МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних інформаційних систем

_____Юрій КОНДРАТЕНКО

«____»_____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ МАГІСТРА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ІДЕНТИФІКАЦІЇ І КЛАСИФІКАЦІЇ DOS-АТАК

Спеціальність 122 Комп'ютерні науки

Освітня програма «Інтелектуальні інформаційні системи»

Здобувач _____ Андрій КАРПЕНКО

«___»_____ 2024 р.

Керівник д-р техн. наук, професор _____ Олександр ГОЖИЙ

«___»_____ 2024 р.

Миколаїв – 2024

Чорноморський національний університет імені Петра Могили
(повне найменування закладу вищої освіти)

Факультет	Комп'ютерних наук
Кафедра	Інтелектуальних інформаційних систем
Рівень вищої освіти	Другий (магістерський)
Освітній ступень	Магістр
Спеціальність	122 Комп'ютерні науки
Освітня програма	Інтелектуальні інформаційні системи

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем

_____ Юрій КОНДРАТЕНКО

«____» _____ 2024 р.

ЗАВДАННЯ
на кваліфікаційну роботу здобувача

Карпенка Андрія Юрійовича
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Інтелектуальна система ідентифікації і класифікації DoS-атак».

Керівник роботи: Гожий Олександр Петрович, професор кафедри ІІС, д-р техн. наук, професор.

Затверджена наказом ЧНУ ім. Петра Могили від «03» червня 2024 р. № 140/1.

2. Строк представлення кваліфікаційної роботи «16» грудня 2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні: висока точність класифікації DoS-атак навченою нейронною мережею; набір даних CIC-IDS2017.

4. Перелік питань, що підлягають розробці: аналіз сучасного стану задачі класифікації DoS-атак; здійснення аналізу існуючих рішень та досліджень; огляд на існуючі архітектури нейронних мереж; обґрунтування вибору архітектури

нейронної мережі для класифікації DoS-атак; порівняльний аналіз отриманих результатів на основі звітів про класифікацію.

5. Перелік графічних матеріалів: презентація, рисунки, таблиця.

Керівник роботи _____ Олександр Гожий
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Андрій Карпенко
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата видачі завдання «07» червня 2024 р.

КАЛЕНДАРНИЙ ПЛАН
кваліфікаційної роботи

Тема: Інтелектуальна система ідентифікації і класифікації DoS-атак

№	Найменування роботи	Початок	Закінчення	Примітки
1	Отримання завдання на виконання КР	03.06.2024	07.06.2024	Виконано
2	Аналіз предметної області та постановка задачі	10.06.2024	20.06.2024	Виконано
3	Огляд літературних джерел за темою кваліфікаційної роботи, зокрема аналіз публікацій та аналогічних систем, щодо діагностування легеневої хвороби	21.06.2024	01.07.2024	Виконано
4	Огляд існуючих архітектур штучних нейронних мереж для вирішення поставленої задачі	01.09.2024	25.10.2024	Виконано
5	Реалізація обраних технологій з аналізом отриманих результатів	26.10.2024	21.11.2024	Виконано
6	Перший попередній захист КР на засіданні комісії кафедри	22.11.2024	22.11.2024	Виконано
7	Корегування роботи за результатами попереднього захисту	23.11.2024	05.12.2024	Виконано
8	Другий попередній захист КР на засіданні комісії кафедри	06.12.2024	06.12.2024	Виконано
9	Доробка та остаточне оформлення КР	07.12.2024	10.12.2024	Виконано
10	Подання КР, її електронної копії та інших документів (відгуку, рецензії) до захисту	16.12.2024	17.12.2024	Виконано

Керівник роботи _____ Олександр ГОЖИЙ
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Здобувач _____ Андрій Карпенко
(Особистий підпис) *(Власне ім'я ПРІЗВИЩЕ)*

Дата складання календарного плану
«19» червня 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи
здобувача групи 601м ЧНУ ім. Петра Могили

Карпенка Андрія Юрійовича

на тему: **“ ІНТЕЛЕКТУАЛЬНА СИСТЕМА ІДЕНТИФІКАЦІЇ І КЛАСИФІКАЦІЇ DOS-АТАК”**

Актуальність даного дослідження полягає у необхідності підвищення точності класифікації DoS-атак на основі аналізу лог-файлів серверів з урахуванням сучасних тенденцій у кібербезпеці та архітектурах глибинного нейронного навчання. Це дозволить покращити захист від зростаючої кількості кібератак, забезпечити безперервність бізнес-процесів і захист критичної інформації.

Об’єктом дослідження є лог-файли серверу.

Предметом дослідження є класифікація DoS-атак.

Метою дослідження є покращення точності класифікації DoS-атак шляхом створення інтелектуальної системи ідентифікації та класифікації DoS-атак із використанням методів глибинного навчання та налаштування гіперпараметрів.

Кваліфікаційна робота складається зі вступу, чотирьох розділів, висновків та додатків. У першому розділі розкрито теоретичні засади розвитку інтелектуальних систем ідентифікації та класифікації DoS-атак та принцип їх дії шляхом дослідження останніх публікацій . У другому проведено аналіз передових архітектур нейронних мереж. Проведено порівняльний аналіз та обрано згорткові та рекурентні нейронні мережі для подальшого аналізу. У третьому розділі сформовано, проаналізовано, очищено, стандартизовано та збалансовано набір даних для подальшої роботи. У четвертому розділі описано моделювання, проєктування моделей, та виконано порівняльний аналіз.

Кваліфікаційна робота містить 88 сторінок, 53 рисунки, 6 таблиць, 47 джерел, 2 додатки.

Ключові слова: *DoS-атака, нейронна мережа, класифікація, стандартизація, лог-файл.*

ABSTRACT

to the qualification work by the student of the group 601m of Petro Mohyla Black Sea National University

Karpenko Andrii

“INTELLIGENT SYSTEM FOR IDENTIFICATION AND CLASSIFICATION OF DOS ATTACKS”

The relevance of this research lies in the necessity to improve the accuracy of DoS attack classification based on server log file analysis, taking into account current trends in cybersecurity and deep neural network architectures. This will enhance protection against the growing number of cyberattacks, ensure the continuity of business processes, and safeguard critical information.

The object of the research is server log files.

The subject of the research is the classification of DoS attacks.

The purpose of the research is to improve the accuracy of DoS attack classification by developing an intelligent system for their identification and classification, using deep learning methods and hyperparameter tuning.

The qualification work consists of an introduction, four chapters, conclusions, and appendices. The first chapter reveals the theoretical foundations of the development of intelligent systems for DoS attack identification and classification, and explains their principle of operation through the study of recent publications. The second chapter analyzes advanced neural network architectures and, after a comparative analysis, selects convolutional and recurrent neural networks for further study. The third chapter involves forming, analyzing, cleaning, standardizing, and balancing the dataset for subsequent work. The fourth chapter describes the modeling, design of the models, and presents a comparative analysis of the results.

The qualification work comprises 88 pages, 53 figures, 6 tables, 47 references, and 2 appendices.

Keywords: DoS attack, neural network, classification, standardization, log file.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

DDoS – Distributed Denial-of-Service

DoS – Denial of Service

DNN – Deep Neural Network

CNN – Convolutional Neural Networks

RNN – Recurrent Neural Networks

LSTM – LSTM

PCAP – PCAP

CSV – Comma-Separate Values

SMOTE – Synthetic Minority Oversampling Technique

ВСТУП

Атаки типу відмови в обслуговуванні (DoS) становлять одну з найбільших загроз у сфері інформаційної безпеки, спрямованих на порушення доступності ресурсів та сервісів. З розвитком технологій та збільшенням складності мережевої інфраструктури, DoS-атаки стають все більш витонченими, що ускладнює їх своєчасне виявлення та нейтралізацію. Існуючі методи виявлення, такі як сигнатурний аналіз або традиційні системи виявлення вторгнень, часто не справляються з цим завданням через високу динаміку та варіативність атак. Успіх організацій у протидії DoS-атакам залежить від здатності впроваджувати сучасні підходи, зокрема використання технологій штучного інтелекту та машинного навчання для аналізу мережевого трафіку та лог-файлів.

Актуальність даної роботи обумовлена необхідністю розробки ефективних інструментів для виявлення та класифікації DoS-атак на основі аналізу лог-файлів серверів. Зростаюча кількість кібератак та їх складність вимагають впровадження інтелектуальних систем, здатних оперативно реагувати на загрози, забезпечуючи безперервність бізнес-процесів та захист критичної інформації.

Метою даної роботи є розробка інтелектуальної системи виявлення та класифікації DoS-атак на основі аналізу мережевого трафіку, використовуючи сучасні алгоритми глибинного навчання. Така система має здатність не лише виявляти відомі атаки, але й адаптуватися до нових загроз за допомогою автоматизованого навчання на великих наборах даних.

Для досягнення цієї мети у роботі вирішуються наступні завдання:

– аналіз існуючих методів класифікації DoS-атак;

– розробка та тестування моделей машинного навчання для виявлення DoS-атак;

– проведення експериментального дослідження на основі реальних наборів даних мережевого трафіку;

– порівняння результатів запропонованих методів з існуючими рішеннями.

Об’єктом дослідження є процеси виявлення та класифікації DoS-атак на основі аналізу мережевого трафіку. Предметом дослідження є методи машинного та глибинного навчання, які використовуються для класифікації атак.

Робота базується на застосуванні методів глибинного навчання, що дозволяють ефективно аналізувати великі обсяги даних та виявляти складні шаблони атак у реальному часі. Використання сучасних алгоритмів, таких як згорткові та рекурентні нейронні мережі, дозволяє досягти високої точності в класифікації DoS-атак та підвищити надійність систем безпеки.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ, ОСТАННІХ ДОСЛІДЖЕНЬ ТА ПУБЛІКАЦІЙ

1.1 Історія та розвиток DoS-атак та протидій

Відмова в обслуговуванні є одним з найпоширеніших та найнебезпечніших типів кібератак (див. рис. 1.1), спрямованих на порушення доступності ресурсів або послуг для легітимних користувачів. З розвитком Інтернету та мережевих технологій DoS-атаки еволюціонували, стаючи все більш складними та руйнівними (див. рис. 1.2). Цей підрозділ присвячений аналізу історії розвитку DoS-атак та методів протидії їм.

Перші відомі DoS-атаки датуються початком 1980-х років, коли були виявлені вразливості в мережевих протоколах та системах. Однією з перших задокументованих атак була "Ping of Death", що використовувала можливість відправки ICMP-пакетів з розміром, більшим за максимально допустимий, що призводило до збою системи жертви [1].

У перші дні Інтернету мережеві протоколи розроблялися з акцентом на функціональність, а не на безпеку. Це створило можливості для зловмисників експлуатувати їх для порушення роботи систем.

"Ping of Death" (1996) атака використовувала можливість відправки ICMP ECHO-запитів з розміром пакету, що перевищує максимально допустимий за стандартом IP-протоколу (65535 байт). Це призводило до переповнення буферів і збою систем, які не могли правильно обробити такі пакети.

Teardrop Attack (1997) використовувала помилки в обробці фрагментованих IP-пакетів. Зловмисник відправляв пакети з перекриваючимися фрагментами, що викликало збій у системах при спробі їх зібрати.

Зі збільшенням кількості підключених до мережі пристроїв та зростанням пропускної здатності Інтернету, зловмисники почали використовувати множинні комп'ютери для одночасного здійснення атак [2].

Trinoo, TFN, Stacheldraht (1999): Ранні інструменти для організації DDoS-атак. Вони дозволяли зловмисникам контролювати мережі зламаних комп'ютерів (ботнети) для одночасного надсилання трафіку на цільову систему.

Атаки MafiaBoy (2000) організовував канадський підліток Майкл Кальче, відомий під псевдонімом MafiaBoy, організував серію DDoS-атак на великі інтернет-компанії, такі як Yahoo!, eBay, Amazon та CNN. Це призвело до тимчасової недоступності цих сайтів і значних фінансових втрат.

З розповсюдженням Інтернету речей (IoT) з'явилися нові можливості для створення великих ботнетів з вразливих пристроїв [3].

Mirai Botnet (2016): Використовував слабко захищені IoT-пристрої, такі як камери спостереження та маршрутизатори. Mirai здійснив одну з найбільших DDoS-атак, спрямовану на компанію Dyn, що призвело до збоїв у роботі багатьох популярних сервісів, включаючи Twitter та Netflix.

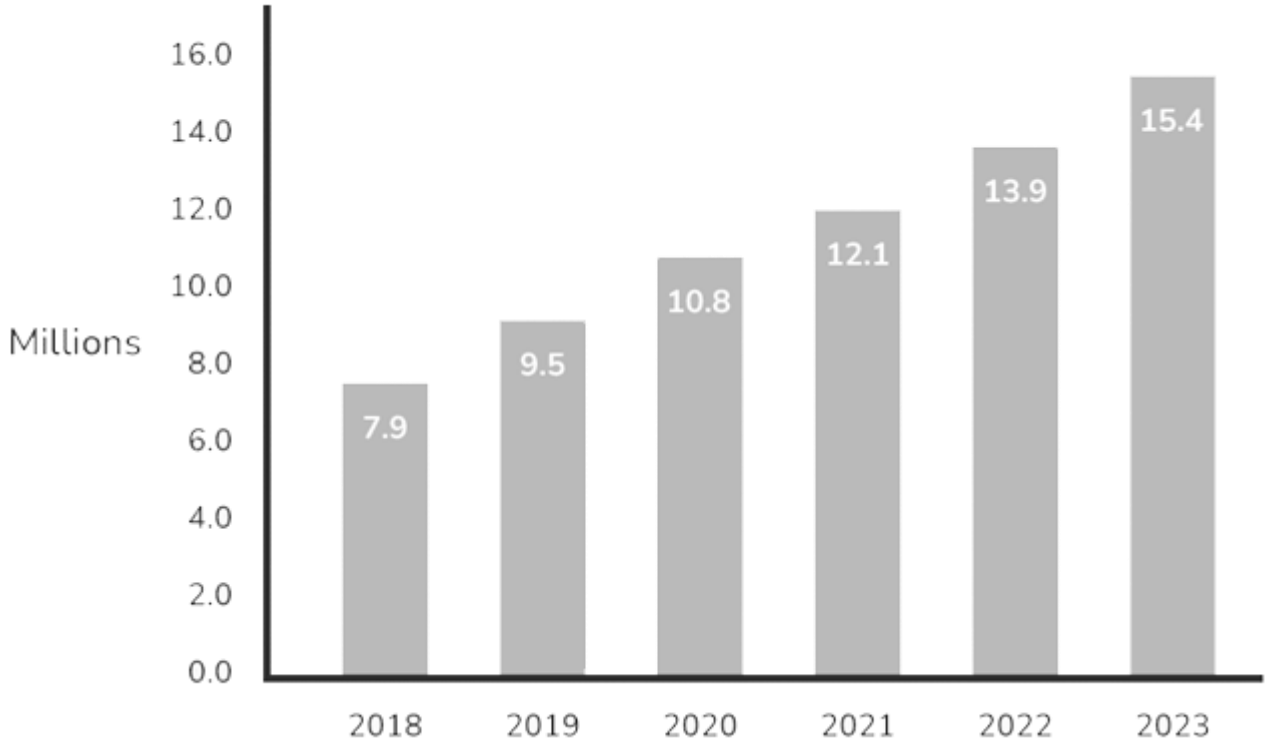

Рисунок 1.1 – Кількість DoS-атак з 2018 по 2023 рік

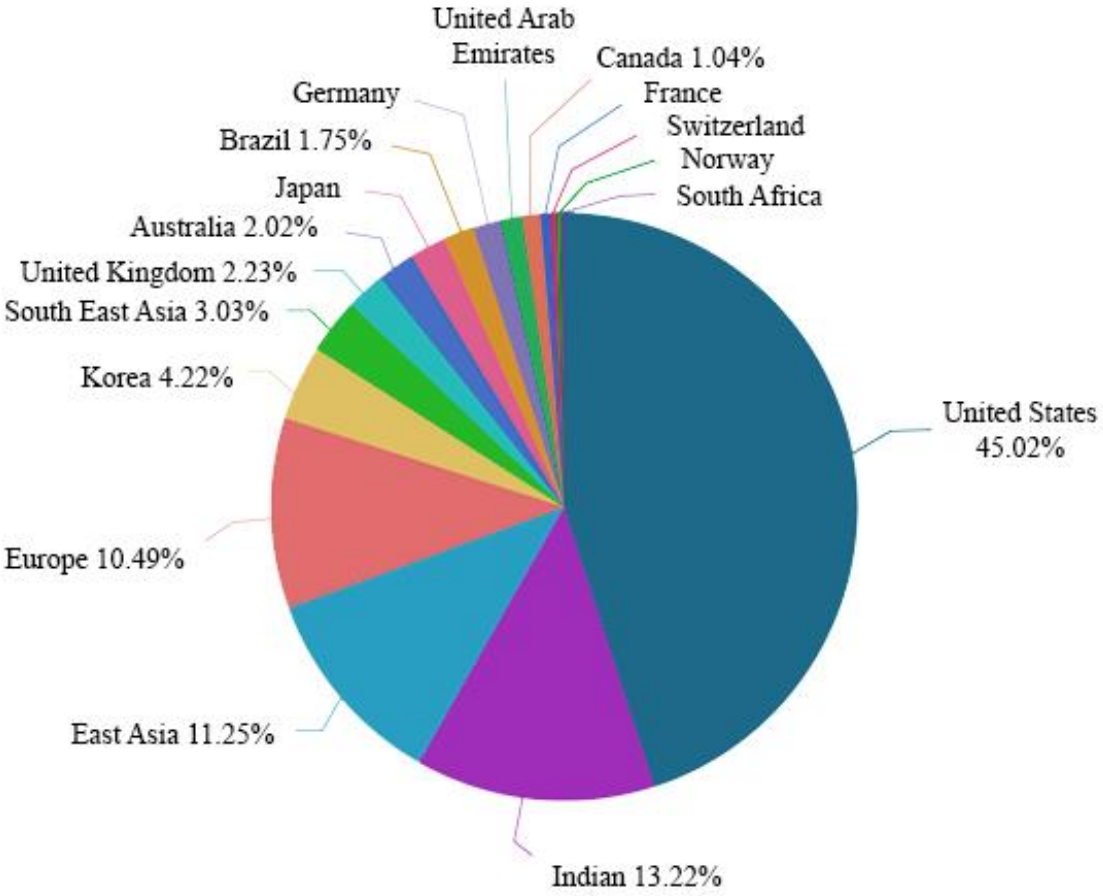

Рисунок 1.2 – Країни на які припала основна частка DoS-атак

З моменту появи перших DoS-атак у 1980-х роках, підходи до їхньої протидії зазнали значної еволюції. На початкових етапах методи захисту були здебільшого реактивними та зосереджувалися на усуненні вразливостей після того, як атака вже сталася. Адміністратори систем покладалися на базові мережеві інструменти, такі як брандмауери та прості фільтри, для обмеження доступу зловмисників. Проте ці засоби були недостатньо ефективними проти нових та складніших типів атак, що з'являлися з розвитком технологій.

У 1990-х роках, зі збільшенням масштабів Інтернету та розповсюдженням мережевих технологій, DoS-атаки стали більш поширеними та руйнівними. Це спонукало дослідників і фахівців з безпеки розробляти нові методи захисту. З'явилися перші системи виявлення вторгнень (IDS), які аналізували мережевий трафік на наявність відомих сигнатур атак. Проте ці системи часто не могли

впоратися з невідомими або модифікованими атаками, що вимагало вдосконалення методів аналізу.

Початок 2000-х років ознаменувався появою розподілених DDoS атак, які використовували мережі зламаних комп'ютерів для одночасного надсилання величезної кількості запитів до цільових серверів. Традиційні засоби захисту були безсилі перед такими атаками. Відповіддю на це стало впровадження систем запобігання вторгненням (IPS), які не лише виявляли, але й автоматично блокували шкідливий трафік. Також почали використовуватися методи розподілу навантаження та надмірності ресурсів, що дозволяло розосередити трафік і зменшити вплив атаки на окремі сервери.

Зі збільшенням складності атак виникла потреба в більш інтелектуальних системах захисту. У 2010-х роках почали активно застосовувати методи машинного навчання та глибинного навчання для аналізу мережевого трафіку. Ці технології дозволили створювати моделі, здатні виявляти аномалії в режимі реального часу та прогнозувати можливі атаки на основі історичних даних. Це суттєво підвищило ефективність виявлення нових та невідомих типів DoS-атак.

Одночасно з технічними рішеннями велика увага почала приділятися організаційним заходам безпеки. Компанії стали впроваджувати комплексні стратегії кібербезпеки, які включали регулярні аудити, навчання персоналу, розробку політик безпеки та планів реагування на інциденти. Це дозволило створити багаторівневий захист, де технічні та людські ресурси працюють разом для запобігання та реагування на атаки.

З появою Інтернету речей (IoT) та зростанням кількості підключених пристроїв з'явилися нові виклики у сфері кібербезпеки. Багато IoT-пристроїв мають слабкі або відсутні механізми захисту, що робить їх вразливими до компрометації та використання в ботнетах для здійснення DDoS-атак. У відповідь на це були розроблені спеціалізовані рішення для моніторингу та захисту IoT-мереж, а також впроваджені стандарти безпеки для виробників таких пристроїв.

Сучасні підходи до протидії DoS-атакам [4,5] все більше зосереджуються на проактивному захисті та адаптивності. Використовуються хмарні сервіси та мережі доставки контенту (CDN) для розподілу трафіку та забезпечення стійкості до навантажень. Інтелектуальні системи аналізують поведінку трафіку, виявляють відхилення від нормальних патернів та автоматично застосовують необхідні заходи. Крім того, активно розвивається міжнародна співпраця у сфері кібербезпеки, що включає обмін інформацією про нові загрози та спільну розробку стандартів і рекомендацій.

У підсумку, розвиток протидій кібер-атакам пройшов шлях від простих реактивних заходів до комплексних, проактивних стратегій, що поєднують технологічні інновації, організаційні практики та міжнародну співпрацю. Цей процес є безперервним і вимагає постійного вдосконалення знань і технологій для ефективної боротьби з еволюціонуючими загрозами в цифровому світі.

1.2 Огляд предметної сфери виявлення та класифікації DoS-атак

DoS-атаки можуть завдати значної шкоди організаціям, спричиняючи фінансові втрати, зниження довіри клієнтів та пошкодження репутації. У сучасному цифровому середовищі, де доступність сервісів є ключовим фактором успіху, ефективне виявлення та класифікація DoS-атак стає критично важливим. Завдяки своєчасному реагуванню на загрози, організації можуть мінімізувати вплив атак, швидко відновлювати нормальну роботу та захищати свої системи від подальших компрометацій.

Сучасні методи виявлення та класифікації DoS-атак базуються на комбінації традиційних технік та передових технологій штучного інтелекту. Глибинні нейронні мережі (DNN) та їхні варіанти, такі як згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN), використовуються для автоматичного витягнення ознак з великих обсягів даних, що дозволяє ефективно розпізнавати складні шаблони атак (див. рис. 1.3). Основні підходи включають [5]:

– глибинне навчання: використання DNN, CNN та RNN для аналізу складних структур даних та виявлення аномалій;

– методи ансамблю: поєднання кількох моделей машинного навчання для підвищення точності та надійності виявлення атак;

– аналіз лог-файлів серверів: використання логів для отримання детальної інформації про мережеві з'єднання та події безпеки, що дозволяє ідентифікувати відхилення від нормальної поведінки;

– обробка великих даних та хмарні технології: використання хмарних сервісів та технологій обробки великих даних для масштабування систем виявлення та забезпечення реального часу реагування на загрози.

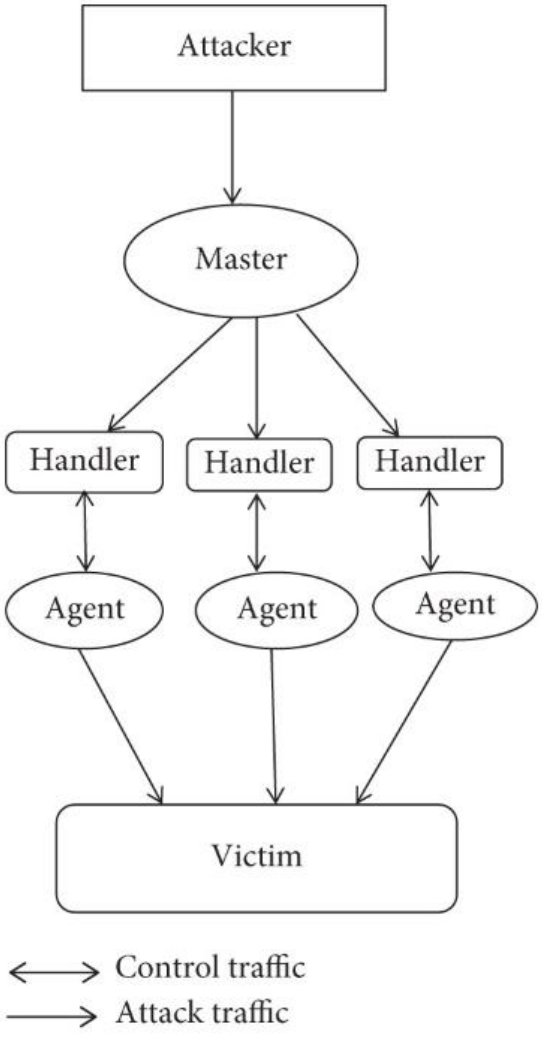

Рисунок 1.3 – Схема ботнет DoS-атаки

Інтелектуальні системи, що базуються на машинному навчанні, можуть ефективно аналізувати ці дані, виявляти відхилення від нормальної поведінки та класифікувати типи атак з високою точністю. Одним із основних викликів у виявленні та класифікації DoS-атак є високий рівень динаміки та різноманіття атак. Зловмисники постійно вдосконалюють свої методи, що ускладнює завдання системам захисту. Іншими викликами є велика кількість даних, що потребує ефективних алгоритмів обробки та аналізу, а також необхідність забезпечення реального часу реагування на загрози. Крім того, проблема дисбалансу класів у даних є важливим аспектом, оскільки кількість нормального трафіку зазвичай значно перевищує кількість атакуючого трафіку. Це може призводити до упередженості моделей, що негативно впливає на їхню здатність ефективно виявляти рідкісні типи атак.

Однією з важливих сучасних тенденцій у сфері DoS-атак є використання Інтернету речей (IoT) [6] для створення масштабних ботнетів. Зі зростанням кількості підключених до Інтернету пристроїв, таких як смарт-телевізори, камери спостереження, домашні маршрутизатори та інші побутові гаджети, зловмисники отримали доступ до значно більшої кількості потенційних ботів. Ці пристрої часто мають обмежені ресурси та слабкі механізми безпеки, що робить їх вразливими до компрометації. Наприклад, ботнет Mirai, який виник у 2016 році, успішно використовував вразливості у багатьох IoT-пристроях для здійснення однієї з найбільших DDoS-атак на той час, спричинивши збої у роботі таких великих сервісів, як Twitter, Netflix та інших.

Зловмисники все частіше використовують ампліфікаційні атаки, які дозволяють значно збільшити обсяг трафіку, спрямованого до жертви, за рахунок використання слабо захищених серверів. Наприклад, протоколи DNS, NTP, LDAP та Memcached можуть бути використані для відправки великих обсягів даних у відповідь на невеликі запити, що суттєво підсилює атакуючий трафік. У 2018 році GitHub пережив одну з найбільших DDoS-атак, що досягла пікової пропускної здатності 1,7 Тбіт/с, завдяки використанню Memcached-серверів як ампліфікаторів.

З розвитком технологій штучного інтелекту (AI) та машинного навчання (ML) [7], зловмисники починають застосовувати ці методи для автоматизації та удосконалення своїх атак. AI та ML дозволяють атакам стати більш адаптивними та динамічними, здатними швидко змінюватися та обходити традиційні механізми захисту. Наприклад, з використанням алгоритмів машинного навчання, зловмисники можуть автоматично генерувати нові типи DoS-атак, аналізувати ефективність різних стратегій та вибирати найефективніші методи для конкретних цілей.

Інтеграція хмарних технологій та впровадження мереж 5G створюють нові можливості для здійснення DoS-атак. Хмарні сервіси, завдяки своїй масштабованості та гнучкості, стають привабливими цілями для атак, оскільки можуть бути використані для створення великого обсягу трафіку або зниження продуктивності сервісів. Мережі 5G, хоча й забезпечують значно вищі швидкості передачі даних та меншу латентність, також відкривають нові вразливості через складність їхньої архітектури та нові протоколи, що впроваджуються. Атакуючі можуть використовувати ці особливості для проведення більш ефективних та складних DoS-атак.

Атаки на рівні додатків стають все більш поширеними та складними. Зловмисники використовують вразливості в програмному забезпеченні додатків для здійснення DoS-атак, що важче виявити та запобігти. Наприклад, атака Slowloris тримає відкритими численні HTTP-з'єднання з сервером, поступово надсилаючи неповні заголовки запитів, що змушує сервер витрачати ресурси на підтримку цих з'єднань. Інший приклад — атака R-U-Dead-Yet (R.U.D.Y.) (див. рис. 1.4), яка використовує довгі HTTP POST-запити з повільною передачею даних, спричиняючи виснаження ресурсів сервера.

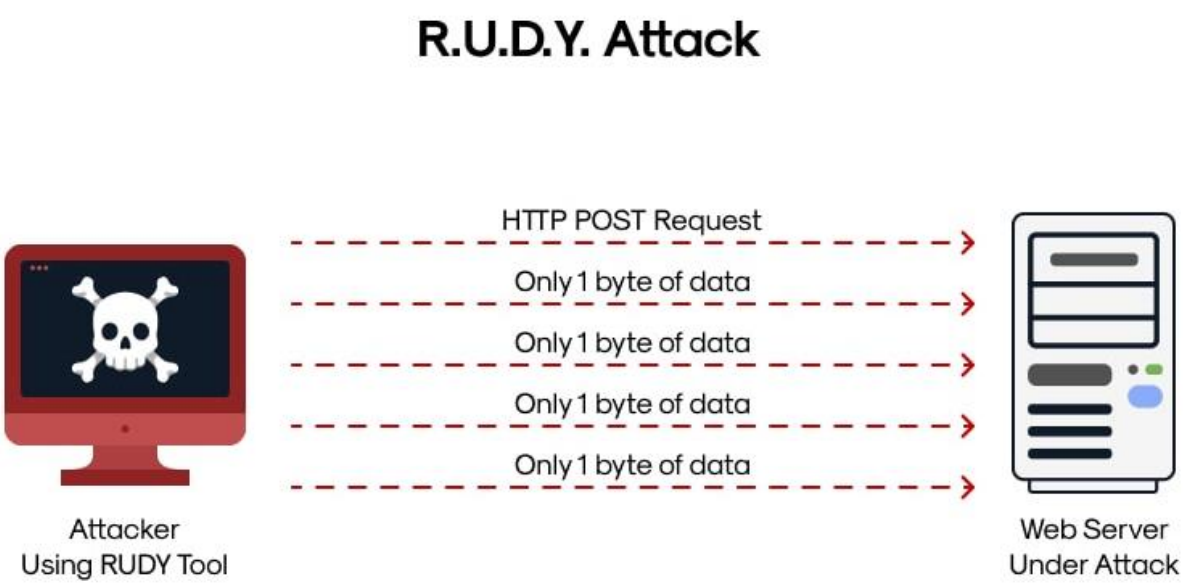

Рисунок 1.4 – Візуалізація атаки R.U.D.Y

Інтеграція штучного інтелекту з розподіленими системами захисту дозволяє створювати масштабовані та ефективні рішення, здатні забезпечити безперервний захист у великих мережевих середовищах. Крім того, міжнародна співпраця та обмін інформацією про нові загрози сприяють більш ефективному реагуванню на глобальні кібератаки.

У відповідь на зростаючі загрози, пов'язані з DoS-атаками, з'являються нові технології та рішення для захисту в реальному часі. Хмарні сервіси та мережі доставки контенту (CDN) використовуються для розподілу трафіку та забезпечення стійкості до навантажень. Інтелектуальні системи аналізують поведінку трафіку, виявляють відхилення від нормальних шаблонів та автоматично застосовують необхідні заходи для запобігання атакам. Використання машинного навчання та глибинного навчання дозволяє створювати більш точні та адаптивні моделі для виявлення нових та невідомих типів атак.

Сучасні підходи до захисту від DoS-атак передбачають інтеграцію багаторівневих систем захисту, які поєднують різні методи та технології для забезпечення комплексного захисту. Це включає використання фільтрації трафіку, систем виявлення вторгнень (IDS), систем запобігання вторгненням (IPS), балансувальників навантаження, а також механізмів аутентифікації та CAPTCHA

для підтвердження легітимності запитів. Такий підхід дозволяє створити багаторівневий захист, де кожен рівень відповідає за певний аспект безпеки, забезпечуючи більш ефективну та стійку систему захисту від DoS-атак.

Однією з нових тенденцій є використання блокчейн-технологій (див. рис. 1.5) для підвищення безпеки та стійкості до DoS-атак. Блокчейн може забезпечити децентралізоване управління трафіком, що ускладнює зловмисникам здійснення масштабних атак на центральні сервери. Крім того, блокчейн може використовуватися для аутентифікації та перевірки легітимності запитів, що зменшує кількість фальшивих запитів та підвищує ефективність захисту.

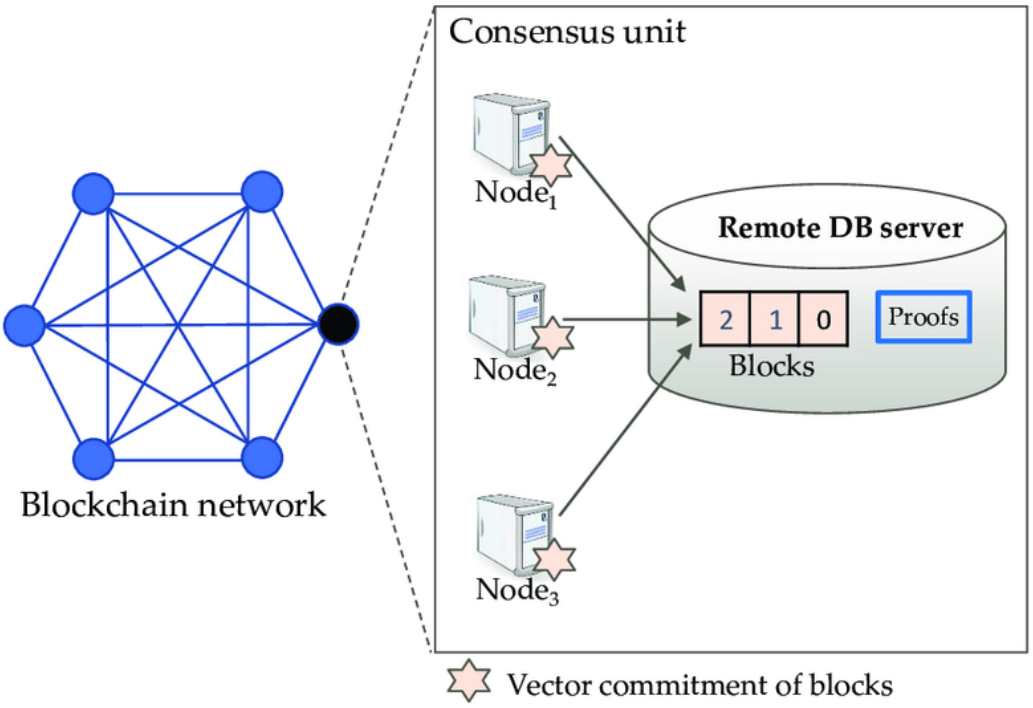

Рисунок 1.5 – Блокчейн-технологія в мережі

Сучасні тенденції в сфері DoS-атак відображають постійну еволюцію методів атак та відповідних засобів захисту. Використання IoT для створення масштабних ботнетів, ампліфікаційні атаки з використанням нових протоколів, автоматизація атак за допомогою AI та ML, а також розвиток хмарних технологій і блокчейн-розробок, створюють нові виклики для систем захисту. Відповідь на ці виклики полягає у впровадженні інтелектуальних, адаптивних та багаторівневих систем

захисту, які поєднують передові технології машинного навчання, глибинного навчання та інших методів аналізу даних. Крім того, важливою є міжнародна співпраця та стандартизація, що дозволяє створювати більш стійкі та ефективні системи захисту на глобальному рівні. Постійний розвиток та вдосконалення методів захисту від DoS-атак є необхідним для забезпечення безпеки інформаційних систем та стабільності сучасних цифрових інфраструктур.

1.3 Аналіз останніх досліджень

1.3.1 Застосування методів глибинного навчання для аналізу мережевого трафіку

Автори обрали для свого дослідження набір даних NSL-KDD [7], який є поліпшеною версією відомого набору даних KDD Cup 99. NSL-KDD усуває деякі недоліки оригінального набору даних, такі як надмірна кількість повторюваних записів, що можуть призвести до перекосу в навчанні моделі. Цей набір даних містить різноманітні типи мережевих атак, включаючи DoS, що робить його придатним для поставленої задачі.

Перед побудовою моделі дані були піддані ретельній попередній обробці. Числові ознаки нормалізувалися для приведення їх до одного масштабу, що сприяє стабільності та швидкості навчання нейронної мережі. Категорійні ознаки, такі як протокол чи тип сервісу, були закодовані методами кодування, зокрема one-hot encoding, для перетворення їх у числовий формат, придатний для обробки нейронною мережею.

Запропонована DNN-модель складається з кількох прихованих шарів, кількість яких визначається експериментально для досягнення оптимальної продуктивності. Кожен прихований шар містить певну кількість нейронів, які застосовують нелінійну активаційну функцію ReLU (Rectified Linear Unit). Використання ReLU сприяє вирішенню проблеми зникання градієнта та прискорює процес навчання.

На виході моделі встановлено шар з софтмакс-активацією, що перетворює виходи нейронів у ймовірності належності до кожного з класів. Це дозволяє моделі здійснювати багато класову класифікацію та визначати тип атаки або відсутність атаки.

Після навчання моделі автори провели серію експериментів для оцінки її ефективності. Результати показали (див. рис. 1.6), що DNN-модель досягає високої точності у виявленні та класифікації DoS-атак. Зокрема, модель демонструє перевагу над традиційними методами машинного навчання, такими як підтримувальні векторні машини, дерева рішень та наївний Байєс. Показники точності, повноти та F1-міри свідчать про значне покращення виявлення атак, особливо тих, що мають низьку частоту появи в наборі даних. Хибно позитивні та хибно негативні спрацювання були суттєво знижені, що є критичним для практичного застосування систем виявлення вторгнень. Високий рівень хибно позитивів може призвести до перевантаження системи безпеки та ігнорування реальних загроз, тоді як хибно негативи означають пропущені атаки. Зниження цих показників свідчить про підвищення надійності та ефективності моделі.

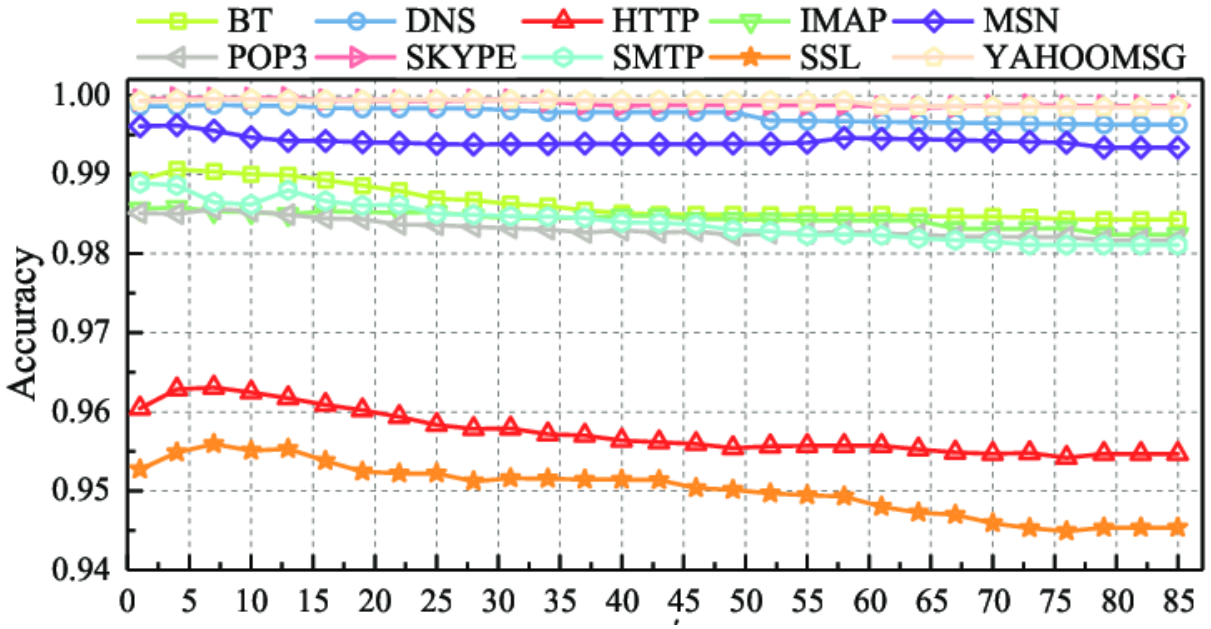

Рисунок 1.6 – Результати навченої моделі на різних типах DoS-атак

Автори відзначають, що успіх DNN у цій задачі пов'язаний зі здатністю глибинних нейронних мереж вловлювати складні нелінійні взаємозв'язки в даних. На відміну від традиційних методів, які можуть неефективно обробляти високо розмірні та складно структуровані дані, DNN здатні автоматично витягати релевантні ознаки та формувати більш точні моделі. Важливим аспектом дослідження є те, що модель демонструє високу здатність до генералізації, тобто вона може ефективно виявляти атаки, які не були явно представлені в навчальних даних. Це особливо важливо в контексті кібербезпеки, де нові та модифіковані атаки постійно з'являються, і системи захисту повинні адаптуватися до них.

Автори визнають, що використання лише одного датасету (NSL-KDD) може обмежувати узагальненість отриманих результатів. Реальні мережеві середовища можуть мати інші характеристики трафіку, що вимагає тестування моделі на різноманітних наборах даних. Крім того, навчання глибинних нейронних мереж вимагає значних обчислювальних ресурсів, що може бути перешкодою для їх впровадження в системи з обмеженими можливостями.

Для подальших досліджень автори пропонують розглянути використання інших архітектур нейронних мереж, таких як згорткові нейронні мережі (Convolutional Neural Networks, CNN) для витягнення просторових ознак, або рекурентні нейронні мережі (Recurrent Neural Networks, RNN) для врахування часових залежностей у трафіку. Також рекомендується проводити експерименти з різними датасетами та в реальних мережевих умовах для перевірки узагальненості та надійності моделі.

Це дослідження є важливим прикладом того, як методи глибинного навчання можуть бути успішно застосовані для виявлення та класифікації DoS-атак. У контексті вашої роботи, ви можете використати підходи та методологію, описані авторами, для розробки власної інтелектуальної системи. Зокрема, варто звернути увагу на попередню обробку даних, вибір архітектури нейронної мережі та методи запобігання перенавчанню.

Враховуючи обмеження, зазначені в дослідженні, ви можете розширити його, використовуючи додаткові датасети, такі як CICIDS2017 або реальні лог-файли серверів, для підвищення узагальненості ваших результатів. Також корисним буде дослідження ефективності інших типів нейронних мереж або гібридних моделей, що поєднують кілька підходів.

Реалізація моделі у реальному часі та її інтеграція в існуючі системи безпеки може стати наступним кроком, що додасть практичної цінності вашій роботі. Таким чином, дослідження Віняякумара та його колег може служити міцною основою для вашого власного дослідження та розвитку в галузі інтелектуальних систем виявлення DoS-атак.

1.3.2 Схема розподіленого виявлення атак із використанням підходу глибокого навчання для Інтернету речей

У статті Діро та Чіламкурті (2018) [8] автори досліджують проблему виявлення кібератак у середовищі Інтернету речей (IoT) (див. рис. 1.7), яке характеризується великим числом підключених пристроїв та високою динамікою трафіку. Зростання кількості IoT-пристроїв створює нові виклики для безпеки, оскільки багато з них мають обмежені ресурси та слабкі механізми захисту, що робить їх вразливими до компрометації та використання у розподілених атаках відмови в обслуговуванні (DDoS). Автори пропонують розподілену схему виявлення атак, що використовує глибинне навчання для аналізу мережевого трафіку та підвищення ефективності виявлення загроз у реальному часі.

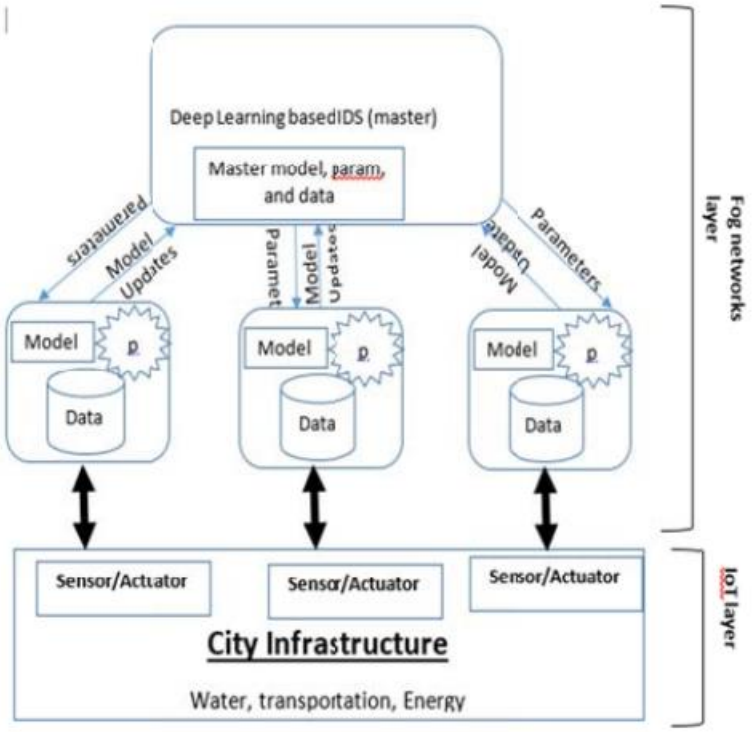

Рисунок 1.7 – Архітектура розподіленого виявлення атак для мереж Fog-to-things

Автори розробили розподілену архітектуру, яка складається з декількох сенсорних вузлів, що збирають дані про мережевий трафік та передають їх до центрального сервера для подальшого аналізу. Для обробки та класифікації даних використовується глибинна нейронна мережа (Deep Neural Network, DNN), яка навчається на великих обсягах даних для розпізнавання патернів, характерних для DDoS-атак. Основні етапи методології включають:

– збір та попередня обробка даних: сенсорні вузли збирають інформацію про мережевий трафік, яка потім нормалізується та кодується для подальшої обробки;

– навчання моделі: використання DNN для навчання на історичних даних про нормальний та атакуючий трафік;

– виявлення атак: модель аналізує поточний трафік у реальному часі та визначає наявність аномалій, що можуть свідчити про атаку.

У експериментальній частині дослідження автори протестували запропоновану схему на реальних даних IoT-трафіку, включаючи різні типи DDoS-атак. Результати показали високу точність виявлення атак з мінімальними хибнопозитивними та хибнонегативними спрацюваннями. Порівняння з традиційними методами виявлення, такими як сигнатурний аналіз та інші алгоритми машинного навчання, продемонструвало переваги використання DNN у плані швидкості та точності. Особливо відзначається здатність моделі ефективно виявляти нові та невідомі типи атак, що є критично важливим у динамічних середовищах IoT.

Автори підкреслюють, що розподілена архітектура дозволяє ефективно масштабувати систему виявлення атак у великих IoT-мережах (див. рис. 1.8). Використання глибинного навчання забезпечує високу точність класифікації та здатність до генералізації, що дозволяє системі адаптуватися до нових загроз без потреби в частому перенавчанні (див. рис. 1.9). Однак, дослідження також вказує на деякі обмеження, такі як вимоги до обчислювальних ресурсів для навчання моделі та необхідність забезпечення безперервного збору та оновлення даних для підтримки актуальності моделі.

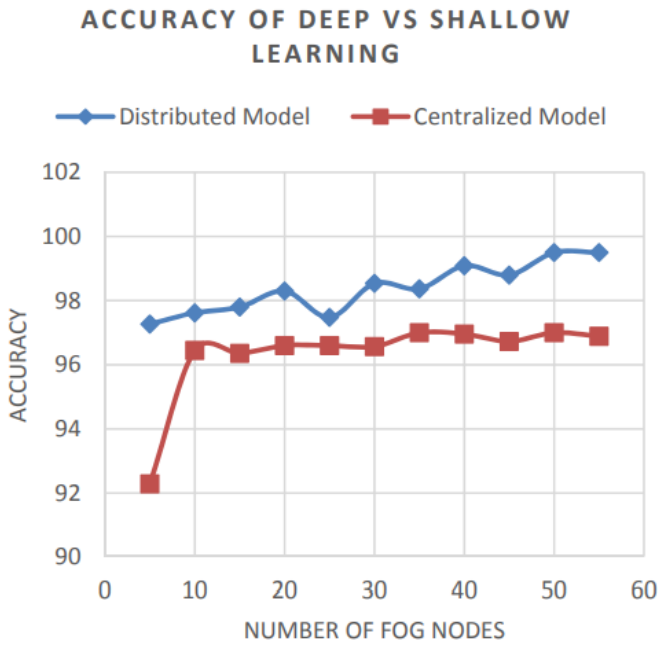

Рисунок 1.8 – Порівняння точності розподілених і централізованих моделей

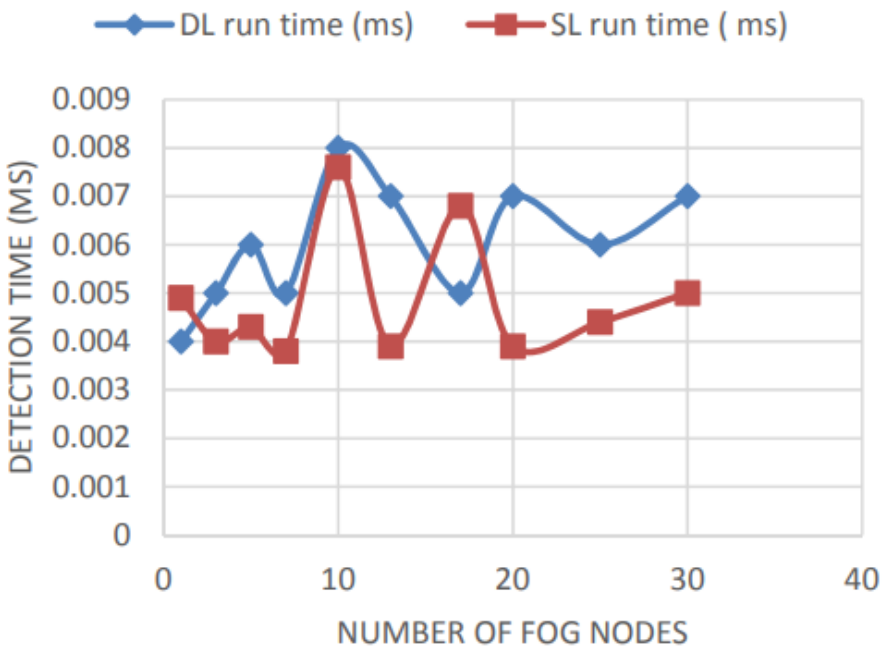

Рисунок 1.9 – Порівняння між DL і SL за часом виявлення

Одним із основних обмежень дослідження є залежність від якості та різноманітності навчальних даних. Для підвищення ефективності моделі необхідно використовувати більш широкий спектр датасетів, що включають різні типи IoT-пристроїв та сценарії атак. Крім того, автори рекомендують дослідження альтернативних архітектур глибинних мереж, таких як рекурентні нейронні мережі (RNN) або згорткові нейронні мережі (CNN), для покращення витягнення ознак та аналізу часових залежностей у трафіку. Важливо також розглянути можливість інтеграції розподіленої системи виявлення з існуючими механізмами безпеки для створення комплексного захисного рішення.

Дослідження Діро та Чіламкурті демонструє ефективність використання глибинного навчання у розподілених системах виявлення DDoS-атак в середовищі IoT. Запропонована схема забезпечує високу точність та адаптивність, що робить її перспективним рішенням для забезпечення безпеки великих IoT-мереж. Водночас, для повного впровадження у реальні системи необхідно подолати обмеження, пов’язані з обчислювальними ресурсами та доступністю якісних навчальних даних.

1.3.3 Новий гібридний метод виявлення вторгнень, що інтегрує виявлення аномалій з виявленням зловживань

У статті Кім, Лі та Кім (2014) [9] автори пропонують новаторський гібридний метод виявлення вторгнень, який поєднує підходи аномального виявлення з методами виявлення зловживань. Основна мета дослідження полягає у покращенні точності та ефективності систем виявлення вторгнень шляхом інтеграції різних методологій аналізу мережевого трафіку. В умовах зростаючої складності та різноманітності кібератак традиційні методи часто не справляються з задачами виявлення нових або змінених типів атак, що підкреслює необхідність використання більш адаптивних та інтелектуальних підходів, таких як рекурентні нейронні мережі (RNN) та їхній розширений варіант — довга короткочасна пам'ять (LSTM).

Автори обрали LSTM як основний інструмент для аналізу послідовностей мережевих пакетів через його здатність ефективно обробляти та запам'ятовувати довготривалі залежності в даних [10]. У своїй роботі вони інтегрували методи аномального виявлення, які базуються на статистичних моделях та машинному навчанні, з методами виявлення зловживань, що ґрунтуються на сигнатурах відомих атак. Такий гібридний підхід дозволяє системі не лише ідентифікувати відомі загрози, але й виявляти нові, невідомі раніше атаки шляхом аналізу відхилень від нормального поведінкового шаблону мережевого трафіку.

Для навчання моделі використовувалися великі набори даних, що містять як нормальний трафік, так і різні типи атак, включаючи DoS-атаки. Перед навчанням дані були піддані ретельній попередній обробці, включаючи нормалізацію та кодування категорійних ознак, що дозволило підготувати їх до ефективної обробки LSTM-моделлю. Архітектура нейронної мережі була налаштована таким чином, щоб забезпечити оптимальне навчання та мінімізувати перенавчання, використовуючи техніки регуляризації та оптимізації. Експериментальні результати демонструють, що запропонований гібридний метод значно покращує точність виявлення вторгнень порівняно з традиційними методами. Використання

LSTM дозволило моделі ефективно аналізувати послідовності пакетів та виявляти аномалії, що свідчить про потенціал цієї технології у сфері кібербезпеки. Особливо відзначається висока точність у виявленні DoS-атак, а також здатність моделі ідентифікувати нові типи атак, які раніше не були відомі системі. Це свідчить про переваги інтеграції методів аномального виявлення з традиційними методами зловживань, що дозволяє створювати більш гнучкі та адаптивні системи захисту.

Автори підкреслюють, що використання LSTM у поєднанні з методами аномального виявлення дозволяє створити більш досконалі системи виявлення вторгнень, здатні ефективно реагувати на динамічні та складні кібератаки. Перевага LSTM полягає у його здатності утримувати інформацію про попередні стани, що є критично важливим для аналізу послідовних даних, таких як мережевий трафік. Це дозволяє моделі виявляти складні шаблони та відхилення, які можуть бути ознаками атаки, що не були відображені у тренувальних даних.

Проте автори також зазначають, що модель має певні обмеження. Зокрема, ефективність LSTM залежить від якості та різноманітності навчальних даних, а також від налаштування гіперпараметрів мережі. Крім того, велика кількість обчислювальних ресурсів, необхідних для навчання глибинних нейронних мереж, може бути перешкодою для впровадження цієї технології в реальні системи з обмеженими ресурсами.

Дослідження Кім, Лі та Кім демонструє значний потенціал використання LSTM у створенні гібридних систем виявлення вторгнень, що поєднують аномальне виявлення з методами зловживань. Запропонований підхід забезпечує високу точність та здатність до генералізації, що робить його перспективним рішенням для сучасних систем кібербезпеки. Водночас, для повного використання потенціалу цієї технології необхідно подолати існуючі обмеження, зокрема шляхом покращення якості навчальних даних та оптимізації архітектури нейронної мережі.

Висновки до розділу 1

У розділі розглянуто сучасні методи виявлення та класифікації DoS-атак, що поєднують традиційні техніки з передовими технологіями штучного інтелекту. Використання глибинного навчання (DNN, CNN, RNN) дозволяє ефективно аналізувати великий обсяг мережевого трафіку та виявляти складні патерни атак. Основні виклики включають високу динаміку атак, обробку великих даних у реальному часі та дисбаланс класів у навчальних даних.

Аналіз досліджень показав ефективність глибинного навчання у виявленні DoS-атак. Розподілені схеми з глибинним навчанням в IoT ефективні, але обмежені можливостями пристроїв і потребують постійного оновлення моделей. Гібридні методи з використанням LSTM підвищують точність і здатність виявляти нові атаки, але потребують оптимізації для реальних умов.

Загалом, еволюція DoS-атак вимагає впровадження інтелектуальних систем ідентифікації та класифікації, здатних адаптивно реагувати на нові загрози. Використання машинного та глибинного навчання є перспективним напрямом для підвищення ефективності виявлення атак і зменшення хибних спрацювань. Наступні розділи зосередяться на аналізі методик класифікації DoS-атак та розробці такої системи та її тестуванні в реальних умовах.

2 АНАЛІЗ ЕФЕКТИВНИХ МЕТОДИК КЛАСИФІКАЦІЇ DOS АТАК

2.1 Огляд методів класифікації

Виявлення та класифікація атак типу DoS/DDoS є критично важливими завданнями в області кібербезпеки. Ефективні методи класифікації дозволяють не лише ідентифікувати атаки, але й визначати їх типи, що сприяє більш точному реагуванню на загрози. Існує широкий спектр методів класифікації, які використовуються для цієї мети, включаючи традиційні алгоритми машинного навчання, сучасні методи глибинного навчання, а також гібридні підходи, що комбінують різні технології для підвищення точності та надійності систем виявлення.

2.1.1 Традиційні методи машинного навчання

Виявлення та класифікація атак є критично важливою задачею в галузі кібербезпеки. Традиційні методи машинного навчання [11, 12], такі як підтримувальні векторні машини (SVM [13]), дерева рішень [14], наївний Байєс [15] та метод K-ближчих сусідів [16], широко використовуються для цієї мети. Кожен з цих методів має свої переваги та недоліки, що визначають їхню ефективність у різних умовах та середовищах. Нижче наведено детальний аналіз та порівняння цих методів (див. табл. 2.1) за кількома ключовими критеріями.

Основні традиційні методи машинного навчання:

- підтримувальні векторні машини (SVM);
- дерева рішень (Decision Trees);
- наївний Байєс (Naive Bayes);
- метод K-ближчих сусідів (KNN).

Критерії порівняння:

- точність класифікації;
- швидкість навчання та прогнозування;
- інтерпретованість;
- обробка високорозмірних даних;

– стійкість до шуму та перенавчання;
– вимоги до обчислювальних ресурсів.

Таблиця 2.1 – Порівняння традиційних методів машинного навчання

Критерій	**SVM**	**Дерева рішень**	**Наївний Байєс**	**KNN**
Точність класифікації	Висока при правильному налаштуванні	Середня-Висока	Середня	Залежить від вибору K, зазвичай середня
Швидкість навчання	Повільна при великих даних	Швидка	Дуже швидка	Повільна при великих даних
Швидкість прогнозування	Швидка	Швидка	Дуже швидка	Повільна при великих даних
Інтерпретованість	Низька ("чорний ящик")	Висока	Висока	Низька
Обробка високорозмірних даних	Ефективна за умови використання ядра	Погано при дуже високій розмірності	Добре, але залежить від припущень	Погано при дуже високій розмірності
Стійкість до шуму та перенавчання	Висока при використанні регуляризації	Середня (схильні до перенавчання)	Середня	Низька (схильні до шуму)
Вимоги до обчислювальних ресурсів	Високі при великих даних	Низькі до середніх	Низькі	Високі при великих даних

Традиційні методи машинного навчання мають різні переваги та недоліки, які роблять їх підходящими для різних сценаріїв виявлення та класифікації DoS-атак. Нижче наведено кілька ключових висновків на основі попереднього аналізу:

– точність та швидкість: SVM та дерева рішень забезпечують високу точність, але SVM може бути повільним при великих наборах даних. Наївний Байєс та KNN є швидкими, але їх точність може бути нижчою або залежною від вибору параметрів;

– інтерпретованість: дерева рішень та наївний Байєс пропонують високу інтерпретованість, що важливо для розуміння та довіри до моделей. SVM та KNN мають низьку інтерпретованість, що ускладнює пояснення їхніх рішень;

– обробка високорозмірних даних: SVM та наївний Байєс добре справляються з високорозмірними даними, тоді як дерева рішень та KNN можуть мати обмеження через складність структури та "прокляття вимірності";

– стійкість до перенавчання: SVM завдяки регуляризації та дерева рішень з методами обрізки можуть бути стійкими до перенавчання. Наївний Байєс та KNN можуть бути більш схильними до перенавчання в залежності від параметрів.

Традиційні методи машинного навчання залишаються важливими інструментами для виявлення та класифікації DoS-атак завдяки своїй простоті, швидкості та ефективності у багатьох задачах. Однак, кожен метод має свої обмеження, які слід враховувати при виборі підходу для конкретної задачі. Для підвищення точності та адаптивності систем виявлення рекомендується комбінувати традиційні методи з сучасними підходами, такими як глибинне навчання та гібридні моделі [17,18], що дозволяє поєднувати переваги різних алгоритмів.

2.1.2 Глибинне навчання

Глибинне навчання є підгалуззю машинного навчання, яка фокусується на використанні багатошарових штучних нейронних мереж [19, 20] для автоматичного витягнення складних ознак та моделювання високорівневих абстракцій з великих обсягів даних. На відміну від традиційних методів машинного навчання, які часто потребують ручного вибору ознак [21], глибинні моделі здатні самостійно навчатися релевантних представлень даних, що робить їх надзвичайно потужними для аналізу складних та нелінійних залежностей.

Існує кілька основних архітектур глибинного навчання, кожна з яких має свої переваги та застосовується в різних контекстах.

Глибокі нейронні мережі (Deep Neural Networks, DNN) складаються з багатьох прихованих шарів між вхідним та вихідним шарами (див. рис. 2.1). Вони здатні моделювати складні нелінійні залежності між ознаками.

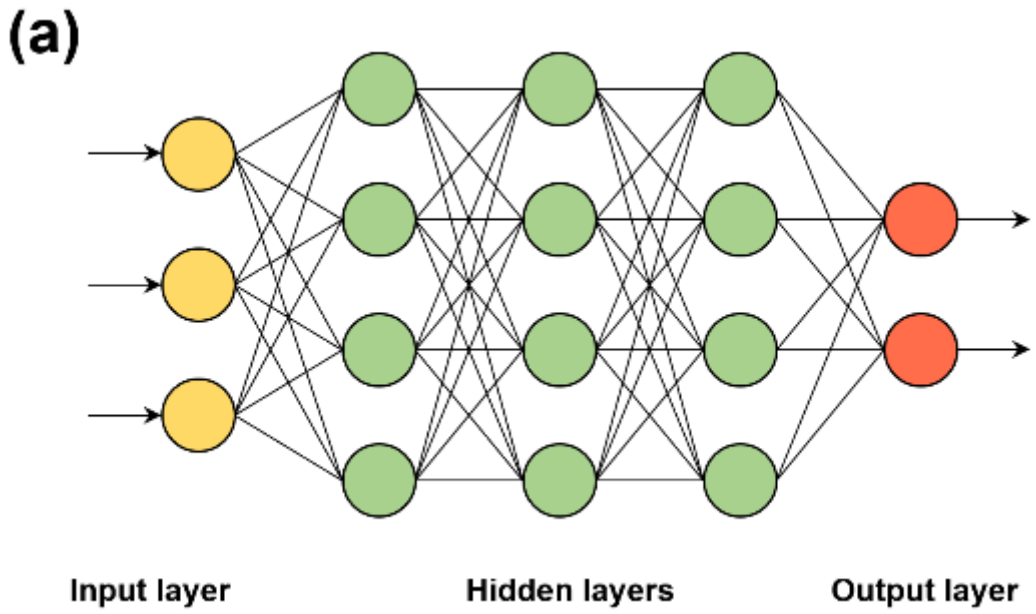

Рисунок 2.1 – Структура глибоких нейронних мереж

Згорткові нейронні мережі (CNN) - це пеціалізовані для обробки даних з певною структурою (див. рис. 2.2), наприклад, зображень або просторових патернів у мережевому трафіку [22]. Використовують згорткові шари для автоматичного витягнення ознак.

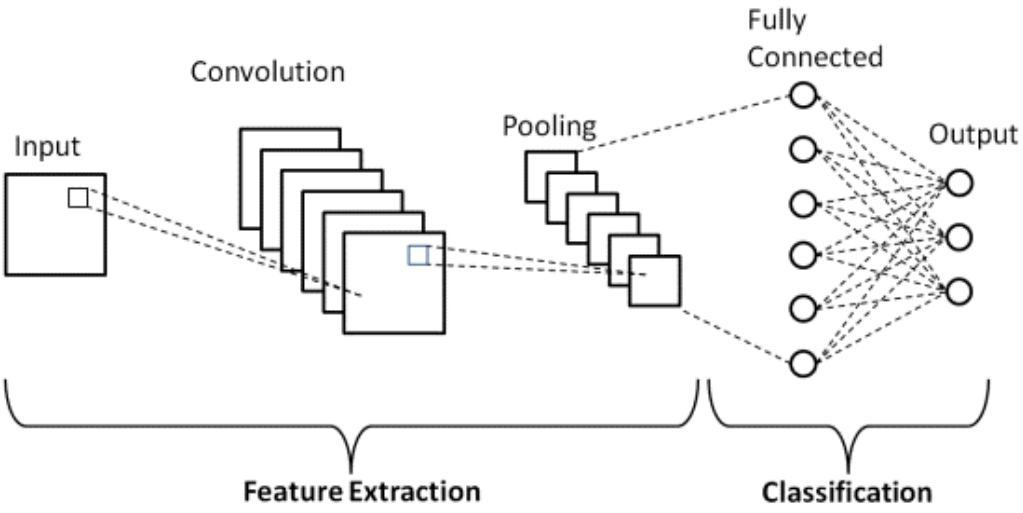

Рисунок 2.2 – Структура згорткових нейронних мереж

Рекурентні нейронні мережі (RNN) орієнтовані на обробку послідовних даних, таких як часові ряди або послідовності мережевих пакетів [23]. Мають зворотні зв'язки (див. рис. 2.3), що дозволяють зберігати інформацію про попередні стани.

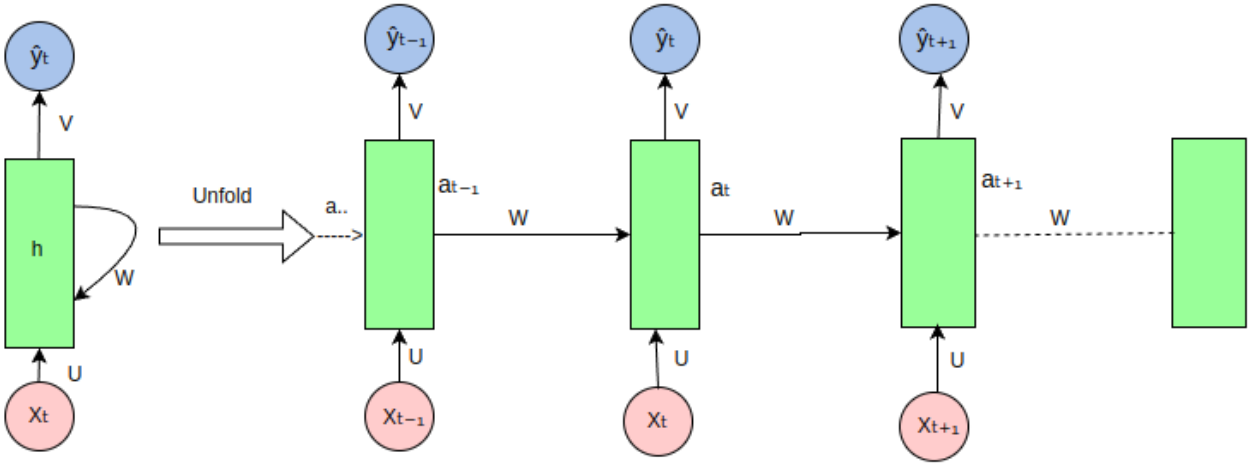

Рисунок 2.3 – Структура рекурентних нейронних мереж

Довга короткочасна пам'ять (Long Short-Term Memory, LSTM) – це розширення RNN, яке вирішує проблему зникання градієнта та дозволяє ефективно моделювати довготривалі залежності (див. рис. 2.4) в послідовних даних [24].

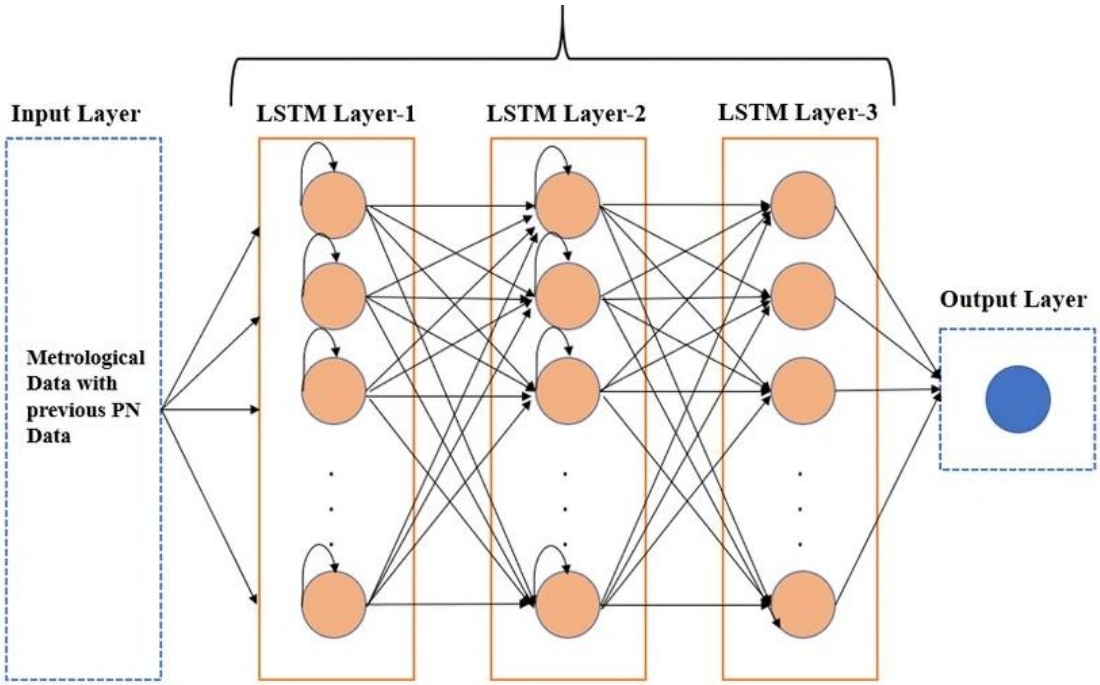

Рисунок 2.4 – Структура LSTM

Глибинне навчання відіграє ключову роль у сучасних системах виявлення та класифікації DoS-атак завдяки своїй здатності аналізувати великі обсяги мережевого трафіку та виявляти складні патерни, що характерні для атак. Основні напрямки застосування включають:

– аналіз поведінки трафіку: глибинні моделі можуть вивчати нормальні патерни мережевого трафіку та ідентифікувати аномалії, які можуть свідчити про DoS-атаки. Наприклад, LSTM можуть аналізувати послідовності пакетів, виявляючи нетипову активність [25];

– автоматичне витягнення ознак: CNN здатні автоматично витягувати релевантні ознаки з сирих даних трафіку, що зменшує потребу в ручному попередньому аналізі та покращує точність класифікації;

– інтеграція з іншими методами: глибинні моделі можуть комбінуватися з традиційними методами машинного навчання або іншими підходами, створюючи гібридні системи, що підвищують загальну ефективність виявлення.

Глибинне навчання пропонує кілька значущих переваг у порівнянні з традиційними методами машинного навчання. Однією з основних переваг є автоматичне навчання ознак, що дозволяє моделям самостійно витягувати складні та релевантні ознаки з великих обсягів даних без потреби в ручному їх виборі. Це значно покращує точність та адаптивність моделей, особливо у випадках, коли дані мають складну структуру або містять нелінійні залежності.

Висока точність є ще однією важливою перевагою глибинних моделей. Завдяки своїй здатності моделювати складні залежності та аналізувати великі обсяги даних, глибинні нейронні мережі демонструють високу точність у класифікації та виявленні аномалій, що робить їх незамінними у сфері кібербезпеки. Крім того, гнучкість архітектур глибинних мереж дозволяє адаптувати їх до різних типів даних та завдань, що робить їх універсальними інструментами для виявлення та класифікації DoS-атак.

Незважаючи на численні переваги, глибинне навчання стикається з кількома викликами та обмеженнями. Одним із основних викликів є велика кількість даних,

необхідних для ефективного навчання глибинних моделей. У випадках обмеженої доступності даних про атаки, особливо нові або невідомі, забезпечення достатнього обсягу тренувальних даних може бути складним завданням. Крім того, навчання глибинних моделей вимагає значних обчислювальних ресурсів, включаючи потужні графічні процесори (GPU), що може бути недосяжним для деяких організацій або дослідницьких груп.

Інтерпретованість є ще одним важливим аспектом, оскільки глибинні моделі часто розглядаються як "чорні ящики". Це ускладнює розуміння механізмів їхніх рішень та довіру до них з боку користувачів та адміністраторів систем безпеки. Крім того, висока складність моделей може призводити до перенавчання, особливо при недостатній кількості даних або надмірній кількості параметрів, що негативно впливає на їхню здатність до генералізації [26].

У сучасних дослідженнях зосереджується на подоланні існуючих обмежень та розширенні можливостей глибинного навчання у сфері кібербезпеки. Оптимізація моделей є одним із ключових напрямків, де розробляються більш ефективні архітектури та алгоритми оптимізації для зменшення обчислювальних витрат та прискорення процесу навчання[27]. Це дозволяє робити глибинні моделі більш доступними для застосування у реальних умовах з обмеженими ресурсами.

Інтерпретованість моделей також є важливим напрямком досліджень. Розробка методів, які дозволяють краще розуміти внутрішню роботу глибинних мереж, сприяє підвищенню довіри до них та їхній прийнятності у критично важливих системах безпеки. Трансферне навчання, яке дозволяє використовувати знання, набуті на одних даних, для інших, також стає все більш популярним, оскільки зменшує потребу в великих обсягах специфічних даних для кожної конкретної задачі.

Інтеграція глибинного навчання з іншими технологіями, такими як блокчейн, Інтернет речей (IoT) та хмарні обчислення, відкриває нові можливості для створення більш стійких та масштабованих систем захисту. Генеративні змагальні мережі (GAN) використовуються для синтезу атакуючого трафіку, що дозволяє

покращити тренувальні набори даних та підвищити здатність моделей до генералізації. Це сприяє створенню більш адаптивних систем, які можуть ефективно реагувати на нові та невідомі типи атак.

Глибинне навчання є потужним інструментом для виявлення та класифікації DoS-атак, завдяки своїй здатності аналізувати великі обсяги даних та виявляти складні патерни атак. Хоча існують певні виклики, такі як потреба в великих обсягах даних та обчислювальних ресурсах, постійний розвиток технологій та методів оптимізації робить глибинне навчання все більш ефективним та доступним для застосування у сфері кібербезпеки [28, 29]. Майбутні дослідження повинні бути спрямовані на подолання існуючих обмежень, підвищення інтерпретованості моделей та інтеграцію з іншими передовими технологіями для створення більш стійких та адаптивних систем захисту від DoS-атак.

2.1.3 Сучасні інновації та тенденції

Однією з ключових інновацій є використання генеративних змагальних мереж (GAN) для синтезу атакуючого трафіку. GAN складаються з двох компонентів: генератора та дискримінатора, які працюють в умовах змагального навчання. Генератор створює фальшиві дані, а дискримінатор намагається відрізнити їх від реальних. У контексті кібербезпеки, GAN можуть бути використані для створення реалістичних симуляцій DoS-атак, що дозволяє розширити тренувальні набори даних без необхідності збору реальних атакуючих сценаріїв, які можуть бути рідкісними або конфіденційними [29, 30, 31]. Це має декілька переваг. По-перше, збільшення обсягу тренувальних даних дозволяє моделям машинного навчання краще узагальнювати та ефективніше виявляти нові, раніше невідомі атаки. По-друге, синтезовані дані можуть бути контрольованими, що дозволяє створювати різноманітні сценарії атак для навчання моделей у відповідь на різні типи загроз. В результаті, системи виявлення стають більш стійкими та адаптивними до атак, що еволюціонують [32].

Іншою важливою інновацією є впровадження методів трансформерів, які спочатку були розроблені для обробки природної мови, у сферу класифікації

мережевого трафіку. Трансформери відзначаються здатністю ефективно працювати з послідовними даними та довготривалими залежностями, що є особливо корисним у аналізі мережевого трафіку, де важливо враховувати контекст та послідовність подій [33].

Основна перевага трансформерів полягає в їхній архітектурі, яка дозволяє моделі одночасно обробляти всі елементи послідовності через механізм уваги (attention mechanism). Це забезпечує більш гнучке та ефективне розпізнавання патернів у даних порівняно з традиційними рекурентними нейронними мережами (RNN) та їхніми варіантами, такими як LSTM [25, 34]. У контексті DoS-атак, трансформери можуть аналізувати складні залежності між різними пакетами даних, виявляючи аномалії та незвичні патерни, що свідчать про початок або розвиток атаки.

Сучасні тенденції також включають застосування ансамблевих методів та гібридних моделей, які поєднують кілька алгоритмів машинного навчання для підвищення точності та надійності виявлення атак. Наприклад, комбінування глибинних нейронних мереж з підтримувальними векторними машинами (SVM) або деревами рішень дозволяє використовувати сильні сторони кожного методу, створюючи більш потужні та стійкі до помилок системи.

Ансамблеві методи, такі як бэггінг (bagging) та бустінг (boosting) (див. рис. 2.5), дозволяють об'єднувати результати декількох моделей для отримання більш стабільних та точних прогнозів. Гібридні підходи можуть також включати інтеграцію моделей машинного навчання з правилами сигнатурного аналізу, що забезпечує комплексний захист від відомих та невідомих типів атак.

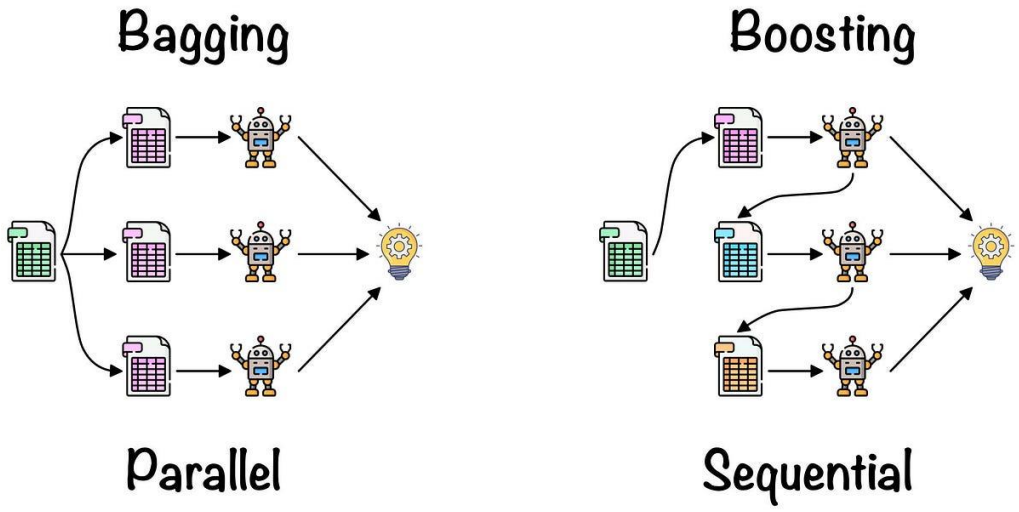

Рисунок 2.5 – Структура методів Bagging та Boosting

Сучасні інновації також спрямовані на підвищення автоматизації та адаптивності систем захисту. Використання методів безперервного навчання (continuous learning) дозволяє системам автоматично оновлювати свої моделі на основі нових даних без необхідності ручного втручання. Це особливо важливо у динамічних середовищах, де загрози постійно змінюються та еволюціонують.

Проактивний підхід до захисту включає прогнозування можливих атак на основі аналізу історичних даних та поточних трендів. Наприклад, системи можуть використовувати алгоритми машинного навчання для прогнозування майбутніх атак на основі поточних патернів трафіку, дозволяючи організаціям вживати превентивні заходи до того, як атака буде здійснена.

Методи трансформерів, завдяки своїй здатності ефективно працювати з послідовними даними та довготривалими залежностями, показують обнадійливі результати у класифікації мережевого трафіку. Ці моделі можуть бути адаптовані для аналізу мережевих пакетів та виявлення аномалій, що свідчать про DoS-атаки. Застосування трансформерів дозволяє створювати більш точні та ефективні системи, здатні до глибокого аналізу складних структур даних

Крім трансформерів, дослідники активно вивчають можливості використання інших сучасних архітектур нейронних мереж, таких як графові

нейронні мережі (GNN) для аналізу взаємозв'язків між різними вузлами в мережі, що дозволяє краще розуміти структуру трафіку та виявляти аномалії на рівні мережевих взаємодій.

Отже, сучасні інновації у сфері глибинного навчання та машинного навчання відкривають нові можливості для ефективного захисту від DoS-атак. Використання генеративних змагальних мереж (GANs) та методів трансформерів дозволяє створювати більш точні, гнучкі та адаптивні системи захисту, які здатні ефективно реагувати на сучасні та майбутні загрози. Гібридні моделі та ансамблеві підходи поєднують переваги різних технологій, забезпечуючи високу точність та надійність виявлення атак. Автоматизація та проактивний захист, забезпечені AI та ML, дозволяють створювати системи, які не лише реагують на вже відомі загрози, але й активно адаптуються до нових, що постійно змінюються [35]. Це робить сучасні методи виявлення та класифікації DoS-атак більш ефективними та надійними, забезпечуючи безпеку інформаційних систем у динамічному та складному цифровому середовищі.

2.2 Аналіз отриманих даних

Для порівняння традиційних методів, глибинного навчання та сучасних створимо порівняльну таблицю (див. табл. 2.2).

Таблиця 2.2 – Порівняння методів машинного та глибинного навчання

Критерій	**Традиційні методи (SVM, Decision Trees, Naive Bayes, KNN)**	**Глибинне навчання (DNN, CNN, LSTM, RNN)**	**Сучасні інновації (GANs, Трансформери, Гібриди)**
Точність класифікації	Середня до високої	Висока	Дуже висока
Швидкість навчання	Швидка (крім KNN)	Повільна (високі вимоги до обчислювальної потужності)	Повільна (особливо для GANs та трансформерів)
Швидкість прогнозування	Швидка (крім KNN)	Середня до швидкої	Середня

Кінець таблиці 2.2

Критерій	**Традиційні методи (SVM, Decision Trees, Naive Bayes, KNN)**	**Глибинне навчання (DNN, CNN, LSTM, RNN)**	**Сучасні інновації (GANs, Трансформери, Гібриди)**
Інтерпретованість	Висока (дерева рішень і наївний Байєс)	Низька ("чорні ящики")	Низька ("чорні ящики")
Стійкість до шуму та перенавчання	Висока (SVM, Naive Bayes)	Залежить від архітектури та налаштувань	Висока (завдяки регуляризації та ансамблевим методам)
Обробка великих даних	Добре, але з обмеженнями (SVM та дерева можуть впоратися з високорозмірними даними, KNN неефективний)	Дуже добре (особливо CNN та LSTM для послідовних даних)	Дуже добре
Адаптивність до нових атак	Обмежена (вимагає оновлення моделей)	Висока	Дуже висока (особливо GANs для генерації нових сценаріїв атак)
Вимоги до обчислювальних ресурсів	Низькі до середніх (більшість моделей можна навчати на CPU)	Високі (потреба в GPU для ефективного навчання)	Дуже високі (GANs і трансформери вимагають значних ресурсів)

Для навчання та перевірки роботи моделей буде використано набір даних CIC-IDS2017 [36]. Набір даних містить великий обсяг мережевого трафіку з різними типами атак, включаючи DoS-атаки. Важливим аспектом для вибору методів є їх здатність працювати з великими обсягами даних, їхня точність та адаптивність до нових типів атак.

Традиційні методи машинного навчання, такі як SVM та дерева рішень, добре підходять для швидкої класифікації, але можуть не мати достатньої точності для складних сценаріїв DoS-атак або великого обсягу даних CIC-IDS2017. Їхня обмежена здатність адаптуватися до нових атак робить їх менш ефективними для довготривалого використання в умовах швидкої еволюції загроз. Однак, ці методи

є дуже корисними для базового аналізу або для побудови початкових моделей з обмеженими ресурсами.

Глибинні нейронні мережі (DNN), згорткові нейронні мережі (CNN) та рекурентні нейронні мережі (RNN, LSTM) можуть значно покращити точність та здатність системи виявляти складні патерни в даних. CNN добре працює з просторовими патернами, тоді як LSTM та RNN ефективно аналізують послідовності даних, такі як мережевий трафік, де часові залежності є ключовими для виявлення аномалій. Використання глибинних моделей дозволяє отримати високу точність класифікації DoS-атак та адаптивність до нових загроз.

Генеративні змагальні мережі (GANs) забезпечують можливість створення синтетичного трафіку, що допомагає покращити навчання моделей та адаптивність до нових атак. Це особливо важливо, якщо реальні дані про нові атаки обмежені. Трансформери, завдяки своїй здатності обробляти послідовні дані з довготривалими залежностями, показують багатообіцяючі результати у класифікації мережевого трафіку. Гібридні моделі, які поєднують CNN і RNN, можуть забезпечити найбільш комплексний підхід до аналізу DoS-атак, використовуючи як просторові, так і часові залежності.

Однак, ці методи вимагають значних обчислювальних ресурсів, особливо для навчання GANs або трансформерів. Тому їх застосування може бути доцільним лише за наявності достатніх ресурсів та при потребі у високоточних моделях.

Висновки до розділу 2

У цьому розділі проведено детальний аналіз ефективних методик класифікації DoS-атак. Розглянуто традиційні методи машинного навчання, такі як SVM, дерева рішень, наївний Байєс та KNN. Виявлено, що ці методи мають певні обмеження у точності та адаптивності, особливо при обробці великих обсягів даних та складних патернів атак.

Глибинне навчання, зокрема DNN, CNN, RNN та LSTM, продемонструвало значні переваги у виявленні та класифікації DoS-атак завдяки здатності

автоматично витягувати складні ознаки та аналізувати нелінійні залежності в даних. Проте його застосування обмежується високими вимогами до обчислювальних ресурсів та потребою у великих наборах навчальних даних.

Сучасні інновації, такі як генеративні змагальні мережі (GANs), трансформери та гібридні моделі, відкривають нові можливості для підвищення точності та адаптивності систем виявлення. Використання GANs дозволяє синтезувати атакуючий трафік для покращення навчання моделей, а трансформери ефективно обробляють послідовні дані з довготривалими залежностями.

Порівняльний аналіз методів показав, що глибинне навчання та сучасні інновації перевершують традиційні методи за точністю та здатністю адаптуватися до нових атак, хоча й вимагають більше ресурсів. Традиційні методи залишаються корисними для базового аналізу та у випадках обмежених ресурсів.

У підсумку, для ефективної класифікації DoS-атак доцільно використовувати глибинне навчання та сучасні інноваційні методи, враховуючи їхні переваги та вимоги. Наступний розділ буде присвячений аналізу та очищенню даних, а також створенню та тестуванню моделей, що базуються на обраних методах, для розробки ефективної системи виявлення DoS-атак.

3 ДОСЛІДЖЕННЯ, ЗБІР, ОБРОБКА ТА СИНТЕТИЧНЕ БАЛАНСУВАННЯ ДАНИХ

3.1 Опис набору даних

Набір даних CIC-IDS2017 був розроблений Канадським інститутом кібербезпеки (Canadian Institute for Cybersecurity) з метою надання дослідникам реалістичного та всебічного набору даних для виявлення та аналізу мережевих атак. Цей набір даних є одним з найпопулярніших для дослідження в області виявлення вторгнень та класифікації атак, зокрема DoS (Denial of Service) атак. Головною метою створення CIC-IDS2017 було подолання недоліків попередніх наборів даних, таких як KDD99 та NSL-KDD, які не відображали сучасних мережевих трафіків та атак. CIC-IDS2017 забезпечує реалістичний трафік, що відображає сучасні мережеві поведінки, включаючи різні типи атак та нормальний трафік. Дані були зібрані протягом п'яти робочих днів у реальному мережевому середовищі. Використовувалась мережа, що складалася з різних пристроїв та операційних систем, з реальними користувачами, які виконували звичайні завдання. Паралельно було генеровано різні типи атак для створення повного та реалістичного набору даних.

Для збору сирих даних використовувались інструменти Wireshark та Tcpdump у форматі PCAP. Файл PCAP (Packet Capture) є бінарним форматом, який зберігає необроблені мережеві дані, отримані через мережевий інтерфейс. Ці файли містять інформацію про кожен пакет (див. рис. 3.1), включаючи заголовки протоколів (TCP/IP, UDP, ARP), розміри пакетів, часові мітки, джерела і призначення трафіку.

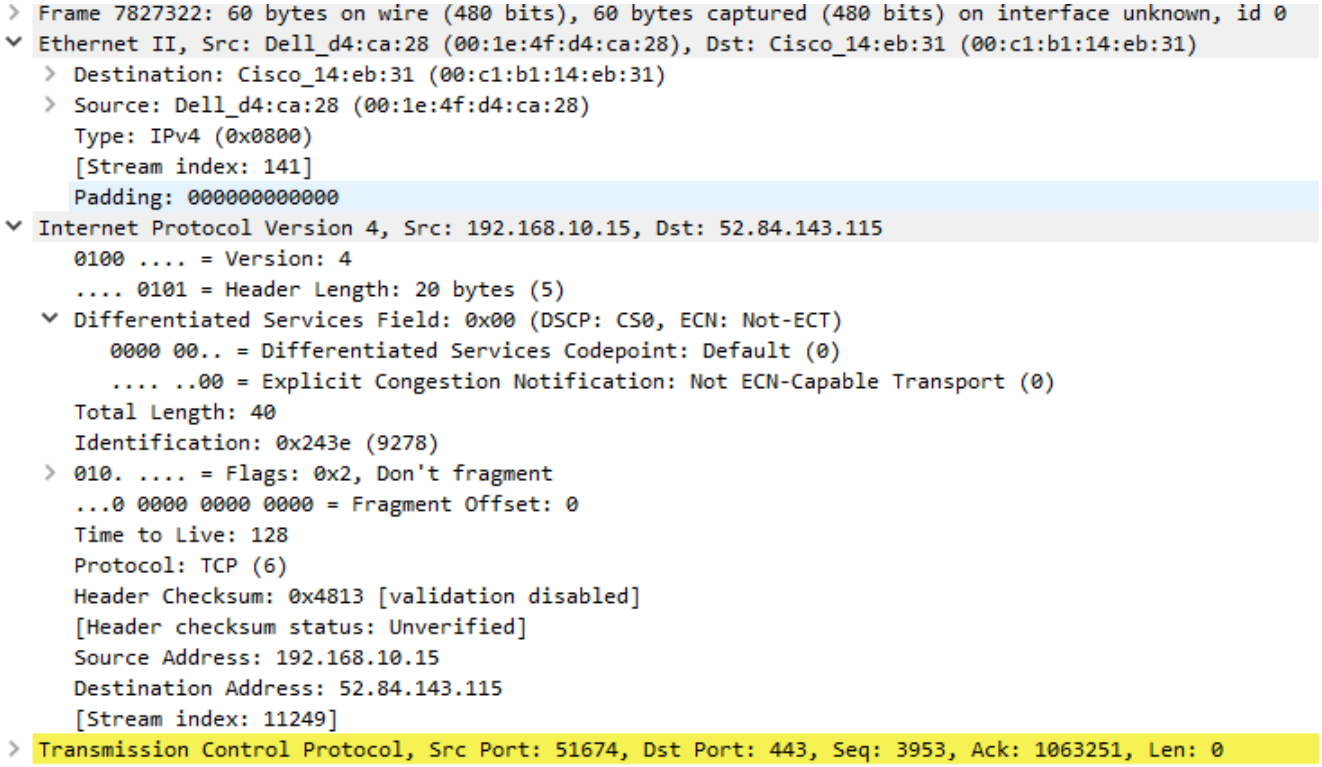

Рисунок 3.1 – Пакет 7827322 у Monday-WorkingHours_2.pcap

Використання PCAP починається з аналізу трафіку за допомогою спеціалізованих інструментів, таких як Wireshark (див. рис. 3.2) або бібліотеки типу pyshark та Scapy. Ці інструменти дозволяють розбирати пакети на структуровані дані. PCAP є дуже гнучким форматом, оскільки з нього можна витягувати будь-які мережеві метрики. Однак це вимагає значних зусиль і технічних знань для вилучення інформації. Головна складність роботи з PCAP у контексті машинного навчання полягає в необхідності перетворення "сирих" даних у корисні ознаки. Наприклад, щоб використовувати CNN+LSTM для класифікації мережевого трафіку, потрібно агрегувати пакети в потоки (flows) і вилучити з них ключові характеристики: кількість пакетів, затримки, розміри тощо. Це значно збільшує час на підготовку даних, але дозволяє отримати повний контроль над тим, які ознаки використовуються.

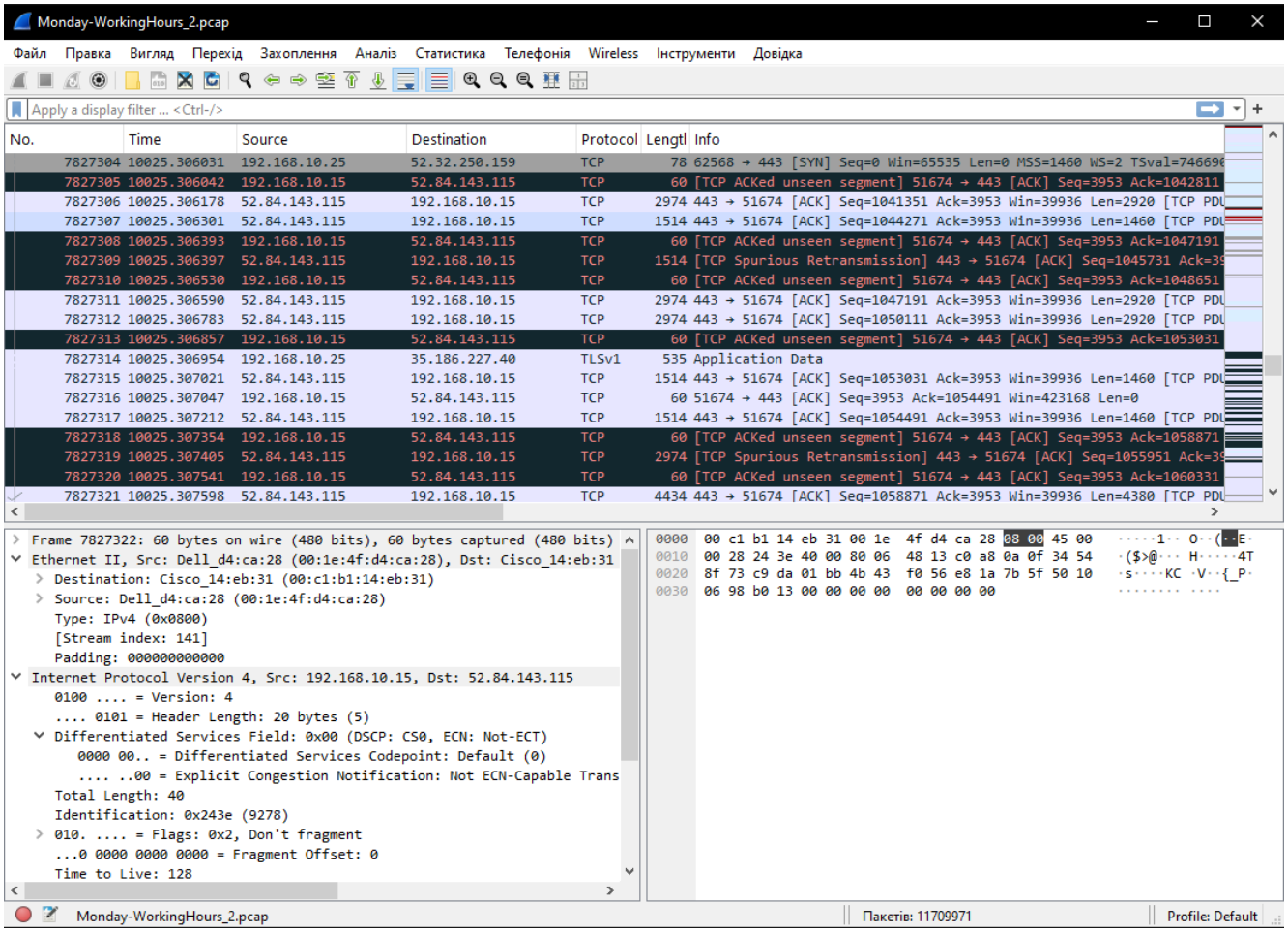

Рисунок 3.2 – Інтерфейс програми Wireshark

Програмне забезпечення Wireshark потребує досить багато ресурсів. При відкриті файлу, весь обсяг відкритих даних записується у оперативну пам'ять ком'ютера. Інтерфейс Wireshark побудований таким чином, щоб максимально полегшити аналіз мережевого трафіку на різних рівнях деталізації. У меню є такі функції: відкриття файлів, налаштування фільтрів, перегляд статистики та інші інструменти для роботи з мережевими даними [37].

Відкривши файл .pcap, можемо бачити, що головний екран розділений на три основні частини. У верхньому блоці відображається список усіх захоплених пакетів. Кожен пакет представлений у вигляді таблиці з ключовими параметрами (див. рис. 3.3), такими як час, джерело, призначення, протокол і додаткова інформація [38].

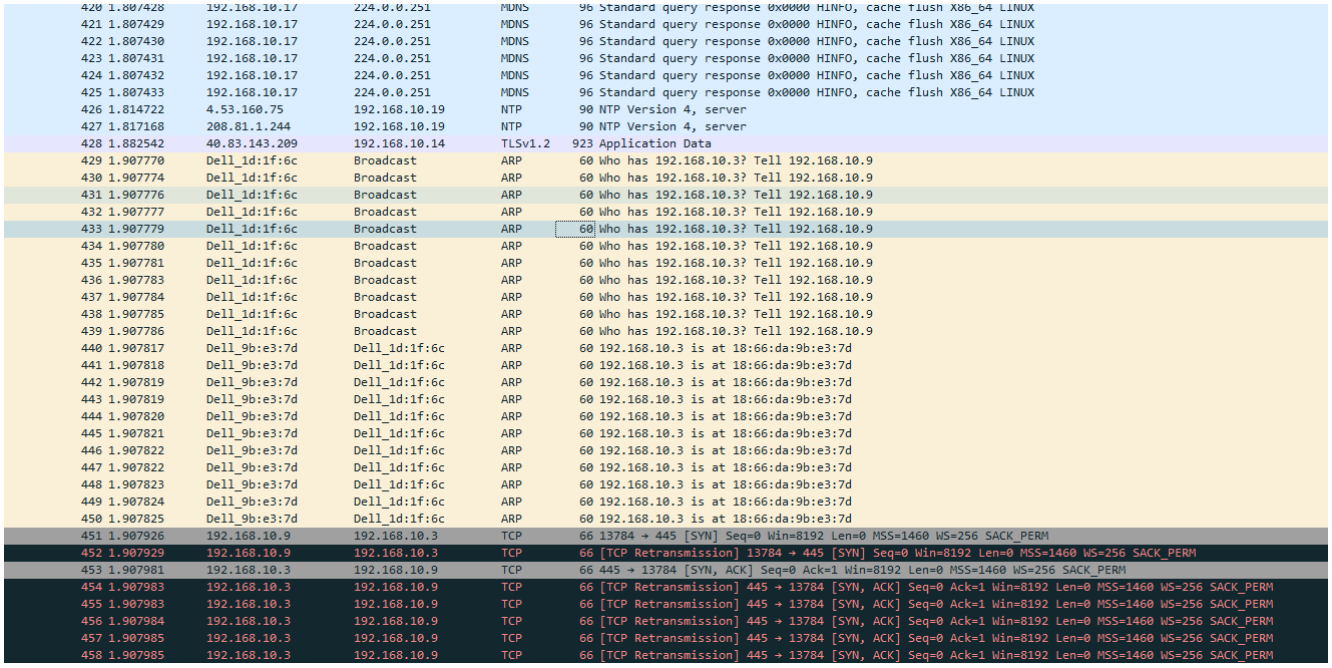

Рисунок 3.3 – Пакети з набору даних Wednesday-workingHours.pcap

Велика кількість ARP-запитів від пристрою Dell_1d:1f:6c, які маю вигляд: «Who has 192.168.10.3? Tell 192.168.10.9», генеруються з високою частотою, що є аномалією та може свідчити про ARP Flood або DoS-атаку (помічено жовтим кольором).

Отже, ця частина дає загальний огляд трафіку і дозволяє вибрати конкретний пакет для детальнішого аналізу. Кольорове кодування пакетів використовується для швидкої ідентифікації їх типу та полегшення аналізу мережевого трафіку. Кожен колір вказує на певний тип протоколу або стан пакета. Пакети фарбуються у наступні кольори:

– зелений: пакети TCP, які передають дані. Зеленим кольором зазвичай виділяються стандартні сеансові пакети;

– синій: використовується для UDP-пакетів. UDP – це менш надійний протокол, часто використовується для стрімінгу або DNS-запитів;

– фіолетовий: зазвичай позначає DNS-запити або інші протоколи прикладного рівня, такі як HTTP;

– сірий: використовується для ICMP-пакетів, наприклад, для команд ping. Такі пакети відображають діагностичні або службові запити;

– жовтий: може вказувати на проблеми в TCP-сесіях, наприклад, повторні передачі, затримки або проблеми із встановленням з'єднання;

– червоний: часто означає серйозні помилки або аномалії, наприклад, пакети з помилками перевірки цілісності чи порушення протоколу.

Нижній лівий блок деталізує інформацію про вибраний пакет. Тут можна побачити, як пакет виглядає на різних мережевих рівнях: від Ethernet і IP до протоколів вищого рівня, таких як TCP, UDP чи HTTPS. Дані представлені у зрозумілому форматі з можливістю розгортання деталей.

Нижній правий блок показує "сирі" дані пакета у вигляді шістнадцяткового і текстового представлення (див. рис. 3.4). Цей блок дозволяє заглибитися у побайтову структуру пакета, що корисно для детального дослідження чи діагностики.

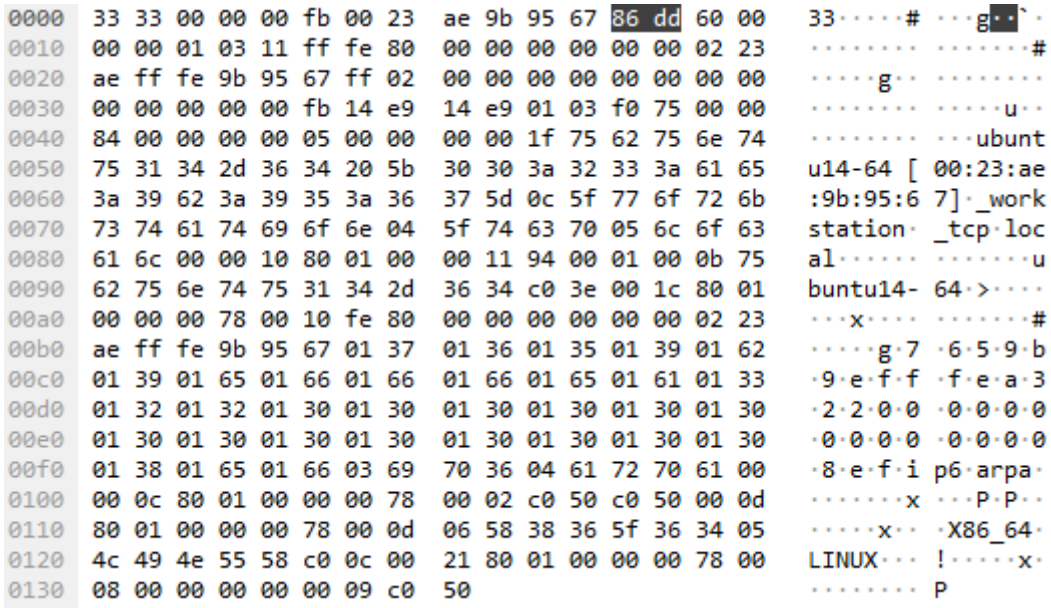

Рисунок 3.4 – «сирі» дані пакету

Таким чином, PCAP підходить для задач, де потрібна максимальна деталізація та аналіз низькорівневих аспектів мережевого трафіку, проте його використання вимагає більшого технічного ресурсу в порівнянні зі структурованими форматами, такими як CSV.

Набір даних містить понад три мільйона записів у CSV файлах, що складаються з нормального трафіку та різних типів атак. PCAP файли містять відповідні сирі дані для кожного дня збору.

CSV файли містять 78 характеристик для колжного мережевого потоку, вкулючаючи:

– статистичні характиристики: середнє значення, медіана стандартне відхилення;

– тимчасові характеристики: тривалість потоку, інтервали між пакетами;

– інформація про протоколи: тип протоколу, порти джерела та призначення;

– характеристика трафіку: кількість байтів та пакетів у напрямку до та від джерела.

Набір даних містить різхні види DoS атак:

– DoS Hulk генерує великий обсях HTTP GET запитів;

– DoS Slowloris утримує відкриті HTTP з’єднання, не дозволяючи серверу їх закрити;

– DoS Slowhttptest використовує низьку швидкість передачі данихз для вичерпання ресурсів сервера;

– DoS Heartbleed експлатує вразливість в OpenSLL.

Для тесту було обрано дані за середу та вечір п’ятниці. У середу було здійснено такі атаки: DoS/DDoS, DoS slowloris, DoS Hulk, DoS Golden Eye. У вечір п’ятниці атака DDoS LOIT.

3.2 Переведення даних з файлів PCAP у CSV

Для переведення даних з файлів PCAP у CSV необхідно завантажити допоміжний інструмент CIFlowMeter [39], який розроблений для аналізу мережевого трафіку та створення потоків на основі PCAP файлів [40]. Дане програмне забезпечення є безкоштовним і завантажити його можна з GitHub.

Посилання на нього можна знайти на сайті, який належить університету Нью-Брансвік. Попередньо там був завантажений набір даних CIC-IDS2017.

CICFlowMeter створює мережеві потоки з PCAP-файлів (див. рис. 3.5), враховуючи такі дані:

– IP-джерела;
– порти;
– протоколи;
– тривалість з’єднання.

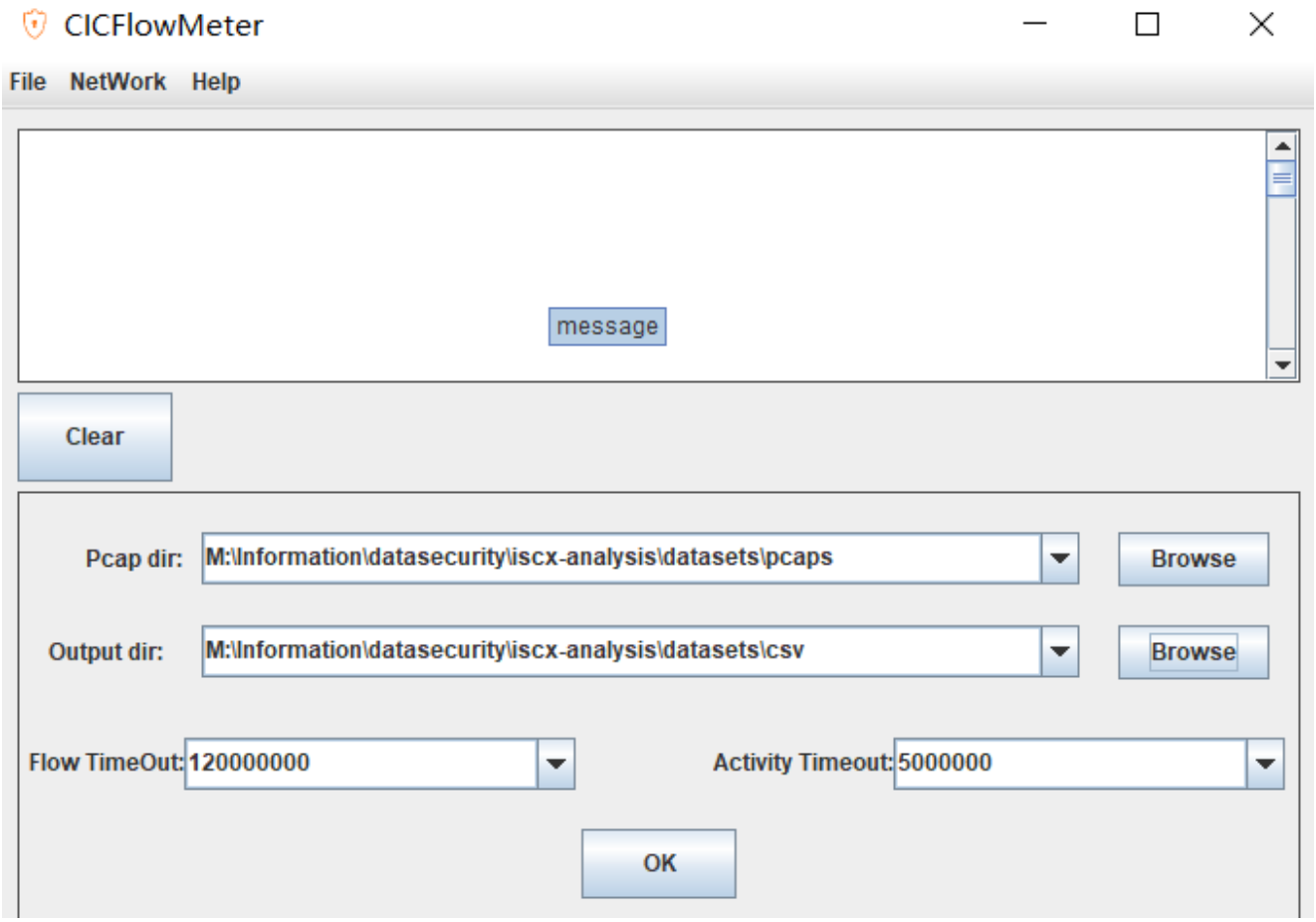

Рисунок 3.5 – Інтерфейс програми CICFlowMeter

Програма досить проста та дозволяє достатньо швидко отримати вихідний CSV файл.

Після перетворення розмір файлу значно зменшився. Розмір набору даних Wednesday-workingHours.pcap_ISCX.csv складає 285,6 МБ, що у 46 разів менше за

розмір лог-файлу. Переглянемо дані за допомогою функції head() бібліотеки pandas (див. рис. 3.6).

```
                              Flow ID      Source IP  ...   Idle Min   Label
0  192.168.10.14-209.48.71.168-49459-80-6  192.168.10.14  ...       0.0  BENIGN
1  192.168.10.3-192.168.10.17-389-49453-6  192.168.10.17  ...       0.0  BENIGN
2   192.168.10.3-192.168.10.17-88-46124-6  192.168.10.17  ...       0.0  BENIGN
3  192.168.10.3-192.168.10.17-389-49454-6  192.168.10.17  ...       0.0  BENIGN
4   192.168.10.3-192.168.10.17-88-46126-6  192.168.10.17  ...       0.0  BENIGN

[5 rows x 85 columns]
(692703, 85)
```

Рисунок 3.6 – Перші 5 значень середи

Можемо побачити тільки декілька параметрів, так як по замовченю в pandas є обмеження на вивід параметрів, для того щоб забезпечити читабельність консолі.

День середа має 692703 спостереження та 85 ознак, одна з яких Label, вказує чи безпечний пакет.

Виведемо всі ознаки для подальшого аналізу (див. рис. 3.7).

```
Index(['Flow ID', ' Source IP', ' Source Port', ' Destination IP',
       ' Destination Port', ' Protocol', ' Timestamp', ' Flow Duration',
       ' Total Fwd Packets', ' Total Backward Packets',
       'Total Length of Fwd Packets', ' Total Length of Bwd Packets',
       ' Fwd Packet Length Max', ' Fwd Packet Length Min',
       ' Fwd Packet Length Mean', ' Fwd Packet Length Std',
       'Bwd Packet Length Max', ' Bwd Packet Length Min',
       ' Bwd Packet Length Mean', ' Bwd Packet Length Std', 'Flow Bytes/s',
       ' Flow Packets/s', ' Flow IAT Mean', ' Flow IAT Std', ' Flow IAT Max',
       ' Flow IAT Min', 'Fwd IAT Total', ' Fwd IAT Mean', ' Fwd IAT Std',
       ' Fwd IAT Max', ' Fwd IAT Min', 'Bwd IAT Total', ' Bwd IAT Mean',
       ' Bwd IAT Std', ' Bwd IAT Max', ' Bwd IAT Min', 'Fwd PSH Flags',
       ' Bwd PSH Flags', ' Fwd URG Flags', ' Bwd URG Flags',
       ' Fwd Header Length', ' Bwd Header Length', 'Fwd Packets/s',
       ' Bwd Packets/s', ' Min Packet Length', ' Max Packet Length',
       ' Packet Length Mean', ' Packet Length Std', ' Packet Length Variance',
       'FIN Flag Count', ' SYN Flag Count', ' RST Flag Count',
       ' PSH Flag Count', ' ACK Flag Count', ' URG Flag Count',
       ' CWE Flag Count', ' ECE Flag Count', ' Down/Up Ratio',
       ' Average Packet Size', ' Avg Fwd Segment Size',
       ' Avg Bwd Segment Size', ' Fwd Header Length.1', 'Fwd Avg Bytes/Bulk',
       ' Fwd Avg Packets/Bulk', ' Fwd Avg Bulk Rate', ' Bwd Avg Bytes/Bulk',
       ' Bwd Avg Packets/Bulk', 'Bwd Avg Bulk Rate', 'Subflow Fwd Packets',
       ' Subflow Fwd Bytes', ' Subflow Bwd Packets', ' Subflow Bwd Bytes',
       'Init_Win_bytes_forward', ' Init_Win_bytes_backward',
       ' act_data_pkt_fwd', ' min_seg_size_forward', 'Active Mean',
       ' Active Std', ' Active Max', ' Active Min', 'Idle Mean', ' Idle Std',
       ' Idle Max', ' Idle Min', ' Label'],
      dtype='object')
```

Рисунок 3.7 – Ознаки набору даних CIC-IDS2017

Проаналізувавши, можна дійсти висновку, що для подальшого аналізу та для класифікації можна зменшити набір даних видаливши такі параметри:

– Flow ID;
– Source IP;
– Destination IP;
– Timestamp.

Всі інші параметри можуть бути стандартизовані або нормалізовані.

3.2 Попередня обробка даних та масштабування

Набори даних містять інформацію про мережевий трафік і мітки класів, які включають як нормальні зразки (BENIGN), так і різні типи атак, такі як DoS Hulk, DoS Slowloris, DoS GoldenEye. Було злиття двох основних файлів (дані п’ятниці та середи), що дозволило об’єднати інформацію про різні атаки [41, 42].

Мітки класів (див. рис. 3.8) у наборі даних, які використовуються для навчання моделі, включають як загальні типи атак (наприклад, DDoS), так і більш специфічні, такі як Heartbleed, DoS Slowloris тощо.

```
Friday labels:
BENIGN
DDoS

Wednesday labels:
BENIGN
DoS slowloris
DoS Slowhttptest
DoS Hulk
DoS GoldenEye
Heartbleed
```

Рисунок 3.8 – Мітки класів

З кількості записів у кожному класів (див. рис. 3.9) видно, що клас BENIGN значно переважає, маючи понад 537 тисяч записів, тоді як класи атак, особливо Heartbleed, мають дуже малу кількість прикладів, лише 11 записів.

Label	
BENIGN	537749
DoS Hulk	231073
DDoS	128027
DoS GoldenEye	10293
DoS slowloris	5796
DoS Slowhttptest	5499
Heartbleed	11

Рисунок 3.9 – Кількість спостережень за класами

Для роботи з такими великими наборами даних, що містять понад 500 тисяч записів, було використано бібліотеки NumPy і Pandas, які дозволяють ефективно аналізувати дані, виконувати їх обробку та забезпечувати високу продуктивність під час масштабних операцій.

У початковому наборі даних виявлено пропущені значення (n/a). Їх присутність може негативно впливати на результати роботи моделі, оскільки моделі, як-от LSTM, чутливі до таких аномалій. Видалення рядків із пропущеними значеннями зменшило кількість записів у наборі, але забезпечило чистоту даних. Кількість об’єктів до та після видалення n/a наведена на рис. 3.10.

```
Розмір DF до видалення пропущених значень:918448
Розмір DF до видалення пропущених значень:917436
```

Рисунок 3.10 – Кількість об’єктів до та після видалення n/a

Під Також необхідно зробити обробку inf та NaN значень, так як є досить багато малих значень. Значення inf, які можуть виникати внаслідок ділення на нуль або інших математичних операцій, було замінено максимальним значенням відповідного стовпця. Це дозволило зберегти інформацію про структуру даних без втрати важливих ознак. Значення NaN було замінено середніми значеннями відповідних стовпців. Використання середнього є ефективним підходом для забезпечення стабільності даних без значного спотворення.

Ці кроки забезпечили усунення будь-яких аномалій у наборі даних, що могло б спричинити помилки під час навчання моделі [43].

Тепер можна перейти до стандартизації, щоб усунути вплив різних масштабів ознак на результати моделі, було застосовано StandardScaler з бібліотеки scikit-learn.

1. Обчислюється середнє значення μ та стандартне відхилення σ кожну ознаку.

2. Масштабує дані шляхом віднімання середнього та ділення на стандартне відхилення.

Формула стандартизації для кожного значення x_i:

$$z_i = \frac{x_i - \mu}{\sigma}, \tag{3.1}$$

де x_i – значення ознаки;

μ – середнє значення;

σ – стандартне відхилення;

z_i – стандартизоване значення.

Обчислення середнього значення:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{3.2}$$

Стандартне відхилення:

$$\sigma = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{3.3}$$

Якщо ж не застосувати стандартизацію, то дані, які мають великі значення, порт чи тривалість з’єднання, можуть непропорційно впливати на результати моделі. Також без стандартизації модель може працювати неефективно або взагалі не сходитися [44].

Стандартизовані дані дозволяють забезпечити швидше та стабільніше навчання моделі. Порівняння даних до та після стандартизації показано на рис. 3.11.

```
Стандартизація даних...
До стандартизації:
    Source Port  Destination Port  Protocol  ...    Idle Std    Idle Max    Idle Min
0       45224.0             443.0       6.0  ...         0.0  58200000.0  58200000.0
1       61987.0              53.0      17.0  ...         0.0         0.0         0.0
2          80.0           20828.0       6.0  ...         0.0         0.0         0.0
3          80.0           65168.0       6.0  ...  44800000.0  72200000.0   8820526.0
4       58996.0             443.0       6.0  ...         0.0         0.0         0.0

[5 rows x 80 columns]
Після стандартизації:
    Source Port  Destination Port  Protocol  ...  Idle Max  Idle Min   Label
0      0.131131         -0.301734 -0.492349  ...  0.819202  0.899024  BENIGN
1      1.020397         -0.327735  2.027585  ... -0.646497 -0.606305  BENIGN
2     -2.263729          1.057349 -0.492349  ... -0.646497 -0.606305  BENIGN
3     -2.263729          4.013529 -0.492349  ...  1.171775 -0.378165  BENIGN
4      0.861726         -0.301734 -0.492349  ... -0.646497 -0.606305  BENIGN
```

Рисунок 3.11 – Стандартизація даних

Через значний дисбаланс у даних моделі можуть демонструвати упередженість на користь класу BENIGN, що становить понад 537 тисяч записів, у порівнянні з лише 11 записами класу Heartbleed. Для вирішення цієї проблеми було застосовано два методи балансування:

– RandomOverSampler виконує просте дублювання зразків з міноритарного класу до досягнення рівності кількості записів для кожного класу. Хоча це дозволяє швидко вирівняти класи, дублювання може призводити до перенавчання моделі;

– SMOTE генерує синтетичні зразки для міноритарних класів на основі інтерполяції між існуючими даними. Це створює нові, реалістичні зразки, що допомагає уникнути проблеми перенавчання. Метод забезпечує краще узагальнення моделі за рахунок введення більшої варіативності даних.

Розберемо покроково, як працює SMOTE [45].

1. Визначається кількість зразків, яку необхідно створити для кожного класу.

2. Для кожного зразка меншості класу знаходяться кількість кластерів, методом k найближчих сусідів серед зразків меншого класу.

3. Для кожного зразка меншого класу генеруються синтетичні зразки за формулою:

$$synthetic\_sample = x_i + \lambda \times (x_{nn} - x_i)\ , \qquad (3.4)$$

де x_i – вектор ознак поточного зразка меншості класу;

x_{nn} – вектор ознак з його k найближчих сусідів;

λ – випадкове число в діапазоні від 0 до 1.

```
Розподіл класів у тренувальному наборі до балансування:
 BENIGN             537686
DoS Hulk            230124
DDoS                128027
DoS GoldenEye        10293
DoS slowloris         5796
DoS Slowhttptest      5499
Heartbleed              11
```

Рисунок 3.12 – Розподіл класів до балансування

Для повторювального результату в цих методах присутній такий параметр як random_state. Він використовується для контролю за генерацією випадкових чисел при створенні синтетичних зразків, що забезпечує відтворювальний результат при наступних запусках скрипта. Це дозволить порівнювати різні методи навчання, балансування та нормалізації.

Після застосування цих методів вдалося вирівняти кількість записів у всіх класах, що показано на рисунку 6. Кожен клас тепер має рівну кількість записів як у навчальному, так і в тестовому наборах, що забезпечує коректність і точність моделі.

Тепер можна виконати розділення на навчальну та тестову вибірку, для того, щоб оцінити продуктивність моделі на невідомих даних та запобігти перенавчанню.

Для того, щоб не тягнути нову бібліотеку, скористаємось методом train_test_split з бібліотеки scikit-learn. Цей метод є одним із найпоширеніших.

Співвідношення вибірки було обрано 80% тренувальних та 20% тестових даних. Створимо два дата фрейми та збережімо їх. У CSV форматі (див. рис. 3.13).

```
y_train_balanced size: (3011041,) and BENIGN             430149
DoS Slowhttptest    430149
DoS slowloris       430149
DDoS                430149
DoS GoldenEye       430149
DoS Hulk            430148
Heartbleed          430148
Name: count, dtype: int64
y_test_balanced size: (752761,) and Heartbleed         107538
DoS Hulk            107538
BENIGN              107537
DDoS                107537
DoS Slowhttptest    107537
DoS slowloris       107537
DoS GoldenEye       107537
```

Рисунок 3.13 – Розподілення класів після балансування

Отже, в результаті маємо 3 011 041 тренувальних спостережень та 752761 спостереження у тестовому наборі.

Висновок до розділу 3

У третьому розділі було проведено детальне дослідження, збір, обробку та синтетичне балансування даних для побудови моделі виявлення DoS-атак. Використано набір даних CIC-IDS2017, розроблений Канадським інститутом кібербезпеки. Це набір нормального мережевого трафіку, включаючи різні типи атак.

Обрано мережевий трафік за середу та п’ятницю. Було завантажено та оброблено PCAP-файли за допомогою інструменту CICFlowMeter, який перетворив сирі мережеві пакети у структуровані дані формату CSV.

Під час очищення даних, було видалено непотрібні ознаки: Flow ID, Source IP, Destination IP, Timestamp, так як вони не несуть корисної інформації для моделі та можуть стати причиною перенавчання. Також виявлено та видалено пропущені значення. Значення inf та NaN замінені на максимальні та середні значення відповідно.

Виконано стандартизацію даних та створено 4 набори даних: два тестових та два навчальних. Для однієї пари було застосовано балансування RandomOverSampler, а інший за допомогою SMOTE.

Дані розділені у співвідношенні 80% тренувальних та 20% тестових даних.

Отже, проведена робота у третьому розділі забезпечила хорошу основу для побудови ефективної моделі виявлення DoS-атак. Завдяки ретельній обробці та балансуванню даних, модель матиме кращу здатність виявляти різні типи атак у мережевому трафіку ніж навчена модель на сирих даних.

4 СТВОРЕННЯ ТА АНАЛІЗ ЕФЕКТИВНОСТІ МОДЕЛЕЙ CNN ТА LSTM У ЗАДАЧІ КЛАСИФІКАЦІЇ МЕРЕЖЕВОГО ТРАФІКУ

4.1 Модель CNN

Для створення та тренування CNN будемо використовувати Keras. Це велика бібліотека, яка також дозволить проводити більшість маніпуляцій з нейромережами не додаючи нових залежностей.

```
from keras import Sequential
from keras.src.callbacks import EarlyStopping
from keras.src.layers import Conv1D, MaxPooling1D, LSTM, Dropout, Dense
from keras.src.utils import plot_model
```

Рисунок 4.1 – Імпорти бібліотеки Keras

Дані мають бути у тривимірному форматі, який складається:

– samples: кількість спостережень (3 011 041 тренувальних);
– timesteps: кількість часових кроків (1);
– features: кількість ознак.

Згорткова нейрона мережа має декілька параметрів, які можна змінювати. Одним з таких є параметр filters, що визначає кількість ядер згортки, які модель буде навчати. Кожне ядро вчиться виявляти певні закономірності в даних. Більша кількість фільтрів дозволяє моделі виявляти більш складні та різноманітні ознаки. Водночас збільшення кількості ядер збільшує обчислювальну складність.

Дуже важливим параметром є функція активації. Було обрано функція активації relu, що задає функцію активації Rectified Linear Unit. Вона вводить нелінійність у модель, дозволяючи їй навчати складніші залежності. ReLU допомагає вирішити проблему зникаючого градієнта та прискорює процес навчання.

Шар Dense приймає на вхід одновимірний вектор. В цьому шарі кожен нейрон з'єднаний з усіма нейронами попереднього шару. Він використовується для узагальнення ознак, виявлених попередніми шарами, і виконання кінцевої

класифікації. Кількість нейронів у Dense шарі впливає на здатність моделі навчатися складним взаємозв'язкам. Потрібно балансувати між потужністю моделі та ризиком перенавчання.

Одним з найважливіших параметрів є Dropout, що визначає яка кількість параметрів буде вимкнена. Це необхідно добре налаштовувати та балансувати, особливо коли модель схильна до перенавчання.

Створимо модель з 2 шарами CNN (див. рис. 4.2) та порівняємо дві моделі з різною кількістю нейронів. У першому згортковому шарі 64 ядер згортки(filters) (див. рис. 4.3, 4.4), а у другому 128 (див. рис. 4.5, 4.6), з параметром dropout = 0.5.

```
model = Sequential()
model.add(Conv1D(filters=64, kernel_size=1, activation='relu', input_shape=(timesteps, features)))
model.add(MaxPooling1D(pool_size=1))
model.add(Conv1D(filters=64, kernel_size=1, activation='relu', input_shape=(timesteps, features)))
model.add(MaxPooling1D(pool_size=1))
model.add(Flatten())
model.add(Dense( units: 128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(np.unique(y_train)), activation='softmax'))
```

Рисунок 4.2 – CNN модель з двома шарами

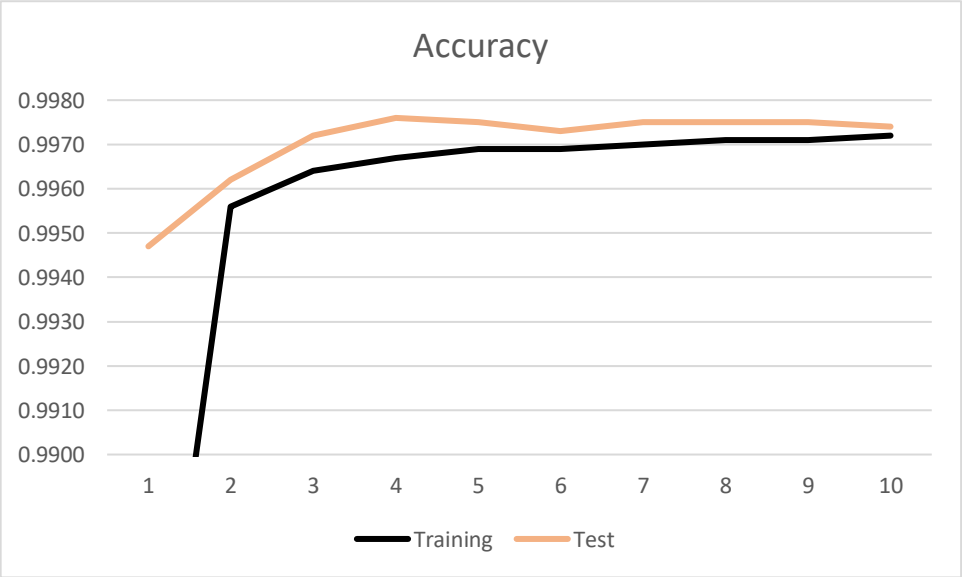

Рисунок 4.3 – Валідація моделі

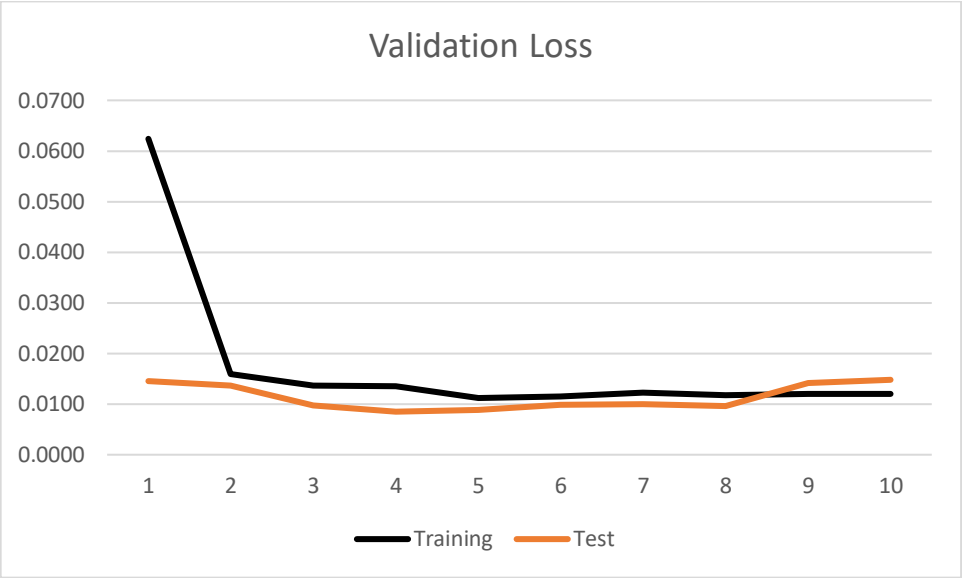

Рисунок 4.4 – Функція втрат

Проаналізуваши графіки можна дійсти висновку, що модель схильна до перенавчання. Хоча точність дость висока, та має позитивний тренд з 1 по 7 епоху, але в наступних епохах збільшується помилка моделі. Це може свідчити про перенавчання. Змінемо параметр Dropout на 0.1, та порівняємо моделі (див. табл. 4.1).

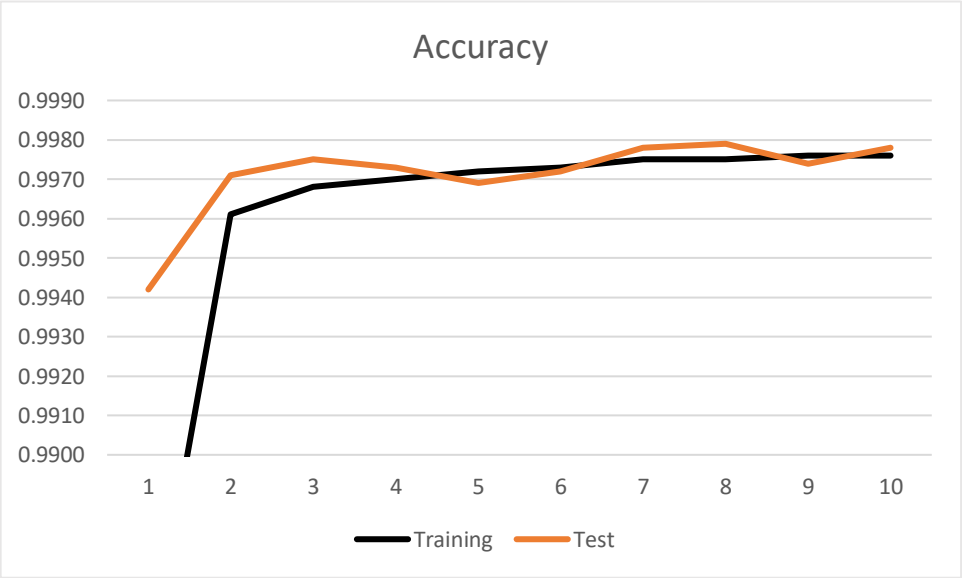

Рисунок 4.5 – Валідація моделі (dropout 0.1)

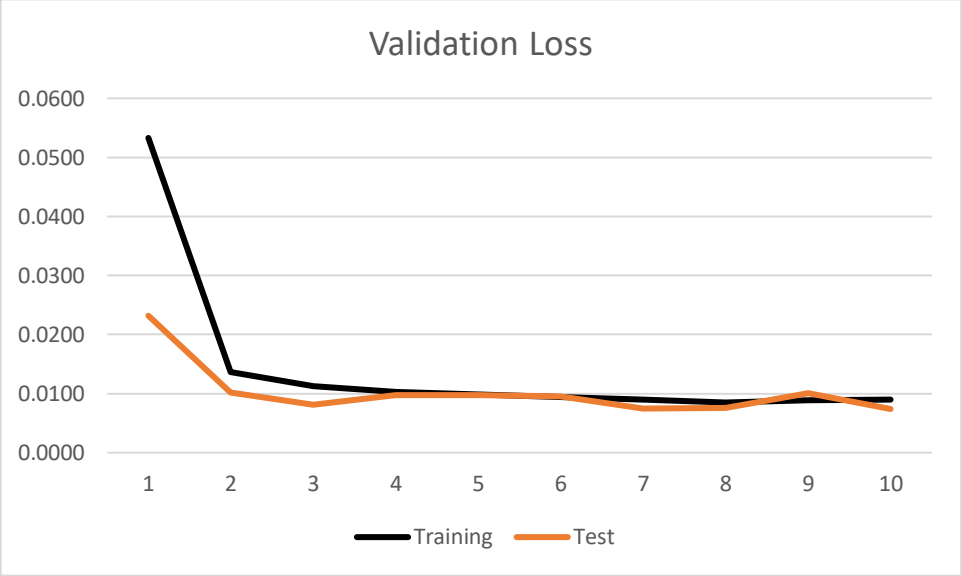

Рисунок 4.6 – Функція втрат (dropout 0.1)

Таблиця 4.1 – Порівняльна таблиця моделей з dropout 0,5 та 0,1

Епоха	Точність M1	Точність M2	Втрати M1	Втрати M2
1	0,9947	0,9942	0,0145	0,0232
2	0,9962	0,9971	0,0136	0,0102
3	0,9972	0,9975	0,0097	0,0081
4	0,9976	0,9973	0,0085	0,0097
5	0,9975	0,9969	0,0089	0,0097
6	0,9973	0,9972	0,0098	0,0095
7	0,9975	0,9978	0,0100	0,0075
8	0,9975	0,9979	0,0096	0,0076
9	0,9975	0,9974	0,0141	0,0101
10	0,9974	0,9978	0,0148	0,0074

Після зменшення значення Dropout до 0.1, точність моделі залишається високою, та немає сильної розбіжності в результатах. Зменшення dropout не вплинуло негативно на результати.

Проаналізувавши графік валідаційних втрат [46] можна підсумувати, що залишаються низькими та зменшується протягом всіх епох, хоча присутні невеличкі коливання.

Зменшення dropout до 0,1 не спричинило значного перенавчання, а навпаки показує стабільний тренд.

Отже, зменшення регуляризації було оптимальним для цієї архітектури моделі.

Перевіримо, як вплине зменшення та збільшення згорткових шарів на точність моделі. Для цього було навчено три моделі з однаковими параметрами, але з різною кількістю шарів.

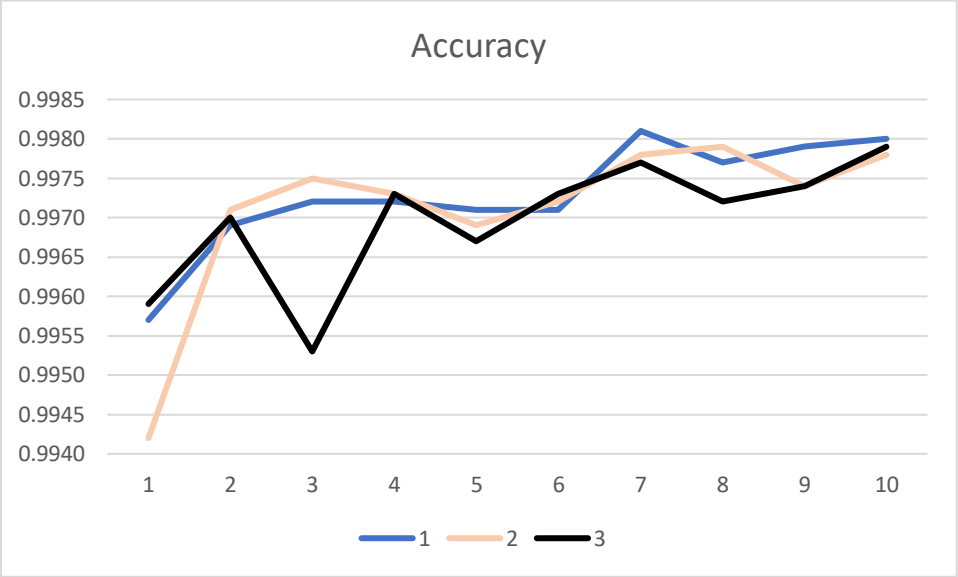

Рисунок 4.7 – Точність CNN моделей з різною кількістю шарів

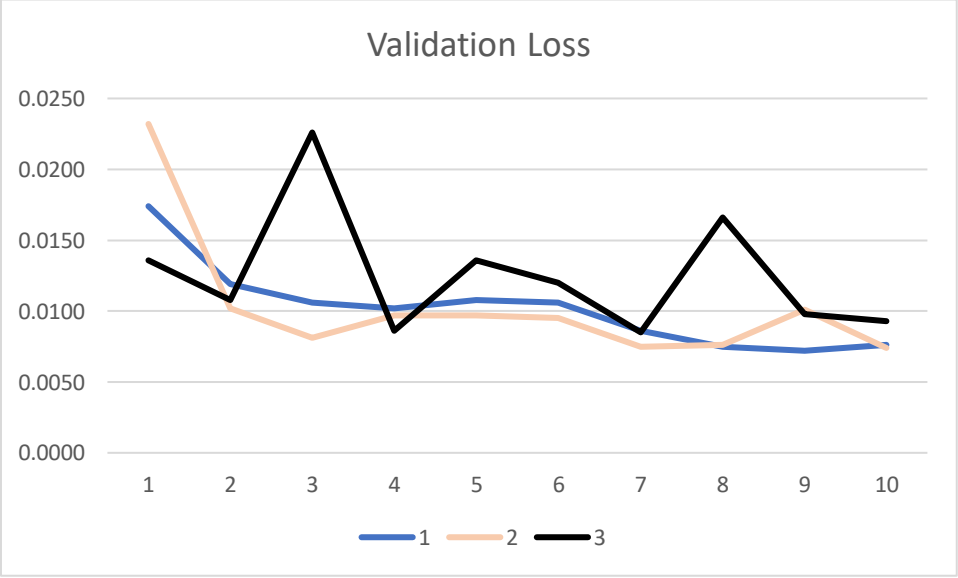

Рисунок 4.8 – Функція втрат CNN моделей з різною кількістю шарів

На рисунках 4.7, 4.8 видно, що модель із 1 згортковим шаром показала стабільний результат як у точності, так і у втрат, демонструючи відсутність перенавчання. У порівнянні, модель із 2 шарами забезпечує трохи кращу точність і

нижчі втрати на тестових даних, зберігаючи баланс між продуктивністю та узагальнюючою здатністю. Проте модель із 3 шарами починає демонструвати ознаки перенавчання, хоча точність на тренувальних даних зростає, на тестових вона супроводжується коливаннями втрат і незначним розривом між тестовими та тренувальними показниками. Таким чином, додаткові шари, хоч і покращують здатність виявляти складні патерни, можуть негативно впливати на узагальнення при недостатньому обсязі даних або слабкій регуляризації.

4.2 Модель LSTM

Для створення та керування використаємо бібліотеку Keras, що було використана раніше, для створення моделей CNN.

Моделі LSTM є типом рекурентних нейронних мереж. Ці моделі здатні запам’ятовувати довготривалі залежності у послідовних даних, що є перевагою у аналізі тимчасових патернів у мережевому трафіку.

Створімо модель з двома шарами LSTM та одного шару для класифікації. Характеристики моделі наступні:

а) перший шар LSTM:

1) 150 нейронів;

2) ‘tanh’ активація;

б) другий шар LSTM:

1) 100 нейронів;

2) ‘tanh’ активація;

в) коефіцієнт виключення 0.2:

г) активація вихідного шару – softmax;

д) кількість епох 10.

```
model = Sequential()
model.add(LSTM( units: 150, activation='tanh',
               return_sequences=True,
               input_shape=(timesteps, features)))
model.add(LSTM( units: 100, activation='tanh'))
model.add(Dropout(0.2))
model.add(Dense(len(np.unique(y_train)), activation='sigmoid'))

model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
history = model.fit(x_train, y_train, epochs=10, batch_size=2048, validation_data=(x_test, y_test))
```

Рисунок 4.9 – Побудова моделі LSTM з двома шарами

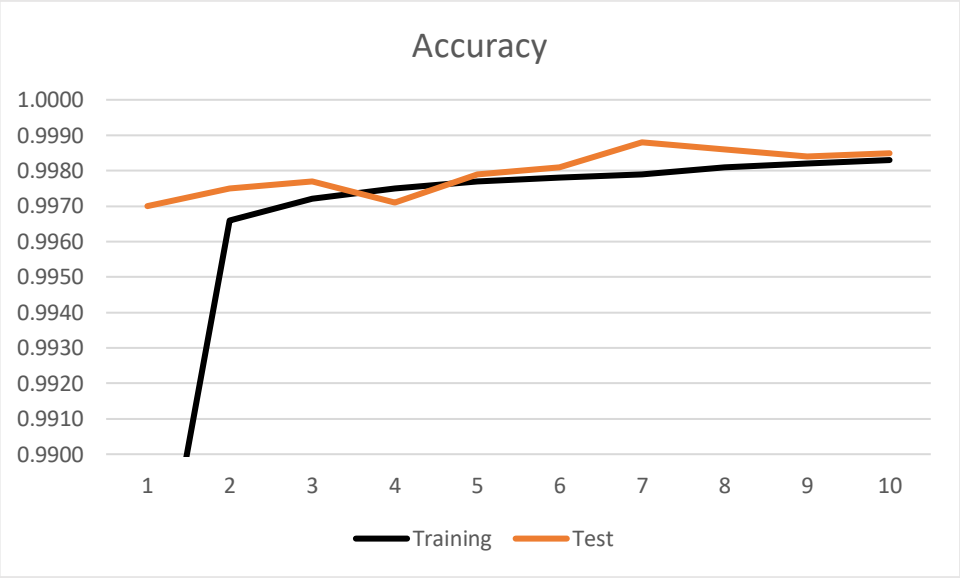

Рисунок 4.10 – Точність моделі з двома шарами LSTM

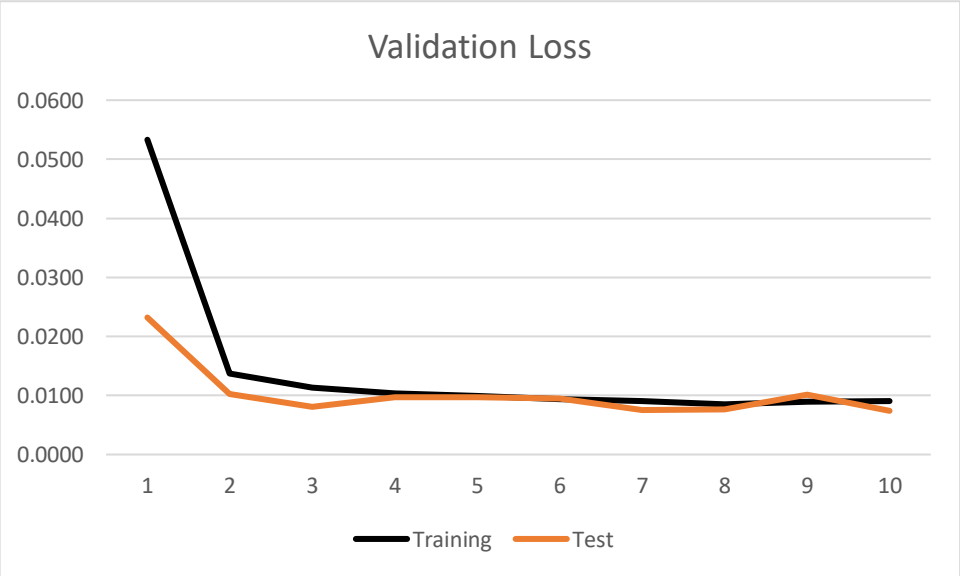

Рисунок 4.11 – Втрати моделі з двома шарами LSTM

Проаналізувавши графіки на рисунках 4.10-4.11, можна дійти висновку, що точність моделі зростає стабільно протягом усіх епох. Відстуність різких стрибків свідчить про поступове навчання без явних ознак перенавчання. Невеликі відхилення характерні для LSTM. Додамо ще один LSTM шар, з 50 нейронами.

Характеристики моделі наступні:

а) перший шар LSTM:

1) 150 нейронів;

2) ‘tanh’ активація;

б) другий шар LSTM:

1) 100 нейронів;

2) ‘tanh’ активація;

в) третій шар LSTM:

1) 50 нейронів;

2) ‘tanh’ активація;

в) коефіцієнт виключення 0.2:

г) активація вихідного шару – softmax;

д) кількість епох 10.

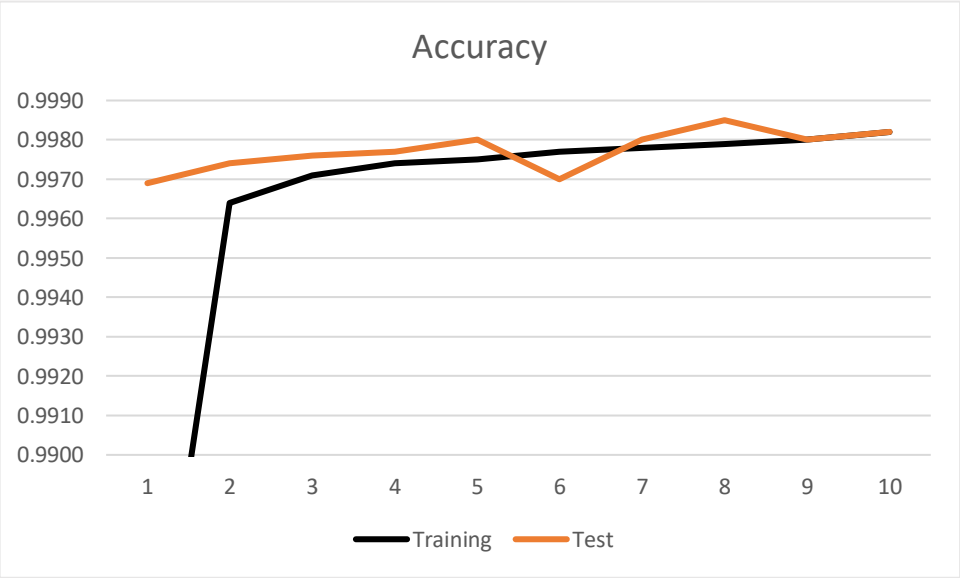

Рисунок 4.12 – Точність моделі з трьома шарами LSTM

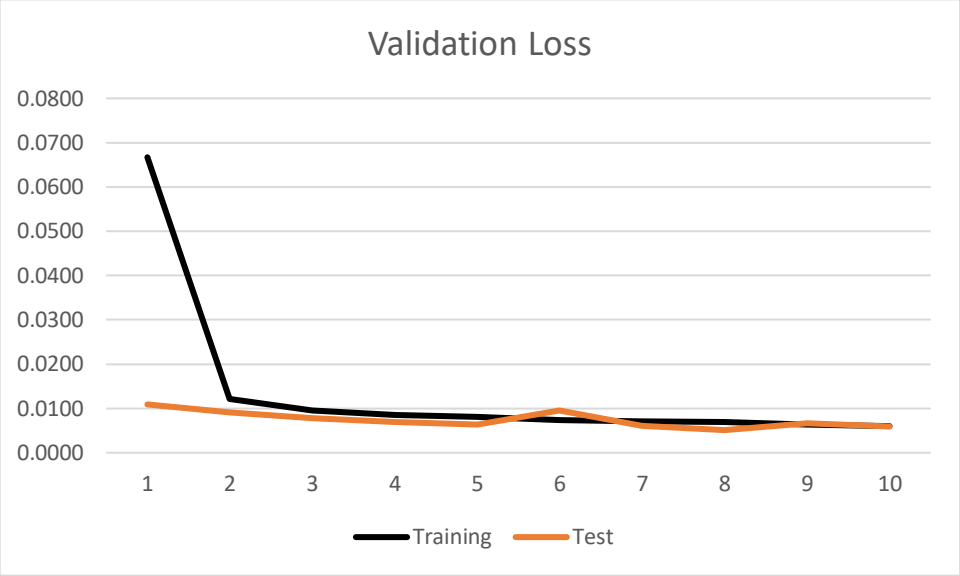

Рисунок 4.13 – Втрати моделі з трьома шарами LSTM

Проаналізувавши та порівнявши графіки на рис. 4.10-4.13 можна сказати, дві моделі вийшли однаково точними, проте у моделі з 2 шарами менше стрибків у графіку втрат. Занотуємо дані у таблицю 4.2, де L2 та L3 це дані моделі з двома шарами та трьома відповідно.

Таблиця 4.2 – Порівняльна таблиця моделей з двома та трьома шарами LSTM

Epoch	L2 accuracy	L3 accuracy	L2 loss	L3 loss
1	0,9970	0,9969	0,0101	0,0109
2	0,9975	0,9974	0,0080	0,0091
3	0,9977	0,9976	0,0060	0,0078
4	0,9971	0,9977	0,0088	0,0070
5	0,9979	0,9980	0,0065	0,0064
6	0,9981	0,9970	0,0056	0,0096
7	0,9988	0,9980	0,0054	0,0061
8	0,9986	0,9985	0,0049	0,0051
9	0,9984	0,9980	0,0054	0,0066
10	0,9985	0,9982	0,0046	0,0059

Отже, різниця хоч і не значна, але модель з двома шарами виявилась кращою і немає потреби у збільшені шарів.

Перевіримо точність моделі на 1 шарі та порівняємо результати з двома попередніми, відобразивши на рис. 4.14, 4.15.

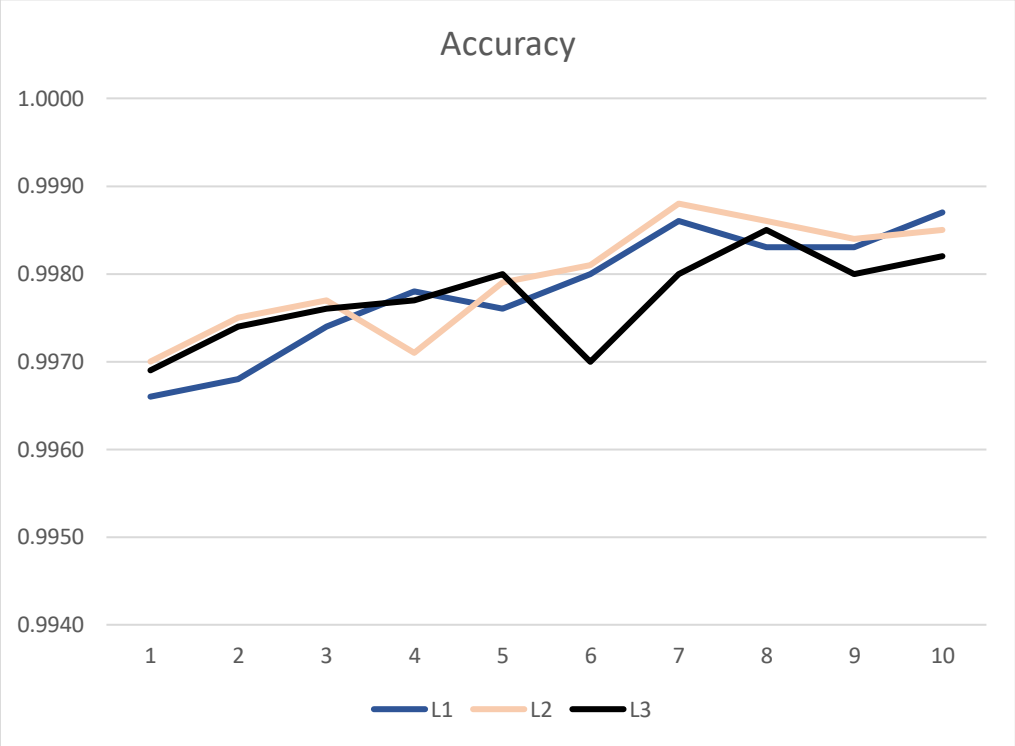

Рисунок 4.14 – Точність трьох моделей LSTM

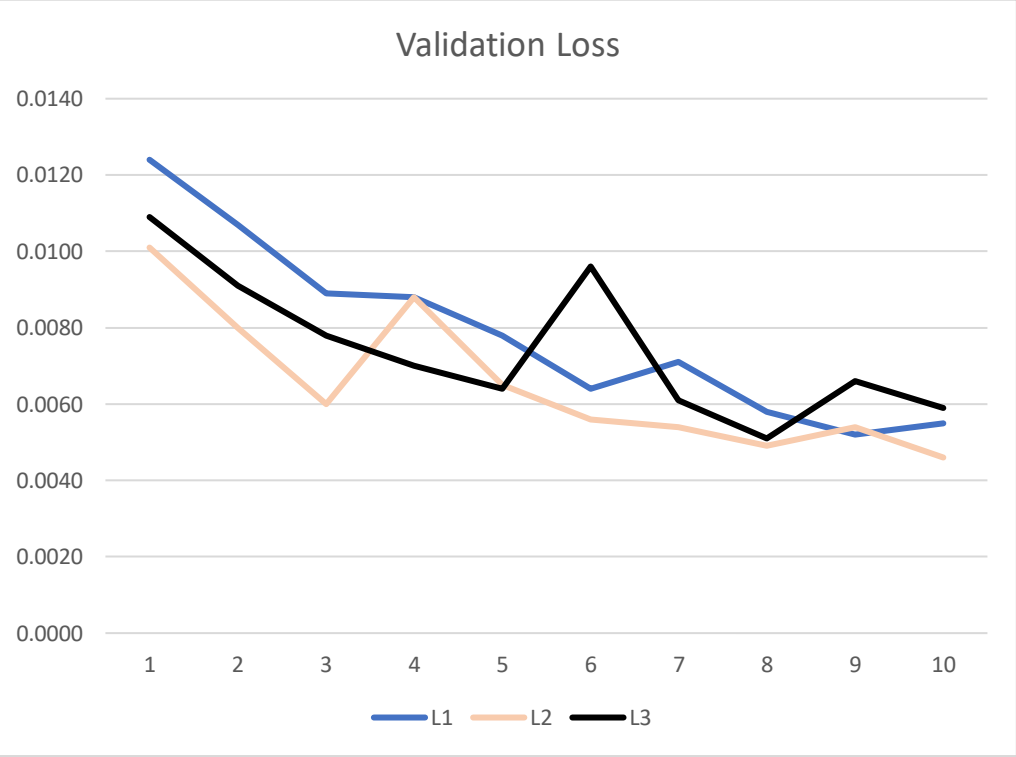

Рисунок 4.15 – Втрати трьох моделей LSTM

Таблиця 4.3 – Порівняльна таблиця трьох моделей LSTM

Епохи	L1 точність	L2 точність	L3 точність	L1 втрати	L2 втрати	L3 втрати
1	0,9966	0,9970	0,9969	0,0124	0,0101	0,0109
2	0,9968	0,9975	0,9974	0,0107	0,0080	0,0091
3	0,9974	0,9977	0,9976	0,0089	0,0060	0,0078
4	0,9978	0,9971	0,9977	0,0088	0,0088	0,0070
5	0,9976	0,9979	0,9980	0,0078	0,0065	0,0064
6	0,9980	0,9981	0,9970	0,0064	0,0056	0,0096
7	0,9986	0,9988	0,9980	0,0071	0,0054	0,0061
8	0,9983	0,9986	0,9985	0,0058	0,0049	0,0051
9	0,9983	0,9984	0,9980	0,0052	0,0054	0,0066
10	0,9987	0,9985	0,9982	0,0055	0,0046	0,0059

Проаналізувавши графік та таблиці можна дійти висновку, що результати точності приблизно однакові і модель з одним шаром LSTM найменше схильна до перенавчання. Проте, модель з двома шарами має найменшу помилку. Оберемо модель з двома шарами LSTM для порівняння з моделлю CNN+LSTM.

4.3 Модель, яка включає CNN та LSTM

Порівняємо моделі, що включають в себе CNN та LSTM. Для цього створимо дві моделі:

– LSTM (250 шарів) + CNN (128 шарів) (див. рис. 4.16);
– CNN (128 шарів) + LSTM (250 шарів) (див. рис. 4.17).

```
model = Sequential()
model.add(LSTM( units: 250, activation='tanh', return_sequences=True,
                input_shape=(timesteps, features)))
model.add(Conv1D(filters=128, kernel_size=1, activation='relu'))
model.add(MaxPooling1D(pool_size=1))
model.add(Flatten())
model.add(Dropout(0.13))
model.add(Dense(len(np.unique(y_train)), activation='softmax'))
```

Рисунок 4.16 – Характеристики моделі LSTM+CNN

```
model = Sequential()
model.add(Conv1D(filters=128, kernel_size=1, activation='relu',
                 input_shape=(timesteps, features)))
model.add(LSTM( units: 250, activation='tanh', return_sequences=True))
model.add(MaxPooling1D(pool_size=1))
model.add(Flatten())
model.add(Dropout(0.13))
model.add(Dense(len(np.unique(y_train)), activation='softmax'))
```

Рисунок 4.17 – Характеристика моделі CNN+LSTM

Отже, обидві моделі мають dropout 0.13. Для згорткової моделі використаємо активацію relu, а для рекурентної tanh. Вихідний шар матиме активацію softmax.

Графіки для порівняння ефективності моделі сформовано на рис. 4.18 та 4.19.

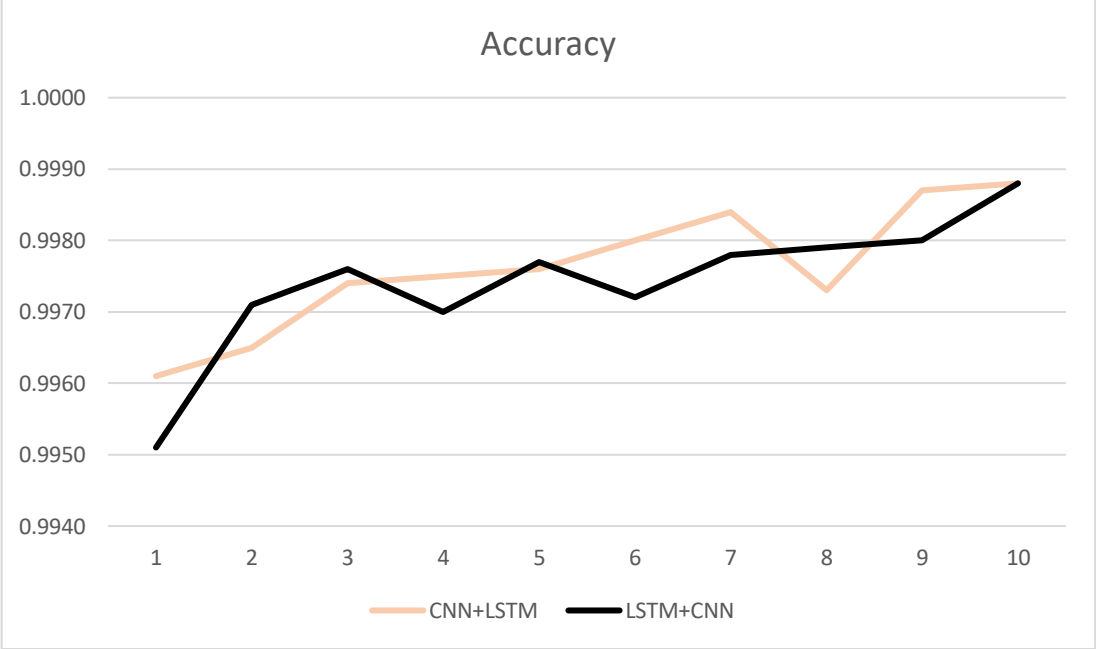

Рисунок 4.18 – Точність моделей CNN+LSTM та LSTM+CNN

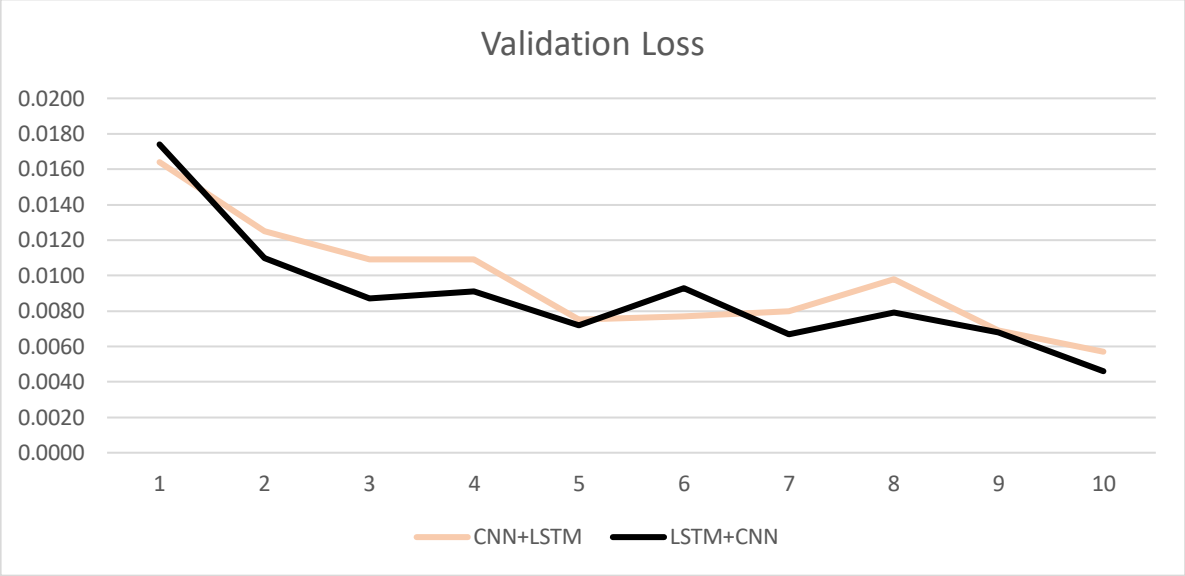

Рисунок 4.19 – Результат функції втрат

Проаналізувавши графіки на рис. 4.18 та 4.19 можна дійти висновку, що обидві моделі точні, проте LSTM+CNN має більш лінійний графік на відміну від CNN+LSTM, як більш схильна до перенавчання.

Підберемо оптимальну кількість нейронів у кожному шарі нейронній мережі шляхом тестування та створення порівняльної таблиці. Для цього в першу чергу змінимо кількість нейронів у згорткових шарів, всі інші параметри буть незмінні. Результати тестування моделі з 64,128 та 172 нейронами відобразимо на рис. 4.20.

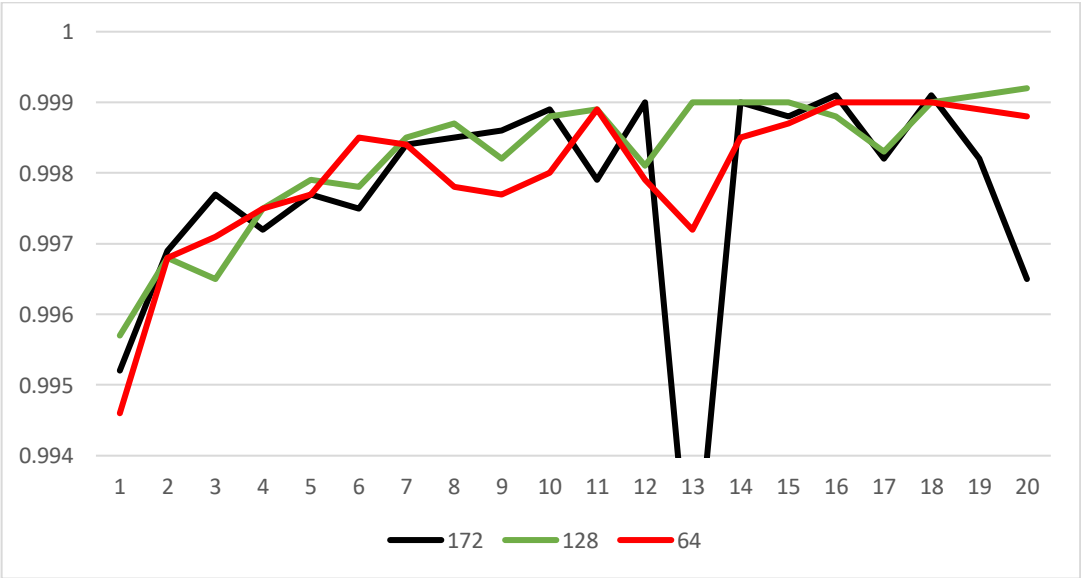

Рисунок 4.20 – Тестування моделей з різною кількістю нейронів у CNN

Провівши аналіз рис. 4.20, можна дійти висновку, що моделі з 172 та 64 нейронами в шарі найбільш схильні до перенавчання, тому для подальшої роботи обрано значення 128, так як ця модель показує найбільш стабільний графік.

Протестуємо модель на різних значеннях нейронів в шарі LSTM. Для цього порівняємо модель з 250, 300 та 350 нейронами у шарі. Дані відобразимо на рис. 4.21.

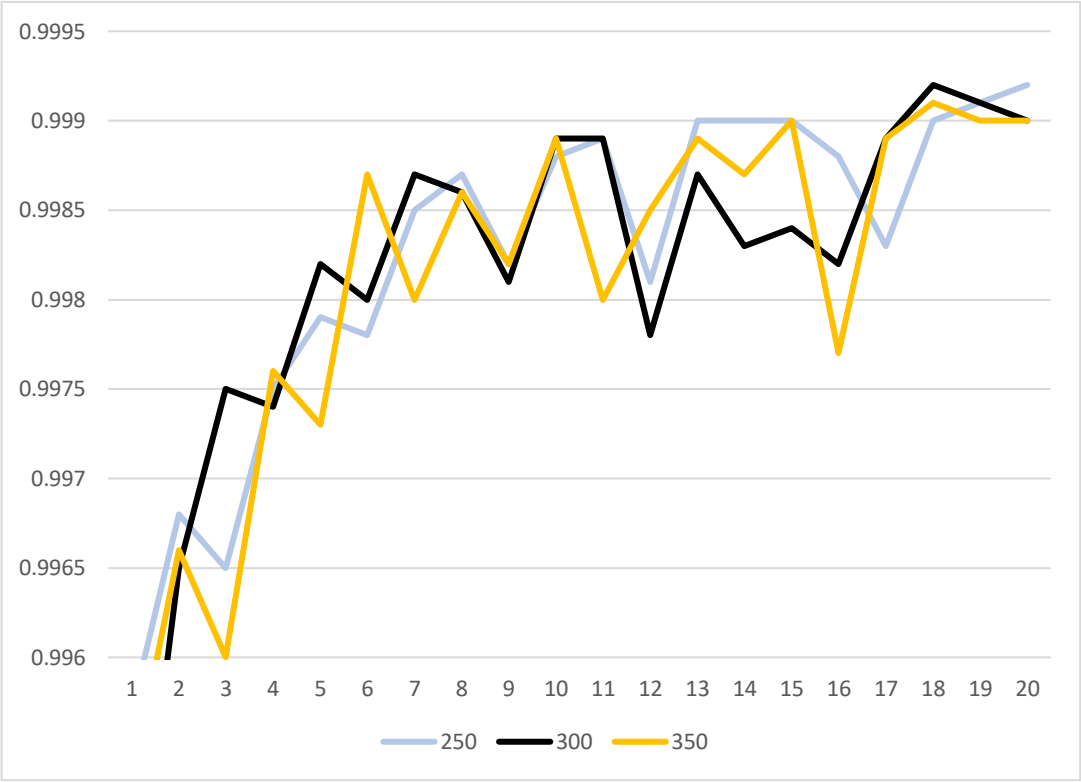

Рисунок 4.21 – Тестування моделі з різною нейронів у LSTM

Проаналізувавши рис. 4.21, дуже важко визначити оптимальну кількість нейронів, тому проаналізуємо метрики та матриці суміжності [47] кожної моделі (див. рис. 4.22-4.25).

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      1.00     60000
           1       1.00      1.00      1.00     60000
           2       1.00      1.00      1.00     60000
           3       1.00      1.00      1.00     60000
           4       1.00      1.00      1.00     60000
           5       1.00      1.00      1.00     60000
           6       1.00      1.00      1.00     60000

    accuracy                           1.00    420000
   macro avg       1.00      1.00      1.00    420000
weighted avg       1.00      1.00      1.00    420000

Confusion Matrix:
[[59695     6    13   258    15    13     0]
 [   11 59988     0     1     0     0     0]
 [    1     0 59999     0     0     0     0]
 [    5     0    34 59961     0     0     0]
 [   16     0     7     0 59974     3     0]
 [   11     0     0     2     4 59983     0]
 [    0     0     0     0     0     0 60000]]
```

Рисунок 4.22 – Результати моделі з 200 нейронами

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     60000
           1       1.00      1.00      1.00     60000
           2       1.00      1.00      1.00     60000
           3       1.00      1.00      1.00     60000
           4       1.00      1.00      1.00     60000
           5       1.00      1.00      1.00     60000
           6       1.00      1.00      1.00     60000

    accuracy                           1.00    420000
   macro avg       1.00      1.00      1.00    420000
weighted avg       1.00      1.00      1.00    420000

Confusion Matrix:
[[59762    12    12   172    17    25     0]
 [    8 59991     0     1     0     0     0]
 [    0     0 60000     0     0     0     0]
 [    8     0    34 59958     0     0     0]
 [   13     0     2     0 59965    20     0]
 [    5     0     0     2     1 59992     0]
 [    0     0     0     0     0     0 60000]]
```

Рисунок 4.23 – Результати моделі з 250 нейронами

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     60000
           1       1.00      1.00      1.00     60000
           2       1.00      1.00      1.00     60000
           3       1.00      1.00      1.00     60000
           4       1.00      1.00      1.00     60000
           5       1.00      1.00      1.00     60000
           6       1.00      1.00      1.00     60000

    accuracy                           1.00    420000
   macro avg       1.00      1.00      1.00    420000
weighted avg       1.00      1.00      1.00    420000

Confusion Matrix:
[[59716     7    14   233    19    11     0]
 [   11 59988     0     1     0     0     0]
 [    0     0 59978     0    22     0     0]
 [   22     0    34 59944     0     0     0]
 [   24     0     0     0 59971     5     0]
 [   15     0     0     2     5 59978     0]
 [    0     0     0     0     0     0 60000]]
```

Рисунок 4.24 – Результати моделі з 300 нейронами

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     60000
           1       1.00      1.00      1.00     60000
           2       1.00      1.00      1.00     60000
           3       1.00      1.00      1.00     60000
           4       1.00      1.00      1.00     60000
           5       1.00      1.00      1.00     60000
           6       1.00      1.00      1.00     60000

    accuracy                           1.00    420000
   macro avg       1.00      1.00      1.00    420000
weighted avg       1.00      1.00      1.00    420000

Confusion Matrix:
[[59702     4    12   245    16    21     0]
 [   17 59982     0     1     0     0     0]
 [    0     0 59975     0    25     0     0]
 [    5     0    34 59961     0     0     0]
 [   16     0     0     0 59969    15     0]
 [   14     0     0     2     2 59982     0]
 [    0     0     0     0     0     0 60000]]
```

Рисунок 4.25 – Результати моделі з 350 нейронами

Проаналізувавши результати моделей на рис. 4.22 – 4.25, можна дійти висновку, що всі три моделі мають високу точність, так як всі метрики дорівнюють одиниці. Матриці суміжностей (Confusion Matrix) показує кількість випадків коли

моделі вдалось правильно класифікувати та кількість помилок. Будемо орієнтуватись на кількість правильно класифікованих нормальних значень, так як їх була найбільша кількість до балансування даних і дуже важливо, щоб блокувались лише шкідливі запити, а для звичайних користувачів ресурс залишався доступним. Найкращий результат показала нейрона мережа з 250 нейронами у LSTM шарі. Вона класифікувала правильно 59762 спостереження з 60000, що є дуже точним результатом. Також для навчання моделі з 250 нейронами треба набагато менше часу ніж з 300 або 350 нейронами.

Збільшимо кількість епох до 40 та переглянемо результати створеної нейронної мережі на рис. 4.26 та 4.27.

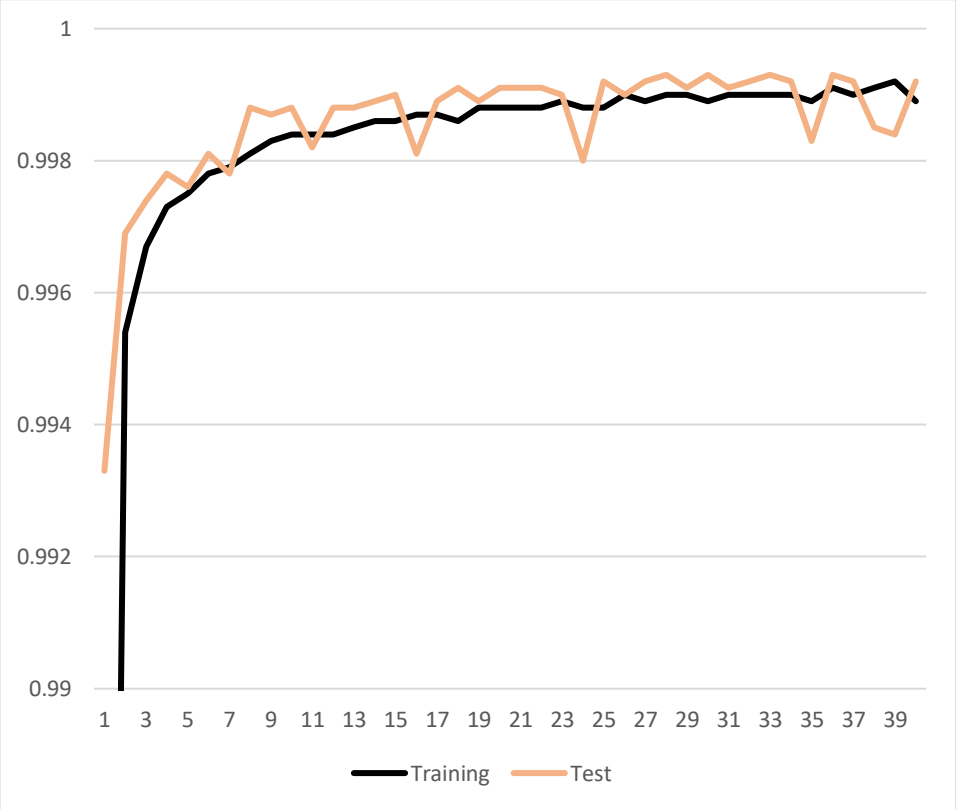

Рисунок 4.26 – Точність моделі LSTM (250) + CNN (128)

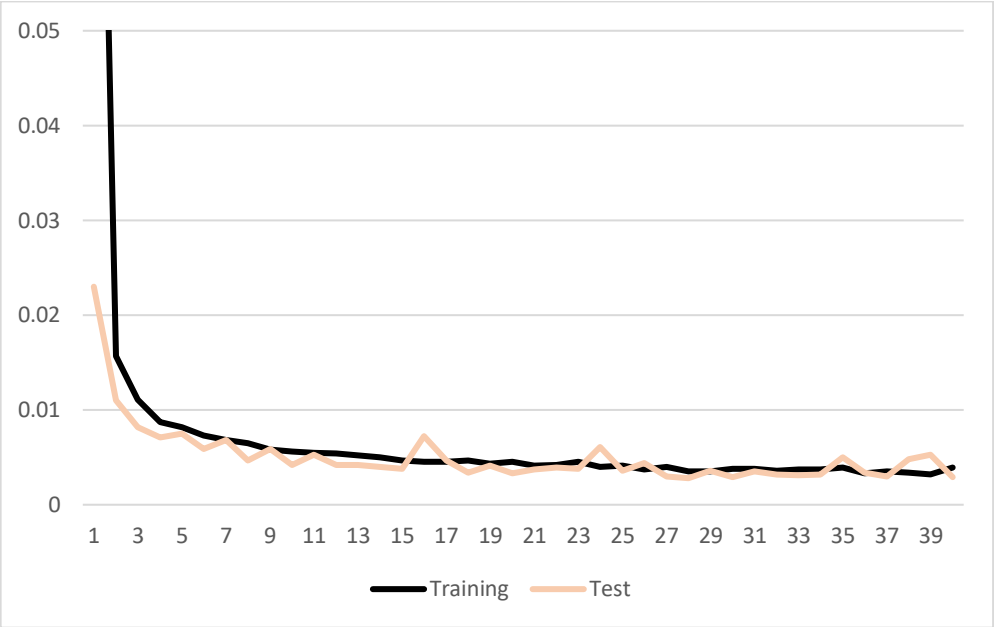

Рисунок 4.27 – Результат функції втрат моделі (250) + CNN (128)

На табл. 4.4 представлені результати моделі, де значення точності була вище за 0.999.

Таблиця 4.4 – Епохи, де точність моделі вище за 0.999

Епоха	**Точність трен. Дані**	**Втрати трен. Дані**	**Точність тестові данні**	**Втрати тестові дані**
18	0,9986	0,0047	0,9991	0,0034
20	0,9988	0,0045	0,9991	0,0033
21	0,9988	0,0041	0,9991	0,0037
22	0,9988	0,0042	0,9991	0,0039
25	0,9988	0,0041	0,9992	0,0036
27	0,9989	0,004	0,9992	0,003
28	0,999	0,0035	0,9993	0,0028
29	0,999	0,0035	0,9991	0,0036
30	0,9989	0,0038	0,9993	0,0029
31	0,999	0,0038	0,9991	0,0035
32	0,999	0,0036	0,9992	0,0032
33	0,999	0,0037	0,9993	0,0031
34	0,999	0,0037	0,9992	0,0032
36	0,9991	0,0033	0,9993	0,0034
37	0,999	0,0035	0,9992	0,003
40	0,9992	0,0029	0,9989	0,0039

Провівши аналіз табл. 4.4, можна дійти висновку, що достатньо 30 епох. Після цього модель перенавчається, хоча серйозних викидів не зафіксовано. В кінці можна спостерігати стабілізацію точності.

Висновок до розділу 4

У ході тестування моделей із застосуванням згорткових нейронних мереж та довготривало-короткочасної пам’яті для класифікації мережевого трафіку вдалося визначити оптимальні параметри і конфігурації цих моделей. Дослідження CNN-моделей продемонструвало, що зменшення Dropout з 0.5 до 0.1 може забезпечити більш стабільне навчання без значного росту перенавчання.

Комбіновані архітектури CNN+LSTM та LSTM+CNN продемонстрували, що послідовність використання шарів впливає на стабільність результатів. Модель LSTM+CNN забезпечила більш лінійний тренд навчання й знизила ризик перенавчання. Оптимізація кількості нейронів у шарах CNN та LSTM довела, що надмірне ускладнення структури не завжди дає кращі результати, а вибір середньої кількості нейронів: 128 для CNN та 250 для LSTM є раціональним компромісом.

Загалом, правильно підібрана глибина моделі, параметри Dropout та кількість нейронів у шарах забезпечують високу точність класифікації та стійкість до перенавчання. За 30 епох, яких достатньо для даної моделі, вдалось досягти точності в 99,93% при результатах функції втрат 0.0028

ВИСНОВКИ

У межах кваліфікаційної роботи розроблено інтелектуальну систему ідентифікації та класифікації DoS-атак із використанням сучасних методів глибинного навчання. В процесі дослідження було проведено детальний аналіз сучасних підходів кластеризації, обрано оптимальні архітектури нейронних мереж, виконано обробку даних, моделювання та тестування моделей.

Було детально підготовлено набір даних, беручи за основу PCAP файли з набір даних канадського університету CIC-IDS2017, що включає сучасний мережевий трафік із різними типами атак. Виконано очищення, стандартизацію та балансування шляхом створення синтетичних даних, методом SMOTE, що забезпечило якісну основу для навчання моделей. У ході роботи було протестовано моделі CNN та LSTM, а також їх комбіновані варіанти. Оптимізовано параметри: кількість шарів, нейронів та рівень Dropout. Вся пророблена робота дозволили досягнути точності 99,93%.

Комбінація LSTM та CNN забезпечила стабільні результати та стійкість до перенавчання. На відміну від класичних методів машинного навчання, дана модель дозволяє вищої точності, що дозволяє зменшити вірогідність блокування доступу до сервісів звичайним користувачам.

Даний підхід може бути корисним великим компаніям, таким як банки, корпорації та багатомільйонні продуктові компанії, оскільки технології глибинного навчання потребують значних ресурсів, наприклад технології хмарних обчислень. Використання таких зовнішніх ресурсів буде дешевшим, ніж створення власних серверів для обробки подібних завдань. При цьому захист буде набагато ефективнішим, а користувацький досвід у таких компаніях – кращим, ніж у компаній, які використовують традиційні методи машинного навчання для ідентифікації та класифікації DoS-атак. До таких методів належать, зокрема, наївний Байєсів класифікатор, метод опорних векторів, дерево рішень, метод k-близьких сусідів та логістична регресія.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Analysis of a denial of service attack on TCP / Schuba, C. L. et al.; IEEE Symposium on Security and Privacy, 1997. P. 145-152. URL: https://ieeexplore.ieee.org/document/595181 (дата звернення: 01.10.2024).

2. Mirkovic, J., Reiher, P. A taxonomy of DDoS attack and DDoS defense mechanisms. ACM SIGCOMM Computer Communication Review, 2004. P. 34(2), 39-53. URL: https://dl.acm.org/doi/10.1145/997150.997156 (дата звернення: 01.10.2024).

3. Beitollahi, H., Deconinck, G. Analyzing well-known countermeasures against distributed denial of service attacks. Computer Communications, 2007. P. 35(11), 1312-1332. URL: https://www.sciencedirect.com/science/article/pii/S0140366411004413 (дата звернення: 03.10.2024).

4. Rossow, C. Amplification hell: Revisiting network protocols for DDoS abuse, 2014. p. 231-237. URL: https://www.ndss-symposium.org/ndss2014/ndss-2014-programme/amplification-hell-revisiting-network-protocols-ddos-abuse/ (дата звернення: 03.10.2024).

5. Buczak, A. L., Guven, E. A survey of data mining and machine learning methods for cyber security intrusion detection. IEEE Communications Surveys & Tutorials, 2016. P. 18(2), 1153-1176. URL: https://ieeexplore.ieee.org/document/7440948 (дата звернення: 03.10.2024).

6. Vinayakumar, R., Soman, K. P., Poornachandran, P. Applying deep learning approaches for network traffic analysis. International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2017. P. 2353-2358.

7. NSL-KDD. Network Security, Information Security, Cyber Security. URL: https://www.kaggle.com/datasets/hassan06/nslkdd (дата звернення: 03.10.2024).

8. Diro, A. A., Chilamkurti, N. Distributed attack detection scheme using deep learning approach for Internet of Things. Future Generation Computer Systems, 2017. P. 82, 761-768.

9. Kim, G., Lee, S., Kim, S. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Systems with Applications, 2014. P. 41(4), 1690-1700. URL: https://www.sciencedirect.com/science/article/pii/S0957417413010896 (дата звернення: 03.10.2024).

10. Mirai Botnet DDoS attack explained. URL: https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/ (дата звернення: 05.10.2024).

11. What is DDoS amplification? URL: https://www.cloudflare.com/learning/ddos/ddos-amplification/ (дата звернення: 11.10.2024).

12. Goldberg, A., Schalk, C. JavaServer Faces: What is JSF? URL: https://docs.oracle.com/javaee/7/tutorial/interceptors001.htm (дата звернення: 11.10.2024).

13. Support Vector Machine (SVM) Algorithm. URL: https://www.geeksforgeeks.org/support-vector-machine-algorithm/ (дата звернення: 12.10.2024).

14. What is a Decision Tree? URL: https://www.geeksforgeeks.org/decision-tree/ (дата звернення: 12.10.2024).

15. Phys J., Detection of dos attacks using naive bayes method based on internet of things, 2021. P. 8-9.

16. What is the KNN algorithm? IBM. URL: https://www.ibm.com/topics/knn#:~:text=The%20k%2Dnearest%20neighbors%20(KNN)%20algorithm%20is%20a%20non,used%20in%20machine%20learning%20today (дата звернення: 12.10.2024).

17. Fog-to-things: What is IoT and its use in fog computing. URL: https://www.ibm.com/cloud/learn/iot (дата звернення: 12.10.2024).

18. Doshi R., Apthorpe N., Feamster N. Machine Learning DDoS Detection for Consumer Internet of Things Devices. Security and Privacy Workshops (SPW), 2018. P. 29–35.

19. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge, MA : MIT Press, 2016. P. 800 .

20. Bishop C. M. Pattern Recognition and Machine Learning. New York : Springer, 2006. P. 738.

21. Understanding the evolution of DDoS attacks. URL: https://www.fortinet.com/resources/cyberglossary/ddos-attack (дата звернення: 13.10.2024).

22. DDoS attack detection and classification via Convolutional Neural Network (CNN), Shaaban A., R. et al, IEEE Ninth, 2019. P. 18-32.

23. Modeling an intrusion detection using recurrent neural networks. URL: https://www.sciencedirect.com/science/article/pii/S2307187723000135 (дата звернення: 13.10.2024).D

24. LSTM explained: A guide to understanding recurrent neural networks. URL: https://towardsdatascience.com/understanding-lstm-networks-74e9d5e8ec0b (дата звернення: 15.10.2024).

25. Luo J., Xia Y. Deep learning based on LSTM for Intrusion Detection. IEEE Access, 2020. P. 54–58.

26. Amplified reflection DDoS attacks: A deep dive. URL: https://www.cloudflare.com/en-gb/learning/ddos/ddos-reflection/ (дата звернення: 16.10.2024).

27. Amplification attacks: Techniques and defense. URL: https://www.imperva.com/learn/application-security/amplification-attacks/ (дата звернення: 18.10.2024).

28. What is TCP SYN flood attack? URL: https://www.cloudflare.com/en-gb/learning/ddos/tcp-syn-flood-ddos-attack/ (дата звернення: 18.10.2024).

29. Insider threats and cyber-espionage explained. URL: https://www.crowdstrike.com/epp-101/insider-threats-and-cyber-espionage (дата звернення: 19.10.2024).

30. Methods to prevent DDoS attacks. URL: https://www.incapsula.com/ddos/ddos-prevention.html (дата звернення: 19.10.2024).

31. How to prevent DDoS attacks in 2024. URL: https://www.prolexic.com/ddos-prevention/ (дата звернення: 23.10.2024).

32. The impact of 5G on network security. URL: https://www.paloaltonetworks.com/cyberpedia/how-5g-will-change-network-security (дата звернення: 23.10.2024).

33. Future trends in DDoS protection. URL: https://www.cyberark.com/what-is/ddos-protection/ (дата звернення: 23.10.2024).

34. Deep learning-based anomaly detection for cybersecurity. URL: https://www.sciencedirect.com/science/article/pii/S095741741930505X (дата звернення: 25.10.2024).

35. What is a SYN flood attack? URL: https://www.imperva.com/learn/ddos/syn-flood/ (дата звернення: 30.10.2024).

36. Intrusion detection evaluation dataset (CIC-IDS2017). URL: https://www.unb.ca/cic/datasets/ids-2017.html (дата звернення: 04.11.2024).

37. Wireshark: офіційний вебсайт. URL: https://www.wireshark.org (дата звернення: 04.11.2023)

38. TCP Analysis using Wireshark. URL: https://www.geeksforgeeks.org/tcp-analysis-using-wireshark/ (дата зверненя

39. CICFlowMeter – An open-source tool: URL: https://www.canadianinstitute forcybersecurity.ca/tools (дата звернення: 05.11.2023).

40. Zuech R., Khoshgoftaar T. M., Wald R. Intrusion detection and Big Heterogeneous Data: a Survey. Journal of Big Data, 2015. Vol. 2, № 1. P. 1–41. DOI: 10.1186/s40537-015-0020-9.

41. Corona I., Giacinto G., Roli F. Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. Information Sciences, 2013. Vol. 239. P. 201–225. DOI: 10.1016/j.ins.2013.03.018.

42. Fernandes A. F., Rodrigues J. J. P. C., et al. A comprehensive survey on network anomaly detection. Telecommunication Systems, 2019. Vol. 70. P. 447–489. DOI: 10.1007/s11235-018-0508-2.

43. Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection / Shiravi A. et al, Computers & Security, 2012. Vol. 31, № 3. P. 357–374. DOI: 10.1016/j.cose.2011.12.012.

44. Brownlee J. Imbalanced Classification with Python. Machine Learning Mastery, 2020. P. 1–380.

45. SMOTE – Azure Machine Learning. URL: https://learn.microsoft.com/ru-ru/azure/machine-learning/component-reference/smote?view=azureml-api-2 (дата звернення: 06.11.2024).

46. Training and Validation Loss in Deep Learning. URL: https://www.baeldung.com/cs/training-validation-loss-deep-learning (07.11.2024).

47. Confusion Matrices and Classification Reports: A Guide to Evaluating Machine Learning Models. URL: https://smuhabdullah.medium.com/confusion-matrices-and-classification-reports-a-guide-to-evaluating-machine-learning-models-385496cf7cee (дата звернення: 09.11.2024).

ДОДАТОК А

Лістинг коду обробки набору даних

```
from dataclasses import replace
import pandas as pd
import numpy as np
from sklearn.utils import resample
from sklearn.preprocessing import StandardScaler
from imblearn.under_sampling import RandomUnderSampler
from imblearn.over_sampling import RandomOverSampler, SMOTE
from sklearn.model_selection import train_test_split
from etc.function import sub_df

# Завантаження датасетів
data_monday = pd.read_csv(r'D:\Your_route\CSV\Friday-WorkingHours-Afternoon-
DDos.pcap_ISCX.csv')
data_wednesday = pd.read_csv(r'D:\Your_route\CSV\Wednesday-workingHours.pcap_ISCX.csv')

# Об'єднання датасетів
df = pd.concat([data_monday, data_wednesday], ignore_index=True)
df=sub_df(df)
df.drop('Flow ID', axis=1, inplace=True)
df.drop('Source IP', axis=1, inplace=True)
df.drop('Destination IP', axis=1, inplace=True)
df.drop('Timestamp', axis=1, inplace=True)
df=df.drop_duplicates()

# Попередній аналіз
print(f"Розмір об'єднаного датасету: {df.shape}")
print("Список колонок:", df.columns.tolist())
print("Унікальні значення 'Label':", df['Label'].unique())
print("Кількість спостережень за класами:\n", df['Label'].value_counts())
print(f"Розмір DF до видалення пропущених значень:{df.shape[0]}")
# Видалення пропущених значень
df = df.dropna()  # Видаляємо рядки з пропущеними значеннями
print(f"Розмір DF до видалення пропущених значень:{df.shape[0]}")


#Зменшення екземплярів BENIGN
majority_class = df[df['Label']=='BENIGN']
minority_classes = df[df['Label']!='BENIGN']
majority_downsampled = resample(
    majority_class,
    replace=False,
    n_samples = 300000,
    random_state=42
)
df_after_undersampling =pd.concat([majority_downsampled,minority_classes])

# Розділення на ознаки (X) та мітки (y)
x = df_after_undersampling.drop(columns=['Label'])
y = df_after_undersampling['Label']
print(f"Розподіл класів: {pd.Series(y.values).value_counts()}")
# Перевірка та обробка нескінченностей (inf)
print("Перевірка на inf:")
if np.isinf(x).values.any():
    print("Знайдено inf! Заміна на максимальні значення.")
```

```
    x = np.where(np.isinf(x), np.nan, x)  # Заміна inf на NaN
    x = pd.DataFrame(x, columns=df.drop(columns=['Label']).columns)  # Відновлюємо
DataFrame

# Перевірка та обробка NaN
if np.isnan(x).values.any():
    print("Знайдено NaN! Заміна на середнє значення.")
    x = x.fillna(x.mean())  # Заповнюємо NaN середніми значеннями

# Стандартизація
print("Стандартизація даних...")
print(f"До стандартизації: \n {x.head()}")
scaler = StandardScaler()
x_normalized = scaler.fit_transform(x)

# Відновлення `Label` після нормалізації
normalized_df = pd.DataFrame(x_normalized, columns=x.columns)
normalized_df['Label'] = y.values

# Перевірка результату
print(f"Після стандартизації: \n {normalized_df.head()}")
smote = SMOTE(random_state=21)
x_balanced, y_balanced = smote.fit_resample(x_normalized, y.values)

# Розділення на тренувальний і тестовий набори до балансування
x_train, x_test, y_train, y_test = train_test_split(
    x_balanced, y_balanced, test_size=0.2, random_state=21, stratify=y_balanced
)

# Перевірка розподілу класів до балансування
print("Розподіл класів у тренувальному наборі до балансування:\n",
pd.Series(y.values).value_counts())

# Перевірка розподілу після балансування
print("Розподіл класів у тренувальному наборі після балансування:\n",
pd.Series(y_train).value_counts())

# Перевірка розмірів наборів
print(f"x_train_balanced size: {x_train.shape}")
print(f"x_test_balanced size: {x_test.shape}")
print(f"y_train_balanced size: {y_train.shape} and {pd.Series(y_train).value_counts()}")
print(f"y_test_balanced size: {y_test.shape} and {pd.Series(y_test).value_counts()}")

print("Перевірка унікальних міток у збалансованому тренувальному наборі:")
print(pd.Series(y_train).unique())

print("Перевірка унікальних міток у збалансованому тестовому наборі:")
print(pd.Series(y_test).unique())

# Створення тренувального і тестового DataFrame
train_df = pd.DataFrame(x_train, columns=df.drop(columns=['Label']).columns)
train_df['Label'] = y_train  # Додаємо мітки, що були згенеровані після балансування
test_df = pd.DataFrame(x_test, columns=df.drop(columns=['Label']).columns)
test_df['Label'] = y_test  # Тестовий набір зберігає оригінальні мітки

# Збереження підготовлених даних
train_df.to_csv('full_train_df.csv', index=False)
test_df.to_csv('full_test_df.csv', index=False)
```

ДОДАТОК Б

Лістинг коду навчання фінальної моделі

```
import pandas as pd
import numpy as np
from keras import Sequential
from keras.src.callbacks import EarlyStopping
from keras.src.layers import Conv1D, MaxPooling1D, LSTM, Dropout, Dense, Flatten
from keras.src.utils import plot_model
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.preprocessing import LabelEncoder
TF_ENABLE_ONEDNN_OPTS=0
train_df = pd.read_csv('full_train_df.csv')
test_df= pd.read_csv('full_test_df.csv')

y_train = train_df['Label']
x_train = train_df.drop(['Label'], axis = 1)

y_test = test_df['Label']
x_test = test_df.drop(['Label'], axis = 1)

label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
y_test = label_encoder.transform(y_test)

print("Відповідність міток класів після кодування:")
for index, class_label in enumerate(label_encoder.classes_):
    print(f"{index} - {class_label}")

print(x_test.columns)

x_train = x_train.values
x_test = x_test.values
y_train = y_train
y_test = y_test

print("After Training data shape:", x_train.shape)
print("After Training labels shape:", y_train.shape)

x_train = x_train.reshape(x_train.shape[0], 1, x_train.shape[1])
x_test = x_test.reshape(x_test.shape[0], 1, x_test.shape[1])
timesteps = x_train.shape[1]
features = x_train.shape[2]
# Створення CNN-LSTM моделі
model = Sequential()
model.add(LSTM(250, activation='tanh', return_sequences=True,
               input_shape=(timesteps, features)))
model.add(Conv1D(filters=128, kernel_size=1, activation='relu'))
model.add(MaxPooling1D(pool_size=1))
model.add(Flatten())
model.add(Dropout(0.13))
model.add(Dense(len(np.unique(y_train)), activation='softmax'))

# Компілювання моделі
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
plot_model(model, to_file='cnn_lstm_model.png', show_shapes=True, show_layer_names=True)
```

```
# Тренування моделі
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
history = model.fit(x_train, y_train, epochs=40
                    , batch_size=512, validation_data=(x_test, y_test))
# Оцінка моделі
y_pred = model.predict(x_test)
y_pred_classes = np.argmax(y_pred, axis=1)

print("Classification Report:")
print(classification_report(y_test, y_pred_classes))

print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred_classes))
```