Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування закладу вищої освіти)

Навчально-науковий інститут інформаційних технологій і робототехніки
(повне найменування інституту, назва факультету (відділення))

Кафедра автоматики, електроніки та телекомунікацій
(повна назва кафедри (предметної, циклової комісії))

Пояснювальна записка

до кваліфікаційної роботи

магістр
(ступінь вищої освіти)

на тему **Методи та системи виявлення мережевих атак на основі нейромережевих технологій**

Виконав: студент 6 курсу, групи 601ТТ
спеціальності 172 «Телекомунікації та
(шифр і назва напряму підготовки, спеціальності)
радіотехніка

Приходько М.С.
(прізвище та ініціали)
керівник Штомпель М.А.
(прізвище та ініціали)
Рецензент
Жученко О.А.
(прізвище та ініціали)

Полтава - 2023 рік

Національний університет «Полтавська політехніка імені Юрія Кондратюка»
Навчально-науковий інститут інформаційних технологій і робототехніки
Кафедра Автоматики, електроніки та телекомунікацій
Ступінь вищої освіти Магістр
Спеціальність 172 «Телекомунікації та радіотехніка»

ЗАТВЕРДЖУЮ

Завідувач кафедри автоматики, електроніки та телекомунікацій

_____ О.В. Шефер
“ 04 ” 09 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Приходьку Миколі Сергійовичу

1. Тема проекту (роботи) **«Методи та системи виявлення мережевих атак на основі нейромережевих технологій»**
керівник проекту (роботи) Штомпель Микола Анатолійович,
затверджена наказом вищого навчального закладу від “04” вересня 2023 року № 986-фа

2. Строк подання студентом проекту (роботи) 13.12.2023 р.

3. Вихідні дані до проекту (роботи) Нейромережеві методи. База даних сигнатур NSL - KDD.

4. Зміст розрахунково – пояснювальної записки (перелік питань, які потрібно розробити) Аналіз стану та напрямки розвитку системи виявлення мережевих атак. Аналіз методів побудови системи виявлення мережевиж на основі нейромережевих технологій. Розробка моделі нейронної мережі для системи виявлення мережевих атак. Висновки по роботі.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових плакатів):
1) Пояснювальна записка (77 ст.)
2) Презентація в PowerPoint (15 – 20 слайдів)

6. Дата видачі завдання 02.10.2023 р.

КАЛЕНДАРНИЙ ПЛАН

Пор. №	Назва етапів магістерської роботи	Термін виконання етапів роботи			Примітка (плакати)
1	Опрацювання необхідної літератури.	11.10.23		15%	
2	Написання та подання на перевірку 1 – го розділу роботи.	18.10.23	I	30%	
3	Опрацювання необхідної літератури	25.10.23		40%	
5	Написання та подання на перевірку 2 – го розділу роботи.	21.11.23	II	50%	
6	Написання та подання на перевірку 3– го розділу роботи.	28.11.23		70%	
7	Реалізація практичної частини та подання на перевірку.	06.12.23		90%	
8	Оформлення магістерської роботи	13.12.23	III	100%	

Магістрант _____ Приходько М.С.
(підпис) (прізвище та ініціали)

Керівник роботи_____ Штомпель М.А.
(підпис) (прізвище та ініціали)

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 5

ВСТУП 6

РОЗДІЛ 1 АНАЛІЗ СТАНУ ТА НАПРЯМКИ РОЗВИТКУ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК 7

1.1 Аналіз мережевих атак 7

1.2 Системи виявлення мережевих атак, застосування та напрямки розвитку 11

Висновки до першого розділу 15

РОЗДІЛ 2 АНАЛІЗ МЕТОДІВ ДЛЯ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК НА ОСНОВІ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ 18

2.1 Аналіз нейромережевих технологій 18

2.2 Методи виявлення атак за допомогою DL 19

2.3 Опис даних мережевих атак 41

Висновок до другого розділу 45

РОЗДІЛ 3 РОЗРОБКА МОДЕЛІ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК 46

3.1 Реалізація програмного коду для зберігання, форматування набору даних 46

3.2 Модель нейронної мережі на основі алгоритму глибокго Q-навчання 47

3.3 Результати ефективності моделі виявлення мережевих атак 56

Висновок до третього розділу 58

ВИСНОВКИ 59

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 60

ДОДАТОК А. ПРИКЛАД ДАНИХ NSL KDD 66

ДОДАТОК В. ЛІСТИНГ КОДА 68

ДОДАТОК С. ПЕРЕКЛД 1 РОЗДІЛУ 71

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

DOS *(Denial of Service)* - Атака на відмову в обслуговуванні
U2R*(User to root)* –Незаконе отримання доступу адміністратора користувачем
R2L*(Remote to local)* – Віддалені атаки через мережу
IDS*(Intrusion Detection System)* – Система виявлення атак
IPS*(ntrusion Prevention System)* – Система запобігання вторгнень
AI*(Artificial Intelligence)* – Штучний інтелект
DL*(Deep Learning)* – Глибоке навчання
AE*(Autoencoders)* – Автокодер
STL*(Seasonal and Trend decomposition using Loess)* – Розкладання часових рядів
DBM*(Deep Boltzmann machine)* – Глибока машина Больцмана
DBN*(Deep Belief Networks)* – Мережі глибокої віри
RBM*(Restricted Boltzmann Machines)* – Обмежені машини Больцмана
GAN*(Generative Adversarial Network)* – Генеративна змагальна мережа
URL*(Uniform Resource Locator)* – Уніфікований покажчик інформаційного ресурсу
DNN*(Deep Neural Networks)* – Глибокі нейронні мережі
CNN*(Convolutional Neural Network)* – Свертальна нейронна мережа
LDOS*(Low Rate Denial of Service)* – Відмова в обслуговуванні
DQN*(Deep Q Learning)* – Глибоке *Q*–навчання
LSTM*(Long short–term memory)* – Довга короткочасна пам’ять
GRU*(Gated Recurrent Units)* – Закриті рекурентні одиниці
SVM*(Support Vector Machines)* – Підтримка векторних машин
RL*(Reinforcement learning)* – Підкріплення навчання
OSI*(The Open Systems Interconnection model)*- абстрактна мережева модель для комунікацій і розробки мережевих протоколів. NDAE(*non-symmetric deep auto-encoder)* – Несиметричний глибокий автокодер
NSL-KDD(*data set*) – Набір даних
DQ-RL(*(Deep Q with Reinforcement learning*) - Глибоке Q навчання з підкріпленням

ВСТУП

Розвиток обчислювальних засобів та інформаційних технологій призводить до автоматизації різноманітних процесів практично у всіх сферах життя суспільства: зростає обчислювальна потужність обчислювальних засобів, зростає технологія, все більше вдосконалюється технологія мережевої взаємодії, формати та вимоги до побудови інформаційних систем. все більше розвивається. Разом з розвитком інформаційних технологій постійними темпами з'являються нові загрози інформаційній безпеці. Тому питання захисту інформації залишається ключовим напрямком наукових досліджень.

Одним із основних засобів захисту комп'ютерних систем є системи виявлення вторгнень (IDS). Системи виявлення вторгнень використовуються для виявлення різних типів зловмисної діяльності від кібератак на різноманітні служби. Штучний інтелект дозволяє швидше виявляти кібератаки, адаптуватися до них і приймати рішення щодо них.

Метою дослідження є вдосконалення системи виявлення мережевих атак за допомогою нейромережевих технологій.

Об’єктом дослідження є процес виявлення кібератак.

Тема дослідження – моделювання нейронної мережі для виявлення кібератак.

РОЗДІЛ 1
АНАЛІЗ СТАНУ ТА НАПРЯМКИ РОЗВИТКУ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК

1.1 Аналіз мережевих атак

Мережева атака - спроба обійти політику безпеки системи, що надає зловмисникам доступ для отримання або модифікації інформації, навіть руйнуючи систему. У зв'язку з технологіями, що розробляються в системах локальної мережі та бездротової мережі, існують серйозні загрози безпеці мережі, особливо безпеці систем. Оскільки ми зараз перебуваємо в епохі машинного навчання та великих даних[1], кібербезпека в системах бездротової та локальної мережі є важливою для користувачів.

Характер впливу[1]:

- активний;
- пасивний.

Активна атака розуміється вплив, який чинить на роботу системи, а саме порушення працездатності системи, зміна конфігурації, тощо. Практично всі типи мережевих атак є атаками з активними діями. Особливістю таких атак є можливість їх виявлення, в різних випадках з більшим чи меншим ступенем, так як в результаті таких атак відбуваються деякі зміни в системі.

Пасивна атака на обчислювальну систему, яка не має безпосереднього впливу на функціонування системи, але може привести до порушення політики безпеки. На відміну від активного при пасивній атаці не залишається ніяких слідів. Саме відсутність безпосереднього впливу на роботу мережі призводить до того, що пасивну атаку практично неможливо виявити. Прикладом атак з пасивним впливом є прослуховування каналу зв'язку в мережі.

Мета атаки[2]:

- порушення працездатності системи (доступу до системи);
- спотворення цілісності інформації;
- порушення конфіденційності інформації.

Мета будь-якої атаки - дістати несанкціонований доступ до інформації. Існують дві принципові можливості доступу до інформації: перехоплення і спотворення.

Перехоплення інформації означає отримання до неї доступу, але неможливість її модифікації.

Перехоплення інформації веде до порушення її конфіденційності. Прикладом перехоплення інформації може служити прослуховування каналу в мережі. В цьому випадку є несанкціонований доступ до інформації без можливості її спотворення. Порушення конфіденційності інформації є пасивним впливом.

Спотворення інформації означає або повний контроль над інформаційним потоком між об'єктами системи, або можливість передачі повідомлень від імені іншого об'єкта. Таким чином, спотворення інформації веде до порушення її цілісності. Має руйнівний вплив який являє собою приклад активного впливу.

Порушення працездатності системи в цьому разі не передбачається отримання атакуючим несанкціонованого доступу до інформації. Основна мета - домогтися, щоб система яка атакується вийшла з ладу, а для всіх система доступ до ресурсів атакованого об'єкта був би неможливий. Прикладом мережевої атаки, метою якої є порушення працездатності системи, може служити атака відмови в обслуговуванні - DOS атака.

Умова початку здійснення впливу: [2]

Віддалений вплив, також, як і будь-який інший вплив, може почати здійснюватися тільки за певних умов. В обчислювальних мережах існують три види умов початку здійснення мережевої атаки:

- атака за запитом від атакуючого об'єкта;
- атака по настанню очікуваної події на атаці системи;
- безумовна атака.

Вплив з боку атакуємого розпочнеться за умови, що потенційна мета атаки передати запит певного типу. Таку атаку можна назвати атакою за запитом від атакуючого об'єкта. Даний тип початку здійснення найбільш характерний для

режиму виявлення систем. Прикладом подібних запитів в мережі Інтернет може служити *DNS*- і *ARP*-запити, а в *Novell NetWare — SAP*-запит.

Атака за настання очікуваної події на атакуємому об'єкті. Атакуючий безперервно спостерігає за станом операційної системи віддаленої мети атаки і починає вплив при виникненні певної події в цій системі. Атакуючий об'єкт сам є ініціатором початку атаки. Прикладом такої події може бути переривання сеансу роботи користувача із сервером без видачі команди *LOGOUT* в *Novell NetWare*.

Безумовна атака здійснюється негайно і безвідносно до стану операційної системи атакуючим об'єкта. Отже, атакуючий є ініціатором початку атаки в даному випадку.

При порушенні нормальної працездатності системи переслідуються інші цілі і отримання атакуючим незаконного доступу до даних не передбачається. Його метою є виведення з ладу операційної системи на атакуючому об'єкті та неможливість доступу для інших об'єктів системи до ресурсів цього об'єкта. Прикладом атаки такого виду може служити DOS-атака.

Наявність зворотнього зв’язку. [2]

- зі зворотним зв'язком;
- без зворотного зв'язку.

Мережева атака, здійснюється при наявності зворотного зв'язку яка атакується об'єктом, характеризується тим, що на деякі запити, передані на атакуємий об'єкт, атакуючому потрібно отримати відповідь, а, отже, між атакуючим і метою атаки існує зворотний зв'язок, яка дозволяє атакуючому адекватно реагувати на всі зміни, що відбуваються на об'єкті. Подібні мережеві атаки найбільш характерні для розподілених обчислювальних мереж.

Мережевим атакам без зворотного зв'язку не потрібно реагувати на будь-які зміни, що відбуваються на атакуємому об'єкті. Атаки даного виду зазвичай здійснюються передачею на об'єкт, що атакується одиночними запитами, відповіді на які атакуючому не потрібні. Прикладом односпрямованих атак є типова DOS-атака.

Розташування атакуючого:

- внутрішньо сегментне;
- між сегментне.

При розгляді мережевої атаки важливо, як по відношенню один до одного розташовуються суб'єкт і об'єкт атаки, тобто в одному або в різних сегментах вони знаходяться. На практиці між сегментну атаку здійснити значно важче, ніж внутрішньо сегментну. Між сегментна мережева атака представляє набагато більшу небезпеку, ніж внутрішньо сегментна. Це пов'язано з тим, що в разі між сегментної атаки об'єкт і безпосередньо атакуючий можуть перебувати на відстані багатьох тисяч кілометрів один від одного, що може істотно перешкодити заходам щодо відбиття атаки.

За рівнями моделі OSI, на якому здійснюється вплив[2]:

Фізичний рівень - моделі OSI, призначений безпосередньо для передачі потоку даних. Здійснює передачу електричних або оптичних сигналів у кабель і відповідно їхній прийом і перетворення в біти даних відповідно до методів кодування цифрових сигналів.[3] На цьому рівні працюють концентратори й повторювачі (ретранслятори) сигналу. Фізичний рівень визначає електричні, процедурні і функціональні специфікації для середовища передачі даних, в тому числі роз'єми, розпаювання і призначення контактів, рівні напруги, синхронізацію зміни напруги, кодування сигналу.

Канальний рівень - моделі OSI призначений для забезпечення взаємодії мереж на фізичному рівні й контролю за помилками, які можуть виникнути[3]. Отримані з фізичного рівня дані він упаковує в кадри даних, перевіряє на цілісність, якщо потрібно — виправляє помилки й відправляє на мережний рівень. Канальний рівень може взаємодіяти з одним або декількома фізичними рівнями, контролюючи цю взаємодією й керуючи нею. Специфікація *IEEE* 802 поділяє цей рівень на 2 підрівня — *MAC* (*Media Access Control*) регулює доступ до поділюваного фізичного середовища, *LLC* (*Logical Link Control*) забезпечує обслуговування мережного рівня. На цьому рівні працюють комутатори, мости й мережеві адаптери.

Мережевий рівень - моделі OSI, призначений для визначення шляху передачі даних. Відповідає за трансляцію логічних адрес й імен у фізичні, визначення найкоротших маршрутів[3], комутацію й маршрутизацію пакетів, відстеження неполадок і заторів у мережі. На цьому рівні працює такий мережний пристрій, як маршрутизатор.

Транспортний рівень - моделі OSI, призначений для доставлення даних без помилок, втрат і дублювання в тій послідовності, у якій вони були передані[3]. При цьому немає значення, які дані передаються, звідки й куди, тобто він визначає сам механізм передачі. Блоки даних він розділяє на фрагменти, розмір яких залежить від протоколу, короткі об'єднує в один, довгі розбиває. Протоколи цього рівня призначені для взаємодії типу точка-точка.

Сеансовий рівень - моделі OSI, відповідає за підтримання сеансу зв'язку, дозволяючи програмам взаємодіяти між собою тривалий час. Рівень управляє створенням і завершенням сеансу, обміном інформацією, синхронізацією завдань, визначенням права на передачу даних і підтримкою сеансу в періоди неактивності програм[3]. Синхронізація передачі забезпечується включенням у потік даних контрольних точок, починаючи з яких поновлюється процес при порушенні взаємодії.

Рівень представлень - моделі OSI, відповідає за перетворення протоколів кодування і декодування даних. Запити додатків, отримані з рівня додатків, він перетворить у формат для передачі по мережі, а отримані з мережі дані перетворить у формат, зрозумілий додаткам[3]. На цьому рівні може здійснюватися стиснення / розпакування або кодування і декодування даних, а також перенаправлення запитів іншому мережному ресурсу, якщо вони не можуть бути оброблені локально.

Прикладний рівень - моделі OSI, забезпечує взаємодію мережі й користувача. Саме на цьому рівні працюють всі прикладні програми, які використовують доступ до мережі, такі як оглядач веб-сторінок, електронна пошта, віддалений доступ до файлів та інші[3]. Всі протоколи рівнів нижчих за прикладний

займаються доставкою даних, але корисних для кінцевого користувача функцій не надають.

1.2 Системи виявлення мережевих атак, застосування та напрямки розвитку

Сучасна мережева інфраструктура є досить розвинутою і нараховує велику кількість обладнання і програмного забезпечення, яке підтримує роботу додатків, сервісів, мережі та мережевого обладнання. Для успішного функціонування, виявлення прогалин у безпеці, роботі сервісів необхідно мати інструмент, який дозволить виявити незаконні дії. Серед рішень є система виявлення атак, вона одна із найефективніших способів, яка пропонує повний та динамічний механізм безпеки для моніторингу, запобігання та протидії атакам. Зокрема, система виявлення атак буде збирати інформацію шляхом моніторингу мережі, стану системи, поведінки та використання системи, що може автоматично виявляти несанкціоновані дії користувачів системи та атаки зовнішніх зловмисників на систему.

Так як атака - спроба обійти політику безпеки системи, що надає зловмисникам легший доступ для отримання або модифікації інформації, навіть руйнуючи систему. У зв'язку з технологіями, що розробляються в системах локальної мережі та бездротової мережі, існують серйозні загрози безпеці мережі, особливо безпеці систем. Оскільки ми зараз перебуваємо в епохі машинного навчання та великих даних[4], кібербезпека в системах бездротової та локальної мережі є важливою перспективою для користувачів.

Вторгненням можна визначити будь-який несанкціонований вид діяльності, що спричиняє пошкодження інформаційної системи. Це означає, що будь-яка атака, яка може становити можливу загрозу конфіденційності, цілісності чи доступності інформації, буде розглядатися як вторгнення. Наприклад, діяльність, яка може призвести до того, що комп’ютерні служби не реагують на законних користувачів, вважається вторгненням.

IDS - поєднання двох слів вторгнення та система виявлення. Вторгнення відноситься до несанкціонованого доступу до інформації в комп'ютері або мережевих системах, щоб порушити її цілісність, конфіденційність або доступність. Система виявлення є механізмом безпеки для виявлення такої незаконної діяльності приклад на малюнку 1.1.

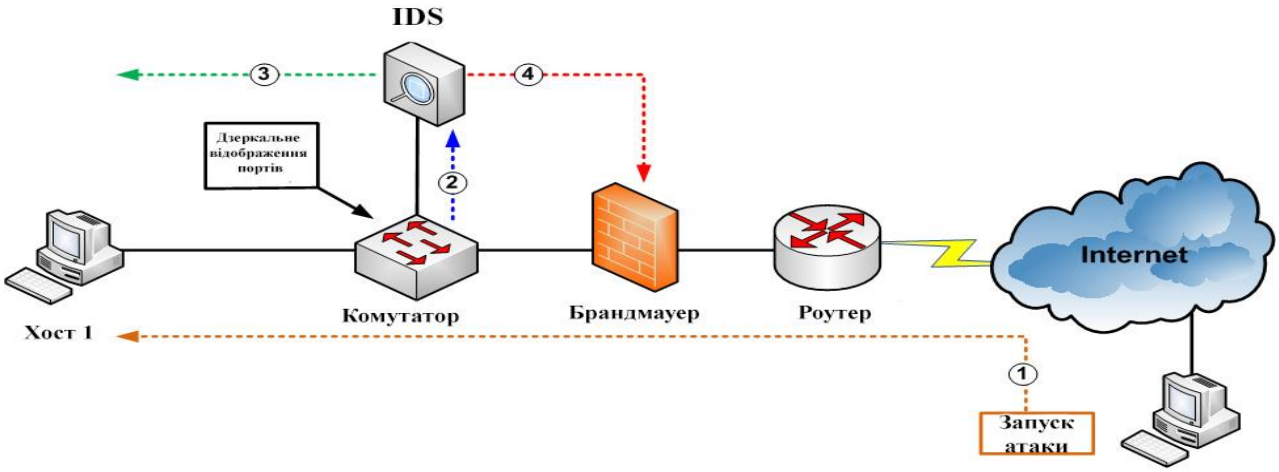


Схема системи виявлення мережевих атак

Система виявлення вторгнень виявляє шкідливі дії, збираючи та аналізуючи поведінку мережі, журнал безпеки та іншу інформацію, доступну в мережі серед підключених комп’ютерів [5]. По суті, система виявлення вторгнень перевіряє наявність ненормальної поведінки щодо політики безпеки системи та ознак нападу в системі, яка здатна захистити систему за допомогою вчасної реакцій і протидійних дій у режимі реального часу. За традиційними налаштуваннями системи виявлення вторгнень працює як розумне, активне та ефективне доповнення до брандмауера, який насправді діє як пасивний захист для атак.

Традиційна система виявлення вторгнень спочатку побудована на технології зловживань, яка в основному витягує характеристики або правила поведінки вторгнення. Після появи технології виявлення аномальної поведінки за допомогою традиційних моделей машинного навчання система виявлення вторгнень розробляє статистичне моделювання ймовірностей для нормальної поведінки, яке може аналізувати аномальну поведінку з великими відхиленнями.

Однак така система може мати незадовільні результати через низькі можливості у визначенні проблемного простору та складність у моделюванні шкідливих дій. Існує в інформаційному світі система запобігання вторгнень IPS, тобто система активного захисту. Як і IDS, він намагається визначити потенційні загрози на основі функцій моніторингу захищеного хоста або мережі і може використовувати сигнатуру, аномалію або методи гібридного виявлення. На відміну від IDS, IPS вживає заходів для блокування або усунення виявленої загрози. Хоча IPS може викликати тривогу, це також допомагає запобігти вторгненню.

IPS діляться на чотири категорії.

Перший – це система запобігання мережевих вторгнень , яка відстежує всю мережу за підозрілою активністю.

Другий тип - системи аналізу мережевої поведінки , які вивчають потік трафіку для виявлення незвичайних транспортних потоків, які можуть бути результатами атаки, таких як розподілена відмова в наданні послуг .

Третій вид - бездротові системи запобігання вторгнень, яка аналізує бездротові мережі на предмет підозрілого трафіку.

Четвертий тип - це хост-системи запобігання вторгнень, де встановлюється програмний пакет для моніторингу діяльності одного хоста.

Як згадувалося раніше, IPS робить активні кроки, такі як скидання пакетів, які містять шкідливі дані, скидання або блокування трафіку, що надходить з образливої *IP*-адреси.

Зрештою, система запобігання вторгнень проти порівняння системи виявлення вторгнень зводиться до того, які дії вони вживаються, якщо виявлено таке вторгнення. IDS призначений лише для оповіщення про потенційний інцидент, що дозволяє аналітику Центру операцій безпеки розслідувати подію і визначити, чи вимагає вона подальших дій. IPS, з іншого боку, вживає заходів, щоб заблокувати спробу вторгнення або іншим чином виправити інцидент.

Хоча їхні відповіді можуть відрізнятися, вони служать подібним цілям, потенційно роблячи їх зайвими. Незважаючи на це, обидва вони мають переваги та сценарії розгортання, до яких один краще підходить, ніж інший:

Система виявлення вторгнень призначена для виявлення потенційного інциденту, створюючи оповіщення, щоб запобігти інциденту. Хоча це може здатися хорошим рішенням для систем з високими вимогами до доступності, таких як промислові системи управління та інша критична інфраструктура. Для цих систем найважливішим є те, що системи продовжують працювати, а блокування підозрілого і потенційно шкідливого трафіку може вплинути на їх роботу. Отримуючи повідомлення про проблему людина - оператор дозволяє їм оцінити ситуацію і прийняти обґрунтоване рішення про те, як реагувати.

Система запобігання вторгнень: IPS, з іншого боку, призначена для вжитку заходів для блокування всього, що він вважає загрозою для захищеної системи. Оскільки атаки та шкідливі програми стають швидшими та витонченішими, це корисна можливість, оскільки існує обмежений потенційний збиток, ніж може завдати атака. IPS ідеально підходить для середовищ, де будь-яке вторгнення може завдати значної шкоди, наприклад, бази даних, що містять конфіденційні дані.

IDS та IPS мають свої переваги та недоліки. При виборі системи для потенційного випадку використання важливо враховувати компроміси між доступністю системи і зручністю використання і необхідністю захисту. IDS залишає вікно для зловмисника, щоб завдати шкоди цільовій системі, в той час як помилкове позитивне виявлення IPS може негативно вплинути на зручність використання системи.

Крім того, нижче в таблиці 1.1 перераховані різниця між IDS проти IPS в деталях.

Таблиця 1.1

Характеристика *IDS* та *IPS*

Параметр	IPS	IDS
Абревіатура	система запобігання вторгнень	система виявлення вторгнень
Філософія	IPS - це пристрій, який перевіряє трафік, виявляє його, класифікує, а потім активно зупиняє шкідливий трафік від атаки.	IDS - це пристрій або програмний додаток, який відстежує трафік на порушення шкідливої активності або політики і надсилає оповіщення про виявлення.
Принцип роботи	перевіряє трафік у реальному часі та шукає схеми руху або підписи атаки, а потім запобігає нападам на виявлення	виявляє трафік у реальному часі та шукає схеми трафіку або підписи атаки, і вони генерують сповіщення
Режим конфігурації	вбудований режим , як правило, у шарі 2	кінцевий хост (через проліт) для моніторингу та виявлення
Розміщення	вбудований зазвичай після брандмауера	Не вбудований через проліт порту (або дотиком)
Дії щодо виявлення несанкціонованого трафіку	Запобігання руху по виявленню аномалії	Оповіщення/тривоги про виявлення аномалії
Пов'язані термінології	– фільтрація пакетів зі станом	– виявлення на основі аномалії виявлення підпису

Висновки до першого розділу:

У даному розділ було визначено місце систем виявлення мережевих атак та потрібність новітніх методів виявлення атак, проаналізовано IDS та IPS. Кіберзлочинці орієнтуються на користувачів комп’ютерів, використовуючи складні методи, а також стратегії соціальної інженерії. Деякі кіберзлочинці стають дедалі витонченішими та мотивованими. Вони продемонстрували свою здатність приховувати свою особистість, приховувати своє спілкування, віддаляти свою особу від нелегальної вигоди та використовувати інфраструктуру, стійку до компромісів. Тому для комп'ютерних систем стає все більш важливим захист за допомогою вдосконалених систем виявлення

вторгнень, здатних виявляти сучасні шкідливі програми. Для проектування та побудови таких систем IDS необхідно мати повний огляд переваг та обмежень сучасних досліджень якому розглянуто в даному розділі.

РОЗДІЛ 2
АНАЛІЗ МЕТОДІВ ДЛЯ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК НА ОСНОВІ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ

2.1 Аналіз нейромережевих технологій

Штучний інтелект (*Artificial Intelligence,* AI) - це область досліджень, метою якої є дати когнітивні здібності комп'ютерів, щоб програмувати їх для вивчення і вирішення проблем. Його мета заснована на імітуванні комп'ютера з людським інтелектом[6]. Штучний інтелект не може повністю імітувати людський інтелект, комп'ютери можуть бути запрограмовані тільки на деякі аспекти людського мозку.

Штучний інтелект може ставитися до будь-чого, а саме від комп'ютерних програм для гри в шахи до систем розпізнавання мови. В цілому системи штучного інтелекту можна розділити на три групи[19]:

– обмежений штучний інтелект;
– загальний штучний інтелект;
– надрозумний штучний інтелект.

Штучні нейронні мережі (*Artificial Neural Networks, ANNs*) - це один з напрямків досліджень в області штучного інтелекту, засноване на спробах відтворити нервову систему людини[7]. А саме: здатність нервової системи навчатися і виправляти помилки, що має дозволити змоделювати, хоча і досить грубо, роботу людського мозку.

Машинне навчання (*Machine Learning, ML*) є одним з напрямків штучного інтелекту. Основний принцип полягає в тому, що машини отримують дані і навчаються на них[8]. В даний час це найбільш перспективний інструмент для бізнесу, заснований на штучному інтелекті. Системи машинного навчання дозволяють швидко застосовувати знання, отримані під час навчання на великих наборах даних, що дозволяє їм процвітати в таких завданнях, як розпізнавання осіб, розпізнавання мови, розпізнавання об'єктів, переклад, і багатьох інших. На відміну від програм з закодованими вручну інструкціями для виконання

конкретних завдань, машинне навчання дозволяє системі навчитися самостійно розпізнавати шаблони і робити прогнози.

Глибоке навчання (*Deep Learning,* DL) є підмножиною машинного навчання і являє собою складний набір нейронних мереж з великою кількістю рівнів обробки, які розвивають високі рівні абстракції[9]. Вони зазвичай використовуються для складних завдань, таких як розпізнавання зображень, класифікація зображень і ідентифікація ручної писанини. Глибоке навчання може бути дорогим і вимагає величезних масивів даних для навчання. Це пояснюється тим, що існує величезна кількість параметрів, які необхідно налаштувати для алгоритмів навчання, щоб уникнути помилкових спрацьовувань. Наприклад, алгоритму глибокого навчання може бути дано вказівку дізнатися, як виглядає кішка. Щоб зробити навчання, потрібно величезна кількість зображень для того, щоб навчитися розрізняти дрібні деталі, які дозволяють відрізнити кішку від, скажімо, собаки або крокодила, чи лисиці.

2.2 Методи виявлення атак за допомогою DL

Методи виявлення атак на основі автокодера

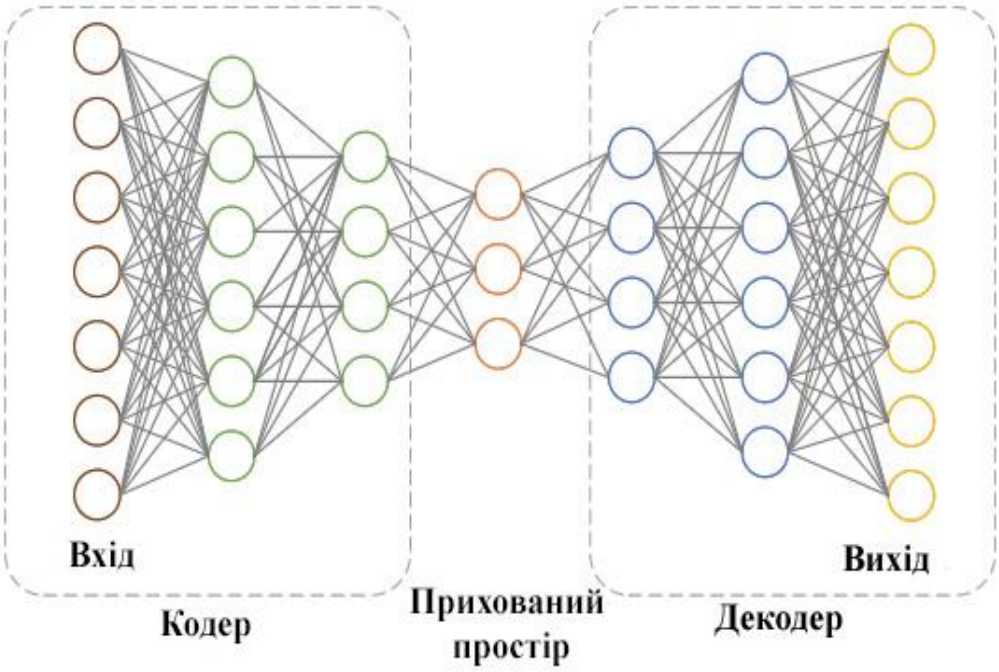
***Autoencoders* (AE)** - це нейронні мережі, які мають на меті копіювати свої входи у свої виходи. Вони працюють, стискаючи вхідні дані у подання з прихованим простором, а потім реконструюючи вихідні дані з цього подання. Цей тип мережі складається з двох частин:

– Кодер

Частина мережі, яка стискає вхідні дані у подання у прихованому просторі. Він може бути представлений функцією кодування $h = f(x)$.

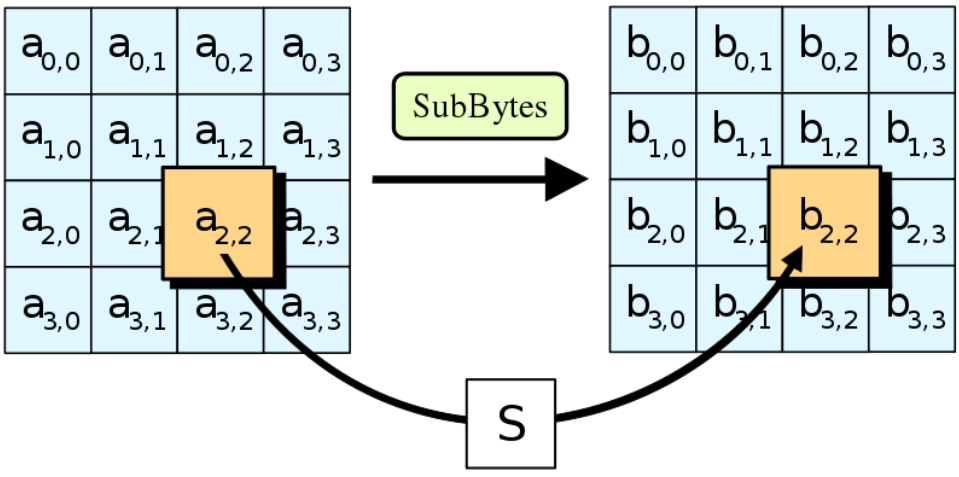
– Декодер

Частина спрямована на реконструкцію вхідних даних із представлення прихованого простору. Він може бути представлений функцією декодування $r = g(h)$.



2.1 Приклад нейронної мережі автоенкодера

АЕ здатний спочатку стиснути вхідні дані в деякий простір об’єкта, а потім реконструювати цей же простір у вихідні дані. Оскільки АЕ можна розглядати як типовий алгоритм (малюнок 2.2), що представляє навчання, він широко використовується для зменшення розмірів. Дослідники в галузі кібербезпеки також застосовують АЕ для представлення ненормальної поведінки в стислому просторі функцій, що приносить перевагу динамічного представлення для невідомої категорії атак.



2.2 Архітектуру АЕ, як алгоритм стиснення даних зі структурою нейронної мережі.

Автокодери вивчаються автоматично на прикладах даних. Це означає, що легко навчити спеціалізовані екземпляри алгоритму, які будуть добре працювати на певному типі введення, і що для цього не потрібні нові технічні засоби, лише відповідні навчальні дані.

Однак, автокодери погано справляються зі стисненням зображень. Оскільки автокодер навчається на певному наборі даних, він досягне розумних результатів стиснення даних, подібних до використовуваного навчального набору, але буде поганим загальним компресором зображень. Такі техніки стиснення, як *JPEG*, справляються значно краще.

Автокодери вивчаються автоматично на прикладах даних. Це означає, що легко навчити спеціалізовані екземпляри алгоритму, які будуть добре працювати на певному типі введення, і що для цього не потрібні нові технічні засоби, лише відповідні навчальні дані.

Існує чотири типи автокодерів:

– Ванільний автокодер;

У найпростішій формі автокодер - це тришарова мережа, тобто нейронна мережа з одним прихованим шаром.

– Багатошаровий автокодер;

Якщо одного прихованого шару недостатньо, то виконується розширення автокодера до більш прихованих шарів.

– Свертовий автокодер;

Вхідне зображення зменшується, щоб отримати приховане представлення менших розмірів і змусити автокодер вивчити стиснуту версію зображень.

– Регуляризований автокодер.

Існують інші способи, якими обмежують реконструкцію автокодера, ніж накласти прихований шар меншої розмірності. Замість того , щоб обмежувати потужність моделі, зберігаючи кодер і декодер неглибоко і розмір коду використовують функцію втрат, що мають інші властивості, крім можливостей скопіювати його вхід до виходу. На практиці зазвичай зустрічається два типи регуляризованого автокодера:

– розріджений автоенкодер;
– деноазуючий автоенкодер.

Розріджені автокодери зазвичай використовуються для вивчення функцій для іншого завдання, такого як класифікація. Автокодер, який регулюється як розріджений, повинен реагувати на унікальні статистичні особливості набору даних, на якому він навчався, а не просто виконувати функцію ідентичності. Таким чином, навчання виконанню завдання копіювання з обмеженим показником може дати модель, яка вивчила корисні функції як побічний продукт.

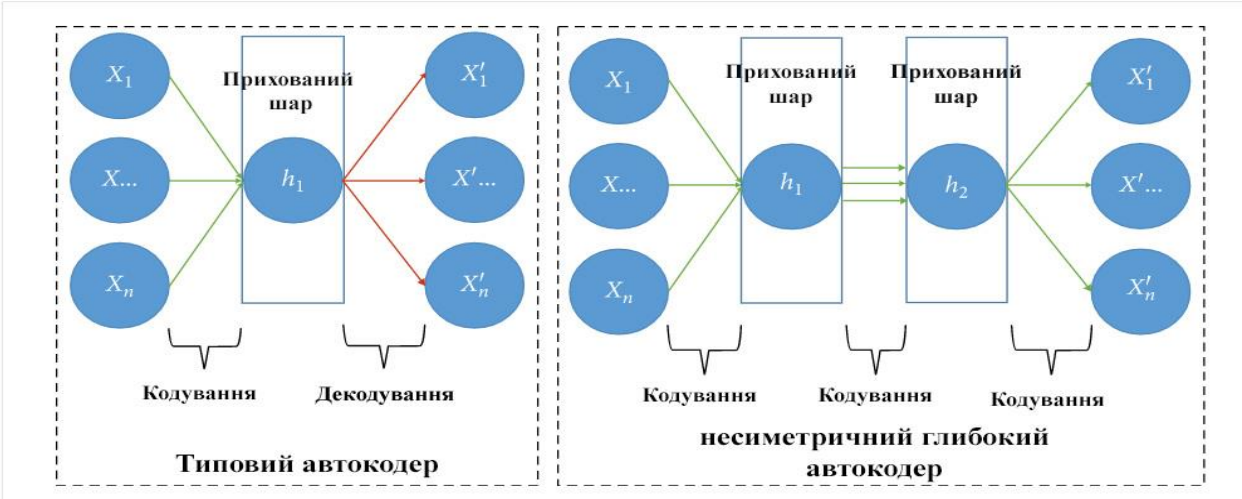
Інший спосіб, може обмежити реконструкцію автокодера, тобто накласти обмеження на його втрату. Наприклад, додати термін регуляції до функції збитків. Завдяки цьому автокодер навчається.

Видалення шуму автокодера. Замість того, щоб додавати штраф до функції втрат, потрібно отримати автокодер, який дізнається щось корисне, змінивши термін помилки відновлення функції втрати. Це можна зробити, додавши трохи шуму вхідного зображення і змусити автокодер навчитися його видаляти. Таким чином, кодер витягне найважливіші функції та вивчить обґрунтоване представлення даних.

Для отримання інформативних дескрипторів функцій з вихідних даних мережевого трафіку, використовують один з методів виявлення вторгнення в мережу шляхом складання розширених згорткових AE (DCAE), що є вдалою комбінацією навчання самоучкам та представництву[11]. Зокрема, оригінальні дані мережевого трафіку спочатку трансформуються у вектор за допомогою етапу попередньої обробки. Під час тренінгу без нагляду DCAE вивчають ієрархічну структуру представлення ознак із великої кількості немаркованих зразків. Згодом користуються алгоритмом зворотного розповсюдження та декількома позначеними екземплярами, щоб відредагувати та покращити здатність опису функцій. Фактично, використання оригінального мережевого трафіку та попередньої підготовки без нагляду робить їх модель більш адаптивною та гнучкою для роботи зі складними вихідними даними.

Спростити виявлення вторгнень за допомогою моделей АЕ, [12] застосувавши несиметричний глибокий АЕ (NDAE) для безконтрольного змішування функцій, що успішно знижує витрати на обчислення аналізу, поєднуючи АЕ з неглибоким навчанням. Зокрема, NDAE має додатковий етап кодування порівняно із типовим АЕ, що може зменшити складність та підвищити точність моделі. Структура висвітлена на (малюнку 2.3), де спостерігається екстракторні ієрархічні особливості. Наприкінці NDAE застосовується в структурі випадкових лісів для розпізнавання ненормальних ситуацій за допомогою представлення ознак, отриманих від *NDAE*.

АЕ здатний вивчити потенційну поведінку модель невідомих атак[13]. Функції зі структурою АЕ для різних програм кібербезпеки, що складається з двох етапів навчання, тобто попередньої підготовки та тонкої настройки. Перший етап призначений для пошуку відповідної відправної точки для етапу тонкої настройки. Після визначення параметрів на етапі попередньої підготовки, етап точної настройки охоплюватиме пропонований опис функцій для вхідних даних. Запропонована схема навчання функцій може значно зменшити розміри функцій, тим самим мінімізуючи вимоги зберігання.

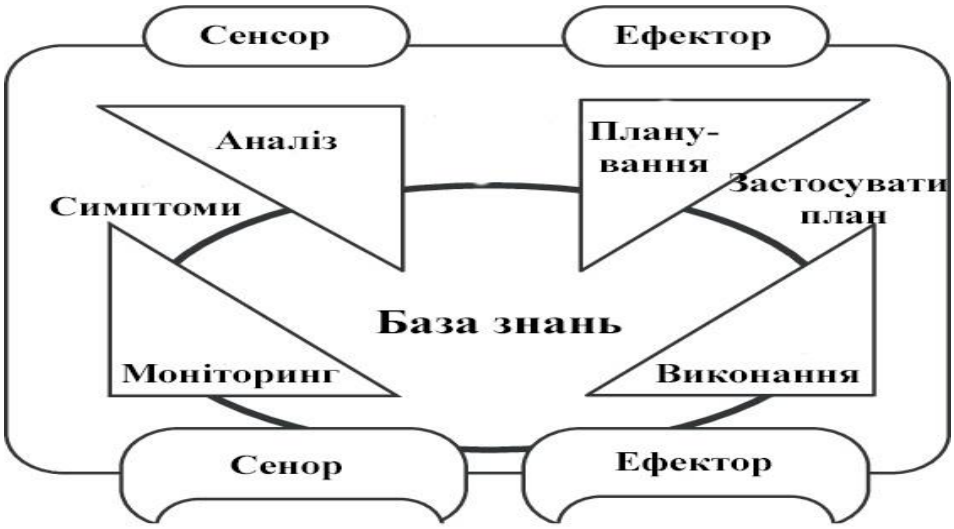


2.3 Структура АЕ, розроблена з несиметричними множинними прихованими шарами.

Для побудуви гнучкої системи виявлення атак вторгнень, використовується рідкісна АЕ та шар регресії *softmax* для побудови та навчання

методом STL для навчального процесу[14]. Зокрема STL розділяється на два етапи, де розріджений АЕ використовується спочатку для некерованого вивчення ознак, а *softmax*-регресія використовується для класифікації після вилучення ознак. Насправді, використання STL може значною мірою покращити здатність до навчання побудованої мережі, яка стикається з невідомими атаками, де нові категорії атак можна поступово аналізувати під час виконання без проблем з підготовкою з нуля.

На основі гнучкої системи, створений[15] більш потужний підхід із структурою MAPE-K (малюнок 2.4), яка може створити систему виявлення зловживання з масштабованими, само адаптивними та автономними характеристиками. Він може витягувати узагальнені функції для реконструкції проблем, навіть стикаючись із невідомим середовищем та використовуючи немарковані дані.



2.4 Струткура гнучкоъ системи MAPE-K

Одна з основних проблем виявлення атак – вилучення функцій. Розглядаючи АЕ як структуру для стиснення інформації та формування об’єктів, використання АЕ приносить переваги автоматичної та динамічної побудови об’єктів, що призводить до високої точності виявлення заздалегідь визначених атак, що існують у наборах даних. Досліджуючи варіанти та невідомі атаки, які

є основними характеристиками кібербезпеки, дослідники наголошували на стратегіях самонавчання, щоб зробити АЕ більш потужним.

Методи виявлення нападів, засновані на глибокій вірі

Глибинна мережа переконань (*deep belief net,* DBN) розділяють на дві категорії

– Обме́жена маши́на Бо́льцмана (ОМБ, restricted Boltzmann machine, RBM);
– Нейронна мережа зворотного розповсюдження (back propagation neural network, BPNN).

Обмежена машина Больцмана - це мережа симетрично зв'язаних стохастичних двійкових одиниць, а також вона складається з RBM. Вона містить набір видимих та прихованих одиниць. На відміну від моделі DBN, усі зв’язки між шарами в моделі DBM неспрямовані. DBM має багато переваг: він зберігає та виявляє представлення шарів вхідних даних за допомогою ефективної відповідної процедури, його можна навчити немаркованим даним, а параметри всіх шарів можна спільно оптимізувати у функції правдоподібності. Однак DBM має недолік, що час навчання зростає в геометричній прогресії із збільшенням розміру машини та кількості рівнів з'єднання, що робить масштабне вивчення моделі DBM непростим. Отже, ми просто зменшуємо розмір документа і усуваємо шум за допомогою моделі DBM в нижніх шарах, а потім продовжуємо навчання з моделлю DBN.

По суті, RBM - це випадкова структура генеруючої нейронної мережі, яка є неспрямованою графічною моделлю, що складається з різних шарів, побудованих видимими нейронами та прихованими нейронами. Через характеристики RBM ефективно використовувати для DBN, щоб тренувати шар за шаром.

Програмне забезпечення включає послідовності операційних кодів і DBN для виявлення зловмисного програмного забезпечення, використовуючи неконтрольоване навчання для попередньої підготовки багатошарової генеративної моделі, щоб допомогти DBN вирішити проблему переобладнання[16]. Структуру на (малюнок 2.5), де висвітлено робота DBN як

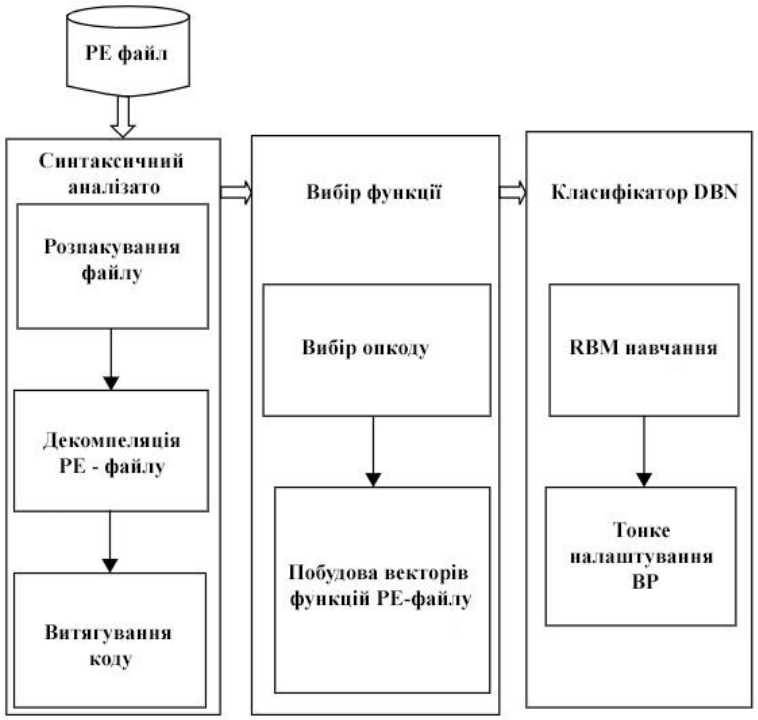
класифікатор у всьому робочому процесі з етапами навчання RBM. За допомогою додаткових немаркованих даних DBN може досягти точності до 96%, що перевершує три інші традиційні моделі штучного інтелекту, тобто *SVM*, *kNN* та дерево рішень. Однак їх методи не виправдані іншими показниками.

Робочий процес підходу до виявлення шкідливого програмного забезпечення складається з трьох основних компонентів[16]:

– аналізатор *PE* (Portable Executable);

– екстрактор функцій;

– модуль виявлення шкідливого програмного забезпечення.



2.5 Структура роботи DBN

Зазначається, що DBN є основним класифікатором модуля виявлення шкідливих програм.

Оскільки поведінкові характеристики спеціальних мереж спричинили великі виклики безпеці мереж, [17] глибинна мережа переконань, заснована на спеціальній структурі виявлення вторгнень мережі. Модель DBN містить шість модулів:

– вузол бездротового моніторингу для отримання даних;

– модуль злиття даних для злиття корисних даних та видалення надмірності;
– навчальний модуль DBN;
– модуль вторгнення DBN для навчання та визначення;
– модуль відповіді, який виражає результати запропонованої моделі для користувачів.

DBN для виявлення атак вторгнення[18], пропонується використовувати ефективну платформу для виявлення спроб вторгнення в мережеві трафіки. Побудована система спочатку використовує цифрове кодування та стандартизований метод для вибору функцій, а потім використовує DBN для класифікації вторгнення в мережу, присвоюючи мітку класу кожному вектору ознак. Згідно з результатами експериментів та аналізом, побудована система може не тільки виявляти атаки, але і точно ідентифікувати та класифікувати мережеві дії за обмеженими, неповними та нелінійними джерелами даних.

Існує ще багато невирішених проблем, таких як надлишкова інформація, яку легко зафіксувати в локальному максимумі. Для вирішення цих проблем [19] виявляють атаки вторгнення, залучаючи DBN та ймовірнісну нейронну мережу (PNN). По-перше, масштабувати вихідні та вхідні дані до низько вимірювальних, використовуючи можливості нелінійного опису DBN. Тим часом DBN може підтримувати основні характеристики вихідних даних у поданні. По-друге, алгоритм оптимізації частинок використовувати для зменшення розміру прихованих вузлів кожного шару. По-третє, PNN вводити для класифікації низько вимірної інформації.

Виявлення атак у режимі реального часу є найбільша проблема, [20] для усунення проблеми існує метод виявлення аномалії, заснований на DBN, який складається лише з одного прихованого рівня RBM та шару тонкої настройки, побудованого за допомогою класифікатора логістичної регресії. Метод пропонує можливість впровадження глибокого навчання для виявлення атак на платформах з низькими обчислювальними ресурсами, таких як безпілотники, стільникові телефони та персональні комп'ютери, що значно розширює сценарії використання таких методів.

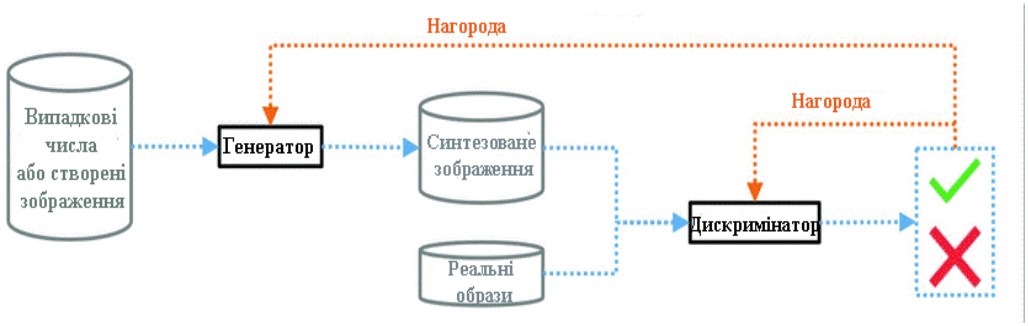
Оскільки традиційні підходи до виявлення вторгнень стикаються з труднощами при роботі з високошвидкісними мережевими даними і в даний час не можуть виявити невідомі атаки, [21] пропонується модель виявлення мережевих атак, що включає обчислення потоку та глибоке навчання, яка складається з двох частин: алгоритму виявлення в реальному часі, заснованого на частих шаблонах, та алгоритму класифікації на основі DBN та SVM. Обробка даних потоку розсувного вікна може реалізувати виявлення в реальному часі.

Генеративні методи виявлення атак на основі змагальних мереж

Генеративна змагальна мережа (GAN) складається з двох модулів, моделі генератора *G* та моделі дискримінатора *D*. Найкращої продуктивності мережі досягають дві моделі, які борються одна проти одної. На малюнку 2.6 показано склад моделі GAN. Метою мережі генераторів є заплутати антагоніста, а метою дискримінатора є розрізнення між сформованими наборами даних та вихідними наборами даних. Мережа поколінь, що складається з багатошарових персептронів, які супроводжується дискримінантною мережею, що складається з багатошарових персептронів. Вхід дискримінатора вибирає реальні вибірки або вихід генераторної мережі. Результатом дискримінатора є ймовірність того, що вхідна картина є реальною. Коли дискримінаційна мережа розрізняє, чи є вихід генератора справжнім зразком чи ні, він може вказати, який зразок ближчий до реального зразка за допомогою градієнта, а потім налаштувати мережу генерації за допомогою цієї інформації. Функція GAN виражається наступним чином (2.1):

$$\min \max V(D,G) = E_{x\sim P_{data}(x)}\left[loq\ D(\frac{x}{y})\right] + E_{z\sim P_z(z)}\left[loq\ (1 - D\left(G\left(\frac{z}{y}\right)\right))\right] \quad 2.1$$

Завдяки властивості виявляти властиві дані для генерації нових зразків, генеративна змагальна мережа (GAN) є одним з найбільш перспективних методів навчання без нагляду, запропонованих за останні роки.



2.6 Модель GAN

Основна ідея GAN походить від ідеї гри з нульовою сумою. Коли GAN застосовується до глибокої нейронної мережі, він продовжує грати в ігри між генератором і дискримінатором, та здатністю вивчити уявлення розподілу фактичних даних. Основна задача GAN полягає в імітації, моделюванні та вивченні характеристик розподілу реальних даних, наскільки це можливо, тоді як завдання полягає в тому, щоб розрізнити, чи вхідні дані походять від реальних даних чи виводу. Завдяки безперервної конкуренції між цими внутрішніми моделями, можливість генерації.

Незважаючи на те, що GAN є новим у концепції та важким у навчальному процесі.[22] Підходи, засновуються на GAN, для виявлення атак на бездротовому зв'язку та захисту їх на основі зібраної інформації про атаки, модель складається з передавача, приймача та перешкод. Підготовлений класифікатор приймається передавачем для прогнозування поточного стану каналу та вирішення питання надсилання на основі останніх результатів зондування, тоді як інший метод збирає стан каналу для побудови класифікатора, який може передбачити наступну передачу та успішно її заблокувати.

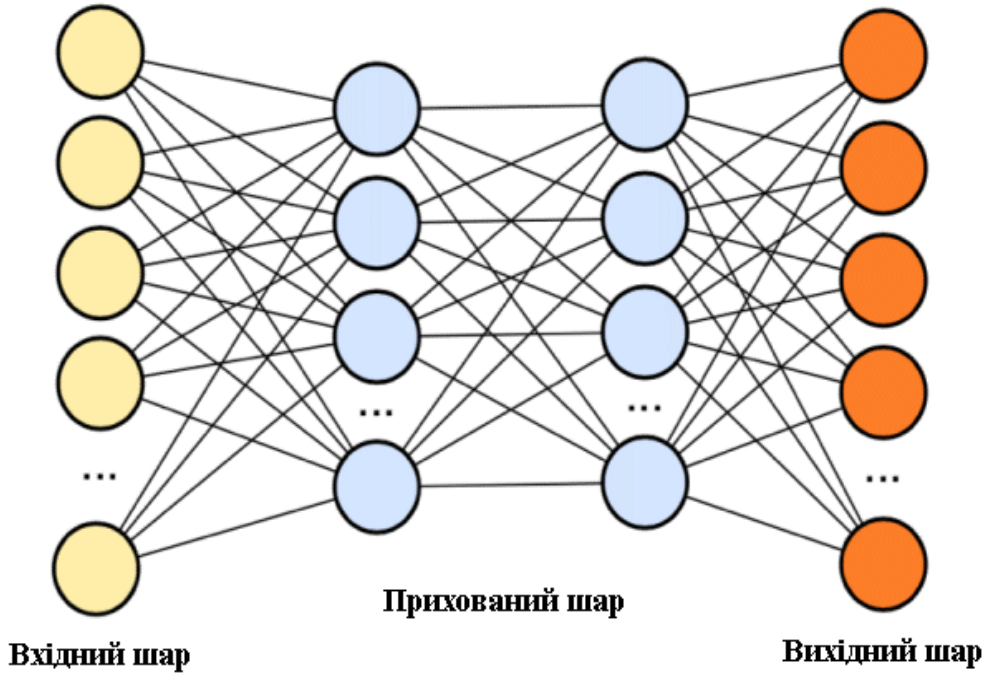
Використання технології машинного навчання для виявлення фішингу, тобто URL-адреси фальшивої веб-адреси, користується популярністю завдяки своїй високій ефективності та реагуванню в режимі реального часу. Однак суперник може обійти алгоритм класифікації *URL*-адрес, модифікуючи компоненти. Щоб вирішити цю проблему, [23] пропонується згенерувати приклади фішингу на основі URL-адрес за допомогою генератора GAN, який потім передається на дискримінатор, тобто детектор фішингу чорної скриньки.

Моделі GAN, де її структура показана на (малюнок 2.6), генерує можливі порушені версії реальних прикладів фішингу та перетворює їх на приклади суперників. Мережа дискримінаторів вчиться класифікувати як згенеровані приклади, так і реальні, що працюють як фішинг-детектор, де параметри та ваги генератора оновлюються інформацією, що надходить від дискримінатора. Після тестування із загальнодоступним набором даних фішингу їх експериментальні результати показують, що запропонований ними GAN є успішним, уникаючи великої кількості невідомих прикладів фішингу.

GAN не часто використовується для поля виявлення атак. Насправді, GAN швидко розвивається з точки зору структур, алгоритмів. В даний час GAN показав багатообіцяючі результати у багатьох сферах, що змушує вважати пропонування нової техніки синтезу спроб є досить важливим для створення захисного механізму. Такий новий захисний механізм може додатково виконувати безліч завдань, таких як запобігання спробам фішингу нульових днів, виконання спаму думок та виявлення атак вторгнень. Тому існує широкий дослідницький простір для зв’язку структури GAN із системою виявлення атак.

Методи виявлення атак на основі глибоких нейронних мереж

DNN застосовуються в кількох областях протягом останніх років, серед яких розпізнавання мови було значно покращено. Крім того, класифікація зображень та розуміння природної мови також отримують вигоду від DNN.



2.7 Структура моделі DNN

Модель DNN - це пряма нейронна мережа з декількома шарами, які є нелінійними трансформаторами. Він перетворює вхідні вектори у відповідні вихідні вектори та сприяє подальшому розпізнаванню. Для системи автоматичного розпізнавання мови (ASR) використовується модель DNN для перетворення особливостей промов у мітки, що відповідають станам у прихованих марковських моделях (HMM).

Структура моделі DNN (малюнок 2.7)., як правило, складається з наступних частин :

– вхідний рівень, який відновлює вхідні дані від функцій;
– приховані блоки, які відновлюють проміжні результати перетворення;
– вихідний рівень, який відновлює результати, що представляють кінцеві результати перетворення;
– активаційні функції, що імітують властивості біологічних нейронів;
– ваги, які представляють важливість кожної одиниці у вхідному шарі або прихованих шарах;
– зміщення, яке забезпечує основні моменти для функцій активації.

У DNN систем ASR звичайна логістична функція є однією з найбільш широко використовуваних функцій активації, що називається «сигмовидною функцією» завдяки своїй формі. Іншими словами, логістична функція формує властивість, подібну до біологічного нейрона, виконуючи стан спокою або активований стан. З цією логістичною функцією також існує порогова точка, подібна до порогового потенціалу біологічного нейрона. Ця властивість призводить до того, що більшість блоків в DNN мають лише два стани при роботі. Оскільки два стани здатні представляти дві різні моделі, ця функція широко використовується для вирішення проблем класифікації.

Математично, логістична функція (2.2) монотонно зростає і нелінійна з визначенням області та діапазону. Його вираженням є $(-\infty,+\infty)(0,1)$

$$y = \frac{1}{(1+e^{-x})} \qquad 2.2$$

Крива логістичної функції. Неважко відзначити, що більшість змін відбувається в середній частині. У міру збільшення, збільшення. Зростання є значним, коли близький до 0. Навпаки, майже стабільний при 0, коли набагато менше 0, і стабільний при 1, коли набагато більше 0. Для класифікації шаблонів можна встановити особливості шаблонів і результати класифікації - 0 або 1. Криву можна також перекласти, додавши константи до (2.3), а саме

$$y = \frac{1}{(1+e^{-x+x_0})} + y_0 \qquad 2.3$$

Де і є константи. $x_0 y_0$

DNN розпізнається як багатошаровий завдяки характеристиці декількох прихованих шарів. Така багатошарова функція дає перевагу для вираження складних функцій з меншою кількістю параметрів, що робить DNN здатним полегшити завдання з вилучення та навчання зображень. По суті, у DNN існують три категорії шарів.

Рішення проблеми безпеки мережі є [24] модель DNN для виявлення аномалії на основі потоку. Перша спроба застосувати DNN для мережевої безпеки призводить до відносно простого *DNN*, який складається з одного вхідного рівня, трьох прихованих шарів та одного вихідного рівня.

Для посилення здатності DNN, [25] використовується нова мережева структура під назвою *HashTran*-DNN для класифікації шкідливих програм *Android*. Архітектурний дизайн показує інноваційний момент, полягає у перетворенні вхідних зразків за допомогою хеш-функцій для збереження характеристик. Після перетворення вхідних даних *HashTran*-DNN використовує AE для виконання завдання відмовки, так що класифікатор DNN може отримувати інформацію про місцевість у потенційному просторі для кращої продуктивності, ефективно захищатися від чотирьох спеціальних тестових атак, де стандартні DNN не можуть виявити всі ці атаки.

Шкідливі атаки постійно змінюються і відбуваються на дуже великих обсягах, що вимагають масштабованих рішень. Для вирішення цієї проблеми використовують структуру DNN із масштабованою та гібридною конструкцією. [26] В режимі реального часу вона спостерігає за мережевим трафіком та подіями рівня хосту, активно попереджаючи можливі мережеві атаки. Зокрема, структура використовує масштабовану обчислювальну архітектуру, метод подання тексту та DNN, щоб задовольнити вимогу обробляти великі дані, де DNN може допомогти покращити продуктивність своєї моделі з функціями нелінійної активації.

Для адміністратора мережі надзвичайним завданням є запобігання вторгненню зловмисних мережевих хакерів та підтримка мережевої системи та комп’ютера в безпечному та нормальному робочому стані. [27] Метод виявлення вторгнень у мережу, заснований на глибокому навчанні, який використовує глибоку нейронну мережу для вилучення особливостей даних моніторингу мережі, а нейронні мережі використовуються для класифікації типів вторгнень.

Методи виявлення атак на основі згорткових нейронних мереж

Згорткова нейронна мережа - це глибока пряма нейронна мережа, яка витягує особливості, вивчаючи вхідну картинку шар за шаром. CNN використовує ядро згортки для вилучення функцій, і воно містить тришарову структуру, згорткові шари, об'єднання шарів та повністю пов'язані шари. Різні шари мають різні функції. Ці функціональні шари складаються з багатьох

нейронів, і кожен нейрон з'єднує лише частину нейронів у сусідньому шарі. Це зменшує складність мережі та покращує ефективність розрахунку.

Згортковий шар складається з деяких згорткових нейронів і розглядається як шар вилучення ознак. Розмір ядра згортки визначатиме розмір вихідної карти функцій. Після того, як функція ядра переведена та транспортується, розмір карти об’єктів (2.4) обчислюється за таким рівнянням:

$$\begin{cases} N_x^l = \frac{N_x^{l-1} - K_x^l + 2P_x^l}{S_x^l} \\ N_y^l = \frac{N_y^l - K_y^l + 2P_y^l}{S_y^l} \end{cases} \quad 2.4$$

Де l - поточна кількість шарів, K - розмір ядра згортки, значення пікселя заливки та S - розмір кроку. Після операції згортки це нелінійне перетворення функції активації. Алгоритм зворотного розповсюдження отримує ваги та параметри кожного нейрона, і вираз нейрону згорткового шару (2.5) такий:

$$x_j^l = Relu\left(\sum_{i \epsilon M_j} x^{l-1} w_{ij}^l + b_j^l\right) \quad 2.5$$

Де M - розмір фільтра, a і b - вага з'єднання та зміщення відповідно.

Шар об’єднання виконує відображення об’єктів, як правило, між двома згортковими шарами. Операції об’єднання включають максимальне об’єднання та середнє об’єднання. Також пропонуються перекриваються пули програм. Шляхом зменшення дисперсії перетворених даних через шар субдискретизації можна розрахувати та об'єднати значення специфічних ознак у вхідному шарі, що може зменшити кількість нейронів, зберігаючи ознаки незмінними. Рівняння шару об'єднання (2.6) є таким:

$$y = \max(x_i), \; x_i \in x \quad 2.6$$

Де x - область карти особливостей і є виходом нейрона в цій області.

Повністю зв’язаний шар з’єднує всі нейрони попереднього згорткового шару з поточним шаром і перетворює всі локальні ознаки на глобальні. Мережа містить три повністю з'єднаних шари в кінці нейронної мережі. Повністю з'єднані шари схильні до проблем із переобладнанням. Щоб подолати цю проблему, ми використовуємо функцію відсіву, щоб зменшити переобладнання перших двох шарів. Останнім повністю зв’язаним шаром є вихідний шар, і ми

замінюємо його вихідним шаром, що містить два нейрони, що вказує на ймовірність того, що вихідним результатом є дим або некуріння. Імовірність визначається функцією softmax (2.7) наступним чином:

$$y_j = \frac{\exp(f_j)}{\sum_{i=1}^{2} \exp(f_i)},\ j = 1,2, \qquad 2.7$$

Де - вихідна ймовірністьго нейрона. $y_j j$

CNN передбачає обчислення згортки та структуру глибини, яка є репрезентативною та загальновживаною технікою в області глибокого навчання. Зокрема, CNN використовує дизайн багатошарового сприйняття, що вимагає мінімальної попередньої обробки. Основна структура CNN складається з вхідних і вихідних шарів та безлічі прихованих шарів, які включають згортку, об'єднання та повний шар з'єднання. Порівняно з іншими алгоритмами класифікації, CNN використовує відносно менше попередньої обробки і не залежить від дизайну об’єктів, що містить попередні знання, що є його головними перевагами.

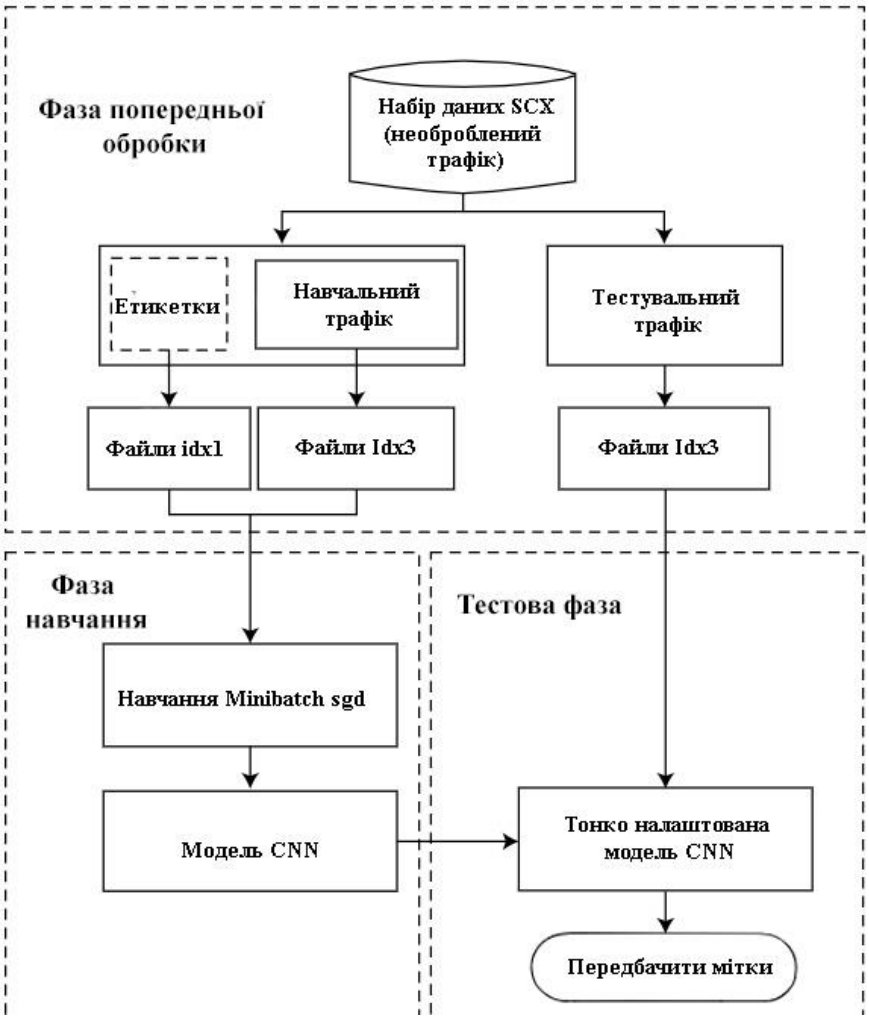
Вимовна нейронна мережа застосовуються до сфери безпеки мережі з багатообіцяючим прогресом.[28] Нейронна мережа із згортковими та рекурсивними мережевими рівнями, отримує класифікаційні ознаки для моделювання системи виявлення шкідливих програм. За допомогою методу отримують ієрархічну архітектуру вилучення ознак, яка поєднує переваги операції згортки із згорткового шару та моделювання послідовностей із рекурсивного мережевого рівня. Надалі розроблятимуть[29] для використання з функцією, отриманою із заголовків портативних виконуваних файлів, яка досягає надзвичайно точної та швидкості відкликання у випадках злиття даних.

Щоб заздалегідь виявити індикатори атак, [30] використовують нейронну мережу eXpose, де мережа бере вихідні короткі рядки як вхідні дані та витягує функції для класифікації за допомогою вбудовувань на рівні символів. Зазначається, що вихідні і вхідні дані є широким і складним діапазоном для роботи з алгоритмами. Завдяки самовидобутому дизайну об’єктів, *eXpose* перевершує базові методи, засновані на ручному вилученні об’єктів. Однак досягається зниження частоти помилкових тривог порівняно з цими базовими

лініями, що доводить, що процес автоматичного вилучення функцій у CNN є недостатньо надійним та обгрунтованим із введенням додаткової або навіть шумової інформації з вихідних входів.

Виявлення шкідливої веб-оболонки є важливим засобом захисту мережевої безпеки. З метою аналізу запитів *HTTP*,[31] використовують *word2vec*, що представляє підхід виявлення зловмисних програм на основі CNN, який є першою спробою поєднати "*word2vec*" та CNN у домені виявлення зловмисних програм. Зокрема, вони спочатку представляють інструмент "*word2vec*" для представлення кожного слова, отриманого з *HTTP*, за ознаками. Потім представляють веб-запит як матрицю фіксованого розміру шляхом об'єднання об'єктів. Нарешті, будують модель класифікації оболонки на основі структури CNN. Проводиться кілька груп експериментів, і запропонований метод найкраще працює при порівнянні з відповідними класичними класифікаторами.

Для досягнення надійної продуктивності при виявленні атак зі структурою CNN. Застосовують наскрізний метод класифікації трафіку, заснований на одновимірному CNN. [32], в якому вилучення, відбір та класифікатор ознак інтегровані у наскрізну структуру. Детальний дизайн мережі на (малюнок 2.9), де запропонований ID-CNN як алгоритм навчання, який безпосередньо вивчає взаємозв'язок між автоматично витягнутими функціями та вихідними даними із передбачуваними мітками на етапі навчання. Що стосується даних мережевого трафіку як двовимірного зображення, то новий підхід до аналізу трафіку, заснований на CNN.



2.9 Робочий процес підходу до аналізу трафіку, який складається з трьох частин: попередньої обробки, тренінгу та фази тестування.

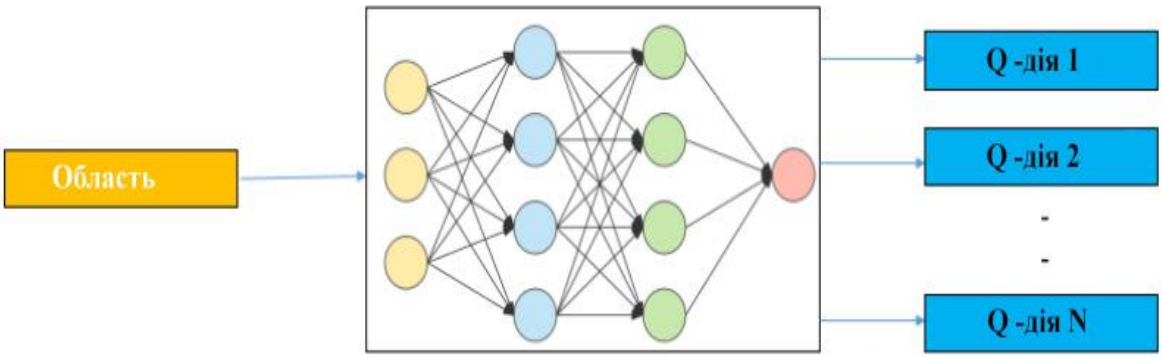
Для вирішення атаки різноманітності трафіку бездротової мережі та поліпшення здатності виявлення зловмисного вторгнення в бездротову мережу застосовують метод виявлення вторгнень, заснований на вдосконаленій згортковій нейронній мережі [33], а саме - виявлення вторгнень бездротової мережі на основі *ICNN*. Модель яка обробляє дані мережевого трафіку, а потім змоделюйте дані за допомогою CNN, що абстрагує дані про низькорівневий трафік на високорівневі функції, автоматично витягує зразкові функції та оптимізує параметри мережі за допомогою алгоритму випадкового градієнтного

Атаки відмови в обслуговуванні (LDOS) знижують продуктивність мережевих служб, і важко відрізнити поведінку атаки від звичайного трафіку. Таким чином, новий метод виявлення атаки *LDOS*, заснований на

багатофункціональному злитті та згортковій нейронній мережі (CNN). [34]. Особливості об'єднують їх у карту об'єктів, щоб описати стан мережі. Модель CNN використовується для розрізнення та виявлення карт функцій, включаючи атаки LDOS. Експерименти проводяться на імітаційній платформі *NS2* та випробувальному стенді, і результати показують, що запропонований метод може ефективно виявляти атаки LDOS.

Глибока Q-мережа

DQN - це метод, що поєднує глибоке навчання та Q-навчання, який досяг успіху в обробці середовищ із введенням високих розмірів(малюнок 2.10). Це багатошарова нейронна мережа, яка дає прогнозовану майбутню винагороду $yQ(s,a|\theta)$ для кожної можливої дії, де θ знаходяться параметри мережі. Іншими словами, DQN використовує нейронну мережу як наближення значення дії.



Глибоке Q - навчання

2.10 Структура роботи методу Q-мережі

У DQN останні чотири кадри спостережень безпосередньо надходять на CNN як перший рівень DQN для обчислення інформації про поточний стан. Потім інформація про стан відображається у вектор значень дії для поточного стану через повний рівень зв'язку. DQN оптимізує функцію значення дії шляхом оновлення ваг мережі, θ щоб мінімізувати диференційовану функцію втрат:

$$L(\theta) = (r + ymaxQ_{a_{i+1}}(s_{t+1}a_{t+1}|\theta) - Q(s_t a_t|\theta))^2 \qquad 2.8$$

Методи виявлення атак, що базуються на нейронній мережі

Оскільки вихідні дані DNN та CNN враховують лише вплив поточного введення, не враховуючи інформацію попереднього та майбутнього часу, вони можуть досягти значних показників щодо завдань класифікації або розпізнавання без змінних характеристик часу. Дані, які залежать від часу, RNN пропонується як спеціальна категорія нейромережевих структур, яка розроблена з функцією пам’ять. Насправді така конструктивна особливість збігається з думкою, що людське пізнання базується на минулому досвіді та пам’яті. Таким чином, RNN добре справляється з інформацією про часові ряди. Однак все ще існують деякі проблеми в структурі конструкції *RNN*, такі як зникнення градієнта або вибух градієнта, що призводить до відмови запам'ятовувати або моделювати тривалу залежність.

Один з варіантів розглянути характеристики часових рядів на відомій поведінці та мережевого трафіку, що може підвищити точність[35] роботи алгоритмів виявлення атак. Проектують мережу пам’яті, яка складається з 4 блоків, кожна з яких містить дві комірки. Мережа здатна підтримувати баланс як в обчислювальних витратах, так і в ефективності виявлення. Експериментальні результати вказують на те, що запропонована модель LSTM є кращою, ніж раніше опубліковані методи, оскільки LSTM може навчитися відслідковувати та співвідносити записи безперервного з'єднання в різний час.

[36] застосовування *RNN* для виконання завдання класифікації атак, де модель побудована система виявлення вторгнень, заснована на структурі RNN, на основі самоерудиції. Під час експериментів запропонована ними система виявлення вторгнень може фільтрувати атаки, але не вдається виявити помилкові спрацьовування. Порівняно з базовими методами, запропонований ними метод покращився у вимірах, як точність класифікації та трудомісткість.

Дослідження використання RNN для виявлення вторгнень з назвою RNN-IDS [37], де оцінення RNN-IDS за допомогою форм двійкової класифікації та багатокласової класифікації. Насправді модель RNN має односторонній потік інформації від перших одиниць до прихованого, а також від попереднього прихованого блоку до поточного, де приховані одиниці можуть розглядатися як

одиниці зберігання для зберігання наскрізної та корисної інформації для класифікації.

LSTM вирішує проблему довготривалої залежності та долає зникаюче падіння градієнта під час тренувань, [38] застосовуючи архітектуру LSTM для виявлення вторгнень, де розмір прихованого шару та швидкість навчання визначаються як 80 та 0,01 після експериментів. Порівняно з іншими методами, побудована модель LSTM має вищу частоту виявлення помилок під час. Слідуючи тенденції застосування LSTM для виявлення нападів, [39] побудування класифікатора LSTM для виявлення вторгнень. Вони знаходять найбільш підходящий оптимізатор для оптимізації градієнтного спуску LSTM, де вони порівнюють шість широко використовуваних методів оптимізації, тобто *Adagrad, Adadelta, RMSprop, Adam, Adamax* та *Nadam*, і знайти найбільш ефективним є *LSTM* з Оптимізатор *Nadam*.

Для зменшення високої частоти помилкових тривог, застосовується метод аналізу системних викликів.[40], яка розроблена для системи виявлення вторгнень хоста на основі аномалій. Як показано на (малюнок 2.11), їх метод складається з двох модулів: інтерфейсного модуля, тобто моделей системних викликів, який використовується для моделювання змінних у часі характеристик системних викликів зі структурою LSTM в різних середовищах, і кінцевого модуля, який використовується для прогнозування винятків на основі інформації, що передається від інтерфейсного модуля набором ансамблевих та порогових класифікаторів.

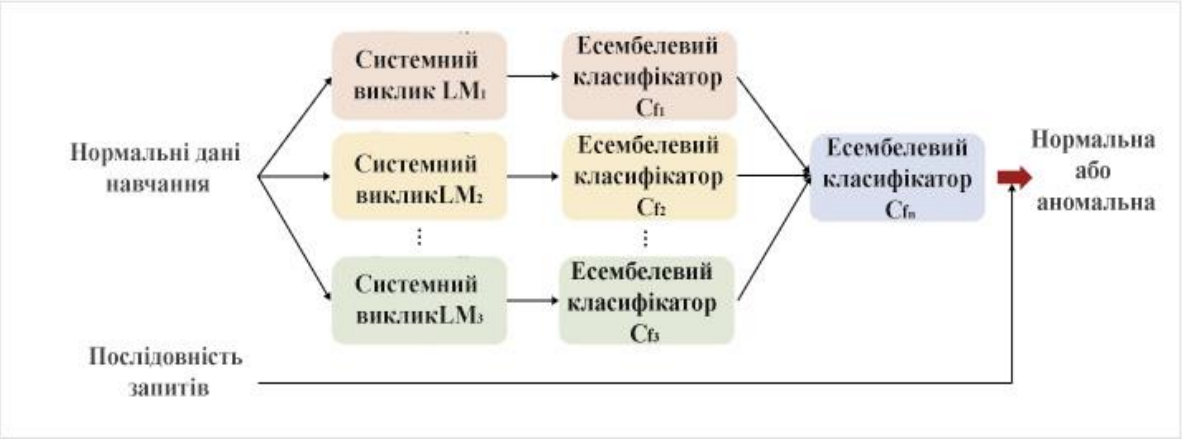


Рис 2.11 Структурна конструкція для системи виявлення вторгнень

GRU - це варіант , в якому функція *softmax* використовується як кінцевий вихідний рівень. Більше того, GRU використовує функцію перехресної ентропії для обчислення своїх втрат. На основі структури GRU, *Agarap* [41] пропонує нову мережу для двійкової класифікації в полі виявлення атак, яка розглядає 21 функцію як вхідні дані моделі. Зокрема, лінійна машина підтримки векторних ліній (*SVM*) представлена для заміни функції *softmax* запропонованої моделі GRU, яка може досягти порівняно кращих ефектів, ніж традиційна мережа GRU-*softmax*, на загальнодоступних наборах даних завдяки швидкій конвергенції та кращій можливості класифікації.

2.3 Опис даних мережевих атак

Для навчання системи виявлення мережевих атак використовується найпоширеніший набір даних NSL-KDD. Покращений набір даних, який створений для зменшення недоліків набору даних *KDDCup* 99. Зокрема, він не тільки видаляє зайві дані з навчальних та тестових даних для досягнення більш точної швидкості виявлення, але також офіційно встановлює кількість записів як у навчальних, так і в тестових даних. Більше того, група різних рівнів складності має різну кількість записів, яка обернена пропорційна відсотку від кількості у первинному наборі даних KDD. Тому дозволяє моделювати будь-яке вторгнення у реальному часі. Набір даних складається із списку файлів, що описано в таблиці 3.1.

Таблиця3.1

Набір даних NSL-KDD

№	Ім’я файлу	Опис
1	KDDTrain+.csv	Повний NSL-KDD набір даних для навчання у форматі csv
2	KDDTrain+.txt	Повний NSL-KDD набір даних для навчання у форматі txt
3	KDDTrain+_20Percent.csv	Вибірка 20% даних із KDDTrain+.csv
4	KDDTrain+_20Percent.TXT	Вибірка 20% даних із KDDTrain+.txt

5	KDDTest+.csv	Повний NSL-KDD набір даних для тестування у форматі csv
6	KDDTest+.txt	Повний NSL-KDD набір даних для тестування у форматі txt
7	KDDTest-21.txt	Вибірка із KDDTest+.txt, яка містить невірно класифіковані під час дослідження дані
8	Small Training Set.csv	Малий набір даних NSLKDD для навчання у форматі csv

Набір даних містить 41 параметр, з яких описано в таблиці 3.2 і будуть використовуватися для тренування нейронної мережі

Таблиця 3.2

Параметри *NSL_KDD*

№	Ім’я параметру	Опис	Тип
1	duration	Час тривалості підключення	Безперервний
2	protocol_type	Протокол транспортного рівня	Дискретний
3	service	Сервіс прикладного рівня	Дискретний
4	flag	Статус підключення	Дискретний
5	src_bytes	Взідний потік, байт	Безперервний
6	dst_bytes	Вихідний поток, байт	Безперервний
7	land	1 якщо адреса джерела і отримувача співпадають інакше 0	Дискретний
8	wrong_fragment	Число неправильних фрагментів	Безперервний
9	urgent	Число термінових пакетів	Безперервний
10	hot	Число «гарячих» індикаторів	Безперервний
11	num_failed_logins	Число невдалих спроб входу	Безперервний
12	logged_in	Успішний вхід	Дискретний
13	num_compromised	Число скомпрометованих станів	Безперервний
14	root_shell	Доступ з адміністративними повноваженнями	Дискретний
15	su_attemped	1 якщо "su root", інакше 0	Дискретний
16	num_root	Число спроб доступу з правами адміністратора	Безперервний
17	num_file_creations	Число операцій створення файлу	Безперервний
18	num_shells	Число спроб використання командного рядка	Безперервний

19	num_access_files	Число операцій з файлами контролю доступу	Безперервний
20	num_outbond_cmds	Число вихідних команд в ftp сеансі	Безперервний
21	is_host_login	1 якщо з'єднання зі списку серверів, інакше 0	Дискретний
22	is_guest_login	1 якщо з'єднання з гостьового списку, інакше 0	Дискретний
23	count	Число з'єднань на той же сервер за 2 секунди	Безперервний
24	srv_count	Число з'єднань на той же сервіс за 2 секунди	Безперервний
25	serror_rate	Відсоток з'єднань, у яких помилка з прапором SYN	Безперервний
26	srv_serror_rate	Відсоток з'єднань, у яких помилка з прапором SYN	Безперервний
27	rerror_rate	Відсоток з'єднань, у яких прапор REJ	Безперервний
28	srv_rerror_rate	Відсоток з'єднань, у яких прапор REJ	Безперервний
29	same_srv_rate	Відсоток з'єднань до тієї ж службі	Безперервний
30	diff_srv_rate	Відсоток з'єднань до різних служб	Безперервний
31	srv_diff_host_rate	Відсоток з'єднань до різних хостам	Безперервний
32	dst_host_count	Число з'єднань на той же сервер за 2 секунди	Безперервний
33	dst_host_srv_count	Число з'єднань на той же сервіс за 2 секунди	Безперервний
34	dst_host_same_srv_rate	Відсоток з'єднань до тієї ж службі	Безперервний
35	dst_host_diff_srv_rate	Відсоток з'єднань до різних служб	Безперервний
36	dst_host_same_src_port_rate	Відсоток з'єднань з однаковим портом джерела	Безперервний
37	dst_host_srv_diff_host_rate	Відсоток з'єднань до різних хостам	Безперервний
38	dst_host_serror_rate	Відсоток з'єднань, у яких помилка з прапором SYN	Безперервний
39	dst_host_srv_serror_rate	Відсоток з'єднань, у яких помилка з прапором SYN	Безперервний
40	dst_host_rerror_rate	Відсоток з'єднань, у яких прапор REJ	Безперервний
41	dst_host_srv_rerror_rate	Відсоток з'єднань, у яких прапор REJ	Безперервний

Кожен параметр належить до певного типу даних таблиця 3.3. Типи *Binary* використовуються в обробці фреймів даних.

Таблиця 3.3

Типи даних параметрів NSL-KDD

Тип даних	Атрибути
Binary	protocol_type, service, flag
Numeric	land, logged_in, root_shell, su_attempted, is_host_login, is_guest_login
Nominal	duration, src_bytes, dst_bytes, wrong_fragment, urgent, hot, num_failed_logins, num_compromised, num_root, num_file_creations, num_shells, num_access_files, num_outbound_cmds, count, srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate

В наборі даних існує так званий 42 параметр, який містить інформацію про типи атак:

– DOS

– Probing

– U2R

– R2L

DoS - мережеві атаки, спрямовані на створення ситуації, коли в атакованій системі відбувається відмова в обслуговуванні. Такі атаки характеризуються генерацією великого обсягу трафіку, що призводить до перевантаження та блокування сервера. Існує шість типів

DoS - атак: спина, земля, нептун, стручок, смурф, сльоза.

Зондові атаки стосуються сканування мережевих портів на наявність конфіденційної інформації. Існує чотири типи атак Probe: ipsweep, nmap, portsweep, satan.

Атаки U2R передбачають отримання привілею локального суперкористувача (адміністратора мережі) зареєстрованим користувачем. Існує чотири типи атак *U2R*: *buffer_overflow, loadmodule, perl, rootkit.*

Атаки R2L характеризуються доступом незареєстрованого користувача з віддаленого комп'ютера. Існує вісім типів атак R2L: *ftp_write, guess_passwd, imap, multihop, phf, шпигун, warezclient, warezmaster.*

Атака є класом, кожен з яких включає в себе достатньо конкретних прикладів. Типи атак та класи приведені в таблиці 3.4. Також ці параметри створюють рядок, приклад якого зображено у додатку А.

Таблиця 3.4

Класи та приклади типів атак в NSL-KDD

Тип даних	Атрибути
DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Apache2, Udpstorm, Processtable, Worm
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint
R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps

Висновок до другого розділу

Таким чином, на сьогоднішній день виявлення мережевих атак перетворюється на серйозну проблему, яка не вирішується лише традиційними методами. У найближчому майбутньому до вирішення цієї задачі будуть залучатися тільки штучний інтелект, який має багато переваг щодо прийняття рішення і виявлення аномального трафіка. При такому підході важливу роль буде відігравати наявність керуючих груп щодо контролю рішення з боку інтелекту.

Застосування методів нейромережевих технологій дозволяє системі навчатися та пристосовуватися до аномалії у мережі, таким чином, підвищити ймовірність виявляння мережевих атак.

Досліджено способи виявлення мережевих атак за допомогою нейромережевих технологій.

Проаналізовано методи DL, та вибраний один з них для навчання нейронної мережі щодо виявлення мережевих атак.

Проаналізовано набір даних мережевих атак NSL-KDD.

РОЗДІЛ 3
РОЗРОБКА МОДЕЛІ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК

3.1 Реалізація програмного коду для зберігання, форматування набору даних

Алгоритм був написаний на *python* та протестований в блокноті *Colaboratory*. Оскільки навчання нейронної мережі залежне від обчислювальної потужності машин було розглянута пропозиції щодо користування блокнотом *Jupiter*, який розміщений в сервісі *Colab*.

Для маніпулювання даними та їхнього аналізу було інстальовано бібліотеки:

– *Tensorflow;*
– *Pandas;*
– *Numpe;*
– *Keras;*
– *Scikit-learn;*
– *Matplotlib.*

Для роботи з набором даних мережеві атаки було написано код який імпортував ці дані, приклад коду:

```
self.train_path=kwargs.get('train_path','datasets/NSL/KDDTrain+.txt')
self.test_path=kwargs.get('test_path',  datasets/NSL/KDDTest%2B.txt')
```

Після імпортування даних було проведено форматування данних та збереження їх:

```
self.df = pd.concat([self.df.drop('protocol_type', axis=1),
pd.get_dummies(self.df['protocol_type'])], axis=1)
    self.df = pd.concat([self.df.drop('service', axis=1),
    pd.get_dummies(self.df['service'])], axis=1)
      self.df = pd.concat([self.df.drop('flag', axis=1),
      pd.get_dummies(self.df['flag'])], axis=1)
```

test_df = self.df.iloc[train_indx:self.df.shape[0]]

test_df = shuffle(test_df,random_state=np.random.randint(0,100))

self.df = self.df[:train_indx]

self.df = shuffle(self.df,random_state=np.random.randint(0,100))

*test_df.to_csv(self.formated_test_path,sep=',',index=**False**)*

Та окремо створили список з існуючими атаками що створило точніше навчати нейронну мережу.

3.2 Модель нейронної мережі на основі алгоритму глибокго Q-навчання

Спираючись на аналіз методів виявлення мережевих атак для систем виявлення мережевих атак на основі нейромережевих технологій було запропоновано модель назвою DQ-RL, яка реалізує класифікатор, заснований на теорії RL. У рамках *RL*, середовище інформує агента про стан мережі, а агент реагує всередині мережі. Це дія врешті-решт спричиняє зміну стану селектора. Враховуючи цю структуру агента, який взаємодіючи з селектором, запропонована модель забезпечує такі модифікації:

- змодельоване середовище, стан якого відповідає випадковим вибіркам, взятим із набору вторгнення в мережу;
- агент реалізує класифікатор, який намагається передбачити мітку вторгнення із станів, передбачених змодельованим навколишнє середовище;
- імітоване середовище приносить результати відповідно до правильних і неправильних прогнозів агента.

Нова структура, дозволяє застосовувати добре відомий алгоритм *RL* (алгоритм *Q*-навчання) для класифікації вторгнень за допомогою набору даних попередньо записаних даних про вторгнення.

Алгоритм базується на пошуку найкращої *Q*-функції для агента, а в нашому випадку для класифікатора. Оцінка *Q*-функції значення для кожної паридії стану, це значе нння відповідає сумі винагород для даного стану,

враховуючи, що ми беремо певні дії, а потім рухатись вперед за поточною політикою. Важливим результатом є те, що *Q*-функцію можна обчислювати ітеративно за таким виразом:

$$Q(S_tA_t) \leftarrow Q(S_tA_t) + \alpha[R_{t+1} + ymaxQ(S_{t+1}A_{t+1}) - Q(S_tA_t)] \quad 3.1$$

Де St - поточний стан, At - поточна дія, $At + 1$ - це наступна дія, $Rt + 1$ - наступне значення винагороди, α – навчання ставка і γ - коефіцієнт дисконтування, який у цьому випадку встановлюється близьким до нуля, оскільки стани не корелюють між собою (вони отримуються шляхом вибірки набору даних, а не в послідовному порядку), і тому немає необхідності запам'ятовувати алгоритм. Використовували значення 1,0 та 0,001 для α та γ відповідно.

Для апроксимації *Q*-функції використовується повністю підключена нейронна мережа (*NN*). Вхідні дані для цієї *NN* є поточний стан, який відповідає характеристикам, виділеним із міченого набору даних. Вихід *NN* представляє функцію *Q*, для набору доступних дій. *Q*-функція відповідає тому, наскільки добре здійснюється певна дія сучасного стану. Описаний алгоритм глибокої мережі *QQ* (*DQN*), за яким слідує алгоритм *DQN*. Модель не застосовує вираз у (1) для оновлення функції *Q*, а використовує основі функції квадратичних втрат:

$$(R_{t+1} + ymaxQ(S_{t+1}A_{t+1}) - Q(S_tA_t))^2 \quad 3.2$$

Структура, яка випливає з вищезазначених приміщень, заснована на алгоритмі *Deep Q-Network*, забезпечує хороші результати виявлення. Однак, враховуючи незбалансований характер набору даних була змінена оригінальна структура, щоб забезпечити навколишнє середовище розумною поведінкою крім випадкової вибірки набору даних. Ця зміна дозволяє вирішити проблему незбалансованого набору даних за допомогою моделі, яка виконує динамічну та інтелектуальну передискретизацію набору даних під час навчання.

У моделі DQ-RL додатково використовується підкріплення до середовище. Обидва агенти, селектор та класифікатор агент, навчаються паралельно обидва з використанням *DQN*. Дія (вихід) агента класифікатора буде передбачення типу вторгнення для вибірки набору даних (вхід), а дія агента середовища буде класом

атаки, які будуть використані для створення нових зразків у процесі навчання. Отже, навіть коли обидва агенти мають схожу структуру, основний агент виступає в ролі класифікатора, а агент навколишнього середовища - як селектор атак, які будуть використані в наступному раунді навчання. Два агенти працюють у змагальній моделі на основі отриманих результат, запропонованих (основному) агенту, які обернені для агент навколишнього середовища. Таким чином, агент середовища спробує спрямувати генерацію зразків на збільшення помилок агентом класифікатора, змушуючи класифікатор зосередитися на найскладніших зразках.

За допомогою цієї основи нова модель генерує більш збалансовану вибірку даних, виробляючи вибірки, в яких класифікатор не працює частіше. Це може бути пов'язано з низькою частотою появи у навчальному наборі або труднощами передбачення для деяких епох. Усі позитивні результати для класифікатора буде пропорціональна негативними для селектора. Таким чином нейрона мережа дізнається, в яких класифікатор найчастіше зазнає невдач і з певною ймовірністю збільшує частоту цих зразків. Для цього використано дві різні Q-функції: функцію $(Q_{e(s,a)})$, відповідальну за оптимізацію класифікатора, та іншу Функція $(Q_{e(s,a)})$ для оптимізації селектора. Обидві функції посилаються на один і той же принцип того, наскільки добре діяти певний стан, але кількість розглянутих дій може бути різною. У цьому випадку для набору даних NSL-KDD Q-функція для оптимізація класифікатора має набір дій ($A_c \in [0-4]$), що відповідає кількості класів класифікатора. З іншого боку, Q-функція, відповідальна за оптимізацію середовища, має певний набір дій що відповідає кожному з можливих атак у наборі даних навчання і тесту

Політика, яка дотримується протягом навчання, відповідає зменшенню потрібності до епсилону, так що атака спочатку є високою та зменшеною по значенню епізоду. Ми розглядаємо епізод як навчальний цикл по всьому набору даних, в даному випадку звичайним номіналом епізоду є епоха. І агент, і оточення обирають свої дії, враховуючи свою політику. Для максимального

виявлення атак нижня межа епсилону для класифікатора політика буде низькою, тоді як нижня межа селектора політики буде встановлена як гіперпараметр.

Метод, який дотримується протягом усього тренінгу, відповідає наступній послідовності:

- функції Q на середовища та агенти класифікатора ініціалізуються до випадкового значення. Крім того, початковий s_0 обраний випадковим чином для всіх станів в наборі даних, щоб подати функцію і отримати значення дії для цього стану. Важливо пам'ятати що стан є вибіркою з набору даних;
- середовище вибирає дію aet (ярлик про вторгнення), виходячи з його політики та сучасний стан.
- середовище вибирає поточний стан випадковим чином із набору даних st, дія якого відповідає тій вибраний середовищем, позначений як S (aet) у таблиці I, де наведена пара міток ознак (st, aet)
- враховуючи вибраний стан селектор атак намагається класифікувати цей стан на основі власної політики та призначає його до дії (act).
- дія: act (ярлик вторгнення) надсилається в навколишнє середовище і порівнюється з ярликом наземної істини, якщо обидва однакові, то правильна класифікація дає позитивний результат агенту, а якщо вони різні позитивна винагорода буде для класифікатора.

Новий стан задається селектором, як у типовому алгоритмі DQN, що базується на функції значення дії, тобто надаючи наступну пару міток об'єктів $(st + 1, aet + 1)$.

З результатів отриманих значень, функція політики як класифікатор, так і селектор оновлюються відповідно до правил оновлення DQN .

Застосована функція з результатів, класифікується як нормальний результат 1/0 зі значенням +1 за позитивний результат та 0 за негативний результат. На додаток до результату 1/0 були перевірені інші функції результату, зокрема, функції перехресної ентропії та категоричний шарнір використовувались для оцінки відстані між передбачуваними діями та мітками,

пов’язуючи цю відстань із результатами (коротша відстань дорівнює більшій винагороді). Ще одна функція - надання різної винагороди відповідно до типу атаки. Нарешті було обрано найпростіший результат 1/0 завдяки кращій ефективності.

Кроки алгоритму, що відхиляються від звичайного алгоритму DQN. Ці додаткові кроки дають можливість навчити селектора із змагальною стратегією.

Алгоритм для DQ-RL:

– Ініціалізувати $Q_C(s, act)$ довільно;
– Ініціалізувати $Q_C(s, act)$ довільно;
– Повторити для кожного епізоду: Ініціалізувати значення стану s_0 =випадкова вибірка (набір даних)
– Вибрати початкову дію середовища aet використовуючи політику, похідну від $Q_e(s, act)$
– Замінити st, на випадкова вибірку ($S(aet)$), де S - це всі зразки, маркування яких aet.

Повторити (для кожного часового кроку, $t \rightarrow 0$ до N):

Вибрати дію агента act використовуючи політику, похідну від $Qc\ (st, act)$

Зробити крок RL для отримання $(rct, ret, st + 1)$:

Отримати винагороду для основного агента та селектора: rct, ret

Вибрати дію наступного середовища $aet + 1$ використовуючи політику, похідну від $Qe\ (st, aet)$ замінити $st + 1$, використовуючи: $st + 1$ = випадкова вибірка ($S\ (aet + 1)$), де $S\ (aet + 1)$) це всі зразки, мітка яких $aet + 1$.

Оновлення функції Q:

Використати градієнтний спуск на функцію втрат (3.3), задану:

$$(ret + \gamma \text{ макс } aet + 1\ Qe\ (st + 1, aet + 1) - Qe\ (st, aet))\text{^}2 \qquad 3.3$$

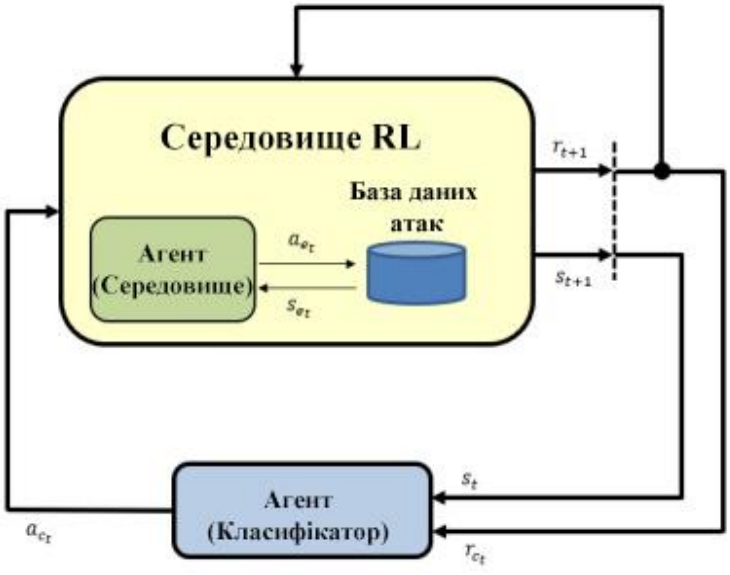
Використати градієнтний спуск на функцію втрат (3.4), задану:

$$(rct + \gamma \text{ макс } act + 1\ Qc\ (st + 1, aet + 1) - Qc\ (st, act))\text{^}2 \qquad 3.4$$

В остаточному алгоритмі, реалізованому *AE-RL*, додано три додаткові уточнення до представленого для простоти:

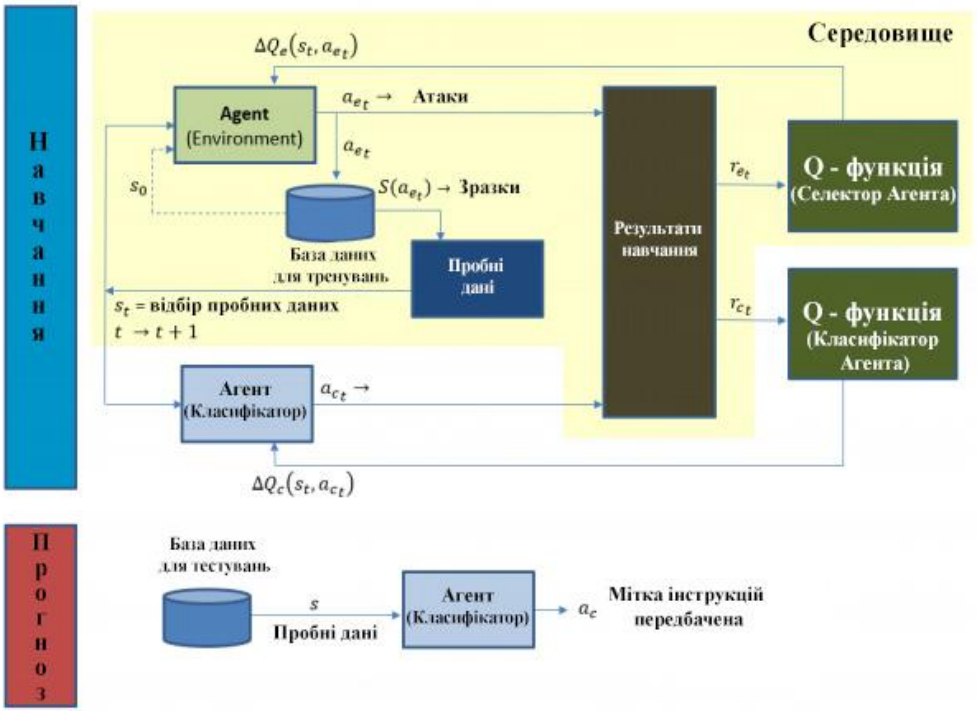
– Остаточний алгоритм, реалізований DQ-RL, базується на DDQN, який є варіантом DQN із використанням двох окремих мережі для вибору та оцінки дії.
– Фактична функція втрат, яка використовується для підготовки агентів, базується на втраті Губера, яка є подібно до квадратичних втрат до порогу та лінійних за цим значенням. Завданням втрати Губера є зменшення значень градієнти, які можуть зазнати хаотичної поведінки.
– Оскільки існує два агенти, реалізовані двома різними нейронними мережами, застосовуються дві стратегії. Обидві мережі з високим зна ченням епсилону, яке під час тренувань зводиться до фіналу значення 0,8 для мережі середовища та 0,01 для мережі класифікатора. Маючи високу цінність епсилону для навколишнього середовища (активна розвідка) є важливою для досягнення хороших результатів.

На малюнку 3.1 показуються елементи, що інтегрують алгоритм, і те, як середовище діє на псевдо агент, який працює в змагальному режимі щодо основного агента (нижня частина діаграми), що і є класифікатор.



3.1 Підсилення навчальної взаємодії між агентом та його посиленим навчанням.

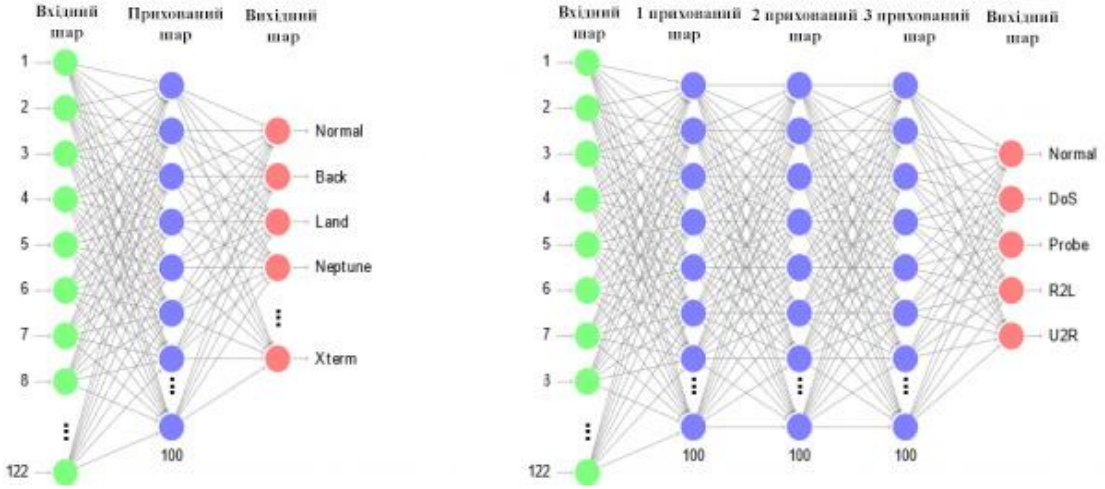
На малюнку 3.1 представлений вид на високому рівні алгоритму DQRL, детальніше подано на малюнку 3.2, який графічно зображує функціонал блоки і етапи. Також відокремлена робота алгоритму на етапах навчання та прогнозування. Верхня частина представляє етап навчання, де блоки зі злегка жовтим фоном пов'язані з функціонування навколишнього середовища. На діаграмі показано поточну операцію та s_0 являє собою початкові умови, де початковий стан береться випадковим чином із набору даних. Етап прогнозування (нижня частина) базується виключно на вже навченому класифікаторі агент. І селектор, і агенти класифікатора базуються на дуже простих нейронних мережах, як показано на малюнку 3.3.



3.2 Деталі алгоритму DRL для етапів навчання та прогнозування

На малюнку 3.3 демонструє графічну архітектуру, на якій відображено атакуючий агент (селектор) і захисний агент (класифікатор). В обох випадках архітектура складається з простої неглибокої нейронної мережі з 1 або 3 шарів та 100 одиницями на шар. Простота цієї архітектури забезпечує конкуруючий час відгуку, тоді як навчальне навчання з підкріпленням оптимально пристосовується ваги та упередження. На цьому рисунку можна зрозуміти, що і

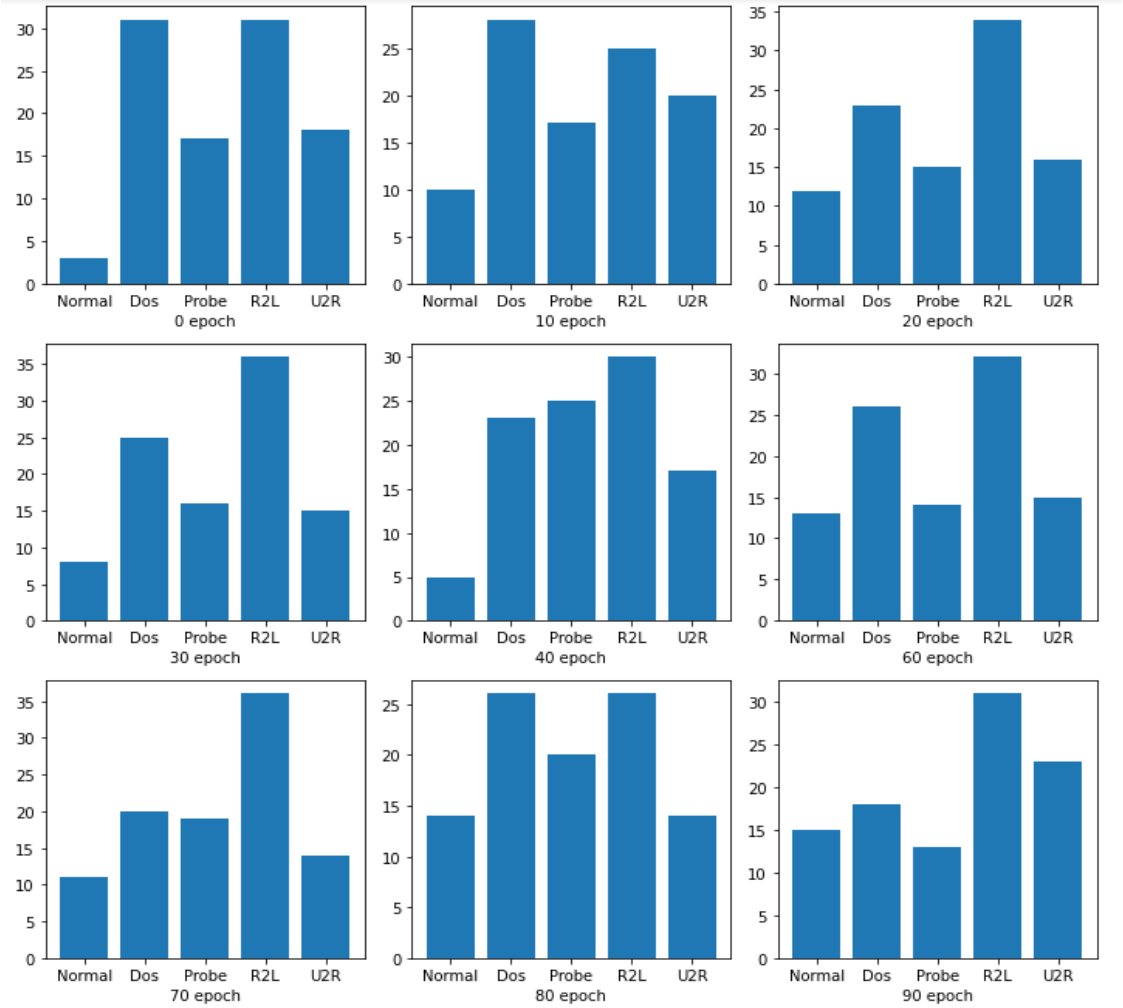
нападник, і захисник мають вибірку на вхід, але виробляє різні результати залежно від мережі. Для захисника вихід є прогнозом класа вторгнення (один із 5). Для зловмисника результат - це одна з 23 можливих атак, яка буде використана для вибору випадкової вибірки що відповідає цій вихідній атаці (одна з 23). Отже, нейронна мережа для захисного агента реалізує класифікатор, але нейронна мережа для атакуючого агента не може розглядатися як класифікатор, а як генератор класів атак.



3.3 Архітектура нейронної мережі для атакуючого агента (середовища) зліва та захисника (класифікатора) праворуч.

Малюнку 3.4 показує еволюцію розподілу (гістограми) атак, що генеруються атакуючим агентом під час тренувань у різних епохах (навчальні ітерації) для набору даних NSL-KDD. Гістограми показують лише 23 можливі атаки для набору даних тренувань. Спочатку (0 епох) середовище здійснює атаки випадковим чином. Продовжуючи навчання, селектор навчається і надсилає атаки, що максимізують його результати, що змінюється з часом. Частота різних класів атакує зміни стохастичним способом, але він демонструє тенденцію до збільшення важливості кількох атак: сатана, *ipsweep* та *warezclient*, зменшуючи при цьому важливість нормального руху. Саме таку поведінку ми хочемо спостерігати в динамічний (інтелектуальний) алгоритм, який намагається

компенсувати навчання упередженості, спричинене незбалансованим набором даних.



3.4 Еволюція розподілу атак, що генеруються атакуючим агентом (середовищем) під час навчання (набір даних *NSL-KDD*).

Порівняння частот типів атак, які присутні у наборі даних навчальної програми, з розподіл атак, породжених агентом середовища DQ-RL на малюнку 3.4 Це порівняння дає чітке уявлення погляд на те, як інтелектуальне середовище активно змінює незбалансоване розподіл зразків для поліпшення класифікації результати.

3.3 Результати ефективності моделі виявлення мережевих атак

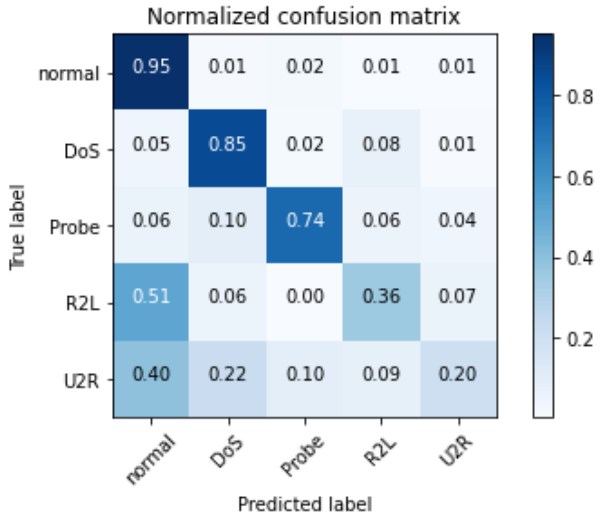
Загальний результат виявлення атак, відображено на рис 3.5. Він показує точність виявлення атак нейронної мережі, загальна кількість атак яка була проведена на мережу. Також потрібно розуміти, що результати залежать від кількості епох навчань, які були проведені з нею.

```
Total reward: 18416 | Number of samples: 22544 | Accuracy = 81.69%
       Estimated Correct Total F1_score
normal     11243    9243  9712  88.2176
DoS         6920    6348  7458  88.3016
Probe       2163    1794  2421  78.2723
R2L         1779     990  2753  43.6893
U2R          439      41   200  12.8326
```

3.5 Загальний результат виявлення атак

На малюнку 3.6 відображено показники ефективності даних тесту.

```
Performance measures on Test data
Accuracy =  0.8169
F1 =  0.8107
Precision_score =  0.8155
recall_score =  0.8169
Normalized confusion matrix
[[0.95 0.01 0.02 0.01 0.01]
 [0.05 0.85 0.02 0.08 0.01]
 [0.06 0.1  0.74 0.06 0.04]
 [0.51 0.06 0.   0.36 0.07]
 [0.4  0.22 0.1  0.09 0.2 ]]
<Figure size 432x288 with 0 Axes>
```



3.6 Ефективності даних тесту

Крім того, прогнозування та час тренувань є важливими для IDS, оскільки трафік постійно змінюється. метрика прогнозування показана на малюнок 3.7.

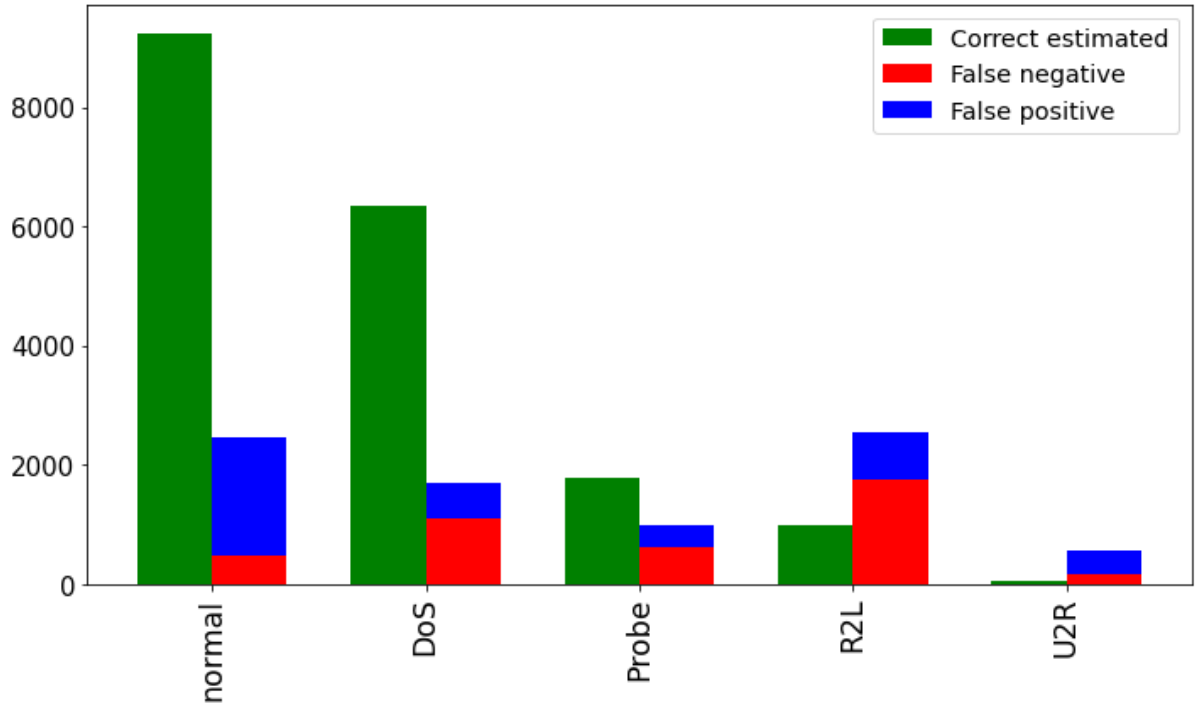
Модель забезпечує відмінні результати, незначно збільшуючи кількість помилкових спрацьовувань та одночасно значно зменшуючи помилкові негативні наслідки є кращим у сценарії IDS. Показано високі значення метрик F1 (> 0,88) та точності (> 0,97) для міток, які не є надзвичайно незбалансованими.

One vs All metrics:

	name	acc	f1	pre	rec
0	normal	0.890481	0.882176	0.822112	0.951709
1	DoS	0.92539	0.883016	0.917341	0.851167
2	R2L	0.886799	0.436893	0.556492	0.359608
3	Probe	0.95582	0.782723	0.829404	0.741016
4	U2R	0.975293	0.128326	0.0933941	0.205

3.7 Значення метрик точності виявлення категорій атак

На малюнок 3.8 відображаються результити правильності виявлення мережевих атак у графічному діапазоні. Зеленим відображено правильність оцінки виявлення атак, червноим – помилкові спрацювання, а синім – позитивні спрацювання.



3.8 Результати правильності нейронної мережі виявлення мережевих атак.

Відмінні результати щодо оцінки правильності та помилкових виявлень атак на основі нейронної мережі. Для покращення виявлення атак нейронною мережею потрібно змінити кількість епох навчання, адже навчання тривало зі

100 епох. Тому випливає, що з збільшенням кількості епох навчання нейронної мережі за *Q*-алгоритмом, то якість і точність буде збільшуватися.

Висновок до третього розділу

В даному розділі було запропоновано модель нейронної мережі на основі методу DQN з підкріпленням.

Описано основні дії щодо навчання нейронної мережі. Вхідні дані були отримані з набору KDD, який містить оброблені відомості у вигляді масивів з 42 ключових значень.

Для тестування нейронної мережі було створено матрицю неточності, яка порівнювала початкові дані та на виході і відображається у графіках.

Даний метод DL показав високі результати щодо точності за 100 епох навчання та яка, у подальшому, може використовуватися у системах виявлення мережевих атак. Для покращення виявлення атак нейронною мережею потрібно змінити ітерацію епох навчання, адже навчання тривало зі 100 епох і зі. Тому випливає, що із збільшенням кількості епох навчання нейронної мережі за -алгоритмом, то якість і точність буде збільшуватися. DQ

ВИСНОВКИ

У даній кваліфікаційній роботі було розглянуто методи виявлення аномалій на основі нейромережевих технологій для використання у системах виявлення мережевих атак.

У результаті виконання кваліфікаційної роботи випливають наступні висновки:

по-перше проаналізовано, які існують мережеві атаки за характером впливу, метою впливу, початку здійснення, наявністю зворотного зв’язку, розташування атакуючого та за рівнем моделі OSI, на якому здійснюється вплив.

по-друге, в роботі визначено основні нейромережеві технології, зроблено аналіз існуючих методів глибокого навчання та детально описаний набір даних NSL-KDD.

по-третє, реалізовано модель нейронної мережі яка виявляє мережеві атаки та яка може бути впроваджена в системи виявлення мережевих атак.

У цілому, мета кваліфікаційної роботи досягнута, а отримані результати можуть бути використанні для впровадження систем виявлення атак як у мережах спеціального призначення, так і в цивільних мережах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Види мережевих атак. Спроби їх виявлення [Електронний ресурс] // Wikipedia – Режим доступу до ресурсу: *https://holodoks.blo gspot.co m/ 2017/12/blog-post.html*.

2 Мережеві атаки [Електронний ресурс] // Wikipedia – Режим доступу до ресурсу: *https://uk.wikipedia.org/wiki/%D0%92% D1%96%D0 %B4%D0%B4%D0%B0%D0%BB%D0%B5%D0%BD%D1%96_%D0%BC%D0%B 5%D1%80%D0%B5%D0%B6%D0%B5%D0%B2%D1%96_%D0%B0%D1%82%D 0%B0%D0%BA%D0%B8*.

3 Мережева модель OSI [Електронний ресурс] // Wikiwand – Режим доступу до ресурсу: *https://www.wikiwand.com/uk/%D0%9 C%D0%B 5%D1%80%D0%B5%D0%B6%D0%B5%D0%B2%D0%B0_%D0%BC%D0%BE% D0%B4%D0%B5%D0%BB%D1%8C_OSI*.

4 Ван, і Вуаїн, “Спільна оптимізація розвантажувальної утиліти та конфіденційності для крайових обчислень увімкнено *iot*,”*IEEE Internet of Things Journal*, ст. 7, ст. 4, p. 2622–2629, 2020.

5 *R. Vinayakumar, K. Soman, and P. Poornachandran, “Evaluating effectiveness of shallow and deep networks to intrusion detection system,” in Proceedings of* 2017 *International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp.* 1282–1289, *IEEE, Udupi, India, September* 2017.

6 Штучний інтелект [Електронний ресурс] // *Wikipedia* – Режим доступу до ресурсу: *https://uk.wikipedia.org/wiki/%D0%A8%D 1%82%D 1%83%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0% B5%D0%BB%D0%B5%D0%BA%D1%82*.

7 Стівен Норткатт, Джуді Новак. Виявлення вторгнень в мережу. Настільна книга фахівця з системного аналізу. – М., «Лорі», 2001. – 384 с.

8 Штучний інтелект [Електронний ресурс] // *Wikipedia* – Режим доступу до ресурсу: *https://uk.wikipedia.org/wiki/%D0%A8 %D1%82%D1%83%D1 5%D0%BB%D0%B5%D0%BA%D1%82*.

9 *Machine Learning* [Електронний ресурс] // *ITinterprise* – Режим доступу до ресурсу: *https://www.it.ua/knowledge-base/technolog y-innovation/machi ne-learning*.

10 Штучний нейрон [Електронний ресурс] // *Wikipedia* – Режим доступу до ресурсу: *https://uk.wikipedia.org/wiki/%D0%A8% D1%82%D1%83%D1%87%D0%BD%D0%B8%D0%B9_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD*.

11 Ю. Ю., Дж. Лонг та З. Цай, “Виявлення вторгнення в мережу шляхом укладання розширених згорткових автокодерів”, Мережі безпеки та зв'язку, вип. 2017, Ідентифікатор статті 4184196, 10 сторінок, 2017.

12 Н. Шон, Т. Н. Нгок, В. Д. Пхай та К. Ши, “Підхід глибокого навчання до виявлення вторгнень у мережу”, Транзакції *IEEE* щодо нових тем в обчислювальному інтелекті, вип. 2, № 1, с. 41–50, 2018.

13 Ф. Фарахнакян та Дж. Хейкконен, “Підхід на основі глибокого автоматичного кодування для системи виявлення вторгнень”, у матеріалах 20-ї Міжнародної конференції з передових комунікаційних технологій (*ICACT*) 2018 р., С. 178–183, *IEEE*, Чанчхон, Південна Корея, Липень 2018 р.

14 А. Джавайд, К. Ніяз, В. Сун та М. Алам, “Підхід глибокого навчання для системи виявлення вторгнень у мережу”, у матеріалах 9-ї Міжнародної конференції *EAI* з питань інформаційно-комунікаційних технологій, натхненних біологією (раніше *BIONETICS*), С. 21–26, Нью-Йорк, Нью-Йорк, США, грудень 2016 року.

15 Д. Папамарціванос, Ф. Гомес Мармоль і Г. Камбуракіс, Введення в які самостійно системи виявлення вторгнень в мережу з глибоким навчанням, *IEEE Access*, Пискатауей, Ню-Джерсі, США, 2019.

16 Ю. Дінг, С. Чень та Дж. Сю, “Застосування мереж глибоких вірувань для виявлення зловмисного програмного забезпечення на основі операційних

кодів”, у матеріалах Міжнародної спільної конференції з нейронних мереж 2016 року (*IJCNN*), с. 3901–3908, Ванкувер, Британія, Липень 2016 р.

17 Тан, В. Хуанг та К. Лі, «Метод виявлення вторгнень, заснований на *dbn* в спеціальних мережах», у «Праці бездротового зв'язку та сенсорної мережі»: Міжнародна конференція з бездротового зв'язку та сенсорної мережі (*WCSN*, с. 477–485 ,*World Scientific*, Ухань, Китай, грудень 2015 року.

18 М. З. Алом, В. Бонтупаллі та Т. М. Таха, “Виявлення вторгнень за допомогою мереж глибоких вірувань”, у Збірнику матеріалів Національної аерокосмічної та електронічної конференції 2015 року (*NAECON*), с. 339–344, Дейтон, Огайо, США, червень 2015 р.

19 Г. Чжао, Чжан Ч. та Л. Чжен, “Виявлення вторгнень за допомогою мережі глибоких вірувань та ймовірнісної нейронної мережі”, у матеріалах Міжнародної конференції *IEEE* з обчислювальної науки та техніки (*CSE*) 2017 року та Міжнародної конференції IEEE з питань вбудованих та повсюдних обчислень (*EUC*), Тайбей, Тайвань, грудень 2017 року.

20 К. Алравашде і К. Перді, “На шляху до онлайнової системи виявлення вторгнень, заснованої на глибокому навчанні”, у матеріалах 15-ї Міжнародної конференції IEEE з машинного навчання та програм (*ICMLA*), с. 195–200, Анахайм, Каліфорнія, США , Грудень 2016 року.

21 Шонк, Ю., Шангайн, і Ті. Хуанг, "Виявлення повсюдної мережевої атаки в режимі реального часу на основі мережі глибоких вірувань та векторної машини підтримки", *IEEE / CAA Journal of Automatica Sinica , v*ol. .7, №3, с. 790–799, 2020.

22 Т. Ерпек, Ю. Є. Сагдую та Ю. Ши, “Поглиблене навчання для запуску та пом’якшення бездротових атак глушіння”,*IEEE Transaction on Cognitive Communications and Networking, vol*. 5, №1, с.2–14, 2018.

23 А. Ал Еруд і Г. Карабатіс, «В обхід виявлення фішингових атак на основі *url* за допомогою генеративних суперечливих глибоких нейронних мереж», у Матеріалах шостого міжнародного семінару з аналітики безпеки та

конфіденційності, с. 53–60, Новий Орлеан, Лос-Анджелес, США , Березень 2020 р.

24 Та Танг, Лерон, Сар Заідай та Май. Хоко, “Підхід глибокого навчання для виявлення вторгнень мережі в мережу, що визначається програмним забезпеченням”, у матеріалах Міжнародної конференції з бездротових мереж та мобільних комунікацій 2016 (*WINCOM*), с. . 258–263, *IEEE*, Реймс, Франція, жовтень 2016 р.

25 Лі Барал, Хай. Ван, “*Hashtran-dnn*: основа для підвищення стійкості глибоких нейронних мереж проти зразків шкідливого програмного забезпечення”, 2018, *http: / /arxiv.org/abs/ 1809.06498*.

26 Штучний інтелект [Електронний ресурс] // *Wikipedia* – Режим доступу до ресурсу: *https://uk.wikipedia.org/wiki/%D0 %A8%D1%82%D1%83%D1%87%D0%BD%D0%B8%D0%B9_%D1%96%D0%BD%D1%82%D0%B5%D0%BB%D0%B5%D0%BA%D1%82*.

27 В. Пен, Х. Конг, Г. Пен, Х. Лі та З. Ван, "Виявлення вторгнення в мережу на основі глибокого навчання", у матеріалах Міжнародної конференції з комунікацій, інформаційних систем та обчислювальної техніки (*CISCE*) 2019р. . 431–435, Хайкоу, Китай, липень 2019.

28 Б. Колошняджі, А. Заррас, Г. Вебстер та К. Еккерт, “Глибоке навчання для класифікації послідовностей системних викликів шкідливих програм”, у Матеріалах Австралійської спільної конференції зі штучного інтелекту, с. 137–149, Спрингер, Хобарт, Австралія , Грудень 2016 р.

29 Б. Колошняджі, Г. Ерайша, Г. Вебстер, А. Заррас та К. Еккерт, “Розширення можливостей згорткових мереж для класифікації та аналізу шкідливих програм”, у матеріалах Міжнародної спільної конференції з нейронних мереж (*IJCNN*) 2017 р., С. 3838– 3845, Сан-Дієго, Каліфорнія, США, червень 2017 року.

30 Дж. Сакс і К. Берлін, «викриття: згорткова нейронна мережа на рівні символів із вбудованими засобами для виявлення шкідливих *URL*-адрес, шляхів до файлів та ключів реєстру», 2017, *http://arxiv.org/abs/ 1702.08568*

31 М. Чжан, Б. Сю, С. Бай, С. Лу та З. Лін, “Метод глибокого навчання для виявлення веб-атак за допомогою спеціально розробленого CNN”, у матеріалах 24-ї Міжнародної конференції з обробки нейронної інформації, с. 828–836, Гуанчжоу, Китай, листопад 2017 р.

32 В. Ванг, М. Цуй, Дж. Вонг, Х. Зенг і З. Янг, "Наскрізна зашифрована класифікація трафіку з одновимірною нейромережею згортки", у матеріалах Міжнародної конференції *IEEE* з питань розвідки та безпеки 2017 року Інформатика (*ISI*), стор. 43–48, Тайбей, Тайвань, червень 2017 р.

33 Х. Янг та Ф. Ван, “Виявлення вторгнень у бездротову мережу на основі вдосконаленої згорткової нейронної мережі”, *IEEE* Access, вип. 7, с. 64366–64374, 2019.

34 Д. Танг, Л. Танг, В. Ши, С. Чжан та К. Ян, *Mf-cnn*: Новий підхід для виявлення атак *Ldos* на основі багатофункціонального злиття та *Cnn*. Мобільні мережі та додатки, Springer, Берлін, Німеччина, 2020.

35 Р.С. Стаудмейєр, “Застосування довготривалих короткочасних нейромереж у пам’яті для виявлення вторгнень”, *South African Computer Journal, vol.* 56, ні. 1, с. 136–154, 2015.

36 Р. Б. Кришнан та Н. Рааджан, “Модель системи інтелектуального виявлення вторгнень для класифікації атак з використанням *rnn*”, Міжнародний журнал фармацевтичних технологій та біотехнологій, вип. 8, №4, с. 23157–23164, 2016.

37 С. Юн, “Підхід глибокого навчання для виявлення вторгнень із використанням періодичних нейронних мереж”, *IEEE Access, vol*. 5, с. 21954–21961, 2017

38 Дж. Кім, Дж. Кім, HLT Чт, і Х. Кім, “Класифікатор довготривалої короткочасної пам'яті нейромережевої мережі для виявлення вторгнень”, у матеріалах Міжнародної конференції з технологій та послуг платформ 2016 (*PlatCon*), с. , Чеджу, Корея, лютий 2016 р.

39 Т. Ле, Дж. Кім та Х. Кім, “Ефективний класифікатор виявлення вторгнень, що використовує тривалу короткочасну пам’ять з оптимізацією

градієнтного спуску", у матеріалах Міжнародної конференції з технологій та сервісу платформ 2017 р, 6, Чеджу, Корея, лютий 2017 р.

40 Г. Кім, Х. І, Дж. Лі, Ю. Паек та С. Юн, "Моделювання мови системних викликів на основі Lstm та надійний метод ансамблю для проектування систем виявлення вторгнень на основі хоста", 2016,*http: // arxiv.org/abs/ 1611.01726*

41 *AFM Agarap*, "Архітектура нейронної мережі, що поєднує керований рекурентний блок (*gru*) та машину з підтримкою векторів (*svm*) для виявлення вторгнень у дані мережевого трафіку", у матеріалах 10-ї Міжнародної конференції з машинного навчання та обчислень, *ACM*, Макао, Китай , Лютий 2018 р.

ДОДАТОК А. ПРИКЛАД ДАНИХ NSL KDD

@relation 'KDDTrain-20Percent'
@attribute 'protocol_type' {'tcp','udp', 'icmp'}
@attribute 'service' {'aol', 'auth', 'bgp', 'courier', 'csnet_ns', 'ctf', 'daytime', 'discard', 'domain', 'domain_u', 'echo', 'eco_i', 'ecr_i', 'efs', 'exec', 'finger', 'ftp', 'ftp_data', 'gopher', 'harvest', 'hostnames', 'http', 'http_2784', 'http_443', 'http_8001', 'imap4', 'IRC', 'iso_tsap', 'klogin', 'kshell', 'ldap', 'link', 'login', 'mtp', 'name', 'netbios_dgm', 'netbios_ns', 'netbios_ssn', 'netstat', 'nnsp', 'nntp', 'ntp_u', 'other', 'pm_dump', 'pop_2', 'pop_3', 'printer', 'private', 'red_i', 'remote_job', 'rje', 'shell', 'smtp', 'sql_net', 'ssh', 'sunrpc', 'supdup', 'systat', 'telnet', 'tftp_u', 'tim_i', 'time', 'urh_i', 'urp_i', 'uucp', 'uucp_path', 'vmnet', 'whois', 'X11', 'Z39_50'}
@attribute 'flag' { 'OTH', 'REJ', 'RSTO', 'RSTOS0', 'RSTR', 'S0', 'S1', 'S2', 'S3', 'SF', 'SH'}
@attribute 'src_bytes' real
@attribute 'dst_bytes' real
@attribute 'land' {'0', '1'}
@attribute 'wrong_fragment' real
@attribute 'urgent' real
@attribute 'hot' real
@attribute 'num_failed_logins' real
@attribute 'logged_in' {'0', '1'}
@attribute 'num_compromised' real
@attribute 'root_shell' real
@attribute 'su_attempted' real
@attribute 'num_root' real
@attribute 'num_file_creations' real
@attribute 'num_shells' real
@attribute 'num_access_files' real
@attribute 'num_outbound_cmds' real
@attribute 'is_host_login' {'0', '1'}
@attribute 'is_guest_login' {'0', '1'}
@attribute 'count' real
@attribute 'srv_count' real
@attribute 'serror_rate' real
@attribute 'srv_serror_rate' real
@attribute 'rerror_rate' real
@attribute 'srv_rerror_rate' real
@attribute 'same_srv_rate' real
@attribute 'diff_srv_rate' real
@attribute 'srv_diff_host_rate' real
@attribute 'dst_host_count' real
@attribute 'dst_host_srv_count' real
@attribute 'dst_host_same_srv_rate' real
@attribute 'dst_host_diff_srv_rate' real
@attribute 'dst_host_same_src_port_rate' real
@attribute 'dst_host_srv_diff_host_rate' real
@attribute 'dst_host_serror_rate' real
@attribute 'dst_host_srv_serror_rate' real
@attribute 'dst_host_rerror_rate' real
@attribute 'dst_host_srv_rerror_rate' real
@attribute 'class' {'normal', 'anomaly'}
@data

0,tcp,ftp_data,SF,491,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,150,25,0.17,0.03,0.17,0.00,0.00,0.00,0.05,0.00,normal

0,udp,other,SF,146,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,13,1,0.00,0.00,0.00,0.00,0.08,0.15,0.00,255,1,0.00,0.60,0.88,0.00,0.00,0.00,0.00,0.00,normal

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,123,6,1.00,1.00,0.00,0.00,0.05,0.07,0.00,255,26,0.10,0.05,0.00,0.00,1.00,1.00,0.00,0.00,anomaly

0,tcp,http,SF,232,8153,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,5,5,0.20,0.20,0.00,0.00,1.00,0.00,0.00,30,255,1.00,0.00,0.03,0.04,0.03,0.01,0.00,0.01,normal

0,tcp,http,SF,199,420,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,30,32,0.00,0.00,0.00,0.00,1.00,0.00,0.09,255,255,1.00,0.00,0.00,0.00,0.00,0.00,0.00,0.00,normal

0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,121,19,0.00,0.00,1.00,1.00,0.16,0.06,0.00,255,19,0.07,0.07,0.00,0.00,0.00,0.00,1.00,1.00,anomaly

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,166,9,1.00,1.00,0.00,0.00,0.05,0.06,0.00,255,9,0.04,0.05,0.00,0.00,1.00,1.00,0.00,0.00,anomaly

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,117,16,1.00,1.00,0.00,0.00,0.14,0.06,0.00,255,15,0.06,0.07,0.00,0.00,1.00,1.00,0.00,0.00,anomaly

0,tcp,remote_job,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,270,23,1.00,1.00,0.00,0.00,0.09,0.05,0.00,255,23,0.09,0.05,0.00,0.00,1.00,1.00,0.00,0.00,anomaly

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,133,8,1.00,1.00,0.00,0.00,0.06,0.06,0.00,255,13,0.05,0.06,0.00,0.00,1.00,1.00,0.00,0.00,anomaly

0,tcp,private,REJ,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,205,12,0.00,0.00,1.00,1.00,0.06,0.06,0.00,255,12,0.05,0.07,0.00,0.00,0.00,0.00,1.00,1.00,anomaly

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,199,3,1.00,1.00,0.00,0.00,0.02,0.06,0.00,255,13,0.05,0.07,0.00,0.00,1.00,1.00,0.00,0.00,anomaly

0,tcp,http,SF,287,2251,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,7,0.00,0.00,0.00,0.00,1.00,0.00,0.43,8,219,1.00,0.00,0.12,0.03,0.00,0.00,0.00,0.00,normal

0,tcp,ftp_data,SF,334,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,2,2,0.00,0.00,0.00,0.00,1.00,0.00,0.00,2,20,1.00,0.00,1.00,0.20,0.00,0.00,0.00,0.00,anomaly

0,tcp,name,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,233,1,1.00,1.00,0.00,0.00,0.00,0.06,0.00,255,1,0.00,0.07,0.00,0.00,1.00,1.00,0.00,0.00,anomaly

0,tcp,netbios_ns,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,96,16,1.00,1.00,0.00,0.00,0.17,0.05,0.00,255,2,0.01,0.06,0.00,0.00,1.00,1.00,0.00,0.00,anomaly

0,tcp,http,SF,300,13788,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,9,0.00,0.11,0.00,0.00,1.00,0.00,0.22,91,255,1.00,0.00,0.01,0.02,0.00,0.00,0.00,0.00,normal

0,icmp,eco_i,SF,18,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,1,16,1.00,0.00,1.00,1.00,0.00,0.00,0.00,0.00,anomaly

0,tcp,http,SF,233,616,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,3,3,0.00,0.00,0.00,0.00,1.00,0.00,0.00,66,255,1.00,0.00,0.02,0.03,0.00,0.00,0.02,0.00,normal

0,tcp,http,SF,343,1178,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,9,10,0.00,0.00,0.00,0.00,1.00,0.00,0.20,157,255,1.00,0.00,0.01,0.04,0.00,0.00,0.00,0.00,normal

0,tcp,mtp,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,223,23,1.00,1.00,0.00,0.00,0.10,0.05,0.00,255,23,0.09,0.05,0.00,0.00,1.00,1.00,0.00,0.00,anomaly

0,tcp,private,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,280,17,1.00,1.00,0.00,0.00,0.06,0.05,0.00,238,17,0.07,0.06,0.00,0.00,0.99,1.00,0.00,0.00,anomaly

0,tcp,http,SF,253,11905,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,8,10,0.00,0.00,0.00,0.00,1.00,0.00,0.20,87,255,1.00,0.00,0.01,0.02,0.00,0.00,0.00,0.00,normal

5607,udp,other,SF,147,105,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,0.00,0.00,0.00,0.00,1.00,0.00,0.00,255,1,0.00,0.85,1.00,0.00,0.00,0.00,0.00,0.00,normal

0,tcp,mtp,S0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,248,2,1.00,1.00,0.00,0.00,0.01,0.06,0.00,255,2,0.01,0.06,0.00,0.00,1.00,1.00,0.00,0.00,anomaly

ДОДАТОК В. ЛІСТИНГ КОДА

```
if __name__ == "__main__":
    kdd_train = "datasets/NSL/KDDTrain%2B.txt"
    kdd_test = "datasets/NSL/KDDTest%2B.txt"
    formated_train_path = "formated_train_adv.data"
    formated_test_path = "formated_test_adv.data"
    # Train batch
    batch_size = 1
    # batch of memory ExpRep
    minibatch_size = 100
    ExpRep = True
    iterations_episode = 100
    # Initialization of the enviroment
    env = RLenv('train',train_path=kdd_train,test_path=kdd_test,
                formated_train_path = formated_train_path,
                formated_test_path = formated_test_path,batch_size=batch_size,
                iterations_episode=iterations_episode)
    # obs_size = size of the state
    obs_size = env.data_shape[1]-len(env.all_attack_names)
    #num_episodes = int(env.data_shape[0]/(iterations_episode)/10)
    num_episodes = 100
    ''' Definition for the defensor agent.'''
    defender_valid_actions = list(range(len(env.attack_types))) # only detect type of attack
    defender_num_actions = len(defender_valid_actions)
    def_epsilon = 1 # exploration
    min_epsilon = 0.01 # min value for exploration
    def_gamma = 0.001
    def_decay_rate = 0.99
    def_hidden_size = 100
    def_hidden_layers = 3
    def_learning_rate = .2
    defender_agent = DefenderAgent(defender_valid_actions,obs_size,"EpsilonGreedy",
                        epoch_length = iterations_episode,
                        epsilon = def_epsilon,
                        min_epsilon = min_epsilon,
                        decay_rate = def_decay_rate,
                        gamma = def_gamma,
                        hidden_size=def_hidden_size,
                        hidden_layers=def_hidden_layers,
                        minibatch_size = minibatch_size,
                        mem_size = 1000,
                        learning_rate=def_learning_rate,
                        ExpRep=ExpRep)
    #Pretrained defender
    #defender_agent.model_network.model.load_weights("models/type_model.h5")

    '''Definition for the attacker agent.
    In this case the exploration is better to be greater
```

```
    The correlation sould be greater too so gamma bigger'"
    attack_valid_actions = list(range(len(env.attack_names)))
    attack_num_actions = len(attack_valid_actions)
    att_epsilon = 1
    min_epsilon = 0.82 # min value for exploration
    att_gamma = 0.001
    att_decay_rate = 0.99

    att_hidden_layers = 1
    att_hidden_size = 100
    att_learning_rate = 0.2
    attacker_agent = AttackAgent(attack_valid_actions,obs_size,"EpsilonGreedy",
                        epoch_length = iterations_episode,
                        epsilon = att_epsilon,
                        min_epsilon = min_epsilon,
                        decay_rate = att_decay_rate,
                        gamma = att_gamma,
                        hidden_size=att_hidden_size,
                        hidden_layers=att_hidden_layers,
                        minibatch_size = minibatch_size,
                        mem_size = 1000,
                        learning_rate=att_learning_rate,
                        ExpRep=ExpRep)
    # Statistics
    att_reward_chain = []
    def_reward_chain = []
    att_loss_chain = []
    def_loss_chain = []
    def_total_reward_chain = []
    att_total_reward_chain = []
   # Print parameters
    print("-------------------------------------------------------------------------------")
    print("Total epoch: {} | Iterations in epoch: {}"
          "| Minibatch from mem size: {} | Total Samples: {}|".format(num_episodes,
                        iterations_episode,minibatch_size,
                        num_episodes*iterations_episode))
    print("-------------------------------------------------------------------------------")
    print("Dataset shape: {}".format(env.data_shape))
    print("-------------------------------------------------------------------------------")
    print("Attacker parameters: Num_actions={} | gamma={} |"
          " epsilon={} | ANN hidden size={} | "
          "ANN hidden layers={}|".format(attack_num_actions,
                           att_gamma,att_epsilon, att_hidden_size,
                           att_hidden_layers))
    print("-------------------------------------------------------------------------------")
    print("Defense parameters: Num_actions={} | gamma={} | "
          "epsilon={} | ANN hidden size={} |"
          " ANN hidden layers={}|".format(defender_num_actions,
                           def_gamma,def_epsilon,def_hidden_size,
                           def_hidden_layers))
```

```
print("--------------------------------------------------------------------------------")

# Main loop
attacks_by_epoch = []
attack_labels_list = []
for epoch in range(num_episodes):
    start_time = time.time()
    att_loss = 0.
    def_loss = 0.
    def_total_reward_by_episode = 0
    att_total_reward_by_episode = 0
    # Reset enviromet, actualize the data batch with random state/attacks
    states = env.reset()

    # Get actions for actual states following the policy
    attack_actions = attacker_agent.act(states)
    states = env.get_states(attack_actions)
    done = False
    attacks_list = []
    # Iteration in one episode
    for i_iteration in range(iterations_episode):
        attacks_list.append(attack_actions[0])
        # apply actions, get rewards and new state
        act_time = time.time()
        defender_actions = defender_agent.act(states)
        #Enviroment actuation for this actions
        next_states,def_reward, att_reward,next_attack_actions, done = env.act(defender_actions,attack_actions)
        # If the epoch*batch_size*iterations_episode is largest than the df
        attacker_agent.learn(states,attack_actions,next_states,att_reward,done)
        defender_agent.learn(states,defender_actions,next_states,def_reward,done)
        act_end_time = time.time()
        # Train network, update loss after at least minibatch_learns
        if ExpRep and epoch*iterations_episode + i_iteration >= minibatch_size:
            def_loss += defender_agent.update_model()
            att_loss += attacker_agent.update_model()
        elif not ExpRep:
            def_loss += defender_agent.update_model()
            att_loss += attacker_agent.update_model()
        update_end_time = time.time()
        # Update the state
        states = next_states
        attack_actions = next_attack_actions
        # Update statistics
        def_total_reward_by_episode += np.sum(def_reward,dtype=np.int32)
        att_total_reward_by_episode += np.sum(att_reward,dtype=np.int32)
    attacks_by_epoch.append(attacks_list)
```

ДОДАТОК С. ПЕРЕКЛАД 1 РОЗДІЛУ

APTER 1

ANALYSIS OF THE STATE AND DIRECTIONS OF THE DEVELOPMENT OF THE NETWORK ATTACK DETECTION SYSTEM

1.1 Analysis of network attacks

A network attack is an attempt to bypass the system's security policy, giving attackers access to obtain or modify information, even destroying the system. Due to the technologies being developed in local area network and wireless network systems, there are serious threats to network security, especially system security. As we are now in the era of machine learning and big data[1], cybersecurity in wireless and LAN systems is important for users.

Nature of influence[1]:

active;

passive.

An active attack is defined as an influence on the operation of the system, namely, system malfunction, configuration change, etc. Virtually all types of network attacks are proactive attacks. A feature of such attacks is the possibility of their detection, in different cases with a greater or lesser degree, as some changes in the system occur as a result of such attacks.

A passive attack on a computer system that does not have a direct impact on the functioning of the system, but can lead to a violation of security policy. Unlike an active attack, a passive attack does not leave any traces. It is the lack of direct impact on the network that makes it almost impossible to detect a passive attack. An example of attacks with passive influence is eavesdropping on a communication channel in the network.

The purpose of the attack[2]:

system malfunction (access to the system);

distortion of information integrity;

violation of confidentiality of information.

The goal of any attack is to gain unauthorized access to information. There are two basic possibilities of access to information: interception and distortion.

Interception of information means gaining access to it, but not being able to modify it.

Interception of information leads to violation of its confidentiality. An example of interception of information can be listening to a channel on the network. In this case, there is unauthorized access to information without the possibility of its distortion. Breach of information privacy is a passive influence.

Distortion of information means either complete control over the information flow between system objects, or the ability to transmit messages on behalf of another object. Thus, distortion of information leads to violation of its integrity. It has a destructive effect, which is an example of an active effect.

Violation of system performance in this case does not involve the attacker gaining unauthorized access to information. The main goal is to ensure that the attacked system fails, and access to the resources of the attacked object would be impossible for everyone in the system. An example of a network attack, the purpose of which is to disrupt the system's performance, can be a denial-of-service attack - a DOS attack.

Condition for starting influence: [2]

Distant influence, like any other influence, can begin to be realized only under certain conditions. In computer networks, there are three types of conditions for starting a network attack:

attack on request from the attacking object;

attack upon the occurrence of an expected event on the system attack;

unconditional attack.

Influence from the attacked party will begin provided that the potential target of the attack is to transmit a request of a certain type. Such an attack can be called an attack on request from the attacking object. This type of start of implementation is most typical for the system detection mode. An example of such requests on the Internet can be DNS and ARP requests, and in Novell NetWare - a SAP request.

An attack upon the occurrence of an expected event on the attacked object. The attacker continuously monitors the state of the operating system of the remote target of the attack and begins to influence when a certain event occurs in this system. The attacking object itself is the initiator of the attack. An example of such an event would be the termination of a user's session with the server without issuing the LOGOUT command in Novell NetWare.

An unconditional attack is carried out immediately and regardless of the state of the operating system of the attacking object. Therefore, the attacker is the initiator of the attack in this case.

In case of violation of the normal functioning of the system, other goals are pursued and the attacker is not expected to gain illegal access to the data. Its purpose is to disable the operating system on the attacking object and make it impossible for other objects of the system to access the resources of this object. An example of this type of attack is a DOS attack.

Availability of feedback. [2]

with feedback;

without feedback.

A network attack carried out in the presence of feedback, which is attacked by the object, is characterized by the fact that the attacker needs to receive a response to some requests sent to the attacked object, and, therefore, there is feedback between the attacker and the target of the attack , which allows the attacker to respond adequately to all changes occurring on the object. Such network attacks are most characteristic of distributed computer networks.

Network attacks without feedback do not need to react to any changes, etc

An intrusion detection system detects malicious activity by collecting and analyzing network behavior, security logs, and other information available on the network among connected computers [5]. Essentially, an intrusion detection system checks for abnormal behavior against the system's security policy and signs of an attack in the system, which is able to protect the system with timely responses and countermeasures in real time. In traditional system settings, intrusion detection works

as a smart, active and effective complement to a firewall that actually acts as a passive defense against attacks.

The traditional intrusion detection system is originally built on exploit technology, which mainly extracts the characteristics or behavior rules of the intrusion. After the advent of abnormal behavior detection technology using traditional machine learning models, the intrusion detection system develops statistical probability modeling for normal behavior that can analyze abnormal behavior with large deviations. However, such a system may have unsatisfactory results due to low capabilities in defining the problem space and complexity in modeling malicious actions.

There is an IPS intrusion prevention system in the information world, that is, an active protection system. Like IDS, it attempts to identify potential threats based on the monitoring functions of the protected host or network and may use signature, anomaly, or hybrid detection techniques. Unlike IDS, IPS takes action to block or eliminate a detected threat. While IPS can be alarming, it also helps prevent intrusion.

IPS are divided into four categories.

The first is a network intrusion prevention system that monitors the entire network for suspicious activity.

The second type is network behavior analysis systems, which study traffic flow to detect unusual traffic flows that may be the result of an attack, such as a distributed denial of service.

The third type is wireless intrusion prevention systems, which analyzes wireless networks for suspicious traffic.

The fourth type is host-based intrusion prevention systems, where a software package is installed to monitor the activity of a single host.

As mentioned earlier, IPS takes proactive steps such as dropping packets that contain malicious data, dropping or blocking traffic coming from the offending IP address.

At the end of the day, an intrusion prevention vs. intrusion detection comparison comes down to what actions they take when an intrusion is detected. IDS is only

intended to alert you to a potential incident, allowing Security Operations Center analysts to investigate the event and determine whether it requires further action. IPS, on the other hand, takes steps to block the intrusion attempt or otherwise correct the incident.

While their answers may differ, they serve similar purposes, potentially making them redundant. Even so, they both have advantages and deployment scenarios that one is better suited to than the other:

An intrusion detection system is designed to detect a potential incident, generating an alert to prevent an incident. Although this may seem like a good solution for systems with high availability requirements, such as industrial control systems and other critical infrastructure. For

of these systems, the most important thing is that the systems continue to work, and blocking suspicious and potentially malicious traffic can affect their performance. When a human operator is notified of a problem, it allows them to assess the situation and make an informed decision about how to respond.

Intrusion Prevention System: An IPS, on the other hand, is designed to take steps to block anything it deems to be a threat to a protected system. As attacks and malware become faster and more sophisticated, this is a useful feature because there is a limited potential damage that an attack can cause. IPS is ideal for environments where any intrusion can cause significant damage, such as databases containing sensitive data.

IDS and IPS have their advantages and disadvantages. When choosing a system for a potential use case, it is important to consider the trade-offs between system availability and usability and the need for security. An IDS leaves a window for an attacker to harm the target system, while a false positive detection by an IPS can negatively impact the usability of the system.

Всього_____аркушів

Опис склав _____
(підпис)

Опис перевірив _____
(підпис)

«____» _____ 20__ року

«МЕТОДИ ТА СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК НА ОСНОВІ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ»

Виконав: студент 6 курсу групи 601дТТ спеціальності 172 **«Телекомунікації та радіотехніка»**
Микола ПРИХОДЬКО

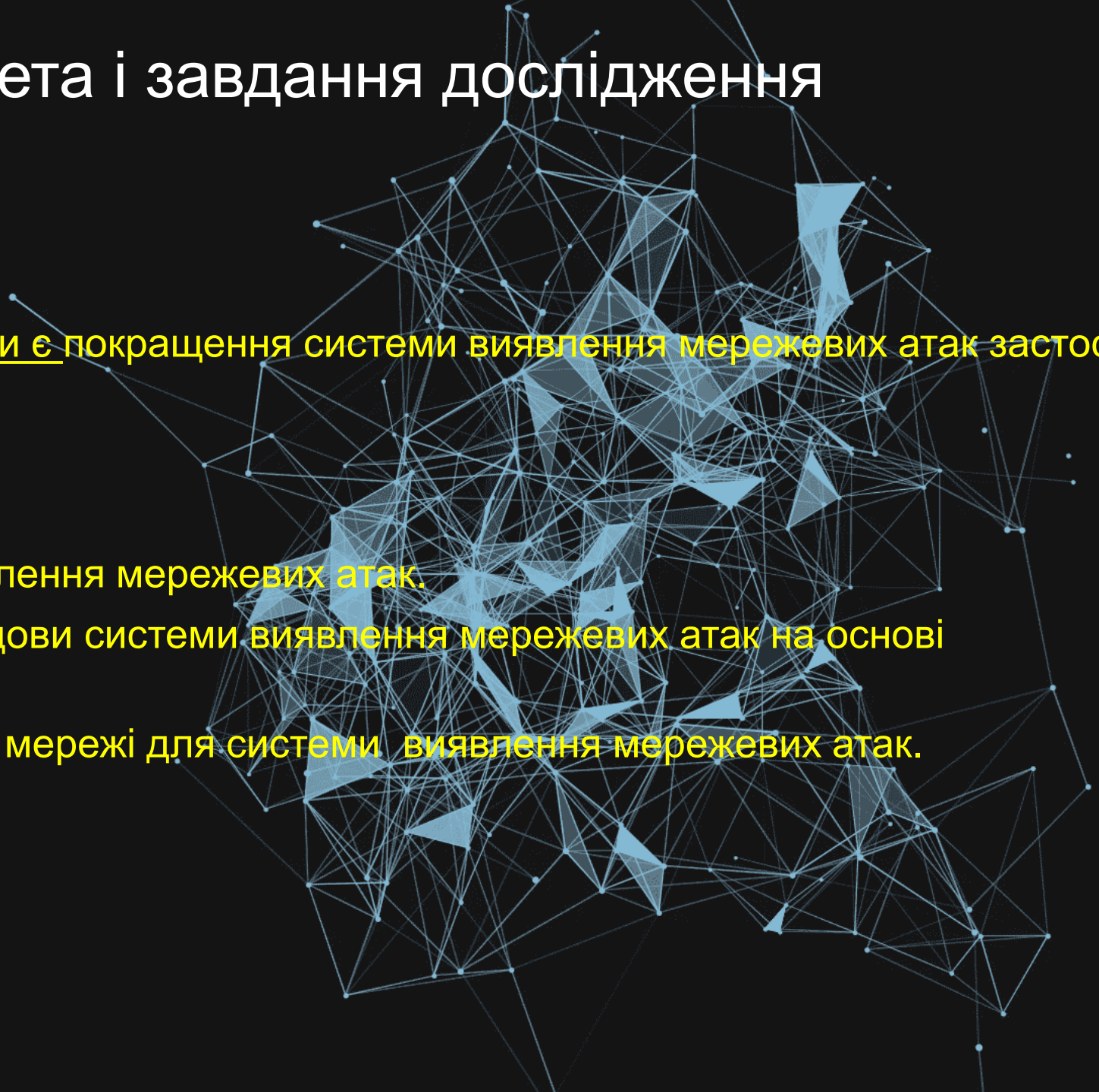
Керівник: ШТОМПЕЛЬ М.А.

Актуальність теми

Розвиток обчислювальних засобів та інформаційних технологій призводить до автоматизації різних процесів практично у всіх сферах життя суспільства: збільшуються обчислювальні потужності комп’ютерних засобів, удосконалюються технології мережевої взаємодії, змінюються формати і вимоги до побудови інформаційних систем. Слідом за розвитком інформаційних технологій з не меншою швидкістю з’являються нові загрози інформаційній безпеці, тому проблема захисту інформацій залишається ключовим напрямком наукових досліджень.

Мережеві атаки є одним з основних видів порушення інформаційної безпеки в розподілених обчислювальних мережах, особливо в ЗСУ.

Мета і завдання дослідження



Метою дослідження даної роботи є покращення системи виявлення мережевих атак застосовуючи нейромережеві технології

Завдання дослідження:

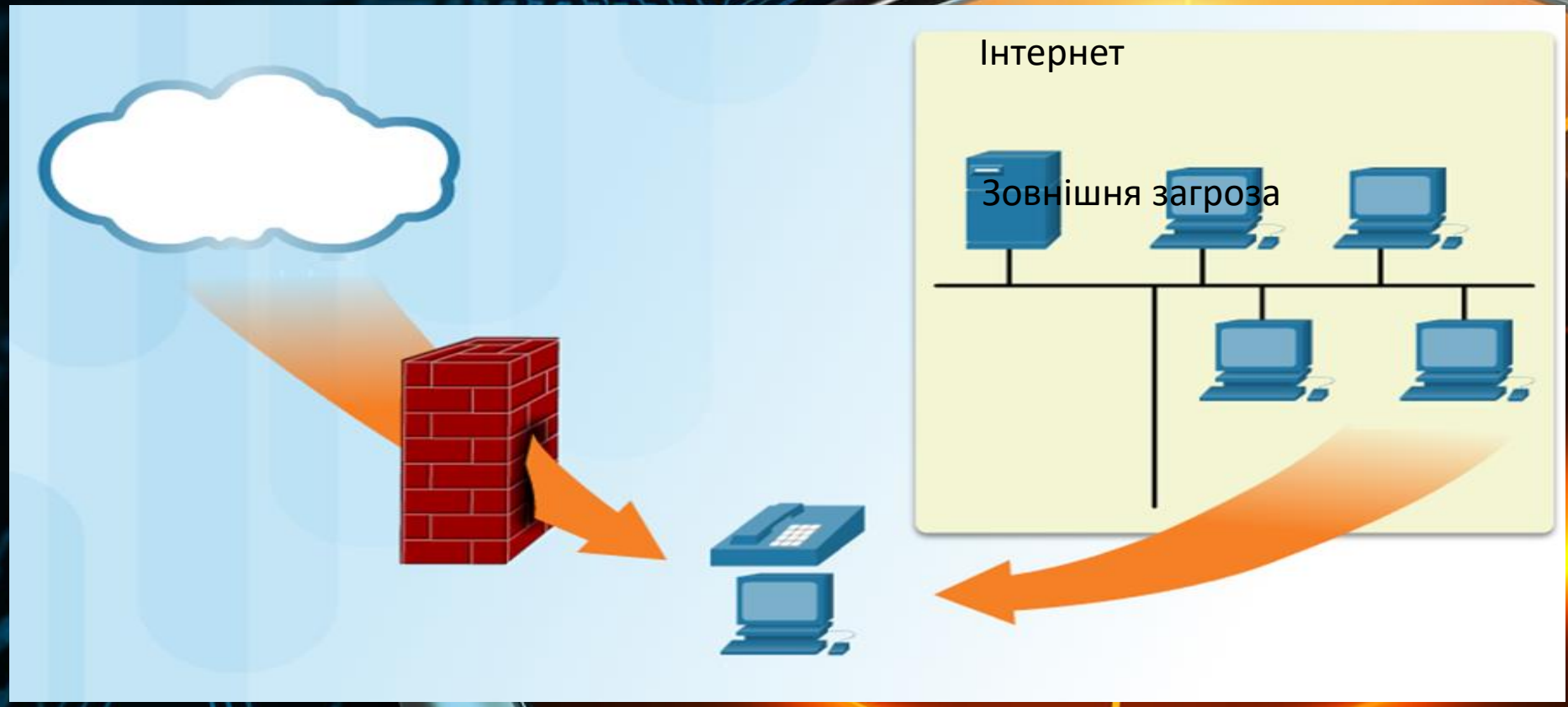
- Провести аналіз систем виявлення мережевих атак.
- Проаналізувати методи побудови системи виявлення мережевих атак на основі нейромережевих технологій.
- Розробити модель нейронної мережі для системи виявлення мережевих атак.

РОЗДІЛ 1

АНАЛІЗ СТАНУ ТА НАПРЯМКИ РОЗВИТКУ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК

Аналіз мережевих атак

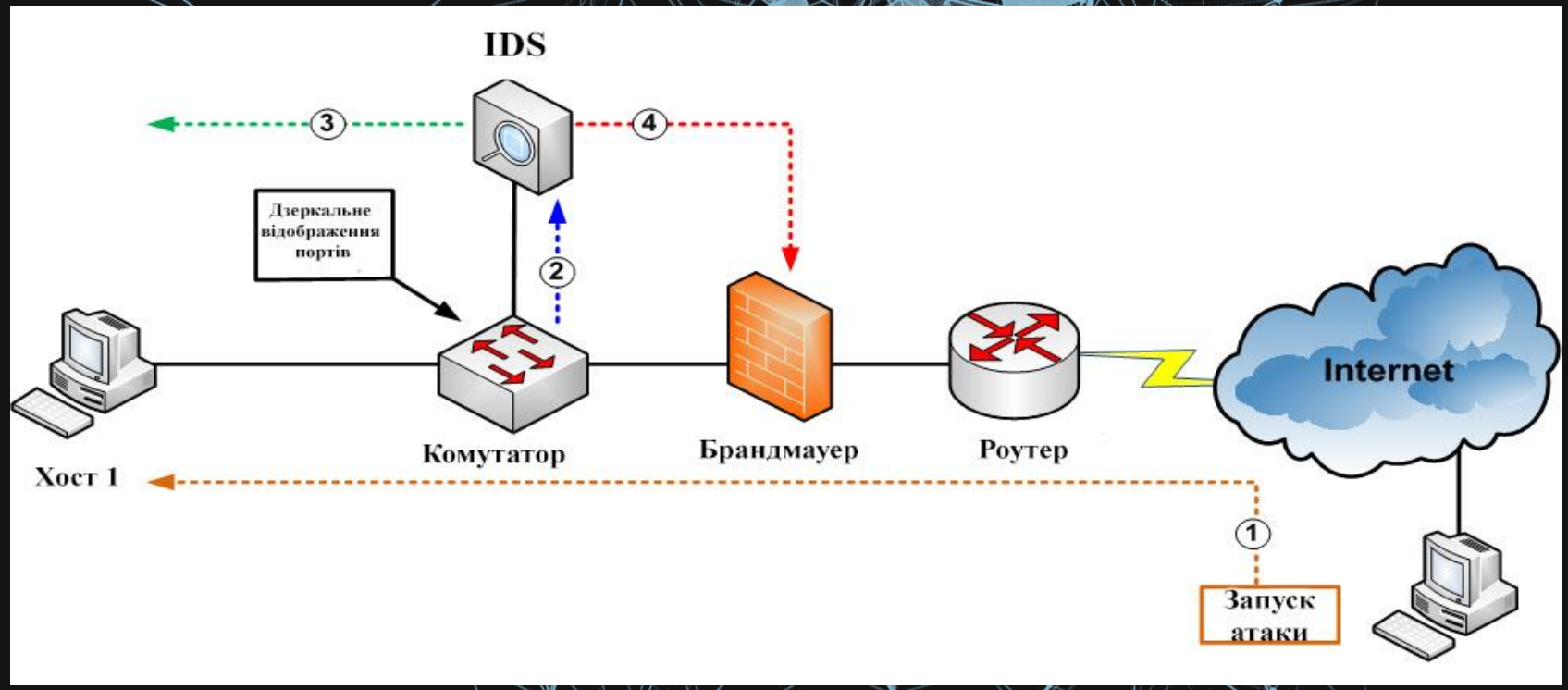
Мережева атака - спроба обійти політику безпеки системи, що надає зловмисникам доступ для отримання або модифікації інформації, навіть руйнуючи систему.



Мета будь-якої атаки - дістати несанкціонований доступ до інформації. Існують дві принципові можливості доступу до інформації: перехоплення і спотворення.

Системи виявлення мережевих атак, застосування та напрямки розвитку

IDS - поєднання двох слів вторгнення та система виявлення. Вторгнення відноситься до несанкціонованого доступу до інформації в комп'ютері або мережевих системах, щоб порушити її цілісність, конфіденційність або доступність. Система виявлення є механізмом безпеки для виявлення такої незаконної діяльності.



IPS - система активного захисту. Як і IDS, він намагається визначити потенційні загрози на основі функцій моніторингу захищеного хоста або мережі і може використовувати сигнатуру, аномалію або методи гібридного виявлення.

Системи виявлення мережевих атак, застосування та напрямки розвитку

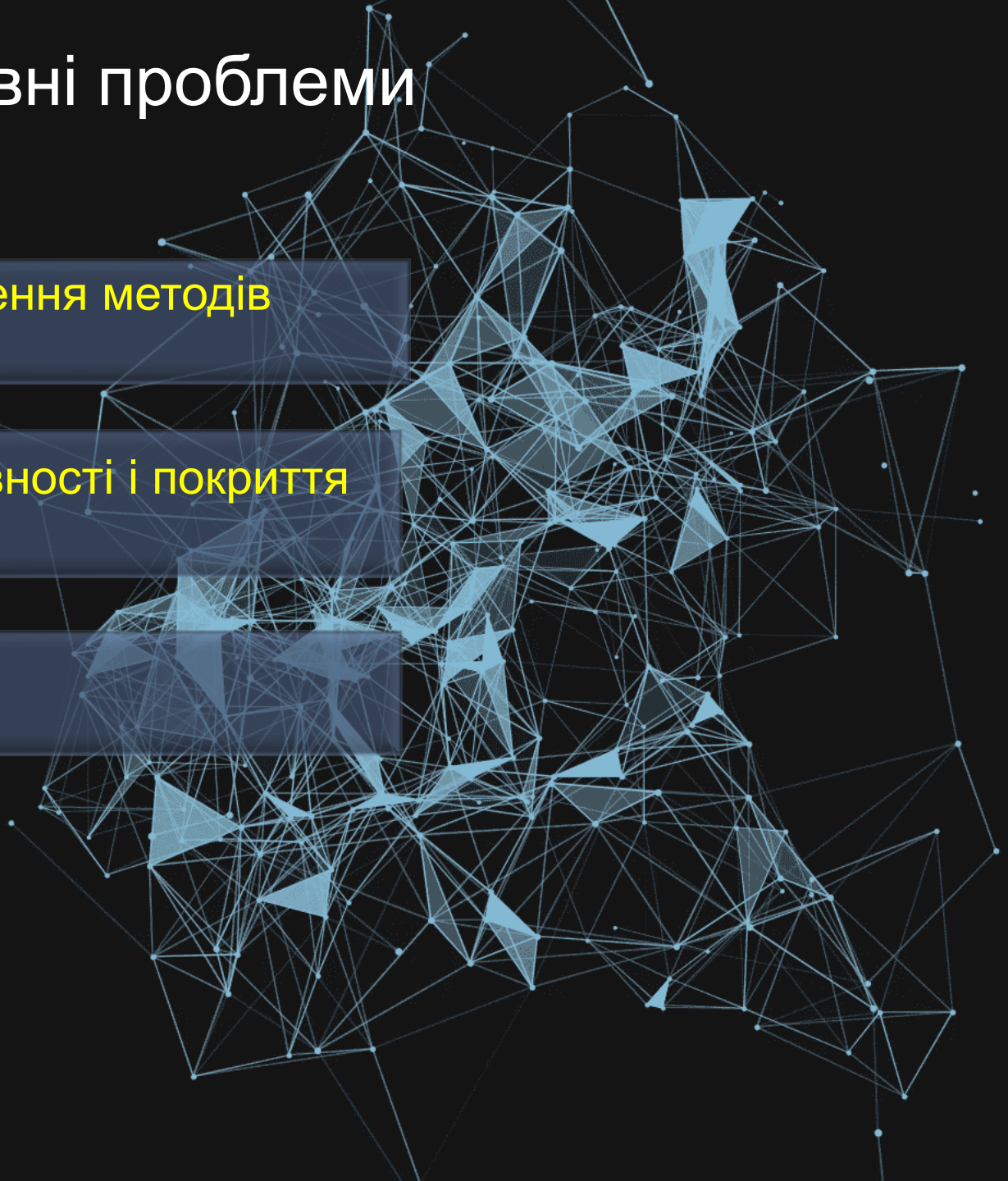
IDS та IPS мають свої переваги та недоліки. При виборі системи для потенційного випадку використання важливо враховувати компроміси між доступністю системи і зручністю використання і необхідністю захисту. IDS залишає вікно для зловмисника, щоб завдати шкоди цільовій системі, в той час як помилкове позитивне виявлення IPS може негативно вплинути на зручність використання системи.

Характеристика *IDS* та *IPS*

Параметр	IPS	IDS
Абревіатура	система запобігання вторгнень	система виявлення вторгнень
Філософія	IPS - це пристрій, який перевіряє трафік, виявляє його, класифікує, а потім активно зупиняє шкідливий трафік від атаки.	IDS - це пристрій або програмний додаток, який відстежує трафік на порушення шкідливої активності або політики і надсилає оповіщення про виявлення.
Принцип роботи	перевіряє трафік у реальному часі та шукає схеми руху або підписи атаки, а потім запобігає нападам на виявлення	виявляє трафік у реальному часі та шукає схеми трафіку або підписи атаки, і вони генерують сповіщення
Режим конфігурації	вбудований режим , як правило, у шарі 2	кінцевий хост (через проліт) для моніторингу та виявлення
Розміщення	вбудований зазвичай після брандмауера	Не вбудований через проліт порту (або дотиком)
Дії щодо виявлення несанкціонованого трафіку	Запобігання руху по виявленню аномалії	Оповіщення/тривоги про виявлення аномалії
Пов'язані термінології	– фільтрація пакетів зі станом	– виявлення на основі аномалії виявлення підпису

Головні проблеми

- Обмежена можливість оновлення методів виявлення
- Відсутність тестів продуктивності і покриття мережі
- Недолік ефективності



Висновок до першого розділу

У даному розділ було визначено місце систем виявлення мережевих атак та потрібність новітніх методів виявлення атак, проаналізовано IDS та IPS.

Кіберзлочинці орієнтуються на користувачів комп’ютерів, використовуючи складні методи, а також стратегії соціальної інженерії. Деякі кіберзлочинці стають дедалі витонченішими та мотивованішими. Вони продемонстрували свою здатність приховувати свою особистість, приховувати своє спілкування, віддаляти свою особу від нелегальної вигоди та використовувати інфраструктуру, стійку до компромісів. Тому для комп'ютерних систем стає все більш важливим захист за допомогою вдосконалених систем виявлення вторгнень, здатних виявляти сучасні шкідливі програми. Для проектування та побудови таких систем IDS необхідно мати повний огляд переваг та обмежень сучасних досліджень якому розглянуто в даному розділі.

РОЗДІЛ 2

АНАЛІЗ МЕТОДІВ ДЛЯ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК НА ОСНОВІ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ

Штучний інтелект

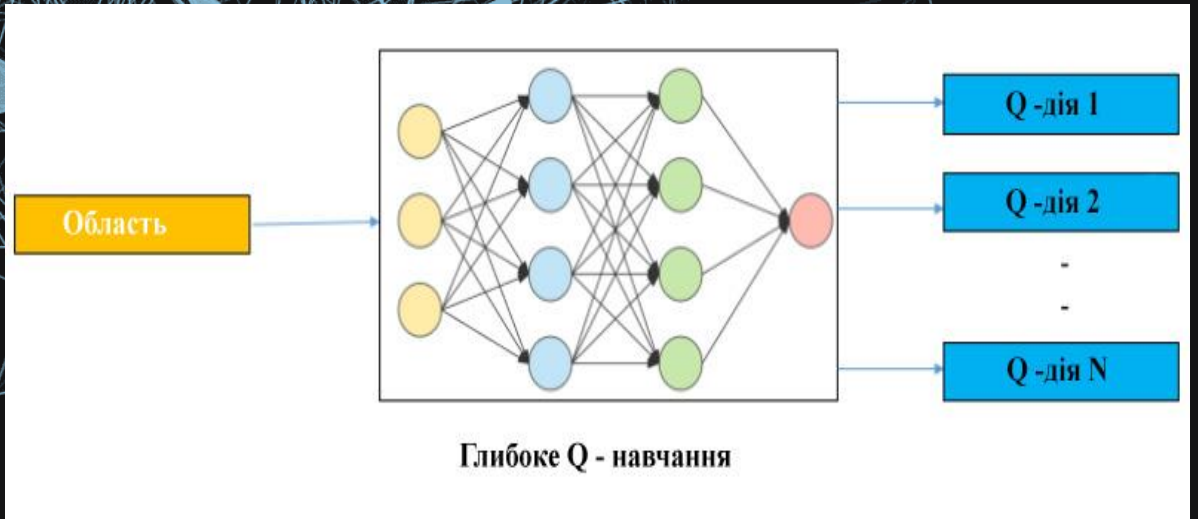
***Autoencoders* (AE)** - це нейронні мережі, які мають на меті копіювати свої входи у свої виходи. Вони працюють, стискаючи вхідні дані у подання з прихованим простором, а потім реконструюючи вихідні дані з цього подання. Цей тип мережі складається з двох частин:

Штучні нейронні мережі (Artificial Neural Networks, ANNs) - це один з напрямків досліджень в області штучного інтелекту, засноване на спробах відтворити нервову систему людини[7]. А саме: здатність нервової системи навчатися і виправляти помилки, що має дозволити змоделювати, хоча і досить грубо, роботу людського мозку.

Машинне навчання (Machine Learning, ML) є одним з напрямків штучного інтелекту. Основний принцип полягає в тому, що машини отримують дані і навчаються на них.

Глибоке навчання (*Deep Learning*, DL) є підмножиною машинного навчання і являє собою складний набір нейронних мереж з великою кількістю рівнів обробки, які розвивають високі рівні абстракції

Глибока Q-мережа DQN - це метод, що поєднує глибоке навчання та Q-навчання, який досяг успіху в обробці середовищ із введенням високих розмірів. Це багатошарова нейронна мережа, яка дає прогнозовану майбутню винагороду $yQ(s,a|\theta)$ для кожної можливої дії, де θ знаходяться параметри мережі. Іншими словами, DQN використовує нейронну мережу як наближення значення дії.



Глибоке Q - навчання

Методи глибинного навчання

Автокодер

Генеративна змагальна мережа

Глибинна межа перекононь

Згорткова нейронна мережа

Глибока Q - мережа

База даних NSL-KDD

Опис даних мережевих атак

Висновок до другого розділу

Таким чином, на сьогоднішній день виявлення мережевих атак перетворюється на серйозну проблему, яка не вирішується лише традиційними методами. У найближчому майбутньому до вирішення цієї задачі будуть залучатися тільки штучний інтелект, який має багато переваг щодо прийняття рішення і виявлення аномального трафіка. При такому підході важливу роль буде відігравати наявність керуючих груп щодо контролю рішення з боку інтелекту.

Застосування методів нейромережевих технологій дозволяє системі навчатися та пристосовуватися до аномалії у мережі, таким чином, підвищити ймовірність виявляння мережевих атак.

Досліджено способи виявлення мережевих атак за допомогою нейромережевих технологій.

Проаналізовано методи DL, та вибраний один з них для навчання нейронної мережі щодо виявлення мережевих атак.

Проаналізовано набір даних мережевих атак NSL-KDD.

РОЗДІЛ 3
РОЗРОБКА МОДЕЛІ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ СИСТЕМИ ВИЯВЛЕННЯ МЕРЕЖЕВИХ АТАК

Реалізація програмного коду для зберігання, форматування набору даних

Алгоритм був написаний на *python* та протестований в блокноті *Colaboratory*. Оскільки навчання нейронної мережі залежне від обчислювальної потужності машин було розглянута пропозиції щодо користування блокнотом *Jupiter*, який розміщений в сервісі *Colab*.

Jupiter, який розміщений в сервісі *Colab*.

Для маніпулювання даними та їхнього аналізу було інстальовано бібліотеки:

- *Tensorflow;*
- *Pandas;*
- *Numpe;*
- *Keras;*
- *Scikit-learn;*
- *Matplotlib.*

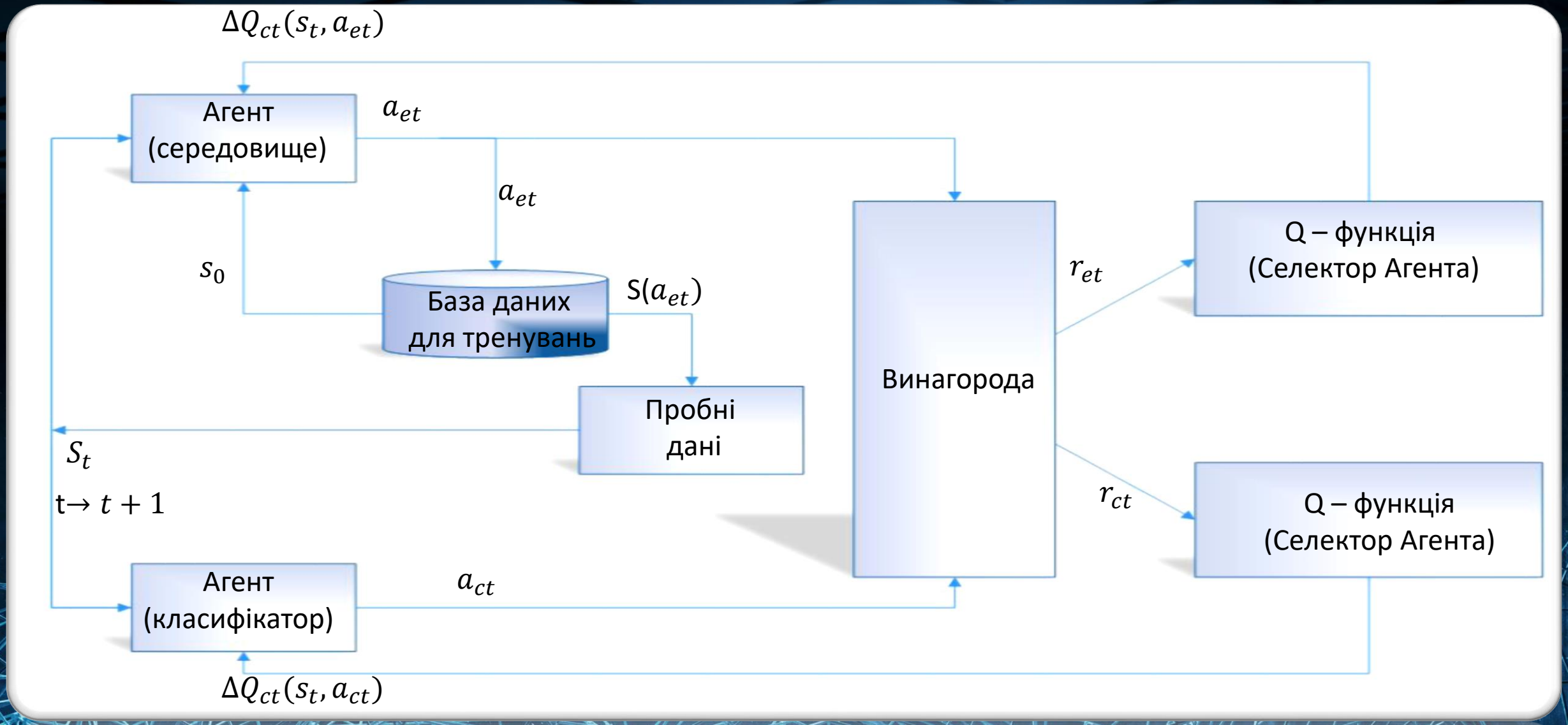
Для роботи з набором даних мережеві атаки було написано код який імпортував ці дані, приклад коду:

```
self.train_path=kwargs.get('train_path','datasets/NSL/KDDTrain+.txt')
self.test_path=kwargs.get('test_path', datasets/NSL/KDDTest%2B.txt')
```

Після імпортування даних було проведено форматування данних та збереження їх:

```
self.df = pd.concat([self.df.drop('protocol_type', axis=1), pd.get_dummies(self.df['protocol_type'])], axis=1)self.df = pd.concat([self.df.drop('service', axis=1), pd.get_dummies(self.df['service'])], axis=1)self.df = pd.concat([self.df.drop('flag', axis=1), pd.get_dummies(self.df['flag'])], axis=1)
test_df = self.df.iloc[train_indx:self.df.shape[0]]test_df = shuffle(test_df,random_state=np.random.randint(0,100))self.df = self.df[:train_indx]self.df = shuffle(self.df,random_state=np.random.randint(0,100))test_df.to_csv(self.formated_test_path,sep=',',index=False)
```

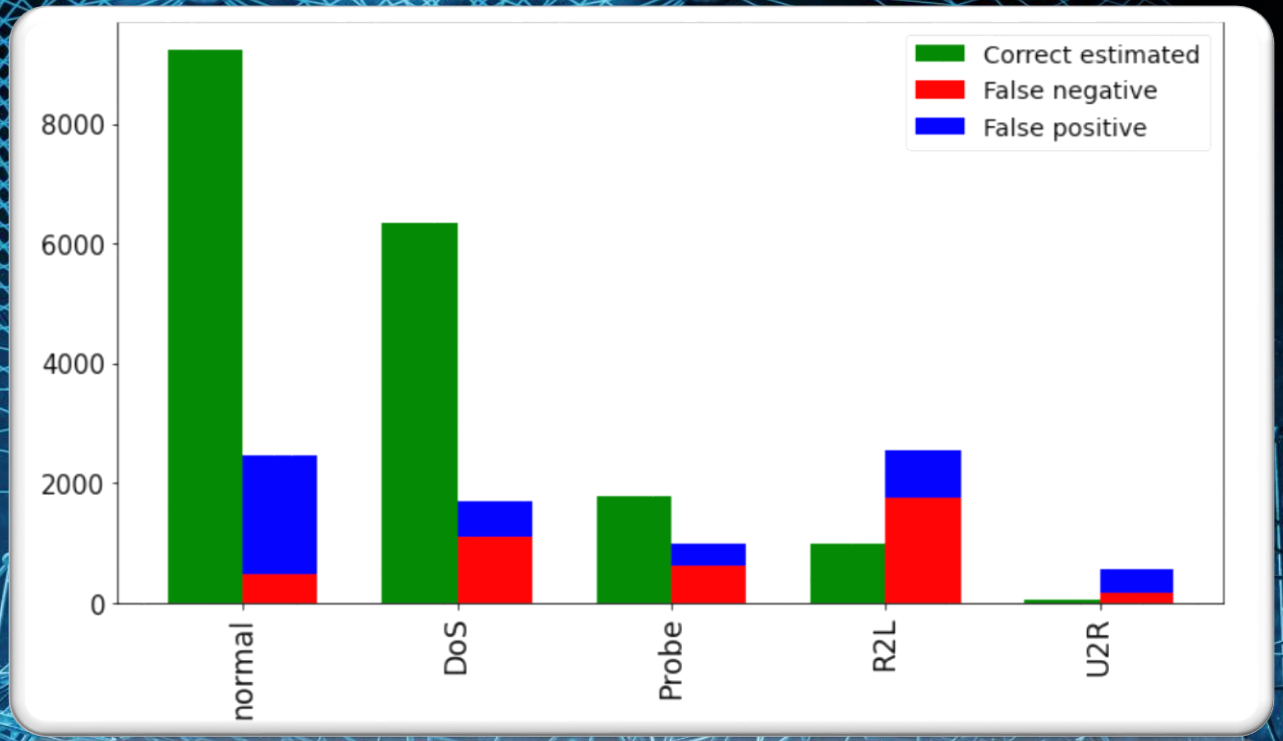
Архітектура нейронної мережі



Результати оцінки ефективності

Total reward: 18416 | Number of samples: 22544 | Accuracy = 81.69%

	Estimated	Correct	Total	F1_score
normal	11243	9243	9712	88.2176
DoS	6920	6348	7458	88.3016
Probe	2163	1794	2421	78.2723
R2L	1779	990	2753	43.6893
U2R	439	41	200	12.8326



Висновок до третього розділу

В даному розділі було запропоновано модель нейронної мережі на основі методу DQN з підкріпленням.

Описано основні дії щодо навчання нейронної мережі. Вхідні дані були отримані з набору KDD, який містить оброблені відомості у вигляді масивів з 42 ключових значень.

Для тестування нейронної мережі було створено матрицю неточності, яка порівнювала початкові дані та на виході і відображається у графіках.

Даний метод DL показав високі результати щодо точності за 100 епох навчання та яка, у подальшому, може використовуватися у системах виявлення мережевих атак. Для покращення виявлення атак нейронною мережею потрібно змінити ітерацію епох навчання, адже навчання тривало зі 100 епох і зі. Тому випливає, що із збільшенням кількості епох навчання нейронної мережі за -алгоритмом, то якість і точність буде збільшуватися. DQ

Висновки

- Аналіз систем виявлення мережевих атак показав застарілість методів виявлення.
- Використання нейронних мереж є оптимальним рішенням.
- Розроблена модель нейронної мережі для впровадження у системи виявлення мережевих атак