

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО
СИСТЕМНОГО АНАЛІЗУ
Кафедра штучного інтелекту**

До захисту допущено:

В. о. завідувачки кафедри

_____ Ірина ДЖИГИРЕЙ

«___» _____ 2024 р.

Дипломна робота

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Системи і методи штучного
інтелекту»**

спеціальності 122 «Комп'ютерні науки»

на тему: «Методи штучного інтелекту в задачах інформаційної безпеки»

Виконав:

Студент IV курсу, групи КІ-02

Москаленко Максим Геннадійович

Керівник:

доцент кафедри ШІ, к. т. н.

Коваленко Анатолій Єпіфанович



Консультант з економічного розділу:

професор кафедри ЕК ФММ, д.е.н.,

Шевчук Олена Анатоліївна

Консультант з нормоконтролю:

фахівець першої категорії кафедри ШІ,

Кравець П.В.

Рецензент:

професор, д.т.н., Бідюк Петро Іванович

ПБідюк

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

Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 122 «Комп'ютерні науки»

Освітньо-професійна програма «Системи і методи штучного інтелекту»

ЗАТВЕРДЖУЮ

В. о. завідувачки кафедри
_____ Ірина ДЖИГИРЕЙ

«31» січня 2024 р.

ЗАВДАННЯ

на дипломну роботу студенту

Москаленку Максиму Геннадійовичу

1. Тема роботи «Методи штучного інтелекту в задачах інформаційної безпеки», керівник роботи доцент, канд. тех. наук Коваленко Анатолій Єпіфанович, затверджені наказом по університету від «31» травня 2024 р. №2240-С
2. Термін подання студентом роботи «10» червня 2024 року.
3. Вихідні дані до роботи: зображення, що містять текст.
4. Зміст роботи: дослідження предметної області, теоретичні основи застосування методів штучного інтелекту в задачах інформаційної безпеки, реалізація системи-експерта для визначення та протидії мережевим втручанням, функціонально-вартісний аналіз програмного продукту.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Шевчук Олена Анатоліївна, професор, д. е. н.		

7. Дата видачі завдання: «05» лютого 2024 року.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Вибір теми і постановка дослідження.	22.04.2024	виконано
2	Аналіз актуальності дослідження.	29.04.2024	виконано
3	Збір та аналіз літератури.	10.05.2024	виконано
4	Формування задачі дослідження.	15.05.2024	виконано
5	Розробка програмної частини.	25.05.2024	виконано
6	Аналіз та структурізація результатів.	30.05.2024	виконано
7	Робота над економічною частиною продукту.	10.06.2024	виконано
8	Оформлення пояснювальної записки.	10.06.2024	виконано

Студент

Максим МОСКАЛЕНКО

Керівник

Анатолій КОВАЛЕНКО

РЕФЕРАТ

Дипломна робота: 84 с., 16 рис., 6 табл., 37 посилань, 2 додатки.

ШТУЧНИЙ ІНТЕЛЕКТ, ІНФОРМАЦІЙНА БЕЗПЕКА, НЕЙРОННІ МЕРЕЖІ,
ВИЯВЛЕННЯ МЕРЕЖЕВИХ ВТРУЧАНЬ

Об'єкт дослідження – методи штучного інтелекту.

Предмет дослідження – використання штучного інтелекту для захисту даних.

Мета дослідження – проаналізувати існуючі методи штучного інтелекту для захисту даних. Запропонувати практичне рішення з застосування штучного інтелекту у прикладній задачі захисту даних.

Актуальність роботи обумовлена зростаючою складністю інформаційного простору, та складністю інформаційних атак, а також останніми досягненнями у галузі штучного інтелекту та збільшенні обчислювальних здатностей комп'ютерних систем.

Отримані результати: запропоноване рішення щодо використання глибинних нейронних мереж та систем-експертів для створення системи реагування на мережеві вторгнення, а саме DoS атаки.

У рамках подальшого розвитку програмного продукту пропонується підвищувати точність розпізнавання мережевих втручань, а також додавання нових правил для системи-експерта з метою більш якісної протидії мережевим втручанням.

ABSTRACT

Master's thesis: 84 p., 16 figures, 6 tables, 37 references, 2 appendices.

ARTIFICIAL INTELLIGENCE, INFORMATION SECURITY, NEURAL NETWORKS, DETECTION OF NETWORK INTRUSIONS

Research object: methods of artificial intelligence.

The subject of research: application of artificial intelligence methods for data protection.

Purpose: To analyze existing methods of artificial intelligence for data protection and propose a practical solution using artificial intelligence for applied data protection tasks.

The relevance of the work is due to the increasing complexity of cyberspace and cyberattacks, as well as recent advancements in the field of artificial intelligence and the enhancement of computational capabilities of computer systems.

Results obtained: A proposed solution involves the use of deep neural networks and expert systems to create a response system for network intrusions, specifically DoS attacks.

As part of the further development of the software product, it is proposed to enhance the accuracy of network intrusion detection and to add new rules for the expert system to improve the quality of counteracting network intrusions.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ	8
ВСТУП.....	9
РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ.....	10
1.1 Актуальність задачі використання методів штучного інтелекту в засадах інформаційної безпеки	10
1.2 Окреслення предметної області	11
1.3 Виклики інформаційної безпеки у сучасних реаліях.....	12
1.3.1 Розвиток природи кіберзагроз.....	12
1.3.2. Підвищення складності інформаційного середовища	12
1.3.3. Обмеження традиційної моделі безпеки даних	13
1.4 Типи комп'ютерних загроз	14
1.4.1 Зловмисне програмне забезпечення	14
1.4.2 Соціальна інженерія.....	15
1.4.3 Фішинг.....	15
1.4.4 DoS атаки.....	16
1.4.5 Розвинені сталі загрози (APT).....	16
1.5 Методи штучного інтелекту, що використовуються в задачах інформаційної безпеки	17
1.5.1 Машинне навчання та глибинне навчання	18
1.5.2 Моделі на основі природної обробки мови	19
1.5.3 Технологія представлення знань та концептуальне моделювання	20
1.5.4 Моделювання систем-експертів з захисту даних	24
1.5 Постановка задачі	26
Висновки до розділу 1	26
РОЗДІЛ 2 МАТЕМАТИЧНА МОДЕЛЬ СИСТЕМИ-ЕКСПЕРТА З БЕЗПЕКИ	28

	7
2.1 Загальний огляд систем-експертів	28
2.2 Використання нейронних мереж для навчання систем-експертів.....	29
2.2.1 Нейрон у глибинній нейронній мережі	29
2.2.2 Функція активації	30
2.2.3 Загальна схема глибинної нейронної мережі	31
2.2.4. Навчання глибинної мережі	32
2.2.5 Згорткова нейронна мережа	34
2.2.6 Пошук архітектури нейронної мережі	35
2.3 Процес судження	37
Висновки до розділу 2.....	39
РОЗДІЛ 3 ОГЛЯД РЕАЛІЗІЇ СИСТЕМИ-ЕКСПЕРТА ІЗ ЗАХИСТУ ДАНИХ	41
3.1 Опис системи	41
3.2 Обґрунтування вибору технологій.....	42
3.3 Навчання нейронної мережі	42
3.4 Заповнення бази знань.	44
Висновки до розділу 3.....	45
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ	46
4.1 Постановка задачі проектування.....	47
4.2 Обґрунтування функцій програмного продукту	47
4.3 Обґрунтування системи параметрів програмного продукту.....	50
4.4 Аналіз експертного оцінювання параметрів	53
4.5 Аналіз рівня якості варіантів реалізації функцій	58
4.6 Економічний аналіз варіантів розробки ПП	59
4.7 Вибір кращого варіанту ПП техніко-економічного рівня	65
Висновки до розділу 4.....	66
ВИСНОВКИ	67
ДОДАТОК А. РЕАЛІЗАЦІЯ СИСТЕМИ-ЕКСПЕРТА	73
ДОДАТОК Б ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ	78

ПЕРЕЛІК СКОРОЧЕНЬ

ІІІ – Штучний інтелект

DDoS attack – Distributed denial of service attack (розподілена атака на відмову в обслуговуванні)

APT – Advanced persistent threats (розвинені сталі загрози)

NLP – Natural language processing (обробка природної мови)

KRR – Knowledge representation and reasoning (представлення знань та суджень)

DNN – Deep neural network (глибинна нейронна мережа)

ReLU – Rectified linear unit (випрямлений лінійний вузол)

CNN – Convolutional neural network (згорткова нейронна мережа)

NAS – Neural architecture search (пошук архітектури нейронної мережі)

ENAS – Evolution neural architecture search (еволюційний пошук архітектури нейронної мережі)

ВСТУП

Інформаційна безпека є однією з ключових сфер, що визначають стабільність та ефективність функціонування сучасного цифрового суспільства. Постійний ріст обсягу даних, глобалізація інформаційних потоків та зростання складності кіберзагроз вимагають нових підходів до захисту інформаційних систем [1]. В цьому контексті методи штучного інтелекту стають незамінним інструментом для забезпечення надійної та проактивної кібербезпеки.

Методи штучного інтелекту, такі як машинне навчання, глибинне навчання, нейронні мережі, обробка природної мови та інші, демонструють високу ефективність у розв'язанні складних завдань, пов'язаних із виявленням аномалій, аналізом шкідливого програмного забезпечення, прогнозуванням кібератак та управлінням ризиками. Завдяки здатності до самонавчання та адаптації до нових загроз, ШІ значно підвищує оперативність та точність заходів з інформаційної безпеки.

У цій роботі розглядаються основні методи штучного інтелекту, що застосовуються в задачах інформаційної безпеки. У ній розглядаються сучасні підходи до виявлення та протидії кіберзагрозам, аналізується ефективність різних алгоритмів ШІ та їх практичне застосування у реальних умовах. Також буде проаналізовано поточний стан досліджень у цій галузі, практичні приклади застосування ШІ в інформаційній безпеці та перспективи розвитку цих технологій.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність задачі використання методів штучного інтелекту в засадах інформаційної безпеки

Так само як комп'ютерні науки масово і безповоротно змінили світ, кіберзлочинність змінила кримінальний світ. Торгівля наркотиками, людьми, крадіжки та пограбування різного масштабу, екологічні злочини, домашнє насильство, військові злочини та втручання у демократичні процеси – все це набуло нових форм та масштабів з появою інформаційних технологій[2][3]. Крім того, слід пам'ятати про злочини меншого масштабу, які загрожують конфіденційності, гідності, фінансовому становищу або навіть життю окремих осіб [4]. Тому кібербезпека є однією з ключових галузей комп'ютерних наук, оскільки вона дозволяє запобігати численним негативним наслідкам насильного втручання в інформаційне середовище.



Рисунок 1.1 – Зацікавленість у темі “Штучного інтелекту” на основі пошукових запитів

Технологія та ідея штучного інтелекту не нова, але на початках вона обмежувалась обмеженою кількістю даних та малими обчислювальними потужностями.

Проте зараз з розвитком обчислювальної техніки та підвищенням кількості інформації ситуація змінилась. Популярність технології штучного інтелекту останнім часом стрімко зростає (графік 1). У свою чергу це створює як вразливості у системі захисту, так і дозволяє зробити її більш ефективною [5]. Тому так важливо досліджувати та знаходити нові застосування, а також покращувати вже існуючі методи такої потужної технології.

1.2 Окреслення предметної області

Штучний інтелект – це набір технологій, розумних машин, а також алгоритмів, що покликані виконувати різноманітні завдання, які потребують аналізу та розуміння ситуації на рівні з людиною. Іншими словами дані системи та алгоритми мімікують під різноманітні функції людського мозку. Для ефективного функціонування систем на основі штучного інтелекту дані системи необхідно навчати. Основним джерелом інформації для штучний інтелект є зовнішні дані, які той використовує для покращення продуктивності та для підвищення якості виконуваної ним роботи. [6]

Кібербезпека або інформаційна безпека – це організація та набір ресурсів, процесів та структур спрямованих на захист кіберпростору та систем, що використовуються його, від виникнень випадків, які порушуються права власності користувачів. Дані права включають в себе доступ, зчитування, внесення, видалення та управління даними [7]. Додатково до кібербезпеки зараховують міри для запобігання та виявлення кіберзагроз, таких як зловмисні програми, мережеве шахрайство, програми-вимагачі і тд. Для досягнення цих цілей застосовують ряд методів, які також є частиною інформаційної безпеки: захист пристроїв, мереж, даних, навчання користувачів, розробка та застосування політик та процедур безпеки та проведення відповідних заходів при виникненні кібератаки [8].

Загалом ціллю кібербезпеки є забезпечення конфіденційності, цілісності та доступності інформації та її систем [9].

1.3 Виклики інформаційної безпеки у сучасних реаліях

1.3.1 Розвиток природи кіберзагроз

З перших днів появи комп'ютерних технологій, а особливо інтернету кіберпорушники постійно вдосконалювали свої інструменти та техніки для кібератак. І хоч з часом комп'ютерні системи стали невразливими до багатьох типів атак, все ж з'являються і розвиваються нові напрямки до яких класичній системи захисту даних складніше пристосуватись. Серед таких можна виділити: розвинені сталі загрози, поширення програм вимагачів, загрози Інтернету Речей, загрози хмарному середовищу і т. д [8]. Загалом швидкий розвиток технологій створює все нові і нові типи та вектори атак, а реалії сучасного світу вимагають їхнього моментального інтегрування в інформаційне середовище. Через це фахівці з кібербезпеки мають справу з великою кількістю нових та неперевіраних та потенційно вразливих технологій. В більшості випадків обмеженість людських та фінансових ресурсів не дозволяє вирішити цю проблему екстенсивно, тож чи не єдиним рішенням даної проблеми є використання штучного інтелекту.

1.3.2 Підвищення складності інформаційного середовища

Разом з тим як користувачі, компанії та уряди застосовують нові технології та збільшують об'єми використання даних, збільшується складність та розмір комп'ютерних мереж, а також їх взаємодій між собою.

Це призводить до того, що стає дуже важко контролювати та керувати такими мережами. Складність же полягає в асиметричності, анонімності, у наявності функціоналу подвійного призначення, географічному та політичному розташуванні мережової архітектури, тощо [10]. Разом з широким спектром застосувань подібних систем це також призводить до їхньої вразливості, що у свою чергу може бути використано кіберзлочинцями.

Іншим прикладом підвищення складності інформаційних середовищ є збільшення частоти використання програмного забезпечення від третіх сторін. Це можуть бути як і окремі самостійні сервіси (наприклад, платіжні сервіси або системи зберігання паролів) так і допоміжні програми (бібліотеки, плагіни, тощо). Тож виникає нова задача інформаційної безпеки, а саме запобігання проникненню в систему зі сторони програмного забезпечення від третіх сторін. Останній пункт особливо важливий у зв'язку з популярністю відкритого програмного забезпечення, так як використання подібних програм створює можливі вектори атаки навіть у закритому програмному забезпеченні [11].

1.3.3 Обмеження традиційної моделі безпеки даних

Традиційна модель безпеки даних, що заснована на напередвизначених правилах, що дозволяють знаходити та протидіяти кібератакам, має певні обмеження, що роблять її непридатною в сучасному інформаційному суспільстві. Такі системи не пристосовані до адаптації до нових загроз, адже традиційні моделі розроблені аби визначати та протидіяти атакам, що підпадають під певні критерії або правила. З появою нових типів вразливостей, традиційну систему безпеки необхідно модифікувати, що займає час та ресурси.

Іншою ж проблемою є швидкість та якість обробки даних – через вищевказані обмеження традиційна система може пропустити атаку або навпаки “захистити” від звичайної активності.

Наостанок, традиційні системи не є ефективними проти складних атак з використанням технології штучного інтелекту. Адже такі загрози мімікують поведінку справжньої людини або знаходять неочікуваний спосіб уникнення системи безпеки, що є повною протилежністю загрозам, проти яких створена більшість традиційних систем захисту даних.

1.4 Типи комп’ютерних загроз

1.4.1 Зловмисне програмне забезпечення

Зловмисне програмне забезпечення – це тип програмного забезпечення, що створене для завдання шкоди або примусового використання комп’ютерів та комп’ютерних мереж. Включає в себе віруси, хробаків, троянські програми, програми-вимагачі, програми для стеження, програми для показу реклами та інші. Поширення зазвичай здійснюється через електронні листи, інфіковані інтернет сайти або через вразливості стороннього програмного забезпечення. Після встановлення зловмисне програмне забезпечення може викрасти персональні дані, пошкодити чи видалити файли, пошкодити процеси в оперативній системі чи надати неавторизований доступ до мережі.

1.4.2 Соціальна інженерія

Соціальна інженерія – це метод кібератаки, що має на меті маніпуляцію людиною для доступу до системи або для отримання персональних та секретних даних. Зазвичай полягає в отриманні довіри жертви і може набувати різноманітних форм: через електронні листи, телефонні дзвінки, представленні себе як колеги або співробітника техпідтримки, створенні профілів у соцмережах або ж навіть вживу (наприклад, tailgating).

Зазвичай формами протидії соціальній інженерії є розвиток усвідомленості, скептицизму та загального рівня знань з інформаційної безпеки у потенційних жертв.

1.4.3 Фішинг

Фішинг – одна з поширених форм мережевого шахрайства та соціальної інженерії. Зазвичай порушник видає себе за представника певної організації для отримання делікатної інформації: доступу до банківських рахунків або паролів. Найбільш поширеним типом є надання жертві посилання на сайт з візуально схожим доменним ім'ям та оформленням, а також на якому присутня форма введення приватних даних.

1.4.4 DoS атаки

Атака на відмову в обслуговуванні (denial-of-service attack) – один з найбільш поширених типів атак, що спрямований в направленні великої кількості запитів або трафіку на мережу, сервер чи сайт з метою перевантаження доступних обчислювальних ресурсів. Метою DoS атаки є порушення нормального функціонування цільової системи або мережі.

Можливою протидією подібним типам атаки є наявність плану дій під час такої атаки, а також можливість системи швидко виявити початок атаки, заблокувати джерела надмірного трафіку або обмежити вже заражені ресурси та перенаправити туди весь надмірний трафік для зменшення завданої шкоди.

1.4.5 Розвинені сталі загрози (APT)

Розвинені сталі загрози - це комплексний вид кібератак, що здійснюється групою хакерів та зазвичай фінансується державами або великими організаціями. Метою даного типу атак є отримання доступу до захищених даних цільової установи, на яку скоюється напад. Даний тип атак відзначається сталістю так як може відбуватись впродовж тривалого проміжку часу для досягнення цілей.

Розвинені сталі загрози зазвичай складаються з декількох стадій та технік. Зазвичай це добре спланована операція з розвідкою, плануванням, використанням методів соціальної інженерії та спеціально розробленим програмним забезпеченням під конкретну ціль.

Зазвичай такі атаки проводяться для отримання розвідувальних даних, крадіжки інтелектуальної власності, проведення шпіонажу або порушення у функціонування критичної інфраструктури, як реальної так і мережевої.

Цілями такого типу атак є установи, які володіють цінною та секретною інформацією: урядові організації, військово-промислові концерни, фінансові інституції, наукові інститути та конструкторські бюро.

Захист від даного типу атак зазвичай найважчий, адже потребує створення сегментованих мереж, надійної системи доступу, періодичної перевірки та діагностики систем на вразливості та ознаки інфікованості, навчання співробітників щодо інформаційної безпеки. Також критичним для захисту від подібного типу атак є наявність систем здатних до збору інтелектуальних даних про кіберзагрозу (threat intelligence) та систем розвиненого виявлення кіберзагроз.

1.5 Методи штучного інтелекту, що використовуються в задачах інформаційної безпеки

Зважаючи на вищенаведені загрози, однією з можливих відповідей на дані виклики є використання штучного інтелекту в системах захисту даних. В даній роботі буде взято до уваги найбільш поширені ШІ технології, а саме: машинне навчання та глибинне навчання, технологію обробки природної мови (Natural language processing, NLP) та технологію представлення знань (Knowledge representation and reasoning, KRR). Моделі інформаційного захисту, що ґрунтуються на даних технологіях, здатні відповісти на більшість сучасних викликів, що постають перед фахівцями з інформаційної безпеки. [13]

1.5.1 Машинне навчання та глибинне навчання

Машинне навчання – це науковий підхід до створення алгоритмів та статистичних моделей, що використовуються комп'ютерними системами, для виконання завдань, на які ті не були спеціально запрограмовані. Існує багато підходів до створення моделей на основі машинного навчання. Проте в рамках даної роботи варто звернути уваги лише на деякі: кероване навчання, некероване навчання та глибинне навчання. [14]

Кероване навчання – метод машинного навчання, що полягає у виведенні функції для відображення вхідних даних у вихідні на основі заданих прикладів (пари вхід-вихід). Даний алгоритм потребує зовнішньої підтримки та великої кількості даних для навчання. Зазвичай використовується у задачах передбачення та класифікації.

Некероване навчання – метод машинного навчання, що як і метод навчання з підкріпленням, має на меті вивести функцію ввід-вивід. Але даний алгоритм не має набору навчальних даних, тож послуговується лише власними судженнями щодо природи даних. Зазвичай використовується у задачах кластеризації та зменшення розмірності вхідних даних.

Глибинне навчання – метод, що застосовується для створення обчислювальних моделей на основі штучних нейронних мереж з декількох шарів обробки даних. [15]

Загалом для захисту даних використовують методи машинного навчання на основі глибокого навчання, так як такий підхід є найбільш ефективним та має найширші можливості у використанні, хоч і складніший у реалізації. Ідея використання машинного навчання полягає в аналізі історичних даних, на кшталт поведінки мережі, активності бази даних, активності застосунків або поведінки користувача та визначені чи є така діяльність нормальною або ж варто вжити запобіжних дій.

Сфера практичного використання достатньо широка починаючи з визначення зловмисних програм і закінчуючи визначенням ризикованої поведінки користувача, що може призвести до фішингових або подібних атак. Також варто відмітити застосування машинного навчання для перевірки програмного коду на наявність поширених вразливостей або вразливостей нульового дня, так як даний підхід є особливо важливим у фінансових інституціях [16].

1.5.2 Моделі на основі природної обробки мови

Природна обробка мови – це галузь штучного інтелекту, що досліджує як комп'ютери здатні розуміти та обробляти людську мову. Дана галузь є однією з найбільш важливих у сфері штучного інтелекту, адже вона дає можливість комп'ютерам розуміти та обробляти людську мову, що значно покращує процес комунікації між людьми та розумними системами. Основна мета використання NLP – це отримання знань з великої кількості неструктурованих даних, тобто інтерпретація, дешифрування, розуміння та надання сенсу людському мовленню в найбільш сприятливий спосіб. Зазвичай аналіз даних методом природної обробки мови відбувається в три етапи: лексичний, синтаксичний та семантичний аналіз, хоча деякі задачі можна вирішити і без застосування всіх етапів [13].

Лексичний аналіз – метод природної обробки мови, що полягає у розмежуванні термінів, які необхідно описати та проаналізувати. Лексичний аналіз розбиває суцільний текст згідно критеріям на параграфи, речення, фрази, ключові слова і т.д.

Синтаксичний аналіз – один з ключових методів в природній обробці мови, що полягає у визначенні наскільки природна мова узгоджується з граматичними правилами. Найбільш поширеними техніками є лематизація, морфологічна сегментація, сегментація тексту, тегування частин мови, стемінг і так далі.

Семантичний аналіз – метод природної обробки мови, що полягає у визначенні контексту, значення слів та структури речень. За допомогою цього методу система здатна зчитати тон повідомлення, визначити метод написання та навіть вірогідність, що текст було створено методами штучного інтелекту.

Моделі захисту даних, що використовуються методи природної обробки мови є одними з найбільш вживаних та використовується в абсолютно різних сферах діяльності. Наприклад, методи синтаксичного аналізу використовуються для розрізнення підозрілих доменних імен (prlvat24.co) від звичайних та вартих довіри (cats.com) або ж більш розширена версія здатна проводити комплексний аналіз повідомлення, що може бути у вигляді електронного листа або web-сторінки, та класифікувати такі повідомлення як фішингові [17]. Іншим прикладом є застосування природної обробки мови для задач пов'язаних з інформаційною безпекою в соціальних мережах, до таких можна віднести визначення та запобігання атаки ботів, класифікації новин як фейкові та запобігання розсилці спаму користувачам.

1.5.3 Технологія представлення знань та концептуальне моделювання

Технологія представлення знань (Knowledge representation and reasoning) – область штучного інтелекту, що займається питаннями представленням даних у символічному вигляді та мисленевій обробці (reasoning) цих даних автоматичними системами. Мисленнєва обробка полягає в аналізі наскільки погляди, наміри та рішення агента можуть бути застосовані до автоматичної обробки та для подальшого вирішення складних проблем. [18] Найбільш прикладним застосування технології представлення знань є концептуальне моделювання, що полягає у створенні абстрактних моделей чи графічних представлень даних з ідей та концептів реального світу.

Загалом для побудови концептуальних моделей безпеки даних на основі знань використовують декілька основних методів представлення знань.

Логічне представлення – набір простих тверджень, що позбавлені двозначності, та можуть набувати значень “Істина” чи “Хибна”. Хоча логічні представлення дозволяють вибудовувати логічні обґрунтування та висновки, проте дані моделі можуть бути не настільки ефективними через обмеження, що їх накладає даний метод.

Семантично мережеве представлення застосовується для представлення даних у формі графа методом семантичних мереж. Дані мережі створені з об’єктів та дуг, що представляють ноди мережі та взаємодії між ними. Загалом дана технологія дозволяє структурно представити твердження про область діяльності і є найбільш природним представленням даних.

Фреймове представлення є похідним від семантичних мереж, єдиною різницею є використання фреймів замість об’єктів у вершинах графу. Фрейм – це об’єктно-орієнтована структура, що містить набір процедур, фактів (або даних) та інші фрейми [19]. В математичному представленні зазвичай зберігається у вигляді онтології множин.

Правила виробництва представлення даних у вигляді пари тверджень: умови та відповідної дії, що можна звести до твердження “Якщо умова, то дія”. Таким чином агент спершу перевіряє твердження умови і лише потім виконує відповідну дію, якщо значення умови задовільне. Основним застосуванням є надання “умови” для вирішення певної задачі та набором можливих рішень. Це дозволяє створити систему, що буде виконувати дію, яка найбільше підходить для вирішення проблеми згідно з умови.

Загалом знання для концептуальних моделей на основі знань можуть бути представлені в багато різних способів. Хоча ефективність даних методів може варіювати в залежності від природи даних та цільового застосунку. Наприклад, для задач інформаційної безпеки найпоширенішим представленням є використання онтологій безпеки (security ontologies).

Онтології в розумінні інформаційних технологій – це концептуальні моделі, що існують лише в певній галузі та представлені у формі, що зрозуміла для машин. У галузі комп'ютерної безпеки для онтологій можна виділити 5 основних понять або концептів: загроза, вразливість, атака, вплив та контроль [20].

- поняття загрози – представляє різноманітні типи небезпек або складнощів спрямованих на заданий набір значень безпеки;
- поняття вразливості – представляє вразливості та слабкі місця в системі захисту даних;
- поняття атаки – представляє різноманітні типи інцидентів безпеки, що можуть бути спричинені кібер порушниками;
- поняття впливу – представляє наслідки, до яких можуть призвести інциденти безпеки;
- поняття контролю – представляє набір відповідних механізмів, що може бути використаним для зменшення або уникнення ефектів інцидентів безпеки або для захисту вразливостей.

Використання цих понять в системі захисту даних можна структурувати наступним чином: кіберзагроза починає атаку або інцидент безпеки, що використовує вразливості системи, що створює певний вплив на систему. Механізм контролю відслідковує, запобігає або блокує атаку і таким чином захищає систему, уникаючи негативного впливу кібератаки.

На основі даних понять та їх відношень можливо побудувати концептуальну модель захисту даних, що здатна вирішувати складні проблеми та задачі з області кібербезпеки. На схемі 4 показана структура концептуальної моделі захисту даних на основі онтологій безпеки в системі захисту даних і відповідні потоки інформації від джерела даних до застосунку.

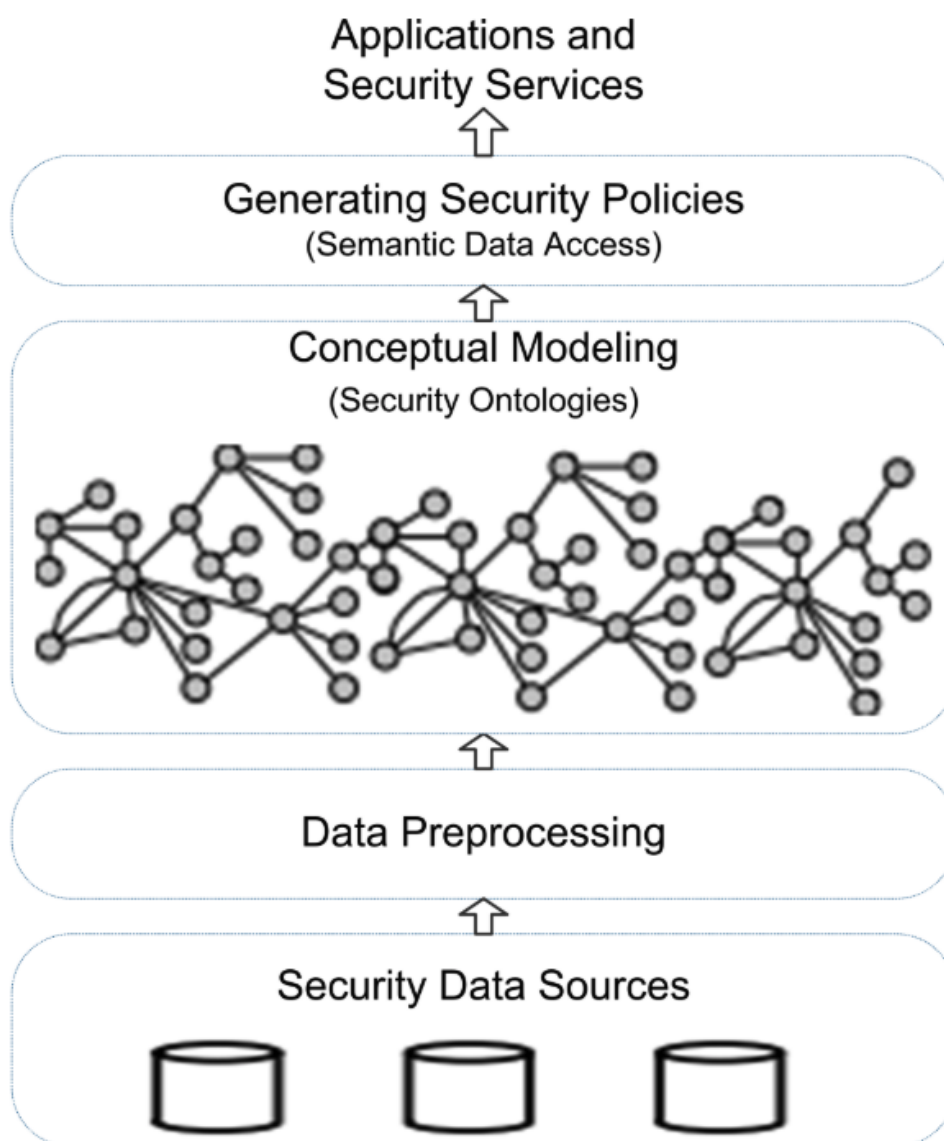


Рисунок 1.2 – Структура концептуального моделювання на основі онтологій безпеки в системі захисту даних [20]

Згідно цієї схеми можливо створити автоматичні політики безпеки з відповідної безпекової онтології, що будуть використовуватись в сервісах або застосунках з захисту даних. Таким чином така система здатна створювати розумні рішення згідно з поняттями та їх семантичними взаємозв'язками, що представленні у вигляді онтологій, на основі даних з різних джерел.

1.5.4 Моделювання систем-експертів з захисту даних

В контексті штучного інтелекту експертна система – це комп’ютерна система, що емулює здатність людини-експерта до прийняття рішень. Система-експерт із захисту даних – це тип експертної системи на основі знань та на основі правил з галузі комп’ютерної безпеки [21]. Така система зазвичай побудована на основі концептуальної моделі та використовує технологію представлення знань, тому такі системи розділені на дві основні частини: базу знань (knowledge base) та механізм судження (inference engine) та декілька допоміжних, але не менш важливих (рис. 2.2).

База знань містить всі необхідні знання для розуміння, формулювання та вирішення проблем. Ці знання являють собою специфічні для галузі знання, що зібрані людськими експертами в даній галузі. Зазвичай знання з цієї бази поділяють на два типи: фактичні та евристичні. Фактичними називають знання, що широко відомі та розповсюджені в даній галузі, такі знання можна знайти в довідниках, статтях, тощо. Евристичні знання є менш чіткими, і більше ґрунтуються на власному досвіді, оціночних судженнях і тд експертів із галузі.

Механізм судження є процесором або “мозком” системи-експерта, її основна функція це аналіз вхідних даних, правил з бази знань та створення методології або ж вирішення даної проблеми. Зазвичай правило складається з двох частин: антецедент (частина “якщо”) відома як стан або передумова та висновка (частина “то”) відома як дія або наслідок. Сукупність правил з бази знань створює т.з. ліс рішень, одним з важливих завдань механізму судження є побудова шляху через ліс рішень для отримання висновку для цього використовують один з двох підходів: прямий або зворотній вивід. Цікаво також розглянути тип правил, що використовуються в експертній системі, адже вони напряду визначають можливості та ефективність таких систем до прийняття рішень.

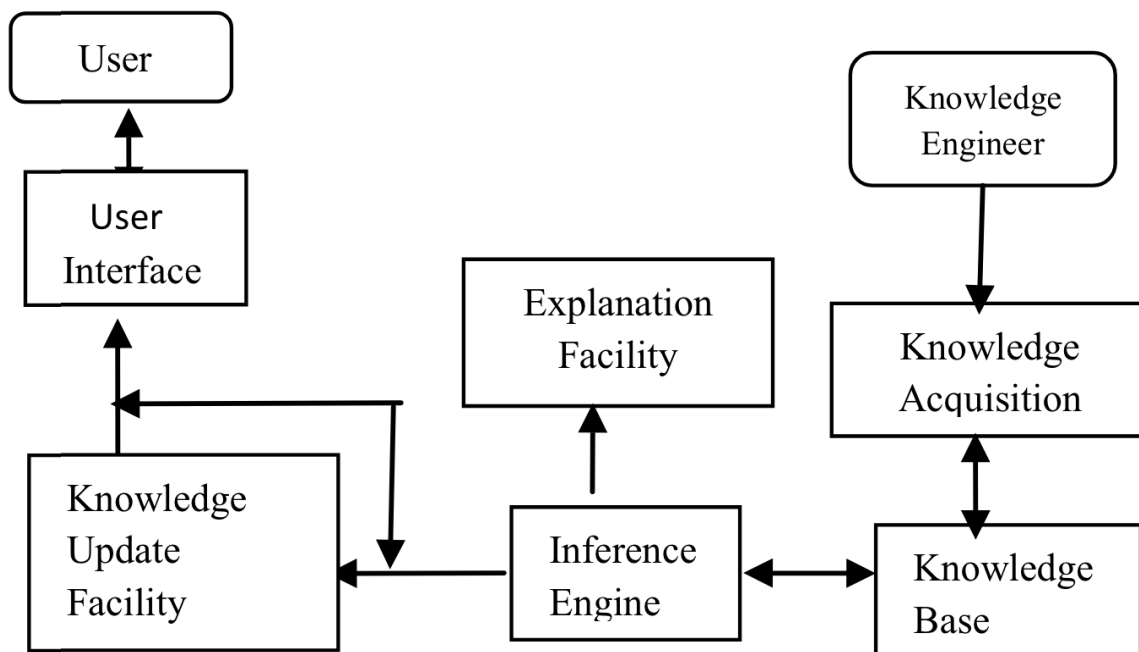


Рисунок 1.3 – Архітектура системи-експерта [21]

Правила навчання на основі класифікації – одна з найбільш відомих технологій, що може бути застосована в багатьох сферах та галузях. Багато з існуючих технік класифікації, таких як дерева прийняття рішень, IntrudTree, BehavDT можна використовувати для створення правил навчання.

Асоційовані правила навчання використовуються для пошуку цікавих відношень між змінними або найбільш поширених Якщо-То патернів у великих вибірках даних. З поширених алгоритмів можна навести: AIS, Apriori, FP-Tree, ABC-RuleMiner та інші.

Правила на основі нечіткої логіки (fuzzy logic). Нечітка логіка – це підхід у підрахунках, що використовує за основу “ступені довіри” замість звичних булевих значень “істина” або “хиба”. Правила на основі такої системи повертають неоднозначні, тобто не 0 або 1, результати, тим самим дозволяють створювати складніші системи з більшим варіантом вихідних дій або висновків.

Концептуально семантичні правила використовують онтології. Особливо ефективно у допоміжних галузях, таких як безпека.

Адже ефективність системи-експерта з безпеки може бути значно підвищена за рахунок онтологій з галузі, яку система-експерт “захищає”.

Таким чином моделі систем-експертів зі захисту даних на основі правил здатні мати можливості до прийняття рішення на рівні експерта з кібербезпеки, але у формі розумного фреймворку з захисту даних. Система створення правил грає значну роль у подальшій ефективності моделі, і дозволяє навчити її як найкраще виконувати конкретний набір завдань. Завдяки обчислювальній здатності та якості прийняття рішень, що не поступається людському, система-експерт з захисту даних є потужним, швидким та ефективним механізмом у архітектурі інформаційного захисту.

1.5 Постановка задачі

Задачею дипломної роботи є розробка програми для визначення та протидії мережевим втручань. Крім того, буде проведено дослідження застосування методів штучного інтелекту в задачах інформаційної безпеки, а саме буде детально розглянуто методи машинного навчання та побудови нейронних мереж для визначення мережових загроз. Додатково буде розглянуто способи задання систе-експертів для протидії знайденим мережевим втручанням.

Висновки до розділу 1

У розділі 1 визначено межі предметної області та розкрито основні поняття, що будуть використовуватися у подальших частинах роботи. Розглянуто виклики сучасної інформаційної безпеки, що вказують на неможливість подальшого

використання традиційних систем безпеки. Наведено приклади найбільш поширених кібератак для кращого розуміння сфери та причин застосування штучного інтелекту для їх вирішення. Крім того, проведено аналіз існуючих методів штучного інтелекту, надано їх опис та наведено приклади їх практичного застосування у сфері захисту даних.

РОЗДІЛ 2 МАТЕМАТИЧНА МОДЕЛЬ СИСТЕМИ-ЕКСПЕРТА З БЕЗПЕКИ

2.1 Загальний огляд систем-експертів

Одним з найбільш поширених та цікавих для дослідження застосуванням ІІІ в задачах інформаційної безпеки є система-експерт на основі механізму судження (inference engine). Дана машина здатна постійно аналізувати велику кількість джерел інформації, таких як трафік у мережі, поведінку користувачів, відхилення від встановлених або очікуваних патернів та інших, що можуть сигналізувати про потенційні загрози. Після виявлення таких загроз машина може проінформувати спеціалістів та ініціювати необхідні контр-заходи для уникнення загрози. Швидкість та точність виявлення загроз механізмом судження є критичною для захисту організації, зниження ризиків витоку даних та інших зловмисних активностей.

Як вже було зазначено у розділі 1, система-експерт складається з двох основних частин та кількох додаткових модулів. Основними частинами є база знань – набір фактів, які вносяться людьми, експертами з цільової області, а також факти, що створюються під час навчання машини. Додатково у таких системах до бази знань додаються модуль отримання знань (knowledge acquisition module), що дозволяє вносити факти у систему з різних джерел.

Іншим важливим компонентом системи-експерта є механізм судження – ядро системи-експерта, що займається інтерпретацією даних, виведенням висновків та створенням передбачень або потенційних дій. В основі даної технології лежать глибинні нейронні мережі (DNN), що дозволяють швидко та точно отримувати характеристики з великих масивів необробленої інформації, для представлення вхідної інформації.

2.2 Використання нейронних мереж для навчання систем-експертів

2.2.1 Нейрон у глибинній нейронній мережі

Глибинні нейронні мережі виникли як спроба імітації роботи мозку, тому в їхній основі ключовим елементом є “нейрон”, а сама глибинна нейронна мережа є сіткою з нейронів, що поєднанні шляхами, які називаються синапсами. Кожен нейрон містить декілька шляхів прийому даних (дендрити) та один шлях передачі даних (аксон). Сигнали, що передаються між синапсами називаються активаціями [30].

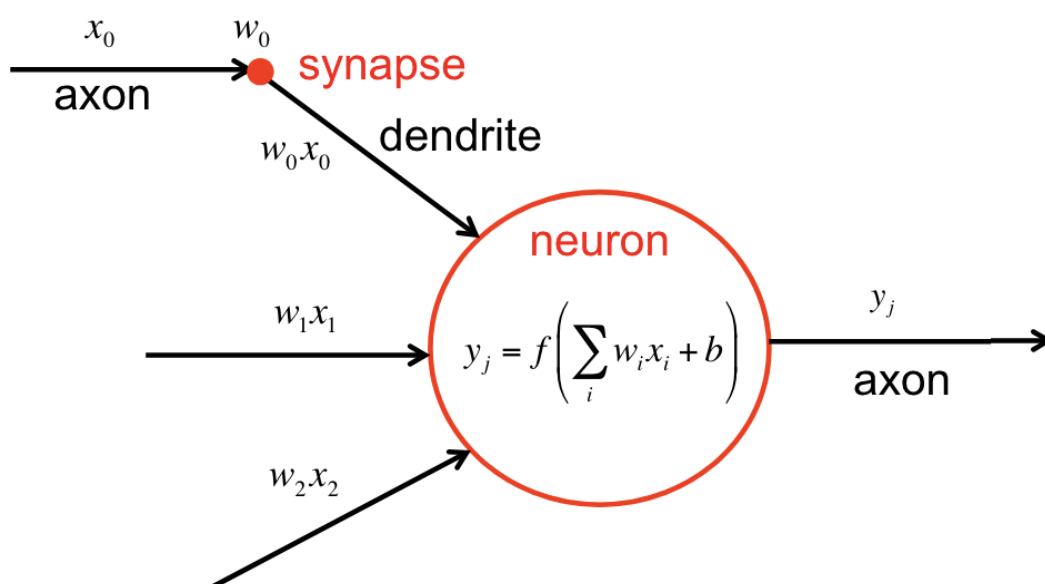


Рисунок 2.1 – Схема нейрона в глибинній нейронній мережі

2.2.2 Функція активації

Кожен з нейронів виконує певну нелінійну функцію, що зазвичай називають функцією активації σ (2.1):

$$a_j^l = \sigma(\sum_k w_{jk}^l a_k^{l-1} + b_j^l) \quad (2.1)$$

де a – значення активацій шару l ;

w – ваговий коефіцієнт;

b – вектор зміщення;

l – індекс поточного шару.

Для механізму судження зазвичай використовують випрямлену лінійну функцію (Rectified linear unit, ReLU) [22] $f(x) = \max(0, x)$. Дана функція є однією з найбільш популярних функцій активацій в нейронних мережах і на те є декілька причин: низька ресурсоємність, практична ефективність та подібність до біологічних аналогів [23].

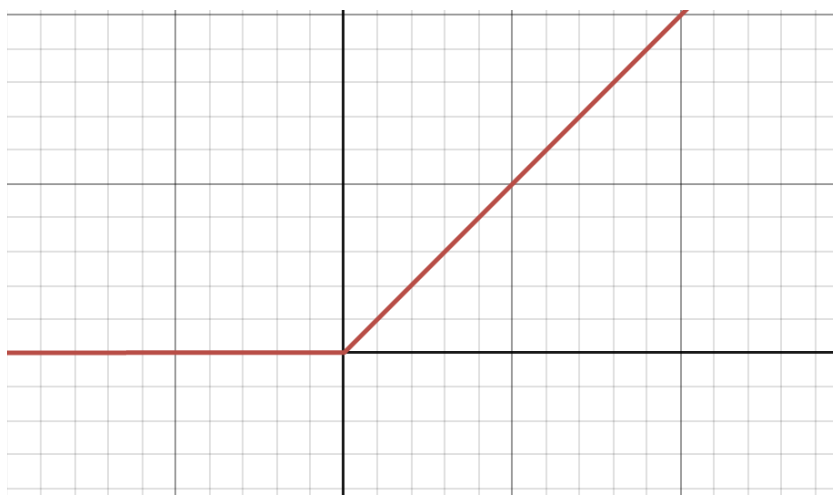


Рисунок 2.2 – Графік випрямленої лінійної функції активації

2.2.3 Загальна схема глибокої нейронної мережі

Як вже було зазначено, нейронна мережа складається з сітки нейронів, що поєднуються між собою за допомогою синапсів. Але нейрони поєднані не хаотично, а у групи, що по чергово обробляються дані. Загалом розділяють три види таких груп: вхідні нейрони, результуючі нейрони та приховані нейрони (рис. 2.3).

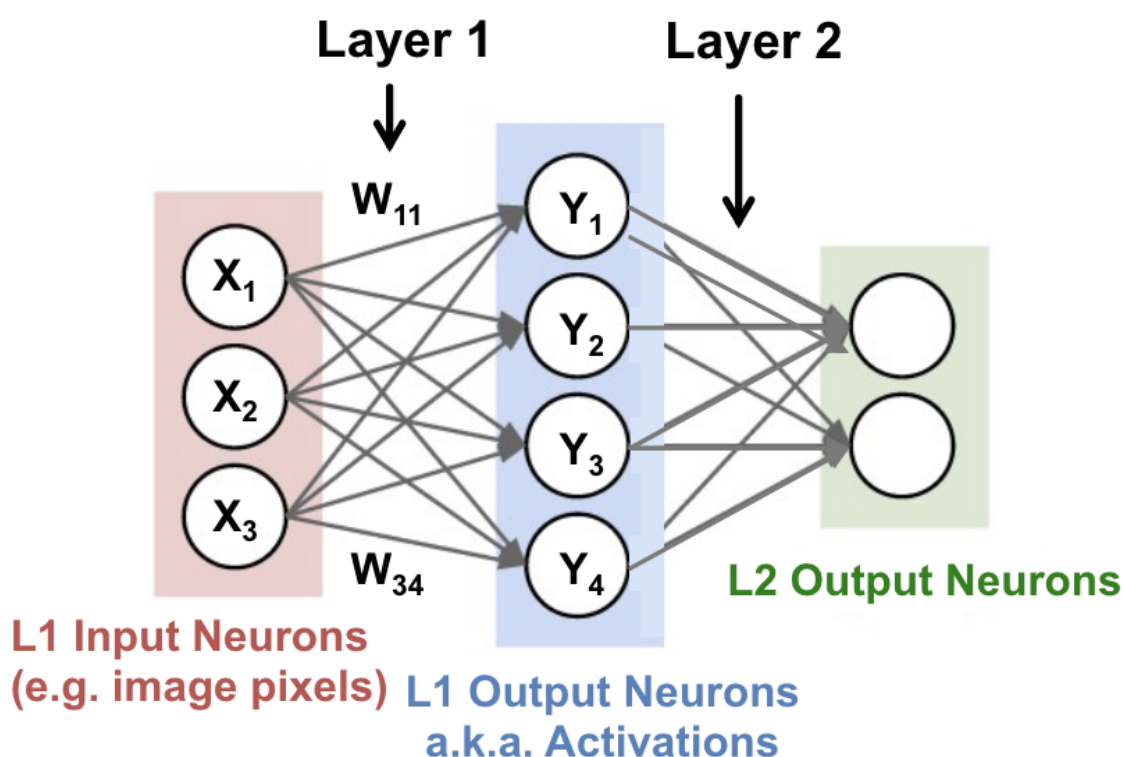


Рисунок 2.3 – Схематичний вигляд нейронної мережі [24]

Нейрони в групах з'єднані між собою за допомогою синапсів. Група синапсів між двома групами нейронів називається шаром (layer). Математично синапси можна виразити як коефіцієнти ваги для вхідних даних, тобто від налаштування синапсів залежить важливість даних вхідного шару для отримання даних з вихідного шару.

2.2.4 Навчання глибинної мережі

Навчання є ключовим аспектом в глибинних нейронних мережах, адже дозволяє налаштувати мережу на отримання максимально точного результату. Існує багато видів навчання нейронних мереж: кероване навчання, некероване навчання, навчання з підкріпленням і т. д. Але їх об'єднує спільна ідея зміни матриці вагів W та вектора зміщення b для якнайкращого виконання завдань мережею.

Наприклад, для керованого навчання використовується спеціальний набір даних, названий навчальною вибіркою, що складається з вхідних даних та очікуваного вихідного результату. Для система-експерта з інформаційної безпеки дана вибірка може складатись з певної поведінки системи (у вигляді системних логів, мережевої активності) як вхідних даних, та потенційної вразливості або проблеми як вихідних даних.

Ефективність глибинної нейронної мережі полягає у зменшенні втрат (loss, L) між передбаченнями або ж результатом роботи мережі на вхідних даних з навчальної вибірки та фактичними значеннями, що поставлені у відповідність до вхідних даних. Кінцевою метою є визначення такої матриці W та вектора b , які б давали найменше середнє відхилення на достатньо великій кількості даних. Існує багато способів обрахування втрат, одним з них є квадратична функція ціни:

$$L = \frac{1}{2n} \sum_x \|y(x) - a^L(x)\|^2.$$

Найбільш поширеним методом розв'язання даної проблеми є оптимізація нейронних мереж за допомогою градієнтного спуску. Градієнтом функції L у точці x називають вектор часткових похідних функції L у точці x .

$$\frac{\partial L}{\partial x} = \left[\frac{\partial L}{\partial x_1}, \frac{\partial L}{\partial x_2}, \dots, \frac{\partial L}{\partial x_n} \right].$$

Іншими словами градієнт визначає наскільки та має змінитись параметр x (в більшу чи меншу сторону) для мінімізації функції L . З цього можна вивести ітеративні формули підрахунку вагів та підрахунку зміщення:

$$w_{ij}^{t+1} = w_{ij}^t - \alpha \frac{\partial L}{\partial w_{ij}}.$$

$$b_i^{t+1} = b_i^t - \alpha \frac{\partial L}{\partial b_i}.$$

Для регулювання інтенсивності спуску використовується коефіцієнт α – темп навчання (learning rate).

Для обрахування часткових похідних функції у нейронних мережах застосовуються метод зворотного поширення помилки [26]. Даний метод базується на правилі диференціювання складеної функції, так як L не залежить напряду від w та b , але залежить від функції пошуку активацій поточного шару, що у свою чергу залежить від функцій пошуку активацій попереднього шару і тд. Іншими словами градієнт втрат від вагових коефіцієнтів для шару l залежить від значень активацій шару $l - 1$.

$$\frac{\partial L}{\partial w_{jk}^l} = a_k^{l-1} \delta_j^l \quad \delta_j^l = \frac{\partial L}{\partial a_j^l} \sigma'(z_j^l).$$

Значення δ_j^l називають локальним градієнтом або помилкою вихідного шару.

2.2.5 Згорткова нейронна мережа

Система-експерт із захисту даних в залежності від основної цілі використання використовує велику кількість даних. Це може бути системна та мережева інформація, текст введений людиною, зображення та звук (наприклад із камер спостережень), біометричні дані і так далі. У більшості випадків використання звичайної нейронної мережі потребуватиме неймовірних обчислювальних ресурсів та часу для тренування такої системи, що є дуже неефективним з економічної точки зору. [27]

Згорткова нейронна мережа складається з декількох типів шарів, крім звичних шарів входу та виходу, вона також містить згорткові, агрегувальні та повноз'єднані шари.

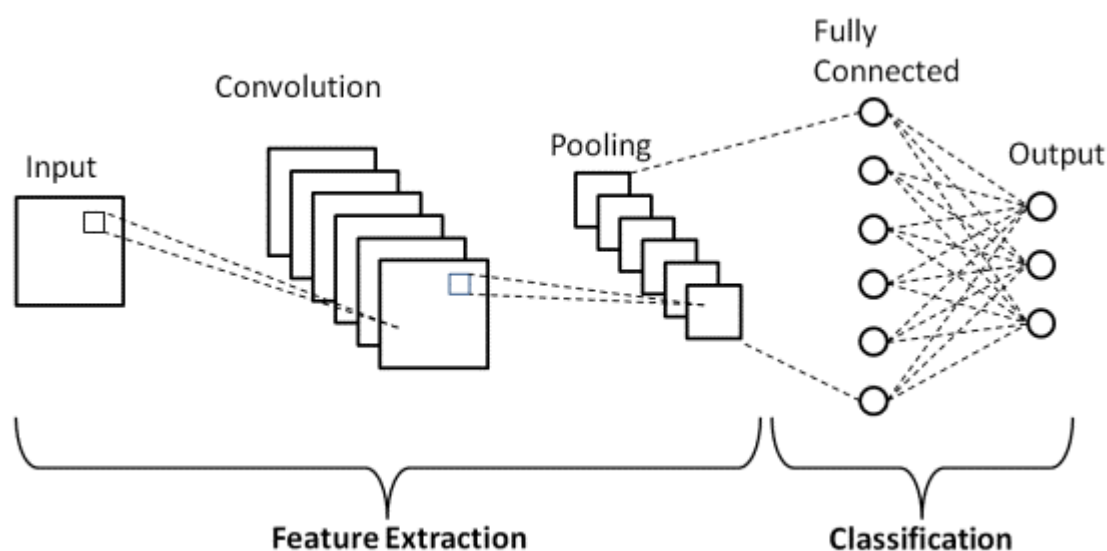


Рисунок 2.4 – Архітектура простої згорткової мережі

Згортковий шар відповідає за первинну обробку вхідних значень, дана обробка подібна до вищеприписаної пошуку активацій за допомогою ReLU функції (див. формула 1). Різницею між класичним ReLU шаром та згортковим є те, що згортковий шар обробляє дані лише зі своєї області (рецептивне поле).

Задачею агрегувального шару є децимація (downsampling) розмірності входу для подальшого зменшення числа параметрів. Повнозв'язний шар виконує роль аналогічну до прихованих шарів у класичних нейронних мережах.

Таким чином згорткові нейронні мережі здатні зменшити кількість та розмірність даних, які необхідно опрацювати. Це значно пришвидшує роботу систем-експертів з інформаційної безпеки, а також дозволяє використовувати більше даних для їхнього навчання, що у свою чергу покращує їхню ефективність.

2.2.6 Пошук архітектури нейронної мережі

Одним з ключових викликів класичної системи захисту даних є передвизначеність правил, за якими вона функціонує. Тобто якщо атака буде відбуватись зі сторони, яку не передбачили розробники, то система захисту виявиться неефективною. Звісно використання штучного інтелекту дозволяє системі автоматично знайти та оцінити можливі вектори атаки на великій вибірці даних. Проте проблема обмеженості, закладена на етапі розробки, все ж залишається. Адже архітектура глибинної нейронної мережі визначена людьми потребує постійного доопрацювання, а зміна кількості, формату чи якості даних може створити потребу у створенні абсолютно нової архітектури.

Одним з методів вирішення даної проблеми є використання методу пошук архітектури нейронної мережі (Neural Architecture Search, NAS) [28]. Даний метод є потужним інструментом, що дозволяє автоматизувати створення архітектур нейронних мереж. Більш того, архітектури розроблені даним методом є не лише більш адаптивними до типу вхідних даних та специфіки поставленої задачі, але й краще справляються з обробкою даних (аніж моделі створені людьми) [29].

Математично пошук архітектури нейронної мережі можна представити наступним чином:

$$\begin{cases} \arg \min_A = \mathcal{L}(A, D_{train}, D_{fitness}), \\ A \in \mathcal{A} \end{cases}$$

де A представляє простір потенційних архітектур;

$\mathcal{L}$ – функція для вимірювання продуктивності роботи архітектури нейронної мережі, зазвичай є недиференційованою та увігнутою;

$D_{fitness}$ – датасет на якому проводяться виміри ефективності архітектури після навчання на датасеті D_{train} .

Даний метод являє собою складну оптимізаційну задачу, так як її розв’язання потребує вирішення чималої кількості викликів: складні обмеження, дискретні представлення, наявність дворівневих структур, обчислювальна складність та конфлікти множинного вибору [28]. Тож використання даного методу потребує потужного оптимізаційного алгоритму для вирішення поставлених задач. Одним з таких є алгоритм еволюційного пошуку нейронної архітектури (ENAS).

Еволюційний пошук нейронної архітектури використовує еволюційні алгоритми, а саме: генетичні алгоритми, генетичне програмування та ройові алгоритми оптимізації.

Ідея еволюційного пошуку нейронної архітектури досить проста (але складна у реалізації). Спочатку ініціалізується вибірка, де кожна особина представляє собою окремо взятую архітектуру. Ефективність кожної з архітектур вимірюється за допомогою фітнес функції. Далі починається еволюційний процес пошуку архітектури, що зазвичай включає у себе основні етапи еволюційного пошуку: відбір, розмноження та мутацію. На виході створена архітектура є найбільш оптимальною серед усіх можливих архітектур простору.

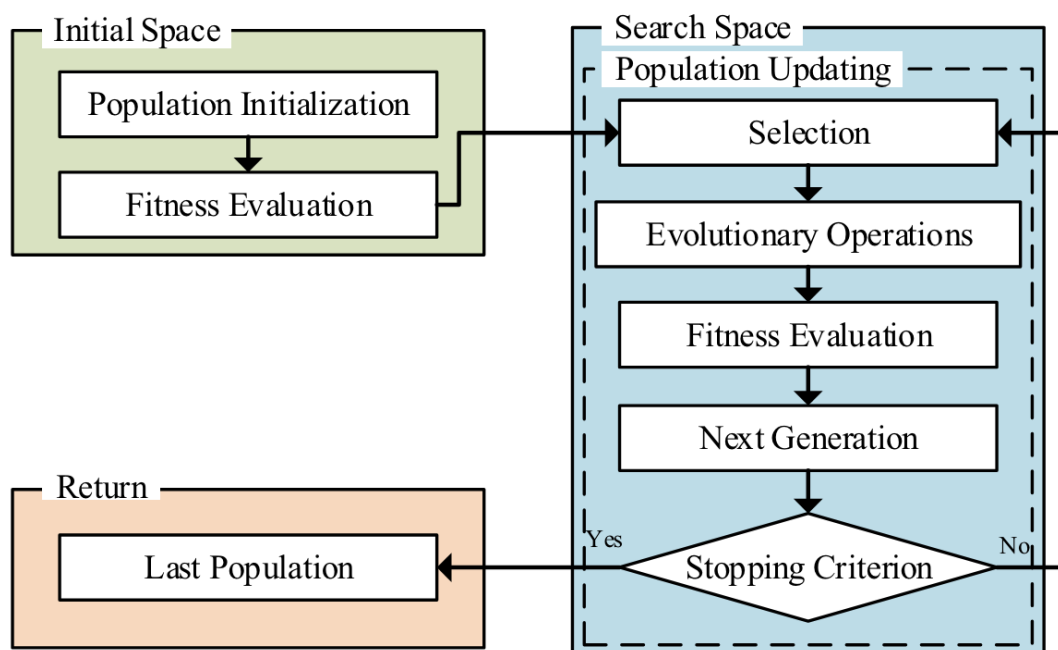


Рисунок 2.5 – Загальна схема алгоритму пошуку нейронної архітектури [2.8]

2.3 Процес судження

Основною задачею механізму судження є власне саме судження, що полягає у виведенні висновків на основі фактів та відношень з бази знань. Для експертних систем виділяють дві основні стратегії: прямого та зворотного виводу. Обидва методи однаково ефективні, хоч мають і протилежно різні підходи до розробки систем-експертів.

Прямий вивід (forward-chaining) є висхідним методом обробки даних. Він починає свій шлях з набору відомих фактів та використовує їх для створення нових правил. Даний процес продовжується поки не буде досягнуто кінцевої цілі або зникає можливість створювати нові факти [31].

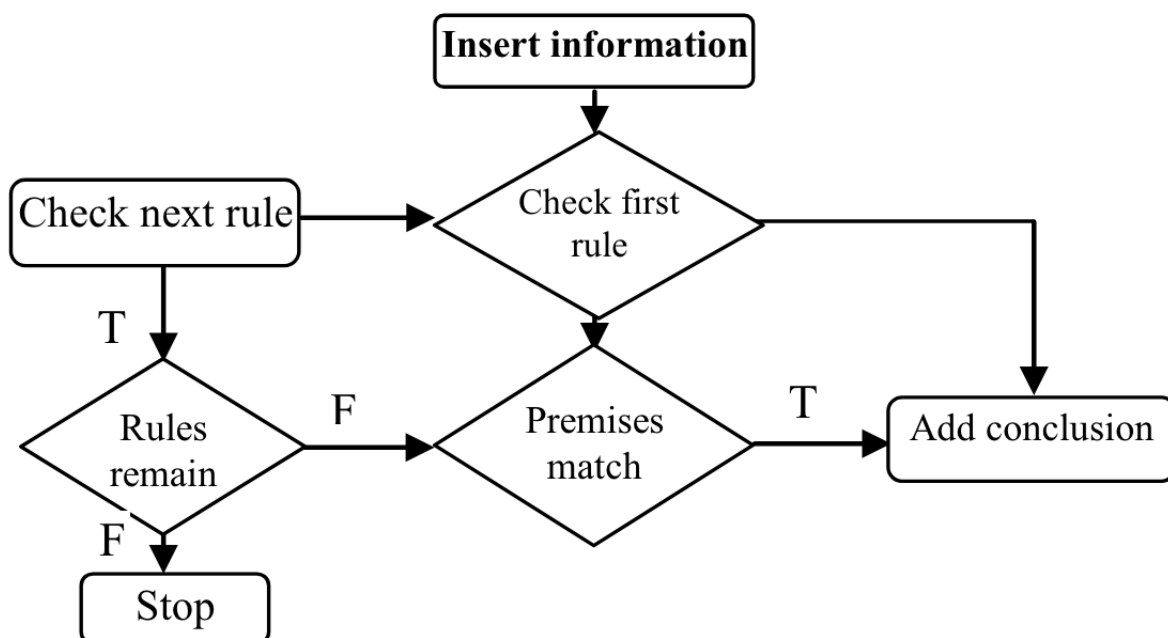


Рисунок 2.6 – Процес судження з прямим виводом [31]

Алгоритм прямого виводу може бути описано наступним чином:

- у систему надається інформація стосовно проблеми;
- механізм судження сканує перед визначені правила на наявність умови, що співпадає з інформацією;
- якщо механізм знаходить правило, то він додає цей висновок до робочої пам'яті;
- механізм продовжує сканування правил, поки всі співпадиння не будуть знайдені;
- після завершення процесу судження, механізм містить висновок щодо заданої інформації, а також нове правило.

Зворотній вивід (backward-chaining) є низхідним методом обробки даних, тобто припускається ціль і перевіряється чи умова задовольняє її. Алгоритм зворотного виводу наступний:

- у систему надається висновок, до якого потрібно дібрати правила або довести;

- механізм перевіряє чи не були висновки щодо цієї проблеми вже доведені або закладені спочатку;
- система переглядає правила в пошуках тих, висновки яких стосувались би цілі;
- у разі знаходження умови правила стають новими під-цілями, якщо такі правила були створені штучно, інакше умова (примітив) зберігається в пам'яті;
- система продовжує пошук поки не знайде усі примітиви до поставленої цілі та під-цілей.

Дані методи хоч і виглядають подібно проте застосовуються для вирішення різних типів проблем. Зворотній вивід застосовується до проблем, які можна легко сформулювати; для яких існує велика кількість правил, що співпадає з фактами поставленої проблеми; для яких інформація щодо проблеми відсутня або має бути отримана безпосередньо від джерела інформації. Прямий вивід застосовується для проблем, які мають велику кількість потенційних цілей, але лише декілька базових фактів; для яких більшість вхідних даних є умовами або для яких складно створити ціль чи гіпотезу [31].

Висновки до розділу 2

Технологія створення систем-експертів з інформаційної безпеки є досить обширною темою. Адже залежить від галузі застосування, формату вхідних даних та типу взаємодії з системою. У даному розділі розглянуто найбільш поширені методи та технології, що застосовуються для розробки таких систем. Глибинні нейронні мережі та похідні від них методи (наприклад, пошук архітектури нейронної мережі) дозволяють створити набір правил та закономірностей, які не були передбачені людьми-експертами з самого початку. Таким чином дана

технологія дозволяє створити систему з захисту даних, що не обмежена правилами, які в неї заклали люди на етапі розробки. А також дана технологія здатна пристосовуватись до змін у форматі вхідних даних, а її загальна архітектура дозволяє легко додавати нові джерела інформації або знань.

РОЗДІЛ 3 ОГЛЯД РЕАЛІЗІЇ СИСТЕМИ-ЕКСПЕРТА ІЗ ЗАХИСТУ ДАНИХ

3.1 Опис системи

Прикладом можливої реалізації системи-експерта із захисту даних є система розпізнавання DDoS атак (distributed denial-of-service attack). Опис даного типу атак було надано у першому розділі. Система складається з декількох компонентів: зчитувач логів, нейронна мережа для класифікації DoS атаки, буфер минулих результатів класифікації, машина висновування та база знань.

З даного опису можна виділити два основні компоненти.

- 1) Навчання нейронної мережі для класифікації типу DoS атак на основі вихідних даних (логів).
- 2) Програмування машини висновування для прийняття рішень щодо дій з уникнення або перешкодження атаки.

Схематично дану систему можна зобразити наступним чином:

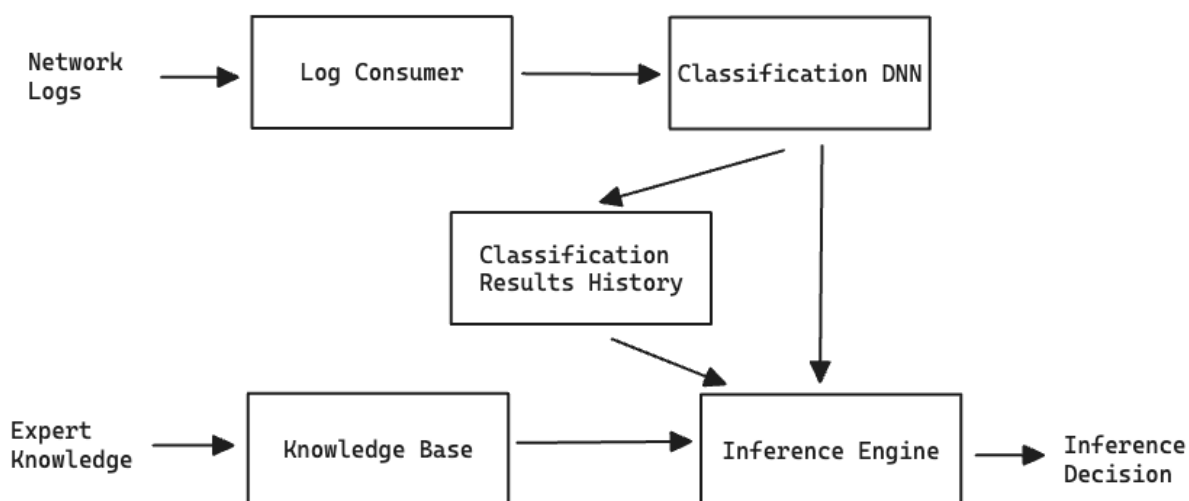


Рисунок 3.1 – Архітектура системи-експерта із захисту даних

3.2 Обґрунтування вибору технологій

Реалізація подібних систем найкраще підходить мова програмування Python, адже вона містить широкий та різноманітний набір інструментів для роботи з методами машинного навчання, тобто для побудови нейронних мереж, аналізу даних та створення систем-експертів. Серед інструментів мови Python варто виділити популярні бібліотеки NumPy для математичних розрахунків, PyTorch для роботи з нейронними мережами та Tensorflow для отримання датасету.

Для побудови системи-експерта рекомендовано використати готовий фреймворк PyKnow [35], що містить простий інтерфейс для створення експертних систем на основі мови Python.

3.3 Навчання нейронної мережі

В даній системі нейронна мереж використовується для класифікації системних логів та визначення вектору атаки. Загалом серед DoS атак виділяють декілька типів: Back, Land, Neptune, Pod, Smurf, Teardrop [32] вони відрізняються методами досягнення цілі, мережевими протоколами, на яких діють і тд.

Для навчання нейронної мережі було використано датасет kddcup99 є одним з найвідоміших датасетів для класифікації втручань в мережу. Даний датасет складається з набору даних підключень до мережі, які можна класифікувати за типом мережевого втручання [34]. Проте в рамках опису даного алгоритму буде використано лише дані, що стосуються DoS атак.

За основу архітектури нейронної мережі було взято мережу запропоновану Dong Binzhi (див. Додаток А) для роботи з датасетом kddcup99 [36].

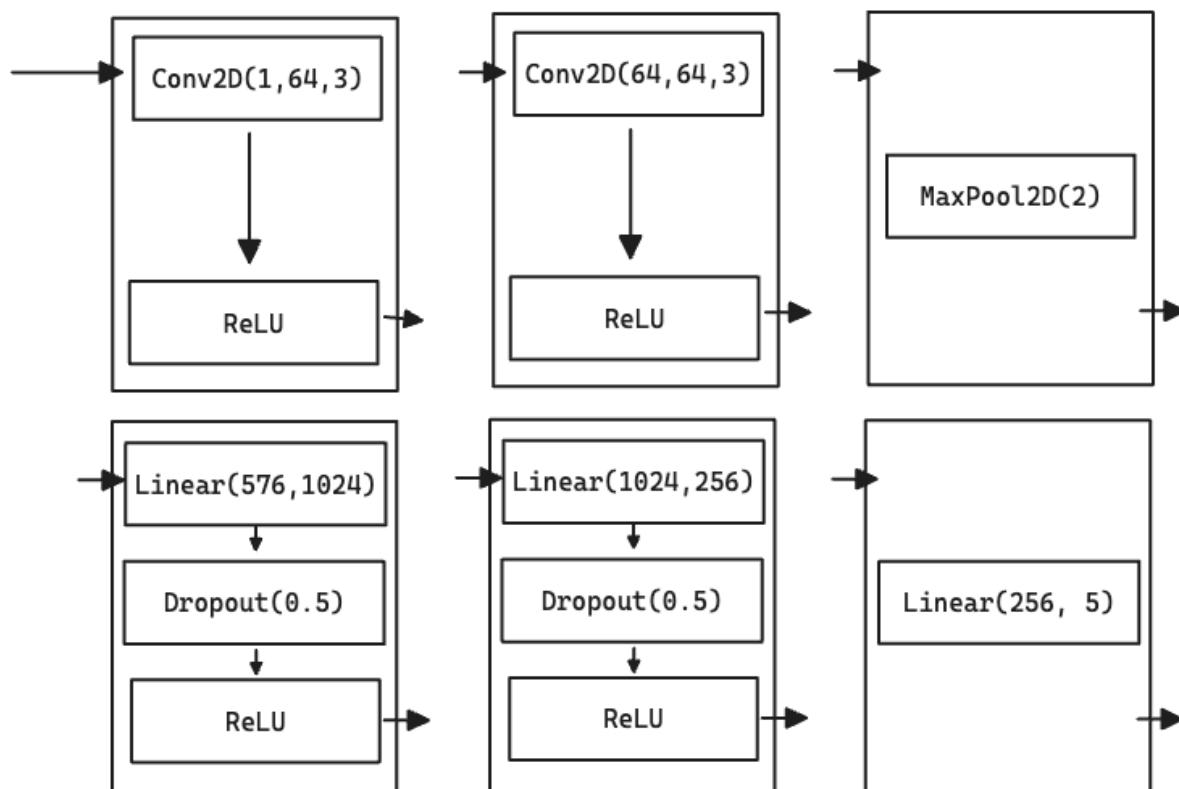


Рисунок 3.2 – Архітектура використаної нейронної мережі

Так як мережа використовує багато різноманітних шарів, для простоти її було розділено на 6 послідовних “контейнерів” (Sequential). Перші два послідовні шари (Conv2D) використовуються для зменшення обчислювальних потужностей та кількості параметрів, що обробляються. Далі застосовується MaxPool2D для зменшення розмірності даних та відкидання менш важливої інформації з попередніх шарів. Далі застосовуються повнозв’язні шари для об’єднання великої кількості вхідних даних у класифікаційне значення.

3.4 Заповнення бази знань

Для ефективної роботи системи-експерта із захисту даних необхідно задати початкові правила, за якими механізм судження буде визначати подальші дії. У даному прикладі імплементації це можуть бути дії в залежності від типу визначеної атаки, наприклад [36-37]:

- якщо виявлена Back атака, то почати ідентифікувати та ізолювати вражені системи;
- якщо виявлена Back атака, то увімкнути обмеження трафіку на уражених сервісах;
- якщо виявлена Back атака, то блокувати трафік зі зловмисних IP адрес;
- якщо виявлена Land атака, то блокувати трафік зі зловмисних IP адрес;
- якщо виявлена Neptune атака, то увімкнути SYN cookies;
- якщо виявлена Neptune атака, то обмежити кількість вхідних SYN пакетів;
- якщо Neptune атака завершена, то розпочати балансування трафіка між наявними ресурсами;
- якщо виявлена Pod атака, то виявлена атака на ICMP рівень;
- якщо виявлена атака на ICMP рівні, то обмежити ICMP трафік;
- якщо виявлена Smurf атака, то блокувати трафік направлений на широкомовні адреси (broadcast addresses);
- якщо виявлена Smurf атака, то виявлена атака на ICMP рівень;
- якщо виявлена Teardrop атака, то блокувати фрагментовані пакети;
- якщо виявлена будь-яка атака, то оновити систему до останньої версії;
- якщо необхідно оновити систему, то оновлювати її лише після припинення будь-яких атак;

- якщо виявлена будь-яка атака, то зберегти системні логи для подальшого аналізу;
- якщо нових атак не трапляється певний час, то повідомити про завершення атаки;
- якщо виявлено тип атаки, який вже відбувається, то не робити нових дій.

Звісно даний список правил не повний, та його можна розширювати, як з точки зору комп'ютерною безпеки, так і з точки зору бізнесу, тобто прописувати ряд правил притаманний конкретному прикладу використанню такої системи з реального світу.

Висновки до розділу 3

У даному розділі було розглянути практичний приклад використання штучного інтелекту в задачах інформаційної безпеки, а саме систему визначення мережових загроз та реагування на них.

Розробка даного продукту складається з двох частин: нейронної мережі для аналізу системної інформації та класифікації загроз, а також машини висновування для визначення подальших дій на основі вихідних даних. Для реалізації даного програмного продукту використано мову програмування Python та допоміжні бібліотеки, особливої уваги заслуговує пакет PyKnow для створення експертних систем.

Поданий приклад реалізації дозволяє досить швидко та просто створити систему-експерт із захисту, використовуючи перевірені інструменти, наприклад нейронну мережу для роботи з датасетом kddcup99.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

У цьому розділі буде проведено комплексну оцінку ключових характеристик майбутнього програмного продукту, призначеного для виявлення та запобігання мережевим втручанням. Цей програмний продукт буде розроблений таким чином, щоб сприяти всебічному дослідженню проблематики мережевої безпеки не лише в Україні, але й у всьому світі.

Для визначення та вибору оптимальної стратегії розробки програмного продукту, яка буде враховувати економічні фактори та забезпечувати сумісність з іншими програмними системами, буде використано метод функціонально-вартісного аналізу (ФВА).

ФВА – це технологія, яка дозволяє об'єктивно оцінити реальну вартість продукту або послуги, незалежно від організаційної структури компанії. Метою ФВА є виявлення резервів для зниження витрат за рахунок оптимізації виробничих процесів та досягнення найкращого співвідношення між функціональністю продукту та витратами на його розробку. Для проведення ФВА буде використано комплексну економічну, технічну та конструкторську інформацію.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих годин, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи реагування на мережеві втручання. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступними:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати системну інформацію, визначати та класифікувати мережеві втручання, а також приймати рішення на основі цих даних. Беручи за основу цю функцію, можна виділити наступні:

- F_1 – вибір мови програмування;
- F_2 – вибір основної бібліотеки для реалізації систем-експертів;
- F_3 – вибір середовища розробки.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1 : Python, C++. Функція F_2 : CLIPS, PyKnow. Функція F_3 : Visual Studio Code, Jupyter Notebook.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

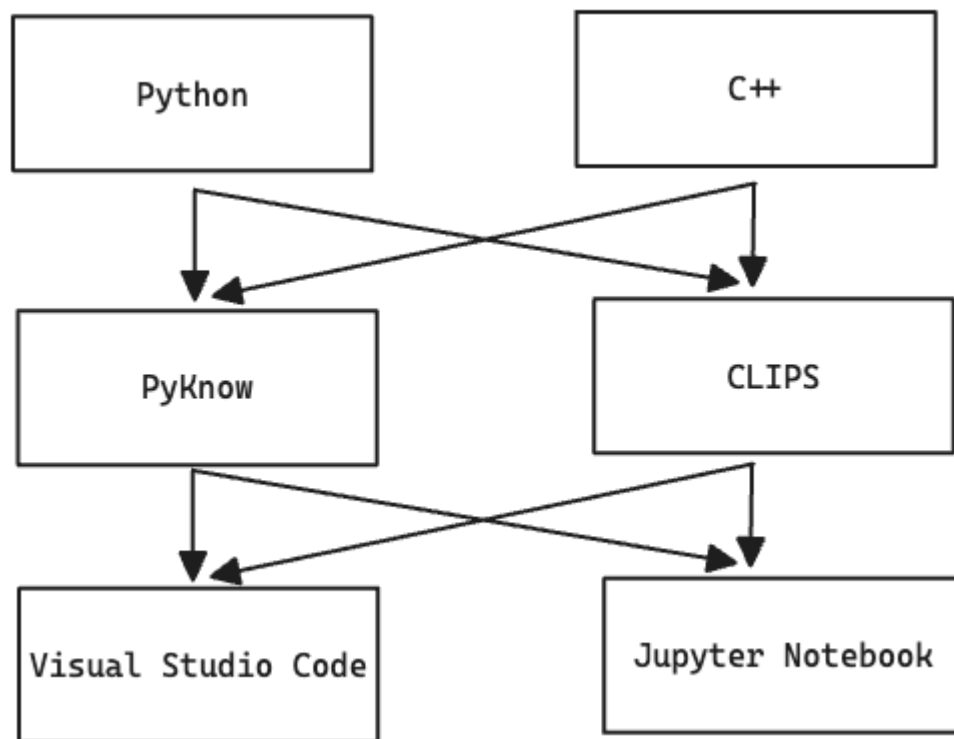


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

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

Таблиця 4.1 – Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	<i>Python</i>	Велика кількість бібліотек, висока швидкість і простота розробки програми	Низька швидкість роботи програми
	<i>C++</i>	Висока швидкість роботи програми,	Мала кількість готових бібліотек для роботи з нейронними мережами, складний синтаксис і великий час розробки
F_2	<i>CLIPS</i>	Складний інтерфейс, відсутність оновлень довгий час	Наявність статичної типізації
	<i>PyKnow</i>	Досить простий інтерфейс, нативна підтримка Python	Досить складно задати типізацію
F_3	<i>Visual Studio Code</i>	Велика кількість доповнень, гнучкість у налаштуваннях, інтелектуальне автозаповнення, підтримка багатьох мов програмування	Вибагливість до комп'ютера користувача
	<i>Jupyter Notebook</i>	Підтримка багатьох мов програмування, легкість використання	Мала кількість додаткових інструментів для полегшення розробки

Функція F_1 .

Перевагу надаємо швидкості розробки та простоті. Python має дуже велику кількість бібліотек і інструментів, які зроблять розробку програми простішим і комфортнішим.

Функція F_2 .

Обидва інструменти задовольняють вимогам до програми. Можливо використати варіанти А чи Б.

Функція F_3 .

Перевагу надаємо більшій кількості додатків та розширень Visual Studio Code, які можуть значно прискорити та спростити розробку програмного застосунку.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

$$F_{1a} - F_{2a} - F_{3a},$$

$$F_{1a} - F_{2b} - F_{3a}.$$

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

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих раніше, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

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

- $X1$ – швидкодія мови програмування;
- $X2$ – об'єм пам'яті для обчислень та збереження даних;
- $X3$ – час реагування на загрозу;
- $X4$ – потенційний об'єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2 – Основні параметри програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови	X1	оп/мс	9000	15000	18000
Об'єм пам'яті	X2	Мб	600	400	200
Час реагування на загрозу	X3	с	7	4	2
Потенційний об'єм програмного коду	X4	кількість рядків коду	1800	1400	1000

За даними таблиці 4.3 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

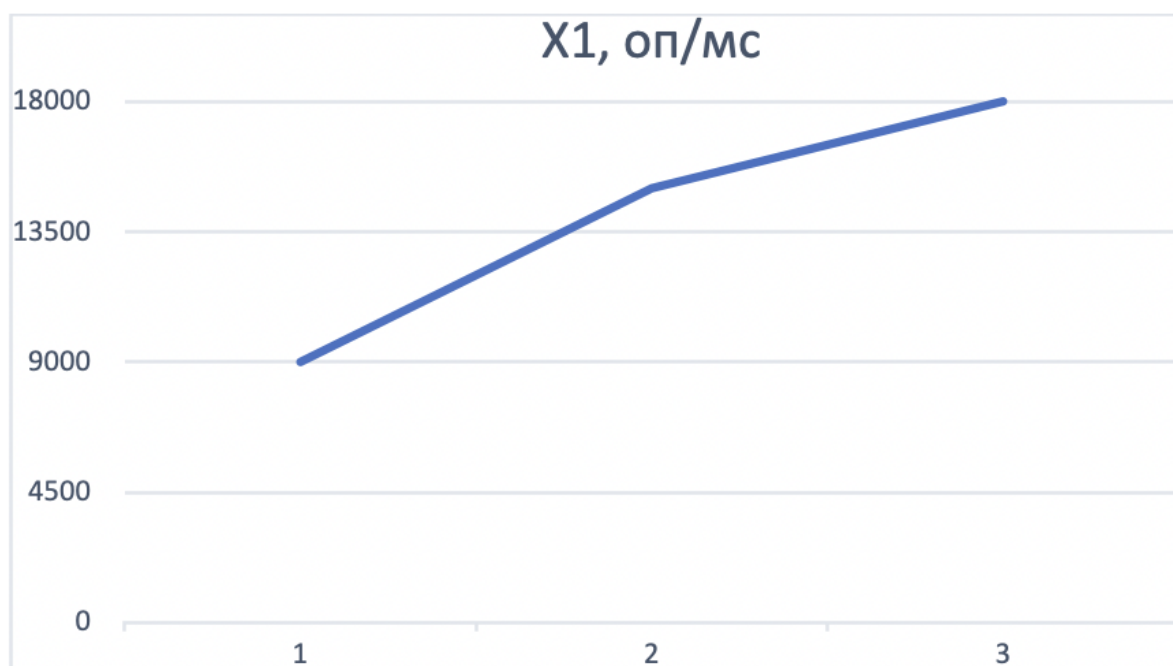


Рисунок 4.2 – X1, швидкодія мови програмування

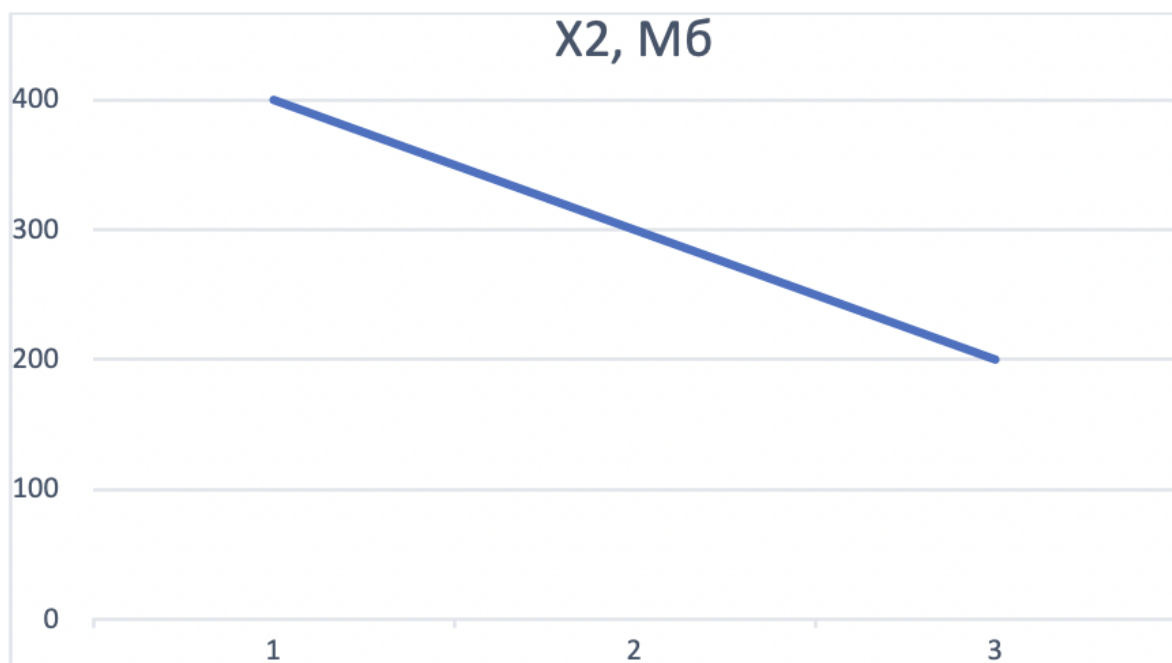


Рисунок 4.3 – X2, об'єм пам'яті

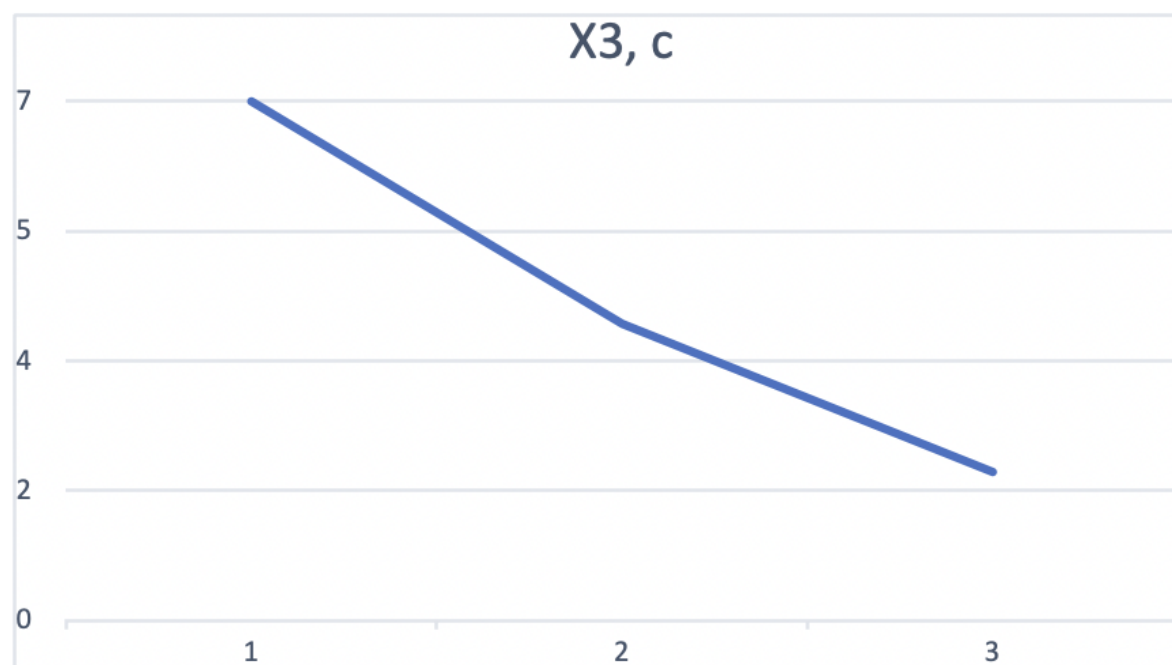


Рисунок 4.4 – X3, час реагування на загрозу

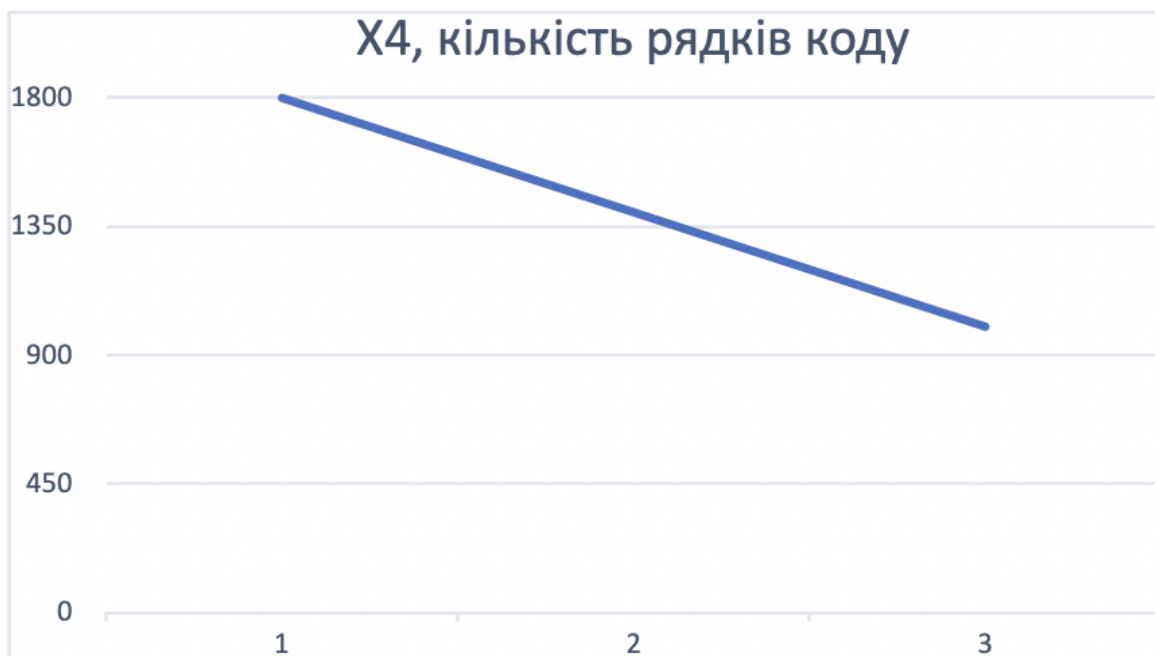


Рисунок 4.5 – X4, потенційний об'єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який визначає та протидіє мережевим втручанням у систему.

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

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;

- визначення оцінки попарного пріоритету параметрів;
- обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Результати ранжування параметрів

Позна-чення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
$X1$	Швидкість мови програмування	Оп/мс	1	2	2	1	2	1	2	1	6,5	42,25
$X2$	Об'єм пам'яті	МВ	3	4	3	3	4	3	4	24	6,5	42,25
$X3$	Час реагування на загрозу	с	2	1	1	2	1	2	1	10	7,5	56,25
$X4$	Потенційний об'єм програмного коду	Кількість рядків коду	4	3	4	4	3	4	3	25	7,5	56,25
	Разом		10	10	10	10	10	10	10	70	0	197

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри.

Сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70,$$

де N – число експертів,

n – кількість параметрів;

Середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17,5.$$

Відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T$$

Сума відхилень по всіх параметрам повинна дорівнювати 0;

Загальна сума квадратів відхилення:

$$S = \sum_{i=1}^N \Delta_i^2 = 197.$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 197}{7^2(4^3-4)} = 0,804 > W_k = 0,67.$$

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

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати заносимо у таблицю 4.4.

Таблиця 4.4 – Попарне порівняння параметрів.

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

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

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases}$$

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

Для кожного параметра зробимо розрахунок вагомості $K_{\text{в}i}$ за наступними формулами:

$$K_{\text{в}i} = \frac{b_i}{\sum_{i=1}^n b_i},$$

$$b_i = \sum_{j=1}^N a_{ij}.$$

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

$$K_{\text{Ві}} = \frac{b_i'}{\sum_{i=1}^n b_i'}$$

$$b_i' = \sum_{j=1}^N a_{ij}.$$

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

Таблиця 4.5 – Розрахунок вагомості параметрів

Параметри	Параметри				Перша ітер.		Друга ітер.		Третя ітер.	
	X1	X2	X3	X4	b_i	$K_{\text{Ві}}$	b_i^1	$K_{\text{Ві}}^1$	b_i^2	$K_{\text{Ві}}^2$
X1	1	0,5	1,5	0,5	3,5	0,2188	12,25	0,2076	44,875	0,2078
X2	1,5	1	1,5	0,5	4,5	0,2813	16,25	0,2754	59,125	0,2737
X3	0,5	0,5	1	0,5	2,5	0,1563	9,25	0,1568	34,125	0,1580
X4	1,5	1,5	1,5	1	5,5	0,3438	21,25	0,3602	77,875	0,3605
Всього:		16	1	59	1	216	1			

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів X_2 (Об'єм пам'яті), X_3 (час реагування на загрозу) та X_4 (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра X_1 (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

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

де n – кількість параметрів;

K_{vi} – коефіцієнт вагомості i -го параметра;

B_i – оцінка i -го параметра в балах.

Таблиця 4.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	A	X1	16000	7	0,21	1,47
F2	A	X2	530	3	0,27	0,81
	Б	X3	3	8	0,16	1,28
F3	A	X4	1100	9	0,36	3,24

За даними з таблиці 4.6 за формулою:

$$K_K = K_{Ty}[F_{1k}] + K_{Ty}[F_{2k}] + \dots + K_{Ty}[F_{zk}].$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 1,47 + 0,81 + 3,24 = 5,52;$$

$$K_{K2} = 1,47 + 1,28 + 3,24 = 5,99.$$

Як видно з розрахунків, кращим є 2 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання.

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

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

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

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

Загальна трудомісткість обчислюється як:

$$T_0 = T_P \cdot K_P \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М},$$

де T_P – трудомісткість розробки ПП;

K_P – поправочний коефіцієнт;

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

K_M – коефіцієнт рівня мови програмування;

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

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

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру ступеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_P = 60$ людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_P = 1.6$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ} = 0.7$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 60 \cdot 1.6 \cdot 0.7 = 67,2 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_p = 40$ людино-днів, $K_{\Pi} = 0.9$, $K_{СК} = 1$, $K_{СТ} = 0.75$:

$$T_2 = 40 \cdot 0.9 \cdot 0.75 = 27 \text{ людино-днів.}$$

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

$$T_I = (67,2 + 27 + 4.8 + 27) \cdot 8 = 1008 \text{ людино-годин.}$$

$$T_{II} = (67,2 + 27 + 6.91 + 27) \cdot 8 = 1024,88 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці бере участь один програміст-аналітик з окладом 26800 грн. Визначимо середню зарплату за годину за формулою:

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

де M – місячний оклад працівників;

T_m – кількість робочих днів тижень;

t – кількість робочих годин в день.

$$C_{\text{ч}} = \frac{26800}{21 \cdot 8} = 159,52 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою:

$$C_{зп} = C_{ч} \cdot T_i \cdot K_d,$$

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

T_i – трудомісткість відповідного завдання;

K_d – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$\text{I. } C_{зп} = 159,52 \cdot 1008 \cdot 1.2 = 192955,39 \text{ грн.}$$

$$\text{II. } C_{зп} = 159,52 \cdot 1024,88 \cdot 1.2 = 196186,63 \text{ грн.}$$

Відрахування на єдиний соціальний внесок становить 22%:

$$\text{I. } C_{від} = C_{зп} \cdot 0.22 = 192955,39 \cdot 0.22 = 42450,18 \text{ грн.}$$

$$\text{II. } C_{від} = C_{зп} \cdot 0.22 = 196186,63 \cdot 0.22 = 43161,05 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. (C_M)

Так як одна ЕОМ обслуговує одного програміста з окладом 26800 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_{Г} = 12 \cdot M \cdot K_3 = 12 \cdot 26800 \cdot 0,2 = 64320 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{зп} = C_{Г} \cdot (1 + K_3) = 64320 \cdot (1 + 0.2) = 77184 \text{ грн.}$$

Відрахування на соціальний внесок:

$$C_{\text{від}} = C_{\text{зп}} \cdot 0.22 = 77184 \cdot 0.22 = 16980,48 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 35500 грн.

$$C_A = K_{\text{тм}} \cdot K_A \cdot C_{\text{пп}} = 1.2 \cdot 0.25 \cdot 35500 = 10650 \text{ грн.,}$$

де $K_{\text{тм}}$ – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

K_A – річна норма амортизації;

$C_{\text{пп}}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{\text{тм}} \cdot C_{\text{пп}} \cdot K_P = 1.2 \cdot 35500 \cdot 0.07 = 2982 \text{ грн.,}$$

де K_P – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$\begin{aligned} T_{\text{еф}} &= (D_K - D_B - D_C - D_P) \cdot t \cdot K_B = (365 - 104 - 12 - 16) \cdot 8 \cdot 0.88 = \\ &= 1640,32 \text{ години,} \end{aligned}$$

де D_K – календарна кількість днів у році;

D_B, D_C – відповідно кількість вихідних та святкових днів;

D_P – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

K_B – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{\text{ЕЛ}} = T_{\text{ЕФ}} \cdot N_{\text{С}} \cdot K_3 \cdot C_{\text{ЕН}} = 1640,32 \cdot 0,35 \cdot 0,88 \cdot 5,21755 = 2636 \text{ грн.},$$

де $N_{\text{С}}$ – середньо-споживча потужність приладу;

K_3 – коефіцієнтом зайнятості приладу;

$C_{\text{ЕН}}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_{\text{Н}} = C_{\text{ПР}} \cdot 0,67 = 35500 \cdot 0,67 = 23785 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{\text{ЕКС}} = 77184 + 16980,48 + 10650 + 2982 + 2636 + 23785 = 134217,48 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{\text{М-Г}} = C_{\text{ЕКС}} / T_{\text{ЕФ}} = 134217,48 / 1640,32 = 81,82 \text{ грн/год.}$$

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

$$\text{I. } C_{\text{М}} = 81,82 \cdot 1008 = 82474,56 \text{ грн.}$$

$$\text{II. } C_{\text{М}} = 81,82 \cdot 1024,88 = 83855,68 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$\text{I. } C_H = 192955,39 \cdot 0,67 = 129280,11 \text{ грн.}$$

$$\text{II. } C_H = 196186,63 \cdot 0,67 = 131445,04 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$\text{I. } C_{\text{ПП}} = 192955,39 + 42450,18 + 82474,56 + 129280,1 = 447160,23 \text{ грн.}$$

$$\text{II. } C_{\text{ПП}} = 196186,63 + 43161,05 + 83855,68 + 131445,04 = 454648,4 \text{ грн.}$$

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР1}} = 5,52 / 447160,23 = 12,3446 \cdot 10^{-6},$$

$$K_{\text{ТЕР2}} = 5,99 / 454648,4 = 13,175 \cdot 10^{-6}.$$

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $K_{\text{ТЕР2}} = 13,175 \cdot 10^{-6}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розробляється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є другий варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $K_{\text{ТЕР}} = 13,175 \cdot 10^{-6}$.

Цей варіант реалізації програмного продукту має такі параметри:

- вибір мови програмування – Python;

- інструмент для реалізації систем-експертів – PyKnow;
- середовище розробки – Visual Studio Code.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку реалізацію програми та доступний функціонал для роботи.

Висновки до розділу 4

В рамках даного розділу було проведено ґрунтовне дослідження програмного продукту з метою його функціонально-вартісного аналізу.

Цей аналіз дозволив: визначити та оцінити ключові функції програмного продукту, детально охарактеризувати його функціональні можливості, отримати дані про параметри, що визначають програмний комплекс та на основі проведеного дослідження обрати оптимальний варіант його реалізації.

Завдяки цьому дослідженню чітко окреслено функціональні можливості програмного продукту, їх відповідність вартості розробки та оптимальний шлях реалізації. Отримані результати будуть використані в наступних розділах дипломної роботи для розробки детального проекту програмного продукту та його втілення.

ВИСНОВКИ

Інформаційне середовище невинно розвивається, разом із ним розвиваються комп'ютерні загрози. Складність та різноманіття деталей не дозволяє створити людям систему захисту даних на основі наперед визначених правил. Для вирішення даної проблеми є необхідним використання нових підходів у обробці даних, яким є штучний інтелект.

Отже метою даної дипломної роботи було дослідження методів штучного інтелекту, які дозволяють вирішувати задачі, захисту даних що виникають у сучасному інформаційному середовищі. Прикладом таких методів є використання глибинних нейронних мереж та проектування систем-експертів для виявлення втручань у мережу та подальшого реагування на загрозу.

При дослідженні практичного застосування технологій було знайдено ряд інструментів на мові Python, що дозволяють швидко створити систему-експерта, а саме бібліотеку для розробки експертних-систем та модель нейронної мережі для виявлення мережових вторгнень у систему.

ПЕРЕЛІК ПОСИЛАНЬ

1. Jang-Jaccard J. A survey of emerging threats in cybersecurity [Електронний ресурс] / J. Jang-Jaccard, S. Nepal // Journal of Computer and System Sciences. – Серпень 2014. – Випуск 80. Режим доступу: <https://www.sciencedirect.com/science/article/pii/S0022000014000178>
2. Interpol Annual Report 2022 [Електронний ресурс]. Режим доступу: https://www.interpol.int/content/download/19843/file/INTERPOL%20%20Annual%20Report%202022_EN.pdf
3. Ni Komang T. A. Cybercrime and Threats to the Electoral System / T. A. Ni Komang, N. P. D. Ni Made Ayu // Journal of Digital Law and Policy. – Вересень 2023. – Випуск 3. – e-ISSN 2808-3458, p-ISSN 2808-3652.
4. Crime Online / наук. ред. Y. Jewkes : Willan, 2006 – ISBN 978-1843921974.
5. Gupta M. From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy [Електронний ресурс] / M. Gupta, Ch. Akiri; Institute of Electrical and Electronics Engineers // IEEE Access. – 2023. – Випуск 11. Режим доступу: <https://ieeexplore.ieee.org/document/10198233>
6. Learning outcomes of classroom research / ред. J. Karthikeyan, Ting Su Hie, Ng Yu Jin – New Delhi, India : L Ordine Nuovo Publication, 2021 – ISBN 978-93-92995-15-6.
7. Craigen D. Defining Cybersecurity [Електронний ресурс] / D. Craigen, N. Diakun-Thibault // Technology Innovation Management Review – Жовтень 2014. Режим доступу: https://www.timreview.ca/sites/default/files/article_PDF/Craigen_et_al_TIMReview_October2014.pdf
8. International Journal of Applied and Natural Sciences [Електронний ресурс] // : Bluemark Publisher – ISSN 3005-3471. Режим доступу: <https://bluemarkpublishers.com/index.php/IJANS/article/view/2/2>

9. Schmittner C. Automotive cybersecurity standards - relation and overview. In Computer Safety, Reliability, and Security / C. Schmittner, G. Macher – Turku, Finland : Springer International Publishing – 2019.
10. Liapopoulos A. Exploring the Complexity of Cyberspace Governance [Електронний доступ] / Liapopoulos A : Journal of Information Warfare. – Осінь 2016. – Випуск 15. Режим доступу: <https://www.jstor.org/stable/26487548>
11. Al-Masri A. Smart Technologies and Innovation for a Sustainable Future / A. Al-Masri, ред. K. Curran – Dubai, UAE : Springer, 2019 – ISBN 978-3-030-01659-3.
12. Sotnik S. Key Directions for Development of Modern Expert Systems / S. Sotnik, Zh. Deineko : International Journal of Engineering and Information Systems – Весна 2022. – Випуск 6. — ISSN 2643-640X
13. Sarker I. H. AI-Driven Cybersecurity: An Overview, Security Intelligence Modeling and Research Directions / I. H. Sarker – 2021.
14. Manesh B. Machine Learning Algorithms – A Review [Електронний ресурс] / B. Manesh // International Journal of Science and Research – Жовтень 2020. – ISSN 2319-7064. Режим доступу: <https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762>
15. Lecun Y. Deep Learning [Електронний ресурс] / Y. Lecun, Y. Bengio // Nature – Травень 2015. – Випуск 521. Режим доступу: <https://hal.science/hal-04206682>
16. Faranak A. Can Machine/Deep Learning Classifiers Detect Zero-Day Malware with High Accuracy? [Електронний ресурс] / A. Faranak, S. Siami-Namini // IEEE International Conference on Big Data – 2019. Режим доступу: <https://www.researchgate.net/publication/339482659>
17. Vijayan J. Using Natural Language Processing to Identify Malicious Domains [Електронний ресурс] / J. Vijayan // Security Intelligence – Березень, 2015.

Режим доступа: <https://securityintelligence.com/news/using-natural-language-processing-identify-malicious-domains/>

18. Branchman R. J. Knowledge Representation and Reasoning / R. J. Branchman, H. J. Levesque – San Francisco, USA : Morgan Kaufmann publications, 2004 – ISBN 1-55860-932-6.
19. Minsky M. A Framework for Representing Knowledge / M. Minsky // MIT-AI Laboratory Memo 306 – Червень, 1974
20. Pereira T. An ontology based approach to information security. / T. Pereira, H. Santos // Research Conference on Metadata and Semantic Research : Springer, 2009
21. Tripathi K. P. A Review on Knowledge-based Expert System: Concept and Architecture [Электронный ресурс] / K. P. Tripathi; Bharati Vidyapeeth Deemed University, Institute of Management – Kolhapur, India : AIT, 2011. Режим доступа:
<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=19f89f1b19b558389e9c70b75403063376cb8e3d>
22. Nair V. Rectified linear units improve restricted Boltzmann machines [Электронный ресурс] / V. Nair, G. E. Hilton // Proceedings of the 27th International Conference on International Conference on Machine Learning – Червень 2010. Режим доступа:
<https://dl.acm.org/doi/10.5555/3104322.3104425>
23. Hahnloser R. H. R. Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit [Электронный ресурс] / R. H. R. Hahnloser, R. Sarpeshkar // Nature – Червень 2000. – Випуск 405. Режим доступа:
<https://www.nature.com/articles/35016072>
24. Li F.-F. Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition [Электронный ресурс] / F.-F. Li, A. Karpathy // Stanford – Весна 2024. Режим доступа: <http://cs231n.stanford.edu/>.

25. Rumelhart D. E. Learning representations by back-propagating errors [Електронний ресурс] / D. E. Runelhart, G. E. Hinton // Nature – Жовтень 1986. Режим доступу: <https://www.nature.com/articles/323533a0>
26. Nielsen M. Neural Networks and Deep Learning [Електронний ресурс] / М. Nielsen – Грудень 2019. Режим доступу: <http://neuralnetworksanddeeplearning.com/>
27. Keiron O'S. An Introduction to Convolutional Neural Networks [Електронний ресурс] / O'S. Keiron, R. Nash. Режим доступу: <https://arxiv.org/pdf/1511.08458>
28. Liu Y. A Survey on Evolutionary Neural Architecture Search [Електронний ресурс] / Y. Liu, Y. Sun. Режим доступу: <https://arxiv.org/pdf/2008.10937>
29. Zoph B. Neural Architecture Search with Reinforcement Learning [Електронний ресурс] / B. Zoph, Q. V. Le. Режим доступу: <https://arxiv.org/pdf/1611.01578v2>
30. Sze V. Efficient Processing of Deep Neural Networks: A Tutorial and Survey [Електронний ресурс] / V. Zhe, Y.-H. Chen. Режим доступу: <https://arxiv.org/pdf/1703.09039>
31. Al-Ajlan A. The Comparison between Forward and Backward Chaining [Електронний ресурс] / A. Al-Ajlan // Internation Journal of Machine Learning and Computing – Квітень 2015. – Випуск 5. Режим доступу: <https://www.ijml.org/vol5/492-A14.pdf>
32. Khaing K. T. Analyzing Denial-of-service Attacks in KDD CUP 99 Data Set for Intrusion Detection System / K. T. Khaing, T. T. Naing // University of Computer Science, Yangon
33. Датасет KDD CUP 99 [Електронний ресурс]. Режим доступу: <https://www.tensorflow.org/datasets/catalog/kddcup99>
34. Ieracitano C. Statistical Analysis Driven Optimized Deep Learning System for Intrusion Detection [Електронний ресурс] / C. Ieracitano, A. Adeel – Серпень 2018. Режим досутупу: <https://www.researchgate.net/publication/327110465>

35. Бібліотека PyKnow [Електронний ресурс]. Режим доступу:
<https://github.com/buguroo/pyknow>
36. Yaar A. Pi: A Path Identification Mechanism to Defend against DDoS Attacks /
A. Yaar, A. Perrig // Carnegie Mellon University
37. Mirkovic J. A Taxonomy of DDoS Attack and DDoS Defence Mechanisms
[Електронний ресурс] / J. Mirkovic, P. Reiher // ACM SIGCOMM Computer
Communication Review – Травень 2004. Режим доступу:
<https://www.researchgate.net/publication/2879658>

ДОДАТОК А. РЕАЛІЗАЦІЯ СИСТЕМИ-ЕКСПЕРТА

Лістинг А1. Архітектура нейронної мережі

```
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    """(Linear => ReLU) => (Linear => [dropout] =>
ReLU) * 2 => (Linear)"""

    def __init__(self, input_dim, hidden_1, hidden_2,
hidden_3, out_dim):
        super(Net, self).__init__()
        self.c1 = nn.Sequential(nn.Conv2d(1, 64,
kernel_size=3,padding=1),
                                #nn.BatchNorm2d(32),
                                nn.ReLU(inplace=True))
        self.c2 = nn.Sequential(nn.Conv2d(64, 64,
kernel_size=3,padding=1),
                                #nn.BatchNorm2d(64),
                                nn.ReLU(inplace=True))
        self.p1 =
nn.Sequential(nn.MaxPool2d(kernel_size=2, stride=2))
        self.fc1 =
nn.Sequential(nn.Linear(64*3*3,hidden_1),
nn.Dropout(0.5),nn.ReLU(inplace=True))
```

```

        self.fc3
nn.Sequential(nn.Linear(hidden_1,hidden_3),
nn.Dropout(0.5), nn.ReLU(inplace=True))
        self.fc4
nn.Sequential(nn.Linear(hidden_3,out_dim))

    def forward(self, x):
        x = self.c1(x)
        x = self.c2(x)
        x = self.p1(x)
        x = x.view(x.size(0), -1)
        x = self.fc1(x)
        x = self.fc3(x)
        x = self.fc4(x)
        return x

```

Лістинг А2. Створення механізму судження

```

from pyknow import *

class Intrusion(Fact):
    """Info about the intrusion type."""
    pass

class Layer(Fact):
    """Info about the intrusion layer"""
    pass

class Restriction(Fact):
    """Info about the restriction"""

```

```

pass

class HarmlessCalls(Fact):
    """Info about the number of harmless calls"""
    pass

class IntrusionDetection(KnowledgeEngine):
    @Rule(Intrusion(type='back'),
          NOT (Restriction(type='back'))
    )
    def back_intrusion(self):
        print("Identify      and      isolate      corrupted
systems")

        self.declare(Layer(type='IP'))
        self.declare(Restriction(type='back'))
        self.declare(HarmlessCalls(number=0))

    @Rule(Intrusion(type='land'),
          NOT (Restriction(type='land'))
    )
    def land_intrusion(self):
        self.declare(Layer(type='IP'))
        self.declare(Restriction(type='land'))
        self.declare(HarmlessCalls(number=0))

    @Rule(Intrusion(type='neptune'),
          NOT (Restriction(type='neptune'))
    )
    def neptune_intrusion(self):
        self.declare(Layer(type='TCP'))
        self.declare(Restriction(type='neptune'))
        self.declare(HarmlessCalls(number=0))

```

```

@Rule(Intrusion(type='pod'),
      NOT (Restriction(type='pod')))
def pod_intrusion(self):
    self.declare(Layer(type='ICMP'))
    self.declare(Restriction(type='pod'))
    self.declare(HarmlessCalls(number=0))

@Rule(Intrusion(type='smurf'),
      NOT (Restriction(type='smurf')))
def smurf_intrusion(self):
    print("Block traffic on broadcast addresses")
    self.declare(Layer(type='ICMP'))
    self.declare(Restriction(type='smurf'))
    self.declare(HarmlessCalls(number=0))

@Rule(Intrusion(type='teardrop'),
      NOT (Restriction(type='teardrop')))
def teardrop_intrusion(self):
    print("Block fragmented packets")
    self.declare(Layer(type='TCP'))
    self.declare(Restriction(type='teardrop'))
    self.declare(HarmlessCalls(number=0))

@Rule(Layer(type='TCP'))
def tcp_layer_intrusion(self):
    print("Enable SYN cookies")
    print("Restrict    number    of    incoming    SYN
packages")

```

```

@Rule(Layer(type='IP'))
def ip_layer_intrusion(self):
    print("Traffic filtering")

@Rule(Layer(type='ICMP'))
def icmp_layer_intrusion(self):
    print("Limit ICMP traffic")

@Rule(Intrusion(),
      HarmlessCalls(number=MATCH.number),
      TEST(lambda number: number > 5))
def recovery():
    print("Update system")
    print("Save logs")
    self.declare(Intrusion(type='none'))

@Rule(Intrusion(type='none'),
      AS.calls
      HarmlessCalls(number=MATCH.number))
def no_intrusion(self, calls):
    print("No intrusion detected")

self.declare(HarmlessCalls(number=calls['number'] + 1))

@Rule(Intrusion(type='neptune'),
      HarmlessCalls(number=MATCH.number),
      TEST(lambda number: number > 5))
def neptune_recovery(self):
    print("Rebalance traffic")

```

ДОДАТОК Б ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

Методи штучного інтелекту в задачах інформаційної безпеки

Виконав студент групи KI-02

Москаленко Максим Геннадійович

Науковий керівник: к.т.н., доцент кафедри ШІ

Коваленко Анатолій Єпіфанович

Рисунок Б1 – Слайд №1

Мета роботи

Мета дослідження: проаналізувати існуючі методи штучного інтелекту для захисту даних. Запропонувати практичне рішення з застосування штучного інтелекту у прикладній задачі захисту даних.

Об'єкт дослідження: методи штучного інтелекту.

Предмет дослідження: використання штучного інтелекту для захисту даних та використання алгоритмів. Конкретні алгоритми та реалізації систем інформаційної безпеки з використанням штучного інтелекту.

Рисунок Б2 – Слайд №2

Актуальність

- Зростаюча складність інформаційного простору
- Зростаюча складністю інформаційних атак
- Останні досягнення у галузі штучного інтелекту
- Збільшення обчислювальних здатностей комп'ютерних систем

Рисунок Б3 – Слайд №3

Виклики інформаційної безпеки

- Розвиток природи кіберзагроз
- Підвищення складності інформаційного середовища
- Обмеження традиційної моделі безпеки даних

Рисунок Б4 – Слайд №4

Загрози інформаційної безпеки

- Зловмисне програмне забезпечення
- Соціальна інженерія/Фішинг
- DoS атаки
- Розвинені сталі загрози (АРТ)

Рисунок Б5 – Слайд №5

Методи штучного інтелекту

- Машинне навчання та глибинне навчання
- Природна обробка мови
- Технологія представлення знань
- Моделювання систем-експертів

Рисунок Б6 – Слайд №6

Система-эксперт

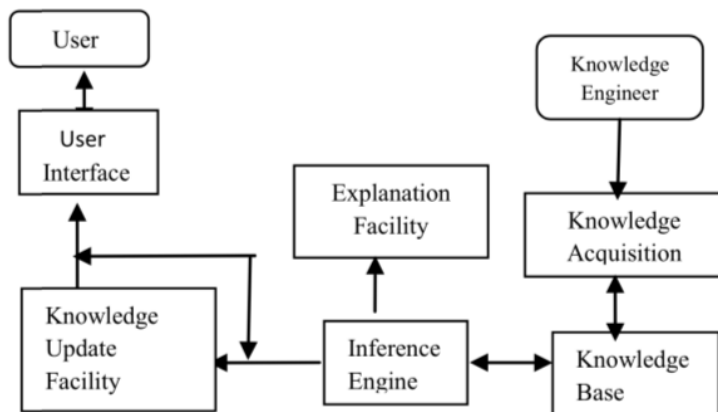


Рисунок Б7 – Слайд №7

Система-эксперт із захисту даних

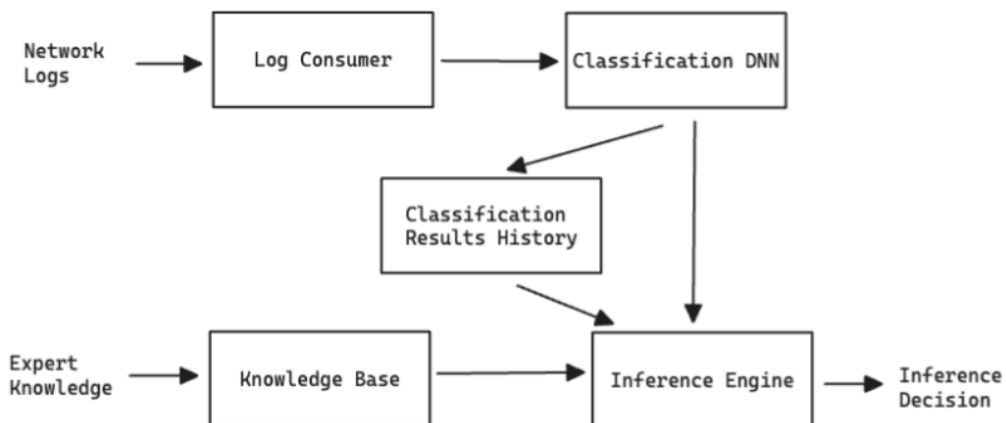


Рисунок Б8 – Слайд №8

Задання правил для системи-експерта

```
class IntrusionDetection(KnowledgeEngine):
    @Rule(Intrusion(type='smurf'),
          NOT (Restriction(type='smurf'))))
    def smurf_intrusion(self):
        print("Block traffic on broadcast addresses")
        self.declare(Layer(type='ICMP'))
        self.declare(Restriction(type='smurf'))
        self.declare(HarmlessCalls(number=0))

    @Rule(Layer(type='ICMP'))
    def icmp_layer_intrusion(self):
        print("Limit ICMP traffic")
```

Рисунок Б9 – Слайд №9

Архітектура нейронної мережі

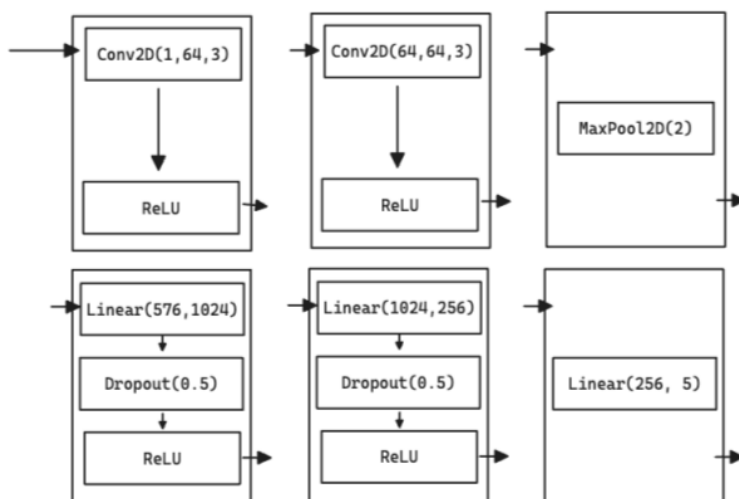


Рисунок Б10 – Слайд №10

Висновки

- Досліджено сфери використання методів штучного інтелекту для задач інформаційної безпеки
- Розглянуто основні виклики та задачі комп'ютерної безпеки, які можна вирішити методами штучного інтелекту
- Запропоновано рішення для задачі виявлення та реагувань на мережеві втручання з використанням нейронних мереж та системи-експерта

Рисунок Б11 – Слайд №11

Подальші дослідження

У подальших дослідженнях планується розширити можливості системи-експерта до протидії АРТ, що включає у себе:

- Аналіз системних втручань
- Аналіз програмних ресурсів на наявність вразливого та/або інфікованого коду
- Визначення часових закономірностей та передбачення моменту наступної атаки
- Аналіз зі сторонніх джерел

Рисунок Б12 – Слайд №12

Дякую за увагу!

Рисунок Б13 – Слайд №13