напрямку[28]. Це досягається за допомогою зворотних зв'язків між нейронами. Завдяки цим зворотнім зв'язкам нейрони можуть знову виконувати свої функції, дозволяючи інформації проходити через рекурентну нейронну мережу кілька разів. На рисунку 2.5 зображено фрагмент рекурентної нейронної мережі.

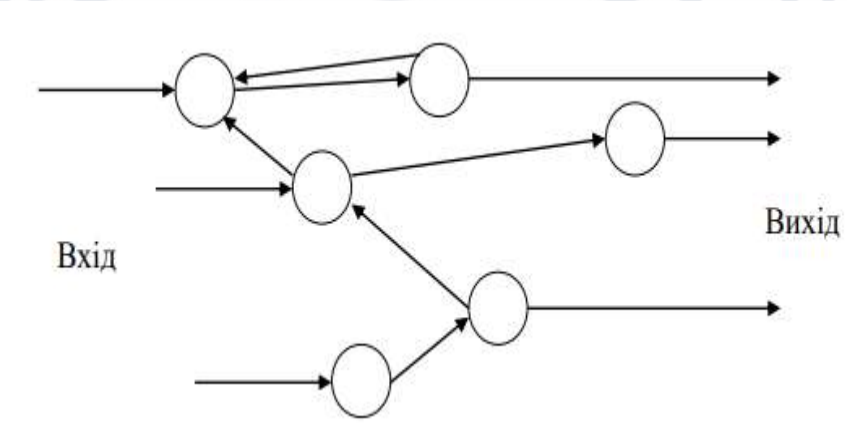

Рисунок - 2.5 Фрагмент рекурентної нейронної мережі[28]

Перевагою рекурентних нейронних мереж є їхні динамічні та ітеративні методи обробки даних, що у майбутньому має позитивний вплив на їх узагальнюючі та обчислювальні можливості. Проте потенційна нестабільність і недостатня вивченість таких мереж, особливо мереж зі зворотнім зв'язком, становлять серйозні перешкоди для їх широкого використання. Нестабільність рекурентних нейронних мереж проявляється у постійній зміні стану нейронів без досягнення стаціонарного стану мережі. Це може призвести як до ускладнення процесу навчання, так і до неоднозначності вихідної інформації при розпізнаванні невідомих образів. Деякі рекурсивні нейронні мережі, зокрема асоціативні пам'яті, базуються на архітектурі мережі Хопфілда. Іншими відомими типами асоціативних нейронних мереж є мережі Коско і Хеммінга, проте деякі їх модифікації не мають зворотніх зв'язків і, отже, не вважаються рекурсивними.

Окрім вже відомих класичних асоціативних мереж, таких як Хопфілда, Хеммінга і Коско, варто звернути увагу на рекурсивну автоасоціативну мережу (RAAM - Recursive Autoassociative Memory), яка може бути розглянута як розвиток мережі Джордана та простої рекурентної мережі (SRN - Simple

Recurrent Network), а також семантичну нейронну мережу, схожу на неї. Особливим призначенням RAAM і семантичної нейронної мережі є розпізнавання семантики тексту, що значно вплинуло на їхню архітектуру та методи навчання. Важливо зазначити, що мережі типу карт Кохонена також можуть вважатися рекурентними через наявність зворотного зв'язку між нейронами. Проте через суттєві відмінності в архітектурі, сфері застосування, методиках навчання та теоретичній базі їхніх мереж розглядаються окремо від рекурентних карт. Крім того, недостатньо вивчені нейронні мережі та типи мереж з обмеженими можливостями, наприклад, лінійний асоціатор, залишаються поза увагою.

2.3.4. Мережа з радіально базисними функціями (RBFNN)

Радіально-базисні мережі використовують радіальні базисні функції як свої активаційні функції [32]. У цих мереж вихід представляє собою лінійну комбінацію радіальних базисних функцій вхідних даних та параметрів нейронів. RBF-мережі мають багато застосувань, а саме:

1) апроксимація функцій,
2) прогнозування часових рядів,
3) класифікація
4) управління системами.

RBF, як показано на рис. 2.6, складається з наступних трьох прошарків.

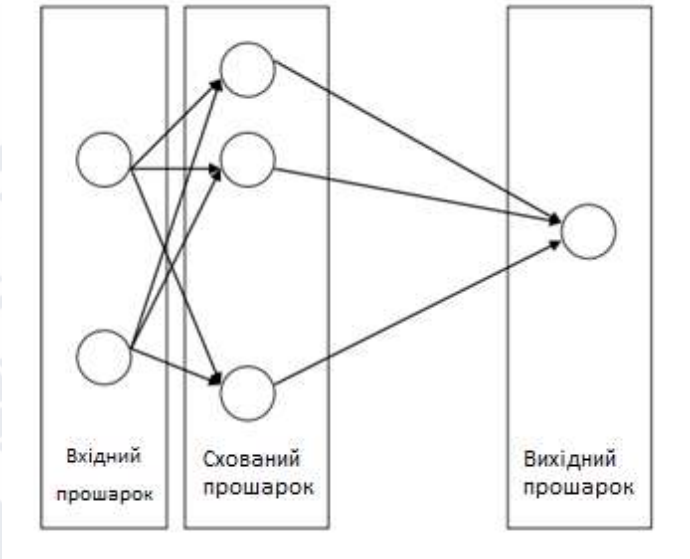

Рисунок - 2.6 Спрощена схема мережі з радіально базисними функціями[28]

Вхідний прошарок: складається з m_o вихідних вузлів, де m_o - розмірність вхідного вектора x.

Прихований прошарок: складається з такої ж кількості обчислювальних одиниць, як і розмір навчальної вибірки, N; кожна одиниця математично описується радіально-базисною функцією.

$$\varphi_j = \varphi\left(\left|\left|x - x_j\right|\right|\right) \quad (2.8)$$

Де, j - та точка вхідних даних x_j визначає центр радіально-базисної функції, а вектор x є сигналом (шаблоном), що застосовується до вхідного прошарку. Таким чином, на відміну від MLP, зв'язки, що з'єднують вихідні вузли з прихованими одиницями, є прямими зв'язками без ваг.

Вихідний прошарок: складається з однієї обчислювальної одиниці. Очевидно, що немає ніяких обмежень на розмір вихідного прошарку, за винятком того, що зазвичай розмір вихідного прошарку набагато менший, ніж розмір прихованого прошарку.

Перевагами RBF-мереж є простота побудови, хороше узагальнення, сильна толерантність до вхідного шуму та здатність до онлайн - навчання. Крім того, ці властивості RBF-мереж роблять їх дуже придатними для проектування гнучких систем керування.

2.3.5. Комплементарна нейронна мережа (CMTNN)

Комплементарна нейронна мережа (CMTNN) - це метод, що використовує пару взаємодоповнюючих нейронних мереж зі зворотним поширенням, які називаються нейронною мережею істини (Truth NN) та нейронною мережею хибності (Falsity NN), як показано на рис. 2.7 У той час як Truth NN - це нейронна мережа, яка навчена передбачати ступінь істинності висловлювань, Falsity NN навчена передбачати ступінь хибності висловлювань. Хоча архітектура та вхідні дані Falsity NN такі ж самі, як і у Truth NN, Falsity NN використовує для навчання мережі комплементарні виходи Truth NN[21].

На етапі тестування, тестовий набір застосовується до обох мереж для прогнозування ступеня істинності та хибності значень належності. Очікується, що для кожного вхідного шаблону передбачення значення хибної належності буде доповненням значення істинної належності. Замість того, щоб використовувати лише істинну приналежність для класифікації даних, як це зазвичай роблять більшість звичайних нейронних мереж, порівнюються прогнозовані результати істинної та хибної приналежності для того, щоб отримати результати класифікації.

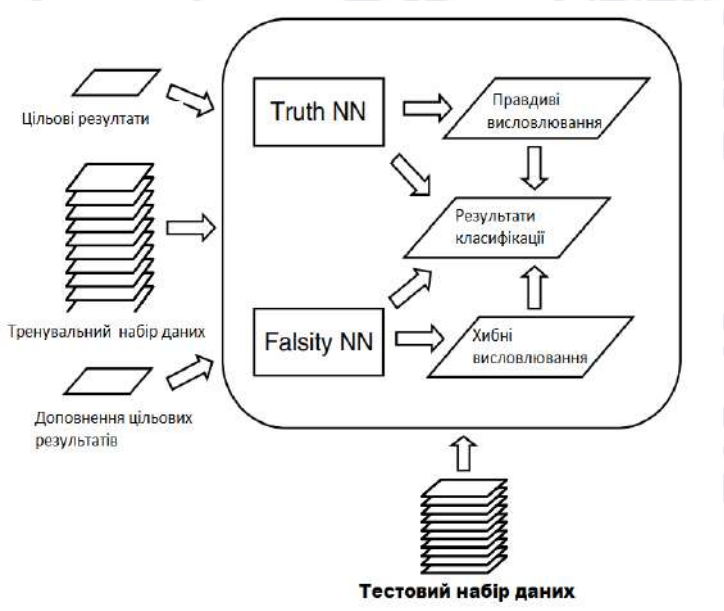

Рисунок - 2.7 Структура комплементарної нейронної мережі[21]

2.3.6. Ймовірнісна нейронна мережа (PNN)

Ймовірнісна нейронна мережа (PNN) - це тип нейронної мережі прямого поширення, що широко використовується для класифікації та розпізнавання образів [20]. У алгоритмі PNN для кожного класу апроксимується батьківська функція розподілу ймовірностей (PDF) за допомогою вікна Парзена та непараметричної функції. Після цього, з використанням PDF кожного класу, оцінюється ймовірність належності нових вхідних даних до кожного класу, і потім застосовується правило Байєса для вибору класу з найвищою апостеріорною ймовірністю для нових вхідних даних. Цей метод мінімізує ймовірність неправильної класифікації. Цей тип штучної нейронної мережі був створений на основі мережі Байєса та статистичного алгоритму, який називається дискримінантний аналіз ядра Фішера. Він був введений Д.Ф. Шпехтом у 1966 році. У нейронній мережі операції організовані в багатошарову мережу прямого поширення з чотирма прошарками показаній на рис. 2.8.

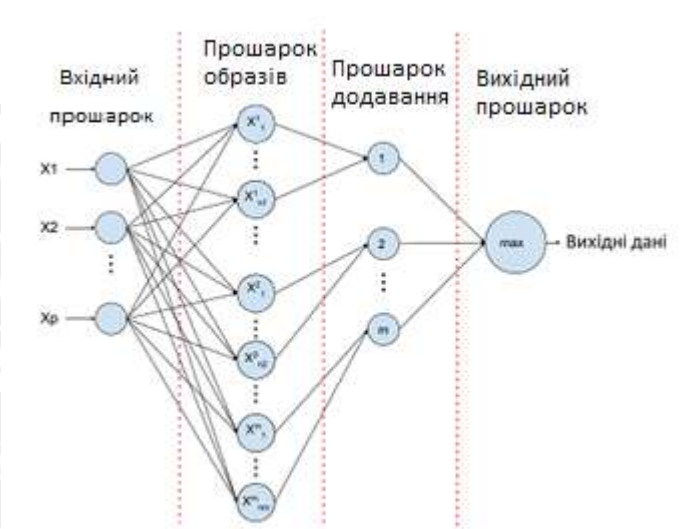

Рисунок - 2.8 Ймовірнісна нейронна мережа[20]

PNN часто використовується в задачах класифікації. Коли є вхідні дані, перший прошарок обчислює відстань від вхідного вектора до навчальних вхідних векторів. Це створює вектор, елементи якого вказують на те, наскільки вхідні дані близькі до вхідних даних навчання. Другий прошарок підсумовує

внесок для кожного класу входів і виробляє чистий вихід у вигляді вектора ймовірностей. Нарешті, конкурентна передавальна функція на виході другого прошарку вибирає максимальну з цих ймовірностей і видає 1 (позитивна ідентифікація) для цього класу і 0 (негативна ідентифікація) для нецільових класів.

Вхідний прошарок

Кожен нейрон вхідного прошарку представляє змінну-предиктор. У категорійних змінних використовується N-1 нейрон, коли є N кількість категорій. Він стандартизує діапазон значень шляхом віднімання медіани та ділення на інтерквартильний діапазон. Потім вхідні нейрони подають значення на кожен з нейронів прихованого прошарку.

Прошарок образів

Цей прошарок містить по одному нейрону для кожного випадку в навчальному наборі даних. Він зберігає значення предикторних змінних для цього випадку разом з цільовим значенням. Прихований нейрон обчислює евклідову відстань тестового прикладу від центральної точки нейрона, а потім застосовує ядро радіальної базисної функції, використовуючи значення сигми.

Прошарок додавання

У PNN для кожної категорії цільової змінної є один нейрон-шаблон. Фактична цільова категорія кожного навчального випадку зберігається з кожним прихованим нейроном; зважене значення, що виходить з прихованого нейрона, подається тільки на нейрон-шаблон, який відповідає категорії прихованого нейрона. Нейрони-шаблони додають значення для класу, який вони представляють.

Вихідний прошарок

Вихідний прошарок порівнює зважені голоси для кожної цільової категорії, накопичені в прошарку шаблонів, і використовує найбільший голос для прогнозування цільової категорії.

До переваг мережі PNN відноситься:

- Можливість проведення якісної класифікації на невеликих наборах учбових даних.

- Низька чутливість до помилкових даних в учбових наборах.

- Простота програмної реалізації та ймовірністний зміст класифікації. Вказані переваги значно полегшуютьнтерпретацію вихідних результатів

Загальними недоліками мережі PNN є:

- Якісна класифікація образів можлива тільки в діапазоні навчальних даних. В класичному вигляді мережа не здатна проводити узагальнення та не володіє асоціативними властивостями.

- Потенційно висока обчислювальна ресурсоємкість. Причиною цього є те, що мережа PNN містить в своєму складі весь навчальний матеріал, а через це вона потребує великого обсягу пам'яті та повільно працює.

- Можливість використання тільки в задачах класифікації.

2.3.7. Загально-регресивна нейронна мережа (GRNN)

GRNN - це нейронна мережа прямого поширення для керованих даних. Вона використовує нелінійні регресійні функції для апроксимації. Подібно до BPNN, GRNN складається з трьох прошарків: вхідного, прихованого та вихідного. GRNN використовує пряме відображення для зв'язку вхідного прошарку з прихованим[34].

Мережа складається з 4 прошарків нейронів. Завдання вхідного прошарку полягає лише в тому, щоб отримати зовнішній сигнал і розподілити його між усіма нейронами першого проміжного прошарку, з функцією активації Гауса. Другий проміжний рівень містить два нейрони для обчислення середньозважених компонентів. Вихідний прошарок призначений для остаточного визначення середньозваженого. Можна модифікувати GRNN так, щоб радіальні елементи відповідали не індивідуальним даним навчання, а їх класам. Це дає можливість зменшити розмір мережі та збільшити швидкість

навчання та розпізнавання. Центри активності можна розрахувати за допомогою методу K-середніх або за допомогою мережі Кохонена[28].

У той час як BPNN використовує швидкість навчання та імпульс для передатної функції, GRNN використовує коефіцієнт згладжування як параметр на етапі навчання. Єдиний коефіцієнт згладжування вибирається для оптимізації передавальної функції для всіх вузлів. Для зменшення обчислювального часу GRNN виконує навчання за один прохід через мережу. Завдяки цьому процесу GRNN може зменшити обчислювальну складність та покращити процес навчання. GRNN схожа на імовірнісну нейронну мережу (PNN), оскільки обидві мережі використовують непараметричні оцінки функцій щільності ймовірності. У той час як GRNN підходить для оцінки неперервних значень, PNN підходить для пошуку меж між категоріями моделей. На рисунку 2.9 показано структуру загально-регресивної нейронної мережі.

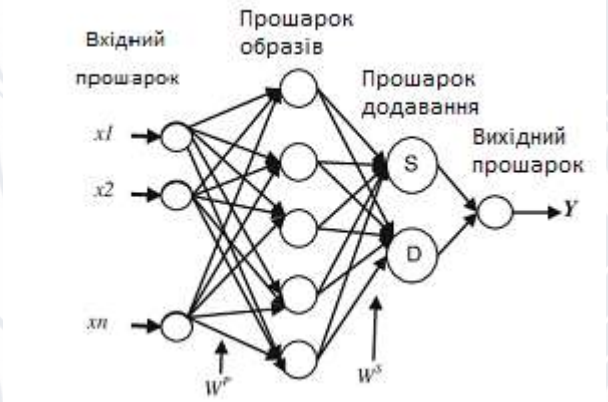

Рисунок - 2.9 Загально-регресивна нейронна мережа[33]

2.3.8. Нейронна мережа зі зворотним поширенням (BPNN)

BPNN є простою та ефективною моделлю нейронної мережі. Вона містить три прошарки - вхідний, прихований та вихідний, як показано на рис. 2.10 [34]. Ця мережа також відома як нейронна мережа прямого поширення зі зворотним поширенням.

На етапі навчання навчальні дані подаються на вхідний прошарок. Вони поширюються як до прихованого прошарку, так і до вихідного прошарку. Цей

процес називається прямим проходженням. На цьому етапі кожен вузол вхідного прошарку, прихованого прошарку та вихідного прошарку обчислює та коригує відповідну вагу між вузлами та генерує вихідне значення отриманої суми. Фактичні вихідні значення будуть порівнюватися з цільовими вихідними значеннями. Похибка між цими вихідними значеннями обчислюється і передається назад до прихованого прошарку, щоб знову оновити вагу кожного вузла. Це називається зворотним проходом або навчанням. Мережа буде повторюватися протягом багатьох циклів, поки помилка не стане прийнятною. Після завершення фази навчання навчена мережа готова до використання для будь-яких нових вхідних даних. Під час фази тестування не відбувається навчання або модифікації вагових матриць. Вхідні дані для тестування подаються на вхідний прошарку, і мережа прямого поширення генерує результати на основі знань, отриманих від навченої мережі.

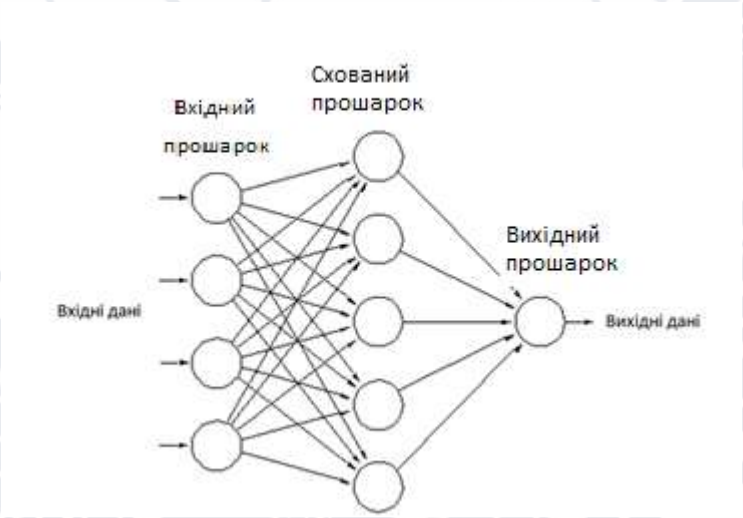

Рисунок - 2.10 Нейронна мережа зі зворотним поширенням[34]

2.4. Процес навчання

Після того, як мережа структурована для конкретної програми, вона готова до навчання. Щоб розпочати цей процес, початкові ваги обираються випадковим чином. Потім починається навчання, або тренування. Існує два підходи до навчання - контрольоване і неконтрольоване. Навчання під

наглядом передбачає механізм надання мережі бажаного результату або шляхом ручного "оцінювання" роботи мережі, або шляхом надання бажаного результату за допомогою входів. Некероване навчання - це коли мережа повинна зрозуміти сенс вхідних даних без сторонньої допомоги. Переважна більшість мереж використовує контрольоване навчання. Некероване навчання використовується для того, щоб виконати деяку початкову характеристику вхідних даних[22].

2.4.1. Навчання під наглядом

Під час навчання, система працює з вхідними даними та видає результати, які порівнюються з очікуваними. Помилки коригуються, ваги, які контролюють мережу, адаптуються. Цей процес повторюється, оскільки ваги постійно змінюються. Набір даних для навчання відомий як "навчальна вибірка". Під час навчання, одні й ті ж дані обробляються багато разів, щоб точніше налаштувати ваги. Сучасні пакети для розробки мереж надають інструменти для моніторингу того, наскільки точно мережа передбачає правильні відповіді. Ці інструменти дозволяють навчальному процесу тривати довше, зупиняючись лише при досягненні певної точки або точності. Проте, деякі мережі можуть залишатися ненавченими[22].

Це може стати причиною того, що вхідні дані не містять достатньої конкретної інформації для виведення бажаного результату. Також мережі можуть не збігатися, якщо наявних даних недостатньо для повного навчання. В ідеалі, даних має бути достатньо, щоб частину даних можна було притримати для тестування. Багато багаторівневих мереж з декількома вузлами здатні запам'ятовувати дані. Щоб контролювати мережу і визначити, чи система просто запам'ятовує дані якимось несуттєвим чином, контрольоване навчання повинно зберігати набір даних, які будуть використані для тестування системи після того, як вона пройде навчання. Якщо мережа не здатна вирішити задачу, розробник має переглянути вхідні та вихідні дані, кількість шарів, кількість

елементів у кожному шарі, зв'язки між шарами, функції активації, методи навчання, а також початкові ваги.

Ці зміни, необхідні для створення успішної мережі, складають процес, в якому відбувається "творчість" нейромережі. Інша частина творчості дизайнера - це правила навчання. Існує безліч алгоритмів, які застосовуються для реалізації адаптивного зворотного зв'язку, що дозволяє коригувати вагові коефіцієнти під час навчання. Найпоширенішою технікою є зворотне поширення помилки, більш відоме як зворотне розповсюдження. Ці різні методи навчання розглядаються більш детально далі в цьому звіті. Навчання - це більше, ніж просто техніка. Воно включає в себе інтуїцію та свідомий аналіз, щоб забезпечити уникнення перенавчання мережі. Спочатку штучна нейронна мережа налаштовується на загальні статистичні закономірності даних. Згодом вона починає "вивчати" інші аспекти даних, які можуть бути помилковими з точки зору загальної тенденції. Коли система нарешті досягає правильного рівня навчання і не потребує подальших коригувань, ваги можна "заморозити". У деяких випадках цю налаштовану мережу перетворюють на апаратне забезпечення для забезпечення швидкої роботи. Інші системи залишаються гнучкими і продовжують навчатися під час експлуатації.

2.4.2. Навчання без нагляду, або адаптивне навчання

Інший тип навчання називається неконтрольованим навчанням[22]. Під час такого навчання мережі надаються вхідні дані, але не вказуються бажані результати. Система повинна сама вирішити, за якими ознаками вона буде групувати вхідні дані. Цей процес часто називають самоорганізацією або адаптацією. В даний час некероване навчання недостатньо вивчене. Ця адаптація до навколишнього середовища є обіцянкою, яка дозволить науково-фантастичним типам роботів безперервно навчатися самостійно, коли вони стикаються з новими ситуаціями і новими середовищами. Життя наповнене ситуаціями, в яких не існує точних навчальних наборів. Деякі з цих ситуацій

пов'язані з військовими діями, де можна зіткнутися з новими методами ведення бою і новою зброєю. Через цей несподіваний аспект життя і бажання людини бути підготовленою, продовжуються дослідження і надії в цій галузі. Проте, в даний час переважна частина роботи нейронних мереж припадає на системи з керованим навчанням. Контрольоване навчання досягає результатів.

2.4.3. Глибоке навчання

Потужна форма машинного навчання, яка дозволяє комп'ютерам вирішувати проблеми сприйняття, такі як розпізнавання зображень і мови, все більше проникає в біологічні науки[50]. Ці методи глибокого навчання, такі як глибокі штучні нейронні мережі, використовують кілька рівнів обробки для виявлення закономірностей і структури в дуже великих масивах даних. Кожен рівень вивчає концепцію на основі даних, на які спираються наступні рівні; чим вищий рівень, тим більш абстрактні концепції, які вивчаються. Глибоке навчання не залежить від попередньої обробки даних і автоматично витягує ознаки. Наведемо простий приклад: глибока нейронна мережа, якій доручено інтерпретувати фігури, навчиться розпізнавати прості ребра на першому рівні, а потім додасть розпізнавання складніших фігур, що складаються з цих ребер, на наступних рівнях. Не існує чіткого правила, скільки прошарків потрібно для глибокого навчання, але більшість експертів сходяться на думці, що потрібно більше двох.

Глибоке навчання буде безцінним у контексті великих даних, оскільки воно витягує високорівневу інформацію з дуже великих обсягів даних. У міру того, як воно набирає обертів в аналізі геному, вирішуються початкові проблеми, такі як надмірне пристосування через рідкісні залежності в навчальних даних і високі обчислювальні витрати.

Моделі глибокого навчання складаються з різноманітних глибоких мереж[49]. Серед них глибокі короткі мережі, глибокі нейронні мережі, згорткові нейронні мережі та рекурентні нейронні мережі є моделями

керованого навчання, тоді як автокодери, обмежені машини Больцмана та генеративні змагальні мережі є моделями неконтрольованого навчання. Кількість досліджень IDS, заснованих на глибокому навчанні, стрімко зросла з 2015 року і дотепер. Моделі глибокого навчання безпосередньо вивчають представлення ознак з вихідних даних, таких як зображення і тексти, не вимагаючи ручної інженерії ознак. Таким чином, методи глибокого навчання можуть виконуватися наскрізним чином. Для великих наборів даних методи глибокого навчання мають значну перевагу над поверхневими моделями.

2.4.4. Функції активації

Функції активації спеціально використовуються в штучних нейронних мережах для перетворення вхідного сигналу на вихідний, який, у свою чергу, подається на вхід наступного прошарку в стеку. У штучній нейронній мережі ми обчислюємо суму добутків входів на відповідні ваги і, нарешті, застосовуємо до неї функцію активації, щоб отримати вихідний сигнал цього прошарку і подати його як вхідний сигнал наступному прошарку.

Точність прогнозування нейронної мережі залежить від кількості використовуваних прошарків і, що більш важливо, від типу використовуваної функції активації[29].

Бінарна ступінчаста функція

Коли ми маємо функцію активації, найважливішим для розгляду є пороговий класифікатор, який означає, що значення лінійного перетворення має активувати нейрон чи ні, або ми можемо сказати, що нейрон активується, якщо вхід на функцію активації перевищує порогове значення, інакше він деактивується[29]. У цьому випадку вихід не подається на вхід наступного прошарку. Бінарна функція кроку - це найпростіша функція активації, яка існує, і її можна реалізувати за допомогою простих операторів if-else у мові Python. При створенні бінарного класифікатора зазвичай використовують бінарні функції активації. Але бінарна функція кроку не може бути використана у

випадку багатокласової класифікації в цільовій області. Крім того, градієнт бінарної функції кроку дорівнює нулю, що може спричинити перешкоду при зворотному поширенні, тобто якщо ми обчислюємо похідну f(x) по відношенню до x, вона дорівнює нулю. Математично бінарну ступінчасту функцію можна визначити наступним чином:

$$f(x) = 1, x \geq 0$$
$$f(x) = 0, x < 0 \quad (2.1)$$

Лінійна функція активації

Лінійна функція активації прямо пропорційна вхідному сигналу[29]. Основним недоліком бінарної ступеневої функції було те, що вона мала нульовий градієнт, оскільки в бінарній ступеневій функції відсутній компонент x. Для того, щоб усунути цей недолік, можна використати лінійну функцію. Її можна задати наступним чином:

$$F(x) = ax \ (2.2)$$

Значення змінної a може бути будь-якою сталою величиною, вибраною користувачем. Тут похідна функції f(x) не дорівнює нулю, а дорівнює значенню використаної константи. Градієнт не дорівнює нулю, а є постійною величиною, яка не залежить від вхідного значення x, що означає, що ваги та зміщення будуть оновлюватися під час кроку зворотного поширення, хоча коефіцієнт оновлення буде тим самим. Використання лінійної функції не має великої користі, оскільки нейронна мережа не зможе покращити помилку через однакове значення градієнта для кожної ітерації. Крім того, мережа не зможе ідентифікувати складні закономірності в даних. Тому лінійні функції ідеально підходять там, де потрібна інтерпретованість і для простих завдань.

Сигмоїдальна функція активації

Це найпоширеніша функція активації, оскільки вона є нелінійною функцією[29]. Сигмоїдна функція перетворює значення в діапазоні від 0 до 1. Її можна визначити як:

$$f(x) = 1/e^{-x} \ (2.3)$$

Сигмоїдна функція неперервно диференційована і є гладкою S-подібною функцією. Похідною функції є:

$$f'(x) = 1 - sigmoid(x) \quad (2.4)$$

Крім того, сигмоїдна функція не є симетричною відносно нуля, що означає, що знаки всіх вихідних значень нейронів будуть однаковими. Цю проблему можна вирішити шляхом масштабування сигмоїдної функції.

Функція гіперболічного тангенса

Це функція гіперболічного тангенса[29]. Функція тангенса схожа на функцію сигмоїда, але вона симетрична відносно початку координат. Це призводить до різних знаків виходів з попередніх прошарків, які будуть подаватися на вхід наступного прошарку. Її можна визначити так:

$$\mathrm{f(x)} = 2\mathrm{sigmoid}(2\mathrm{x}) - 1 \quad (2.5)$$

Функція Тана є неперервною і диференційованою, її значення лежать в діапазоні від -1 до 1. У порівнянні з сигмоїдною функцією, градієнт функції тангенса є більш крутим. Таненсна є кращою за сигмоїдну функцію, оскільки вона має градієнти, які не обмежені у зміні в певному напрямку, а також має нульовий центр.

Функція ReLU

ReLU розшифровується як rectified liner unit і є нелінійною функцією активації, яка широко використовується в нейронних мережах[29].

Перевагою використання функції ReLU є те, що всі нейрони не активуються одночасно. Це означає, що нейрон буде деактивовано лише тоді, коли результат лінійного перетворення дорівнює нулю. Математично це можна виразити як:

$$\mathrm{f(x)} = \max(0, \mathrm{x}) \quad (2.6)$$

ReLU є більш ефективною, ніж інші функції, оскільки всі нейрони не активуються одночасно, а лише певна кількість нейронів активується одночасно. У деяких випадках значення градієнта дорівнює нулю, завдяки чому ваги та зсуви не оновлюються на кроці зворотного розповсюдження при навчанні нейронної мережі.

Функція активації Softmax

Функція Softmax - це комбінація декількох сигмоїдних функцій[29]. Оскільки ми знаємо, що сигмоїдна функція повертає значення в діапазоні від 0 до 1, їх можна розглядати як ймовірності точок даних певного класу. Функція Softmax, на відміну від сигмоїдних функцій, які використовуються для бінарної класифікації, може бути використана для задач багатокласової класифікації. Ця функція для кожної точки даних усіх окремих класів повертає ймовірність. Вона може бути виражена як:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \; for \; j = 1, \dots, K. \; (2.7)$$

Коли ми будуємо мережу або модель для багатокласової класифікації, то вихідний прошарок мережі матиме таку ж кількість нейронів, як і кількість класів у цільовій вибірці.

2.5. Висновки до розділу 2

У цьому розділі було визначено, що нейроні мережі представляють собою математичні моделі, інспіровані біологічними нейронними мережами людини, які використовуються для розв'язання складних завдань в області штучного інтелекту та машинного навчання. Історія нейронних мереж розкриває їхній шлях від початкових концепцій до сучасних технологій, де кожен етап сприяв розвитку та вдосконаленню їх функціональності та ефективності. Класифікація нейромереж забезпечує систематизацію різноманітних видів нейронних мереж відповідно до їх архітектури та специфікацій, що дозволяє краще розуміти їхні можливості та області застосування. Аналіз основних видів нейронних мереж, таких як одношарові мережі, багатошаровий перцептрон, рекурентні мережі та інші, дозволив визначити їхні особливості та застосування у різних сферах, включаючи обробку даних, розпізнавання образів та прогнозування. Загально висновуючи, розділ про нейронні мережі відображає важливість їхнього

вивчення та використання у сучасних технологіях, що сприяє розвитку інтелектуальних систем та досягненню нових наукових та технічних досягнень.

РОЗДІЛ 3. ПОБУДОВА СТРУКТУРИ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ ВИЯВЛЕННЯ АТАК ТИПУ U2R ТА R2L

3.1. Вибір архітектури нейронної мережі

При виборі архітектури нейронної мережі перевагу отримав такий клас нейронних мереж, як багатошарові мережі прямого поширення, які, зазвичай, називають багатошаровими персептронами і які є окремим випадком персептрона Розенблата.

Нейронні мережі прямого поширення працюють за принципом передавання вхідного сигналу від одного шару до іншого. Таким чином функціонує багатошаровий персептрон, який містить вхідний шар, приховані обчислювальні шари та вихідний шар нейронів. Багатошаровий персептрон є односпрямованою мережею сигмоїдального типу [10].

Багатошарові персептрони часто застосовуються для контрольованих завдань навчання: вони навчаються на безлічі пар вхідних та вихідних даних і моделюють кореляцію (або залежності) між цими входами і виходами. Навчання включає в себе налаштування параметрів або ваг і зсувів моделі з метою мінімізації помилок.

У нашому випадку вхідний прошарок складається з 41 нейрона (параметри мережевого з’єднання), три прихованих прошарки з експериментально підібраною кількістю нейронів та трьома вихідними прошарками які виявляють факт атаки типів U2R, *L2R* або нормальне мережеве з’єднання.

Запропонована модель представляє собою нейронну мережу з трьома прихованими прошарками та функцією активації *ReLU* (*Rectified Linear Unit* - функція активації, яка широко використовується в нейронних мережах. Для будь-якого вхідного значення *x*, якщо воно менше або дорівнює нулю, вихід буде нульовим. Якщо ж вхід більше нуля, вихід буде рівний вхідному значенню) на кожному з них, а також вихідним прошарком з активацією *Sigmoid*.

Розглянемо детальний опис моделі.

- *Вхідний прошарок*: Вхідний прошарок має розмірність *dataset_inputs*, що визначається кількістю функцій (ознак) у вхідних даних. У цьому випадку використовуються дані *train_X* та *test_X*.
- *Прихований прошарок №1*: Перший прихований прошарок має 46 нейронів. Активація цього прошарку виконується за допомогою функції активації *ReLU*.
- *Прихований прошарок №2*: Другий прихований прошарок має 32 нейрони. Активація також виконується за допомогою *ReLU*.
- *Прихований прошарок №3*: Третій прихований прошарок має 12 нейронів, активація - *ReLU*.
- *Вихідний прошарок*: Вихідний прошарок має розмірність *dataset_outputs*, який визначається кількістю класів (або кількістю вихідних значень). У цьому випадку використовується активація *Sigmoid*, яка призначена для задач класифікації бінарних даних.

Запропонована модель навчається за допомогою оптимізатора *Adam* з коефіцієнтом навчання *lr*=0.00456789. Для оцінки втрат використовується середньоквадратична помилка (*MSE*). Навчання проводиться протягом 500 циклів з пакетами розміру *batch_size*=300. На кожному циклі вимірюється середньоквадратична помилка на тестовому наборі. Якщо поточне значення *MSE* на тесті є кращим, ніж попереднє краще значення *MSE*, то модель оновлюється. У цьому випадку краща модель зберігається.

3.2. Вибір мови програмування

Для розробки програмного забезпечення використовувалась мова програмування *Python 3*.

Python – об’єктно-орієнтована мова програмування, яка набула надзвичайної популярності в сфері машинного навчання та штучного інтелекту

завдяки великій системі бібліотек і фреймворків та можливості легкої інтеграції між наявними компонентами. Перевагами *Python* перед іншими мовами програмування для роботи зі штучними інтелектом і, зокрема, нейронними мережами можна вважати:

– широкий вибір бібліотек та фреймворків;

– кросплатформеність;

– адаптивність та гнучкість у стилях використання;

– простий та зрозумілий синтаксис;

– вбудовані засоби візуалізації даних

3.3. Використані бібліотеки

Для розробки програмного забезпечення використовувались такі бібліотеки мови програмування *Python* [11]:

- ***NumPy* (*numpy*)**: Це бібліотека для наукових обчислень, яка надає підтримку для масивів та матриць, разом із великою кількістю функцій для роботи з ними.
- ***Pandas* (*pandas*)**: Це бібліотека для обробки та аналізу даних, яка забезпечує структури даних, такі як *DataFrame*, що спрощують роботу з даними.
- ***Scikit-learn* (*sklearn*)**: Це бібліотека машинного навчання для *Python*, яка надає інструменти для класифікації, регресії, кластеризації та інших видів аналізу даних.
- ***Matplotlib* (*matplotlib.pyplot*)**: Це бібліотека для візуалізації даних в *Python*, яка дозволяє будувати різноманітні графіки, діаграми, гістограми тощо.
- ***LabelEncoder* (*sklearn.preprocessing*)**: Це клас для кодування міток (*labels*) у числові подання. Використовується для конвертації категоріальних даних у числові.

- ***PyTorch* (*torch*)**: *PyTorch* - це бібліотека для машинного навчання та обчислювальних графів, яка широко використовується для розробки та навчання нейронних мереж. Вона надає інструменти для створення, навчання та валідації нейронних мереж, а також для обробки даних.
- ***torch.nn* (*torch.nn*)**: Модуль *nn* в *PyTorch* містить різноманітні вбудовані функції та класи, які допомагають збудувати та навчити нейронні мережі.
- ***torch.optim* (*torch.optim*)**: Цей модуль містить різноманітні оптимізатори, які використовуються для навчання нейронних мереж в *PyTorch*. Оптимізатори визначають методи оптимізації, такі як стохастичний градієнтний спуск (*SGD*), *Adam*, *RMSprop* тощо.
- ***copy* (*copy*)**: Модуль *copy* в *Python* містить функції для копіювання об'єктів. Використовують його для створення глибоких або поверхневих копій об'єктів, щоб уникнути проблем з посиланнями.
- ***tqdm* (*tqdm*)**: Це бібліотека для створення прогрес-барів в *Python*, яка дозволяє вам візуалізувати прогрес виконання ітераційних процесів, таких як цикли або завантаження даних.
- ***train_test_split* (*sklearn.model_selection*)**: Цей модуль надає функцію для розділення набору даних на випадкові тренувальні та тестові підмножини. Використовується для оцінки точності та узагальнення моделі. Зазвичай дані розділяються на тренувальні дані, які використовуються для навчання моделі, і тестові дані, які використовуються для оцінки її продуктивності на невідомих даних.

3.4. Використання датасету *KDD99* для проведення експериментальних досліджень

Для навчання нейронної мережі була використана база даних KDD99, яка містить стандартний набір інформації із широким спектром атак. Усього ця база включає близько 5 мільйонів записів про мережеві з'єднання. Кожен запис

представляє собою зображення мережевого з'єднання і містить 41 параметр мережевого трафіку (див. табл. 3.1), позначений як "атака" або "не атака"[9].

Таблиця 3.1 Параметри мережевого трафіка[51]

№ з/п	Параметр	Опис
1	duration	Тривалість (у секундах) з'єднання
2	protocol_type	Тип протоколу (TCP, UDP, etc.)
3	service	Атакований сервіс
4	src_bytes	Кількість байтів від джерела до призначення
5	dst_bytes	Кількість байтів відповіді клієнту
6	flag	Прапорці з'єднання
7	land	1, якщо з'єднання від/до того самого хоста/порта
8	wrong_fragment	Кількість „хибних" фрагментів
9	urgent	Кількість термінових пакетів
10	hot	Кількість „гарячих" індикаторів
11	num_failed_logins	Кількість невдалих спроб реєстрації
12	logged_in	1, якщо успішний вхід в систему; 0 неуспішне
13	num_compromised	Кількість „компроментуючих" умов
14	root_shell	1, якщо root shell отриманий; інакше 0
15	su_attempted	1, якщо виконувалась „su root" ; інакше 0
16	num_root	Кількість „root" доступів
17	num_file_creations	Кількість операцій створення файлів
18	num_shells	Кількість запитів на надання оболонки
19	num_access_files	Кількість операцій на доступ до контролю файлів
20	num_outbound_cmds	Кількість вихідних команд для FTP сесії
21	is_hot_login	1, якщо логін належав до „гарячого" списку

22	is_guest_login	1, якщо „гостьовий” вхід
23	count	Кількість з'єднань на хост в поточній сесії за останні 2 с
24	serror_rate	% з'єднань що мали „SYN” помилки
25	rerror_rate	% з'єднань що мали „REJ” помилки
26	same_srv_rate	% з'єднань що мали однаковий сервіс
27	diff_srv_rate	% з'єднань на різні сервіси
28	srv_count	Кількість з'єднань на такий самий сервіс за останні 2 с
29	srv_serror_rate	% з'єднання з помилкою в „SYN” пакеті
30	srv_rerror_rate	% з'єднання, що мають „REJ” помилки
31	srv_diff_host_rate	% з'єднання від інших хостів
32	dst_host_count	Кількість з`єднань до локального хоста, встановлених віддаленою стороною
33	dst_host_srv_count	Кількість з`єднань до локального хоста, встановлених віддаленою стороною та використовуючих одну службу
34	dst_host_same_srv_rate	% з`єднань до локального хоста, встановлених віддаленою стороною та використовуючих одну службу
35	dst_host_diff_srv_rate	% з`єднань до локального хоста, встановлених віддаленою стороною та використовуючих різні служби
36	dst_host_same_src_port_rate	% з`єднань до даного хоста при поточному номері порту джерела
37	dst_host_srv_diff_host_rate	% з`єднань до служби різних хостів
38	dst_host_serror_rate	% з`єднань з помилкою типу SYN для даного хост-приймача
39	dst_host_srv_serror_rate	% з`єднань з помилкою типу SYN для даної

		служби приймача
40	dst_host_rerror_rate	% з`єднань з помилкою типу REJ для даного хост-приймача
41	dst_host_srv_rerror_rate	% з`єднань з помилкою типу REJ для даної служби приймача

У базі зазначено 22 різновиди атак, які розділяються на чотири основні групи: *DoS*, *U2R*, *R2L* і *Probe*. Серед них найбільше уваги здобули *U2R* і *R2L* атаки.

U2R атаки спрямовані на отримання привілеїв локального суперкористувача (адміністратора) користувачем з наявним реєстраційним записом. Існує чотири основні типи *U2R* атак: buffer_overflow, loadmodule, perl, rootkit.

R2L атаки характеризуються намаганням незареєстрованого користувача отримати доступ до комп'ютера через віддалений доступ. Існує вісім типів *R2L* атак: ftp_write, guess_passwd, imap, multihop, phf, spy, warezclient, warezmaster.

Зовнішній вигляд бази KDD99 - це текстовий файл, де дані представлені у вигляді матриць, що описують певний тип атаки або нормального з'єднання. (рис. 3.1).

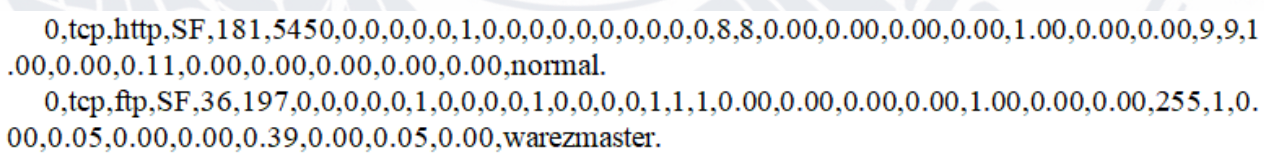

Рисунок - 3.1 Зовнішній вигляд бази *KDD99*

3.5. Результати експериментальних досліджень розробленої структури нейронної мережі для виявлення атак

Побудована нейронна мережа була навчена з отриманою найкращою середньоквадратичною помилкою *MSE*=0.0035732425749301 91 (рис. 3.2).

Середньоквадратична помилка (MSE) обчислюється як квадрат відхилення між прогнозованими значеннями моделі та реальними значеннями цільової змінної, і потім усереднюється по всіх прикладах в тестовому наборі даних.

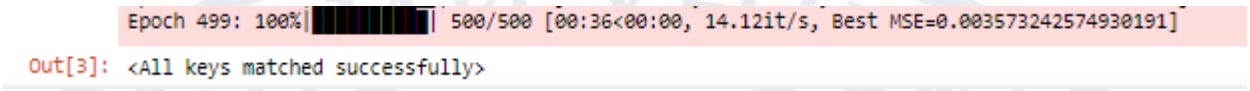

Рисунок - 3.2 Скрін-шот результату навчання

Була написана функція що обраховує статистику на тестовому наборі даних. Функція виконує проходження через кожну точку тестового набору, обчислює різницю між фактичними та передбаченими значеннями і виводе статистику у вигляді списку.

Статистика щодо розподілу результатів виконання програми на тестовому наборі даних наступна:

- Загальна кількість даних(записів про нормальні мережеві з'єднання та атаки у датасеті): 11178
- Кількість даних у тренувальному наборі: 8383
- Кількість даних у тестовому наборі: 2795
- Кількість даних, в яких нейромережа впевнена у правильності передбачень (*confident*): 2778
- Кількість сумнівних даних (*doubtful*): 2
- Кількість неправильно передбачених даних (*wrong*): 15

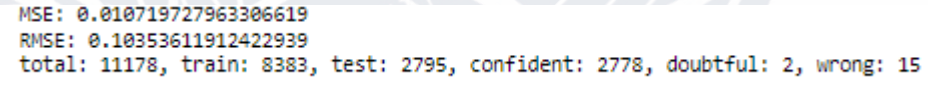

Рисунок - 3.3 Скрін-шот статистики результатів

На рис. 3.4 представлені результати перевірки навчання запропонованої моделі мережі.

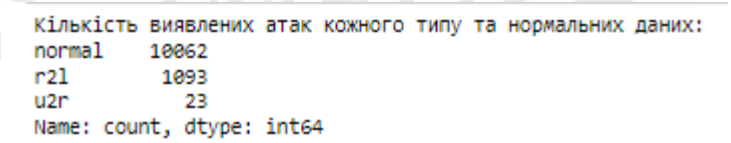

Рисунок - 3.4 Результат перевірки навчання запропонованої моделі мережі

З результатів перевірки роботи запропонованої моделі нейронної мережі видно, що з 1126 атак типу *R2L* модель виявляє 1093, а з 52 атак типу *U2R* виявляє 23. На рис. 3.5 представлено діаграму виявлених атак обох типів.

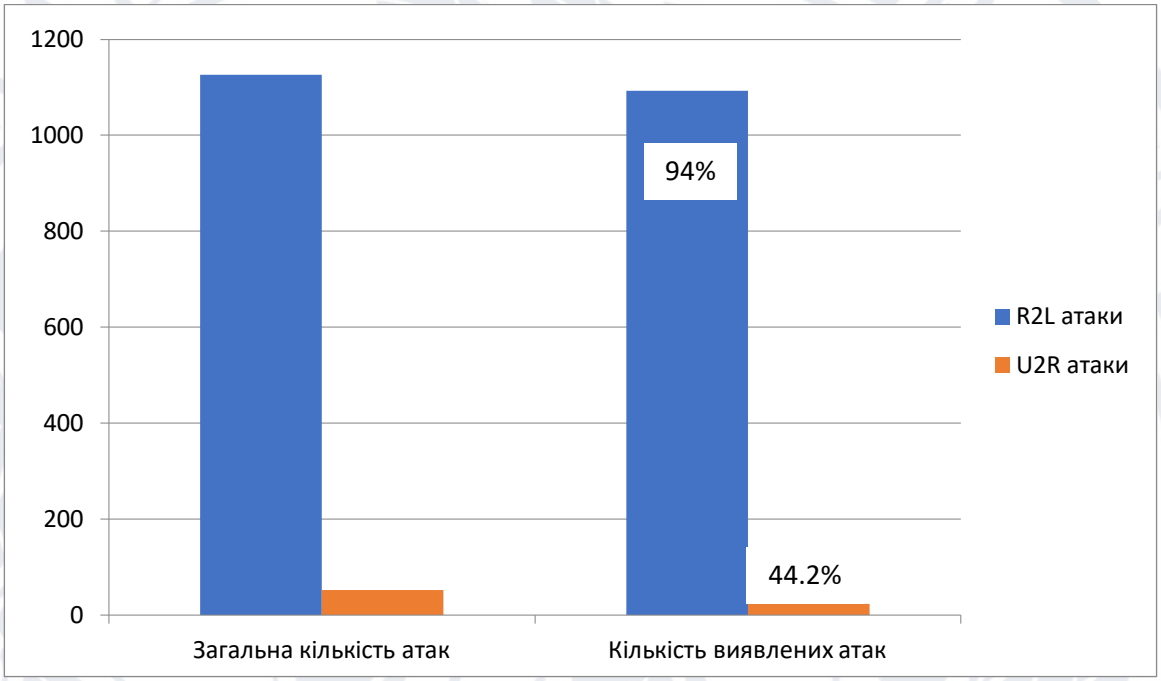

Рисунок - 3.5 Діаграма виявлених кібератак

3.6. Висновки до розділу 3

У цьому розділі пропонується використання багаторівневого персептрона для запобігання та виявлення кібератак R2L та U2R на організації критичної інфраструктури, а також його реалізація на мові програмування Python. Результатом розробки конструкції багатошарового перцептрона та його навчання є те, що розроблена модель нейронної мережі здатна розпізнавати 94% атак R2L і 44,2% атак U2R.

ВИСНОВКИ

У цій роботі запропоновано використовувати багатошаровий персептрон для попередження та виявлення кібератак типів *R2L* і *U2R* на підприємствах критичної інфраструктури, а також її реалізація мовою програмування *Python*.

В результаті розробки архітектури багатошарового персептрона та його навчання, розроблена модель нейронної мережі виявляє 94% атак типу *R2L* та 44.2% атак типу *U2R*. Менший відсоток виявлених кібератак типу *U2R* спричинений значно меншою кількістю цих атак в датасеті *KDD99*.

В цілому, отримані результати свідчать про ефективність запропонованої моделі виявлення атак типу *R2L*, а також вказують на потенційні можливості подальшого вдосконалення запропонованої моделі, зокрема, використовуючи збільшення обсягу даних для атак типу *U2R*.

Важливість систем попередження та виявлення атак на об'єкти критичної інфраструктури не можна недооцінювати в сучасному світі, де залежність від технологій і *Internet*-мереж зростає щодня. Критичні інфраструктури, такі як енергетика, транспорт, фінанси та інші сектори, є ключовими для ефективності існування суспільства і все частіше стають об'єктами кібератак. Це може мати серйозні наслідки для безпеки та економіки не тільки окремих об'єктів критичної інфраструктури, а й держави в цілому.

Системи виявлення вторгнень грають важливу роль у забезпеченні безпеки цих об'єктів, допомагаючи вчасно виявляти, аналізувати та реагувати на потенційні загрози. Вони дозволяють виявляти аномальні дії та спроби несанкціонованого доступу до систем, що допомагає уникнути потенційно серйозних наслідків, таких як втрата конфіденційної інформації, переривання роботи систем або навіть матеріальні збитки та загрози людським життям.

СПИСОК ВИКОРИСТАНИХ ПОСИЛАНЬ

1. Ş. Sağıroğlu, E.N. Yolaçan, and U. Yavanoğlu, “Zeki Saldırı Tespit Sistemi Tasarımı ve Gerçekleştirilmesi”, J. Fac. Eng. Arch. Gazi Univ., vol. 26(2), pp. 325-340, 2011.

2. P.A.R. Kumar, S. Selvakumar, “Detection of distributed denial of service attacks using an ensemble of adaptive and hybrid neuro-fuzzy systems”, Computer Communications, vol. 36, pp. 303-319, 2013.

3. S.Y. Wu, E. Yen, “Data mining-based intrusion detectors”, Expert Systems with Applications, vol. 36, pp. 5605– 5612, 2009.

4. K.K. Gupta, B. Nath, R. Kotagiri, “Layered Approach Using Conditional Random Fields for Intrusion Detection”, IEEE Transactions On Dependable And Secure Computing, vol. 7(1), pp. 35-49, 2010.

5. F. Kuang, W. Xu, S. Zhang, “A novel hybrid KPCA and SVM with GA model for intrusion detection”, Applied Soft Computing, vol. 18, pp. 178–184, 2014.

6. I. Mukhopadhyay, M. Chakraborty, S. Chakrabarti, T. Chatterjee, “Back Propagation Neural Network Approach to Intrusion Detection System”, 2011 International Conference on Recent Trends in Information Systems, Kolkata, pp. 303-308, December 2011.

7. T. Subbulakshmi, A.F. Afroze, “Multiple Learning based Classifiers using Layered Approach and Feature Selection for Attack Detection”, 2013 IEEE International Conference on Emerging Trends in Computing, Communication and Nanotechnology (ICECCN 2013), Tirunelveli, pp. 308- 314, March 2013.

8. N. Sharma, S. Mukherjee, “A Novel Multi-Classifier Layered Approach to Improve Minority Attack Detection in IDS”, Procedia Technology, vol. 6, pp. 913–921, 2012.

9. KDD Cup 1999 Data [Електронний ресурс] – Режим доступу до ресурсу: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

10. Swingler K. Lecture 4: Multi-Layer Perceptrons URL: http://www.cs.stir.ac.uk/courses/ ITNP4B/lectures/kms/4-MLP.pdf

11. Best Python Libraries for Machine Learning and Deep Learning. URL: https://towardsdatascience.com/best-python-libraries-for-machine-learning-and-deeplearning-b0bd40c7e8c

12. M. H. Kamarudin, C. Maple, T. Watson and H. Sofian, "Packet Header Intrusion Detection with Binary Logistic Regression Approach in Detecting R2L and U2R Attacks," 2015 Fourth International Conference on Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec), Jakarta, Indonesia, 2015, pp. 101-106, doi: 10.1109/CyberSec.2015.28.

13. R. Shanmugavadivu and Dr. N. Nagarajan, " Network intrusion detection system using fuzzy logic", in Indian Journal of Computer Science and Engineering, Vol. 2, pp. 101-111.

14. P. G. Jeya, M. Ravichandran, and C. S. Ravichandran, "Efficient Classifier for R 2 L and U 2 R Attacks," International Journal of Computer Applications, vol. 45, no. 21, 2012.

15. P. Sornsuwit and S. Jaiyen, "Intrusion detection model based on ensemble learning for U2R and R2L attacks," 2015 7th International Conference on Information Technology and Electrical Engineering (ICITEE), Chiang Mai, Thailand, 2015, pp. 354-359, doi: 10.1109/ICITEED.2015.7408971.

16. B. Debojit, N. Bernard, and K. B. Dhruba, "Anomaly based intrusion detection using meta ensemble classifier," in Proceedings of the Fifth International Conference on Security of Information and Networks, Jaipur, India, 2012.

17. S. Kumar and A. Yadav, "Increasing performance Of intrusion detection system using neural network," 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, Ramanathapuram, India, 2014, pp. 546-550, doi: 10.1109/ICACCCT.2014.7019145.

18. Beghdad. R, "Training all the KDD dataset to classify and detect attacks" in International Journal on Neural and Mass – Parallel computing and Information Systems, Vol. 17, March 2007.

19. ATICI, Mehmet Ali, Ibrahim Alper DOGRU, and Seref SAGIROGLU. "Detecting Remote to Local User Attacks with High Ratio."

20. Mohebali, Behshad, et al. "Probabilistic neural networks: a brief overview of theory, implementation, and application." Handbook of probabilistic models (2020): 347-367.

21. Jeatrakul, P., Wong, K.W., Fung, C.C. (2010). Classification of Imbalanced Data by Combining the Complementary Neural Network and SMOTE Algorithm. In: Wong, K.W., Mendis, B.S.U., Bouzerdoum, A. (eds) Neural Information Processing. Models and Applications. ICONIP 2010. Lecture Notes in Computer Science, vol 6444. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-17534-3_19

22. Maind, Sonali B., and Priyanka Wankar. "Research paper on basic of artificial neural network." International Journal on Recent and Innovation Trends in Computing and Communication 2.1 (2014): 96-100.

23. Macukow, Bohdan. "Neural networks–state of art, brief history, basic models and architecture." Computer Information Systems and Industrial Management: 15th IFIP TC8 International Conference, CISIM 2016, Vilnius, Lithuania, September 14-16, 2016, Proceedings 15. Springer International Publishing, 2016.

24. М.А. Новотарський, Б.Б. Нестеренко. Штучні нейронні мережі: обчислення // Праці Інституту математики НАН України. – Т50. – Київ: Ін-т математики НАН України, 2004. – 408 с.

25. Добровська, Людмила Миколаївна, and Ірина Арбіківна Добровська. "Теорія та практика нейронних мереж." (2015).

26. Ткаліченко, Сергій Володимирович. "Штучні нейронні мережі: навчальний посібник." (2023).

27. Субботін, Сергій Олександрович, and Сергей Александрович Субботин. "Нейронні мережі: теорія та практика." (2020).

28. Терейковський, Ігор Анатолійович, Денис Антонович Бушуєв, and Людмила Олексіївна Терейковська. "Штучні нейронні мережі: базові положення." (2022).

29. Sharma, Sagar, Simone Sharma, and Anidhya Athaiya. "Activation functions in neural networks." Towards Data Sci 6.12 (2017): 310-316.

30. Park, Y-S., and S. Lek. "Artificial neural networks: Multilayer perceptron for ecological modeling." Developments in environmental modelling. Vol. 28. Elsevier, 2016. 123-140.

31. Popescu, Marius-Constantin, et al. "Multilayer perceptron and neural networks." WSEAS Transactions on Circuits and Systems 8.7 (2009): 579-588.

32. Sharkawy, Abdel-Nasser. "Principle of neural network and its main types." Journal of Advances in Applied & Computational Mathematics 7 (2020): 8-19.

33. Ladlani, Ibtissem, et al. "Modeling daily reference evapotranspiration (ET 0) in the north of Algeria using generalized regression neural networks (GRNN) and radial basis function neural networks (RBFNN): A comparative study." Meteorology and Atmospheric Physics 118 (2012): 163-178.

34. Jeatrakul, Piyasak, and Kok Wai Wong. "Comparing the performance of different neural networks for binary classification problems." 2009 Eighth International Symposium on Natural Language Processing. IEEE, 2009.

35. Mell, Peter; Bace, Rebecca, “NIST Special Publication on Intrusion Detection Systems”, Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18 298-102.

36. B. Abdullah, I. Abd-alghafar, Gouda I. Salama & A. Abdalhafez, (2009) “Performance Evaluation of a Genetic Algorithm Based Approach to Network Intrusion Detection System”, 13th International Conference on Aerospace Sciences and Aviation Technology (ASAT), May 26-28.

37. J. Gomez & D. Dasgupta, (2002) “Evolving Fuzzy Classifiers for Intrusion Detection”, IEEE Proceedings of the IEEE Workshop on Information Assurance, United States Military Academy, West Point, NY.

38. R. H. Gong, M. Zulkernine & P. Abolmaesumi, (2005) “A Software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection”, Proceedings of the 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing.

39. T. Xia, G. Qu, S. Hariri & M. Yousif, (2005) “An Efficient Network Intrusion Detection Method Based on Information Theory and Genetic Algorithm”, Proceedings of the 24th IEEE International Performance Computing and Communications Conference, Phoenix, AZ, USA.

40. A. S. Mohammad and Z. Mohammad, “Efficacy of Hidden Markov Models over Neural Networks in Anomaly Intrusion Detection,” 30th Annual International Computer Software and Applications Conference, Chicago, 2006, pp. 325-332.

41. Asmaa Shaker Ashoor (Department computer science, Pune University) Prof. Sharad Gore (Head department statistic, Pune University),” Importance of Intrusion Detection System (IDS)”, International Journal of Scientific & Engineering Research, Vol. 2, Issue 1, Jan 2011.

42. K. Ilgun, R. A. Kemmerer and P. A. Porras, “State Transition Analysis: A Rule-based Intrusion Detection Approach,” IEEE Transactions on Software Engineering, Vol. 21, No. 3, March 1995, pp. 181-199. doi: 10.1109/32.372146

43. Alireza Osareh, Bita Shadgar (Computer Science Department, Faculty of Engineering, Shahid Chamran University, Ahvaz, Iran), ”Intrusion Detection in Computer Networks based on Machine Learning Algorithms”, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.11, November 2008.

44. A. Abraham, R. Jain and J. Thomas, “D-SCIDS: Distributed soft computing intrusion detection system,” Journal of Network and Computer Applications, vol. 30, pp. 81-98, 2007.

45. James Cannady, Jay Harrell,” A Comparative Analysis of Current Intrusion Detection Technologies”.

46. Vangie Beal,” Intrusion Detection (IDS) and Prevention (IPS) Systems”, posted 2005 [07-15-2005], last updated 2010[08-31-2010].

47. Indraneel Mukhopadhyay, Mohuya Chakraborty, Satyajit Chakrabarti (Department of Information Technology, Institute of Engineering & Management, Kolkata, India)“A Comparative Study of Related Technologies of Intrusion Detection & Prevention Systems”, Journal of Information Security, 2011, 2, 28-38 doi:10.4236/jis.2011.21003 Published Online January 2011

48. Uppal, Hussain Ahmad Madni, Memoona Javed, and M. Arshad. "An overview of intrusion detection system (IDS) along with its commonly used techniques and classifications." International Journal of Computer Science and Telecommunications 5.2 (2014): 20-24.

49. Liu, H.; Lang, B. Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *Appl. Sci.* **2019**, *9*, 4396. https://doi.org/10.3390/app9204396

50. Rusk, Nicole. "Deep learning." Nature Methods 13.1 (2016): 35-35.

51. Петренко, Т. А., and М. С. Титаренко. "Проблеми використання бази ознак мережевих атак KDD-99 в інтелектуальних системах виявлення вторгнень." (2019).

ДОДАТОК А

```
# Навчання нейромережі
# Імпорт потрібних бібліотек
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
# Визначення колонок як 41 параметр датасету
columns = (['duration'
,'protocol_type'
,'service'
,'flag'
,'src_bytes'
,'dst_bytes'
,'land'
,'wrong_fragment'
,'urgent'
,'hot'
,'num_failed_logins'
,'logged_in'
,'num_compromised'
,'root_shell'
,'su_attempted'
,'num_root'
,'num_file_creations'
,'num_shells'
,'num_access_files'
,'num_outbound_cmds'
,'is_host_login'
,'is_guest_login'
,'count'
,'srv_count'
,'serror_rate'
,'srv_serror_rate'
,'rerror_rate'
,'srv_rerror_rate'
,'same_srv_rate'
,'diff_srv_rate'
,'srv_diff_host_rate'
,'dst_host_count'
,'dst_host_srv_count'
,'dst_host_same_srv_rate'
,'dst_host_diff_srv_rate'
,'dst_host_same_src_port_rate'
,'dst_host_srv_diff_host_rate'
,'dst_host_serror_rate'
,'dst_host_srv_serror_rate'
,'dst_host_rerror_rate'
,'dst_host_srv_rerror_rate'
,'Attack_type'])
```

```
# Завантаженя датасету
df = pd.read_csv(r'E:\kdd99\kddcup.data_10_percent_corrected', header=None, names=columns)
# Виділяємо таблицю з нормальними даними
df_normal = df[df['Attack_type'] == 'normal.'].head(10000)
df_normal = df_normal.drop(columns=['Attack_type'])
df_normal = df_normal.assign(r2l_attack=0, u2r_attack=0, normal=1)
# Виділяємо таблицю з r2l атаками
r2l_attacks = ['ftp_write.', 'guess_passwd.', 'imap.', 'multihop.', 'phf.', 'spy.', 'warezclient.',
'warezmaster.']
df_r2l = df[df['Attack_type'].isin(r2l_attacks)]
df_r2l = df_r2l.drop(columns=['Attack_type'])
df_r2l = df_r2l.assign(r2l_attack=1, u2r_attack=0, normal=0)
# Виділяємо таблицю з u2r атаками
u2r_attacks = ['buffer_overflow.', 'loadmodule.', 'perl.', 'rootkit.']
df_u2r = df[df['Attack_type'].isin(u2r_attacks)]
df_u2r = df_u2r.drop(columns=['Attack_type'])
df_u2r = df_u2r.assign(r2l_attack=0, u2r_attack=1, normal=0)
# Обєднання в одну таблицю
df_data = pd.concat([df_normal, df_r2l, df_u2r])
# Перетворюємо категорійні дані числовий формат
for col in df_data.columns:
    if(df_data[col].dtypes=="object"):
        encoded=LabelEncoder()
        encoded.fit(df_data[col])
        df_data[col]=encoded.transform(df_data[col])
# Задаємо базові значення для навчання
train_size, confident_limit, doubtful_limit = 0.75, 0.1, 0.5
dataset_inputs, dataset_outputs = 41, 3
dataset_rows = len(df_data)
# Задаємо вхідні і вихідні значення
data_X = df_data.values[:, : dataset_inputs]
data_y = df_data.values[:, dataset_inputs : dataset_inputs + dataset_outputs]
print(f'Inputs: {dataset_inputs}, outputs: {dataset_outputs}, dataset size: {dataset_rows}')
# Розділяємо наш датасет на дані для навчання і для тестування нейромережі
from sklearn.model_selection import train_test_split

train_X, test_X, train_y, test_y = train_test_split(data_X, data_y, train_size=train_size,
                                    random_state=4, stratify=data_y, shuffle=True)

draft_train_rows, test_rows = len(train_X), len(test_X)

print(f'Total size: {dataset_rows}, draft train size: {draft_train_rows}, test size: {test_rows}')
# Встановлюється експериментально підібрана кількість шарів
lay_h1 = 46
lay_h2 = 32
lay_h3 = 12
print(f'Net config: i({dataset_inputs})-h1({lay_h1})-h2({lay_h2})-h3({lay_h3})-
o({dataset_outputs})')
# Побудова моделі
import torch.nn as nn
import torch
model = nn.Sequential(
```

```
    nn.Linear(dataset_inputs, lay_h1),
    nn.ReLU(),
    nn.Linear(lay_h1, lay_h2),
    nn.ReLU(),
    nn.Linear(lay_h2, lay_h3),
    nn.ReLU(),
    nn.Linear(lay_h3, dataset_outputs),
    nn.Sigmoid()
)
import torch.optim as optim
# функція втрат та оптимізатор
loss_fn = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.00456789)
import copy
from tqdm import tqdm
# Формування набору даних Train-Test
X_train = torch.tensor(train_X, dtype=torch.float32)
y_train = torch.tensor(train_y, dtype=torch.float32)
X_test = torch.tensor(test_X, dtype=torch.float32)
y_test = torch.tensor(test_y, dtype=torch.float32)
# Параметри навчання
num_epochs = 500  # кількість епох для проходження
batch_size = 300   # розмір кожної партії
batch_start = torch.arange(0, len(X_train), batch_size)
# Затримка найкращої моделі
best_mse = np.inf
best_model = None
history = []
# Візуалізація прогресу
pbar = tqdm(total=num_epochs)
# Тренувальний цикл
for epoch in range(num_epochs):
    model.train()
    pbar.set_description(f"Epoch {epoch}")
    for start in batch_start:
        # взяття партії
        X_batch = X_train[start:start+batch_size]
        y_batch = y_train[start:start+batch_size]
        # проходження вперед
        y_pred = model(X_batch)
        loss = loss_fn(y_pred, y_batch)
        # зворотнє проходження
        optimizer.zero_grad()
        loss.backward()
        # оновлення ваг
        optimizer.step()
    # обчислення точності в кінці кожної епохи
    model.eval()
    y_pred = model(X_test)
    mse = loss_fn(y_pred, y_test)
    mse = float(mse)
    history.append(mse)
```

```
    if mse < best_mse:
        best_mse = mse
        best_model = copy.deepcopy(model.state_dict())
    pbar.set_postfix({'Best MSE': f'{best_mse}'})
    pbar.update()
# Відновлення моделі і повернення найкращої точності
model.load_state_dict(best_model)

# Початок виконавчої програми використання навченої мережі

# Позначення шляху до збереженої навченої моделі
model_path = r'E:\model_df\model3.pth'
# Завантаження навченої моделі
model.load_state_dict(torch.load(model_path))
# Завантаження датасету для перевірки навченої нейромережі
df_example = pd.read_csv(r'E:\kdd99\kddcup.data_10_percent_corrected', header=None,
names=columns)
# Побудова таблиці для тесту нейромережі
df_normal_ex = df_example[df_example['Attack_type'] == 'normal.'].head(10000)
df_normal_ex = df_normal_ex.drop(columns=['Attack_type'])

r2l_attacks = ['ftp_write.', 'guess_passwd.', 'imap.', 'multihop.', 'phf.', 'spy.', 'warezclient.',
'warezmaster.']
df_r2l_ex = df_example[df_example['Attack_type'].isin(r2l_attacks)]
df_r2l_ex = df_r2l_ex.drop(columns=['Attack_type'])

u2r_attacks = ['buffer_overflow.', 'loadmodule.', 'perl.', 'rootkit.']
df_u2r_ex = df_example[df_example['Attack_type'].isin(u2r_attacks)]
df_u2r_ex = df_u2r_ex.drop(columns=['Attack_type'])

df_data_ex = pd.concat([df_normal_ex, df_r2l_ex, df_u2r_ex])
# Перетворення категорійних даних в числові
protocol_type_mapping = {'tcp': 1, 'udp': 2, 'icmp': 0}
service_mapping = {'http': 7, 'ftp_data': 6, 'smtp': 12, 'ftp': 5, 'domain_u': 1, 'telnet': 13, 'finger': 4,
'ntp_u': 10, 'ecr_i': 3, 'auth': 0, 'other': 11, 'imap4': 8, 'eco_i': 2, 'login': 9}
flag_mapping = {'SF': 7, 'RSTO': 1, 'REJ': 0, 'S1': 4, 'RSTR': 2, 'SH': 8, 'S3': 6, 'S2': 5, 'S0': 3}

df_data_ex['protocol_type'] = df_data_ex['protocol_type'].map(protocol_type_mapping)
df_data_ex['service'] = df_data_ex['service'].map(service_mapping)
df_data_ex['flag'] = df_data_ex['flag'].map(flag_mapping)
# Перетворення вхідних даних DataFrame у формат PyTorch tensor
input_data = torch.tensor(df_data_ex.values, dtype=torch.float32)
# Переведення моделі в режим оцінки та обчислення вихідних даних
model.eval()
with torch.no_grad():
    output = model(input_data)
# Створення словника для відображення індексів на мітки класів
index_to_label = {0: 'r2l', 1: 'u2r', 2: 'normal'}
# Перетворення вихідних даних у мітки класів
output = output.numpy()
predicted_labels = [index_to_label[np.argmax(x)] for x in output]
# Обчислення кількості кожного типу атак та нормальних даних
```

```
attack_counts = pd.Series(predicted_labels).value_counts()
# Виведення кількості виявлених атак кожного типу та нормальних даних
print("Кількість виявлених атак кожного типу та нормальних даних:")
print(attack_counts)
```