МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІМ. Ю.М. ПОТЕБНІ ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Комп'ютерна система виявлення загроз в мережі VANET з використанням машинного навчання**

Виконав: студент 2 курсу, групи 8.1212-іпз-2
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
В.Ю. Шумаков
(ініціали та прізвище)
Керівник доцент, к.т.н. Н.П. Полякова
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Алтер Віжн Груп»
В.С. Тряпічко
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІМ. Ю.М. ПОТЕБНІ
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти другий (магістерський)
Спеціальність 121 Інженерія програмного забезпечення
(код та назва)
Освітня програма Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри Т.В. Критська
"01" вересня 2023 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Шумакова Віталія Юріївича
(прізвище, ім'я, по батькові)

1.Тема роботи Комп'ютерна система виявлення загроз в мережі VANET з використанням машинного навчання

керівник роботи доцент, к.т.н. Полякова Наталія Петрівна,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 09.10. 2023 р. №1577-с

2. Строк подання студентом кваліфікаційної роботи 01.12.2023
3. Вихідні дані магістерської роботи
- комплект нормативних документів;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми формування безпечного середовища на дорогах та розробка методів її вирішення;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
16 слайдів презентації

6. Консультанти розділів магістерської роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.09.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	02.09-10.09.23	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	11.09-12.09.23	виконано
3	Аналіз основних видів бездротових мереж	13.09-14.09.23	виконано
4	Аналіз політики безпеки та існуючих загроз мережі VANET	15.09-20.09.23	виконано
5	Дослідження засобів класифікації аномального стану мережі VANET	21.09-26.09.23	виконано
6	Узгодження подальших дій з науковим керівником	27.09-28.09.23	виконано
7	Генерація трафіку та початок розробки Date Parsing для підготовки тренувальної та тестової вибірки	29.09-13.10.23	виконано
8	Навчання моделі нейронної мережі за допомогою підготовленого тренувального набору	14.10-16.10.23	виконано
9	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого дослідження	17.10-19.10.23	виконано
10	Тестування створеної моделі та розробка методів для графічного зображення отриманих результатів	20.10-09.11.23	виконано
11	Оцінка продуктивності та якості класифікації	10.11-17.11.23	виконано
12	Виправлення технічних проблем	18.11-22.11.23	виконано
13	Оформлення пояснювальної записки	23.11-27.11.23	виконано

Студент _____ Шумаков В.Ю.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Полякова Н. П.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ Скрипник І. А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок: 99

Рисунків: 58

Таблиць: 5

Джерел: 33

Шумаков В. Ю. Комп’ютерна система виявлення загроз в мережі VANET з використанням машинного навчання: кваліфікаційна робота магістра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник Н. П. Полякова. Запоріжжя : ЗНУ, 2023. 99 с.

Мета і завдання дослідження полягають у вивченні алгоритмів забезпечення безпеки VANET-мереж у галузі захисту від деяких видів атак за допомогою нейромережевих методів, створивши для цього відповідну програмну систему — класифікатор на основі нейронної мережі, яка в якості вихідних даних буде використовувати параметри про стан мережі для ефективного визначення аномального стану та типу загрози.

У процесі дослідження була розглянута проблема формування безпечного середовища на дорогах та існуючі підходи до її вирішення. Як результат, була розроблена та навчена оптимальна модель класифікатора на основі багатошарового перцептрона. Розроблена мережа визначає аномальний стан мережі та класифікує тип загрози за короткий проміжок часу. Окрім цього, були розроблені додаткові методи графічного зображення якості класифікації та побудови матриці помилок для підтвердження ефективності розробленої системи.

Ключові слова: комп’ютерна система, класифікатор загроз, транспортна мережа VANET, нейронна мережа, багатошаровий перцептрон, Ad-hoc, NS-3, Keras.

SUMMARY

Pages: 99

Figures: 58

Tables: 5

Sources: 33

Shumakov V. Yu. Computer-based threat detection system in the VANET network using machine learning: qualification work of the master of specialty 121 "Software Engineering" / Science head N. P. Polyakova. Zaporizhzhia : ZNU, 2023. 99 p.

The goal and objectives of the study are to study the algorithms for ensuring the security of VANET networks in the field of protection against some types of attacks using neural network methods, creating a corresponding software system for this - a classifier based on a neural network, which will use network state parameters as input data for the effective identification of an abnormal state and the type of threat.

In the course of the study, the problem of forming a safe environment on the roads and existing approaches to its solution were considered. As a result, an optimal classifier model based on a multilayer perceptron was developed and trained. This network identifies an abnormal network state and classifies the type of threat in a short period of time. In addition, methods for graphical representation of the quality of classification and construction of the error matrix were developed to confirm the effectiveness of the developed system.

Keywords: computer system, threat classifier, VANET transport network, neural network, multilayer perceptron, Ad-hoc, NS-3, Keras.

ЗМІСТ

ВСТУП

Актуальність теми

Останнім часом проблеми формування безпечного середовища на дорогах та при цьому надання доступного та багатофункціонального «розумного» середовища з повною інтеграцією бездротових пристроїв призвели до створення бездротових транспортних мереж з динамічною топологією VANET [32]. Основним інструментом підвищення рівня безпеки стало своєчасне інформування учасників руху про поточну ситуацію на дорозі, погодні умови тощо. Аналіз актуальних досліджень у сфері забезпечення безпечного руху транспортних засобів (ТЗ) у бездротових мережах VANET показав, що необхідна розробка ефективних протоколів/алгоритмів поширення даних [1, 4]. Найбільш просунуті системи сьогодні використовують протоколи поширення даних з урахуванням поділу карт (Display Power Management Signaling, DPMS) [16] чи протоколи Geocast [29]. У цьому випадку реалізується зв'язок один до багатьох між об'єктами руху, при цьому ТЗ здійснює багатоадресне поширення даних у певну підмножину об'єктів, що знаходяться в заданій географічній області.

Системи інформування про дорожній рух потребують агрегування великих даних для надання рекомендацій транспортним засобам у чинних умовах дорожнього рухи, що призводить до підвищення комфорту користувачів. У цьому випадку, якщо об'єкт мережі буде атакований і дані при передачі будуть замінені, то по всій зоні видимості сегмента VANET можливе розкриття конфіденційної інформації, створення аварійних ситуацій і т.і. Саме тому необхідно проводити додатковий аналіз великих даних про аномалії та аналіз проведених несанкціонованих дій.

В рамках даного дослідження розглянуто питання побудови гібридної моделі ефективного розміщення вихідних та проміжних даних у бездротових транспортних мережах з динамічною топологією VANET, що є по суті структурним поданням програмно — конфігурованої мережі та інструментів проведення граничних обчислень, з можливістю оптимально щодо часу аналізувати

дані вузлів мережі. Крім того, проведено аналіз продуктивності та ефективності використання даної моделі при несанкціонованих аномальних дії об'єктів мережі.

Мета роботи

Мета роботи полягає у дослідженні алгоритмів забезпечення безпеки VANET-мереж у галузі захисту від деяких видів атак за допомогою нейромережевих методів. Також метою є створення програмної системи — класифікатора на основі нейронної мережі, яка за обраними ознаками, буде автоматично визначати кожну обрану атаку.

Об’єкт дослідження

Потік даних, що надходить до мережі VANET для виявлення нормального чи аномального стану мережі.

Предмет дослідження

Детектування, розпізнавання аномального стану мережі та виявлення конкретного типу загрози.

Методи дослідження

Для розв’язання задач використовуються такі методи дослідження:

- Аналіз особливостей та існуючих рішень для проблеми виявлення загроз в мережі VANET.
- Аналіз існуючих моделей нейронних мереж для вирішення проблем мережі VANET.
- Аналіз різноманітних бібліотек для глибокого навчання.
- Дослідження різноманітних видів загроз на мережу VANET.
- Аналіз трафіку всередені мережі для обрання ознак, за якими можна класифікувати загрози.

- Синтез отриманих в процесі дослідження та навчання проблем та методів їх вирішення.
- Експерементування з різноманітними оптимізаторами навчання.

Наукова новизна

Одержані результати є наочним відображення переваг та недоліків використання нейронних мереж для виявлення та швидкого класифікування атаки, яка здійснюється на мережу для подальшого знищення та модифікування безпечного середовища.

Практичне значення одержаних результатів

На базі отриманих результатів можна зрозуміти які загрози легше знаходяться та класифікуються та які параметри є найбільш ефективними для вирішення задач розпізнавання загрози мережі.

Проаналізувавши дану роботу можна зробити висновки що до ефективності побудованого класифікатора (проаналізувати його точність на навчальних та тестових даних), ознайомитися з проблемами, які виникають під час розробки такої системи, та методами їх вирішення.

Глосарій

Автомобільна мережа, що самоорганізується — децентралізована бездротова мережа, учасниками якої є транспортні засоби.

Аутентифікація — процедура аутентифікації, наприклад: автентифікація користувача шляхом порівняння введеного ним пароля (для вказаного логіна) з паролем, збереженим у базі даних логінів користувачів.

Батч (batch gradient descent) — реалізація градієнтного спуску, коли на кожній ітерації навчальний вибір проглядається повністю, і тільки після цього змінюється маса моделі.

Бездротова мережа ad hoc — децентралізована бездротова мережа, яка не має постійної структури.

Впевненість нейронної мережі (англ. Confidence) — це значення на вихідному шарі нейронної мережі, що визначає ймовірність правильності для будь-якого зі своїх прогнозів.

Глибинне навчання (англ. Deep learning) — це сукупність методів машинного навчання які ґрунтуються на навчанні за допомогою ознак, отриманих з різноманітних даних (зображень, текстів чи звуків), використовуючи при цьому багатошарові нейронні мережі.

Глибока нейронна мережа (англ. Deep neural network) — це нейронна мережа яка складаються з кількох рівнів нелінійних операцій, наприклад нейронні мережі з багатьма прихованими шарами.

Згорткова нейронна мережа (англ. Convolutional neural network) — це нейронна мережа прямого поширення спеціального виду, в основі якої є фільтри, основна мета яких розпізнавати характеристик певних класів об’єктів.

Набір даних, датасет (англ. Dataset) — це оброблена та структурована певним чином інформація, яку модель нейронної мережі використовує під час навчання та тестування.

Нейронна мережа — це низка алгоритмів, які намагаються розпізнати основні взаємозв’язки в наборі даних за допомогою процесу, що імітує спосіб роботи людського мозку.

Парсер — синтаксичний аналіз — процес зіставлення лінійної послідовності лексем (слів, токенів) природної чи формальної мови з її формальною граматикою. Результатом зазвичай є дерево розбору (синтаксичне дерево).

Помилка нейронної мережі (англ. Loss) — це значення, що визначається функцією помилки, вона виступає дійсним числом, яке характеризує якість вирішення нейронною мережею поставленої задачі.

Рут файли (route files) — автозгенерований файл із набором маршрутизації.

Фреймворк (Framework) — абстракція, в якій програмне забезпечення, що забезпечує загальну функціональність, може вибірково змінюватися додатковим кодом, написаним користувачем, забезпечуючи таким чином програмне забезпечення, яке відповідає вимогам майбутнього застосунку. Можна вважати своєрідною комплексною бібліотекою.

Функція активації штучного нейрону (англ. Activation function) — визначає чи слід активувати нейрон, шляхом обчислення зваженої суми та додаткового додавання до неї зсуву.

Штучний нейрон (англ. Artificial neuron) — це спрощена модель природного нейрону, являє собою складову частину штучної нейронної мережі, та математично представлена як деяка нелінійна функція від лінійної комбінації її вхідних сигналів.

Штучні нейронні мережі (ШНМ) — математичні моделі, а також їх програмні чи апаратні реалізації, побудовані за принципом організації та функціонування біологічних нейронних мереж нервових клітин живого організму.

Keras — це відкритий фреймворк для мови програмування Python, який призначений для глибинного навчання, з версії 2.3 є надбудовою над бібліотекою TensorFlow.

NS — це назва серії симуляторів мережі з дискретними подіями, зокрема NS-1, NS-2 та NS-3, в основному використовуються в дослідженнях і викладанні.

NumPy — це бібліотека Python, яка використовується для роботи з масивами та матрицями.

Pandas — це бібліотека програмного забезпечення, написана мовою програмування Python для обробки та аналізу даних.

Sklearn (Scikit-learn) — один з найбільш широко використовуваних пакетів Python для Data Science та Machine Learning.

TensorFlow — це безкоштовна бібліотека програмного забезпечення з відкритим кодом для машинного навчання та штучного інтелекту,

використовується в низці завдань, але основне завдання полягає в навчанні та використані глибоких нейронних мереж.

V2I (vehicle-to-infrastructure) — це комунікаційна модель, яка дозволяє транспортним засобам обмінюватися інформацією з компонентами, які підтримують систему автомобільних доріг країни.

V2V (vehicle-to-vehicle) — Зв'язок транспортних засобів за допомогою бездротового сигналу для обміну інформації про свою швидкість, місцезнаходження та напрямок.

РОЗДІЛ 1 СУЧАСНА КЛАСИФІКАЦІЯ БЕЗДРОТОВИХ САМООРГАНІЗУЮЧИХ МЕРЕЖ

Мережа ad hoc є набором бездротових мобільних вузлів (або маршрутизаторів), які формують тимчасову мережу без використання будь-якої існуючої мережної інфраструктури або централізованого управління. Маршрутизатори можуть вільно переміщатися випадково і самоорганізуються довільно; Таким чином, топологія бездротової мережі може змінюватися швидко та непередбачено. Така мережа може працювати в автономному режимі або може бути підключена до Інтернету. Багатоскачковість (multihop), мобільність, велика за розміром мережа у поєднанні з неоднорідністю пристроїв, пропускною здатністю та обмеженнями за потужністю батареї роблять розробку адекватних протоколів маршрутизації досить серйозною проблемою. На рисунку 1 показано схему бездротової самоорганізуючої мережі Ad Hoc. А таблиця 1 демонструє різницю між мобільними та Ad Hoc мережами.

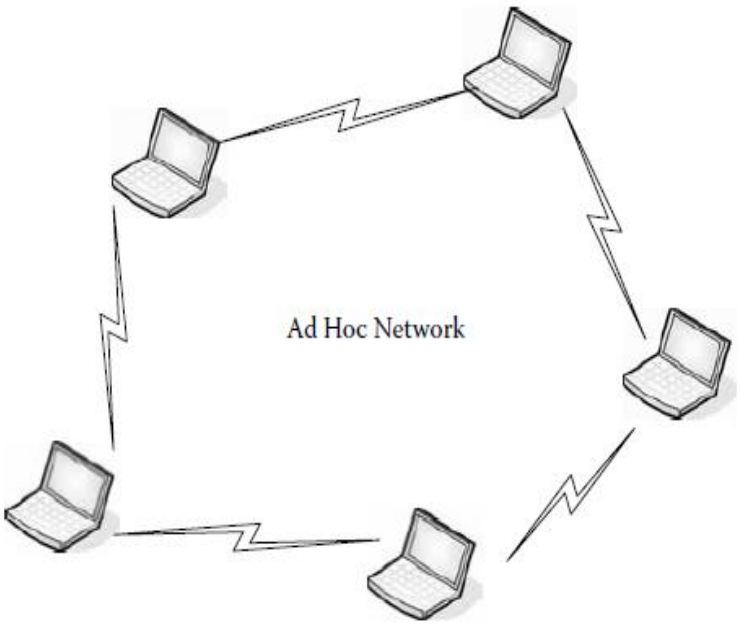

Рисунок 1 — Бездротова самоорганізуюча мережа Ad Hoc

Таблиця 1 — Різниця між мобільною та Ad Hoc мережею

Мобільна мережа	Ad Hoc Мережа
Інфраструктурні мережі	Мережа без конкретної інфраструктури.
Стаціонарна, попередньо розташовані вузли мережі та централізована	Відсутня централізація та швидке розгортання
Використовує топологію "зірка"	Динамічна мережа з multihop
Слабоагресивне середовище та стабільний зв'язок	Не доброзичливе середовище (шум, втрати) можлива втрата з'єднання
Детальне планування перед встановленням базової станції	Ad Hoc мережа автоматично формується та адаптується до змін
Висока вартість установки	Економічно ефективні
Великий час налаштування	Малий час налаштування

1.1 Основні види бездротових мереж, що самоорганізуються, та їх характеристики

Виділяють шість основних видів бездротових мереж, що самоорганізуються:

1. Mesh-мережі або мережі з пористою топологією.
2. Wireless Sensor Network (WSN) або бездротова сенсорна (датчикова) мережа.
3. VANET чи транспортна ad-hoc мережа.
4. MANET чи мобільна ad-hoc мережа.
5. FANET або літаюча ad-hoc мережа.
6. Мережі систем надзвичайного реагування (911 — США, ГЛОНАСС — РФ, eCall — Євросоюз).

При побудові мережі можна скористатися мобільними пристроями, які працюють з технологіями бездротового доступу: Bluetooth, Wi-Fi, ZigBee та іншими [9, 33, 14].

Ad-hoc мережа, що самоорганізується, — це радіомережа. Така мережа не покладається на існуючу інфраструктуру. Програма може бути мобільною, а навколишнє середовище динамічно змінюється, протоколи повинні самостійно адаптуватися до всіх змін середовища. Через свою мобільну природу мережа, що самоорганізується, пред'являє нові вимоги до дизайну [27, 15].

Міжмережева взаємодія має на увазі зв'язок між зовсім різними вузлами чи сегментами мережі. Сегмент як одиниця зони встановлює з'єднання через пристрої-посередники, маршрутизатори чи шлюзи. Таке з'єднання часто здійснюється між приватними та урядовими мережами. Таким чином, об'єднана мережа є комбінацією декількох окремо взятих мереж, з'єднаних проміжними пристроями, які функціонують як одна мережа.

Розглянуті у випускній кваліфікаційній роботі бездротові мережі, що самоорганізуються, мають такі проблеми:

- проблеми з енергоефективністю;
- проблеми із синхронізацією;
- проблеми із забезпеченням безпеки вузлів у мережі.

Для вирішення вищеописаних проблем можна використовувати метод просторового кодування сигналу, що збільшує смугу пропускання каналу (multiple-input and multiple-output, MIMO) [11]. Зазначені вище проблеми тісно пов'язані з проблемами в управлінні мережами, оскільки методи, які застосовуються до мереж, що самоорганізуються, необхідно врахувати динаміку мережі [33]. Крім того, в таких мережах неможливо застосувати методи управління. Оскільки застосовані в традиційних мережах методи, спрямовані переважно на централізоване управління, що суперечить концепції децентралізації.

Як одне із рішень, часто виступає використання властивостей централізованого та децентралізованого підходів в управлінні мережею.

Технологія доступу, яка використовується в мережах, що самоорганізуються, має низьку серйозних недоліків, які призводять до обмежень у застосуванні. В даний час однією з основних проблем такої технології є створення ефективних протоколів маршрутизації, які будуть ефективними у плані масштабованості. Таким чином методи управління, засновані на теорії ігор, набувають все більшого поширення [2].

1.1.1 Переваги Ad Hoc мереж

Існує багато причин, чому краще використовувати ad hoc, ніж звичайну інфраструктуру. Найбільшою перевагою ad hoc мережі є повна незалежність від будь-якої інфраструктури. Тому є можливість встановити ad hoc мережу в будь-яких складних ситуаціях. Серед переваг ad hoc мереж можна визначити:

1. *Відсутність інфраструктури та мала вартість:* Бувають ситуації, в яких користувач комунікаційної системи не може покластися інфраструктуру бо використання послуги з конкретної інфраструктури може бути дорогим для конкретних застосувань [23].

У місцевості з дуже низькою щільністю, наприклад у пустелі, горах або на ізольованій території зазвичай дуже складно або інколи навіть неможливо створити інфраструктуру. Але якщо порівняти, як часто люди там користуються вищезазначеними послугами та скільки даних передається за день то можна зробити висновок про дуже велику вартість встановлення, обслуговування та ремонту.

Майже така ж проблема з військовою мережею. Очевидно, що будувати інфраструктуру на полі бою дуже невигідно, через дуже велику вартість установки та розуміння того що ворог може знищити інфраструктуру в короткі терміни. Саме тому незалежно від ланшафту та поточних проблем місцевості потреба мережі в обох випадках дуже гостра.

2. *Мобільність (*лише mobile ad hoc network, MANET*):* Настає час коли виникає потреба в швидкому розгортанні незалежних мобільних користувачів [23]. Найпопулярнішими прикладами є: військові мережі, надзвичайні ситуації або рятувальні операції, ліквідація стихійних лих. У цих сценаріях ми не можемо покладатися на централізоване підключення. А завдяки тому що вузли підтримки MANET дуже мобільні, ми все ще можемо спілкуватися за допомогою наших мобільних пристроїв доки пункт призначення доступний.

3. *Децентралізовані та надійні:* Ще одна перевага ad hoc мереж полягає в тому, що вони за своєю суттю дуже надійні [29]. Уявіть собі що з якихось причин не працює одна з базових станцій. У цьому випадку всі користувачі цієї базової станції втратять зв’язок до інших мереж.

У Ad Hoc мережах такої проблеми можна уникнути. Якщо один вузол покидає мережу або не працює, ви все ще можете мати підключення до інших вузлів, і, можливо, ви зможете використовувати ці вузли для багаторазового переходу вашого повідомлення до вузлів призначення доти, доки існує принаймні один шлях до потрібного вузла. На рисунку 2 та рисунку 3 показано функціонування Ad Hoc мережі у штатному режимі та після виникнення несправності.

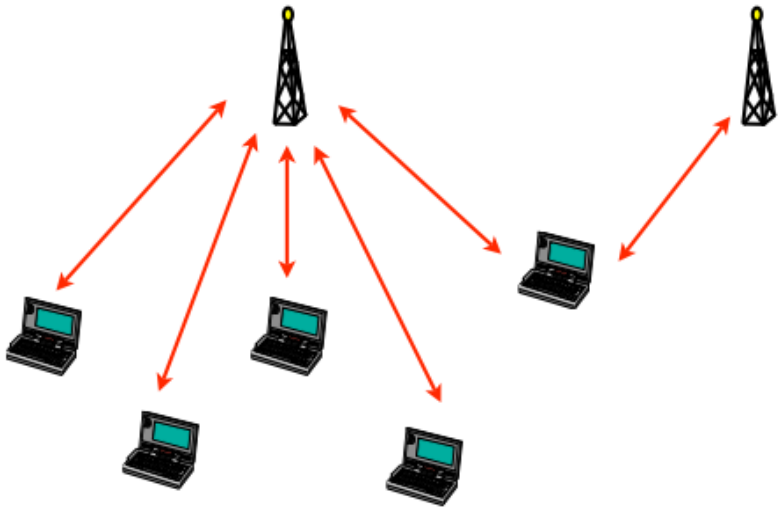

Рисунок 2 — Мережа до виникнення несправності

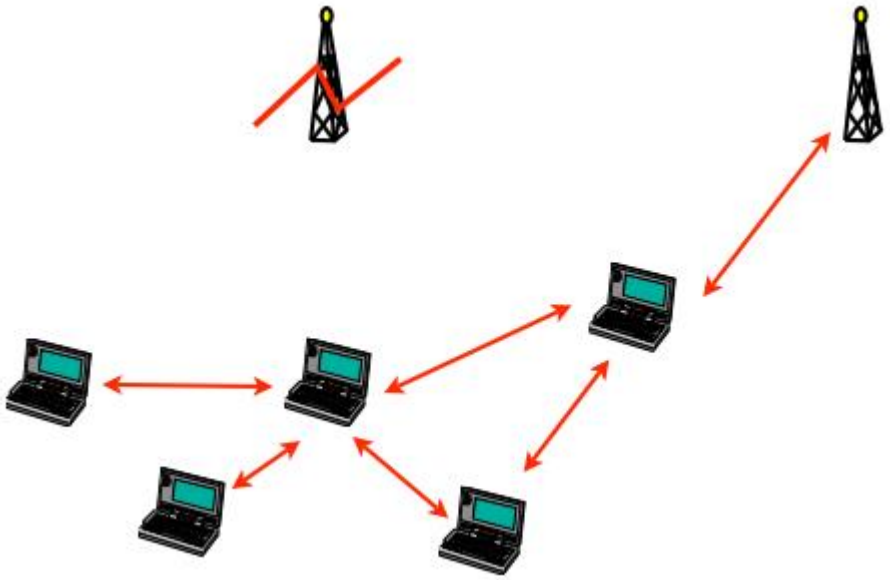

Рисунок 3 — Використання ad hoc мережі після несправності

4. *Простота створення мережі та можливість спонтанної зміни інфраструктури:* Несправності мережевої інфраструктури іноді неможливо уникнути. Очевидно, що усунення несправності складної інфраструктури в короткі терміни є дуже важливим, бо необхідно забезпечити її безперебійну роботу. Створення ad hoc є хорошим компромісом в такій ситуації. Учасники мережі можуть діяти як ad hoc вузли і передавати повідомлення у будь який час.

1.1.2 Недоліки Ad Hoc мереж

Бездротовий зв'язок дуже популярний у наш час, використання бездротового зв’язку може зробити кімнати кращими, оскільки менше використовуються кабелі. Слабкість бездротового з’єднання ad hoc, нижча швидкість передачі даних, безпека та середній контроль доступу — поширені проблеми бездротового зв’язку. Сильні сторони ad hoc також викликають деякі проблеми. Нижче наведено недоліки ad hoc мереж:

1. *Високий рівень помилок*: на відміну від дротової передачі, бездротова може мати справу з проблемою характеристики електронної хвилі. У вільному приміщенні без перешкод електронна хвиля поширюється лінійно незалежно

від частоти. Перешкода викликає затінення, відбиття, розсіювання, згасання, заломлення, дифракцію хвилі. Такий тип передачі може призвести до того, що пакети, які передаються, будуть спотворені, тобто будуть отримані з помилкою.

2. *Низька швидкість передачі даних*: Одна з найбільших проблем ad hoc мереж — це зниження швидкості передачі даних. Характеристика хвилі, яка використовується при бездротовому зв'язку, не дозволяє передавати дані краще, ніж при дротовому. Вища частота може передавати більше даних, але вона більш вразлива до перешкод і добре працює у короткостроковій перспективі.

3. *Динамічна топологія та масштабованість*: Оскільки мережі ad hoc не допускають тих же методів агрегування, які доступні для стандартних протоколів інтернет-маршрутизації, вони схильні до проблем масштабованості [23].

Оскільки вузли MANET є мобільними, маршрутизація змінюється у міру переміщення вузлів. Поточна інформація про підключення повинна поширюватись на всіх учасників мережі. Керуючі повідомлення повинні часто надсилатися по мережі. Збільшена кількість повідомлень, що відповідають за керування мережею, навантажують доступну смугу пропускання. Отже, спеціальні протоколи зазвичай розробляються для зменшення кількості керуючих повідомлень, наприклад, з допомогою збереження поточної інформації.

Хороший алгоритм для ad hoc мереж повинен вміти оцінити та порівняти відносну масштабованість мереж безпосередньо збільшенням кількості та мобільності вузлів. Дуже важливо знати, скільки контрольних повідомлень потрібно, та саме тоді ми зможемо контролювати використання смуги пропускання.

4. *Безпека*: Через динамічно розподілену природу без інфраструктури та відсутність централізованих точок моніторингу Ad Hoc мережі вразливі для різних видів атак. На відміну від дротового каналу, бездротовий канал

доступний як законним користувачам мережі, так і зловмисникам. Саме тому, такі мережі схильні до атак, що варіюються від пасивних атак, таких як підслуховування, та активних атак, таких як втручання. Для MANET присутнє обмеження енергоспоживання та обчислювальних можливостей через обмеження енергії, це призводить до нездатності виконувати алгоритми з великим обсягом обчислень, такі як алгоритми з відкритим ключем.

Пасивна атака означає, що зловмисник не надсилає жодних повідомлень. Зловмисник просто прослуховує канал, тому виявити цю атаку практично неможливо. А активні атаки модифікують, видаляють пакети, відправляють пакети в неприпустимий пункт призначення, через що активну атаку можна виявити.

Існує безліч проблем з безпекою в ad hoc мережі. Нижче наведено деякі проблеми безпеки мереж, побудованих за стандартом IEEE 802.11..

1. Підслуховування (пасивне), нелегітимне прослуховування передач між двома вузлами.
2. Аналіз трафіку (пасивний), перехоплення та вивчення зловмисником пакетів, що передаються.
3. Маскування (активне), зловмисник видає себе за авторизованого користувача системи, щоб отримати доступ до її функцій чи даних користувача або отримати більші привілеї.
4. Відтворення (активне), зловмисник переглядає передачі та повторно передає повідомлення як законний користувач.
5. Модифікація повідомлення (активна), зловмисник змінює оригінальне повідомлення, видаляючи, додаючи, змінюючи його.
6. Відмова в обслуговуванні або переривання (активний), зловмисник перешкоджає або забороняє нормальне використання або управління засобами зв'язку.

На рисунку 4 та рисунку 5 зображений звичайний потік даних та потік даних під час атаки на мережу.

Рисунок 4 - Нормальний потік даних

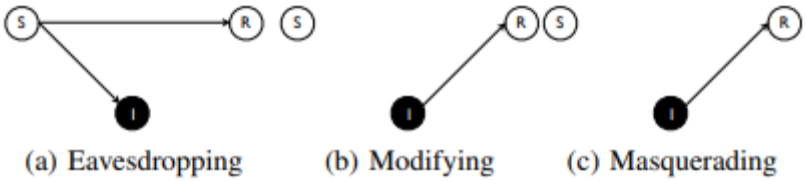

Рисунок 5 — Атаки на безпеку

5. *Обмеження енергії* (лише MANET): Мережа MANET дозволяє мобільним вузлам обмінюватися даними за відсутності стаціонарної інфраструктури, тому що вони працюють від батареї. Через ці обмеження вони повинні мати алгоритми, які є енергоефективними, а також працюють з обмеженими ресурсами обробки та пам'яті. Використання доступної пропускної спроможності буде обмежено, оскільки вузли можуть бути не в змозі пожертвувати енергією, що споживається під час роботи на повній швидкості каналу.

Також дуже дратує, коли під час передачі даних від джерела, яке використовує LAB (Lead-acid battery) батарею, яка практично розряджена. Названий сценарій призводить до того, що процес передачі призупиняється, тому необхідно повторити процес передавання вже після повної підзарядки, саме тому MANET не може виконувати роль постійної мережі.

1.2 Класифікація бездротових мереж, що самоорганізуються

Бездротові мережі, що самоорганізуються, відрізняються від традиційних кабельних мереж, проте, в порівнянні з ними, вони пропонують набагато меншу смугу пропускання; отже, їх конструкція потребує значно більшої уваги. Більше того, умови каналу, що постійно змінюються і

непередбачувані, приховані і відкриті проблеми вузлів, змінне навантаження на мережу, змінна продуктивність пристрою, різні діапазони передачі і спрацьовування, а також мобільність однорангових мереж роблять це завдання ще складнішим.

Оскільки зв'язок між різними пристроями дозволяє забезпечити унікальні та інноваційні послуги, цей зв'язок є досить сильним, але також складним та грубим механізмом, що призводить до безлічі складнощів у сучасних реаліях. Це не тільки ускладнює роботу в мережі, але й обмежує її гнучкість.

Розвиток бездротових мереж, що самоорганізуються, проходить під контролем Інституту інженерів електротехніки та електроніки IEEE. Все, що має відношення до бездротових мереж, бездротові стандарти та мережеве обладнання, контролює робоча група WLAN, до якої входять більш ніж 100 представників різних університетів та компаній-розробників обладнання. За основу для вдосконалення старих та створення нових стандартів беруться останні дослідження та досягнення в області комп’ютерних технологій [14].

Сьогодні існує безліч стандартів для підключення різних пристроїв. У той же час, кожен пристрій має підтримувати більше ніж один стандарт, щоб бути сумісним з іншими пристроями [15].

Стрімке зростання, створення та використання бездротових сенсорних мереж розпочалося наприкінці 90-х років. Останні кілька років комунікаційні технології надзвичайно розвинулися. Потреби людей у комунікації збільшилися, це викликало гостру необхідність використовувати бездротові мережі у повсякденному житті. Моніторинг навколишнього середовища, охорона здоров'я та багато іншого може стати прикладом тісного зв'язку суспільства та бездротових мереж [10].

Використання бездротових мереж зумовлено величезним потенціалом в будь-якій галузі нашого життя. Для задоволення потреб людини ця галузь зв'язку має поступово перетворилася з традиційної щодо безпечної телекомунікаційної мережі, на мережу наступного покоління (The next-

generation network, NGN). Мобільна ad-hoc мережа MANET у зв'язку з сенсорною мережею WSN часто використовуються в цивільних сферах життя. Крім того, в останнє десятиліття вчені з усього світу все частіше наголошують на необхідності розгорнути мережу WSN-MANET для створення на її основі «розумного міста» [25].

Бездротова мережа, що самоорганізується — це децентралізована бездротова мережа, яка не покладається на існуючу інфраструктуру. Програма може бути мобільною, а навколишнє середовище може динамічно змінюватися. Отже, протоколи повинні самостійно налаштовуватись, щоб адаптуватися до змін середовища, трафіку та завдань. Через свою мобільну природу мережа, що самоорганізується, пред'являє нові вимоги до дизайну. По-перше, це самоконфігурація. На рівні додатків вузли мережі зазвичай спілкуються та співпрацюють як команда (наприклад, поліція, пожежники, команди медичного персоналу у пошуково-рятувальних операціях). Таким чином, такого роду додаткам потрібен ефективний груповий зв'язок (багатоадресне розсилання) як для даних, так і для трафіку в реальному часі. Основною відмінністю від дротових та керованих бездротових мереж, керування в яких виконують маршрутизатори або точки доступу, усі дії в Ad-hoc мережах відбуваються незалежно.

У цих мережах виділяють:

1. Незалежна конфігурація.
2. Інфраструктурна конфігурація.

Ці конфігурації особливо не різняться між собою, однак вони впливають на такі показники мережі, як:

- кількість вузлів, що підключаються;
- радіус дії мережі;
- завадостійкість і т.д.

Незалежно від конфігурації мережі, стандарти визначають один із типів протоколу доступу та специфікації фізичного каналу.

1.2.1 Режим незалежної конфігурації

Під режимом незалежної конфігурації (рисунок 6) або незалежним базовим набором служб (IBSS) мається на увазі підключення типу "точка-точка". Такий режим є одним із найпростіших у застосуванні, побудові та налаштуванні мережі.

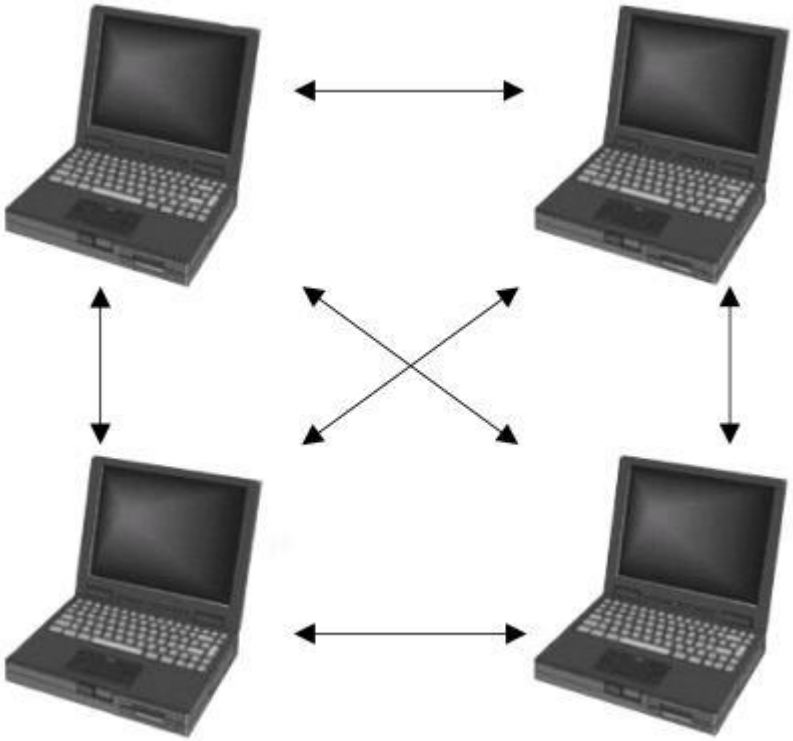

Рисунок 6 — Режим незалежної конфігурації

Щоб об'єднувати пристрої в бездротову мережу, необхідно обладнати їх адаптером бездротового зв'язку. Більшість переносних комп'ютерів вже укомплектовані вбудованими адаптерами бездротового зв'язку, отже завдання побудови мережі зводиться до нормального настроювання ресурсів та обмежень [31].

Такий спосіб використовується в основному для організації тимчасової мережі та має відповідні плюси та мінуси. З плюсів можна виділити швидку розгортку мережі за будь-яких умов. До мінусів можна віднести низьку швидкість обміну даними, яка ділиться на всіх учасників мережі. Режим незалежної конфігурації відмінно підходить лише тоді, коли необхідно

швидко об'єднати два або кілька пристроїв для передачі відносно невеликого обсягу даних [21].

1.2.2 Інфраструктурна конфігурація

Інфраструктурна конфігурація — відносно нова інфраструктура мережі, що швидко розвивається. Цей режим має багато переваг, серед яких можна виділити:

1. Можливість підключення кількох користувачів.
2. Стійкість до перешкод.
3. Високий рівень контролю підключень.

Крім того, можливість використовувати комбіновану топологію мережі та провідні сегменти [21].

Цей режим роботи бездротової мережі (рисунок 7) передбачає використання точок доступу, які виконують роль центрального керуючого вузла. Такий принцип організації є гнучким та досить ефективним. Провівши аналогію з дротовими мережами, можна побачити, що інфраструктурна конфігурація практично повторить топологію «зірка».

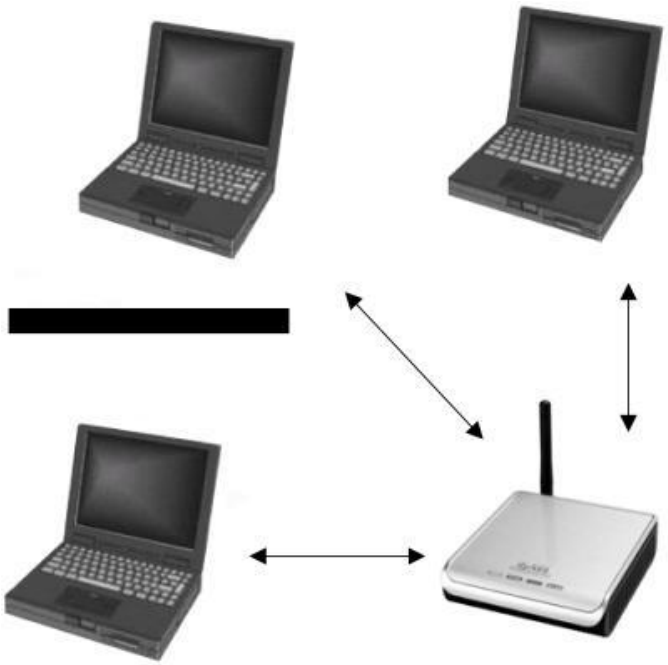

Рисунок 7 — Режим базового набору служб

Звичайно ж, однією точкою доступу мережа може не обмежуватися, що зазвичай відбувається в міру зростання мережі. У такому випадку базові набори служб утворюють одну єдину мережу. Таку конфігурацію мережі вже називають розширеним набором служб, вона зображена на рисунку 8.

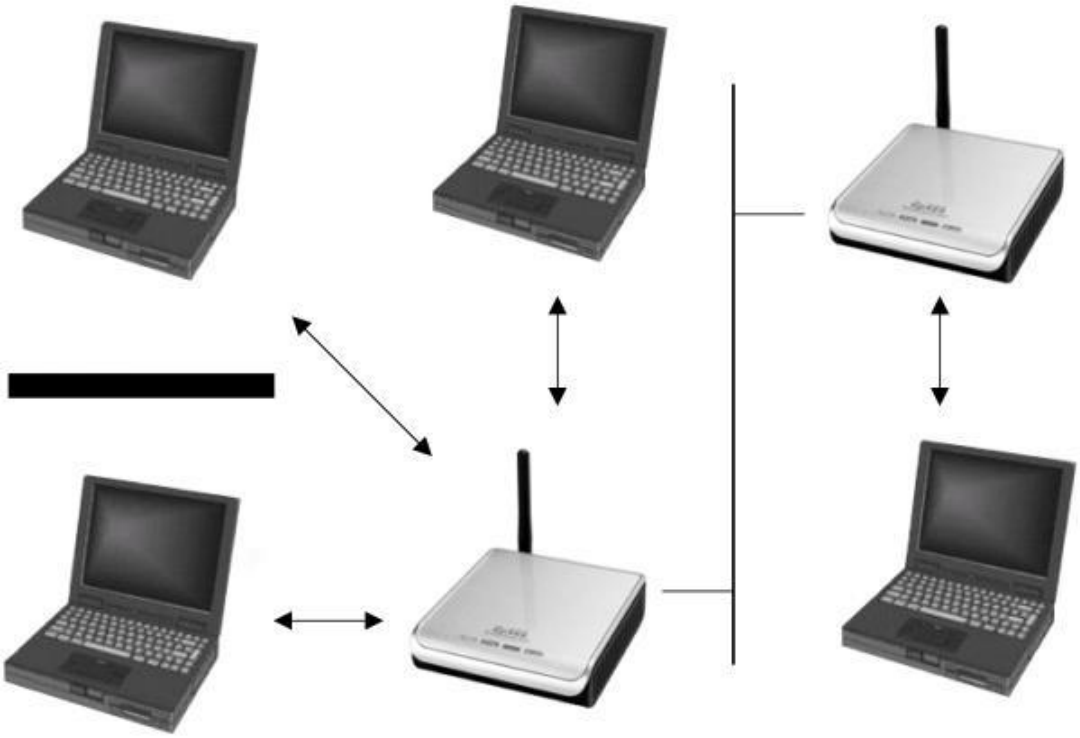

Рисунок 8 — Розширений набір служб

У такому разі точки доступу обмінюються між собою інформацією, що передається за допомогою провідного з'єднання або за допомогою радіомостів, що дозволяє ефективно організовувати трафік між сегментами мережі. Використання кількох точок доступу дозволяє обмінюватися інформацією між ними. Крім того, адаптери самостійно вирішують до якої точки доступу їм підключатися, що дозволить отримати більш стійкий зв'язок.

1.3 Бездротові транспортні мережі з динамічною топологією VANET

Автомобільні бездротові мережі з динамічною топологією VANET отримали широке визнання через забезпечення більшої зручності та

ефективності для водіїв з численними додатками, починаючи від безпеки, ефективності транспорту та закінчуючи інформацією та розвагами. У VANET задіяно багато джерел інформації для забезпечення багатофункціонального середовища інтелектуальних транспортних систем (ITC), таких як транспортні засоби, оснащені бортовими блоками OBU (onboard unit), придорожні блоки RSU (roadside unit), базові станції BS (base station), транспортна інфраструктура (світлофори, камери відеоспостереження), дані GPS та інші. Транспортні засоби (OBU) оснащені додатками та датчиками з можливістю обробки, системою локалізації, такою як глобальна система позиціонування GPS, та радіопередавачем для бездротового доступу до автомобільного середовища. Крім того, OBU також обладнані широкосмуговим радіопередавачем, таким як WiMax/3G/4G LTE, для зв'язку з базовою станцією стільникового зв'язку. Прямий бездротовий зв'язок від транспортного засобу до транспортного засобу дозволяє обмінюватися даними навіть там, де немає інфраструктури зв'язку, такої як базові станції стільникових телефонів або точки доступу бездротової мережі.

1.3.1 Опис мережі

Ключову роль мережі VANET грає транспортний засіб (ТЗ), англ. vehicle, будучи оснащеним пристроями телекомунікацій (ПТК), що утворює локальні мережі всередині транспортних засобів, що позначається як автономні телекомунікаційні мережі (АТКМ). ТЗ разом із розташованими в ній ПТК, з'єднаними до АТКМ, також позначається як апаратно-програмний комплекс (АПК).

Під мережею зв'язку (МЗ) транспортних засобів (ТЗ) мається на увазі система зв'язку між апаратно-програмними комплексами, створеними на базі транспортних засобів (vehicle) та оснащених пристроями телекомунікацій. Якщо така мережа є такою, що самоорганізується, вона позначає її як ad hoc мережу транспортних засобів, що є аналогом прийнятого терміну VANET — vehicular ad hoc network (рисунок 9). Особливістю МЗ ТЗ є відсутність опорної

стаціонарної мережі, тому що сфера застосування таких мереж може бути найрізноманітнішою (тундра, тайга, болота, степ, пустеля і т.і., де технічна можливість розміщувати базові станції операторів стільникового зв'язку може бути відсутня протягом багатьох кілометрів). І навіть якщо опорна мережа присутня, МЗ може функціонувати і без неї, використовуючи маршрутизацію ad hoc. Незважаючи на можливість присутності виділених ТЗ, що в цілому використовуються клієнтами зв'язку, базовими станціями та навіть серверами, що надають послуги зв'язку в даному випадку, проте також існують апаратно-програмні комплекси, створені на базі використовуваних в мережі транспортних засобів [9].

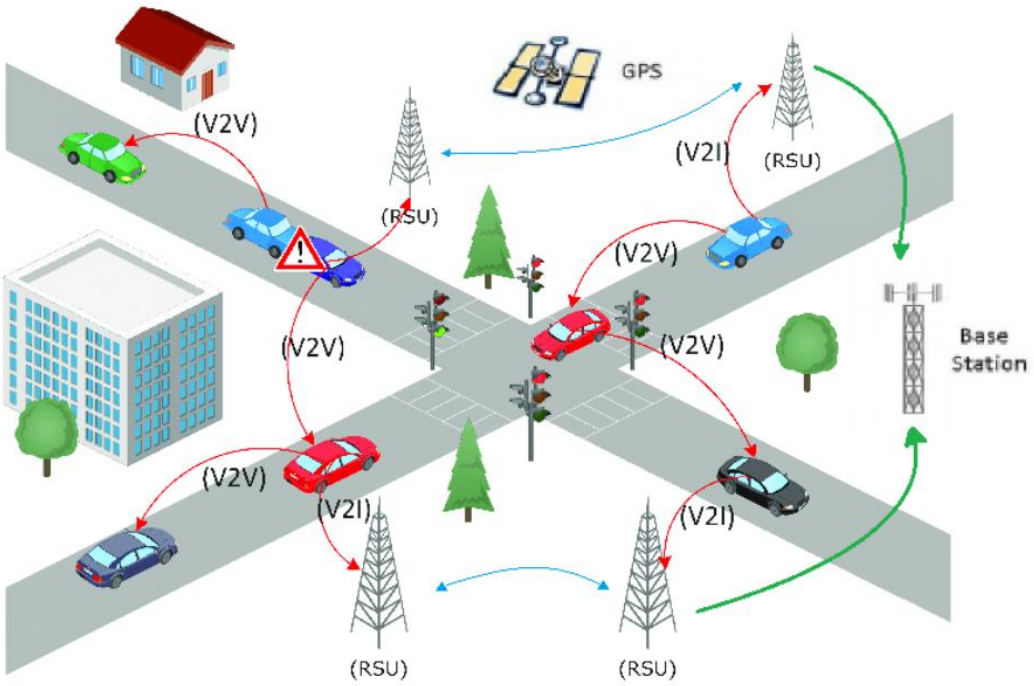

Рисунок 9 — Приклад VANET — ad hoc мережі транспортних засобів

RSU встановлюються вздовж дорожніх мереж і є стаціонарною інфраструктурою, такою як системи відеоспостереження, мережеві камери, камери контролю смуги руху, світлофор, датчики і т. д. Кожен кластер RSU підключений до станції управління RSU (RSUC) для зберігання та обробки даних перед відправкою в єдину хмарну інфраструктуру VANET.

При цьому обробка даних у хмарах може давати затримки. Це пов'язано з тим, що центри обробки даних розташовані досить далеко від кінцевих пристроїв. У зв'язку з цим актуальним завданням є перенесення частини обчислювальних операцій ближче до кінцевих пристроїв. Цю ідею реалізує концепція граничних обчислень MEC за допомогою програмно-конфігурованих мереж SDN.

Основна ідея Edge computing полягає в розташуванні обчислювальних потужностей географічно розподілених обчислювальних пристроях. Таким чином, на кордоні будуть оброблені дані, чутливі до затримок, а дані великого трафіку будуть оброблені у хмарі. Програмно-конфігуровані мережі SDN знімуть частину функцій управління та фізичної передачі з маршрутизаторів та комутаторів, зменшуючи навантаження. Управління всієї мережею буде реалізовано на центральному контролері.

Таким чином, за допомогою Edge computing і SDN гібридна модель ефективного представлення даних VANET, що розробляється, буде мати мінімальні затримки.

Архітектура VANET повинна забезпечувати зв'язок між прилеглими транспортними засобами, а також між транспортними засобами та придорожнім обладнанням, що призводить до наступних типів зв'язків (рисунок 10). Основними типами зв'язків є "ТЗ — інфраструктура" (V2I) або "інфраструктура — ТЗ" (V2I), а також "ТЗ — ТЗ" (V2V):

— V2V (рисунок 11) дозволяє організовувати прямий автомобільний зв'язок без підтримки фіксованої інфраструктури та може в основному використовуватись для додатків безпеки, захисту та поширення;

— V2I (рисунок 12) дозволяє ТЗ зв'язуватися з придорожньою інфраструктурою в основному для додатків збору інформації та даних;

— Гібридний зв'язок (рисунок 13) дозволяє об'єднувати як V2V, так і V2I. У цьому випадку ТЗ може зв'язуватися з придорожньою інфраструктурою або в режимі одиночного або множинного переходу, залежно від відстані.

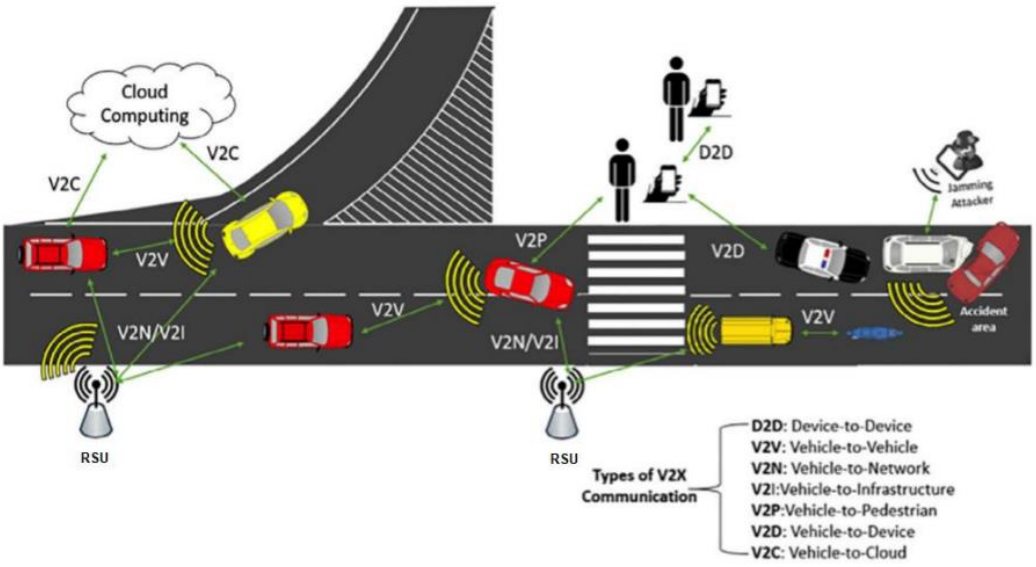

Рисунок 10 — Інтелектуальна транспортна система VANET.

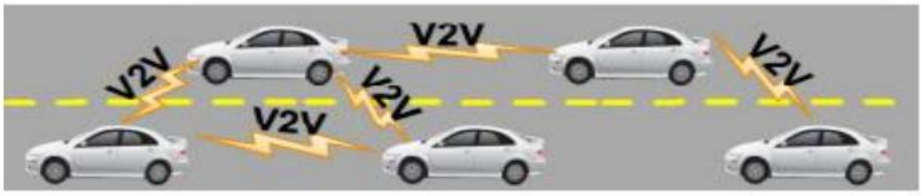

Рисунок 11 — Тип передачі даних V2V в VANET

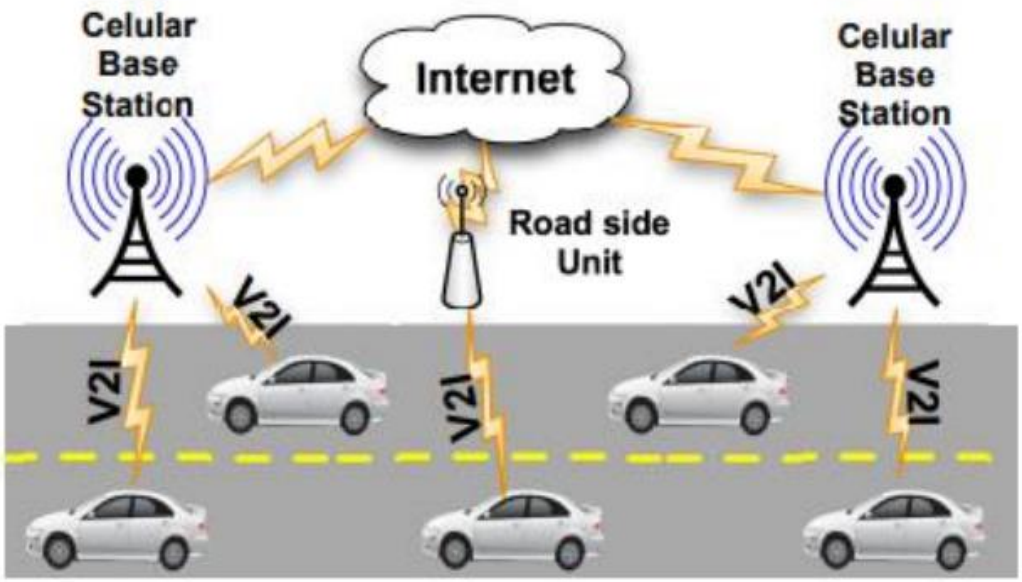

Рисунок 12 — Тип передачі даних V2I в VANET

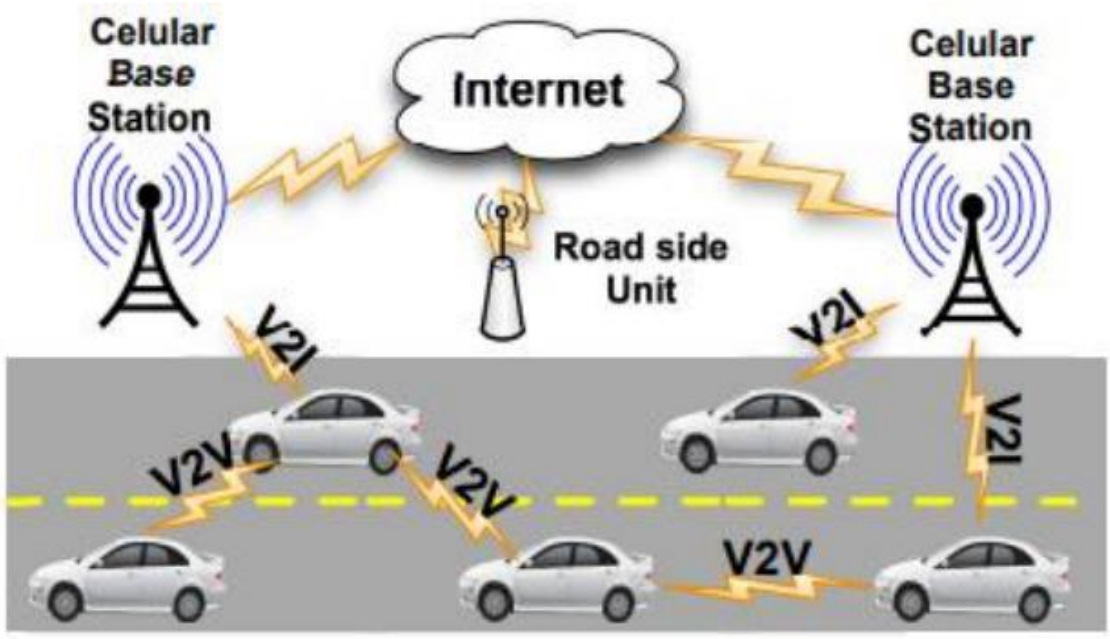

Рисунок 13 — Гібридний зв'язок в VANET

При розробці архітектури системи необхідно враховувати, як дані будуть передаватися в мережі, а для VANET основним джерелом інформації є програми AU (application unit). AU — це пристрої ТЗ (транспортний засіб), які

виконують функції забезпечення комунікаційних можливостей OBU. AU може бути спеціалізованим пристроєм для програм безпеки або звичайним пристроєм, таким як персональний цифровий помічник (PDA).

Додатки AU мають різні вимоги, такі як пропускна здатність, затримка, безпека та надійність. Системи VANET, як правило, підтримують ряд додатків, включаючи дорожні сигнали, покращення зору, погодні умови, безпеку та розваги. Більшість додатків можуть бути розділені на дві категорії: додатки, пов'язані з безпекою, та додатки, пов'язані з комфортом.

Гібридна модель ефективного розміщення даних VANET повинна відображати реалістичний зв'язок між ТЗ та сервером, який представлений на рисунку 14. Зазначимо, що додатки для підключених ТЗ обмінюються інформацією за допомогою мобільного зв'язку із серверами. Кожен ТЗ або сервер працює в ізольованому середовищі. Модель обчислювальної мережі представляє собою взаємодію контейнерів МС (Mobile communications) та контейнера сервера. Програми цієї моделі розділені на 2 типи.

1. Додаток користувача: генерує трафік, пов'язаний з розважальними послугами, доступними в ТЗ.

2. Автомобільна програма: пов'язана з роботою ТЗ, наприклад, телеметрією або оновленням.

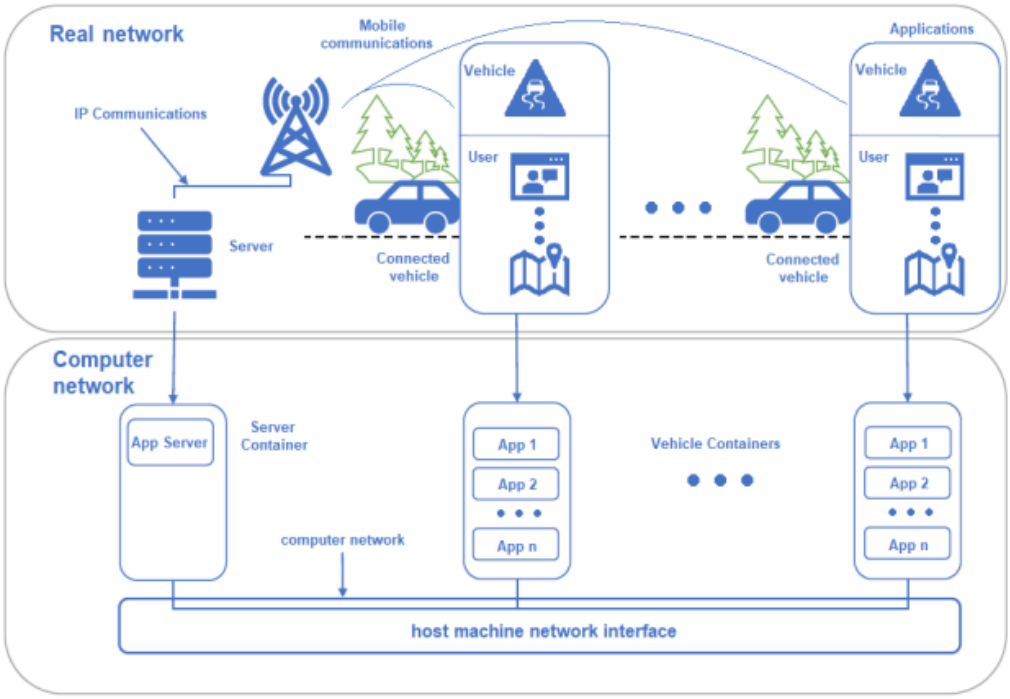

Рисунок 14 — Архітектура взаємодії ТЗ та їх додатків у мережі VANET

Таким чином, архітектура і принципи взаємодії ТЗ в мережі VANET додається до моделі розміщення вихідних і проміжних даних, підходи програмно-конфігурованих мереж і інструментів проведення граничних розрахунків для підвищення загальної ефективності і продуктивності.

1.3.2 Класифікація інтелектуальної транспортної системи VANET

VANET відрізняється від інших бездротових мереж наступними особливостями [28, 3]:

1. динамічна топологія: в VANET вузли рухаються з порівняно високою швидкістю, можуть змінювати напрямок руху непередбачуваним чином, у результаті топологія мережі часто змінюється;

2. нерівномірність щільності вузлів: зазвичай, щільність розташування транспортних засобів на трасі нерівномірна, залежить від часу та від місцевості. Наприклад, у нічний час щільність транспортного потоку нижче, ніж удень; у віддалених малонаселених районах транспортних засобів значно менше, ніж у містах;

3. обмеження руху: будемо вважати, що рух автомобілів обмежено трасами і прилеглої до них території;

4. наявність перешкод (будівель, споруд тощо): в VANET рух вузлів здійснюється по проїжджій частині дороги, яка, як правило, оточена висотними будинками, деревами (у містах), що створює перешкоду для поширення радіохвиль;

5. відсутність єдиного центру управління та контролю за топологією: VANET є децентралізованими мережами, що об'єднують вузли на великих територіях кілька десятків чи сотень квадратних кілометрів. При цьому неможливо виділити єдиний центр (базову станцію), за допомогою якого можна було б організувати та підтримувати топологію, протоколи безпеки (обмін криптографічними ключами та сертифікатами для автентифікації), синхронізацію пристроїв. Очевидно, що протоколи та додатки, що функціонують у VANET, повинні бути адаптованими, підтримувати самоорганізацію, додавання нових пристроїв та видалення старих;

6. нерівномірність комунікаційного трафіку та проблеми забезпечення якості обслуговування та безпеки: оскільки транспортний потік нерівномірний, обсяги інформації, що передається, також можуть змінюватися з часом. Сучасні розважальні програми, Інтернет-телебачення, онлайн-ігри тощо. можуть викликати відмову мережі в обслуговуванні. При цьому необхідно врахувати, що різні служби мають різні вимоги щодо якості обслуговування та безпеки. Інформація передається по відкритому радіоканалу, тому очікуються наявність перешкод, ненавмисні та цілеспрямовані атаки на окремі служби та мережу в цілому з боку користувачів, хакерів, хуліганів тощо.

1.4 Політика безпеки VANET та існуючі загрози

1.4.1 Вимоги безпеки

Зв'язок VANET має бути безпечним і мати гарантію, що передане повідомлення не буде додано або змінено зловмисниками. У мережі зв’язок здійснюється між V2V і V2I. В обох типах комунікацій вузли збирають інформацію з інших вузлів або з RSU, яка має бути надійною. Перш ніж приймати будь-яку інформацію, необхідно перевірити автентичність відправника, який переслав дані. Обробкою розподілу ключів важко керувати через високий рівень мобільність транспортного засобу. Кожне повідомлення шифрується, і його потрібно розшифрувати на стороні одержувача за допомогою того самого або іншого ключа. Різні виробники можуть використовувати для цього різні рішення, що є дуже складним завданням. Динамічні зміни відбуваються в топології автомобілів через їх швидкий рух. Дані повинні передаватись через оптимізовану систему шляху. Тут необхідно вибрати автентифікований оптимізований шлях для безпечного зв’язку.

Нижче наведено деякі вимоги до інформаційної безпеки, сформульовані у роботі Gongjun Yan [32].

1. аутентифікація — головна вимога до інформаційної безпеки (ІБ) у VANET. Це викликано необхідністю захисту від різних повідомлень, відправлених від неіснуючих вузлів (наприклад, атака Сівіли) або від вузлів, які видають себе за існуючі. Атака може бути і від реального вузла, коли водій спрямовує фальшиві повідомлення про корки на дорозі для звільнення дороги проїзду;

2. цілісність повідомлень, що гарантує відкидання фальшивих повідомлень;

3. забезпечення неможливості відправника відмовитися від переданих їм повідомлень;

4. контроль доступу, який дозволяє гарантувати, що всі вузли функціонують відповідно до наданих їм повноважень та привілеїв;

5. конфіденційність повідомлень у випадках, пов'язаних із підозрами у криміналі;

6. забезпечення захисту приватної інформації користувача.

Необхідно зауважити про існуючі обмеження зв’язку V2V та V2I, такі як пропускна здатність, безпека, конфіденційність транспортних засобів та участь транспортних засобів.

1.4.2 Види загроз

Атаки на VANET можна умовно розділити на наступні основні категорії:

- атаки, що становлять загрозу доступності;
- атаки, що становлять загрозу автентичності;
- атаки, що становлять загрозу конфіденційності водія;
- атаки, що становлять загрозу цілісності.

Таблиця 2 демонструє перелік різних видів існуючих загроз з їх типом поведінки у мережі, вимоги безпеки, що порушуються та їх вплив на мережу

Таблиця 2 — Атаки Vanet та їх вплив на мережу

Назва атаки	Активний/ Пасивний	Вимоги безпеки	Вплив на мережу
(DOS) Атака	Активний	Доступність	Високий
(DDoS) Атака	Активний	Доступність	Високий
Атака Сивілли	Активний	Аутентифікація	Середній
Атака імітації вузла	Активний	Цілісність	Середній
Підслуховування	Пасивний	Конфіденційність	Середній
Маскування	Активний	Аутентифікація	Високий
Спуфінг GPS	Активний	Автентифікація, відстеження	Середній
Атака грубою силою (Брутинг)	Активний	Конфіденційність та автономність	Високий
Пранкинг	Активний	Цілісність	Високий
Програмна атака на захищені та не захищені повідомлення	Активний	Доступність, цілісність	Високий
Атака чорної діри	Активний	Доступність	Високий
Атака червоточини	Активний	Доступність	Високий
Атака сірої діри	Активний	Доступність	Високий

Загрози доступності

1. Атака відмови в обслуговуванні (DOS). У DOS головною метою є запобігання легальному користувачеві доступу до мережевих служб і мережевих ресурсів (рисунок 15). DOS атака може відбутися шляхом блокування системи каналів, так що жоден справжній транспортний засіб не

зможе отримати до неї доступ [13]. У VANET це найбільш серйозна проблема, оскільки користувач не може спілкуватися в мережі та передавати інформацію іншим транспортним засобам, що може призвести до більшого руйнування важливих частин системи. Зловмисник може пройти через три різні шляхи щоб досягти такого результату:

1. На базовому рівні зловмисник перевантажує ресурс вузла, щоб він не зміг виконувати інші необхідні завдання, що призводить до того, що вузол постійно зайнятий та не береться за виконання інших робіт.
2. На розширеному рівні зловмисник блокує канал, генеруючи високу частоту всередині самого каналу, тому жоден транспортний засіб не може спілкуватися з іншим транспортним засобом у мережі.
3. Відкидання пакетів.

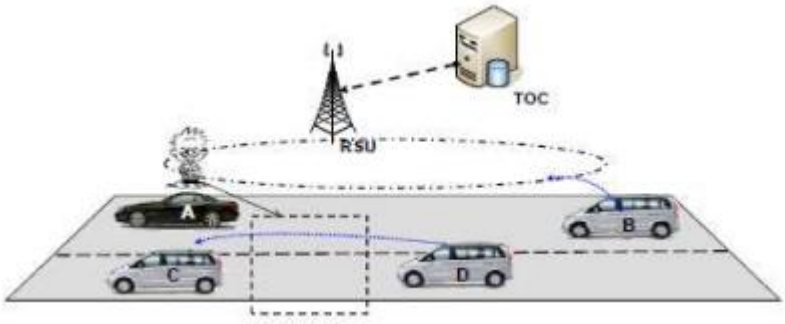

Рисунок 15 — Атака DOS між V2V та V2I

2. Розподілена DOS (DDoS) атака. Атака DDoS є більш серйозною, ніж атака DOS, оскільки вона викликана з кількох точок (рисунок 16). У цьому випадку зловмисники використовують різні місця, щоб почати атаку, також вони можуть використовувати різний часовий інтервал для надсилання повідомлення. Або часовий інтервал і характер повідомлення можуть відрізнятися від цільового автомобіля до автомобіля зловмисників. Основна мета полягає в тому, щоб зробити мережу повністю не функціональною, тобто зробити мережу недоступною для користувачів [13]. Існують два варіанти DDoS-атак:

a. Транспортний засіб до автомобіля (V2V);

b. Транспортний засіб до інфраструктури (RSU).

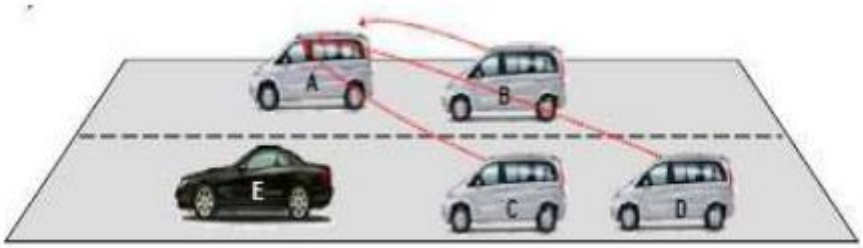

Рисунок 16 — Атака DDoS за варіантом V2V

3. Чорна діра. Під час загрози вузол відмовляється брати участь у мережі, або він повністю випадає з утворення так званої "чорної діри". При цьому весь трафік мережі перенаправляється на конкретний вузол, якого насправді не існує, це призводить до втрати великого обсягу даних. Атаки цього типу особливо ефективні, коли зазначений вузол є одночасно посередником та збірним пунктом. Гнучке налаштування шкідливого коду може призвести до того що він буде вирішувати, чи буде пакет відкинутий, чи використовуватиме його місце на маршруті як перший крок у атаці "людина посередині". На рисунку 17 показано приклад, коли Автомобіль-А хоче надіслати пакети даних до Автомобіля E та Автомобіля G, але він не має жодних деталей маршруту до обох.

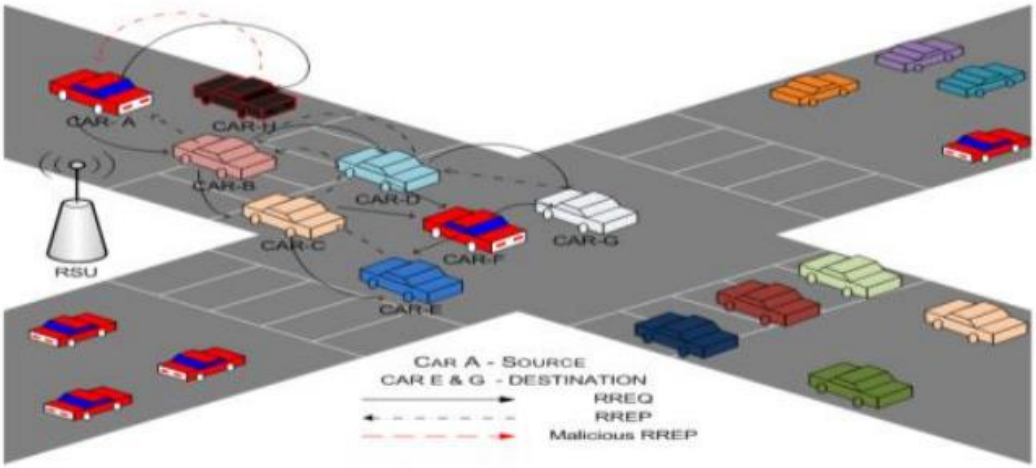

Рисунок 17 — Атака типу "чорна діра" у VANET

Таким чином, Автомобіль А ініціює процес виявлення маршруту, а RREQ (запит створення маршруту) пересилається до Автомобіля B і Автомобіля H. Будучи зловмисним вузлом, Автомобіль H стверджуватиме, що він має найкоротший маршрут до Автомобіля E та Автомобіля G. На основі доступної відповіді Автомобіль А надсилатиме всі повідомлення в автомобіль H і стає жертвою атаки чорної діри.

4. Атака червоточини. Це також один із типів атаки маршрутизації, під час якої шкідливий вузол зловмисника отримує пакет даних від законного користувача в будь-якій точці мережі, тунелює їх і пересилає до іншої точки мережі. Тунель, створений між двома шкідливими вузлами, називається "атакою через червоточину". На рисунку 18 показано атаку через червоточину у VANET.

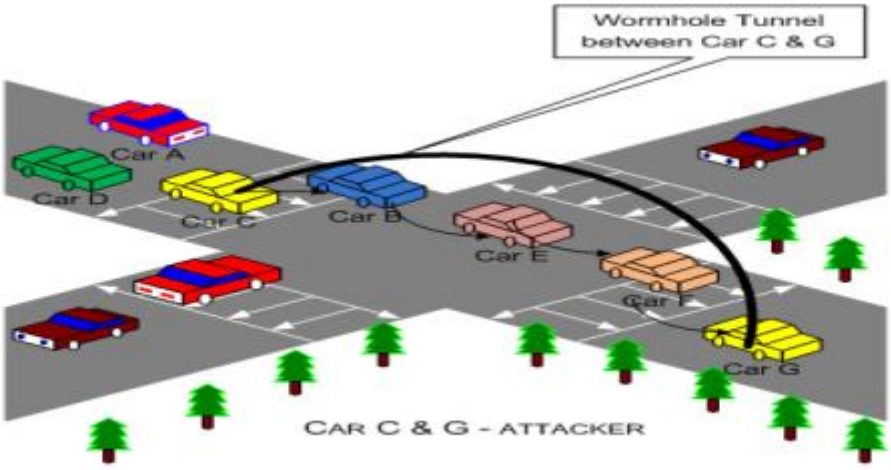

Рисунок 18 — Атака типу червоточина у VANET

5. Атака сірої діри. Ще один типовий представник атаки маршрутизації, також відомий як розширення атаки на чорну діру, у якій замість видалення всіх пакетів відкидаються лише вибрані пакети. Виявити таку атаку дуже важко, оскільки вона не є безперервною. Вона створюється лише на певний час і лише для певного типу пакетів. На рисунку 19 показано приклад, коли Автомобіль А хоче надіслати пакети даних до Автомобіля Е та Автомобіля G, але він не має жодних деталей маршруту для обох.

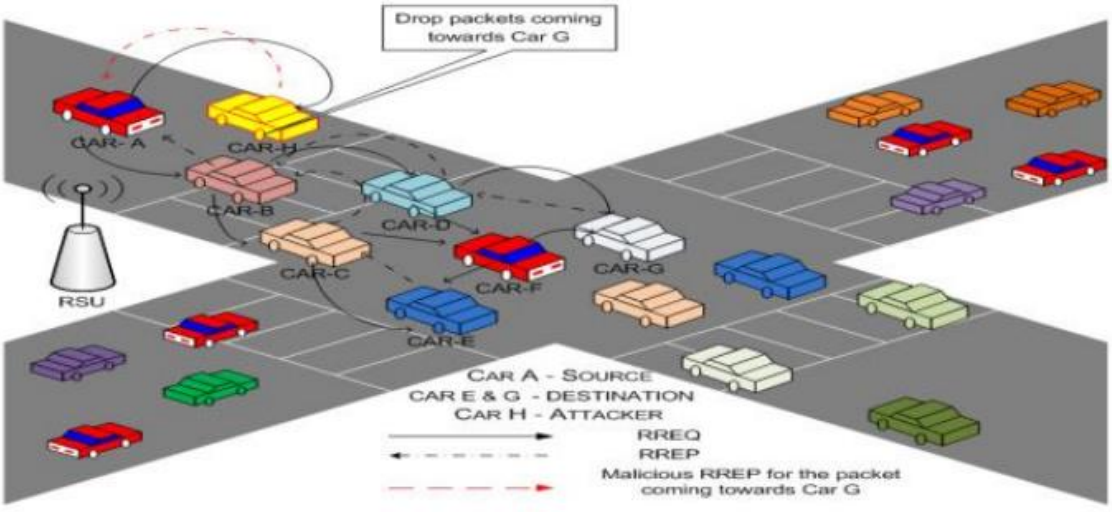

Рисунок 19 — Атака сірої діри у VANET мережі

Таким чином, Автомобіль A ініціює процес виявлення маршруту, а RREQ пересилається до Автомобіля B і Автомобіля H. Як шкідливий вузол, Автомобіль H зацікавлений лише в тому, щоб скидати пакети, що надходять до Автомобіля G, і тому він стверджуватиме, що має найкоротший шлях для досягнення в автомобілі G. На основі доступної відповіді автомобіль A надсилатиме всі повідомлення для автомобіля G через автомобіль H і стає жертвою атаки сірої діри.

6. Атаки зловмисного програмного забезпечення. Даний тип атак схожий на віруси, оскільки вони як і віруси перешкоджають нормальній роботі мережі. VANET зазвичай заражається цими атаками, коли є оновлення програмного забезпечення в блоках VANET або RSU [12].

Загрози автентифікації

1. Напад Сивілли. У атаці Sybil зловмисник створює кілька ідентифікацій вузлів, які поширюють неправильну інформацію в мережі. У цьому типі атак дані транслюються зі сфабрикованою ідентичністю. Цей тип атаки здійснюється OBU (пристрій геолокації) зловмисника на інший законний OBU для отримання різних переваг. У цій атаці транспортний засіб зловмисника створює кілька ідентифікацій і надсилає повідомлення законному користувачеві, ніби на вибраній дорозі рух більше, тому краще змінити маршрут. Зловмисник створить одну ілюзію та на той самий автомобіль надішле повідомлення подібного типу. Тепер законний користувач отримуватиме повідомлення такого ж типу, і через ілюзію, що повідомлення надсилає інший відправник, транспортний засіб змінить маршрут, повіривши в це. Це рішення є вигідним для зловмисника, і тепер автомобіль зловмисника отримає чіткий маршрут під час вибраної поїздки. Цей тип атаки також використовуватиметься для перенаправлення користувача в неправильне місце. На рисунку 20 зображено атаку Sybil, під час якої зловмисник який знаходиться у Car C створює кілька ідентифікаційних даних і надсилає повідомлення іншому користувачеві про те, що на дорозі інтенсивний трафік.

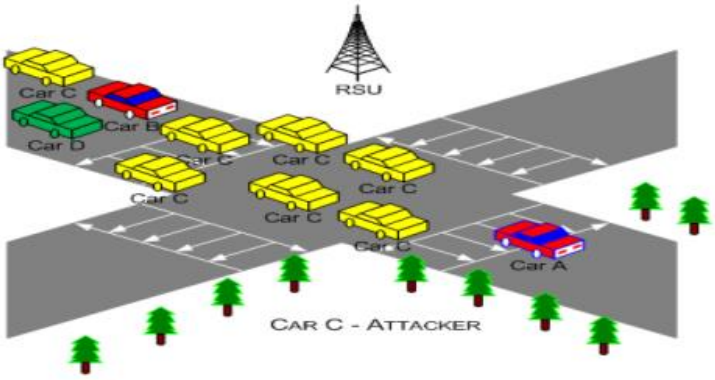

Рисунок 20 — Напад Сивілли у VANET мережі

Отже, отримавши такі повідомлення, автомобіль B і автомобіль D виберуть інший альтернативний шлях, і тепер автомобіль C отримає вільну дорогу.

2. Атака імітації вузла. У Node Impersonation Attack зловмисник оновлює повідомлення та стверджує, що повідомлення надійшло з оригінального автентифікованого джерела. На рисунку 21 можна побачити як транспортний засіб D надсилає повідомлення про аварію в місці x для отримання допомоги, але вузол атакуючого C оновить повідомлення та перешле його до швидкої допомоги, що аварія сталася в місці Y.

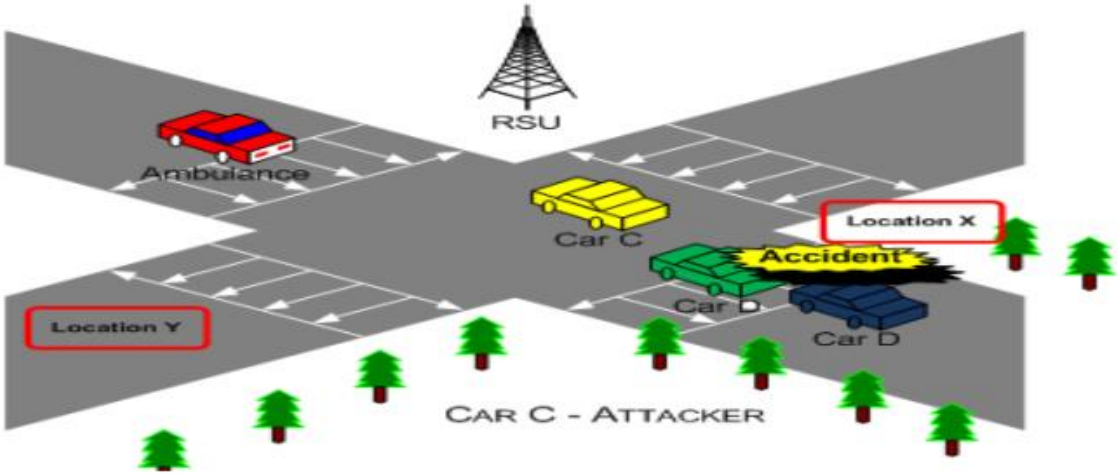

Рисунок 21 — Атака типу Node Impersonation у VANET

У цій атаці зловмисник навмисно надсилає неправдиву інформацію в мережі. Мета цього типу повідомлення полягає в тому, щоб створити плутанину в спілкуванні або сповістити про егоїстичну поведінку вузла, щоб отримати певну послугу. Це також відоме як атака загартовування повідомлень. Такий тип оновлень у життєво важливих повідомленнях буде дуже дорогим у VANET.

3. Спуфінг GPS. Ця атака також відома як атака підробки позиції. У цьому типі атаки зловмисник намагається змінити ідентифікатор поточного географічного розташування та отримати неправдиву інформацію з системи GPS, використовуючи таку техніку, користувач приховує своє поточне місцезнаходження від мережі та показує неправильне положення іншим (рисунок 22). Ця атака може бути здійснена одним транспортним засобом або групою транспортних засобів.

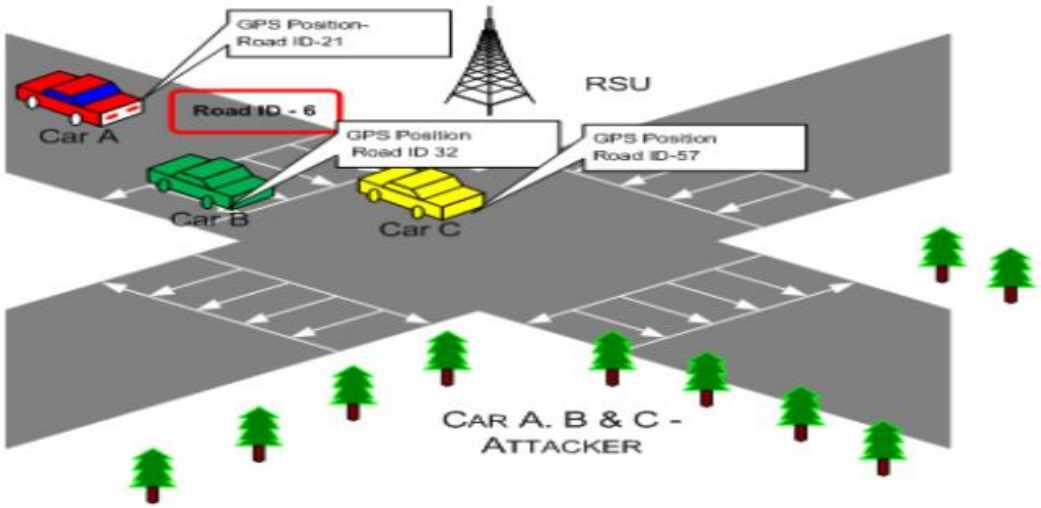

Рисунок 22 — Спуфінг GPS у мережі VANET

Як показано, п’ять транспортних засобів рухаються по дорозі ID-6, але вони приховують своє поточне положення та надсилають неправильну інформацію в мережу. Отримавши такі подробиці, RSU може зробити вигляд, що зараз на дорозі Id-6 немає жодного транспортного засобу.

Загрози конфіденційності

1. Загроза з використанням грубої сили (Брутинг). Транспортний засіб-відправник повинен надіслати свої дані до транспортного засобу призначення за допомогою іншого транспортного засобу, якщо вузол призначення знаходиться далеко від його діапазону. Для збереження секретності транспортний засіб відправника може зашифрувати свої дані та надіслати їх до місця призначення через будь-які транспортні засоби посередника (рисунок 23). Це один із типів криптографічної атаки, у якій транспортний засіб-посередник діятиме як зловмисник і намагатиметься розшифрувати зашифровану інформацію, постійно пробуючи різні альтернативні можливі рішення.

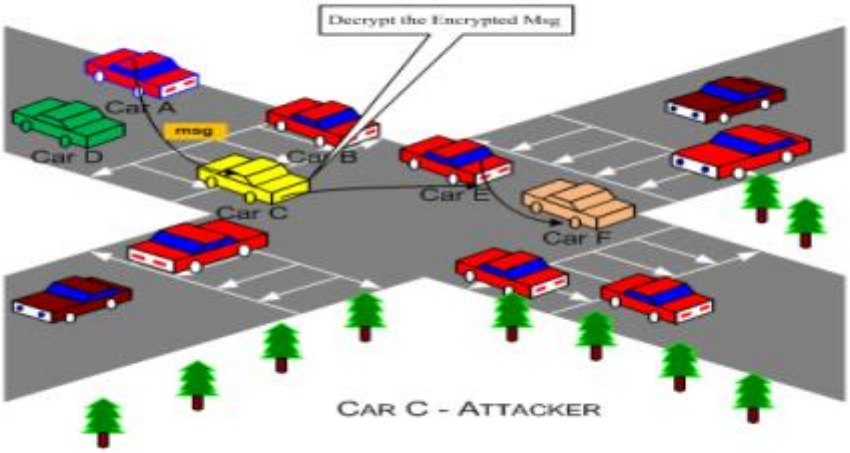

Рисунок 23 — Брутинг у мережі VANET

Автомобіль А хоче надіслати повідомлення до автомобіля F, але він не знаходиться в його діапазоні. Таким чином, Автомобіль А надсилає зашифроване повідомлення Автомобілю F через Автомобіль С, який є зловмисним вузлом, і шляхом застосування грубої сили намагається розшифрувати зашифровані дані, нав’язуючи різні можливі рішення.

2. Підслуховування. Це пасивна атака, при якій здійснюється напад на конфіденційність мережі. Зловмисники збирають конфіденційні дані мережі та мовчки спостерігають за мережевим трафіком або поточним положенням і

діяльністю певного транспортного вузла (рисунок 24). Виявлення таких зловмисників дуже складне, оскільки вони не реагують у поточній мережі.

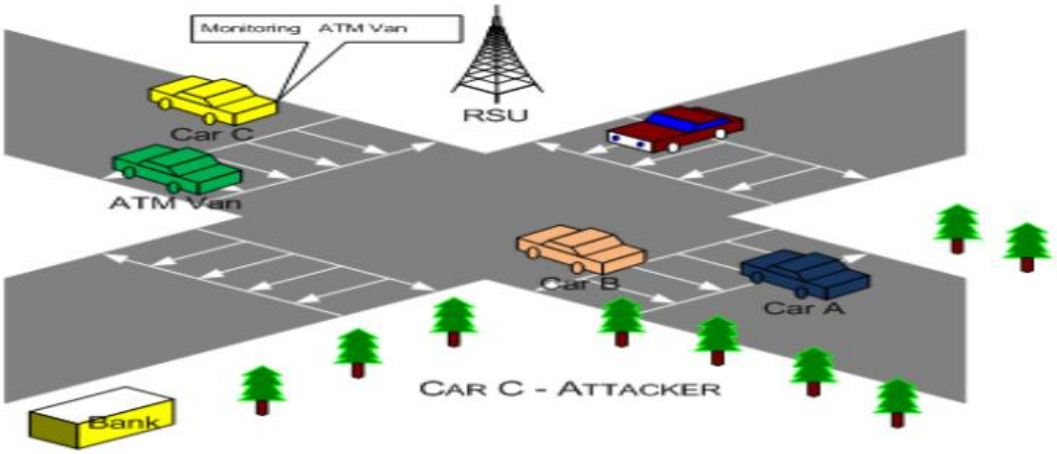

Рисунок 24 — Підслуховування у мережі VANET

Автомобіль С постійно спостерігає за деталями фургона банкомату та передає таку інформацію грабіжнику. Атака з розголошенням ідентифікатора — це одна з підкатегорій прослуховування, у якій зловмисник розкриває ідентифікаційні дані вузла та використовує їх для відстеження цільового вузла.

1.4.3 Рішення для різних атак

Нижче наведено запропоновані рішення для деяких атак, розглянутих вище:

Рішення для атаки DOS базується на використанні OBU (On Board Unit), який встановлюється в автомобілях. У разі атаки DOS блок обробки запропонує OBU перемкнути канал, технологію або використати техніку стрибкоподібної зміни частоти чи багаторазовий трансивер [13].

Рішення для атаки грубою силою запропоновано Langley et al [19]. Це безпечний метод автентифікації, який вимагає використання певної унікальної ідентифікації для транспортних засобів, об’єднаної з деяким великим

випадковим значенням, а потім хешування за допомогою певного алгоритму хешування.

Для боротьби з атаками аналізу трафіку Cencioni et al [7] запропонував VIPER: протокол забезпечення конфіденційності зв'язку між транспортним засобом і інфраструктурою. Він стійкий до атак аналізу трафіку. Знаходячись у транспортному засобі вони надсилатимуть свої повідомлення безпосередньо до RSU, але матимуть додатковий транспортний засіб, який буде діяти як змішані вузли.

Щоб запобігти атакам підробки та Sybil, Yan et al [32] запропонували нове рішення, яке використовує бортовий радар як віртуальне «око» автомобіля. Незважаючи на те, що «зір» обмежений через скромну дальність передачі радара, транспортний засіб може бачити навколишні транспортні засоби та отримувати звіти про їхні GPS-координати. Шляхом порівняння побаченого з тим, що було почуто, транспортний засіб може підтвердити реальну позицію сусідів і ізолювати шкідливі транспортні засоби.

Щоб запобігти повторним атакам у мережах транспортних засобів [18], існує два варіанти: Перший варіант — використання глобально синхронізованого часу для усіх вузлів, другий варіант пропонує використовувати nonce (Timestamp). Одним із запропонованих рішень для пом'якшення цієї атаки є перевірка отриманих даних у співвідношенні з даними, отриманими з інших джерел. Важливим питанням у цьому контексті є коректність отриманих даних, а не їх джерело [26].

1.5 Огляд досліджень над Vanet мережами

Дослідженням питань безпечної передачі даних у мережах з динамічною топологією VANET займаються вчені з усього світу. Підвищення надійності каналів та захищеності трафіку мережі розглядається з погляду впровадження відповідних протоколів передачі даних. Для цього була розглянута низка

наукових досліджень, присвячених розробці підходів до ефективного розміщення даних у бездротових транспортних мережах.

Під час дослідження [24] було розглянуто застосування технології автономних обчислень у мережах VANET за допомогою протоколу Ad hoc Discovery Mechanism (ADM). Поданий підхід дозволяє здійснювати передачу інформації залежно від щільності мережі та рівня пріоритету повідомлень з високою надійністю та автономністю. Протокол ADM використовує попередньо обчислені стратегії передачі за допомогою еволюційного алгоритму. Для кожного вузла мережі динамічно підбираються власні параметри передачі в залежності від щільності мережі та рівня пріоритету повідомлення. Результати проведених експериментів показали, що ADM перевершує два інші методи протоколу широкомовної передачі — Smart-flooding та Simple flooding як передачі за рахунок меншої середньої кількості колізій в залежності від числа вузлів мережі. Автори роботи розглядають адаптацію алгоритму ADM для кількох рівнів пріоритетів повідомлень зі збільшенням кількості джерел передачі.

Авторами роботи [17] запропоновано модель пріоритезації даних від транспортних засобів для прискорення аналізу надвеликих даних. Для визначення пріоритету сполучення транспортних засобів використовується кластеризація. Запропонована модель використовує алгоритм, заснований на виділенні базових точок у мережі транспортних засобів, для прискорення роботи алгоритму кластеризації та підвищення ефективності кількості передачі інформації. транспортних засобів у зоні релевантності з мінімальними накладними витратами. Застосування запропонованого алгоритму в реальних умовах скоротило час кластеризації, що дозволяє програмам для захисту від атак проводити виявлення в режимі реального часу.

Дослідження [22] присвячено вивченню поточного стану технологій VANET, а також розгортання даних бездротових мереж у реальному світі. Було виділено особливу увагу до розробки додатків, специфічних для VANET, які забезпечують якусь конкретну функцію та основу для належної підтримки

ТЗ. Крім того, представлено концепцію їх класифікації: загального призначення, допомоги водієві, забезпечення безпеки, рекламні програми, а також розважальні програми. Більшість додатків полегшують керування і забезпечують безпеку. Виходячи з представлених особливостей доступної інформації трафіку, слід виділяти характеристики даних програм для аналізу стану об'єкта мережі

Під час дослідження [30] було проведений порівняльний аналіз актуальних мережевих загроз та методів забезпечення безпеки протоколів маршрутизації для мереж із динамічною топологією. Також необхідно розповісти про важливості комбінованого застосування підходів до захисту передачі інформації. Розглянуто проблему мережевих атак, пов'язаних із зловмисним погіршенням якості маршрутів, а також проблему «егоїстичності», коли рідко враховуються соціальні характеристики вузлів мережі, що дозволяє внутрішнім порушникам ігнорувати співпрацю з іншими вузлами. Були запропоновані рекомендації щодо визначення оптимального порогового значення рівнів репутації або комплексної метрики маршрутів у різних сценаріях, а також вибір порогового значення для мінімізації кількості помилок першого та другого роду для забезпечення ефективності та безпеки маршрутизації. Проведено дослідження залежності моделі забезпечення безпеки маршрутизації від архітектури, призначення та властивостей аналізованої мережі, що динамічно організується.

Різні методи поширення даних та проблеми, пов'язані з ними, представлені в роботі [20]. Автори відзначають, що тип додатків VANET і властиві їм характеристики, такі як різна щільність мережі, швидке переміщення транспортних засобів роблять поширення даних досить складним. Головними особливостями VANET є висока динамічність топології, можливість частково відключити мережу, моделювання та прогнозування мобільності, комунікаційне середовище, жорсткі обмеження затримки, взаємодія з бортовими датчиками. Найбільш актуальним напрямом досліджень автори виділяють розробку методу розповсюдження даних, який

забезпечує прийнятну продуктивність при різних додатках та зі змінною архітектурою VANET.

В рамках дослідження [8] представлений новий протокол агрегування даних інформаційних систем дорожнього руху, що називається Smart Directional Data Aggregation (SDDA), здатний зменшити навантаження мережі при одночасному отриманні високоточної інформації про умови руху на великих ділянках доріг. Автори ввели три рівні фільтрації: фільтрація всіх повідомлень Floating Car Data (FCD), інтеграція методу придушення на етапі збору інформації, агрегування відфільтрованих даних та їхнє подальше поширення. Проведені експерименти підтвердили високу ефективність SDDA: досягнення низького коефіцієнта навантаження, а також високу точність агрегування.

Таким чином, огляд проведених досліджень показав, що розробка гібридної моделі ефективного розміщення вихідних та проміжних даних у бездротових транспортних мережах з динамічною топологією VANET є актуальною проблемою, яка потребує розробки докладного структурного представлення програмно-конфігурованої мережі.

1.6 Висновки з розділу 1

На основі знайденої та отриманої інформації з актуальних джерел, в ході роботи були отримані необхідні відомості про бездротові мережи, що самоорганізуються. На основі отриманих даних були виділені основні шляхи дослідження та отримання додаткової інформації в галузі VANET мереж. Виходячи з цієї інформації можна зробити наступний висновок.

Бездротові мережі, що самоорганізуються, являють собою телекомунікаційну технологію, що швидко розвивається. Їхня популярність пов'язана з їх простотою розгортання та швидким налаштуванням. Ці особливості роблять їх ідеальними для користувачів, провайдерів Інтернет-послуг та реагування на надзвичайні ситуації, за яких нормальний зв'язок

неможливий. Їх можна успішно використовувати у зонах стихійних лих (землетруси, повені, урагани), на військових полігонах, школах, на конференціях, готелях, аеропортах, будинках тощо. Такий вид мережі — найкраща альтернатива для країн, що розвиваються, і там де комунікаційна інфраструктура відсутня зовсім.

Зв’язок VANET має бути захищеним від різних типів зловмисників. Якщо зловмисник змінює вміст даних, створює непотрібну затримку, змінює особисту ідентифікацію або неправильно поводиться в мережі, це стає критичним для мережі VANET.

Також під час дослідження я зміг детально проаналізувати структуру, сферу застосування та особливості використання VANET мереж, розглянув актуальні вимоги та дослідив принцип використання технології автономних обчислень у мережах VANET за допомогою протоколу ADM.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ЗАСОБІВ КЛАСИФІКАЦІЇ ЗАГРОЗ У МЕРЕЖАХ

2.1 Машинне навчання та класифікація нейронних мереж

Машинне навчання — це напрямок штучного інтелекту, що пов’язаний з побудовою аналітичних моделей для виявлення певних закономірностей. Ключова ланка машинного навчання — певний набор даних, завдяки якому і відбувається побудова певних передбачень стосовно нового вихідного набору даних.

Класифікація нейронних мереж за характером навчання:

- нейронні мережі, які використовують навчання з учителем;
- нейронні мережі, які використовують навчання без учителя.

Нейронні мережі, які використовують навчання з учителем.

Навчання з вчителем припускає, що для кожного вхідного вектора існує цільовий вектор, що є необхідним виходом. Разом вони називаються навчальною парою. Зазвичай мережа навчається на наборі деяких навчальних пар. Пред'являється вихідний вектор, обчислюється вихід мережі та порівнюється з відповідним цільовим вектором. Далі ваги змінюються відповідно до алгоритму, що прагне мінімізувати помилку. Вектори навчальної множини пред'являються послідовно, обчислюються помилки і ваги підлаштовуються для кожного вектора, поки помилка по всьому навчального масиву не досягне прийнятного рівня.

Нейронні мережі, які використовують навчання без учителя.

Навчання без вчителя є набагато правдоподібнішою моделлю навчання з точки зору біологічного коріння штучних нейронних мереж. Розвинена Кохоненом і багатьма іншими, вона не потребує цільового вектора для виходів, отже не вимагає порівняння з наперед ідеальними відповідями. Навчальна множина складається лише з вхідних векторів. Навчальний алгоритм підлаштовує ваги мережі так, щоб виходили узгоджені вихідні

вектори, тобто щоб пред'явлення достатньо близьких вхідних векторів давало однакові виходи. Процес навчання виділяє статистичні властивості навчальної множини і групує подібні вектори до класів.

Класифікація нейронних мереж за типом налаштування вагових коефіцієнтів:

• мережі з фіксованими зв'язками — вагові коефіцієнти нейронної мережі вибираються відразу, з умов завдання;

• мережі з динамічними зв'язками — для них у процесі навчання відбувається налаштування синаптичних ваг.

Класифікація нейронних мереж за типом вхідної інформації:

• аналогові — вхідна інформація представлена у формі дійсних чисел;

• двійкові — вся вхідна інформація в таких мережах подається у вигляді нулів та одиниць.

2.2 Штучні нейронні мережі

Штучна нейронна мережа — це обчислювальна модель, яка навчається, комбінуючи штучні нейрони для розуміння вхідних даних, і прогнозує очікуваний результат. Однією з головних особливостей нейронних мереж є — вміння самостійно навчатися та адаптуватися під прецеденти, тобто під минулий досвід, неухильно та поступово зменшуючи кількість та вагомість помилок.

Концепція нейронних мереж моделює принципи взаємодії нейронів головного мозку, імітує структуру нервової системи людини. Вона являє собою певну кількість нейронів, що є обчислювальним елементом, які відносяться до певного шару мережі. Після цього вхідні дані проходять обробку через всі шари цієї мережі. Під час цієї обробки параметри обчислювальних елементів можуть змінюватися залежно від результатів, які були отримані на минулих вхідних даних.

Кожен нейрон пов'язаний один з одним і зовнішнім середовищем за допомогою зв'язків, кожен з яких має певний коефіцієнт, на який множиться значення вхідних даних, які проходять через нього.

Нейронні мережі являють собою моделі, які засновані на машинному навчанні, тобто вони набувають всі властивості в процесі навчання.

Навчання має під собою знаходження певних коефіцієнтів зв’зяків між нейронами. Нейронна мережа здатна виявляти складні залежності між вхідними даними, які можливо ще невідомі людям, та певними результуючими даними. Це означає, що вона здатна повернути правильний результат на вхідних даних, які раніше не зустрічалися.

Різні архітектури нейронних мереж, такі як багатошаровий перцептрон (англ. Multilayer Perceptron, MLP) і згорткова нейронна мережа (англ. Convolutional Neural Network, CNN), а також відома архітектура багатошарової згорткової нейронної мережі Visual Geometry Group 16, використовуючи добре відому базу даних цифрових зображень рукописних цифр MNIST, допоможуть знайти різницю в точності методів розпізнавання, що лежать в їх основі.

2.3 Багатошаровий перцептрон

Багатошаровий перцептрон — це архітектура штучної нейронної мережі з прямим зв'язком (рисунок 25). Він відображає вхідні дані у відповідний вихідний набір. Кожна функція активації шару зазвичай використовується для створення нелінійного відображення між входом і виходом в діапазоні від 0 до 1. Традиційно багатошаровий перцептрон складається з двох-трьох шарів, розташованих один за одним.

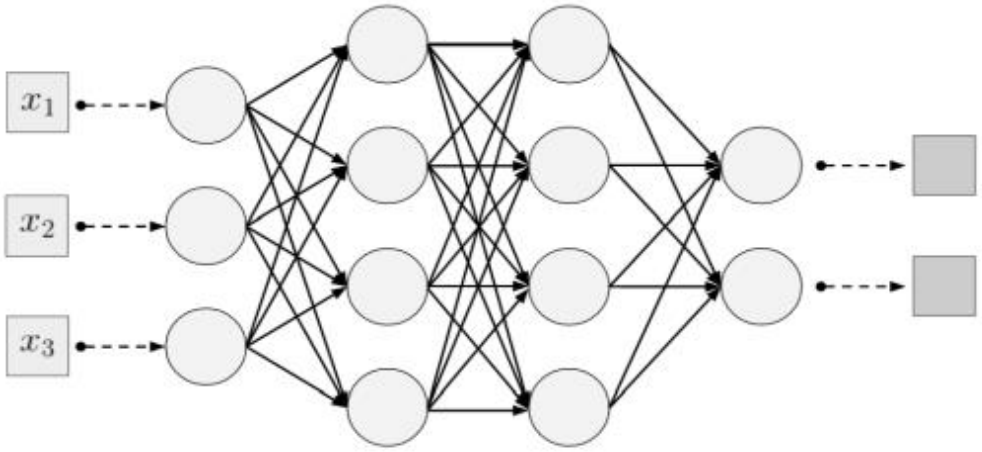

Рисунок 25 — Нейромережі з прямим зв'язком

В основному процес навчання складається з трьох кроків. Перший крок - це прямий прохід, на якому вхідні дані відправляються далі до подальших шарів, за допомогою множення значень входів на ваги нейронів, підсумовування, додавання величини зсуву. На другому етапі після отримання прогнозованого результату він порівнюється з фактичним результатом, а потім розраховується величина помилки (втрати). На останньому етапі виконується так зване зворотне поширення помилки по шарам. Воно передбачає два проходи по всім шарам мережі: прямий і зворотній. При прямому проході вхідні дані подаються на вхідний прошарок нейронної мережі, після чого поширюється по мережі від шару до шару. Потім формуються результати, котрі і є реакцією на певний образ. Зворотній прохід — це поширення сигналів помилки від виходів мережі до її входів, в напрямку, зворотному прямому поширенню сигналів у звичайному режимі роботи. Після закінчення роботи цього методу відбувається оновлення ваг. У багатошаровому перцептроні вихід вузла визначається певною функцією активації. Це функція, аргументом якої є виважена сума входів штучного нейрона, а значенням — вихід нейрона.

Вихід багатошарового перцептрона визначається функціями активації нейронів у кожному шарі. Ці функції можуть бути різними, але найчастіше

використовуються логістична функція, сигмоїдальна функція (рисунок 26) та гіперболічний тангенс.

2.3.1 Сигмоїдальна функція

Формула розрахунку сигмоїдальної функції

$$A = \frac{1}{1 + e^{-x}} \tag{1}$$

Вона прагне привести значення до однієї зі сторін кривої. Така поведінка дозволяє знаходити чіткі межі при прогнозі.

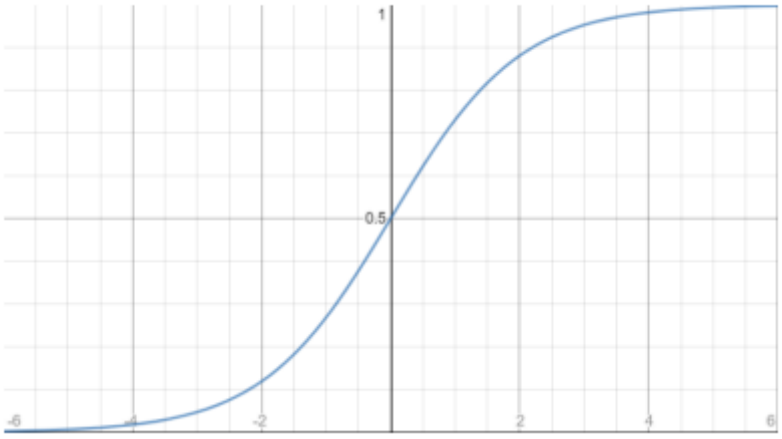

Рисунок 26 — Графік сигмоїдальної функції

2.3.2 Гіперболічний тангенс

Формула розрахунку гіберболічного тангенсу

$$A = \frac{2}{1 + e^{-2x}} - 1 \tag{2}$$

Гіперболічний тангенс (рисунок 27) дуже схожий на сігмоїду. Він нелінійний, добре підходить для комбінації шарів. Діапазон значень функції (-1, 1). Однак варто зазначити, що градієнт тангенсіальної функції більше, ніж у сігмоїдальної (похідна крутіше).

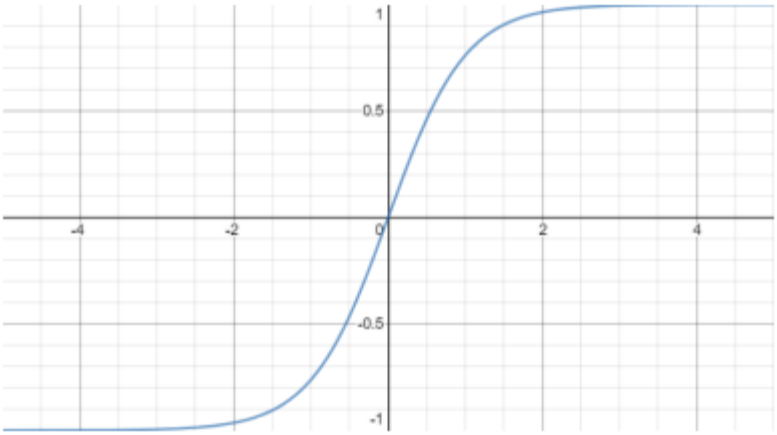

Рисунок 27 — Графік тангенсіальної функції

2.3.3 ReLu (англ. Rectified Linear Unit)

Формула для визначення активаційної функції ReLu

$$A = \max(0, x) \quad (3)$$

ReLu повертає значення x, якщо x позитивно, і 0 в іншому випадку (рисунок 28). ReLu менш вимогливо до обчислювальних ресурсів, ніж гіперболічний тангенс або сигмоїда, так як виробляє більш прості математичні операції. Тому має сенс використовувати ReLu при створенні глибоких нейронних мереж.

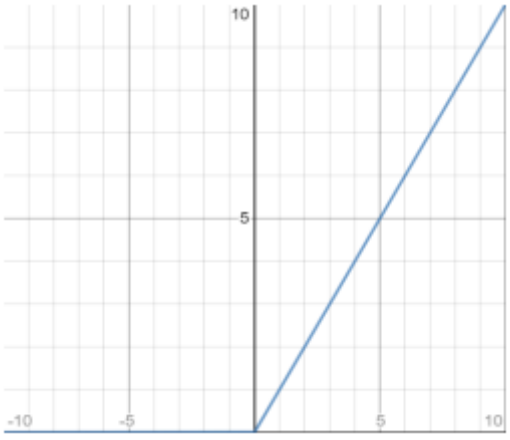

Рисунок 28 — Графік ReLu функції

Багатошаровий перцептрон як і раніше є гарним рішенням для більшості завдань, які не можна достатньо прийнятно вирішити нейронною мережею з

маленькою кількістю шарів. Більш сучасні методи глибокого навчання значно поліпшили стан справ в області розпізнавання мовлення, візуального розпізнавання і виявлення об'єктів, а також у багатьох інших областях. Термін «глибинне навчання» відноситься до нейронних мереж з більш ніж одним прихованим шаром.

Глибина нейронної мережі дозволяє їй будувати ієрархію ознак зі зростаючою абстракцією, при цьому кожен наступний шар діє як фільтр для все більш і більш складних ознак, які об'єднують ознаки попереднього рівня. Ця ієрархія ознак і фільтрів, які моделюють значення даних, створюються автоматично, коли глибинні мережі навчаються відновлювати невідомі залежності. Оскільки вони можуть працювати з невивченими даними, які формують більшість даних в світі, глибинні мережі можуть виявитися більш точними, ніж традиційні алгоритми машинного навчання, які не здатні обробляти такі дані.

2.4 Фреймворки та бібліотеки глибинного навчання

2.4.1 Фреймворк Keras

Keras — це високорівневий та зручний API, який використовується для створення та навчання нейронних мереж. Це фреймворк з відкритим кодом, побудований на мові Python, який працює поверх TensorFlow. Він був розроблений, щоб уможливити швидке експериментування та повторення та знизити необхідний бар’єр для того щоб розпочати роботу з глибоким навчанням.

Sequential API — це один з найпростіших спосіб використання Keras для побудови нейронної мережі. Використовуючи клас Sequential, можна складати різні типи шарів один за одним, для створення нейронної мережі. Також Keras Sequential API має багато різноманітних типів шарів:

- Згортковий шар: шар для обробки зображень — використовується для згорткових нейронних мереж.

- Повторюваний рівень: рівень обробки послідовностей даних — використовується для повторюваних нейронних мереж.
- Шар MaxPooling: шар для зменшення вибірки карт функцій шляхом отримання максимального значення в неперекриваючих прямокутних блоках.
- Згладжувальний шар: шар, який зводить багатовимірні вхідні тензори в єдиний вимір — використовується як перехідний шар між згортковими або рекурентними шарами та повністю пов’язаними шарами в нейронній мережі.

Наведені приклади є лише маленькою частиною усього функціоналу доступного у Keras Sequential API. Кожен рівень призначений для виконання певного типу обчислень на вхідних даних, саме тому їх можна комбінувати для створення потужних архітектур нейронних мереж.

2.4.2 Бібліотека NumPy

NumPy — це бібліотека для мови програмування Python, яка додає підтримку великих багатовимірних масивів і матриць разом із великою колекцією математичних функцій високого рівня для роботи з цими масивами [30]. Попередник NumPy, Numeric, спочатку був створений Джимом Хугуніним за участю кількох інших розробників. У 2005 році Тревіс Оліфант створив NumPy, включивши функції конкуруючого Numarray у Numeric із великими змінами. NumPy є програмним забезпеченням з відкритим вихідним кодом і має багато учасників. NumPy — це проект, який фінансується NumFOCUS.

NumPy націлений на еталонну реалізацію Python CPython, яка є неоптимізуючим інтерпретатором байт-коду. Математичні алгоритми, написані для цієї версії Python, часто працюють набагато повільніше, ніж скомпільовані еквіваленти через відсутність оптимізації компілятора. NumPy частково вирішує проблему повільності, надаючи багатовимірні масиви, функції та оператори, які ефективно працюють з масивами; їх використання

вимагає переписування коду, переважно внутрішніх циклів, за допомогою NumPy.

Використання NumPy у Python надає функціональність, порівнянну з MATLAB, оскільки вони обидва інтерпретуються, і вони обидва дозволяють користувачеві писати швидкі програми, якщо більшість операцій працюють із масивами чи матрицями замість скалярів. Для порівняння, MATLAB може похвалитися великою кількістю додаткових наборів інструментів, зокрема Simulink, тоді як NumPy внутрішньо інтегрований з Python, більш сучасною та повною мовою програмування. Крім того, доступні додаткові пакети Python; SciPy — це бібліотека, яка додає більше функціональних можливостей, схожих на MATLAB, а Matplotlib — це пакет для побудови графічних зображень, який забезпечує функціональні можливості побудови, подібні до MATLAB. Внутрішньо і MATLAB, і NumPy покладаються на BLAS і LAPACK для ефективних обчислень лінійної алгебри.

2.4.3 Бібліотека Pandas

Pandas — це бібліотека програмного забезпечення, написана на мові програмування Python для обробки та аналізу даних. Зокрема, він пропонує структури даних і операції для роботи з числовими таблицями та часовими рядами. Це безкоштовне програмне забезпечення, випущене за ліцензією BSD із трьох пунктів (авторські права, дозволи та відсутність гарантій). Назва походить від терміну «панельні дані», економетричного терміну для наборів даних, які включають спостереження за кілька періодів часу для тих самих осіб. Його назва є грою самої фрази «аналіз даних Python». Уес МакКінні почав будувати те, що стане пандами в AQR Capital, коли він був там дослідником з 2007 по 2010 рік.

Pandas в основному використовується для аналізу даних і відповідної обробки табличних даних у DataFrames. Pandas дозволяє імпортувати дані з різних форматів файлів, таких як значення, розділені комами, JSON, Parquet, таблиці або запити бази даних SQL і Microsoft Excel. Pandas дозволяє

виконувати різноманітні операції маніпулювання даними, такі як об’єднання, зміна форми, вибір, а також функції очищення та суперечки даних. Розробка pandas ввела в Python багато порівнянних функцій роботи з DataFrames.

2.4.4 Бібліотека Matplotlib

Matplotlib — це бібліотека для побудови графіків для мови програмування Python та її розширення чисельної математики NumPy. Вона надає об’єктно-орієнтований API для вбудовування графіків (рисунок 29) та гістограм (рисунок 30) у програми за допомогою наборів інструментів загального призначення GUI, таких як Tkinter, wxPython, Qt або GTK. Існує також процедурний інтерфейс "pylab", заснований на кінцевій машині (як OpenGL), розроблений, щоб дуже нагадувати інтерфейс MATLAB, хоча його використання не рекомендується. SciPy використовує Matplotlib.

Matplotlib спочатку був написаний Джоном Д. Хантером. З тих пір вона має активну спільноту розробників і поширюється за ліцензією у стилі BSD. Майкл Дреттбум був призначений провідним розробником matplotlib незадовго до смерті Джона Хантера в серпні 2012 року.

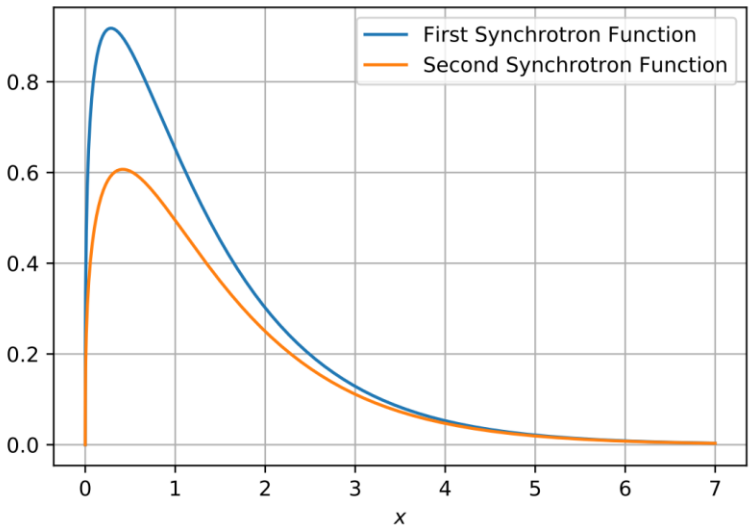

Рисунок 29 — Приклад побудови лінійного графіку за допомогою бібліотеки Matplotlib

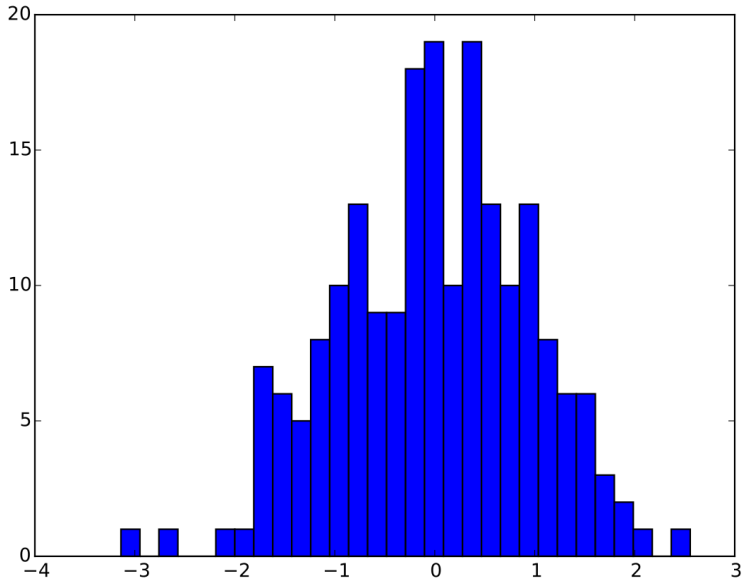

Рисунок 30 — Побудова гістограми за допомогою бібліотеки Matplotlib

2.5 Висновки з розділу 2

Проаналізувавши та дослідивши різноманітні засоби роботи та навчання нейронних мереж я можу зробити наступні висновки:

1. Побудова нейронної мережі буде здійснюватися на основі багатошарового перцептрону за допомогою фреймворка Keras.

2. У якості датасетів були обрані процедурно згенеровані дані за допомогою аналізатора мережі NS-3 інформацію про якого можна знайти за посиланням https://www.nsnam.org.

3. Під час навчання нейронної мережі, доцільно робити замірювання від епохи до епохи для аналізу якості навчання.

4. Ефективність навчання та використання нейронних мереж у класифікації та детектуванні загроз транспортних мережах VANET підтверджено.

РОЗДІЛ 3 РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ДЛЯ ДЕТЕКТУВАННЯ ТА КЛАСИФІКАЦІЇ ЗАГРОЗ У ТРАНСПОРТНІЙ МЕРЕЖІ VANET

3.1 Розробка та навчання моделі класифікації загроз VANET

3.1.1 Налаштування середовища

Встановлення різних компонентів та подальше налаштування буде відбуватися на апаратному середовищі за наступними характеристиками:

1. Двухядерний процесор Intel Core i3-4160 з тактовою частотою 3.6 ГГц.
2. Графічний процесор NVIDIA GeForce GTX 1050 Ti.
3. 8 ГБ оперативної пам'яті.
4. Операційна система Windows 10.

Встановлення робочої мови програмування, середовища розробки, бібліотеки та всі інші необхідні компоненти було зроблено за наступними кроками:

1. Встановлено мову програмування Python 3.10. та бібліотек NumPy 1.23.5. та Keras 2.12.0.
2. Встановлено середовище розробки PyCharm 2023.1.2 Professional по програмі студентської підписки;
3. Попередньо оновлено драйвер графічного процесора за допомогою NVIDIA GeForce Experience.
4. Встановлено бібліотеки pandas 2.0.2 та matplotlib 3.7.1
5. Створення проекту за посиланням `D:\ResearchWork`
6. Створення каталогу для завантаження та зберігання майбутніх графіків `D:\ResearchWork\images`
7. Створення каталогу для зберігання та доступу заздалегіть згенерованих датасетів для навчання та тестування `D:\ResearchWork\datasets\testing`

```
D:\ResearchWork\datasets\training
```

3.1.2 Генерація трафіку

Оскільки на даний момент VANET-мережі реалізуються лише з метою тестування та стек протоколів, що використовується для VANET-мереж, остаточно не сформований та не стандартизований, для генерації трафіку використовувався симулятор NS-3. Усього вдалося згенерувати 4763 зразків.

В результаті виконання сценарію аналізатором пакетів NS-3 генеруються файли трьох типів, для аналізу атак було обрано два з них:

1. Файли **routes**. Генеруються для кожної секунди виконання сценарію та містять таблиці маршрутизації кожного вузла.

```
Node: 0 Time: 25s
AODV Routing table
Destination Gateway     Interface   Flag    Expire      Hops
10.1.2.2    10.1.2.2    10.1.2.1    UP  2.44            1
10.1.2.4    10.1.2.4    10.1.2.1    UP  1.44            1
10.1.2.8    10.1.2.48   10.1.2.1    DOWN    7.13            9
10.1.2.11   10.1.2.26   10.1.2.1    DOWN    9.25            3
10.1.2.12   10.1.2.26   10.1.2.1    UP  4.71            4
10.1.2.18   10.1.2.48   10.1.2.1    DOWN    7.23            4
10.1.2.19   10.1.2.19   10.1.2.1    UP  2.43            1
10.1.2.20   10.1.2.20   10.1.2.1    UP  2.44            1
10.1.2.22   10.1.2.22   10.1.2.1    DOWN    0.04            1
10.1.2.26   10.1.2.26   10.1.2.1    UP  2.43            1
10.1.2.29   10.1.2.61   10.1.2.1    UP  2.14            3
```

Рисунок 31 — Вміст файлу .routes

2. Файли **flowmon.** Має формат XML. Містить інформацію про потік трафіку між кожними двома вузлами. Інформація включає час відправлення першого та останнього пакета, час прийому першого та останнього пакета, загальна затримка та джиттер при відправленні пакетів, обсяг переданих та прийнятих даних, кількість втрачених пакетів та кількість пересилок пакета.

```
<?xml version="1.0" ?>
<FlowMonitor>
  <FlowStats>
    <Flow flowId="1" timeFirstTxPacket="+6000000000.0ns" timeFirstRxPacket="+0.0ns"
      <packetsDropped reasonCode="0" number="0" />
      <packetsDropped reasonCode="1" number="0" />
      <packetsDropped reasonCode="2" number="0" />
      <packetsDropped reasonCode="3" number="0" />
      <packetsDropped reasonCode="4" number="0" />
      <packetsDropped reasonCode="5" number="1" />
      <bytesDropped reasonCode="0" bytes="0" />
      <bytesDropped reasonCode="1" bytes="0" />
      <bytesDropped reasonCode="2" bytes="0" />
      <bytesDropped reasonCode="3" bytes="0" />
      <bytesDropped reasonCode="4" bytes="0" />
      <bytesDropped reasonCode="5" bytes="679" />
      <delayHistogram nBins="0" >
      </delayHistogram>
```

Рисунок 32 — Вміст файлу .flowmon

Для зручності обробки файлів flowmon буде розроблений та використаний відповідний парсер, який виділяє основні параметри у файл з розширенням .flwtxt зміст якого зображено на рисунку 33.

```
FlowID: 1 (UDP 10.1.2.29/49153 --> 10.1.2.24/3)
    Transmitted: 18340
    Recieved: 18340
    Delay: 387515773
    Packet Loss: 0

FlowID: 2 (UDP 10.1.2.13/654 --> 10.1.2.29/654)
    Transmitted: 88
    Recieved: 88
    Delay: 27154271
    Packet Loss: 0
```

Рисунок 33 — Вміст файлу .flwtxt

3.1.3 Обрання параметрів для визначення типів атак

Усього для аналізу було обрано чотири атаки (blackhole, grayhole, ddos і wormhole). Для чотирьох атак було виділено одинадцять параметрів. Таким чином, було виділено п'ять класів (включаючи відсутність атаки).

Список параметрів, що виділяються, виглядає наступним чином:

1) число підмереж;
2) число вузлів, що знаходяться одночасно у двох підмережах;
3) мінімальне сумарне число записів щодо однієї підмережі;
4) середня кількість хопів (переходів);
5) число down інтерфейсів до загальної кількості інтерфейсів (down + up);
6) загальна кількість хостів;

7) середнє число пересланих пакетів одним хостом;

8) середня кількість прийнятих пакетів одним хостом;

9) максимальне число пересланих пакетів одним хостом;

10) середнє число втрачених пакетів одним хостом;

11) середня затримка одного хоста.

3.1.4 Розробка парсера даних

Для виділення параметрів із файлів Routes та Flowmon був розроблений парсер мовою програмування Python. Цей парсер за підсумками своєї роботи складає csv таблицю з параметрами. Для аналізу отриманих параметрів були побудовані різні графіки, які демонструють відмінності між середніми значеннями атак і їх відсутності. Для побудови графіків було використано бібліотеку pyplot. На рисунках 34 — 37 показані графіки порівняння кожного відібраного параметра для певної атаки та її відсутності.

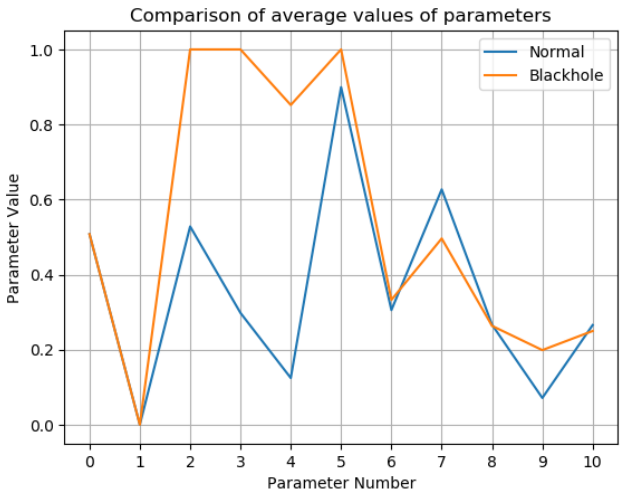

Рисунок 34 — Порівняння нормалізованих «середніх» значень для випадку відсутності атаки та blackhole

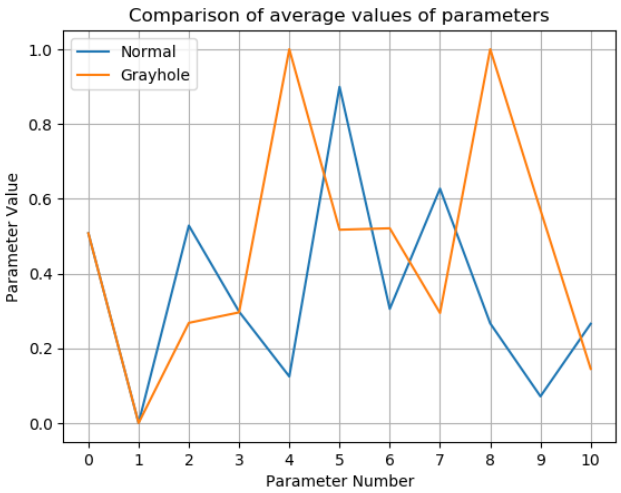

Рисунок 35 — Порівняння нормалізованих «середніх» значень для випадку відсутності атаки та grayhole

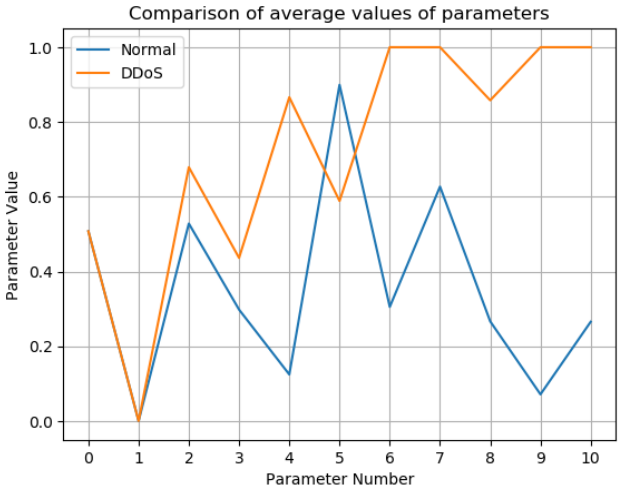

Рисунок 36 — Порівняння нормалізованих «середніх» значень для випадку відсутності атаки та DDoS

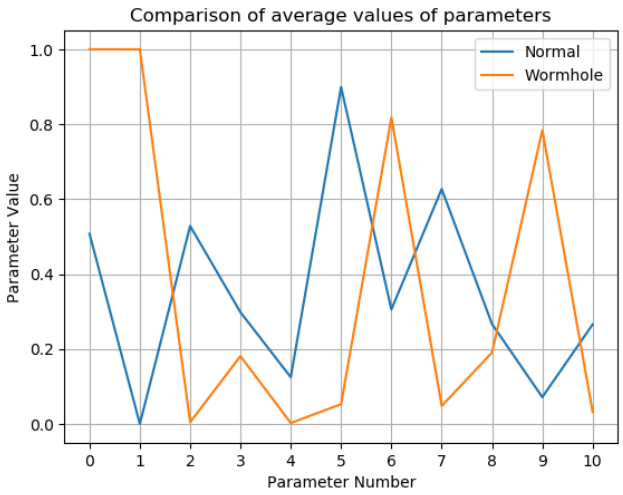

Рисунок 37 — Порівняння нормалізованих «середніх» значень для випадку відсутності атаки та Wormhole

На основі вище зазначених даних було складено таблицю (табл. 3), у якій описується вплив кожної ознаки (параметра) на певну атаку.

За цією таблицею видно, кожен відібраний параметр є важливим для виявлення атак. У таблицях маршрутизацій (Routes) міститься набір параметрів, що покриває всі атаки. Однак, у такому випадку, через малу кількість параметрів для **DDoS** і **grayhole** класифікатору, швидше за все, не вистачало б параметрів для якісної класифікації. Тому необхідно використовувати і файли **flowmon**.

Таблиця 3 — Параметри виявлення атак

Номер ознаки	*Файл, звідки парсується ознака*	*Вплив ознаки на виявлення атаки*			
		blackhole	*DDoS*	*grayhole*	*wormhole*
1	Routes (таблиці маршрутизацій)				+
2					+
3		+			+
4		+			
5		+	+	+	
6	Flowmon (потоки між двома вуз-лами)				+
7			+		
8			+		
9			+	+	
10			+	+	+
11			+		

Лістинг 1 демонструє парсінг параметрів файлу **flowmon**. Вхідні параметри: кількість хостів; прийняті/переслані пакети; макс. кількість пакетів; затримка та втрати пакетів. На виході отримуємо список параметрів, які були отримані з файлу.

Лістинг 1. *Реалізація функціоналу парсинга параметрів потоку між двома вузлами*

```
def parseFlowmonFile(filename):
    parameters = [0, 0, 0, 0, 0, 0]
    with open(filename) as file:
        fileLines = [row.strip() for row in file]
    # Отримання списку кількості отриманих, переданих та
    # втрачених пакетів для кожного хоста
    transmittedList, recievedList, delayList, lossList =
getPacketCount(fileLines)
    parameters[0] = len(transmittedList)
    parameters[1] = int(getAverageValue(transmittedList))
    parameters[2] = int(getAverageValue(recievedList))
```

```
    parameters[3] = int(getMaxValue(transmittedList) / 100)
    parameters[4] = int(getAverageValue(lossList))
    parameters[5]    =    int(getAverageValue(delayList)    /
1000000)

        return parameters
```

Лістинг 2 реалізовує Отримання кількісного списку пакетів окремо взятого хоста. На вхід передаються розпаковані рядки з кількісними показниками прийнятих, переданих та втрачених пакетів хоста під час передачі мережею. На виході отримуємо відсортовані списки з кількістю кожного окремого показника

Лістинг 2 — *функція отримання даних конкретного хоста*

```
def getPacketCount(lines):
    hostsList = []
    currentHost = 0
    transmittedList = []
    recievedList = []
    delayList = []
    lossList = []

    for i in range(0, len(lines) - 2):
        if (lines[i].find("UDP") != -1):
            startIndex = lines[i].find("-->") + 4
            endIndex   =   lines[i].find("/",   startIndex,
len(lines[i]))
            ip = lines[i][startIndex:endIndex]
            if not ip in hostsList:
                hostsList.append(ip)
                transmittedList.insert(hostsList.index(ip),
0)
                recievedList.insert(hostsList.index(ip), 0)
                lossList.insert(hostsList.index(ip), 0)
                delayList.insert(hostsList.index(ip), 0)
            currentHost = hostsList.index(ip)
        if (lines[i].find("Transmitted") != -1):
            startIndex = lines[i].find(":") + 2
            value = int(lines[i][startIndex:len(lines[i])])
            transmittedList[currentHost] += value
        if (lines[i].find("Recieved") != -1):
            startIndex = lines[i].find(":") + 2
            value = int(lines[i][startIndex:len(lines[i])])
            recievedList[currentHost] += value
        if (lines[i].find("Delay") != -1):
```

```
                startIndex = lines[i].find(":") + 2
                value = int(lines[i][startIndex:len(lines[i])])
                delayList[currentHost] += value
            if (lines[i].find("Packet Loss") != -1):
                startIndex = lines[i].find(":") + 2
                value = int(lines[i][startIndex:len(lines[i])])
                lossList[currentHost] += value

    return transmittedList, recievedList, delayList,
lossList
```

3.1.5 Підготовка навчальних та тестових датасетів

Після успішного парсингу та об'єднання файлів Routes та Flowmon, маємо 4 файли формату .csv (рисунок 38).

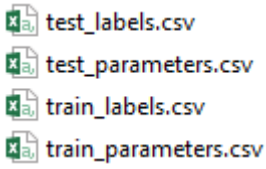

Рисунок 38 — Датасети для навчання та тестування

Таблиці з назвою **lables** — зберігають ідентифікатор типу атаки. В свою чергу таблиці **parameters** — зберігають параметри для визначення загрози. Вміст цього файлу з параметрами для перших 10 зразків наведено нижче (рисунок 39).

1	1,0,1222,3503,48,51,4153,1938,257,3,631
2	1,0,834,2113,24,54,3809,1682,305,2,892
3	1,0,916,2552,40,57,3632,775,312,4,996
4	1,0,815,1747,29,52,5995,3801,467,2,755
5	1,0,908,2165,35,54,4068,1439,450,3,776
6	1,0,846,2600,39,58,3864,1698,315,3,1645
7	1,0,1329,4567,55,59,3888,2921,303,1,857
8	1,0,1046,3208,50,55,4151,1400,263,4,1311
9	1,0,874,2118,46,57,5080,2255,267,3,1273
10	1,0,743,1595,36,57,4043,2168,267,3,1458

Рисунок 39 — Вміст файлу test_parameters

З усього обсягу згенерованого трафіку в 4763 зразки, відбувся поділ даних з яких 4290 зразків використовувалися для навчання мережі, а 473 — для тестування.

3.2 Основний функціонал класифікатора загроз

3.2.1 Допоміжні інструменти для розробки системи

Для розробки класифікатора були використані наступні технології:

1. Мова програмування Python 3.10 — для реалізації функціоналу.
2. Фреймворк Keras — як основний інструмент для взаємодії зі штучними нейронними мережами.
3. Бібліотека NumPy — для надання підтримки у роботі з великими багатовимірними масивами та матрицями.
4. Бібліотека Matplotlib — для реалізації функціоналу побудови графіків апробації результатів навчання та тестування системи.
5. Бібліотека Pandas — для аналізу даних і відповідної обробки табличних даних у DataFrames.

Для встановлення фреймворка та бібліотек була використана система керування пакетами, написана мовою Python під назвою Pip за допомогою наступної команди: `pip install`.

3.2.2 Програмна архітектура

Нейронна мережа побудована на основі багатошарового перцептрона. На рисунку 40 показано візуалізацію моделі розробленої нейронної мережі.

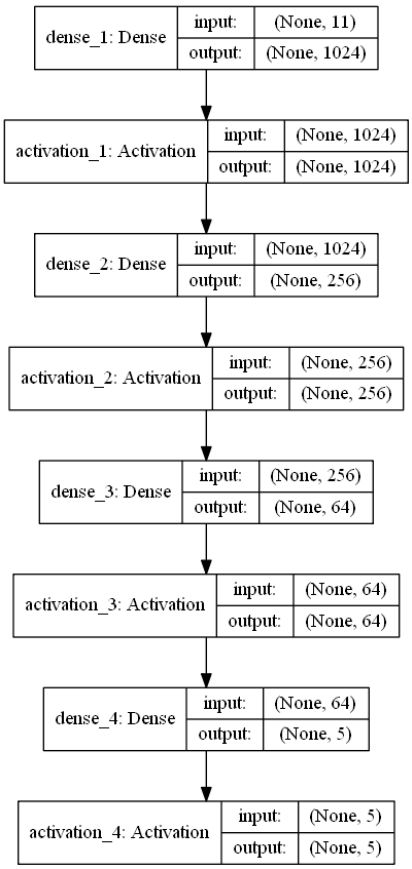

Рисунок 40 — Візуальна модель розробленої ШНМ

Додатково відображені розміри входів та виходів для шарів. `None`, слідуючий першим в кортежі розмірів — це розмірність батча (кількість одиниць даних, які обробляються одночасно). Через те, що розмірність є `None`, то батч може бути довільним.

Більш детальний опис моделі розробленої ШНМ представлено у таблиці (табл. 4).

Таблиця 4 — модель розробленої ШНМ

Шар	Тип шару	Функція активації	Розмір входу	Розмір виходу
Вхідний	Dense	tanh	11	1024
Прихований	Dense	tanh	1024	256
Прихований	Dense	tanh	256	64
Вихідний	Dense	softmax	64	5

Як функція втрат була використана категоріальна крос-ентропія (оскільки працюємо з категоріальною класифікацією), як метрика оцінки — точність.

3.2.3 Методи покращення точності класифікатора

Для покращення процесу навчання ШНМ необхідно дізнатись про необхідну кількість епох навчання при який точність класифікації буде досягати максимального значення та виключити епохи під час яких мережа почне перенавчатись що погіршить кінцевий результат.

Лістинг 3 демонструє реалізацію створення графіка зміни точності класифікатора від однієї епохи до іншої за допомогою засобів бібліотеки matplotlib. Основний параметр `x_values` приймає значення діапазону загальної кількості епох, в свою чергу параметр `y_values` приймає значення точності мережі під час кожної епохи.

Лістинг 3 — *Реалізація фунціоналу створення графіка зміни точності класифікатора під час різних епох навчання*

```
import matplotlib.pyplot as plt
# Accuracy - Epochs
x_values = range(1, 11)
y_values = [0.8502, 0.9362, 0.9462, 0.9564, 0.9637,
            0.9662, 0.9734, 0.9678, 0.9695, 0.9721]

def main():
    plt.plot(x_values, y_values)
    plt.title('Accuracy growth from epoch to epoch')
```

```
    plt.xlabel('Epoch Number')
    plt.ylabel('Accuracy')
    plt.xticks(range(1, 11))
    plt.yticks([x / 100.0 for x in range(75, 89, 1)])
    plt.grid(True)
    plt.savefig('images/simple_graph.png',
bbox_inches='tight')
    plt.show()
```

Зміна точності при навчанні ШНМ від епохи до епохи показаний на рисунку 41. Максимальна точність досягається після сьомої епохи і дорівнює 97,34%, далі модель починає перенавчатися і результат коливається в межах від 94% до 97%.

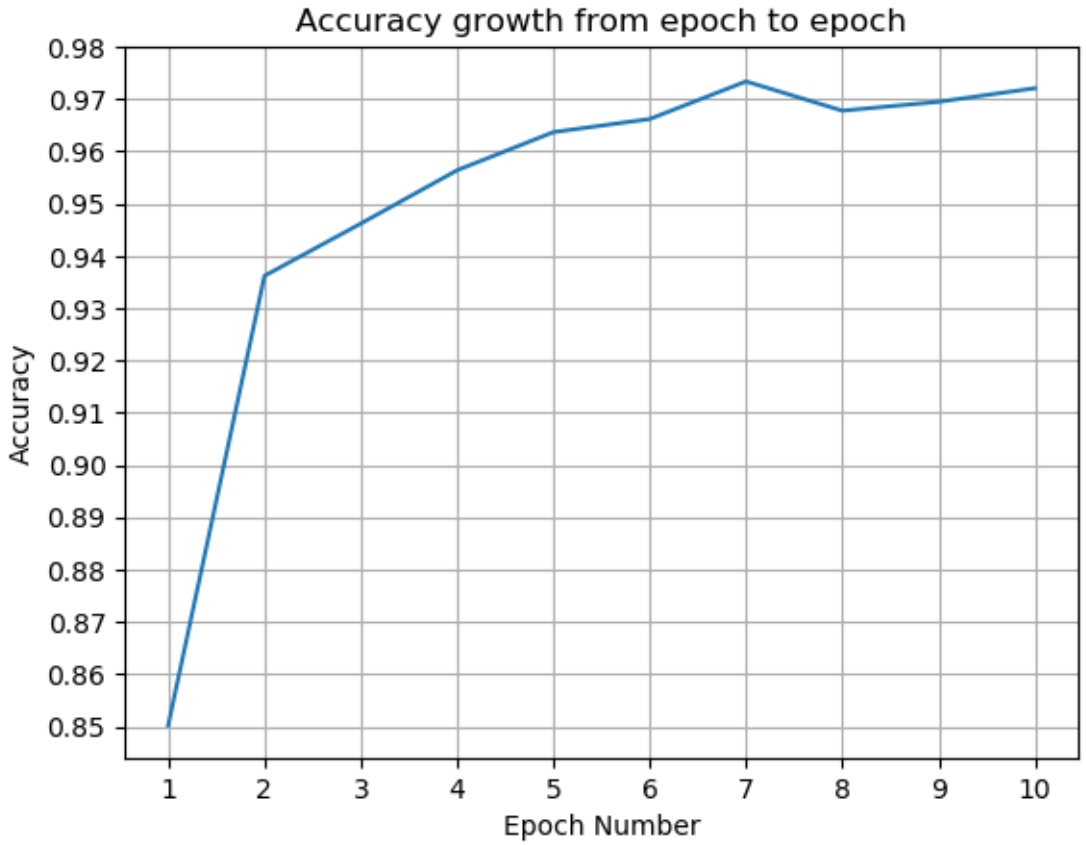

Рисунок 41 — Зміна точності ШНМ від епохи до епохи

Також в ході навчання нейронної мережі були використані різноманітні оптимізатори навчання. Оптимізатор — це метод досягнення найкращих результатів, допомога у прискоренні навчання. Тобто, алгоритм, який

використовується для незначної зміни параметрів, таких як вага та швидкість навчання, щоб модель працювала правильно та швидко.

Лістинг 4 демонструє реалізацію графічного зображення точності кожного з обраних оптимізаторів. Параметр `data_names` приймає назву кожного з оптимізаторів, в свою чергу `data_values` приймає значення точності кожного оптимізатора під час навчання.

Лістинг 4 — *Реалізація фунціоналу створення графіка для відображення точності всіх обраних оптимізаторів навчання.*

```
import numpy as np
import matplotlib.pyplot as plt
column_width = 0.6
index = np.arange(7)
data_names = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta',
              'Adam', 'Adamax', 'Nadam']
data_values = [0.935, 0.966, 0.956, 0.971,
               0.964, 0.975, 0.954]

def main():
    plt.figure()
    plt.title('Comparison of optimizers')
    plt.bar(data_names, data_values, width=column_width)
    plt.yticks([x / 10.0 for x in range(0, 12, 2)])
    plt.xticks(index, data_names)
    plt.grid(True)
    plt.ylabel('Accuracy')
    plt.xlabel('Optimizer')
    plt.savefig('images/plot_diagram.png',
bbox_inches='tight')
    plt.show()
```

На рисунку 42 представлено порівняння точності різних оптимізаторів (SGD, RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam). Всі оптимізатори показали високий результат (вище 93,00%), проте найкращий результат показав оптимізатор Adamax — 97,47%.

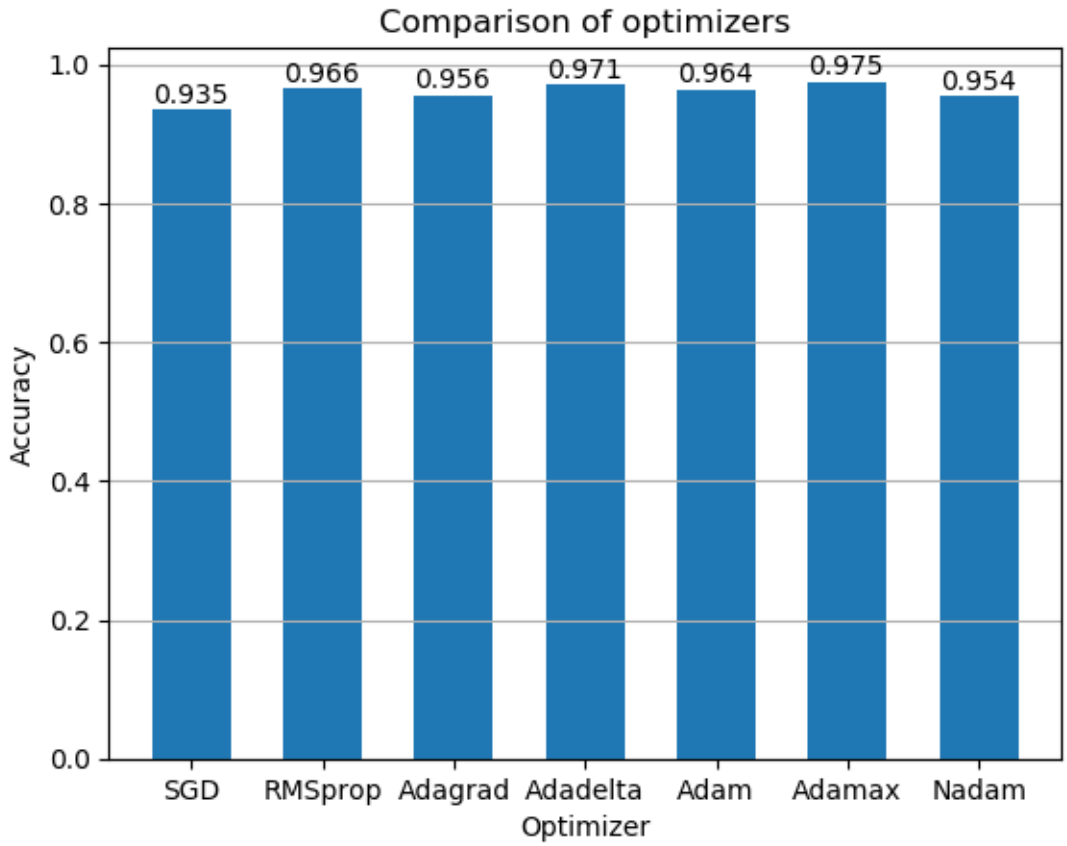

Рисунок 42 — Порівняння точності використаних оптимізаторів

3.2.4 Реалізація основного функціоналу

Лістинг 5 демонструє формування моделі навчання для того щоб підготувати її до роботи. У кожному шарі keras має аргумент ініціалізатора, який можна використовувати для передачі свого методу. Аргумент шару conv2D згортки в методі `Sequential` не використовується, замість нього використовується аргумент шару Dense, який обчислює скалярний добуток вхідних даних та даних ядра keras. Метод `compile` має наступні параметри: loss — це функція помилки, у нашому випадку — це перехресна ентропія, саме для неї готувались мітки у вигляді матриць; `optimizer` — оптимізатор, що буде використовуватися під час навчання, за замовчуванням використовується звичайний стохастичний градієнтний спуск; `metrics` — метрики, які показують якість моделі, в якості метрики використовується точність (accuracy), тобто частка правильно вгаданих відповідей.

Лістинг 5 — *Реалізація фунціоналу створення моделі для навчання*

```
def my_model():
    num_classes = 5
    # Створення моделі, в якій слої йдуть один за одним
    model = Sequential()
    model.add(Dense(1024,kernel_initializer='he_normal',
activation='tanh', input_dim=11))
    model.add(Activation('tanh'))
    model.add(Dense(256,kernel_initializer='he_normal',
activation='tanh'))
    model.add(Activation('tanh'))
    model.add(Dense(64,kernel_initializer='he_normal',
activation='tanh'))
    model.add(Activation('tanh'))
    model.add(Dense(num_classes,
kernel_initializer='he_normal'))
    model.add(Activation('softmax'))
    model.compile(loss='categorical_crossentropy',
                  # adamax, adam, rmsprop, adagrad, adadelta,
nadam, SGD
                  optimizer='adamax',
                  metrics=['accuracy'])
    return model
```

Лістинг 6 демонструє реалізацію функціоналу створення, навчання та тестування за допомогою виклика скомпільованої моделі з функції `my_model`, методів `fit` та `evaluate` які викликаються в функції `main`. Метод `fit` приймає на вхід навчальну вибірку разом з мітками — `x_train` і `y_train`, розмір батча `batch_size`, який обмежує кількість прикладів, що подаються за раз та кількість епох для навчання `epochs` (одна епоха — це один раз повністю пройдена моделлю навчальна вибірка), `shuffle=True` — дані перетасовуються перед кожною епохою. Метод `evaluate` отримує на вхід тестову вибірку разом із мітками для неї та розмір батча.

Лістинг 6 — *Реалізація методів виклику створеної моделі, тренування та тестування нейронної мережі.*

```
# Model creating
    model = my_model()
    print('[+] Model has been compiled')
# Network training
    model.fit(x_train,  y_train,  batch_size=64,  epochs=7,
shuffle=True, verbose=0)
    print('[+] Model has been trained')
```

```
# Network testing
    score = model.evaluate(x_test, y_test, batch_size=32)
    print('[+] Model has been tested. Accuracy: %.2f%%' %
(score[1] * 100))
```

Лістинг 7 демонструє реалізацію функції відтворення та виведення матриці помилок. На вхід йде параметр нормалізації (`true / false`), назва, дані (лейбли стовпців / рядків, дані матриці). На виході отримуємо матрицю оцінки точності та помилок з нормалізацією або без, залежно від переданого на вхід параметра.

Лістинг 7 — *Реалізація побудови матриці помилок*

```
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title=None,
                          cmap=plt.cm.Blues):
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'
    if normalize:
        cm   =   cm.astype('float')   /   cm.sum(axis=1)[:,
np.newaxis]
        fig_filename = 'images/plot_confusion_matrix.png'
        print("Normalized confusion matrix")
    else:
        fig_filename                                          =
'images/plot_confusion_matrix_norm.png'
        print('Confusion matrix, without normalization')
    print(cm)
    fig, ax = plt.subplots()
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    ax.figure.colorbar(im, ax=ax)
    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
           xticklabels=classes, yticklabels=classes,
           title=title,
           ylabel='True label',
           xlabel='Predicted label')
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
             rotation_mode="anchor")
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
```

```
        for j in range(cm.shape[1]):
            ax.text(j, i, format(cm[i, j], fmt),
                    ha="center", va="center",
                    color="white" if cm[i, j] > thresh else
"black")
    fig.tight_layout()
    plt.savefig(fig_filename, bbox_inches='tight')
    #plt.show()
    return ax
```

Лістинг 8 демонструє розрахунок середніх значень кожного параметра для кожного класу атаки з датасета. На вхід передається датасет та список `labels` атаки. На виході виходить розраховане середнє значення параметра, яке можна використовувати для побудови графіка з подальшим дослідженням.

Лістинг 8 — *Реалізація розрахунку середніх значень параметрів атаки*

```
def calculate_average_values(dataset, labels):
    parametersCount = len(dataset[0])
    averageValueLists = [[0] * parametersCount] *
len(class_labels)
    classLists = [[]] * len(class_labels)
    for i in range(len(class_labels)):
        classSize = labels.count(class_labels[i])
        classBeginIndex = labels.index(class_labels[i])
        data = dataset[classBeginIndex:classBeginIndex +
classSize]
        classLists[i] = data

    # Calculating of the average values for each
parameter for each class
    for i in range(len(class_labels)):
        averageParametersList = []
        for j in range(parametersCount):
            # Taking each i'th element from the all
sublists
            argsList_j = [entry[j] for entry in
classLists[i]]
            # Average value calculating
            averageValue = np.mean(argsList_j)
            averageParametersList.append(averageValue)
        averageValueLists[i] = averageParametersList

  #Changing dimension

  for i in range(parametersCount):
```

```
        valueList_i = [valueList[i] for valueList in
averageValueLists]
        newValueList_i                                  =
change_values_dimension(valueList_i)
        for j in range(len(class_labels)):
            averageValueLists[j][i] = newValueList_i[j]
    return averageValueLists
```

3.3 Висновки з розділу 3

1. Було налаштоване середовище для навчання моделі нейронної мережі за допомогою фреймворка Keras.

2. У налаштованому середовищі були підготовлені датасети для навчання та тестування. Окрім цього, були написані скрипти, які генерують графіки для відстеження навчання нейронної мережі від епохи до епохи, графік для відстеження ефективності різноманітних автоматизаторів навчання.

3. Для виділення параметрів файлів Routes та Flowmon та подальшої генерації csv таблиць, розроблений парсер даних.

4. Виділені всі ключові характеристики досліджуваних атак.

5. Була розроблена Комп’ютерна система виявлення загроз в мережі VANET з використанням машинного навчання. Були описані її програмна архітектура, реалізація основного функціоналу.

РОЗДІЛ 4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ КОМП'ЮТЕРНОЇ СИСТЕМИ ВИЯВЛЕННЯ ЗАГРОЗ В МЕРЕЖІ VANET З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

4.1 Результати навчання моделі нейронної мережі для класифікації загроз VANET

4.1.1 Оцінка продуктивності операційної системи під час навчання

У процесі розробки комп'ютерної системи була навчена модель нейронної мережі з використанням власного датасету попередньо поділеного з використанням даних, отриманих від мережевого симулятора NS-3. Для оцінки якості роботи системи було попередньо замірено навантаження на систему (навантаження на процессор та об'єм використаної операційної пам'яті).

На рисунках 43 — 45 показано, як системні процеси використовують пам'ять у разі відсутності навантаження. Увесь моніторинг здійснюється за допомогою вбудованого в операційну систему Window *диспетчера завдань*

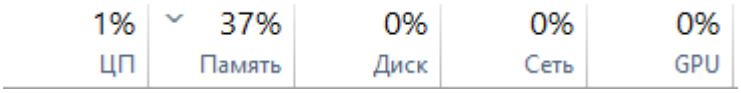

Рисунок 43 — Обсяг використаних системних ресурсів

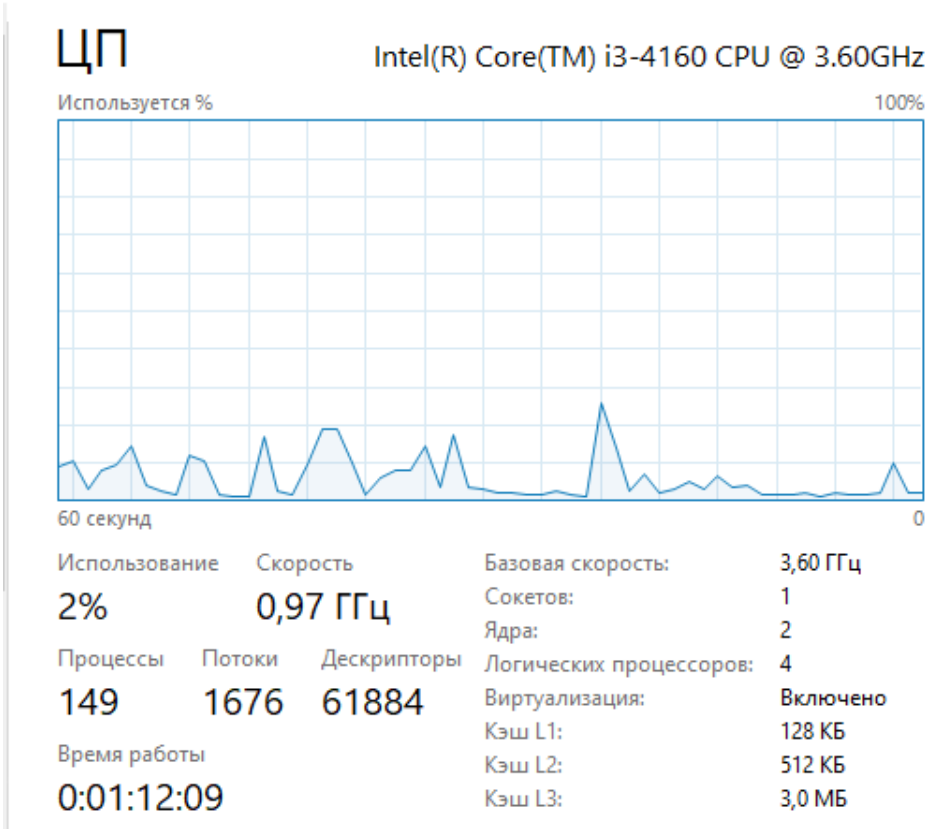

Рисунок 44 — Навантаження на процесор під час роботи системних процесів

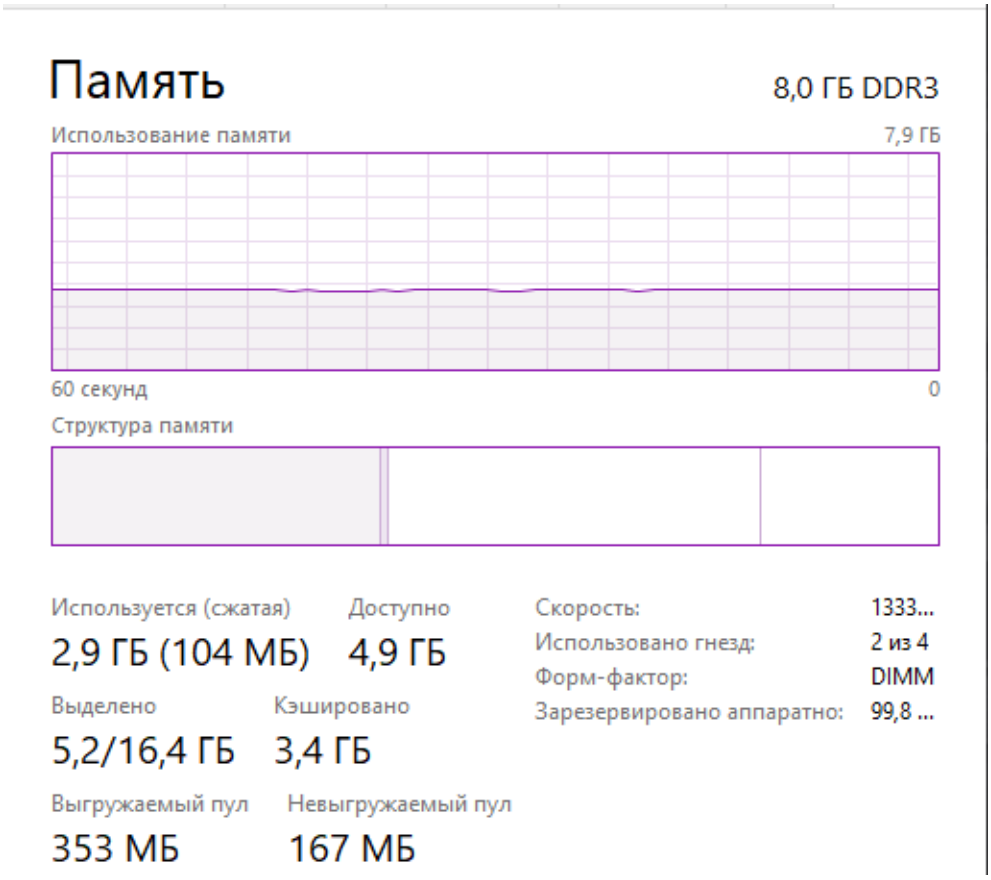

Рисунок 45 — Використання операційної пам’яті системними процесами

На рисунку 46 відбувається аналогічний моніторинг з використанням вбудованого в ОС *монітору ресурсів*.

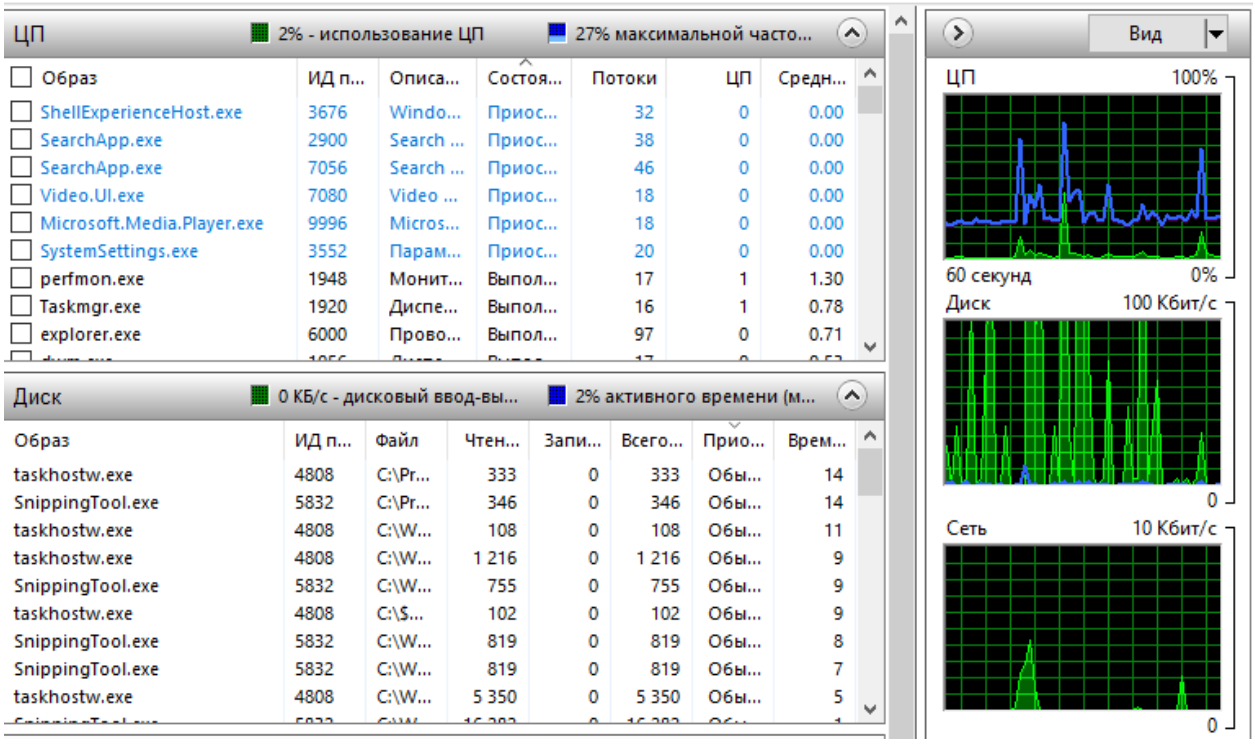

Рисунок 46 — Моніторинг використаних ресурсів з відсутності навантажень

На рисунках 47 — 49 можна побачити моніторинг навантаження з увімкненою середовищем розробки PyCharm.

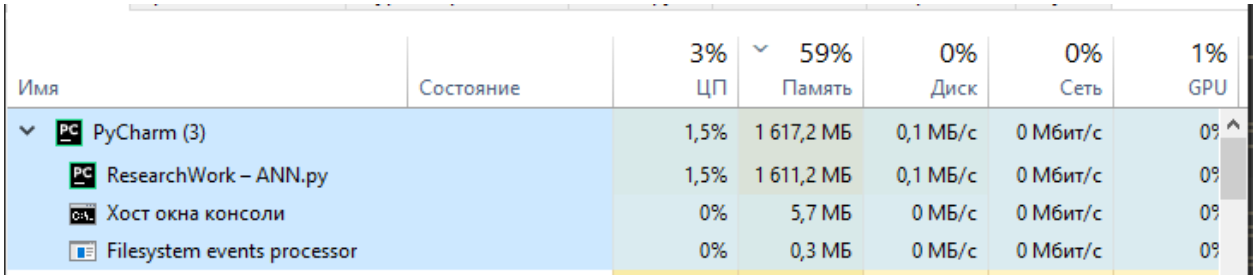

Рисунок 47 — Використання системних ресурсів під час увімкнутого PyCharm

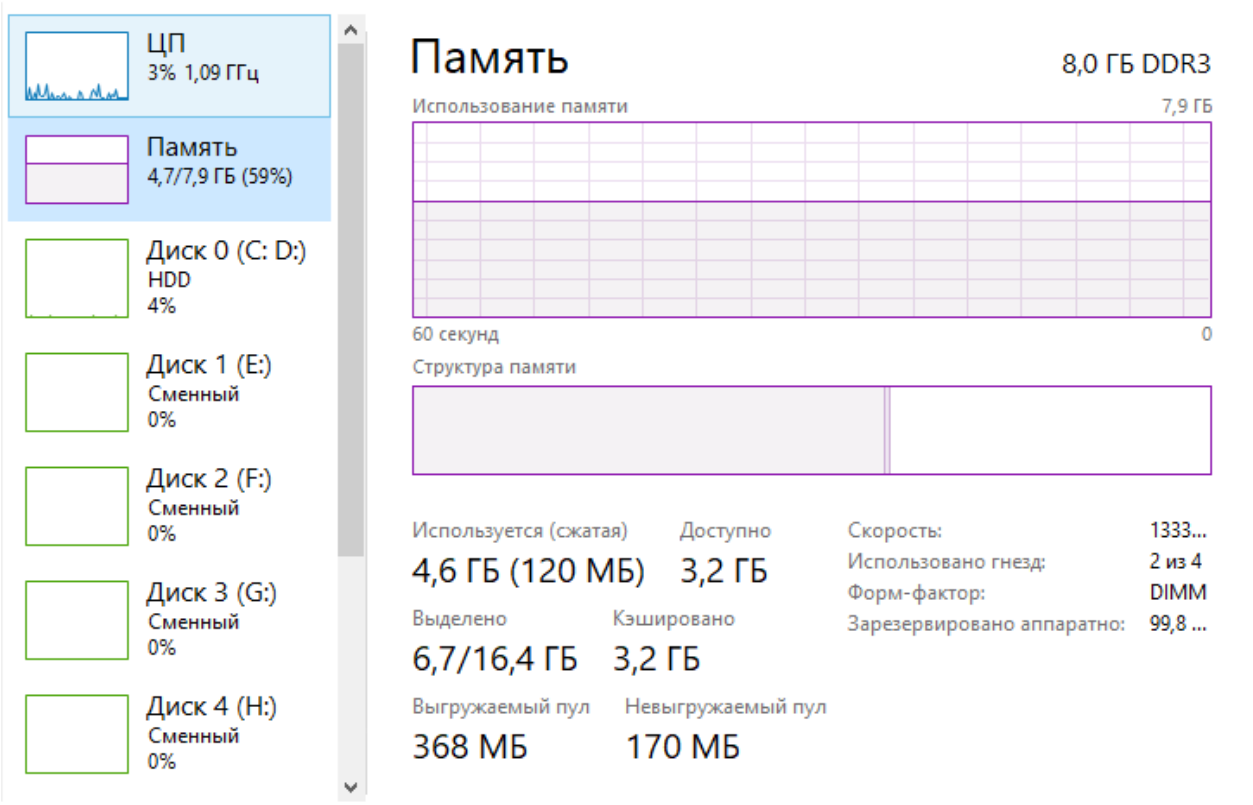

Рисунок 48 — Збільшений обсяг використання операційної пам’яті

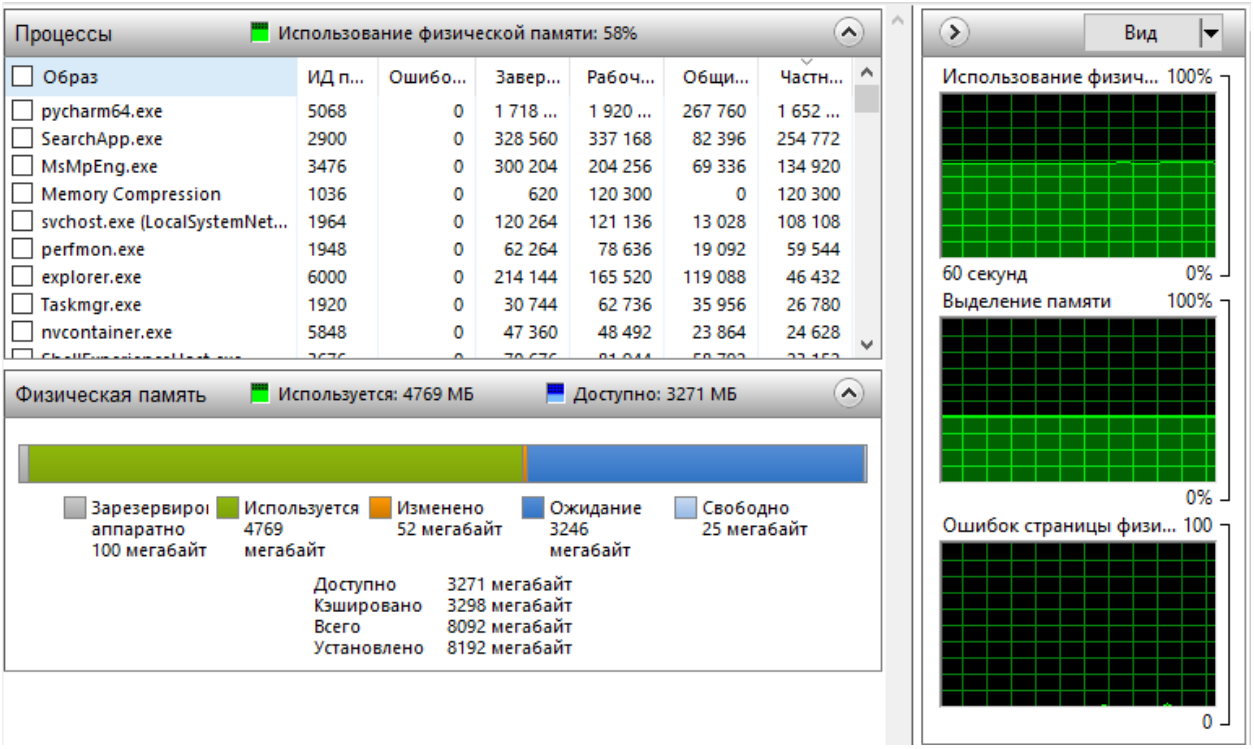

Рисунок 49 — Обсяг використаної та вільної операційної пам’яті

На основі проведеного моніторингу можна зробити висновок, що необхідний обсяг оперативної пам’яті збільшився на 20 — 25%.

На рисунках 50 — 54 можна побачити моніторинг навантаження з увімкненою середовищем розробки PyCharm та увімкненим розробленим класифікатором загроз на мові Python.

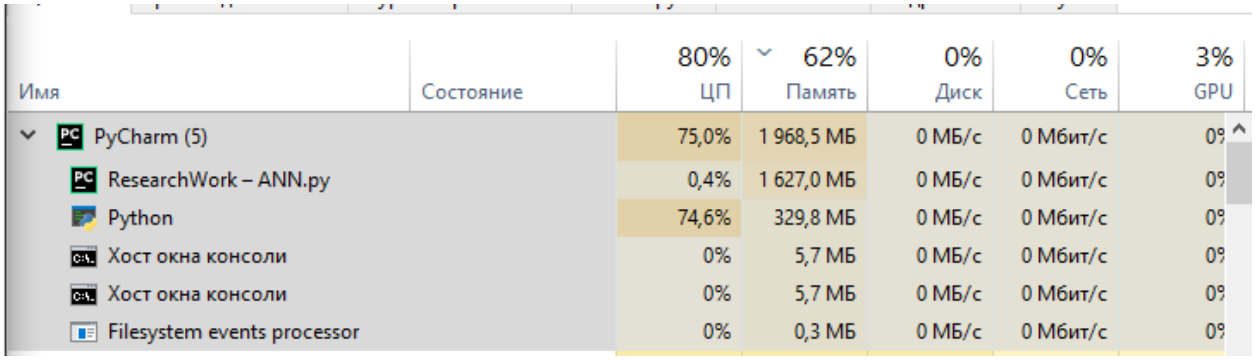

Рисунок 50 — Використання системних ресурсів під час роботи класифікатора загроз

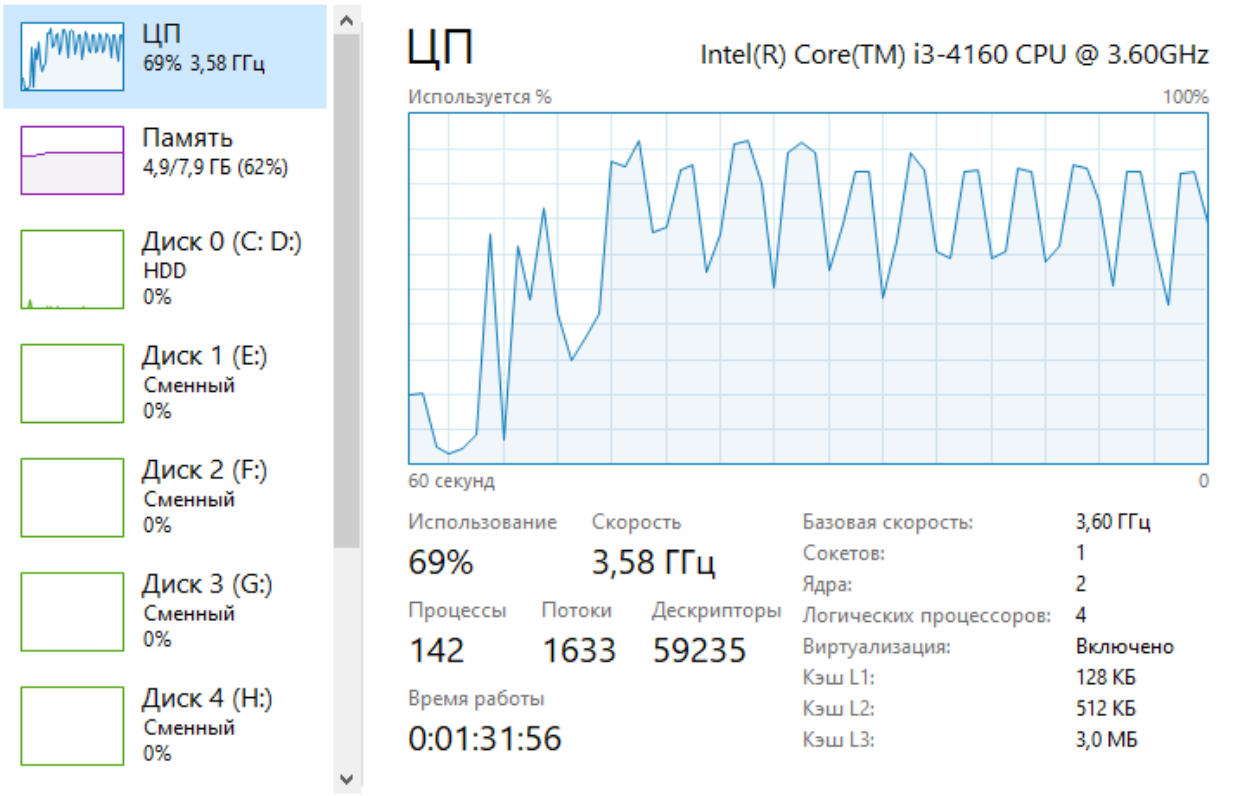

Рисунок 51 — Навантаження ЦП під час роботи системи класифікатора

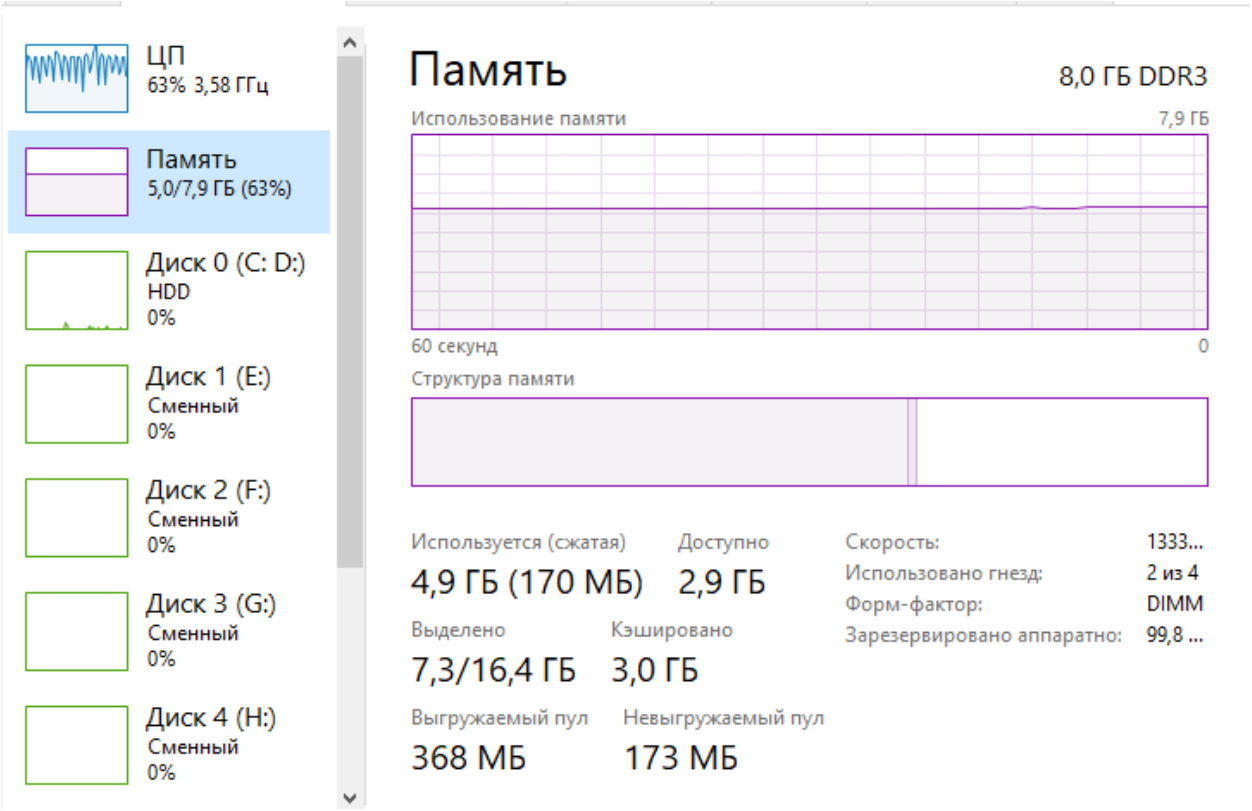

Рисунок 52 — Обсяг використаної операційної пам’яті під час роботи класифікатора

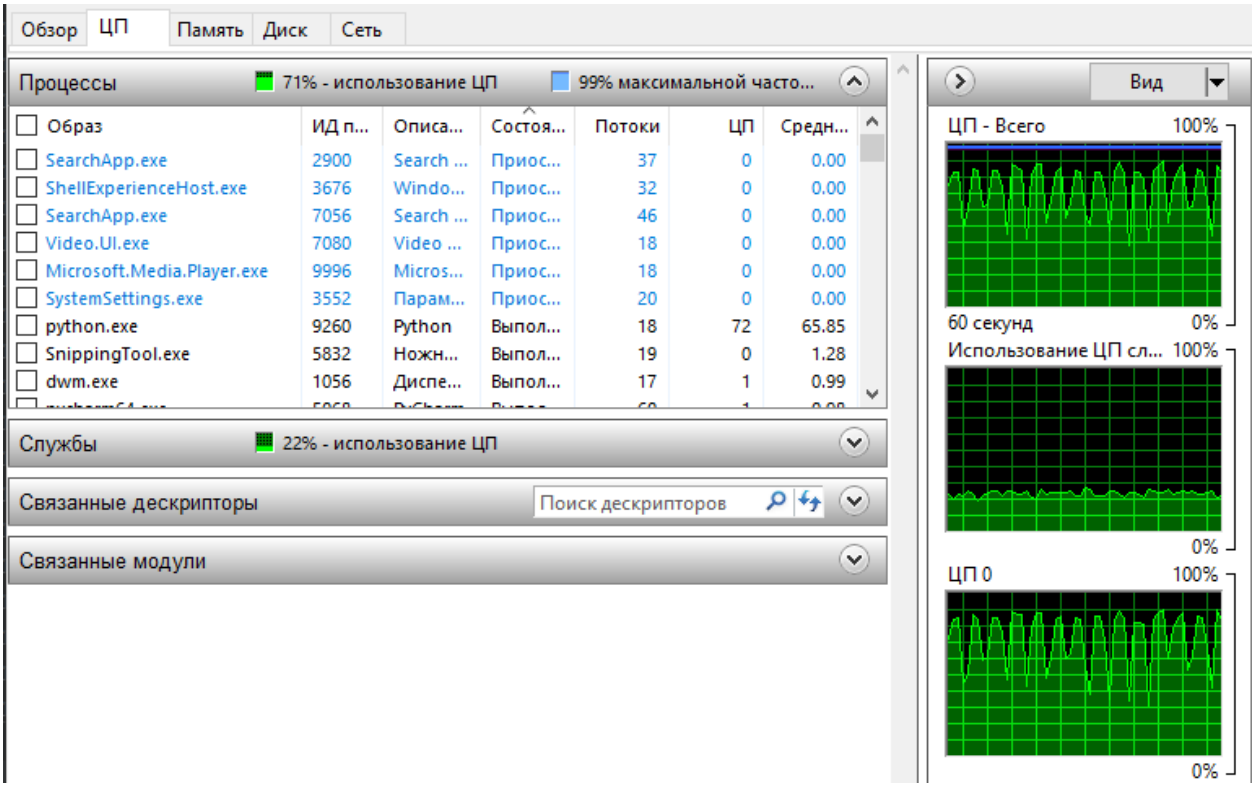

Рисунок 53 — Використання ЦП в моніторингу ресурсів під час роботи класифікатора

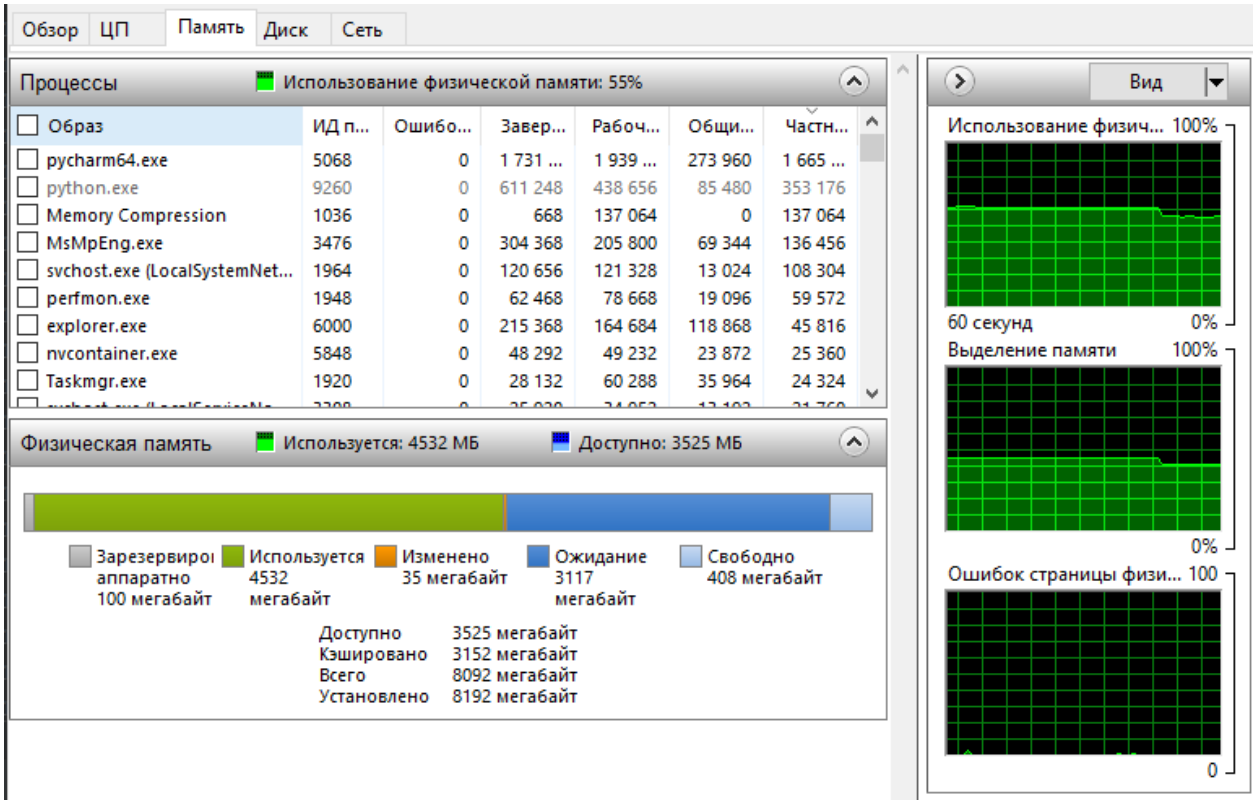

Рисунок 54 — Обсяг використаної оперативної пам’яті в моніторингу ресурсів під час роботи класифікатора

Виходячи з представленого вище моніторингу можна зробити наступні висновки:

- навантаження на ОЗУ під час навчання моделі особливо не збільшилося у порівняні з моніторингом під час простоювання системи. Різниця навантажень 3 — 4%;
- основне навантаження відбувається на процесор. Завантаження процесора відбувається до 80% під час роботи з моделлю, у той час як при простоюванні навантаження не перевищує 5%;
- При використанні процесора на 2/4 ядра/потоку з урахуванням часу на основні етапи з роботою над моделлю та завантаження процесорного ресурсу результат можна вважати задовільним. За наявності процесора на 6/12 ядер/потоків можна було б отримати виграш за часом роботи, а також значно скоротити витрати ресурсів системи.

4.1.2 Оцінка моделі програмної системи

На рисунку 55 можна побачити безпосередньо процес компіляції, тренування та тестування розробленої моделі класифікатора. На кожному кроці модель бере по 15 елементів з усього обсягу даних та оброблює їх. Після чого виводить безпосередньо час який вона витрачає на кожен крок, коефіцієнт втрати та точності. З усього обсягу згенерованих даних в 4763 зразків, було вирішено розподілити їх: 4290 елементів використовувалися для навчання нейронної мережі та 473 для тестування.

```
15/15 [==============================] - 0s 2ms/step - loss: 0.1431 - accuracy: 0.9535
[+] Model has been tested. Accuracy: 95.35%
[+] Model has been compiled
[+] Model has been trained
15/15 [==============================] - 0s 2ms/step - loss: 0.1249 - accuracy: 0.9683
[+] Model has been tested. Accuracy: 96.83%
[+] Model has been compiled
[+] Model has been trained
```

Рисунок 55 — Процес класифікації

Час навчання та повної класифікації моделі нейронної мережі становить 7 хвилин та 26 секунд.

Час виконання класифікації попередньо навченої моделі нейронної мережі становить: 56.36 секунд.

Максимальна точність моделі на тестових даних досягла 98,10%.

4.2 Результати роботи розробленої комп’ютерної системи для класифікації загроз автомобільної самоорганізованої мережі VANET

Результатом виконання класифікації моделі маємо матрицю помилок (помилок класифікації) яка була отримана за допомогою засобів бібліотеки sklearn та безпосередньо сам звіт про класифікації, який демонструє значення

точності, повноти, F1 міри та кількості елементів які були присутні при класифікаціїї кожного стану мережі. На рисунку 56 можна побачити результат класифікації в консолі.

```
Confusion matrix, without normalization
[[ 75   1   3   1   0]
 [  1  83   0   0   0]
 [  0   0  93   2   0]
 [  0   1   2 207   0]
 [  0   0   0   0   4]]
Normalized confusion matrix
[[0.9375     0.0125     0.0375     0.0125     0.        ]
 [0.01190476 0.98809524 0.         0.         0.        ]
 [0.         0.         0.97894737 0.02105263 0.        ]
 [0.         0.0047619  0.00952381 0.98571429 0.        ]
 [0.         0.         0.         0.         1.        ]]
[*] Classification report:
              precision    recall  f1-score   support

           0       0.99      0.94      0.96        80
           1       0.98      0.99      0.98        84
           2       0.95      0.98      0.96        95
           3       0.99      0.99      0.99       210
           4       1.00      1.00      1.00         4

    accuracy                           0.98       473
   macro avg       0.98      0.98      0.98       473
weighted avg       0.98      0.98      0.98       473
```

Рисунок 56 — Результат класифікації розробленої моделі

На рисунку 57 та рисунку 58 показані матриці помилок (з нормалізацією та без) які були отримані за допомогою засобів бібліотеки sklearn. Як очевидно за даними рисунками, діагональні елементи матриць явно виражені, що означає високий результат виявлення елементів кожного класу.

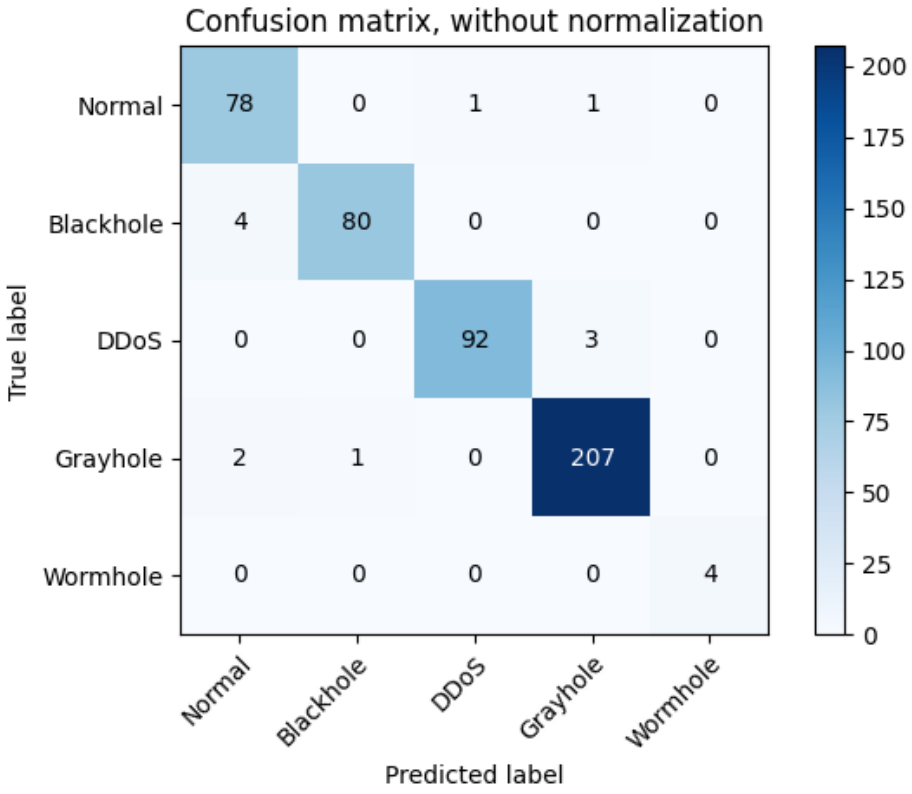

Рисунок 57 — Матриця помилок без нормалізації

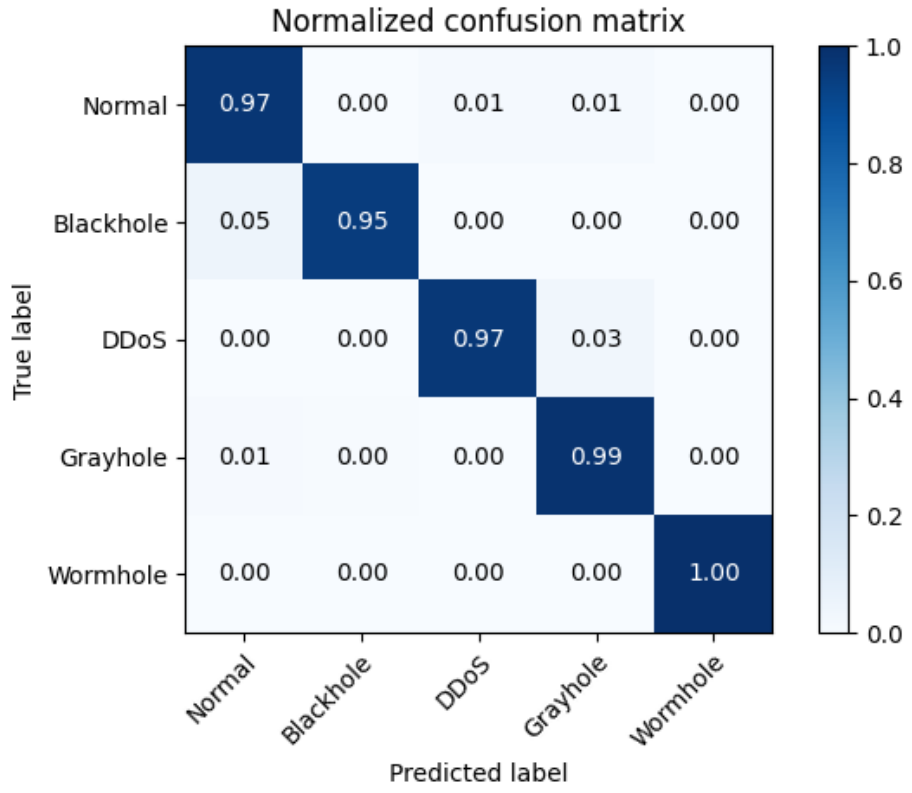

Рисунок 58 — Нормалізована матриця помилок

Для оцінки якості роботи алгоритму на кожному з класів окремо вводять метрики precision (точність), recall (повнота) та F1-міра (F1-score). Точність можна інтерпретувати як частку об'єктів, названих класифікатором позитивними і при цьому дійсно є позитивними, а повнота показує, яку частку об'єктів позитивного класу з усіх позитивного об'єктів знайшов алгоритм. Чим вища точність та повнота, тим краще. Але на практиці максимальна точність і повнота не досяжна одночасно і доводиться шукати певний баланс. Тому, вигадали метрику, яка поєднує в собі інформацію про точність і повноту алгоритму - F1-мера. Дана метрика є гармонійним середнім між точністю і повнотою. Вона прагне нуля, якщо точність чи повнота також прагнуть нуля.

У табл. 5 представлені різні метрики для кожного класу, до яких нейронна мережа відносить вхідні дані.

Таблиця 5 — метрики оцінки якості ШНМ

Клас	*Точність (Precision)*	*Повнота (Recall)*	*F_1-міра (F_1-score)*
Normal	0,97	0,96	0,97
Blackhole	0,98	0,99	0,98
DDoS	0,99	0,95	0,97
Grayhole	0,98	1,00	0,98
Wormhole	1,00	1,00	1,00
Середнє (macro-averaging) / спільне	0,98	0,98	0,98

Як видно з даної таблиці, максимальні значення точності, повноти та F1-міри спостерігаються для класу wormhole (червоточина). Це можна пояснити

тим, що ця атака легко виявлена, і тестова вибірка має невелику кількість елементів для даного класу. Всі значення точності та повноти для кожного класу приблизно рівні, найбільшу різницю показує клас DDoS — 0,99 проти 0,95. Це означає, що частка об'єктів, які класифікатор відніс до класу DDoS і які при цьому дійсно є DDoS (точність) більше, ніж частка об'єктів, які класифікатор відніс до DDoS щодо всіх об'єктів цього класу (повнота).

4.3 Висновки з розділу 4

1. Розроблена комп’ютерна система детектування та класифікації загроз у мережі VANET з використання машинного навчання за допомогою засобів бібліотеки Keras.

2. Була зроблена оцінка продуктивності та навантаження ОС під час 3 різних сценаріїв, результат показав що навіть досить слабкий процесор може впоратись з повною класифікацією моделі за досить невеликий обсяг часу.

3. Результат класифікації моделі показав досить високий коефіцієнт точності на тестовій вибірці з 473 елементів — 98%.

4. Створена матриця помилок з нормалізацією та без, для оцінки результатів виявлення елементів кожного класу.

5. Для оцінки якості роботи алгоритму на кожному з класів були введені відповідні метрики precision (точність), recall (повнота) та F1-міра (F1-score).

ВИСНОВКИ

1. Досліджена проблема класифікації загроз у мережі VANET: розглянуті переваги та недоліки існуючих технологій та систем для її вирішення, сфери застосування таких систем та сучасні підходи до її розв'язання, розглянуто принципи використання технології автономних обчислень у мережах VANET за допомогою протоколу ADM.

2. Відбулося ознайомлення з політикою безпеки мережі VANET, розглянуті вимоги безпеки, різноманітні види загроз та їх вплив на мережу транспортних засобів з методами для їх вирішення.

3. Досліджені засоби для вирішення поставленої проблеми: з існуючих архітектур нейронних мереж був обраний багатошаровий перцептрон для вирішення проблеми, обрані відповідні бібліотеки для навчання та використання моделі нейронної мережі.

4. Був згенерований трафік мережі VANET за допомогою симулятора NS-3 у кількості 4763 зразків, частина з яких (90%) застосовувалися для навчання мережі, а решта (10%) використовувалися для тестування, розроблений парсер даних для виделення ознак з файлів, отриманих за допомогою симулятора та занесення їх до csv-таблиць для використання у якості датасетів під час навчання та тестування.

5. Розроблена комп'ютерна система виявлення загроз в мережі VANET з використанням машинного навчання.

6. Модель нейронної мережі була навчанна з використанням 4763 зразків даних згенерованих симулятором NS-3 та перетворених за допомогою власного парсера даних у csv-таблицю для навчання, час за який мережа навчалась становив 7 хвилин 26,31 секунд, після всього модель була протестована з використанням 473 зразків з таблиці для тестування, процес класифікації зайняв 56,36 секунд.

7. Досліджені результати роботи розробленої комп'ютерної системи: під час процесу класифікації модель показала досить високий коефіцієнт точності

на тренувальних та тестових даних, середня точність 94.5%, а максимальна досягла відмітки 98,1% що каже про високу ефективність розробленої системи.

8. Для оцінки якості роботи алгоритму на кожному з класів були введені метрики точності (precision), повноти(recall) та F1-міри(F1-score). Усі значення для кожного классу (загрози) були приблизно рівні що підтверджує високу точність класифікатора, а значення для класу wormhole досягли 1, це говорить про легкість виявлення даної загрози.

9. Поставлена проблема класифікації загроз у мережі VANET була повністю вирішена з допустимою точністю, та може буди розширена новими видами загроз для підвищення цінності розробленої системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Abu Taleb A. VANET Routing Protocols and Architectures: An Overview. Journal of Computer Science. 2018;14:421–434. URL: https://doi.org/10.3844/jcssp.2018.423.434 (дата звернення: 23.04.2023).

2. Akyildiz I.F., Akan O.B., Chen C. et al. InterPlaNetary Internet: state-of-the-art and research challenges // Computer Networks. 2013.

3. Allal S., Boudjit S. Geocast Routing Protocols for VANETs: Survey and Guidelines. In: 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing; 2012. P. 323–328.

4. Allani S., Chbeir R., Yeferny T., Yahia S. Smart Directional Data Aggregation in VANETs. In: 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA); 2018. P. 63–70.

5. Allani S., Yeferny T.R., Yahia C.S. DPMS: A Swift Data Dissemination Protocol Based on Map Splitting. In: 40th IEEE Annual Computer Software and Applications Conference; 2016. P. 817–822.

6. Bangui H., Ge. M., Buhnova B., Trang L.H. Towards faster big data analytics for anti-jamming applications in vehicular ad-hoc network. Transactions on Emerging Telecommunications Technologies. 2021;32(10). URL: https://doi.org/10.1002/ett.4280 (дата звернення: 07.06.2023).

7. Cencioni P., Di Pietro R. (2008) Comput. Commun (12), pp. 2790–2802.

8. Dharmendra S., Pradhan S. Data Dissemination Techniques in Vehicular Ad Hoc Network. International Journal of Computer Applications. 2010;8(10):35–39.

9. Do N.M., HSU C.-H., SINGH J.P., VENKATASUBRAMANIAN N. Massive live video distribution using hybrid cellular and ad hoc networks // IEEE International Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM), June 2018, Lucca, Italy. P. 1-9.

10. Dotzer F., Kohlmayer F., Kosch T., Strassberger M. (2005) 2nd International Workshop on Intelligent Transportation, Hamburg, Germany.

11. Eidenbenz S., Kumar A., Zust S. Equilibria in Topology Control Games for Ad Hoc Networks and Generalizations // Mobile Network and Applications. - 2006. -Vol. 11, No. 2.-P. 143–159.

12. Farzad Sabahi (2011) Third International Conference on Computational Intelligence, Communication Systems and Networks.

13. I. Ahmed Soomro, Hasbullah H.B., J.lb. Ab Manan (2010).

14. IEEE Std. 802.11 Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. 2012. URL: http://standards.ieee.org/getieee802/download/802.11-2012.pdf (дата звернення: 21.07.2023).

15. IEEE Std. 802.15.4 Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPAN). 2011. - URL: http://standards.ieee.org/getieee802/download/802.15.4-2011.pdf (дата звернення: 21.07.2023).

16. Jackson M.O. A Survey of Models of Network Formation: Stability and Efficiency // In: Group Formation in Economics: Networks, Clubs and Coalitions, edited by Gabrielle Demange and Myrna Wooders. -- Cambridge University Press: Cambridge, UK, 2005.

17. J. Schiller, Mobilkommunikation. Addison-Wesley Verlag, 2000.

18. Khakimov A.A., Suminov A.V., Mutkhanna A.S. [Development of a method for organizing the distribution of edge computing in VANET networks]. Informatsionnye tekhnologii i telekommunikatsii. 2019;7(2):47–55. (In Russ.).

19. Langley C., Fucas R., Fu H. (2008) IEEE Int. Conf. on Electro/Information Technology, pp. 223–226.

20. Lee M., Atkison T. VANET applications: Past, present, and future. Vehicular Communications. 2021;28:2214–2096.

21. Mbarek N., Abdou W., Darties B. Autonomic Computing and VANETs: Simulation of a QoS-based Communication Model. In: Networking Simulation for Intelligent Transportation Systems; 2017. P. 211–234.

22. National institute of standards and technology. URL: http://www.antd.nist.gov/index.shtml (дата звернення: 13.05.2023).

23. Nikonov V.I., Litvinov G.A., Shcherba E.V. [Securing Routing Protocols for Dynamic Telecommunication Networks]. Doklady TUSUR. 2018;21(3):19–29.

24. Perkins and C. E., Ad hoc networking. Addison-Wesley Verlag, 2001.

25. Proskochylo A., Vorobyov A., Zriakhov M. Overview of possibilities to improve efficiency of self-organizing networks 2014, First International Scientific-Practical Conference «Problems of Infocommunications. Science and Technology». IEEE, 2014. - P. 118–119.

26. Raya M., Hubaux J. P. The security of vehicular ad hoc networks Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks. – 2005. С. 11–21.

27. Santi P. Topology Control in Wireless Ad Hoc and Sensor Networks // Journal ACM Computing Surveys (CSUR). Vol. 37, No. 2. P. 164–194.

28. Sarkar S. K., Basavaraju T. G., Puttamadappa C. 2007. Ad hoc mobile wireless networks: principles, protocols and applications. CRC Press, 349.

29. S. Toumpis and D. Toumpakaris, “Wireless ad hoc networks and related topologies: applications and research challenges,” e & i Elektrotechnik und Informationstechnik, vol. Volume 123, pp. 232–241, 2006.

30. Tomar R, Prateek M, Sastry GH. Vehicular Adhoc Network (VANET) – An Introduction. International Journal of Control Theory and Applications. 2016;9(18):8883–8888.

31. Wattenhofer R., Zollinger A. XTC: A Practical Topology Control Algorithm for Ad-Hoc Networks // 18th International Parallel and Distributed Processing Symposium, Santa Fe, NM, USA, 2014.

32. Yan G., Olariu S., Weigle M. (2008) Comput. Commun (12), pp. 2883–2897.

33. Zarifzadeh S., Yazdani N., Nayyeri A. Energy-efficient topology control in wireless ad hoc networks with selfish nodes // Computer Networks. 2016. Vol. 56. P. 902–914.

Декларація
академічної доброчесності
здобувача вищої освіти ЗНУ

Я, Шумаков Віталій Юрійович, студент_2_курсу,
форми здобуття освіти денної,
Інженерного навчально-наукового інституту ім. Ю.М. Потебні ЗНУ
Спеціальності 121 Інженерія програмного забезпечення,
адреса електронної пошти ipz18bd-11@stu.zsea.edu.ua,
підтверджую, що написана мною кваліфікаційна робота на тему «Комп'ютерна система виявлення загроз в мережі VANET з використанням машинного навчання»
відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений/ознайомлена;
- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;
- згоден/згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою Інтернет-системи, а також на архівування роботи в базі даних цієї системи.

Дата_30.12.2023_____ Підпис_____ В.Ю. Шумаков
(студент)

Дата_30.12.2023_____ Підпис_____ Н.П. Полякова
(науковий керівник)