Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Модуль глибинного навчання для виявлення аномалій у сигналах з CAN шини автомобіля
(тема)

Виконав:
студент 2 курсу, групи СШІм-19-1
Остапенко М.О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник к.т.н., доц. Узлов Д. Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«» _____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

студентові Остапенку Максиму Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Модуль глибинного навчання для виявлення аномалій у сигналах з CAN шини автомобіля

затверджена наказом університету від 30 10 20 20 р. № 1497

2. Термін подання студентом роботи до екзаменаційної комісії 15 грудня 20 20 р.

3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел та відомих наукових проектів щодо розробки систем виявлення аномалій у сигналах з CAN шини автомобіля, Python documentation

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та формалізована постановка задачі
2) Алгоритм глибинного навчання для виявлення аномалій у сигналах з CAN шини автомобіля
3) Імітаційне моделювання

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)

Рисунок 1 – приклад аномалій у простому двовимірному наборі даних
Рисунок 2 – шарова структура вузла CAN
Рисунок 3 – схематичне зображення кадру даних (Data Frame)
Рисунок 4 – схематичне представлення даних шини CAN після попередньої обробки байтів сигналів
Рисунок 5 – кількість різних ідентифікаторів пакетів, за якими спостерігається кожен CAN ID
Рисунок 6 – архітектура для виявлення аномалій в сигналах CAN шини автомобіля
Рисунок 7 – структура вузлу LSTM
Рисунок 8 – структура найпростішого автокодувальника
Рисунок 9 – графік функції активації ELU
Рисунок 10 – 99.99% перцентиль заданого розподілу даних
Рисунок 11 – графічне представлення фрагменту нормальних даних

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	Дата
Основна частина	к.т.н., доц. Узлов Д.Ю.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на дипломну роботу	01.11.2020	виконано
2	Аналіз предметної галузі і постановка завдання	07.11.2020	виконано
3	Дослідження методів виявлення аномалій	14.11.2020	виконано
4	Аналіз існуючих проблем даної предметної галузі	17.11.2020	виконано
5	Розробка алгоритму з використанням	21.11.2020	виконано
	штучномережевої системи		
6	Створення імітаційної моделі	23.11.2020	виконано
7	Тестування і опрацювання імітаційної моделі	25.11.2020	виконано
8	Написання пояснювальної записки	01.12.2020	виконано
9	Попередній захист	9.12.2020	виконано
10	Захист перед ЕК	16.12.2020	

Дата видачі завдання 01 11 2020 р.

Студент _____
(підпис)

Керівник роботи _____ к.т.н., доц. Узлов Д.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Записка пояснювальна: 77 с., 9 табл., 19 рис., 2 дод., 28 джерел.

АВТОКОДУВАЛЬНИК, АНОМАЛІЯ, ВУЗЬКЕ МІСЦЕ, ГЛИБИННЕ НАВЧАННЯ, ДОВГА КОРОТКОЧАСНА ПАМ'ЯТЬ, МЕРЕЖА КОНТРОЛЕРІВ, РЕКУРЕНТНА НЕЙРОННА МЕРЕЖА

В магістерській атестаційній роботі розглядається задача автоматичного виявлення аномалій у повідомленнях з CAN шини автомобіля.

Метою магістерської атестаційної роботи є створення модуля глибинного навчання на основі нейронної мережі довгої короткочасної пам’яті.

Об'єктом дослідження є процес обробки повідомлень з CAN шини, які представляють собою декілька сигналів, кількість яких залежить від CAN ID, за допомогою послідовно працюючих нейронної підмережі LSTM та повно зв’язних шарів.

Предметом дослідження є методи методи виявлення аномалій в задачах інтелектуального аналізу даних.

Методи дослідження базуються на теорії обчислювального інтелекту, а саме на методах теорії штучних нейронних мереж для побудови необхідної архітектури нейромережевої системи, яка складається з LSTM підмереж, спільного прихованого вектора та повно зв’язних шарів, що дозволяє отримувати реконструкції сигналу для подальшого їх порівняння з вхідними значеннями задля вивлення аномалії.

Імітаційне моделювання застосовується для перевірки якості виявлення аномалій у сигналах з CAN шини автомобіля за допомогою розробленої системи.

РЕФЕРАТ

Пояснительная записка: 77 с., 9 табл., 19 рис., 2 прил., 28 источников.

АВТОКОДИРОВЩИК, АНОМАЛИЯ, ГЛУБИННОЕ ОБУЧЕНИЕ, ДОЛГАЯ КРАТКОСРОЧНАЯ ПАМЯТЬ, РЕКУРРЕНТНАЯ НЕЙРОННАЯ СЕТЬ, СЕТЬ КОНТРОЛЛЕРОВ, УЗКОЕ МЕСТО

В магистерской аттестационной работе рассматривается задача выявления аномалий в сигналах CAN шины автомобиля.

Целью магистерской аттестационной работы является создание модуля глубинного обучения на основе нейронной сети долгой краткосрочной памяти.

Объектом исследования является процесс обработки сообщений с CAN шины автомобиля, которые представляют собой несколько сигналов, количество которых зависит от CAN ID, с помощью последовательно работающих нейронной подсети LSTM и полносвязных слоев.

Предметом исследования являются методы определения аномалий в задачах интеллектуального анализа данных.

Методы исследования базируются на теории вычислительного интеллекта, а именно на методах теории искусственных нейронных сетей для построения необходимой архитектуры нейросетевой системы, состоящей из LSTM подсетей, общего скрытого вектора и полносвязных слоев, что позволяет получать реконструкции сигнала для дальнейшего их сравнения с входными значениями, чтобы определять аномалии.

Имитационное моделирование применяется для проверки качества выявления аномалий в сигналах с CAN шины автомобиля с помощью разработанной системы.

ABSTRACT

Explanatory note: 77 pages, 19 figures, 9 tables, 28 sources, 17 formulas.

ANOMALY, AUTOENCODER, BOTTLENECK, CONTROLLER AREA NETWORK, DEEP LEARNING, LONG SHORT-TERM MEMORY, RECURRENT NEURAL NETWORK

In the Master's certification work, the problem of anomalies detection for CAN bus data is considered.

The aim of the bachelor's certification work is to create a deep learning module based on the long short-term memory neural network.

The object of the study is the process of processing CAN bus messages, which represent several signals, the number of which depends on the CAN ID with the help of an ensemble of consistently working neural subnet LSTM and fully connected layers.

The subject of the study is the methods of anomaly detection in data mining tasks.

The methods of investigation are based on the theory of computational intelligence, namely, on the methods of the theory of artificial neural networks for constructing a neural net system, which consists of LSTM subnetworks, joint latent vector, and fully connected layers that allows getting signal reconstructions for their further comparing with the input values to detect anomalies.

Simulation modeling is used to test the quality of anomalies detection for CAN bus data using the synthesized architecture.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ACK – Acknowledgment – підтвердження;

CAN – Controller Area Network – локальна мережа контролерів;

CRC – Cyclic Redundancy Check – циклічний надлишковий код;

DLC – Data Length Code – довжина даних;

DOS – Denial Of Service – відмова обслуговуванні;

ECU – Electronic Control Unit – електронні блоки управління;

ELU – Exponential Lineal Unit – експоненціально-лінійна одиниця;

LSTM – Long Sort-term Memory – довга короткочасна пам’ять;

MSB – Most significant bit – старший біт;

OCSVM – One Class Support Vector Machines – однокласовий метод опорних векторів;

RELU – Rectified Linear Unit – випрямляч;

RNN – Recurrent Neural Network – рекурентна нейронна мережа;

RTR – Remote Transmission Request – запит на передачу;

SOF – Start Of Frame – початок кадру;

SVM – Support Vector Machines – метод опорних векторів;

WI-FI – Wireless Fidelity – бездротова правдивість відтворення.

ВСТУП

У сучасному світі автомобілі все більше і більше стають комп’ютеризованими, використовуючи такі технології, як Bluetooth, Wi-Fi або підключаються до смартфонів. Хоча це спрощує життя водія, воно одночасно відкриває нові шляхи для потенційних віддалених атак на електронні блоки управління автомобілів.

Захоплений блок управління може дозволити зловмисникам розміщувати повідомлення у внутрішній комунікаційній мережі автомобіля і, таким чином, наприклад, викликати раптову поломку або вимкнення двигуна, що потенційно може спричинити дорожньо-транспортні пригоди. Це може мати ще більш згубні наслідки для автономних транспортних засобів. Отже, виявлення замаху на атаки в автомобільних мережах відповідає інтересам безпеки руху [1].

Локальна мережа контролерів (CAN) – це стандартний метод зв'язку між електронними блоками управління (ЕБУ) автомобілів. Наприклад, ЕБУ може надсилати інформацію про предмети на дорозі, щоб система гальмування автомобіля могла реагувати відповідним чином. Однак у CAN шині автомобіля відсутні механізми безпеки, і нещодавно було показано, що на неї можна атакувати віддалено, що відкриває безліч можливостей для зловмисників.

Отже, бажано контролювати трафік CAN шини автомобіля для виявлення вторгнень, які далі будуть називатися аномаліями. Основна увага приділяється заснованим на правилах та статистичним методам виявлення відомих сценаріїв атак. Хоча багато типів вторгнень можна ефективно виявити за допомогою цих підходів, конфігурація такої системи виявлення займає багато часу, вимагає експертизи домену, і навряд чи вдасться виявити невідомі сценарії атак [2].

Більше того, складно створювати правила, які фіксують основну поведінку сигналів або фізичні залежності між ними. З досягненнями

глибокого навчання за останні роки стають доступними нові інструменти, які мають потенціал виявлення невідомих атак [3].

Треба зазначити, що виявлення аномалій, або як ще їх називають вторгненнями, у просторі сигналів даних шини CAN автомобіля може мати інші додатки, крім виявлення вторгнень, наприклад раннє виявлення технічних несправностей.

В свою чергу істинно негативний показник точності системи має наближатися до відмітки ста відсотків, тому що кількість помилкових тривог має бути мінімальною, щоб зменшити кількість звернень до механіків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ФОРМАЛІЗОВАНА ПОСТАНОВКА ЗАДАЧІ

1.1 Поняття аномалії в сигналах з CAN шини автомобіля

1.1.1 Загальне поняття аномалії

Виявлення аномалій відноситься до проблеми пошуку шаблонів у даних, які не відповідають очікуваній поведінці. Ці невідповідні закономірності часто називають аномаліями, відхиленнями, суперечливими спостереженнями, винятками, викидами, несподіванками, особливостями або забрудненнями в різних областях застосування. З них аномалії і викиди – це два терміни, що використовуються найчастіше в контексті виявлення аномалій; іноді взаємозамінні. Виявлення аномалій знаходить широке застосування в самих різних додатках, таких як виявлення шахрайства на кредитних картках, страхування чи охорона здоров'я, виявлення вторгнень для кібербезпеки, виявлення несправностей у критично важливих системах безпеки та військове спостереження за діями ворога.

Важливість виявлення аномалій обумовлена тим, що аномалії даних перетворюються на важливу інформацію в самих різних областях застосування. Наприклад, аномальна картина трафіку в комп'ютерній мережі може означати, що зламаний комп'ютер розсилає конфіденційні дані в несанкціонований пункт призначення. Аномальне МРТ-зображення може свідчити про наявність злоякісних пухлин. Аномалії даних про трансакції на кредитній картці можуть свідчити про крадіжку кредитної картки чи особистих даних, або аномальні показники датчика космічного корабля можуть означати несправність якоїсь складової космічного корабля.

Виявлення відхилень чи аномалій у даних вивчалось у статистичній

спільноті ще в дев'ятнадцятому столітті. З часом у декількох дослідницьких спільнотах було розроблено різноманітні методи виявлення аномалій. Багато з цих методів були спеціально розроблені для певних доменів програм, тоді як інші є більш загальними [3].

Аномалії – це закономірності в даних, які не відповідають чітко визначеному поняттю нормальної поведінки. Рисунок 1.1 ілюструє аномалії у простому двовимірному наборі даних. Дані мають дві нормальні області,. N_1 та N_2, оскільки більшість спостережень лежать у цих двох регіонах. Точки, які знаходяться на відстані від регіонів, наприклад, точки o_1 та o_2 та точки в області O_3, є аномаліями.

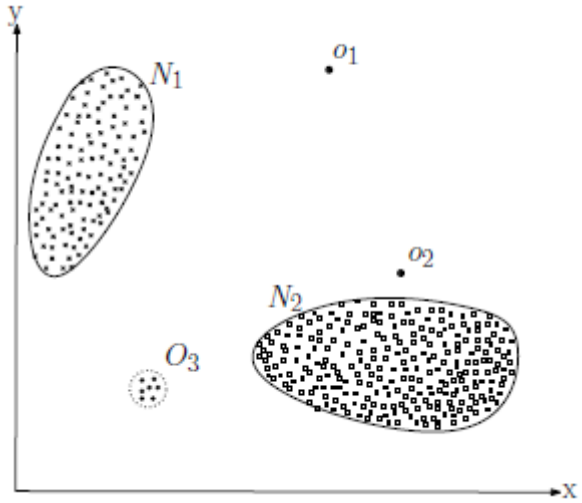

Рисунок 1.1 – Приклад аномалій у простому двовимірному наборі даних

Аномалії можуть бути спричинені даними з різних причин, таких як зловмисна діяльність, наприклад, шахрайство з кредитними картками, терористична діяльність або поломка системи, але всі причини мають спільну характеристику, якою вони цікаві аналітику.

Виявлення аномалії пов'язане із видаленням та пристосуванням шуму, які мають справу з небажаним шумом у даних. Шум можна визначити як явище в даних, яке не цікавить аналітика, але є перешкодою

для аналізу даних.

Видалення шуму зумовлене необхідністю видалення небажаних об’єктів перед тим, як проводитись будь-який аналіз даних. Пристосування шуму відноситься до імунізації оцінки статистичної моделі проти аномальних спостережень.

Іншою темою, пов’язаною з виявленням аномалій, є виявлення новизни, яка спрямована на виявлення раніше неспостережуваних, нових, закономірностей у даних, наприклад, нової теми обговорення в групі новин. Різниця між новими шаблонами та аномаліями полягає в тому, що нові шаблони зазвичай включаються в нормальну модель після виявлення [4].

На абстрактному рівні аномалія визначається як шаблон, який не відповідає очікуваній нормальній поведінці. Отже, прямий підхід до виявлення аномалії полягає у визначенні області, що представляє нормальну поведінку, та оголошенні будь-якого спостереження в даних, яке не належить до цієї нормальної області, як аномалії. Але кілька факторів роблять цей, здавалося б, простий підхід дуже складним:

– визначення нормальної області, яка охоплює всі можливі нормальні поведінки, є дуже складною задачею. Крім того, межа між нормальною та аномальною поведінкою часто не є точною. Таким чином, аномальне спостереження, яке лежить близько до межі, насправді може бути нормальним, і навпаки;

– коли аномалії є результатом зловмисних дій, зловмисники часто пристосовуються до того, щоб аномальні спостереження здавалися нормальними, тим самим роблячи завдання формування нормальної поведінки більш важким;

– у багатьох доменах нормальна поведінка продовжує розвиватися, і сучасне поняття нормальної поведінки може не бути достатньо репрезентативним у майбутньому;

– точне поняття аномалії є різним для різних доменів. Наприклад, у

медичній галузі невелике відхилення від норми, наприклад, перепади температури тіла, може бути аномалією, тоді як подібне відхилення в домені фондового ринку може розглядатися як нормальний. Таким чином, застосування техніки, розробленої в одному домені, до іншого не є правильним рішенням;

– доступність маркованих даних для навчання та перевірки моделей, що використовуються методами виявлення аномалій, як правило, є основною проблемою;

– часто дані містять шум, який, як правило, схожий на фактичні аномалії, а отже, його важко розрізнити та видалити.

Через вищезазначені проблеми, задачу виявлення аномалії у найзагальнішому вигляді вирішити непросто. Насправді, більшість існуючих методів виявлення аномалій вирішують конкретну постановку проблеми. Формулювання залежить від різних факторів, таких як природа даних, наявність маркованих даних, тип аномалій, що підлягають виявленню тощо. Часто ці фактори визначаються областю застосування, в якій необхідно виявити аномалії. Дослідники прийняли концепції з різних дисциплін, таких як статистика, машинне навчання, видобуток даних, теорія інформації, спектральна теорія, і застосували їх до конкретних проблем.

1.1.2 Основні характеристики CAN шини

Мережа контролерів (CAN) – це послідовний протокол зв'язку, який ефективно підтримує розподілене управління в режимі реального часу з дуже високим рівнем безпеки.

Область його застосування варіюється від високошвидкісних мереж до недорогих мультиплексних проводів. В автомобільній електроніці блоки управління двигуном, датчики, протиковзні системи тощо підключаються за допомогою CAN із бітрейтом до 1 Мбіт / с [5].

CAN наділена наступними можливостями та властивостями:

– встановлення пріоритетів повідомлень;

– гарантія часу затримки;

– гнучкість конфігурації;

– багатоадресний прийом із синхронізацією часу;

– загальносистемна узгодженість даних;

– виявлення помилок і сигналізація;

– автоматична повторна передача пошкоджених повідомлень, як тільки шина знову не працює;

– розрізнення тимчасових помилок та постійних відмов вузлів та автономне відключення дефектних вузлів;

– мультимайстер.

Вузол CAN має шаровату структуру, яка зображена на рисунку 1.2, та складається з наступних елементів:

– фізичний рівень, котрий визначає, як передаються сигнали;

– рівень передачі даних, що є ядром протоколу CAN. Він представляє повідомлення, отримані на об'єктному рівні, і приймає повідомлення, що передаються з об'єктного рівня. Рівень передачі даних відповідає за синхронізацію, кадрування повідомлень, арбітраж, підтвердження, виявлення та сигналізацію помилок;

– об'єктний рівень стосується фільтрації повідомлень, а також стану та обробки повідомлень.

Для підтримання узгодженості даних та прийняття контрольних рішень, блоки управління двигуном обмінюються данними за допомогою CAN-фреймів. В CAN існує чотири типи повідомлень:

– data frame;

– remote frame;

– error frame;

– overload frame.

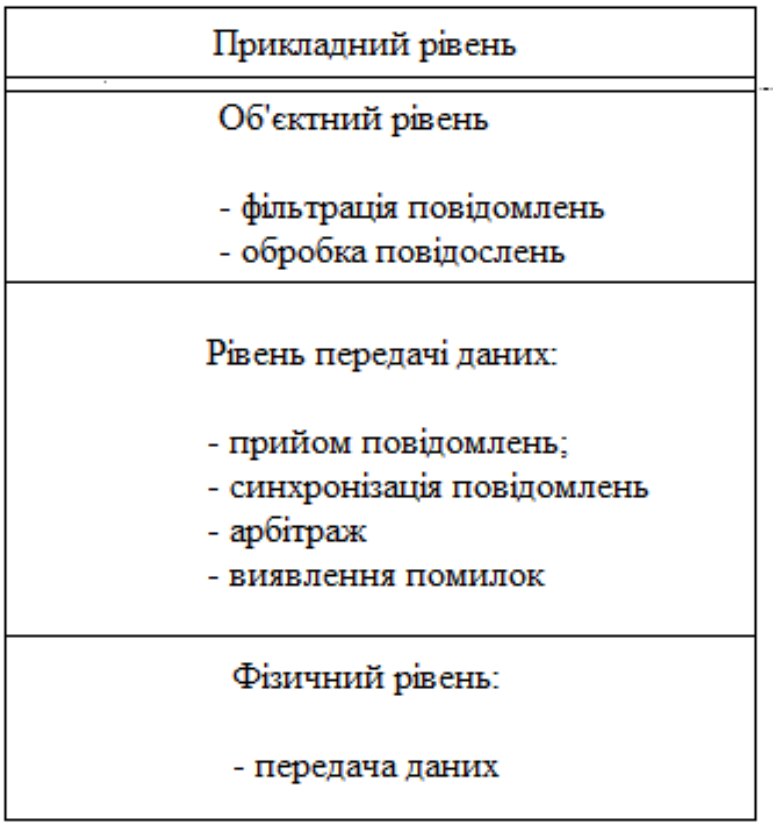

Рисунок 1.2 – Шарова структура вузла CAN

Саме за допомогою кадру даних (Data Frame) найчастіше за все передаються данні від передавача до приймачів. Цей фрейм, схематичне зображення якого наведено на рисунку 1.3, складається з семи різних бітових полів, а саме:

- початок кадру (SOF);
- поле арбітражу (arbitration field);
- поле контролю (control field);
- поле даних (data field);
- поле CRC (CRC field);
- поле підтвердження (ACK/Acknowledgement field);
- кінець кадру (end of frame).

Слід зазначити, що поле даних може мати нульову довжину.

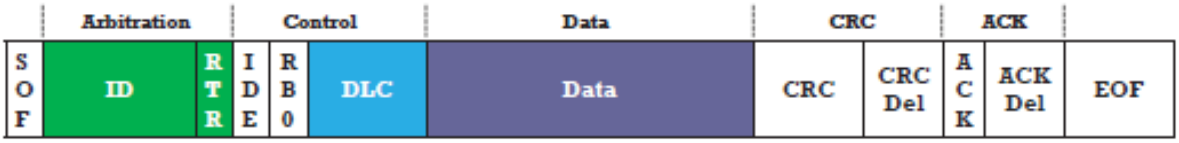

Рисунок 1.3 – Схематичне зображення кадру даних (Data Frame)

Поле початку кадру вказує на початок фрейму даних, що складається с одного «домінуючого» біта, логічний нуль.

Поле арбітражу визначає пріоритет повідомлень у випадку, коли два або більше разів одночасно намагаються передавати дані в мережу. Поле арбітражу в свою чергу складається з ідентифікатору та RTR (Remote Transmission Request) біту, який в фреймі даних завжди дорівнює логічному нулю (домінантний сигнал) [6].

Слід зазначити, що поле ідентифікатора, незважаючи на свою назву ніяк не ідентифікує ні вузол в мережі, ні вміст поля даних, а вказує на пріоритет повідомлення, як було сказано раніше.

Поле контролю складається з шести бітів. Воно включає DLC (Data Length Code), довжиною чотири біти, та два зарезервованих біти для подальшого розширення. Зарезервовані біти повинні бути надіслані «домінантними». Код довжини даних (DLC) вказує на кількість байтів у полі даних.

Поле даних включає дані, які передаються у кадрі даних. Воно може містити від нуля до восьми байт, кожен з яких містить вісім бітів, які передаються у порядку MSB спочатку.

CRC поле складається з CRC-послідовності та CRC розмежувача. Послідовність контролю кадру, що отримується з надлишкового циклічного коду, найкраще підходить для кадрів з числом бітом менше ніж 127 біт. При обчисленні CRC, поліном розділяється і визначається як

поліном, коефіцієнти якого задані послідовністю біт, що складається з полів: «початок кадру», «поле арбітражу», «керуючий поле», «поле даних» (якщо є) і, для 15 наймолодших коефіцієнтів, 0. Цей поліном наведено у формулі 1.1:

$$X^{15} + X^{14} + X^{10} + X^{8} + X^{7} + X^{4} + X^{3} + 1, \quad (1.1)$$

Залишок від цього полиномиального ділення і є послідовність CRC, що передається по шині. Послідовність CRC супроводжується роздільником CRC, який складається з одного одиничного біта [7].

Поле підтвердження (ACK Field) має довжину дві біти і включає слот підтвердження (Acknowledgement Slot) та розділяючий біт підтвердження (Acknowledgement Delimiter). Кожен CAN-контролер, який правильно прийняв повідомлення посилає біт підтвердження в мережу. Вузол, який послав повідомлення слухає цей біт, і в разі якщо підтвердження не прийшло, повторює передачу. У разі прийому слота підтвердження передавальний вузол може бути впевнений лише в тому, що хоча б один з вузлів в мережі правльно прийняв його повідомлення. Acknowledgement Delimiter – це других біт поля підтвердження, який повинен бути «рецесивним». Як результат, слот підтвердження оточений двома «рецесивними» бітами, а саме CRC Delimiter та Acknowledgement Delimiter.

Кінець кадру є послідовністю прапорів, що складається з семи «рецесивних» бітів, логічних одиниць, для розмежування кадрів даних.

CAN-фрейм віддаленого запиту, Remote Frame, використовується для ініціації одним з вузлів передачі даних в мережу даних іншим вузлом. Це дозволяє значно зменшити сумарний трафік мережі. Цей фрейм складається з шести різних бітових полів, а саме:

– початок кадру (SOF);

– поле арбітражу (arbitration field);

– поле контролю (control field);

– поле CRC (CRC field);

– поле підтвердження (ACK/Acknowledgement field);

– кінець кадру (end of frame).

На відміну від кадру даних, у фреймі віддаленого запиту немає поля даних, а RTR-біт є «рецесивним», тобто дорівнює логічному нулю. Слід зазначити, що саме цей біт вказує, чи переданий кадр є фреймом даних (RTR-біт «домінуючий») або фреймом віддаленого запиту (RTR-біт «рецесивний»). На практиці Remote Frame використовується нечасто.

Кадр помилки (Error Frame) – це повідомлення, яке порушує правила формування кадрів повідомлень CAN. Фрейм складається з двох полів: флаг помилки (Error Flag) та розмежувач помилки (Error Delimiter). Існує дві форми флагу помилки:

– флаг активної помилки (active error flag), що складається з шести послідовних «домінуючих» бітів;

– флаг пасивної помилки (passive error flag), який складається з шести «рецесивних» бітів. За винятком випадків, коли він перезаписаний «домінуючими» бітами з інших вузлів.

Вузол «активної помилки», що виявляє помилку, сигналізує про це шляхом передачі флаг активної помилки. Форма флагу помилки порушує бітстафінг, застосованй до всіх полів, починаючи з початку кадру до CRC Delimiter, або знищує фіксовану форму поля підтвердження або поля кінця кадру. Як наслідок, всі інші вузли виявляють стан помилки і зі свого боку починають передачу флагу помилки. Отже, послідовність «домінантних» бітів, яку насправді можна контролювати на шині, є результатом суперпозиції різних флагів помилок, переданих окремими вузлами. Загальна довжина цієї послідовності коливається як мінімум від шести до максимум дванадцяти бітів.

Вузол «пасивної помилки» намагається сигналізувати про помилку

шляхом передачі флагу пасивної помилки. Вузол очікує шість послідовних бітів однакової полярності, починаючи з початку флагу пасивної помилки. Прапор є завершеним, коли виявлено шість однакових бітів.

Ще одне поле, з якого складається кадр помилки, Error Delimiter, містить вісім «рецесивних» бітів. Після передачі флагу помилки кожен вузол відправляє «рецесивні» біти і контролює шину, поки не виявить «рецесивний» біт. Потім він починає передавати ще сім «рецесивних» бітів.

Кадр перевантаження (Overload Frame) – один з чотирьох типів повідомлень CAN, що повторює структуру та логіку роботи кадру помилки. Найважливіша відмінність полягає в тому, що він використовується перевантаженим вузлом, що в даний момент не має можливості обробити повідомлення, тому просить допомоги у Overload-фрейма.

Існує два типи умов перевантаження, обидва з яких призводять до передачі флага перевантаження:

– внутрішні умови приймача, що вимагає затримки наступного фрейму даних або фрейму віддаленого запиту;

– виявлення «домінуючого» біта під час інтермісії.

Кадр перевантаження складається з двох бітових полів: флага перевантаження і розмежувача перевантаження. Флаг перевантаження (Overload Flag) містить шість «домінуючих» бітів. Загальна форма відповідає формі флагу активної помилки.

Розмежувач перевантаження (Overload Delimiter) складається с восьми «рецесивних» бітів, загальна форма якого відповідає формі розмежувача помилки. На практиці в даний час Overload Frame майже не використовується.

Як було зазначено раніше, повідомлення CAN характеризується міткою часу, ідентифікатором (ID) та, як правило, 8-байтовим полем корисної інформації (payload). Ідентифікатор представляє тип поточного

повідомлення, а поле даних використовується для передачі поточних значень сигналів транспортного засобу. Кожен ідентифікатор пов'язаний з набором сигналів, і поле payload повідомлень, що несуть цей ідентифікатор, надає їх поточні значення.

Загалом повідомлення з різними ідентифікаторами містять різний набір сигналів. Кодування значень сигналу варіюється від одиничних бітів до декількох байт поля payload. Матриця CAN надає інформацію для кожного ідентифікатора, вказуючи, які біти поля payload кодують який сигнал.

На рисунку 1.4 зображено схематичне представлення даних шини CAN після попередньої обробки байтів сигналів. Зверніть увагу, що при кожній мітці часу передаються лише значення сигналу одного ідентифікатора. Різні ідентифікатори можуть містити різну кількість сигналів, наприклад ID A складається з шести сигналів, тоді як ID D має один сигнал. Мітка часу подана в мілісекундах, час дискретизований.

CAN Bus Data

Time Stamp	ID	Signals of *A*						Signals of *B*			Signals of *C*			Signals of *D*
1.045	B	-	-	-	-	-	-	54.71	0	7.24	-	-	-	-
3.102	D	-	-	-	-	-	-	-	-	-	-	-	-	31.47
4.978	A	12	44.15	38.02	2	0	1	-	-	-	-	-	-	-
7.014	C	-	-	-	-	-	-	-	-	-	17.79	7	2	-
8.993	B	-	-	-	-	-	-	55.02	1	7.21	-	-	-	-
9.750	A	13	44.01	39.67	1	0	2	-	-	-	-	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Рисунок 1.4 – Схематичне представлення даних шини CAN після попередньої обробки байтів сигналів

1.2 Існуючі методи для виявлення аномалій в сигналах з CAN шини автомобіля

Мережа контролерів (CAN) у автомобілях має вирішальне значення для їх безпеки та використання, і зараз вона розглядається як об’єкт, вразливий до кібератак. Отже, бажано контролювати трафік CAN-шини для виявлення вторгнень [8].

Виявлення вторгнень базується на припущенні, що їх поведінка відрізнятиметься від звичайної, і декілька аналітичних методів було запропоновано як потенційних кандидатів. Зазвичай вони класифікуються як методи, засновані на сигнатурах та аномаліях.

Підходи, засновані на аномаліях, додатково класифікуються за типом використовуваних методів, таких як статистичні, засновані на знаннях, кластеризація (щільність), вектор опорних векторів та моделі Маркова. Останні три категорії – це приклади використання машинного навчання.

Виявлення вторгнень на основі сигнатур передбачає, що шаблони атаки відомі заздалегідь. Дискримінаційні ознаки сигнатур зберігаються в базі даних, за допомогою якої здійснюється моніторинг наступних пакетів. Пакети з даними, які відповідають сигнатурам з бази даних, позначуються як порушення, вторгнення.

Головною перевагою цих методів є те, що вони можуть надійно виявляти вторгнення та генерувати дуже низький показник хибно позитивних даних, коли відома підпис атаки, і вони можуть визначати, який саме тип атаки зазнає система. Однак, враховуючи все ще виникаючий режим автомобільних кібератак та неоднаковий життєвий цикл транспортних засобів, методи на основі сигнатур рідко використовуються для виявлення аномалій CAN. Також додатковим ускладненням є те, що база даних потребує постійного оновлення.

Системи виявлення вторгнень на основі аномалій виявляють поведінку, яка відхиляється від звичайної. Метод полягає в тому, що

нормальну поведінку можна сформулювати та встановити порогові значення, щоб забезпечити точну ідентифікацію відхилень. Оскільки виявлення аномалій не базується на встановлених сигнатурах, ці методи мають потенціал узагальнення, що робить можливим виявлення раніше невідомих атак, крім того, вони налаштовуються алгоритмом навчання, що ускладнює можливість зловмисникам обійти систему.

Однак такі системи вимагають навчання та збору даних, що є достатньо репрезентативними для звичайного руху, щоб можна було визначити точні та надійні порогові значення для виявлення аномалії. Слід зазначити, що отримання таких даних для CAN є проблематичним. Крім того, оскільки методи виявлення аномалій шукають аномальні події, а не конкретну атаку, частота помилкових спрацьовувань та показник хибно негативних даних може бути високим. Незважаючи на це, непередбачуваність сценаріїв атак, разом із труднощами у визначенні надійних сигнатур, призвела до багатьох досліджень із застосування підходу на основі виявлення аномалій.

Загалом, ці підходи можна класифікувати на підходи, що базуються на статистиці, підходи, засновані на знаннях, та машинне навчання, які можна додатково класифікувати відповідно до застосованої моделі машинного навчання.

1.2.1 Статистичні методи виявлення аномалій

Статистичні методи порівнюють статистичний профіль системи, що спостерігається в даний час, із попередньо визначеним профілем.

Наприклад, система може оцінити відхилення у властивостях, таких як середнє, медіана та мода. У випадку даних часових рядів, таких як трафік шини CAN, статистичні методи можуть використовувати ковзне вікно, яке зменшує розмір набору даних, який повинен буди проаналізований одразу. Часові ряди можуть бути однофакторними або

багатовимірними. Одновимірна техніка аналізує поля CAN ID незалежно один від одного. Багатовимірні методи можуть більш придатні для виявлення вторгнень, що впливають на дані з датчика CAN [9].

Одним із статистичних методів для виявлення вторгнень, є метод, заснований на аналізі частот отримання повідомлень. Головна ідея полягає в тому, що існує унікальний інтервал часу кожного ідентифікатора CAN, оскільки кожен ECU, підключений до шини CAN, регулярно надсилає повідомлення. Запропонована система виявляє атаки вторгнення за допомогою наступної процедури:

– коли на шині CAN з'являється нове повідомлення, система виявлення вторгнень перевіряє CAN ID і обчислює інтервал часу від моменту надходження останнього повідомлення;

– якщо часовий інтервал нового повідомлення коротший, ніж зазвичай, тоді система виявлення вторгнень вважає повідомлення зловмисним. Зазвичай повідомлення розглядається як вторгнення, якщо інтервал часу нижче половини норми;

– якщо інтервал часу між останніми повідомленнями поспіль менше 0,2 мілісекунд, тоді вони розглядаються, як DoS-атака.

Перевагою статистичних методів є те, вони не вимагають попередніх знань про звичайний CAN-трафік і можуть ідентифікувати вторгнення, що розвиваються протягом тривалого періоду. Недоліками є можлива складність визначення порогових значень, а також вірогідність того, що зловмисник знає про цю систему та відповідно моделює вторгення [10].

1.2.2 Методи, що базуються на знаннях

Методи, що базуються на знаннях, виводять набір правил з навчальних даних. Подальші події класифікуються відповідно до цих правил.

Ці методи можуть забезпечити високий рівень виявлення вторгнень та низький показник хибно позитивних результатів, якщо база знань є достатньо повною. Однак вони покладаються на знання, які важко отримати у випадку CAN-трафіку.

Так, було запропоновано побудувати матрицю переходів, яка містила б неаномальні парні послідовності ідентифікаторів CAN (CAN IDs). Послідовні повідомлення, що транслюються на CAN, потім перевірятимуться за допомогою матриці; і повідомлення, що транслюються не з послідовності, яка міститься в матриці, будуть позначатися як аномалії. Такому підходу потрібно мало пам’яті, а також він має низькі обчислювальні витрати. Цей підхід не дає помилкові спрацьовування, і надійно виявляє змодельовані атаки. Однак метод виявився менш успішним при виявленні атак відтворення, коли відомі послідовності ретранслюються.

Хоча такий матричний підхід, заснований на наступному ідентифікаторі, може точно виявляти невідповідності, коли за ідентифікатором слідує ідентифікатор з дуже маленької підмножини, але менш вдало виявляє, коли за ідентифікатором може слідувати ідентифікатор із великої підмножини. Аналіз даних CAN з популярного сімейного автомобіля показує, що це стосується багатьох ідентифікаторів. На рисунку 1.5 зображений розподіл кількості різних ідентифікаторів пакетів, за якими спостерігається кожен CAN ID.

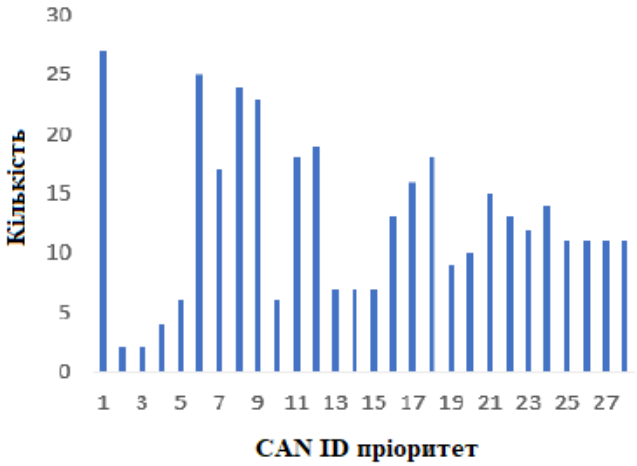

Рисунок 1.5 – Кількість різних ідентифікаторів пакетів, за якими спостерігається кожен CAN ID

1.2.3 Методи, засновані на машинному навчанні, кластеризація

Методи машинного навчання, як правило, намагаються вивчити закономірності в даних, а режим навчання може бути з вчителем і без. Навчання з вчителем використовує марковані дані, тоді як навчання без учителя використовує дані без маркування. Потім алгоритм машинного навчання спробує отримати класи на основі певної подібності. Деякі алгоритми машинного навчання пропонують варіанти методів, які прагнуть визначити викиди після навчання, використовуючи дані з відомих нормальних, звичайних випадків. Під час такого однокласного навчання класифікації аналізуються дані, отримані при нормальній роботі, для визначення граничної межі, яка охоплює звичайні екземпляри. Подальші випадки класифікуються як звичайні, якщо вони знаходяться у межах, або як аномалії, якщо вони перебувають за їх межами. Класифікація одного класу була запропонована як ймовірний механізм виявлення вторгнень в CAN шину.

Методи машинного навчання припускають, що екземпляри певного класу групуються у кластери навколо архетипу. Потім кожен новий

екземпляр можна класифікувати за віддаленістю від цього архетипу. Недавні підходи до виявлення сторонніх процесів використовували багатовимірні розрахунки відстані, що дозволяє проводити багатовимірний аналіз, часто поєднаний із розрахунками щільності.

Кластерні підходи можна застосовувати як приклад навчання без вчителя. При навчанні на даних, що включають кілька класів, припускають, що аномалії утворюватимуть менший з кластерів. Для навчання без вчителя на нормальних даних, міра щільності та розрахунок відстані можуть визначити границю нормального класу [11].

Використовувався алгоритм класифікації чотирьох найближчих сусідів, щоб розрізняти нормальні пакети CAN та модельовані пакети CAN, які містили виключно інформацію з полів даних. Класифікатор зміг класифікувати вставлені пакети, що містять дані передач та обертів на хвилину. Головним недоліком методу є прокляття розмірності.

1.2.4 Прихована модель Маркова

Приховані моделі Маркова намагаються передбачити подальший стан системи, враховуючи поточний стан. Хоча їх не можна застосувати до деяких процесів; наприклад, там, де майбутній стан не залежить від поточного стану, наприклад, прогнозування результатів у серії підкидань монет, вони вважаються потенційно можливими у виявленні аномалії CAN-трафіку.

Для реалізації детектора аномалій на основі моделі Маркова, використовуються не абсолютні дані, отримані з пакетів CAN, градієнти. З цих даних будується вектор часових рядів, що містить дані швидкості та обертів на хвилину. Прихована модель Маркова використовується для генерування ймовірностей переходу, тобто ймовірності переходу в наступний стан, та ймовірностей викидів, ймовірності спостережуваного виходу для даного стану. Під час виявлення аномалії, апостеріорна

ймовірність наступного спостережуваного зчитування визначалася за допомогою ковзного вікна безпосередньо попередніх спостережень. Модель успішно виявляє згенеровані аномалії, але головним недоліком методу є те, що результати залежать від припущень про поведінку самої системи [12].

1.2.5 Метод опорних векторів

Метод опорних векторів (SVM) прагне розділити класи, використовуючи гіперплощину, з максимальним зазором в цьому просторі. Екземпляри на краю кожного класу, стають векторами підтримки.

SVM має можливість вчитися на невеликих вибірках; може справлятися з високо мірними даними та може навчатися без вчителя. Варіант SVM, особливо важливий для виявлення аномалій, – це однокласні SVM (OCSVM), які намагаються створити відповідну межу при навчанні, використовуючи лише екземпляри одного класу [13].

На рисунку 1.6 зображено результат використання однокласового методу опорних векторів. У колі зображені дані, які є нормальними, а за межею – аномальні.

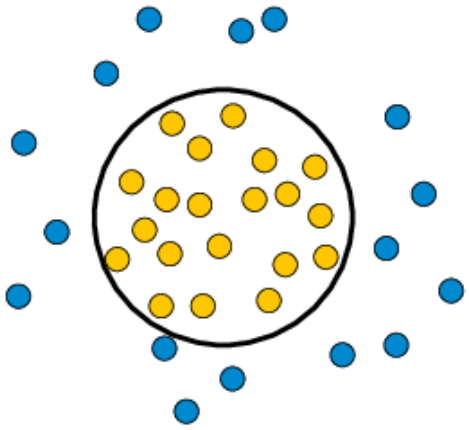

Рисунок 1.6 – результат використання однокласового методу опорних векторів

Переважна кількість CAN ID пересилаються з фіксованою частотою. Саме для виявлення вторгнень аналізуються частоти та зміни вмісту даних в середньому, використовуючи відстань Хеммінга. Ці метрики порівнюються з попередніми задля того, що визначити аномальні.

Однокласовий метод опорних векторів здатний виявляти атаки тривалістю 0,5 секунди або більше, і надійно виявляє короткі атаки тривалістю 0,2 секунди. Головним недоліком цього методу є те, що для навчання він потребує багато ресурсів, а також має високий хибно позитивний показник.

У таблиці 1.1 наведено порівняльні характеристики, а саме необхідну кількість пам’яті для реалізації методу, переваги та недоліки вище зазначених методів для виявлення вторгнень у CAN шину.

Таблиця 1.1 – Порівняння методів для виявлення вторгнень у CAN шину

Метод	Рівень необхідної кількості пам'яті	Переваги	Недоліки
1	2	3	4
На основі сигнатур	Середня	Низький коефіцієнт хибно позитивних даних для відомих атак	Сигнатуру атаки потрібно знати заздалегідь. Легко обійти модифікованими атаками
Статистичні	низька	Не вимагає попередніх знань про звичайний CAN-трафік,	Складність визначення порогових значень

Продовження таблиці 1.1

1	2	3	4
		може ідентифікувати вторгення, що розвиваються протягом тривалого періоду	
Засновані на знаннях	Середня	Низька кількість помилкових спрацьовувань.	Розробка та підтримка бази знань
Кластеризація	Висока	Простота і зрозумілість	Прокляття розмірності
Прихована модель Маркова	низька	Підходить для оцінки перехідних послідовностей	Результати залежать від припущень про поведінку системи
Метод опорних векторів	низька	Може мати справу з великою кількістю даних	Схильний до високого хибно позитивного показника

1.3 Постановка завдання дослідження

В ході аналізу предментної області було виявлено, що шина CAN, локальна мережа контролерів, розроблена компанією Bosch, відіграє ключову роль у автомобілі, а саме дозволяє йому зчитувати та відправляти команди, користуватися такими системами управління, як подушки безпеки, гальмівні системи, регулювання швидкості руху,круїз контроль та

інше. Вищеописана мережа є вразливою до вторгнень з боку зловмисників, що у майбутньому може призвести до трагічних наслідків.

Вважається, що зловмисник вже отримав доступ до шини CAN, наприклад, викравши один із підключених блоків управління двигуном. Також припускається, що зловмисник намагається впливати на поведінку транспортного засобу, маніпулюючи повідомленнями.

Головна мета – виявити сигнали, що відхиляються від їх «нормальної» поведінки, або сигнали, що мають розриви у фізичних взаємозв’язках. Останні, як правило, складні, потенційно невідомі і їх важко отримати за допомогою систем виявлення вторгнень, заснованих на правилах, для CAN [14].

Також встановлено, що методи для виконання виявлення вторгнень у процес обміну повідомлень CAN шини мають свої недоліки, що призводять до того, що вони не використовуються на сьогоднішній день.

Для досягнення мети дослідження необхідно виконати наступні задачі:

– вивчити існуючі методів обчислювального інтелекту для вирішення завдання виявлення вторгнень у процес обміну повідомлень CAN, які використовуються на сьогоднішній день;

– розробити алгоритм для автоматичного виявлення вторгнень у процес обміну повідомлень CAN шини, який базується на використанні глибинних нейронних мереж, а саме мереж довгої короткострокової пам’яті для вилучення необхідних ознак з вхідного сигналу, а також автокодувальника для отримання реконструкції сигналу для подальшого виявлення аномальних сигналів;

– розробити модуль глибинного навчання для автоматичного виявлення вторгнень у процес обміну повідомлень CAN шини;

– провести експериментальну валідацію і тестування отриманого модуля за допомогою відповідних метрик визначення точності роботи системи.

2 АЛГОРИТМ ГЛИБИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ АНОМАЛІЙ В СИГНАЛАХ CAN ШИНИ АВТОМОБІЛЯ

У попередньому розділі було розглянуто методи, які використовувалися для виявлення аномалій в сигналах CAN шини автомобіля, кожен з яких має свої недоліки на переваги.

Пропонується наступна система для автоматичного виявлення вторгнень у CAN шину. Основною ідеєю є те, що для аналізу складної структури даних CAN, запроваджується кілька незалежних рекурентних нейронних мереж на вхідній стороні. Це дозволяє системі вивчати часові залежності сигналів. Детальніше, для кожного CAN ID вводиться окрема підмережа довгої короткочасної пам’яті, яка на вході отримує значення сигналу, пов’язаного саме з цим ідентифікатором та зберігає поточний стан оброблених даних. Потім стан усього трафіку CAN шини автомобіля представляється у вигляді спільного прихованого вектору (Joint Latent Vector), який представляє собою об’єднання поточних станів всіх вхідних підмереж. Цей вектор не містить жодної інформації про те, який CAN ID оброблявся останнім. На наступному кроці спільний прихований вектор подається на вхід підмережі, яка складається с послідовних лінійних шарів, що представляє собою автокодувальник.

Тобто на кожній позначці часу завданням цієї підмережі є реконструкція значень сигналу вхідного повідомлення виключно на основі поточного значення спільного прихованого вектора. Потім відхилення між значеннями сигналу справжнього вхідного повідомлення та його реконструкцією використовується як міра нормальності повідомлення. Оскільки мережа навчається виключно на звичайних даних, очікується, що ця помилка реконструкції невелика для звичайних даних і велика для аномальних.

На рисунку 2.1 зображено схематично запропоновану вище архітектуру для виявлення аномалій в сигналах CAN шини автомобіля.

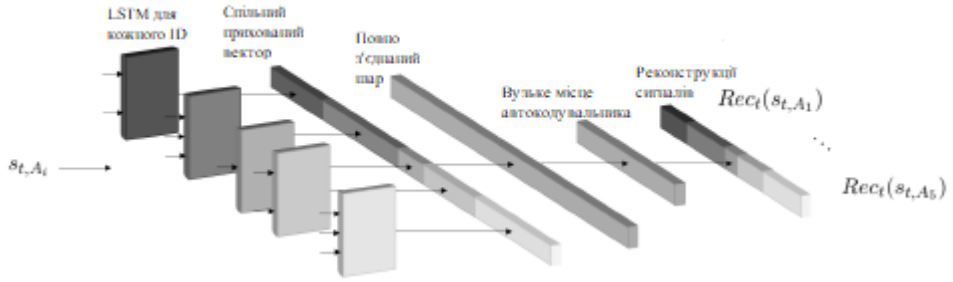

Рисунок 2.1 – Архітектура для виявлення аномалій в сигналах CAN шини автомобіля

Кожен ідентифікатор (CAN ID), як було зазначено раніше, має власну вхідну підмережу довгої короткочасної пам’яті. Коли сигнал s_{t,A_t}, який відноситься до одного з CAN ID, надходить до відповідної вхідної підмережі, тільки відповідна частина спільного прихованого вектору оновлюється. Весь прихований вектор використовується для реконструкції сигналів усіх ідентифікаторів.

2.1 Архітектура запропонованої нейромережевої системи

Для того, щоб ефективно описати архітектуру мережі, спочатку необхідно встановити наступні позначення. Нехай $\boldsymbol{A} = \{A_1, \dots, A_K\}$ - порядкована множина всіх ідентифікаторів $K \in \mathbb{N}$. Для кожного CAN ID $A \in \boldsymbol{A}$ ми беремо n_A декодованих сигналів з CAN шини. Позначимо загальну кількість сигналів як N. Щоразу, коли на відмітці часу t спостерігається CAN ID $A \in \boldsymbol{A}$, вектор, що містить відповідні значення сигналу, позначається як $s_{t,A} \in R^{n_A}$. Слід зазначити, що саме цим дискретизується час.

Як було зазначено раніше, запропонована архітектура складається з

вхідної підмережі довгої короткочасної пам’яті для кожного CAN ID $A \in \boldsymbol{A}$. Кожна i-та підмережа LSTM, пов’язана з ідентифікатором A_i, має вхідний вектор розмірністю n_{A_i} та прихований шар розміром $n_{A_i} \cdot h_{scale}$. Позначення h_{scale} представляє обчислювану потужність архітектури. Кожного разу, коли нове повідомлення відповідного ідентифікатора $A \in \boldsymbol{A}$ подається на вхід відповідної підмережі довгої короткочасної пам’яті, її вектор вихідних значень розміром $n_{A_i} \cdot h_{scale}$ використовується для оновлення відповідного фрагменту спільного прихованого вектору. Отже, спільний прихований вектор (Joint Latent Vector) має довжину $N \cdot h_{scale}$. Він відображає та зберігає поточний стан трафіку CAN шини. Спільний прихований вектор супроводжується набором повно з’єднаних шарів, з яких передостанній має строго менше нейронів, ніж вихідний шар, який має N нейронів. Кожен з повно з’єднаних шарів має функцію активації ELU. Головне завдання вихідного шару – реконструювати всі потенційні поточні сигнали від усіх ідентифікаторів. У таблиці 2.1 наведено специфікацію шарів нейромережевої системи.

Таблиця 2.1 – Специфікація шарів запропонованої нейромережевої системи

Тип шару	Кількість вихідних нейронів
LSTM для кожного ID	Кількість сигналів відповідного ID $\times h_{scale}$
Спільний прихований вектор	$N \times h_{scale}$
Повно з’єднаний шар (ELU)	$N \times h_{scale}/2$
Повно з’єднаний шар (ELU)	$N - 1$
Повно з’єднаний шар (ELU)	N

Під час навчання на кожній відмітці часу $t \in \mathbb{N}$ повідомлення подається на вхід відповідної вхідної підмережі LSTM. Потім вихідне значення використовується для оновлення спільного прихованого вектору з метою реконструкції сигналу для всіх $A \in \boldsymbol{A}$. Формально, реконструкція R_t сигналу для відмітки часу t визначається за допомогою наступної формули:

$$R_t(s_{t,A_i}) = (Rec_t(s_{t,A_1}), \ldots, Rec_t(s_{t,A_K})), \tag{2.1}$$

де $Rec_t\left(s_{t,A_j}\right)$ позначає реконструкцію повідомлення ідентифікатора A_j.

Далі порівнюються справжні значення сигналу ID A_i з їх реконструкціями $Rec_t(s_{t,A_i})$. Використовується функцію квадратичної помилки, задану у формулі (2.2):

$$loss(s_{t,A_i}) = \left\| Rec_t(s_{t,A_i}) - s_{t,A_i} \right\|_{\ell_2}^2. \tag{2.2}$$

Детальніше, при обчисленні зворотного розповсюдження посилки щодо цієї функції втрат повинні бути обчислені лише градієнти ваг підмережі LSTM для відповідного ID A_i та ваги, які з'єднують спільний латентний вектор та вихідним значенням реконструкції сигналу $Rec_t(s_{t,A_i})$.

Оскільки часові залежності зберігаються для кожного ідентифікатора окремо у відповідній підмережі LSTM, навчальний процес має ту перевагу, що модель у цілому не чутлива до точного порядку послідовних ідентифікаторів повідомлень. Насправді, в реальних даних CAN існує певна мінливість в порядку ідентифікаторів, навіть у межах одного набору даних.

2.2 Загальні поняття про нейронну мережу довгої короткочасної пам’яті

Нейронна мережа довгої короткочасної пам’яті, LSTM, являє собою особливий вид рекурентної нейронної мережі, RNN, здатний вивчати довгострокові залежності. LSTM використовується у багатьох додатках, таких як розпізнавання мови, рукописного тексту, машинний переклад, генерування опису до зображення, синтаксичний розбір, робота з часовими рядами, обробка природної мови тощо. На відміну від повністю зв’язаних нейронних мереж, вони мають зворотні зв’язки. Спочатку вони були розроблені для вирішення проблеми зникаючого градієнта стандартних RNN [16] і широко використовуються в наш час.

На рисунку 2.2 зображено структуру вузлу LSTM.

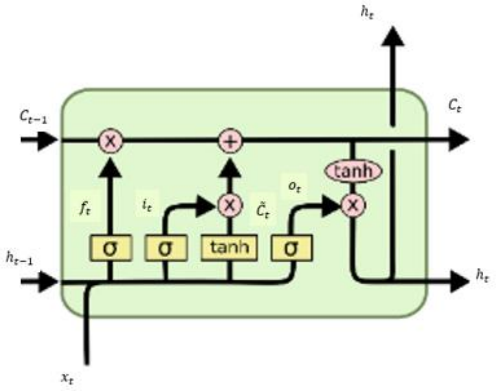

Рисунок 2.2 – Структура вузлу LSTM

Ключовим поняттям у LSTM є вектор стану, до якого нейронна мережа може додавати або видаляти інформацію, за допомогою регулюючих структур, що називаються вентилями. Всього LSTM має три вентилі для захисту і контролю вектору стану.

Першим кроком у LSTM є вирішення того, яку існуючу інформацію буде збережено. Це рішення приймається за допомогою сигмоїдального шару, який називається забувальним вентилем. Він приймає на вхід вихід

попереднього LSTM-шару h_{t-1}і вхідне значення x_t, на виході отримується значення між 0 та 1 для кожного елементу вектору стану C_{t-1}. Отримане значення, яке дорівнює нуло вказує, що інформацію необхідно «забути», одиниці – «запам'ятати». Обчислюється воно наступним чином:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_t). \tag{2.3}$$

Наступний крок, який складається з двох частин, полягає у визначенні нової інформації, яка буде зберігатися у векторі станів. Спочатку за допомогою сигмоїдального шару, який називається входовим вентилем, вирішується, які значення потрібно оновити. Далі, за допомогою функції гіперболічного тангенсу створюється вектор нових значень-кандидатів, $\tilde{C}_t$, які буде додано до вектору стану:

$$\mathrm{i_t} = \sigma(\mathrm{W_i} \cdot [\mathrm{h_{t-1}}, \mathrm{x_t}] + \mathrm{b_t}), \tag{2.4}$$

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C). \tag{2.5}$$

Далі попередній стан C_{t-1} оновлюється до нового C_t. Для цього перемножується старий стан з f_t, забуваючи те, що було вирішено забути. Потім додаються нові значення $\tilde{C}_t$, які перемножуються з їх кількістю i_t:

$$\mathrm{C_t} = \mathrm{f_t} \cdot \mathrm{C_{t-1}} + \mathrm{i_t} \cdot \tilde{\mathrm{C}}_\mathrm{t}. \tag{2.6}$$

Після цього обчислюється вихідне значення на основі фільтрованої версії вектору стану. Спершу, за допомогою сигмоїдального шару визначається, які частини вектору стану будуть виходом. Потім функція

гіперболічного тангенсу приймає вектор стану C_t і перемножується зі значенням виходового вентиля:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{2.7}$$

$$h_t = o_t \cdot \tanh(C_t). \tag{2.8}$$

2.3 Загальні поняття про автокодувальник

Клас нейромережевих структур, який відомий як автокодувальник, має на меті вивчити структуру навчального набору даних, наприклад нормальна поведінка технічної системи у режимі роботи без учителя [17]. Це означає, що в процесі навчання не потрібні марковані дані. Навчений автокодувальник може бути використаний для ідентифікації точок даних, які відхиляються від нормальної поведінки, і тому можуть функціонувати як система виявлення аномалій. Основна ідея, що стоїть за автокодувальниками, полягає в тому, що дані з великою розмірістю, що походять з якоїсь базової системи, часто містять кореляційні зв’язки. Отже, значуще векторне представлення в меншому за вимірністю підпросторі зазвичай існує.

Автокодувальник складається з двох блоків нейронної мережі, кодера та декодера. І те, і інше може бути реалізоване набором послідовних повністю з'єднаних шарів. Завдання кодера - відобразити вхідні дані, які є великими за розмірністю у векторне представлення, простір меншої розмірності, який називається вузьким місцем автокодувальника. Завдання декодера - реконструювати вхідні дані з їх подання у вигляді векторного представлення. Тоді як відхилення між реальними вхідними даними та їх реконструкцією має бути невеликим за

звичайних даних, очікуються великі відхилення для аномальних даних, яких система не бачила в процесі навчання [18].

Формально кодер та декодер може бути визначено як переходи ϕ та ψ, а саме:

$$\phi: \chi \longrightarrow \mathcal{F}, \tag{2.9}$$

$$\psi: \mathcal{F} \longrightarrow \chi, \tag{2.10}$$

$$\phi, \psi = \arg\min_{\phi,\psi} \|X - (\phi \circ \psi)X\|^2. \tag{2.11}$$

На рисунку 2.3 зображено структуру найпростішого автокодувальника.

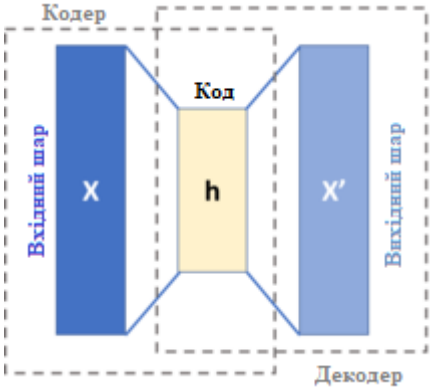

Рисунок 2.3 – Структура найпростішого автокодувальника

В найпростішому випадку, на стадії кодування автокодувальник приймає на вході дані $x \in \mathbb{R}^d = \chi$ та відображає їх на $h \in \mathbb{R}^p = \mathcal{F}$, в свою чергу відображення визначається як:

$$h = \sigma(Wx + b), \tag{2.12}$$

де відображення h зазвичай називають кодом або прихованим представленням, latent representation. В свою чергу σ – функція активації, сигмоїдальна або ReLU. W – матриця ваги, а b – вектор зміщення. Останні два елемента зазвичай ініціалізуються випадковим чином, а потім ітеративно оновлюються під час тренування за допомогою метода зворотного поширення помилки [19].

Після цього на етапі декодування автокодувальник переводить h у відображення x' такого ж розміру, як і x:

$$x' = \sigma'(W'h + b'), \tag{2.13}$$

де значення σ', W' та b' для декодера можуть бути не пов'язані з відповідними σ, W та b для кодера.

Автокодувальники навчаються мінімізувати помилки при реконструкції даних, які ще називають функцією втрат:

$$\mathcal{L}(x, x') = \|x - x'\|^2 = \|x - \sigma'(W'(\sigma(Wx + b)) + b')\|^2, \tag{2.14}$$

де x зазвичай усереднюється за деяким набором навчальних матеріалів.

Як вже згадувалося раніше, навчання автокодувальника виконується шляхом зворотного поширення помилки, як і звичайна нейронна мережа прямого поширення.

2.4 Загальні поняття про функцію активації ELU

ELU – це нелінійна функція активації, що використовується в глибоких нейронних мережах. Частіше за все мета її використання – пришвидшення навчання.

Головною перевагою експоненціально-лінійної одиниці є полегшення проблеми зникаючого градієнта. Область допустимих значень

включає негативні значення, що дозволяє змістити середню активацію одиниці ближче до нуля, тим самим зменшуючи обчислювальну складність [20]. Вона визначається як вказано у формулі 2.15:

$$ELU(x) = \begin{cases} x, & \text{якщо } x > 0 \\ a(e^x - 1), & \text{якщо } x \leq 0 \end{cases}, \tag{2.15}$$

де $a \geq 0$ – константа, яка контролює точку насичення для негативних виходів мережі, зазвичай дорівнює одиниці.

В свою чергу градієнт рівняння ELU визначається як вказано у формулі нижче:

$$ELU'(x) = \begin{cases} 1, & \text{якщо } x > 0 \\ ELU(x) + a, & \text{якщо } x \leq 0 \end{cases}. \tag{2.16}$$

На рисунку 2.4 зображено графік функції активації ELU.

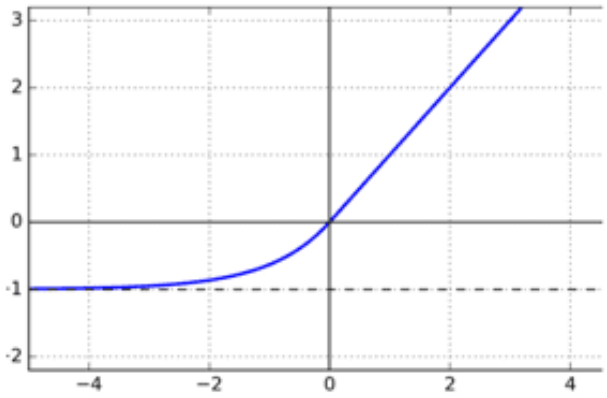

Рисунок 2.4 – Графік функції активації ELU

Однак, критичним обмеженням функції активації ELU є те, що вона не центрує значення на нулі [21].

2.5 Оцінка аномальності даних

Квадратичну помилку між сигналом та його реконструкцією, як визначено у формулі 2.2, можна використовувати для прогнозування того, чи є сигнал на відмітці часу $t \in \mathbb{N}$ аномальним. Цей прогноз можна зробити шляхом перевірки, чи перевищує значення помилки деякий фіксований поріг.

У зв'язку з тим, що ми розглядаємо проблему навчання без вчителя, цей поріг необхідно обирати виключно на основі нормальних даних. Для кожного сигналу він обчислюється окремо і визначається як 99.99% перцентиль усіх відповідних квадратичних помилок на валідаційному наборі даних. На етапі оцінки для кожного сигналу зберігається індивідуальний індикатор аномалії. Кожен раз, коли модель обробляє CAN ID, індикатор аномалії відповідних сигналів оновлюється, і встановлюється значення одиниці, якщо помилка реконструкції перевищує відповідний 99.99% перцентиль, а значення нуль отримує індикатор в іншому випадку. Загальна оцінка аномалії на відмітці часу t встановлюється рівною одиниці лише тоді, коли принаймні один із збережених показників аномалії сигналу дорівнює одному, а в іншому випадку встановлюється нуль.

Узагальнене визначення P-го центилю звучить наступним чином: P-ий перцентиль має значення, нижче якого виявлено P відсотків спостережень, і вище якого знайдено (100 - P) відсотків [22].

На рисунку 2.5 зображено 99.99% перцентиль заданого розподілу даних.

Слід зазначити, що цей показник аномалії чутливий до занадто великої кількості сигналів, оскільки він страждає від такого ефекту, як багаторазова проблема тестування. Тобто, якщо число сигналів стає значним, ймовірність того, що нормальна точка даних ідентифікується як

така (істинно негативна), показник true negative rate, відповідно зменшується.

Рисунок 2.5 – 99.99% перцентиль заданого розподілу даних

У випадку зі системою для виявлення вторгнень у CAN шину автомобіля, істинно негативний показник повинен бути майже стовідсотковий, оскільки вартість кожного помилкового спрацювання відносно висока. Тобто, залежно від механізму реагування на аномалію, помилкові спрацювання можуть мати такі наслідки, як рекомендація водієві зупинитися, або відвідати механіка для діагностики автомобіля, але на справді ніякої проблемі існувати не буде.

3 ІМІТАЦІЙНЕ МОДЕЛЮВАННЯ

3.1 Опис набору даних, необхідних для моделювання процесу виявлення аномалій у CAN шині автомобіля

Для виконання імітаційного моделювання модуля автоматичного виявлення аномалій в сигналах CAN шини автомобіля було використано набір даних SynCAN. Він наданий німецькою групою. компаній, Robert Bosch GmbH, провідним світовим постачальником технологій і послуг в області автомобільних і промислових технологій, споживчих товарів, будівельних і пакувальних технологій.

Набір даних SynCAN складається з 10 різних CAN ID повідомлень, кожен з різною кількістю сигналів та різними часовими частотами. Загальна кількість сигналів дорівнює 20. Дані створюються таким чином, що вони схожі на реальний трафік CAN шини в автомобілі

Дані містять фізичні значення, лічильники та сигнали, які залежать від одного або декількох інших сигналів. Загальна тривалість набору навчальних даних дорівнює приблизно 16,5 годин, а набір тестових даних – приблизно 7,5 годин роботи CAN трафіку.

Набір даних складається з набору навчальних даних та шести тестових наборів даних, які всі містять такі стовпці:

– «label»;
– «id»;
– «time»;
– «signal1_of_ID»;
– «signal2_of_ID»;
– «signal3_of_ID»;
– «signal4_of_ID».

Стовпець «label» вказує, чи слід вважати рядок даних нормальним, у такому випадку значення дорівнює нулю, або як вторгнення, та значення

дорівнює одиниці. Оскільки ми припускаємо, що система виявлення аномалій тренується в режимі без вчителя, цей стовпець на наборі навчальних даних завжди встановлюється рівною нулю, що означає відсутність вторгнень.

Стовпець «id» містить вказівники ідентифікаторів, CAN ID, наприклад, «id1», …, «id10».

У свою чергу стовпець «time» міститься позначка часу поточного повідомлення, що представлена в мілісекундах.

Стовпці «signal1_of_ID», ..., «signal4_of_ID» містять фактичні значення сигналу. Необхідно зазначити, що різні ідентифікатори мають різну кількість сигналів, і що, як правило, сигнали від різних ідентифікаторів представляють різні сигнали, наприклад, Signal1_of_ID від id0 та Signal1_of_ID від id5 відрізняються, не є однаковими. В таблиці 3.1 наведено розподіл між CAN ID та відповідною кількістю сигналів.

Таблиця 3.1 - розподіл між CAN ID та відповідною кількістю сигналів

CAN ID	Кількість відповідних сигналів
id1	2
id2	3
id3	2
id4	1
id5	2
id6	2
id7	2
id8	1
id9	1
id10	4

Тестові дані поділяються на шість підмножин рівної тривалості часу.

Одна з підмножин використовується для оцінки моделі на звичайних даних. Інші п’ять тестових наборів даних використовуються для оцінки моделі на наступних типах атак:

- атака плато (plateau attack);
- атака безперервних змін (continuous attack);
- атака повторного відтворення (playback attack);
- флуд атака (flooding attack);
- атака придушення (suppress attack).

Тривалість типового інтервалу атаки становить від 2 до 4 секунд. Фрагмент нормальних даних наведено на рисунку 3.1. Розглянемо кожен з видів атак більш детально.

Атака плато полягає в тому, що один сигнал замінюється на постійне значення протягом певного періоду часу, наприклад моделюється стрибок або заморожування сигналу. Графічно приклад цього типу атаки наведено на рисунку 3.2, рожевим кольором відмічено аномальний інтервал часу.

Атака безперервних змін представляє собою вторгнення, коли сигнал перезаписується так, що він повільно відходить від справжнього значення. Це передбачає, що зловмисник бажає встановити сигнал конкретного значення, намагаючись обдурити CAN ID реальними невеликими змінами в сигналі. На рисунку 3.3 наведено приклад атаки безперервних змін.

Атака відтворення має на меті перезапис значення сигналу протягом певного періоду із заздалегід записаними значеннями цього сигналу. Зловмисник сподівається скоїти вторгнення, надіславши цілком реальні значення сигналу в іншій ситуації руху. Приклад цього типу атаки наведено на рисунку 3.4.

Флуд атака полягає в тому, що зловмисник надсилає повідомлення певного ідентифікатора з високою частотою на шину CAN. Цю атаку на практиці простіше здійснити, ніж вищезазначені, оскільки зловмиснику не потрібно керувати ЕБУ. Потрібно лише надіслати додаткові повідомлення на шину CAN автомобіля, щоб «перезаписати» реальні значення

повідомлень. На рисунку 3.5 наведено графічний приклад цієї атаки.

Атака придушення скоюються наступним чином: зловмисник заважає ЕБУ надсилати повідомлення, наприклад, вимикаючи його. Цей вид атак передбачає, що повідомлення певного ідентифікатора не відображаються на CAN-трафіку протягом певного періоду часу. Приклад цього типу атаки наведено на рисунку 3.6, пунктиром відмічено фрагмент сигналу, який було придушено.

В тестових даних введені багаторазові атаки даного типу. Кожна атака супроводжується декількома секундами звичайного руху. Атака завжди відбувається на один сигнал або його ідентифікатор протягом певного періоду часу. Прогятом кожного інтервалу часу атаки ідентифікатор аномалії для всіх CAN ID встановлюється рівним одиниці.

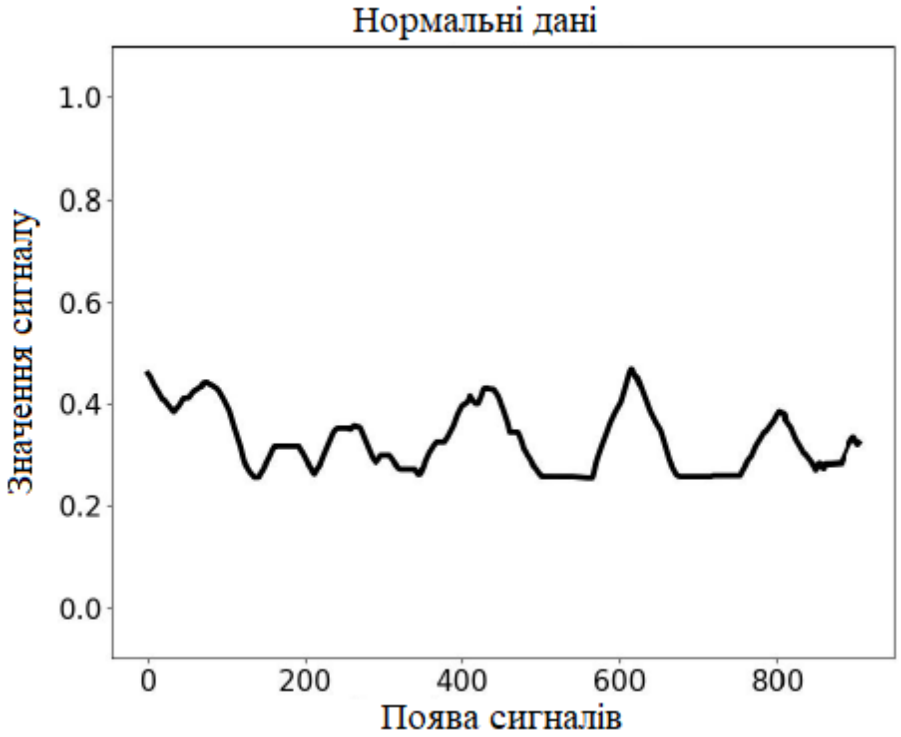

Рисунок 3.1 – Графічне представлення фрагменту нормальних даних

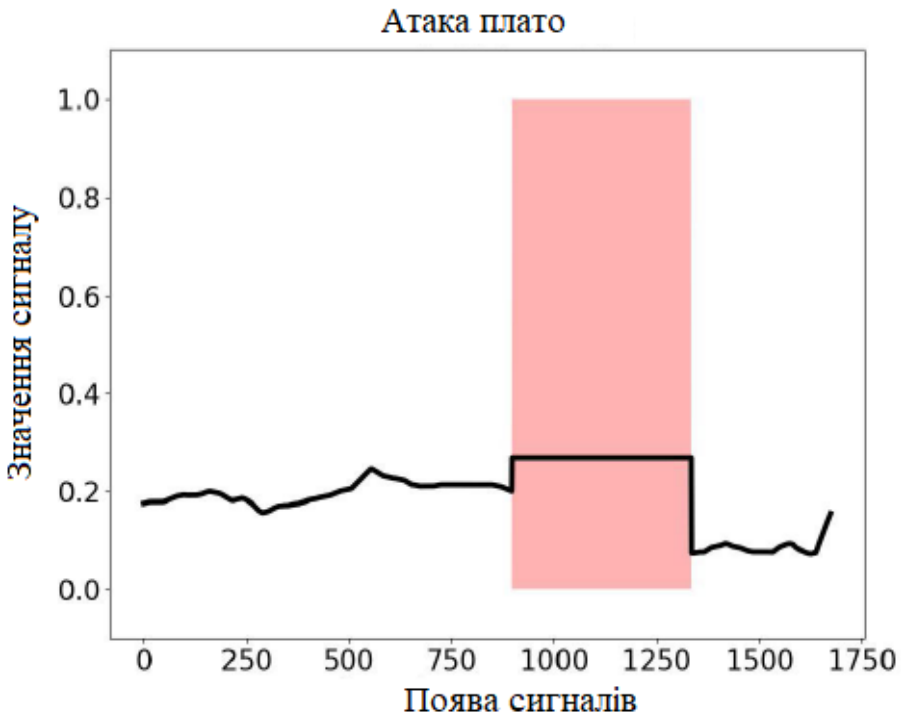

Рисунок 3.2 – Графічне представлення фрагменту даних з типом атаки плато

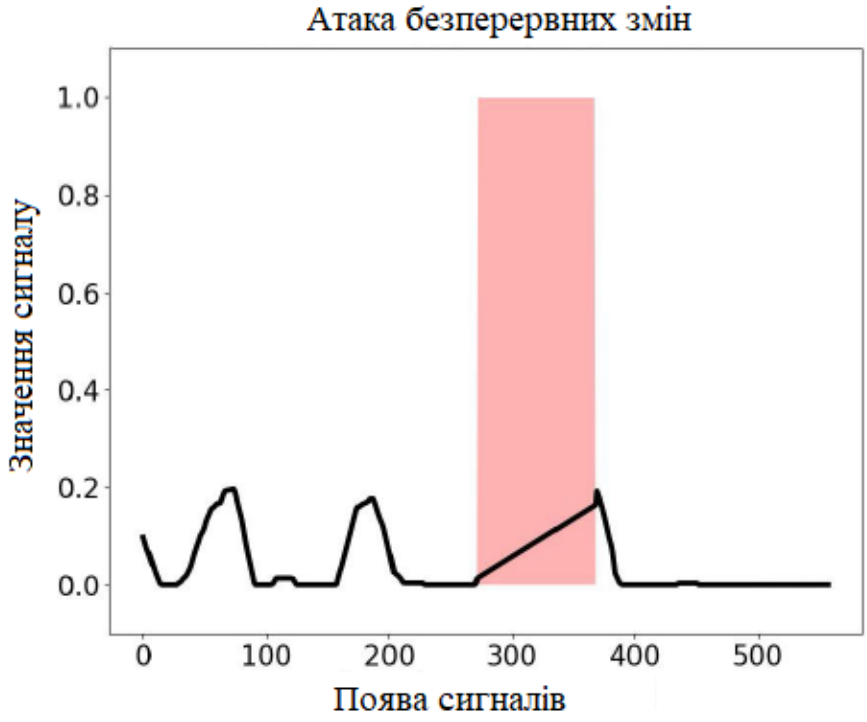

Рисунок 3.3 – Графічне представлення фрагменту даних з атакою типу безперервних змін

Рисунок 3.4 – Графічне представлення фрагменту даних з типом атаки повторне відтворення

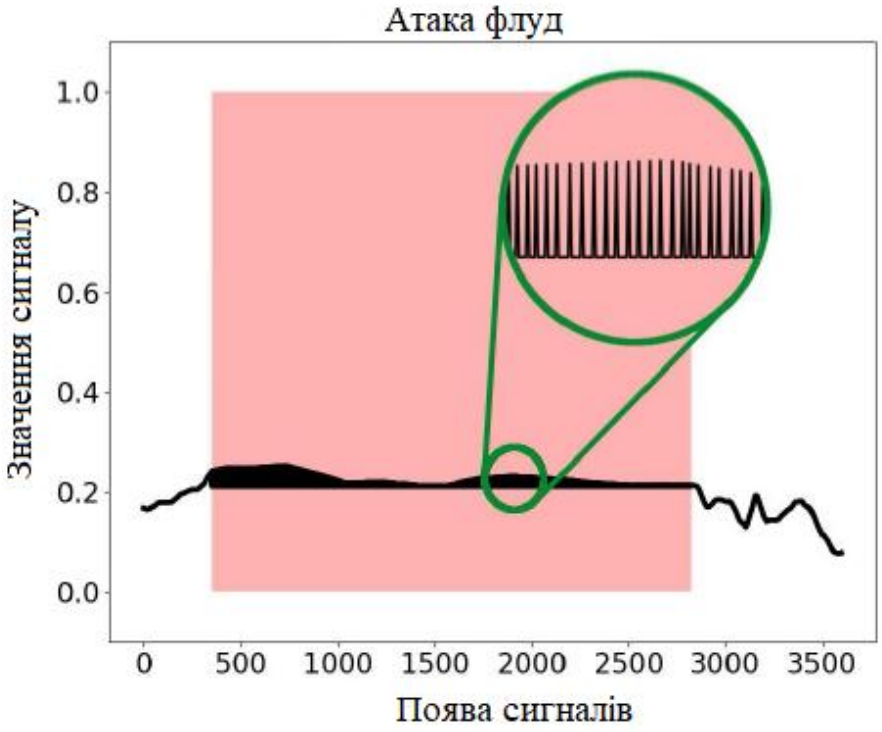

Рисунок 3.5 – Графічне представлення фрагменту даних з типом атаки флуд

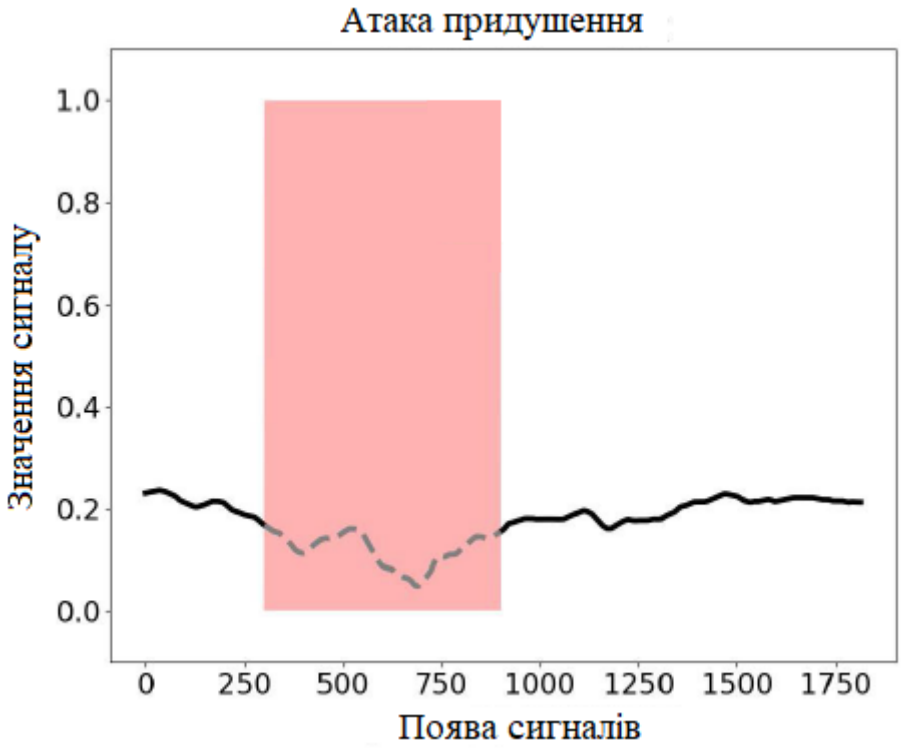

Рисунок 3.6 – Графічне представлення фрагменту даних з атакою типу придушення

На рисунку 3.7 зображено фрагмент файлу з даними test_plateau.csv.

	Label	Time	ID	Signal1_of_ID	Signal2_of_ID	Signal3_of_ID	Signal4_of_ID
0	0	6.750704e+07	id3	0.600000	1.000000	NaN	NaN
1	0	6.750704e+07	id1	0.165060	0.250000	NaN	NaN
2	0	6.750705e+07	id7	0.059647	0.500000	NaN	NaN
3	0	6.750705e+07	id8	0.230451	NaN	NaN	NaN
4	0	6.750705e+07	id5	0.037628	0.972196	NaN	NaN
5	0	6.750706e+07	id6	0.973684	0.333333	NaN	NaN
6	0	6.750706e+07	id3	0.800000	1.000000	NaN	NaN
7	0	6.750706e+07	id1	0.165400	0.500000	NaN	NaN
8	0	6.750706e+07	id9	0.415013	NaN	NaN	NaN
9	0	6.750706e+07	id7	0.059647	0.750000	NaN	NaN

Рисунок 3.7 – Фрагмент файлу з даними test_plateau.csv

3.2 Програмна реалізація алгоритму виявлення аномалій у сигналах з CAN шини автомобіля

3.2.1 Опис мови програмування та фреймворків для реалізації алгоритму виявлення аномалій у сигналах з CAN шини автомобіля

Для програмної реалізації алгоритму діаризації розмов спікерів було використано високорівневу мову програмування, Python. Головною перевагою цієї мови є кросплатформність, тобто за допомогою Python можливо писати програми на такі операційні системи, як Windows, Max OS X та Unix.

Також Python пропонує краще відстеження помилок, ніж мова C, і, будучи мовою високого рівня, має вбудовані типи даних високого рівня, такі як словники та динамічні масиви.

Обрана мова дозволяє розділити програму на модулі, які можна повторно використовувати в інших програмах на Python. Одразу при встановлюванні користувачу надається велика колекція стандартних модулів, які можна використовувати як основу програм, наприклад, для роботи з файлами, системні виклики і навіть засоби для розробки графічного інтерфейсу користувача.

Python дозволяє писати програми компактно та читабельно. Програми, написані за допомогою цієї мови, зазвичай коротші, ніж еквівалентні на C, C++ або Java, з кількох причин:

– типи даних високого рівня дозволяють виразити складні операції більш коротко;

– групування виразів здійснюється шляхом відступу замість початкових та кінцевих дужок;

– не потрібні декларації змінних або аргументів.

Мова програмування Python розширювана: при володінні мовою C, можна легко додати нову вбудовану функцію або модуль до

інтерпретатора, або виконувати критичні операції на максимальній швидкості, або зв’язати програму на Python з бібліотеками, які можуть бути доступними тільки у двійковій формі [23].

NumPy є основним пакетом Python для наукових обчислень. Він надає можливості виконання сновних математичних операцій, наприклад, з лінійної алгебри, над N-вимірними масивами. Також може виступати обгорткою для коду, написаному на C,C++ або Fortran. Чисельна обробка виконується за допомогою багатовимірних масивів, які називаються ndarray [24].

Pandas є бібліотекою Python із розширеними структурами даних та інструментами для роботи зі структурованими наборами даних, загальними для статистики, фінансів, соціальних наук та багатьох інших галузей. Бібліотека забезпечує інтегровані інтуїтивно зрозумілі процедури для виконання загальних маніпуляцій з даними та аналізу таких наборів даних. Вона прагне стати основним рівнем майбутнього статистичних обчислень у Python. Pandas служить потужним доповненням до існуючого наукового стеку Python, одночасно впроваджуючи та вдосконалюючи види інструментів маніпулювання даними, знайдені в інших мовах статистичного програмування, таких як R [25].

Scikit-learn – це модуль Python, що інтегрує широкий спектр найсучасніших алгоритмів машинного навчання проблем навчання з вчителем і без. Цей пакет фокусується на наданні машинного навчання неспеціалістам, які використовують мову загального призначення високого рівня. Акцент робиться на простоті використання, продуктивності, документації та узгодженості API. Він має мінімальні залежності і поширюється за спрощеною ліцензією Berkeley Software Distribution, заохочуючи його використовувати як в академічних, так і в комерційних умовах [26].

Основним фреймворком, який використовувався, є PyTorch. Він є інструментом машинного та глибокого навчання, який розроблений

підрозділом штучного інтелекту компінії Facebook для розробки модулів з аналізу широкомасштабних зображень, включаючи виявлення об'єктів, сегментацію та класифікацію. Однак сфера його застосування не обмежується цими завданнями. Він може бути використаний з іншими фреймворками для реалізації більш складних алгоритмів. PyTorch написаний з використанням мов програмування Python та C++. Є можливість використовувати даний фреймворк в середовищі графічного процесору (GPU), щоб прискорити виконання алгоритмів [27].

Для полегшення прототипування PyTorch не дотримується символічного підходу, що використовується в багатьох інших системах глибокого навчання, а зосереджується на диференціації суто обов’язкових програм, з акцентом на розширюваність та низькі накладні витрати.

Цей фреймворк ґрунтується на кількох проектах, зокрема, Lua Torch, Chainer та HIPS Autograd. PyTorch, як і більшість інших бібліотек глибокого навчання, підтримує зворотний режим автоматичної диференціації скалярних функцій (або векторно-якобієвих продуктів функцій з кількома виходами), найважливішою формою автоматичної диференціації для програм глибокого навчання, які зазвичай диференціюють лише одне скалярне значення втрати. У цьому домені підтримка PyTorch для автоматичної диференціації випливає з етапів Chainer, HIPS autograd та twitter-autograd (сам twitter-autograd був портом HIPS autograd до Lua). Ці бібліотеки мають дві відмінні риси: динамічне виконання, визначене за запуском (define-by-run execution) та негайне виконання (eager execution).

Динамічний фреймворк визначає функцію, яку слід диференціювати, просто виконуючи потрібне обчислення, на відміну від зазначення статичної структури графіка, яка символічно диференціюється заздалегідь, а потім запускається багато разів. Це дозволяє користувачам використовувати будь-які функції хост-мови, які вони хочуть (наприклад, довільні конструкції потоку керування). Динамічне виконання відрізняє

PyTorch від статичних фреймворків, таких як TensorFlow, Caffe тощо.

Eager execution запускає обчислення тензорів у міру зустрічі з ними; це дозволяє записydfnи лише те, що необхідно для диференціації обчислень. Негайне виконання дозволяє конвеєрувати обчислення процесора та графічного процесора, але позбавляє можливості для оптимізації цілої мережі та пакетної обробки.

Незважаючи на те, що підтримка PyTorch автоматичної диференціації була натхненна попередниками (особливо twitter-autograd та Chainer), вона представляє деякі нові варіанти дизайну та реалізації, що робить її однією з найшвидших реалізацій серед бібліотек автоматичної диференціації, що підтримує такий тип динамічного eager execution, а саме:

– операції на місці (in-place operations);
– відсутність стрічки;
– основна логіка в C ++.

Операції на місці представляють небезпеку для автоматичної диференціації, оскільки операція на місці може призвести до втрати даних, необхідних на фазі диференціації. Крім того, вони вимагають нетривіального перетворення стрічки. PyTorch реалізує прості, але ефективні механізми, які вирішують обидві ці проблеми.

Традиційна диференціація в зворотному режимі записує стрічку (також відому як список Венгерта), що описує порядок, у якому операції виконувались спочатку; ця оптимізація дозволяє реалізаціям уникати топологічного сортування. PyTorch уникає цієї стрічки; натомість кожен проміжний результат записує лише підмножину графіку обчислень, яка мала значення для їх обчислення. Це означає, що користувачі PyTorch можуть змішувати та поєднувати незалежні графіки, як їм подобається, у будь-яких потоках, які їм подобаються (без явної синхронізації). Додатковою перевагою структурування графіків таким чином є те, що коли частина графіку стає мертвою, вона автоматично звільняється; важливим

фактором, коли ми хочемо якнайшвидше звільнити великі фрагменти пам'яті.

PyTorch розпочав своє життя як бібліотека Python; однак швидко стало зрозуміло, що накладні витрати на інтерпретатор занадто великі для основної логіки AD. Сьогодні більшість із них написано на C++. Ретельно налаштований код C ++ є однією з основних причин, чому PyTorch може досягти значно менших накладних витрат порівняно з іншими фреймворками.

На рисунку 3.8 наведено простий приклад автоматичного диференціювання в PyTorch. Користувач пише код так, ніби безпосередньо виконує тензорні операції; однак, замість того, щоб оперувати тензорами, користувач маніпулює змінними, які зберігають додаткові метадані, необхідні для AD. Змінні підтримують метод backward(), який обчислює градієнт усіх вхідних змінних, що беруть участь у обчисленні цієї величини.

За замовчуванням ці градієнти накопичуються у полі grad вхідних змінних, конструкції, успадкованій від Chainer. Однак PyTorch також надає функціональний інтерфейс HIPS у стилі автоградієнта для обчислення градієнтів: функція torch.autograd.grad (f (x, y, z), (x, y)) обчислює похідну f тільки від x та y (градієнт для z не обчислюється). На відміну від API у стилі Chainer, цей виклик не мутує атрибути .grad; натомість він повертає кортеж, що містить градієнт [28].

```
from torch.autograd import Variable
x, prev_h = Variable(torch.randn(1, 10)), Variable(torch.randn(1, 20))
W_h, W_x = Variable(torch.randn(20, 20)), Variable(torch.randn(20, 10))

i2h = torch.matmul(W_x, x.t())
h2h = torch.matmul(W_h, prev_h.t())
(i2h + h2h).tanh().sum().backward()
```

Рисунок 3.8 – Приклад використання модуля автоматичної диференціації PyTorch

3.2.2 Опис програмної реалізації

Першим кроком у розв’язанні будь-якої задачі машинного навчання є попередня обробка даних. Спочатку було сконкатеновано чотири файли формата csv у один загальний. Уривок коду з файлу concat_data.py наведено нижче, лістинг 3.1.

Лістинг 3.1 – Фрагмент файлу concat_data.py

```
    train_1 = pd.read_csv(PATH_TRAIN_DATA.format(1))
    columns = train_1.columns
    train_2      =      pd.read_csv(PATH_TRAIN_DATA.format(2),
names=columns)
    train_3      =      pd.read_csv(PATH_TRAIN_DATA.format(3),
names=columns)
    train_4      =      pd.read_csv(PATH_TRAIN_DATA.format(4),
names=columns)
    train_full   =   pd.concat([train_1,   train_2,   train_3,
train_4])
    train_full.to_csv(PATH_TRAIN_DATA.format("full"),
index=False)
```

Наступним кроком була реалізація запропонованої нейромережевої структури. У лістингу 3.2 наведено перевизначення обов’язкового методу forward() з файлу model.py, який визначає, як буде функціонувати мережа, перетворювати вхідне значення у вихідне.

Лістинг 3.2 – Фрагмент файлу model.py

```
    def forward(self, x, i):
            """
            @brief:  performs  corresponding  model  branch
forward pass
            @param x: input. torch tensor with shape of (1, 1,
payload_size)
            @param i: model id
            @return  x:  result  of  model  forward  pass.
Reconstruction of the input signal
            """
            x,        states        =        self.layers[i](x,
(self.hidden_states[i],  self.cell_states[i]))    #  forward  on
corresponding LSTM
            self.hidden_states[i] = x  # update
```

Далі було реалізовано процедуру навчання. Вибраним оптимізатором є Adam з початковим значенням коефіцієнту швидкості навчання 0.01. Вхідні сигнали мережі масштабуються з нормалізацією у діапазоні від нуля до одиниці.

Нейромережева система навчається протягом 1000 епох із розміром пакета 25. Кожен елемент у ньому являє собою серію з 5000 послідовних повідомлень, яка бере початок у випадковому місці навчальних даних. На початку кожної епохи вектори прихованого стану та стану ядра підмереж LSTM ініціалізуються нулем. Під час однієї епохи зворотне поширення помилки здійснюється кожні 250 позначок часу. Для більш надійного навчання функція втрат помножується на фіксований скаляр для різних ідентифікаторів, CAN ID. Тобто, цей скаляр лінійно менший, чим частіше його відповідний ідентифікатор з’являється у наборі навчальних даних. Це робиться для того, щоб CAN ID, які з’являлися частіше, не набирали більшої ваги, ніж рідші ідентифікатори під час навчання.

Навчання проводиться на наборі навчальних даних після видалення невеликої підмножини для обчислення порогових значень для оцінки аномалії.

У лістингу 3.3 наведено фрагмент файлу train.py, в якому у методі setup() визначено функцію втрат, оптимізатор та коефікієнт швидкості навчання.

Лістинг 3.3 – фрагмент файлу train.py

```
    def setup(self):
            """
            @brief: setup training configuration
            @return: None
            """
            self.main_model    =    CanNet(self.signals_per_id,
self.h_scale)
            self.loss = nn.MSELoss(reduction='sum')
            self.optimizer                                    =
torch.optim.Adam(self.main_model.parameters(), lr=0.01)
            #     Not     sure     if     max_decay_steps=200,
end_learning_rate=0.0001 are the "best" parameters
```

Також у файлі training.py визначено всі необхідні константи для виконання процедури навчання нейромережевої системи, а саме кількість епох, кількість сигналів для кожного CAN ID, обчислювальну потужність системи, розмір пакета даних, частоту виконання зворотного поширення помилки, розмір послідовності та заздалегідь обчисленні скаляри для різних ідентифікаторів. У лістингу 3.4 наведено фрагмент файлу training.py.

Лістинг 3.4 – фрагмент файлу training.py

```
if __name__ == "__main__":
    # The Number of training epochs.
    epochs_num = 1000
    # The list of payloads' sizes.
    signals_per_id = [2, 3, 2, 1, 2, 2, 2, 1, 1, 4]
    # The computational power of CANet.
    h_scale = 5
    # Number of the parallel executions in one batch
    batch_size = 25
    # Number of time steps after which we should perform
back propagation
    update_interval = 250
    # The continuous sequence length
    seq_size = 5000
    # Occurrence frequencies of IDs
    dataset_inv_freqs = [0.8604701494979494,
0.930226829399656, 0.860469893331757, 0.9534846063010607,
                         0.8604697972694348,
0.9302270535450743, 0.860469893331757, 0.8604700534356273,
                         0.9302269254619782,
0.9534847984257051]
```

Набір даних, попередньо виділених з навчальних, використовувався для обчислення порогових значень оцінки аномалій. У лістингу 3.5 наведено фрагмент файлу anomalies_utils.py, якому реалізовано метод get_anomalies_thresholds(), що обчислює порогове значення для кожного сигналу.

Лістинг 3.5 – фрагмент файлу anomalies_utils.py

```
    def get_anomalies_thresholds(self, model, psizes,
consecutive=True, perc=99):
            model.eval()
            validation_set                                    =
self.__get_validation_set(consecutive=consecutive)
            validation_set    =    validation_set.reshape(-1,
validation_set.shape[-1])
            losses_per_signal   =   {i:   []   for   i   in
range(len(psizes))}
            for i, sample in enumerate(tqdm(validation_set)):
                signal_id = int(sample[0] - 1)
                payload_size = psizes[signal_id]
                x    =    torch.as_tensor(sample[1:    1    +
payload_size],        dtype=torch.float32).reshape(1,        1,
payload_size)
                sample_loss   =   self.__evaluate(x,   model,
psizes, signal_id)
                if i < self._ignore_first:
                    continue

losses_per_signal[signal_id].append(sample_loss.numpy())
            thresholds = [np.percentile(losses_per_signal[i],
perc) for i in range(len(psizes))] ###
            return thresholds
```

Для обчислення якості виявлення аномалій, було обрано використовувати істинно позитивний та істинно негативний показники. Їх оцінку реалізовано у файлі evaluation_metrics.py, фрагмент якого наведено у лістингу 3.6.

Лістинг 3.6 – фрагмент файлу evaluation_metrics.py

```
    def get_tp_tn_rates(ground_truth, prediction):
        conf_matrix     =     confusion_matrix(ground_truth,
prediction)
        if np.shape(conf_matrix) == (1, 1):
            raise ValueError("There is only one class in
'ground_truth' and 'prediction' lists.")
        tn, fp, fn, tp = conf_matrix.ravel()
        if (tp == 0 and fn == 0) or (tn == 0 and fp == 0):
            raise ValueError("Cannot count the true positive
or true negative rate
    .")
        true_positive_rate = tp / (tp + fn)
        true_negative_rate = tn / (tn + fp)
        return true_positive_rate, true_negative_rate
```

Файл evaluating.py містить код для оцінки виявлення аномалій навченою нейромережевою системою. У лістингу 3.7 наведено його фрагмент, де зчитується тестовий файл, завантажується навчена мережа та запускається процедура оцінки.

Лістинг 3.7 – фрагмент файлу evaluating.py

```
    if __name__ == "__main__":
        # The path to the test file.
        TEST_FILE_PATH = "./data/test/test_plateau.csv"
        # The path to pre-trained models' weights.
        WEIGHTS_PATH  =  "./weights/weights_dump_final_epoch-
999.pt"
        model.load_state_dict(torch.load(WEIGHTS_PATH))
        # Sets the module in evaluation mode.
        model.eval()
        # Evaluating.
        test_model(model,       psizes,       TEST_FILE_PATH,
anomaly_thresholds=anomaly_thresholds)
```

3.3 Оцінка точності

Нейромережева система була навчена протягом 1000 епох на навчальних даних з кінцевим значенням функції втрат 0.0005. Отримано графік залежності функції втрат від кількості епох, який наведено нижче на рисунку 3.9.

Наступним кроком, після етапу навчання та обчислення порогових значень аномалій, стала оцінка точності моделі. Як було зазначено раніше, для тестових даних з атаками, для обчислення якості виявлення аномалій, було обрано використовувати істинно позитивний та істинно негативний показники. Для тестових даних без атак використовувалася звичайна точність бінарної класифікації. Всі метрики були оцінені при трьох різних значеннях параметра h_{scale}, а саме 5, 10 та 20, який відповідає за обчислювальну потужність розробленої неромережевої системи. У таблиці 4 наведено показник точності виявлення аномалій на нормальних тестових даних, без атак. Накращий результат, згідно цієї метрики, досягнуто при

значенні параметра h_{scale}, який відповідає за обчислювальну потужність мережі, 20.

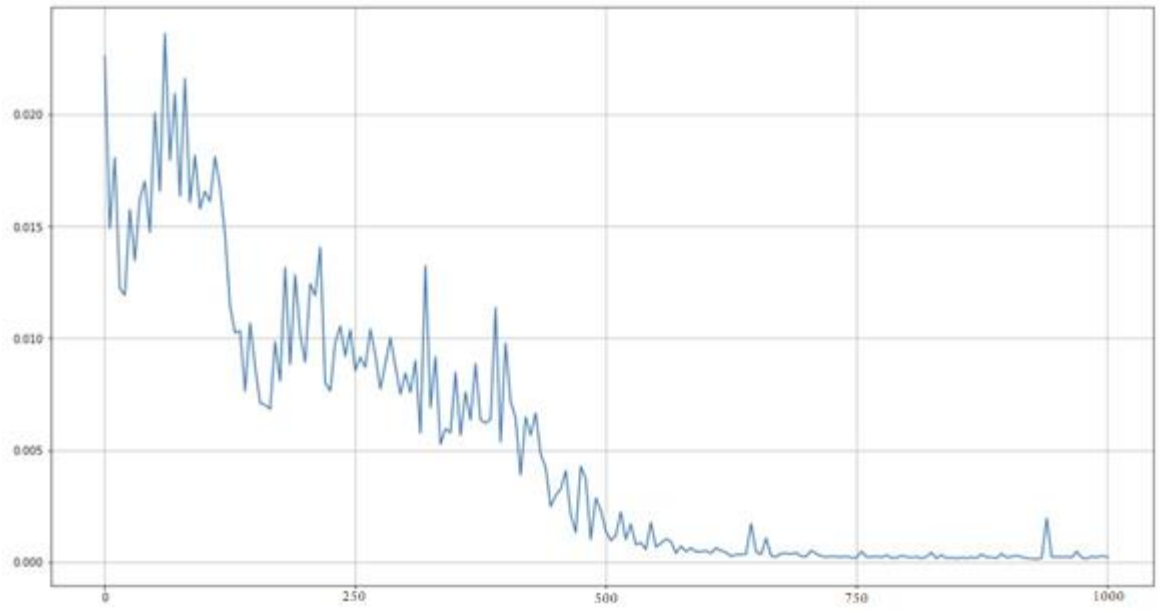

Рисунок 3.9 – Графік залежності функції втрат навчання моделі від кількості епох

Наступним кроком, після етапу навчання та обчислення порогових значень аномалій, стала оцінка точності моделі. Як було зазначено раніше, для тестових даних з атаками, для обчислення якості виявлення аномалій, було обрано використовувати істинно позитивний та істинно негативний показники.

Для тестових даних без атак використовувалася звичайна точність бінарної класифікації. Всі метрики були оцінені при трьох різних значеннях параметра h_{scale}, а саме 5, 10 та 20, який відповідає за обчислювальну потужність розробленої неромережевої системи.

У таблиці 3.2 наведено показник точності виявлення аномалій на нормальних тестових даних, без атак. Накращий результат, згідно цієї метрики, досягнуто при значенні параметра h_{scale} 20.

Таблиця 3.2 – показник точності виявлення аномалій на нормальних тестових даних

h_{scale}	Точність виявлення аномалій
5	0.985
10	0.984
20	0.986

У таблиці 3.3 наведені істинно позитивний (true positive rate) та істинно негативний (true negative rate) показники точності виявлення аномалій, на тестових даних з типом атаки плато. Найкраще значення істинно позитивного показника досягнуто при $h_{scale} = 10$, а істинно негативного – $h_{scale} = 20$.

Таблиця 3.3 – показник точності виявлення аномалій на тестових даних з типом атаки плато

h_{scale}	true positive rate	true negative rate
5	0.886	0.97
10	0.945	0.965
20	0.875	0.983

У таблиці 3.4 наведені true positive rate та true negative rate точності виявлення аномалій, на тестових даних з типом атаки безперервних змін. Найкращі значення досягнуто при $h_{scale} = 20$.

Таблиця 3.4 – показник точності виявлення аномалій на тестових даних з типом атаки безперервних змін

h_{scale}	true positive rate	true negative rate
5	0.73	0.984
10	0.755	0.984
20	0.761	0.986

Наступним типом атак, на якому оцінювалась точність виявлення аномалій, стала атака повторного відтворення. Результати тестування наведено у таблиці 3.5. Найкращі значення досягнуто при $h_{scale} = 20$.

Таблиця 3.5 – показник точності виявлення аномалій на тестових даних з типом атаки повторне відтворення

h_{scale}	true positive rate	true negative rate
5	0.886	0.986
10	0.895	0.986
20	0.896	0.987

У таблиці 3.6 наведені істинно позитивний (true positive rate) та істинно негативний (true negative rate) показники точності виявлення аномалій, на тестових даних з типом атаки флуд. Найкраще значення істинно позитивного показника досягнуто при $h_{scale} = 10$, а істинно негативного – $h_{scale} = 20$.

Останнім типом атак, на якому оцінювалась точність виявлення аномалій, стала атака повторного придушення. Результати тестування наведено у таблиці 3.7. З точки зору значень істинно позитивного показника, цей вид атаки є найпроблемнішим. Найкращі значення досягнуто при $h_{scale} = 10$.

Таблиця 3.6 – показник точності виявлення аномалій на тестових даних з типом атаки флуд

h_{scale}	true positive rate	true negative rate
5	0.89	0.983
10	0.891	0.986
20	0.874	0.987

Таблиця 3.7 – показник точності виявлення аномалій на тестових даних з типом атаки придушення

h_{scale}	true positive rate	true negative rate
5	0.486	0.986
10	0.601	0.986
20	0.571	0.985

Роблячи висновок, найкраший результат було досягнуто при значенні параметра h_{scale} 20, який відповідає за обчислювальну потужність розробленої неромережевої системи. Однак, для того, щоб навчити модель з таким параметром, необхідно значно більше часу, ніж, наприклад, з h_{scale}=20.

На рисунку 3.10 зображено візуалізацію приблизних вимог до пам'яті, а саме приблизну кількість параметрів нейромережевої системи звлежно від значення параметра h_{scale} та кількості сигналів.

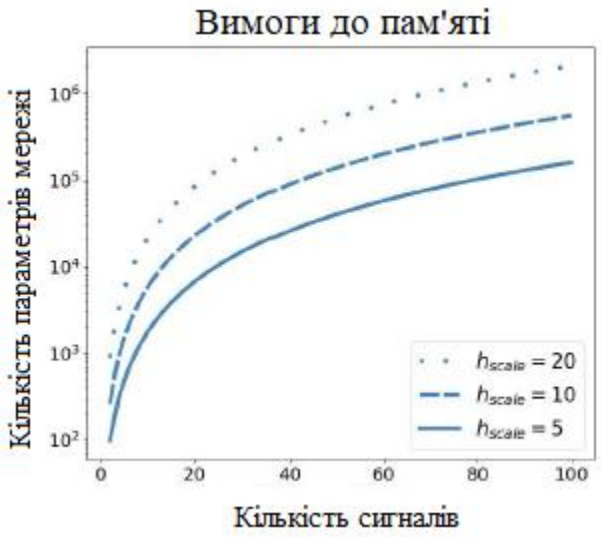

Рисунок 3.10 – Візуалізація приблизних вимог до пам'яті для навчання системи

ВИСНОВКИ

У цій атестаційній роботі магістра був запропонований алгоритм для виявлення аномалій у сигналах з CAN шини автомобіля на основі методів глибинного навчання.

Застосування запропонованого методу дає можливість таких типів атак, як плато, повторне відтворення, придушення, флуд та атака безперервних змін, у свою чергу це може запобігти дорожньо-транспортним пригодам.

Реалізація полягає в об'єднанні нейромережевих систем на основі мереж довгої короткочасної пам'яті та автокодувальника для отримання реконструкції вхідного сигналу та ідеї визначення порогових значень функції втрат для виявлення аномалій. Оцінка якості виявлення аномалій проводилася за допомогою істинно позитивних (true positive) та істинно негативних (true negative) показників разом з точністю (accuracy).

В атестаційній роботі магістра представлені результати, які є відповідно до поставленої мети рішенням актуального завдання виявлення аномалій у сигналах з CAN шини автомобіля на основі методів глибинного навчання.

Проведені дослідження дозволили зробити наступні висновки:

– проведений аналіз існуючих алгоритмів виявлення аномалій, вторгнень, у сигналах з CAN шини автомобіля показав, що основні алгоритми, які використовуються в рамках напряму Machine Learning, призначені тільки для роботи з даними, які належать одному з CAN ID одночасно;

– в рамках атестаційної роботи магістра було розроблено алгоритм для виявлення аномалій, вторгнень, у сигналах з CAN шини автомобіля на основі методів глибинного навчання, який здатний обробляти складну структуру даних із сигналів декількох ідентифікаторів CAN в одній моделі;

– проведено імітаційне моделювання на наборі даних, фкий

предоставлено компанією Robert Bosch GmbH для дослідження якості виявлення аномалій, вторгнень, у сигналах з CAN шини автомобіля;

– було оцінено якість діаризації розмов спікерів за допомогою істинно позитивних (true positive) та істинно негативних (true negative) показників разом з точністю (accuracy);

– розроблена система виявлення аномалій у сигналах з CAN шини автомобіля на основі мереж довгої короткочасної пам'яті та автокодувальника для отримання реконструкції вхідного сигналу показав високу точність, значення істинно негативного показника позначки близько 98%;

– розроблена система навчалась у режимі без вчиталя, тобто вона ніколи не бачила атак під час тренувань. Отже, очікується, що модель знайде подальші невідомі сценарії атак, крім тих, що представлені в цій роботі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Taylor A., Leblanc S., Japkowicz N. Anomaly detection in automobile control network data with long short-term memory networks. *2016 IEEE International Conference on Data Science and Advanced Analytics*. IEEE. 2016. P. 130–139.

2. Martinelli F., Mercaldo F., Nardone V., Santone A. Car hacking identification through fuzzy logic algorithms. *2017 IEEE International Conference on Fuzzy Systems*. IEEE. 2017. P. 1–7.

3. Marchetti M., Stabili D., Guido A., Colajanni M. Evaluation of anomaly detection for in-vehicle networks through informationtheoretic algorithms. *2016 IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI)*. IEEE. Sept 2016. P. 1–6.

4. Chandola V., Banerjee A., Kumar V. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*. 2009. P. 1–72.

5. Markou M., Singh S. Novelty detection: a review – part 1: statistical approaches. *Signal Processing*. 2003. March 31. P. 1–17.

6. Checkoway S., McCoy D., Kantor B., Anderson D. Comprehensive experimental analyses of automotive attack surfaces. *USENIX Security Symposium*. San Francisco, P. 77–92.

7. Cho K./, Shin K. Fingerprinting electronic control units for vehicle intrusion detection. *USENIX Security Symposium*. Austin. 2016. P. 911–927.

8. CAN Specification, Bosch. Robert Bosch GmbH. Stuttgart, Germany, 1991. 72 p.

9. Müter M., Asaj N. Entropy-based anomaly detection for in-vehicle networks. *2011 Intelligent Vehicles Symposium (IV)*. IEEE. 2011. P. 1110–1115.

10. Miller C., Valasek C. Remote exploitation of an unaltered passenger vehicle. Black Hat USA. 2015. 91 p.

11. Song H. M., Kim H. R., Kim H. K. Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. *2016 International Conference on Information Networking (ICOIN)*, IEEE. 2016. P. 1–9.

12. Narayanan S. N., Mittal S., Joshi A. OBD SecureAlert: An Anomaly Detection System for Vehicles, *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE. 2016. P. 1–6.

13. Tomlinson A., Bryans J., Shaikh S. A. Towards viable intrusion detection methods for the automotive controller area network. *2nd Computer Science in Cars Symposium - Future Challenges in Artificial Intelligence Security for Autonomous Vehicles (CSCS 2018)*. September 2018, P. 1–9.

14. Taylor A., Japkowicz N., Leblanc S. Frequency-based anomaly detection for the automotive CAN bus. *2015 World Congress on Industrial Control Systems Security (WCICSS)*. WCICSS. 2015. P. 45–49.

15. Hanselmann M., Strauss T., Dormann K., Ulmer H. CANet: An Unsupervised Intrusion Detection System for High Dimensional CAN Bus Data. IEEE Access. 2020. 11 p.

16. Hochreiter S., Schmidhuber J. Long short-term memory. *Neural Computation*, 1997. Vol. 9. P. 1735–1780.

17. Baldi P. Autoencoders, unsupervised learning, and deep architectures. *Workshop on Unsupervised and Transfer Learning*. Jun. 2012. P. 37–49.

18. Zhou C., Paffenroth R. C. Anomaly detection with robust deep autoencoders. *23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA. 2017. P. 665–674.

19. Goodfellow I., Bengio Y., Courville A. Deep Learning. 7th ed. MIT Press, 2016. 802 p.

20. Clevert D.-A., Unterthiner T., Hochreiter S. Fast and accurate deep network learning by exponential linear units (elus). *2016 International Conference on Learning Representations*. 2016. P. 1–14

21. Elliot D. L. A better activation function for artificial neural networks.

University of Maryland, College Park. 1993. 4 p.

22. Stuart A.; Ord K. Kendall's Advanced Theory of Statistics, Classical Inference and the Linear Model. 3rd ed. Wiley, 1994. 912 p.

23. Rossum G. Python Tutorial. Release 3.9.0. Python Software Foundation, 2020. 149 p.

24. Bressert E. SciPy and NumPy: An Overview for Developers. O'Reilly Media, 2012. 66 p.

25. McKinney W. Pandas: a foundational python library for data analysis and statistics. Python for High Performance and Scientific Computing, 2011. 9 p.

26. Pedregosa F., Varoquaux G., Gramfort A. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*. 2011. P. 2825 – 2830.

27. Pradeepta M. PyTorch Recipes: A Problem-Solution Approach. Apress, 2019. 183 p.

28 Paszke A., Gross S., Chintala S. Automatic differentiation in pytorch. *Neural Information Processing Systems Workshop*, 2017. P. 1–4.