Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів інтелектуального
аналізу мережевого трафіку веб-орієнтованих додатків
(тема)

Виконав:
студент 2 курсу, групи СШІм-20-3
Корнієнко О. О.
(прізвище, ініціали)

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

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

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник зав. каф. ІУС Петров К. Е.
(посада, прізвище, ініціали)

Допускається до захисту

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

2022 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-наукова
(освітньо-професійна або освітньо-наукова)

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

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Корнієнко Олексію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів інтелектуального аналізу мережевого трафіку веб-орієнтованих додатків
затверджена наказом університету від 24 березня 20 22 р. № 414Ст
2. Термін подання студентом роботи до екзаменаційної комісії _____ _____ 20 ___ р.
3. Вихідні дані до роботи документація до мові програмування Python, документація до фреймворку Keras, документація до СУБД PostgreSql

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної галузі та постановка задачі, аналіз існуючих систем, огляд існуючих методів та алгоритмів передбачення часових даних, реалізація обраного типу моделі та порівняльний аналіз різних модифікацій, проектування системи для моніторингу та аналізу навантаження веб орієнтованої системи

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 1.1 – Структура МАМТ, Рисунок 1.2 – загальна структура МАМТ, Рисунок 2.1 – структура глибокої нейронної мережі, Рисунок 2.2 – зв'язок між ШІ там ГН, Рисунок 2.3 – Графічний опис глибокої архітектури згорткових нейронних мереж, Рисунок 2.4 – рекурента нейронна мережа, Рисунок 2.5 – внутрішня структура LSTM, Рисунок 2.6 – клієнт-серверна архітектура, Рисунок 2.7 – модернізована клієнт-серверна архітектура, Рисунок 2.8 – схема БД, Рисунок 2.9 – архітектура додатку для аналізу та передбачення навантаження, Рисунок 2.10 – Структура докеру, Рисунок 3.1 – приклад даних, Рисунок 3.2 – код для візуалізації тестових даних, Рисунок 3.3 – візуалізація тестових даних, Рисунок 3.4 – нормалізація даних, Рисунок 3.5 – розділення даних на тестову та тренувальну вибірку, Рисунок 3.6 – функцію для створення нового набору даних, Рисунок 3.7 – підготовка навчальної та тестової вибірки, Рисунок 3.8 – підготовка навчальної та тестової вибірки, Рисунок 3.9 – реалізація моделі, Рисунок 3.10 – перевірка точності моделі

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання	28.03.2022	виконано
2	Аналіз предметної області та постановка задачі	02.04.2022– 07.04.2022	виконано
3	Аналіз методів щодо вирішення поставленої задачі	08.04.2022– 13.04.2022	виконано
4	Вибір методу та вирішення задачі	14.04.2022– 19.04.2022	виконано
5	Експериментальне моделювання та навчання моделі	20.04.2022– 29.04.2022	виконано
6	Розробка моделей	30.04.2022– 07.05.2022	виконано
7	Написання пояснювальної записки	08.05.2022–13.05.2022	виконано
8	Попередній захист	14.05.2022	виконано
9	Захист перед ЕК	15.05.2022	виконано

Дата видачі завдання 28 березня 20 22 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 58 с., 32 рис., 1 табл., 1 дод., 21 джерел.

АНАЛІЗ МЕРЕЖЕВОГО ТРАФІКУ, НЕЙРОННІ МЕРЕЖІ, ПРОЕКТУВАННЯ СИСТЕМ, ЧАСОВО-ПРОСТОРОВІ ДАННІ

Об'єкт дослідження – процес моніторингу та аналізу мережевого трафіку на базі моделей машинного навчання.

Предмет дослідження – методи моніторингу та аналізу мережевого трафіку сервісів веб-орієнтованих додатків для виявлення небезпечних запитів, шляхом використання методів машинного навчання.

Метою роботи є проектування системи моніторингу та аналізу мережевого трафіку сервісів веб-орієнтованих додатків для виявлення небезпечних запитів та аномальної поведінки в мережі, шляхом використання методів машинного навчання для своєчасного реагування на небезпечні запити та аномалії в мережевому трафіку.

Дослідження, що проведені в роботі базуються на використанні теорії ймовірностей та математичної статистики, теорії графів, апарату штучних нейронних мереж та методів машинного навчання для розв'язання завдання розробки методу аналізу мережевого трафіку з сервісом його моніторингу; методів комп’ютерного моделювання для верифікації точності запропонованої моделі.

У роботі проведено порівняння різних методів моніторингу та аналізу мережевого трафіку. Детальний опис методу моніторингу та аналізу мережевого трафіку для обраної предметної області. Опис архітектури системи моніторингу мережевого трафіку в синтезі з відповідним методом аналізу мережевого трафіку, а також опис інтерфейсів взаємодії з користувачем.

РЕФЕРАТ

Пояснительная записка: 58 с., 32 рис., 1 таб., 1 прил., 21 источник.

АНАЛИЗ СЕТЕВОГО ТРАФИКА, ВРЕМЕННО-ПРОСТРАННЫЕ ДАННЫЕ, НЕЙРОННЫЕ СЕТИ, ПРОЕКТИРОВАНИЕ СИСТЕМ

Объект исследования – процесс мониторинга и анализа сетевого трафика на основе моделей машинного обучения.

Предмет исследования – методы мониторинга и анализа сетевого трафика сервисов веб-ориентированных приложений для выявления опасных запросов путем использования методов машинного обучения.

Цель работы – проектирование системы мониторинга и анализа сетевого трафика сервисов веб-ориентированных приложений для выявления опасных запросов и аномального поведения в сети путем использования методов машинного обучения для своевременного реагирования на опасные запросы и аномалии в сетевом трафике.

Исследования, проведенные в работе, базируются на использовании теории вероятностей и математической статистики, теории графов, аппарата искусственных нейронных сетей и методов машинного обучения для решения задачи разработки метода анализа сетевого трафика с сервисом его мониторинга; методов компьютерного моделирования для верификации точности предлагаемой модели

В работе проведены сравнения различных методов мониторинга и анализа сетевого трафика. Подробное описание метода мониторинга и анализа сетевого трафика выбранной предметной области. Описание архитектуры системы мониторинга сетевого трафика в синтезе с подходящим способом анализа сетевого трафика, также описание интерфейсов взаимодействия с пользователем.

ABSTRACT

Explanatory note: 58 p., 32 fig., 1 table, 1 ann., 21 sources.

NEURAL NETWORKS, NETWORK TRAFFIC ANALYSIS, SYSTEM DESIGN, TIME-SPACE DATA

The object of study – is the process of monitoring and analysis of network traffic based on machine learning models.

The subject of study – is methods of monitoring and analysis of network traffic of web-oriented application services to detect dangerous requests by using machine learning methods.

The purpose of the work – is to design a system for monitoring and analyzing network traffic of web-oriented application services to detect dangerous requests and unexpected behavior in the network, using machine learning methods to respond in a timely manner to dangerous requests and anomalies in network traffic.

The research cond ucted in this work is based on the use of probability theory and mathematical statistics, graph theory, artificial neural networks and machine learning methods to solve the problem of developing a method of network traffic analysis with its monitoring service; computer modeling methods to verify the accuracy of the proposed model.

In the graduate work compared different methods of monitoring and analyzing network traffic. Detailed description of the method of monitoring and analysis of network traffic for the selected subject area. Description of the architecture of the network traffic monitoring system in synthesis with the appropriate method of network traffic analysis, and a description of the user interfaces.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МН – машинне навчання;

НМ – нейронна мережа;

ГН – глибинне навчання;

МАМТ – моніторинг та аналіз мережевого трафіку;

ШІ – штучний інтелект;

РНН – рекурентна нейрона мережа;

LSTM – long short-term memory – довга короткочасна пам'ять.

ВСТУП

Використання методів машинного навчання в наш час є дуже актуальним. Ці методи широко застосовуються при створенні різноманітних інтелектуальних систем. Машинне навчання (МН) − це галузь штучного інтелекту, що швидко розвивається. За визначенням знаного фахівця з комп’ютерних наук та піонера машинного навчання [1] Тома М. Мітчелла: «Машинне навчання − це вивчення комп’ютерних алгоритмів, які дозволяють комп'ютерним програмам автоматично вдосконалюватися через досвід» [2].

Алгоритм можна розглядати як набір інструкцій, які задає програміст, котрі комп'ютер здатний обробити. Простіше кажучи, алгоритми МН вивчаються за досвідом, подібно до того, як люди. Наприклад, побачивши декілька прикладів об'єкта, алгоритм МН зможе розпізнати цей об'єкт у нових, раніше небачених сценаріях.

МН використовує різноманітні методи інтелектуальної обробки великої та складної кількості інформації для прийняття рішень та / або прогнозів.

На практиці, закономірності, які вивчає комп'ютер (система МН), можуть бути дуже складними і важкими для пояснення. Наприклад, пошук зображень автомобіля у Google. Google неймовірно гарний у досягненні відповідних результатів, але як пошук в Google досягає цього результату? Пошук Google спочатку отримує велику кількість прикладів фотографій з написом «автомобіль» − потім комп'ютер (система МН) шукає візерунки пікселів і кольорів, які допомагають йому відгадати (передбачити) що це справді автомобіль.

МН сьогодні неймовірно важливе, по-перше, тому що воно може вирішувати складні реальні проблем, а, по-друге, тому, що це торкнулося різних галузей протягом останнього десятиліття [3], і буде продовжувати це робити і надалі, оскільки все більше лідерів та дослідників галузі

спеціалізуються на МН, для того, щоб продовжувати свої дослідження та / або розробляти інструменти МН для позитивного впливу на певні сфери. Швидкість, з якою розвивається МН, дуже вражає [4], завдяки різким змінам у зберіганні та обчисленні даних та підвищенню потужності їх обробки − оскільки все більше людей залучається, ми можемо лише очікувати, що МН і надалі буде сприяти приголомшливому прогресу у різних сферах.

Один з прикладів використання МН це створення нейронних мереж (НМ) для побудови прогнозів, саме з цим наміром я буду використовувати МН в цій роботі. «Прогноз» є результатом роботи алгоритму після того, як він був навчений на історичному наборі даних і застосований до нових даних. Алгоритм генеруватиме ймовірні значення для кожного запису в нових даних.

Слово «прогноз» може ввести в оману. У деяких випадках це дійсно означає, що ви прогнозуєте майбутній результат, наприклад, коли ви використовуєте МН для визначення наступної найкращої дії в маркетинговій кампанії. Однак в інших випадках «прогноз» пов’язаний, наприклад, із тим, чи була транзакція, яка вже відбулася, шахрайською. У цьому випадку транзакція вже відбулася, але ви робите обґрунтовані припущення щодо того, чи була вона законною чи ні, що дозволяє вам прийняти відповідне рішення.

Для різних видів прогнозу можуть бути використані різні види НМ, наприклад: багатошарові персептрони, згорткові нейронні мережі, повторювані нейронні мережі.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис предметної області

Протягом останніх років методам моніторингу та аналізу мережевого трафіку, далі МАМТ, приділяється велика увага як важливій темі досліджень в підтримці продуктивності мережі. В якості поширених рішень в управлінні мережею, методи МАМТ були впроваджені як промисловістю, так і науковцями. Хоча були запроваджені різні методи МАМТ, нові мережеві технології та парадигми зробили створення ефективних мереж складним. Нові мережі з тисячами вузлів, наприклад, інтернет речей необхідно контролювати на регулярній основі, щоб підтримувати їх ефективність. Різні цілі в мережі спонукають адміністраторів оцінювати її з точки зору, наприклад, проблеми безпеки, підтримки вимог до якості обслуговування та покращення споживання ресурсів, і це лише деякі з них. Ці цілі задовольняються шляхом застосування методів МАМТ, наприклад. виявлення аномалій, класифікації мережевого трафіку, керування несправностями та прогнозування трафіку. Методи МАМТ поділяються на дві основні групи: активні та пасивні методи.

Активні методи [4] передбачають генерацію та введення пробного трафіку в мережу, щоб дізнатися про стан мережі. Точніше, дані тестового трафіку вводяться в мережу на основі запланованої вибірки, а потім будуть виміряні різні показники продуктивності мережі. Приклади показників включають пропускну здатність мережі, коефіцієнт втрат пакетів, затримку та тремтіння (або зміну затримки). Оскільки активні методи моніторингу дають уявлення про продуктивність у режимі реального часу, вони є основними методами контролю послуг, що базуються на угоді про рівень обслуговування. На противагу цьому, пасивні методи в основному використовуються для моніторингу та аналізу реального мережевого трафіку в мережі. Пасивні методи МАМТ представляють великий інтерес

для промисловості, з точки зору розв'язання задач управління та планування. Ці методи можна використовувати для ретельного моніторингу трафіку, особливо в ситуаціях після події, наприклад, відмовостійкість та усунення несправностей. Крім того, вони ідеально підходять для отримання глибокого уявлення про якість досвіду користувача. Застосування активних та пасивних методів узагальнено в таблиці 1.1.

Таблиця 1.1 – Порівняння активних та пасивних методів

Активні методи	Пасивні методи
Прямий і наскрізний аналіз	Кроки для усунення несправностей
Якість обслуговування	Якість досвіду
Моніторинг у режимі реального часу	Діагностика проблем протоколу
Моніторинг ефективності мережі та послуг	Моніторинг у режимі реального часу
Моніторинг наскрізних транспортних процесів у режимі реального часу	Моніторинг обслуговування та досвіду клієнтів

Зростання комунікаційних систем і мереж з точки зору кількості користувачів і обсягу створеного трафіку ставить перед МАМТ різні щоденні проблеми, які включають:

1) зберігання та аналіз даних про трафік;
2) використання даних про трафік для досягнення бізнес-цілей завдяки отриманню інформації;
3) інтеграція даних трафіку;
4) перевірка даних трафіку;
5) безпека даних трафіку;
6) отримання даних про трафік.

Безпрецедентне збільшення кількості підключених вузлів і обсягу даних посилюють складність мережі, що вимагає продовження досліджень для аналізу та моніторингу продуктивності мережі. Крім того, наявність величезної та неоднорідної кількості даних трафіку вимагає прийняття нових підходів до моніторингу та аналізу даних керування мережею. Через ці проблеми більшість робіт зосереджені саме на одному аспекті MAMT: виявлення аномалій та класифікація трафіку [4].

Серед проблем, згаданих вище, отримання даних про трафік представляє величезні технічні труднощі в області MAMT, особливо для активних вимірювань, оскільки доводиться використовувати зонди для оцінки прогресу важливих параметрів мережі з часом. Зонди є одними з найефективніших методів для отримання уявлень про наскрізну продуктивність, яку відчувають кінцеві користувачі. Активний і пасивний зонди − це дві поширені стратегії, які можуть покращити продуктивність наскрізного вимірювання та визначити якість обслуговування та якість експлуатації шляхом здобуття детальних даних про трафік. Активний зонд намагається емулювати мережевий трафік, а потім надсилати емульований трафік всередині мережі, щоб виміряти наскрізну продуктивність (наприклад, затримку). У порівнянні з активними зондами, пасивні зонди представляють особливий погляд на мережу. Пасивні зонди розміщуються на каналах мережі, і вони пропускають весь трафік, який передається через з'єднання, що контролюється.

Що стосується конкретних мережевих сценаріїв і цілей збору даних про трафік (наприклад, класифікація трафіку та виявлення вторгнень), можна визначити різні вимоги до отримання даних про трафік. Іншими словами, не потрібно отримувати всі доступні дані з мережі в задачі збору трафіку. Отже, мережеві пакети зазвичай розглядаються як центральні цілі, які повинні бути перевірені в задачах збору даних про трафік. Для моніторингу мережевого трафіку для оцінки його продуктивності існують два основних методи, включаючи перевірку неглибоких пакетів і глибоку

перевірку пакетів. Оскільки перший стосується збору інформації із заголовків пакетів мережевого трафіку, другий обробляє весь вміст пакету, включаючи дані користувача. Зонди можуть використовувати обидві методи для збору інформації про мережу, але глибока перевірка пакетів має деякі недоліки, зокрема:

1) аналіз даних користувачів може поставити під загрозу конфіденційність користувачів;

2) обробка всього пакета потребує більше часу та ресурсів, ніж обробка заголовка;

3) у деяких типах мережевого трафіку, наприклад, віртуальна приватна мережа та зашифрований мережевий трафік, глибока перевірка пакетів неможлива.

Виходячи зі згаданих вище проблем із використанням глибокої перевірки пакетів, більшість зондів у нових методах MAMT використовують неглибоку перевірку пакетів. Надавши довідкову інформацію про дослідження, хочу підкреслити, що однією з важливих проблем у MAMT є отримання великої кількості надійних даних про трафік [5]. Щоб впоратися з цією проблемою, протягом останніх років були запропоновані деякі інструменти. Однак адаптивний та ефективний підхід до збору даних, який можна було б широко використовувати в гетерогенних і великомасштабних сучасних мережах, все ще відсутній.

Мережеві пакети поки що є найпоширенішим форматом даних для збору мережевого трафіку. Проте найкращі методи збору мережевих пакетів стикаються з втратою пакетів, особливо коли вони стають із великою кількістю проблемних пакетів. Більше того, ці методи мають труднощі з високошвидкісними каналами і стають неефективними через низьку пропускну дію. Методики збору даних на основі потоку є ще одним популярним механізмом збору даних. Потокова мережа − це набір мережевих пакетів з однаковими характеристиками, такими як адреса джерела/призначення та порти джерела/призначення [5]. У порівнянні з

такими механізмами, методи збору даних на основі потоків можуть зменшити кількість необхідних завдань для аналізу пакетів і забезпечити кращу продуктивність, особливо в гігабітних мережах. Тим не менше, фільтрація пакетів і потоків може серйозно заперечити ці методи.

Сучасні мережеві рішення знаходяться під тиском нового явища, відомого як великі дані. Цей факт ґрунтується на особливих характеристиках даних керування мережею, наприклад, великий обсяг, висока швидкість, висока правдивість і велика різноманітність. Дані керування мережею відносяться до всіх даних, які відображають ситуацію в мережі, в основному витягнуті із заголовків пакетів (функція на рівні пакетів), наприклад, затримка пакета, часові позначки та тип пакета. Методи МАМТ можна вважати одними з основних споживачів великих даних. Крім того, через складність даних він стає важливою галуззю дослідження в контексті аналітики великих даних. Звичайні методи обробки даних для МАМТ включають:

1) математичні та статистичні методи (наприклад, регресія для аналізу часових рядів);

2) алгоритми МН та підходи до обробки великих даних (наприклад, контрольоване навчання для виявлення вторгнень).

Методи МАМТ повинні виконувати послідовність кроків для перетворення необроблених даних про трафік в корисну інформацію. Використання звичайних методів для аналітики великих даних стикається з багатьма проблемами та проблемами, включаючи точність, високошвидкісну аналітику та ефективну обробку великих даних у режимі реального часу. Крім того, на основі нових парадигм, таких як Інтернет речей, велика кількість підключених пристроїв щодня виробляє величезний обсяг необроблених даних, і, отже, нам потрібні ефективніші методології для моніторингу та аналізу такої величезної кількості необроблених даних. дані більш ефективним способом з точки зору обробки часу та простору.

Як згадувалося вище, техніка МН приділяла велику увагу в техніці МАМТ. Методи ML згруповані в чотири групи, як зазначено нижче:

1) навчання з наглядом;
2) навчання з напівнаглядом;
3) навчання без нагляду;
4) навчання з підкріпленням.

Серед різних методів МН, глибоке навчання, далі ГН, є ключовим кроком для значного полегшення аналітики та отримання знань у сфері великих даних. ГН використовується в багатьох областях, включаючи комп'ютерний зір, охорону здоров'я, транспорт і розумне сільське господарство. Крім того, ГН також привернув увагу технологічних компаній. Великі компанії, такі як Twitter, YouTube і Facebook, виробляють величезні обсяги даних щодня, і тому для них надзвичайно важливо обробляти ці великі дані. Алгоритми ГН використовуються для аналізу отриманих даних та вилучення значущої інформації, оскільки традиційні методи обробки даних майже неможливо обробити такою величезною кількістю даних. Можна виділити чотири ключові напрями: класифікацію трафіку, прогнозування трафіку, управління помилками та безпеку мережі як основні теми МАМТ (рисунок 1.1).

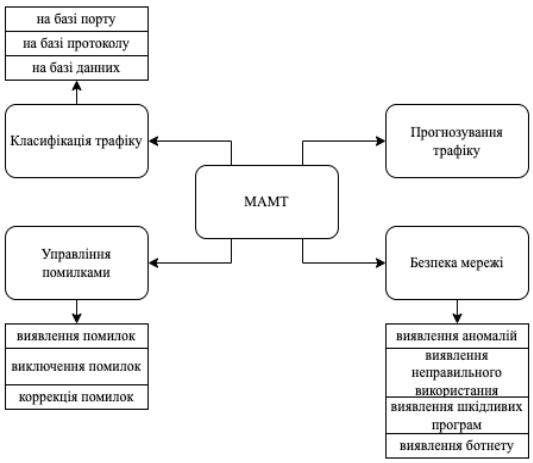

Рисунок 1.1 − Структура МАМТ

2.2 Структура методів моніторингу та аналізу мережевого трафіку

МАМТ відноситься до низки методів моніторингу мережевого трафіку на відповідному рівні деталізації (наприклад, на рівні пакетів). Методи МАМТ оперують даними та знаннями про роботу та продуктивність мережі та поведінку користувачів. У контексті комунікаційних систем і мереж МАМТ відіграє важливу роль у розумінні того, як працюють мережі і процеси моніторингу ступеню продуктивності мереж; як кінцеві споживачі використовують ресурси та оптимізують використання ресурсів; як можна ефективно контролювати та керувати телекомунікаційними інфраструктурами для забезпечення SLA.

У зв'язку зі стрімким зростанням кількості підключених пристроїв і обсягу даних про трафік, потрібно розробляти більш досконалі методи МАМТ для забезпечення стабільності та доступності систем зв'язку. Нижче ми зосередимося на загальній структурі МАМТ, яка складається з п'яти кроків [6]. Більшість існуючих дослідницьких робіт повністю або частково відповідають структурі, що представлена на рисунку 1.2.

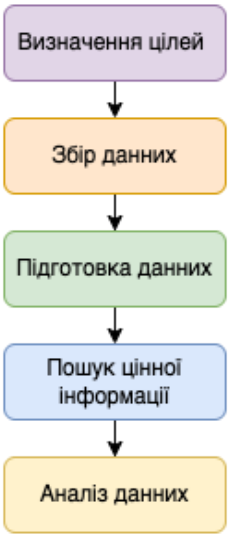

Рисунок 1.2 − Загальна структура МАМТ

Першим кроком до МАМТ є чітке визначення цілей МАМТ. Як згадувалося вище, типові цілі включають: класифікацію трафіку, прогнозування трафіку, управління помилками та безпеку мережі. Залежно від цілей аналізу трафіку, можливо, доведеться працювати над різними підцілями, щоб служити головній меті. Наприклад, якщо метою МАМТ є класифікація мережевого трафіку, підціллю може бути категоризація даних про трафік на різні класи на основі їх міток, наприклад трафік VPN і не-VPN або Firefox і Chrome. Другим кроком є збір даних керування мережею за допомогою пасивних або активних методів моніторингу. У зв'язку з тим, що ці два методи забезпечують різні погляди на стан мережі, їх можна використовувати разом, щоб скористатися перевагами обох методів.

Попередня обробка та очищення даних можуть сильно вплинути на продуктивність МАМТ, особливо підходів на основі МН. У мережі розподіл функцій на рівні пакетів може змінюватися через деякі звичайні дії, такі як повторна передача пакетів і дубльовані ACK. Отже, видалення таких даних керування мережею може покращити продуктивність додатків МАМТ, наприклад, прогнозування трафіку. Нормалізація є ще одним методом попередньої обробки для покращення продуктивності додатків МАМТ, особливо це надзвичайно важливо для підходів на основі МН та ГН.

Потім, після попередньої обробки даних, МАМТ має пройти етап вибору ознак, щоб вибрати найбільш інформативні ознаки для досягнення цілі. Вибір ознак може виконуватися автоматично або вручну [7]. У першому випадку алгоритми вибору ознак використовуються для виділення найбільш релевантних ознак, а в другому використовуються знання предметної області для виконання вибору ознак.

Після наведених вище кроків експерти з аналізу даних проводять поглиблений аналіз попередньо оброблених даних, щоб отримати значущу інформацію. Як згадувалося у вступному розділі, математичні та статистичні методи, алгоритми машинного навчання та підходи до великих даних є традиційними підходами для отримання значущих знань із

необроблених даних. Вибір найбільш підходящої моделі чи методики з існуючих підходів є важливим для надійного та відтворюваного статистичного висновку. Підходи, засновані на МН, долають математичні та статистичні методи завдяки їх здатності виявляти приховані закономірності щодо вихідних даних. У комунікаційних системах і мережах методи МН використовуються в багатьох програмах, таких як система виявлення вторгнень, виявлення аномалій, моніторинг та виявлення шаблонів.

1.3 Постановка задачі

Для вирішення проблеми моніторингу та аналізу мережевого трафіку потрібно щоб методи вирішення проблеми та засоби реалізації відповідали сучасним потребам.

В роботі буде проведено порівняльний аналіз класичних методів моніторингу та аналізу мережевого трафіку та методів моніторингу та аналізу мережевого трафіку які базуються на машинному навчанні. Серед різноманіття методів МАМТ потрібно виділити найбільш підходящі. Методи МАМТ повинні вміти в великою кількістю даних в реальному часі, також вони повинні адаптуватися під різні види систем та мати можливість легко інтегруватися в існуючі мережі веб орієнтованих додатків.

Система МАМТ повинна працювати з часово-просторовими даними, фіксуючи відповідні залежності. Більшість даних управління мережею, зібраних у вигляді наборів даних часових рядів, можна буде з високою точністю аналізувати моделями ГН. Розгортання точних та ефективних методів для різних додатків МАМТ має першорядне значення.

На вході система МАМТ буде отримувати постійний потік даних, вхідні данні будуть проаналізовані системою МАМТ та система повинна автоматично проаналізувати наявність можливих аномалій в мережевому

трафіку веб-орієнтованого додатку та надати можливість живого моніторингу трафіку в мережі.

Метою даної кваліфікаційної роботи є дослідження наявних на сьогоднішній день методів машинного навчання для прогнозування навантаження веб-орієнтованих високонавантажених систем.

Для досягнення поставленої мети необхідно розглянути наступні задачі:

1) провести аналіз предметної області;
2) дослідити існуючі методи та порівняти їх;
3) розробити та описати власний підхід до розв’язання задачі прогнозування навантаження веб-орієнтованих високонавантажених систем;
4) навчити та протестувати розроблену модель рекурентної нейронної мережі з довгою короткостроковою пам’яттю для прогнозування навантаження веб-орієнтованих високонавантажених систем;
5) проаналізувати отримані результати.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ ТА ПРОЕКТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ МОНІТОРИНГУ ТРАФІКУ

2.1 Методи моніторингу та аналізу мережевого трафіку

Останніми роками ШІ привернув великий інтерес до багатьох випадків використання, таких як самокеровані автомобілі, чат-боти, віртуальні помічники та тощо. Історія ШІ починається з 50-х років минулого сторіччя, коли дослідники намагалися автоматизувати інтелектуальні завдання, які зазвичай виконують люди. Протягом дуже довгого часу багато експертів стверджували, що, сформулювавши великий набір явних правил для маніпулювання знаннями, вони можуть реалізувати штучний інтелект, схожий на людину. Цей підхід, також відомий як символічний ШІ, був домінуючим методом для досягнення штучного інтелекту на рівні людини з 1950-х до кінця 1980-х років. Незважаючи на те, що символічний ШІ успішно справлявся з чітко визначеними завданнями, такими як гра в шахи, він зіткнувся з труднощами з вирішенням більш складних завдань, таких як розпізнавання мовлення та класифікація зображень. Щоб вирішити цю проблему, машинне навчання виникло як новий підхід.

Поява машинного навчання вводить нову парадигму в програмування. У парадигмі символічного ШІ людина-агент вводить правила (програму) і дані, якими потрібно маніпулювати відповідно до цих правил, і дає результати. На відміну від цього, у машинному навчанні людина-агент вводить дані та очікувані результати від даних, а потім модель навчання дає правила. Потім ці правила застосовуються до нових даних для досягнення оригінальних результатів. Системи машинного навчання можна навчати, а не програмувати в явному вигляді. Це означає, що в ці системи надходить величезна кількість даних, щоб знайти значущі характеристики в цих даних. Потім ці характеристики можна використовувати для створення правил для автоматизації завдання. Машинне навчання зазвичай має проблеми з

великими та складними наборами даних, такими як набори даних зображень із тисячами чи навіть мільйонами екземплярів. Для класичного статистичного аналізу, такого як байєсівський аналіз, майже неможливо обробити такі великі набори даних. Отже, машинне навчання і особливо ГН показують відносно мало теорії математики і є інженерно-орієнтованим підходом.

ГН є специфічним розділом МН, в якому глибока нейронна мережа використовується для пошуку представлення даних на кожному рівні. Глибоке визначення ГН відноситься до ідеї послідовних шарів уявлень. Більше того, кількість шарів для моделювання даних відома як глибина моделі. Для складних завдань, таких як розпізнавання зображень, моделі ГН часто мають десятки або навіть сотні послідовних шарів уявлень. На відміну від ГН, інші моделі машинного навчання часто включають один або два шари для представлення даних. Архітектура ГН представлена на рисунку 2.1.

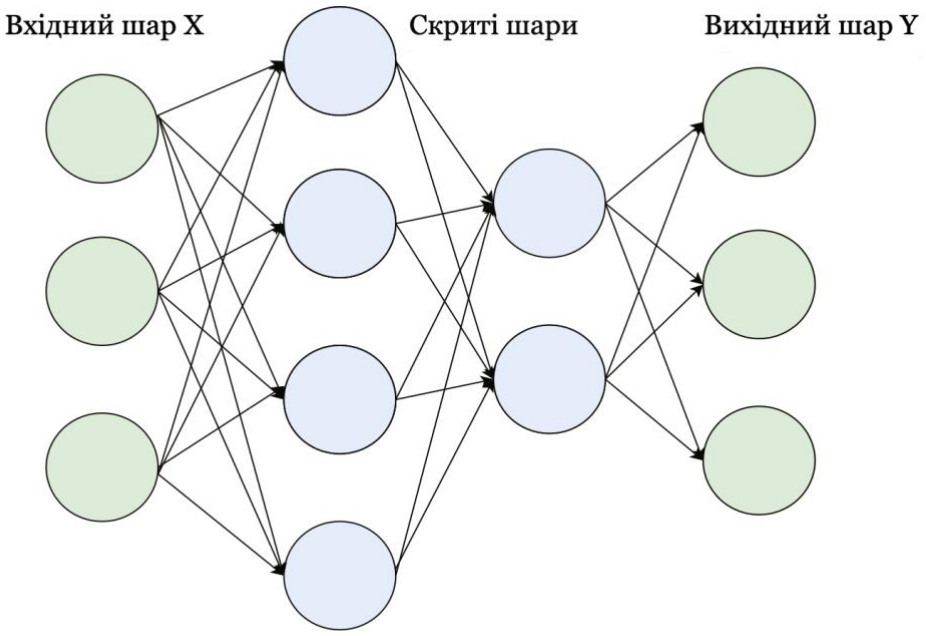

Рисунок 2.1 − Структура глибокої нейронної мережі

Як загальне визначення, можна стверджувати, що машинне навчання – це відображення вхідних даних (наприклад, відео та зображень) на цілі (наприклад, мітку «собака»), що досягається шляхом надання моделі безлічі випадків введення і цілей. Подібним чином можна побачити, що ГН виконує відображення вхідних даних і цілей через глибокі послідовні шари перетворення даних. Модель ГН вивчає ці перетворення, спостерігаючи за багатьма прикладами введення/цілі.

У моделі ГН ваги шару, також відомі як параметри, визначають, які перетворення будуть виконані до вхідних даних шару. Згідно з простим визначенням «ваги», вони являють собою набір чисел. У контексті ГН навчання відноситься до пошуку набору правильних значень для ваг усіх шарів у моделі, щоб модель точно відображала вхідні дані до відповідних цілей. Через те, що моделі DL можуть мати десятки мільйонів параметрів (ваги), визначення правильного значення для всіх цих параметрів є складним завданням. На рисунку 2.2 узагальнено показаний взаємозв’язок між ШІ, машинним навчанням та ГН. Далі ми детально досліджуємо основні моделі ГН.

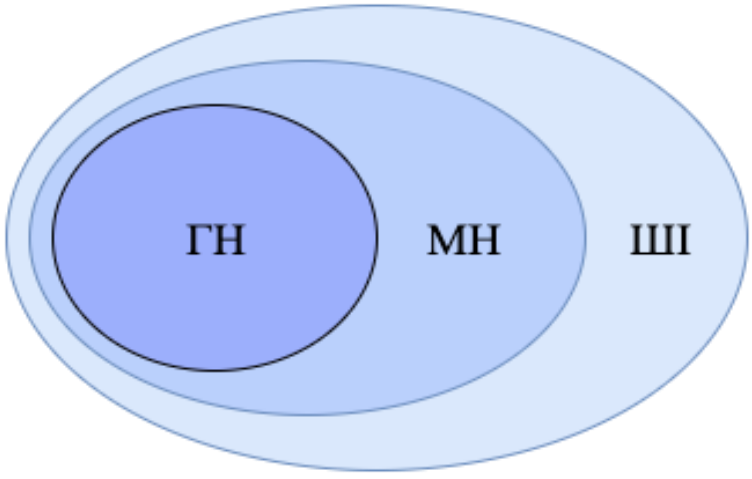

Рисунок 2.2 − Зв'язок між ШІ там ГН

2.2 Багатошаровий персептрон

Добре відомою категорією моделі ГН є пряма глибока мережа або багатошаровий персептрон . Модель багатошарового персептрону − це штучна нейронна мережа, яка відображає деякі приклади вхідних даних у цільові значення. Мережа формується шляхом складання декількох простих шарів (не менше трьох шарів). Ми можемо розглядати застосування кожного шару як надання нового представлення кожної точки даних.

Основною метою моделі багатошарового персептрону є апроксимація деякої функції f. Наприклад, у моделі класифікатора $y = f(x)$ відображає вхідні дані x у мітку y. Багатошаровий персептрон визначає відображення $y = f(x;\theta)$ і знаходить правильні значення для параметрів θ, які ведуть до найближчого наближення функції. У глибоких мережах із прямим зв'язком визначення прямої зв'язку відноситься до ідеї, що вхідні дані проходять через функцію, яка оцінюється з x, потім протікає через проміжні обчислювальні одиниці, що використовуються для визначення f, і, нарешті, надходять до вихідних y. Слід зазначити, що в моделі багатошарового персептрону немає зв'язків зворотного зв'язку для зворотного зв'язку вихідних даних моделі в себе.

Багатошаровий персептрон має щонайменше три шари, в яких обчислювальні блоки (або нейрони) щільно з’єднані з одиницями наступного шару. Враховуючи ці налаштування, багатошаровий персептрон виконує наступну операцію:

$$\mathbf{y} = \sigma(W*x+b), \qquad (2.1)$$

У цьому виразі y є вихідним сигналом шару, W позначає ваги навчання, а b вказує нейрони зміщення. Крім того, $\sigma()$ − це функція активації, яка має на меті покращити навчання моделі, дозволяючи її

нелінійність. Найпоширенішими нелінійними функціями активації є такі [7]:

1) сигмовидна (або логістична);
2) гіперболічний тангенс (або Tanh);
3) ReLU (або випрямлений лінійний блок);
4) Leaky ReLU.

Функції активації ReLU та Leaky ReLU пропонуються для вирішення критичної проблеми в інших функціях активації, яка називається градієнтним зникненням. Проблема стосується того, коли градієнти функції втрат будуть зникаюче малими і не можуть поширюватися через шари.

2.3 Згорткові нейронні мережі

Згорткові мережі, також звані згортковими нейронними мережами, є специфічним типом нейронних мереж, які спеціалізуються на обробці даних, подібних до сітки. Прикладами цього типу даних є часові ряди та зображення, які можна розглядати як 1-вимірну сітку та 2-вимірну сітку пікселів відповідно. Згорткові мережі широко використовуються в різноманітних реальних проблемах, таких як обробка природної мови, комп'ютерний зір, розпізнавання мовлення тощо [8]. Термін «згортка» в згорткових нейронних мережах підтримує цю ідею про те, що CNN використовують математичну операцію. називається згорткою. У своїй найбільш поширеній формі оператор згортки − це специфічний тип лінійної операції, яка виконує інтеграл від добутку двох функцій/сигналів. Іншими словами, згорткові нейронні мережі − це нейронні мережі, які використовують оператори згортки замість загального множення матриці принаймні на одному з їх мережевих рівнів. Згорткові нейронні мережі застосовують три ключові принципи, які можна застосувати для підвищення продуктивності системи МН шляхом скорочення простору параметрів

моделі: спільне використання параметрів або ваг, розріджені взаємодії та еквіваріантні уявлення.

Велика розмірність є очевидним недоліком архітектури глибинних нейронних мереж, особливо коли вхідні дані занадто великі та складні, наприклад, зображення. Щоб вирішити цю проблему, оператор згортки (або рівень згортки) був введений як альтернатива для повного підключення в архітектурі глибоких нейронних мереж [9]. Графічний опис глибокої архітектури згорткових нейронних мереж представлено на рисунку 2.3.

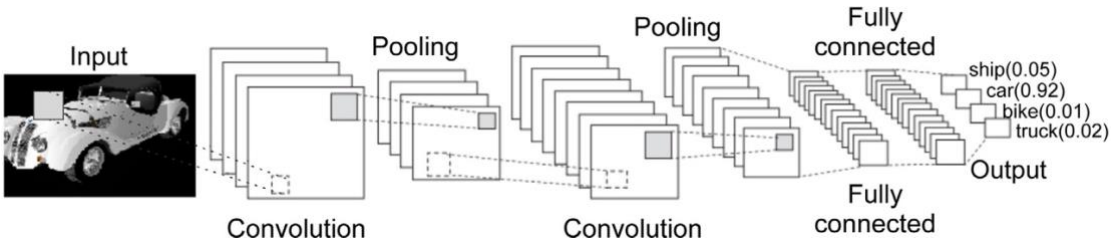

Рисунок 2.3 − Графічний опис глибокої архітектури згорткових нейронних мереж

Згорткова нейронна мережа приймає багатоканальні зображення (наприклад, автомобілі та кораблі) як вхідні дані для навчальних цілей. Згорткова нейронна мережа використовує переваги кількох шарів згортки з нелінійними функціями активації, щоб полегшити складність вхідних даних (тобто зображень) і отримати вихід, тобто ймовірність того, що кожне зображення належить до класу (або категорії). У згортковаій нейронній мережі кожна вхідна зона з'єднана з нейроном на виході, відомим як локальне підключення. Кожен шар використовує різні фільтри для розпізнавання абстрактних понять, наприклад, кордону транспортного засобу. Згорткова нейронна мережа може вивчати функції вищого рівня, такі як різні частини автомобіля, на більш глибоких шарах. Фільтри не визначаються заздалегідь в згортковій нейронній мережі; замість цього він автоматично дізнається значення кожного фільтра під час фази навчання.

Більше того, згорткова нейронна мережа використовує рівень об’єднання як метод нижньої вибірки. У вихідному шарі застосовується класифікатор, щоб використовувати функції високого рівня для завдання класифікації.

2.4 Рекурентні нейронні мережі

Рекурентні нейронні мережі (також відомі як РНМ) − це категорія штучних нейронних мереж, придатних для аналізу послідовних даних [9]. На відміну від згорткових штучних мереж, які призначені для роботи з даними топології, подібними до сітки, наприклад, зображеннями, РНМ є нейронними мережами, які мають спеціальні характеристики для роботи з послідовністю значень $x1,x2,...,\ xt$. Крім того, більшість РНМ [10] здатні обробляти послідовності змінної довжини. Головна ідея рекурентних мереж та деяких інших методів машинного навчання та статистичних методів полягає в тому, щоб ділитися параметрами на різних рівнях моделі, щоб розширити використання моделі для екземплярів даних з різними формами. Завдання спільного використання параметрів особливо важливо, коли певний елемент даних може з’являтися в кількох позиціях у послідовності. Цей метод оптимізації зазвичай призводить до значної економії пам’яті в моделях машинного навчання. Також можна використовувати РНМ для 2-вимірних просторових даних, таких як зображення. Ключова перевага використання рекурентних мереж перед звичайними нейронними мережами полягає в тому, що РНМ може обробляти послідовність даних, так що кожну вибірку можна вважати залежною від попередніх.

Як згадувалося, РНМ спеціалізовані для моделювання послідовностей [11], де існує сильна послідовна кореляція між зразками послідовностей. На кожному кроці часу РНМ використовує дані вхідні дані та інформацію, пов’язану з тим, що вже спостерігалося (тобто стан), щоб генерувати вихідні дані. Зауважте, що ця інформація передається через повторювані з’єднання між блоками, як показано на рисунку 2.4.

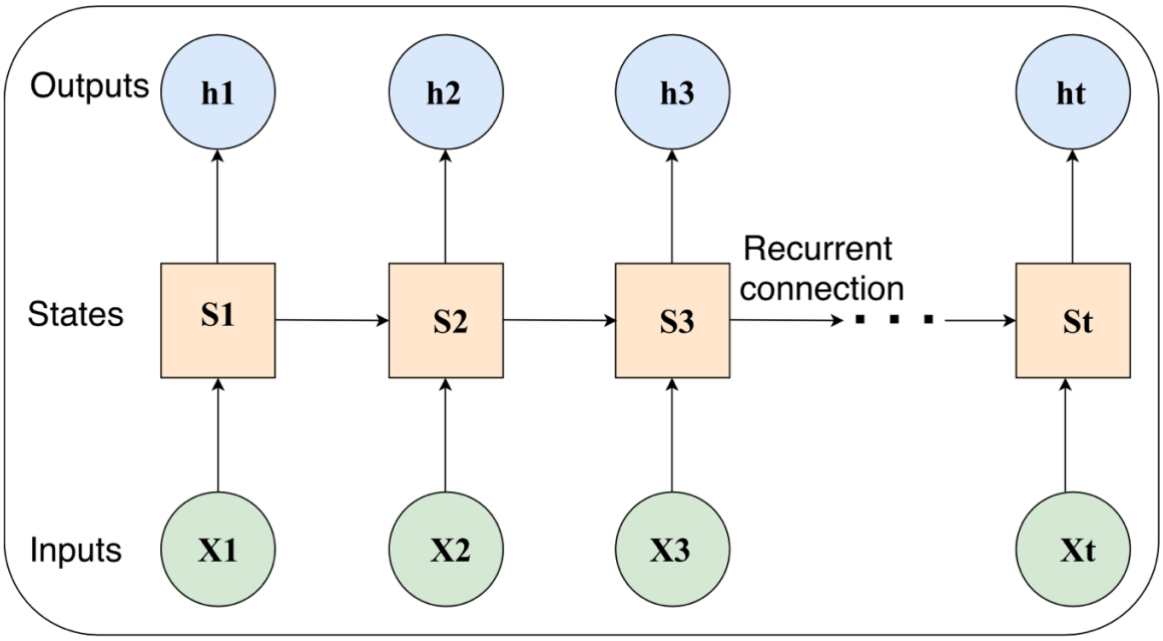

Рисунок 2.4 − Рекурента нейронна мережа

Припустимо, що ми маємо послідовність вхідних елементів **x** = ($x1$,$x2$,...,xt). За цього параметра РНМ проводить обчислення зображені в формулі 2.2 та формулі 2.3.

$$S_t = \sigma_S(W_x x_t + W_s s_{t-1} + b_s), \quad (2.2)$$

$$\mathrm{h}_t = \sigma_\mathrm{h}(W_\mathrm{h} s_t + b_\mathrm{h}), \quad (2.3)$$

де S_t − це стан РНМ на етапі часу t і він діє як одиниця пам'яті для РНМ. Щоб обчислити значення S_t, була розрахована функція вхідного значення в момент t (тобто xt) і попереднього стану РНМ, тобто $st-1$. Більше того, Wx і $W\mathrm{h}$ − це ваги, які необхідно засвоїти під час тренувального процесу, а bs і $b\mathrm{h}$ − упередження. У РНМ для оновлення ваг або навчання мережі використовується алгоритм зворотного поширення через час.

2.5 Моделі довгої короткочасної пам’яті (LSTM мережа)

РНМ може використовувати цикли для збереження градієнта останніх подій введення протягом тривалого часу. Це основна ідея моделі довгострокової пам’яті (LSTM). Ця функція потенційно важлива для широкого спектру застосувань, таких як розпізнавання мовлення, розпізнавання рукописного вводу, машинний переклад, створення рукописного тексту, підписи зображень та синтаксичний аналіз. LSTM було введено для вирішення двох серйозних проблем: зникнення градієнта та збільшення градієнта, у попередніх техніках. Точніше, за допомогою звичайних методів навчання на основі градієнта, таких як BPTT і повторюване навчання в реальному часі (RTRL), сигнали помилок можуть зменшуватися або збільшуватися, коли вони поширюються назад по моделі. Мережа LSTM пропонується вирішити проблеми зворотного потоку сигналів помилок шляхом введення ідеї використання набору шлюзів. LSTM успішно застосовується до багатьох проблем, таких як розпізнавання мовлення та класифікація тексту [12]. Графічна ілюстрація структури LSTM представлена на рисунку 2.5.

У цій структурі «forget gate» вирішує, яку інформацію зі стану клітинки буде забути, оскільки вони нерепрезентативні. Дійсно, «forget gate» приймає це рішення через сигмовидний шар. «forget gate» виконує наступну операцію представлену в формулі 2.4.

$$f_t = \sigma(W_{xf}X_t + W_{\mathrm{h}f}H_{t-1} + W_{cf} \odot C_{t-1} + b_f), \qquad (2.4)$$

У цьому виразі операція «⊙» − це Адамара або поелементний добуток, Ct представляє вихідні дані стану комірки, Ht позначає приховані стани. Forget gate пом’якшує зникнення градієнта і збільшення градієнта і значно покращує продуктивність LSTM, ніж RNN.

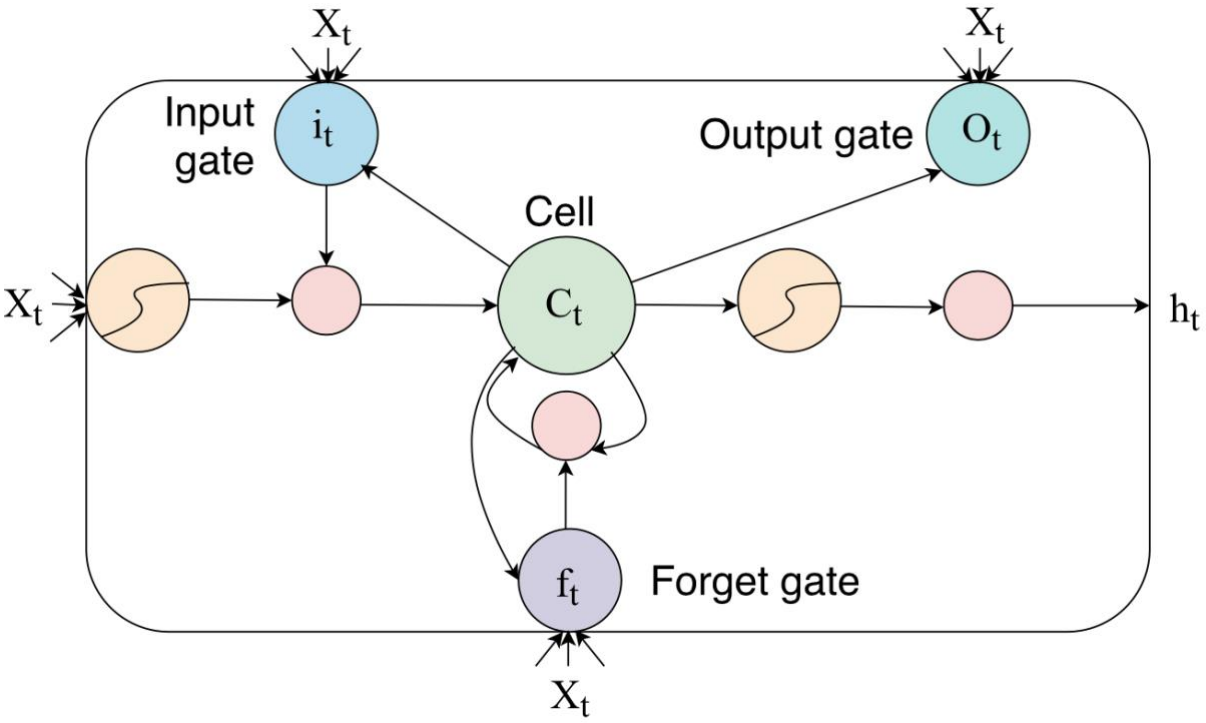

Рисунок 2.5 – Внутрішня структура LSTM

Іншою важливою функцією LSTM є вирішення, яка нова інформація повинна зберігатися в стані клітинки [12]. З цією метою вхідні ворота it (2.5) та (2.6) вирішують, яка інформація буде оновлена, і ця інформація забезпечить оновлення старого стану клітинки (тобто $Ct{-}1$).

$$it = \sigma(W_{xi}X_t + W_{\mathrm{h}i}H_{t-1} + W_{ci} \odot C_{t-1} + b_f), \quad (2.5)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot tan\mathrm{h}(W_{xc}X_t + W_{\mathrm{h}c}H_{t-1} + b_c), \quad (2.6)$$

І останній крок для LSTM − це вирішити, що має бути виведено, на основі стану клітинки. Це можна зробити за допомогою вихідних вентилів (2.7) та (2.8) (тобто ot), які вирішують, яка інформація про стан комірки буде надходити на вихід. Стан комірки також проходить через tanh, а потім помножується на вихідні вентилі.

$$o_t = \sigma(W_{xo}X_t + W_{\text{h}o}H_{t-1} + W_{co} \odot C_t + b_o), \quad (2.7)$$

$$H_t = o_t \odot tan\text{h}(C_t). \quad (2.8)$$

2.6 Глибоке навчання для моніторингу та аналізу трафіку в мережі

Методи машинного навчання, особливо алгоритми ГН, є одними з найпопулярніших методів обробки даних мережевого трафіку. Можливо, це пояснюється тим, що сучасні комунікаційні системи та мережі, наприклад, інтернет речей та стільникові мережі, мають відмінні характеристики, які відповідають алгоритмам ГН. Ці особливості включають генерацію великих даних, складність, мультимодальні дані, їх масштабність, зростаючу кількість протоколів у таких мережах тощо. Традиційні методи для моніторингу та аналізу мережевого трафіку мають свої проблеми; наприклад, вони неточні або сильно залежать від експертів . На відміну від традиційних методів, методи на основі ГН мають деякі переваги [13], які можна використовувати як методи МАМТ, перераховані нижче.

Моделі ГН не вимагають значних зусиль від людини і не залежать від вибору характеристик. Моделі ГН можуть використовувати різні репрезентативні шари та ефективні алгоритми для вилучення прихованих знань із величезної кількості даних про трафік без виділення характеристик. Ця перевага моделей ГН є дуже ефективною для методів МАМТ, оскільки більшість даних керування мережею не мічені або напівмарковані.

Моделі ГН здатні працювати з часово-просторовими даними [13], фіксуючи відповідні залежності. Більшість даних управління мережею, зібраних у вигляді наборів даних часових рядів, можна з високою точністю аналізувати моделями ГН. Розгортання точних та ефективних методів для різних додатків МАМТ має першорядне значення. Наприклад, точне прогнозування мобільного трафіку важливо для розподілу трафіку (наприклад, розподіл ресурсів на вимогу), економії енергії та аналітики

мобільності користувачів у стільникових мережах (наприклад, прогнозування руху).

Реалізація моделей DL за допомогою нових парадигм МН дозволяє ГН навчати свою модель окремо на кожній машині [13]. Це вважається великою перевагою, оскільки методи МАМТ повинні збирати інформацію керування мережею від різних машин до центральної точки. Використовуючи методи розподіленого машинного навчання, моделі ГН можна навчати окремо на кожній машині, що зменшує накладні витрати на мережу та можливість створення загрози безпеці та конфіденційності.

2.7 Проектування системи та інструменти для її розробки

Ми будемо проектувати систему, яка буде працювати за популярною в наш час клієнт-серверною архітектурою. Архітектура клієнт-сервер – архітектура комп’ютерної мережі, в якій багато клієнтів (віддалені процесори) запитують і отримують послугу від централізованого сервера (хост комп’ютера). Клієнтські комп’ютери забезпечують інтерфейс, який дозволяє користувачеві комп’ютера запитувати послуги сервера та відображати результати, які повертає сервер. Сервери чекають надходження запитів від клієнтів, а потім відповідають на них. В ідеалі сервер надає клієнтам стандартизований прозорий інтерфейс, щоб клієнтам не потрібно було знати про особливості системи (тобто апаратного та програмного забезпечення), яка надає послугу. Клієнти часто знаходяться на робочих станціях або на персональних комп’ютерах, тоді як сервери розташовані в інших місцях мережі [14], як правило, на більш потужних машинах. Ця обчислювальна модель особливо ефективна, коли клієнти і сервер мають різні завдання, які вони регулярно виконують. При обробці лікарняних даних, наприклад, на клієнтському комп’ютері може бути запущена прикладна програма для введення інформації про пацієнта, тоді як на серверному комп’ютері запущена інша програма, яка керує базою даних, в

якій постійно зберігається інформація. Багато клієнтів можуть отримувати доступ до інформації сервера одночасно [15], і в той же час клієнтський комп'ютер може виконувати інші завдання, наприклад надсилати електронну пошту. Оскільки і клієнтський, і серверний комп'ютери вважаються незалежними пристроями, модель клієнт-сервер повністю відрізняється від старої моделі мейнфрейма [16], в якій централізований мейнфрейм виконував усі завдання для пов'язаних з ним «німих» терміналів, які лише спілкувалися з центральним мейнфреймом. Приклад клієнт-серверної архітектури зображено на рисунку 2.6.

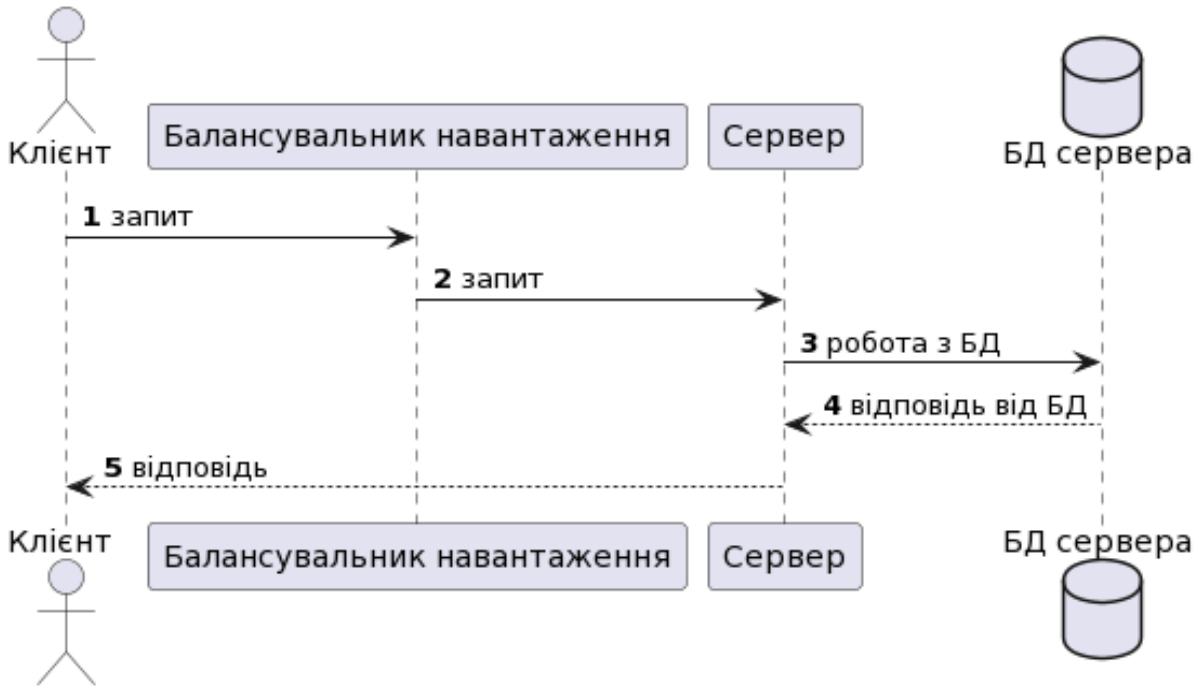

Рисунок 2.6 – Клієнт-серверна архітектура

В сучасних клієнт-серверверних системах ще використовують балансувальник навантаження. Балансувальник навантаження − це пристрій, який діє як зворотний проксі-сервер [17] і розподіляє мережевий або програмний трафік між кількома серверами. Балансувальники навантаження використовуються для збільшення ємності (одночасних користувачів) і надійності програм. Вони покращують загальну

продуктивність додатків, зменшуючи навантаження на сервери, пов’язані з керуванням і підтримкою сеансів програм і мережі, а також за рахунок виконання завдань, що стосуються програми.

Балансувальники навантаження, як правило, поділяються на дві категорії [18]: рівень 4 і рівень 7. Балансувальники навантаження рівня 4 діють на основі даних, знайдених у протоколах мережевого та транспортного рівня (IP, TCP, FTP, UDP). Балансувальники навантаження рівня 7 розподіляють запити на основі даних, знайдених у протоколах прикладного рівня, таких як HTTP.

Для того щоб оброблювати данні, їх для початку потрібно почати зберігати. Для цього ми дещо доповнимо схему з рисунку 2.5 додавши додаток для фіксації запитів. Модернізовану архітектуру можна побачити на рисунку 2.7.

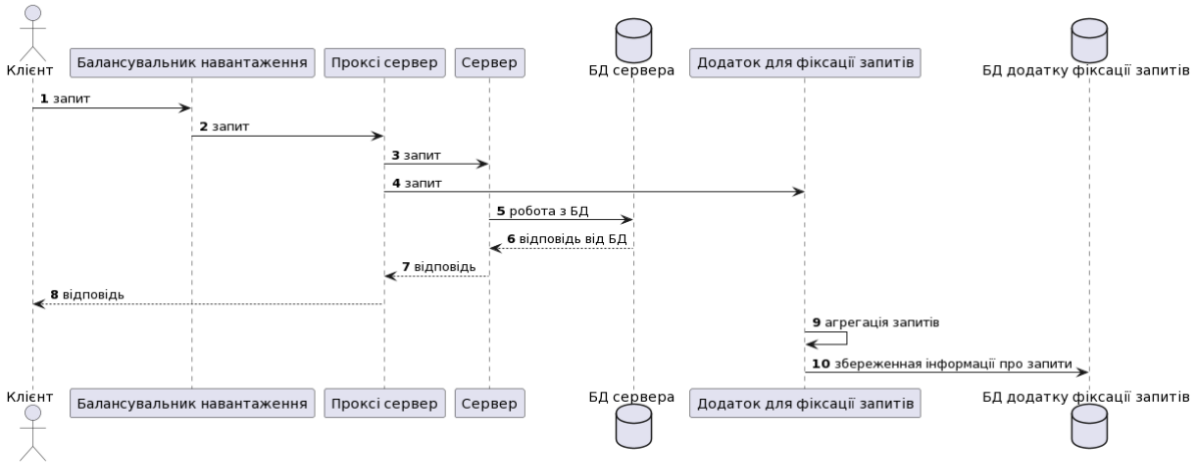

Рисунок 2.7 – Модернізована клієнт-серверна архітектура

В модернізованій схемі запит від балансувальника навантаження буде відправляти запит на проксі сервер, котрий в свою чергу буде відправляти запит як на основний сервер нашого додатку, так і на сервер додатку для фіксації запитів, така архітектура допоможе позбутися зайвого навантаження на основному сервері. В ролі проксі сервера можна використовувати Nginx. Nginx − це HTTP-сервер та зворотний проксі-сервер

[19], поштовий проксі-сервер, а також TCP/UDP проксі-сервер загального призначення, який може витримувати високе навантаження. Для реалізації додатку для фіксації запитів краще обрати Go. Go − це статично типізована, компільована мова програмування, розроблена в Google Робертом Гріземером, Робом Пайком і Кеном Томпсоном. Go виразний, лаконічний, чистий та ефективний. Його механізми паралельності дозволяють легко писати програми, які отримують максимальну віддачу від багатоядерних і мережевих машин [19], в той час як його нова система типу забезпечує гнучку і модульну конструювання програм. Go швидко компілюється в машинний код. Це швидка, статично типізована, скомпільована мова, яка схожа на динамічно типізовану, інтерпретовану мову. Go добре підходить для розробки додатків котрі працюють з великою кількістю мережевого трафіку, саме тому Go чудово підходить для реалізації додатку для фіксації запитів. Наш додаток буде зберігати інформацію про запити в базу даних. Для цих цілей ми будемо використовувати СУБД PostgreSQL.

PostgreSQL – це передова реляційна база даних з відкритим кодом корпоративного класу [20], яка підтримує запити SQL (реляційні) та JSON (нереляційні). Це дуже стабільна система управління базами даних, що підкріплюється більш ніж 20-річним розвитком громади, що сприяло її високій стійкості та цілісності. PostgreSQL використовується як основне сховище даних або сховище даних для багатьох веб, мобільних, геопросторових та аналітичних програм. Остання версія PostgreSQL 14.

Підтримка різноманітних даних PostgreSQL [20] та підтримка JSON дозволяють їй зв’язуватися з іншими сховищами даних, включаючи типи NoSQL, і виступати в якості об'єднаного центру для систем баз даних.

Так як запитів буде багато, то данні будуть агрегуватить в додатку та зберігатися в базу даних.

В базі даних буде зберігатися інформація про сервери та запити, які були зроблені до цього серверу. На рисунку 2.8 зображена схема БД.

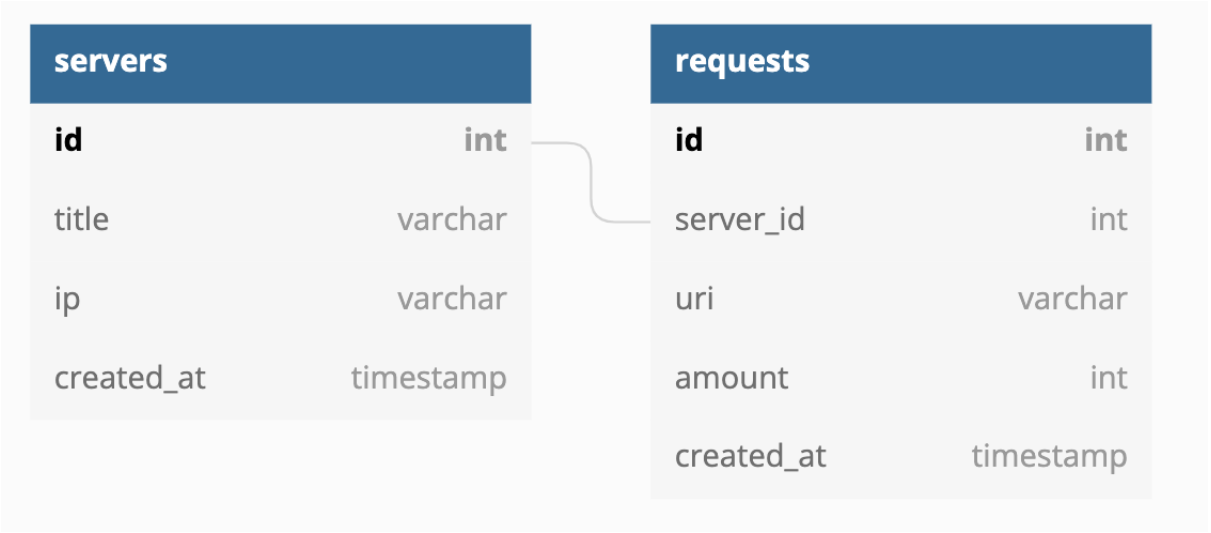

Рисунок 2.8 – Схема БД

В таблиці servers буде зберігатися наступна інформація про сервери:

1) Id – ідентифікатор сервера;
2) Title – умовна назва сервера;
3) Ip – ip адреса сервера;
4) Created_at – дата додавання запису в БД.

В таблиці requests буде зберігатися наступна інформація про запити до серверів:

1) Id – ідентифікатор запису; запису; запису; запису; запису; запису;
2) Server_id – зовнішній ключ до таблиці servers, шо відображає зв'язок один-до-багатьох;
3) URI – уніфікований ідентифікатор ресурсу;
4) Amount – кількість запитів до ресурсу з попередньої фіксації;
5) Created_at – дата додавання запису в БД.

Наступною частиною нашої архітектури буде виступати додаток для аналізу та передбачення навантаження на сервер. На рисунку 2.9 можна побачити продовження схеми архітектури на рисунку 2.7.

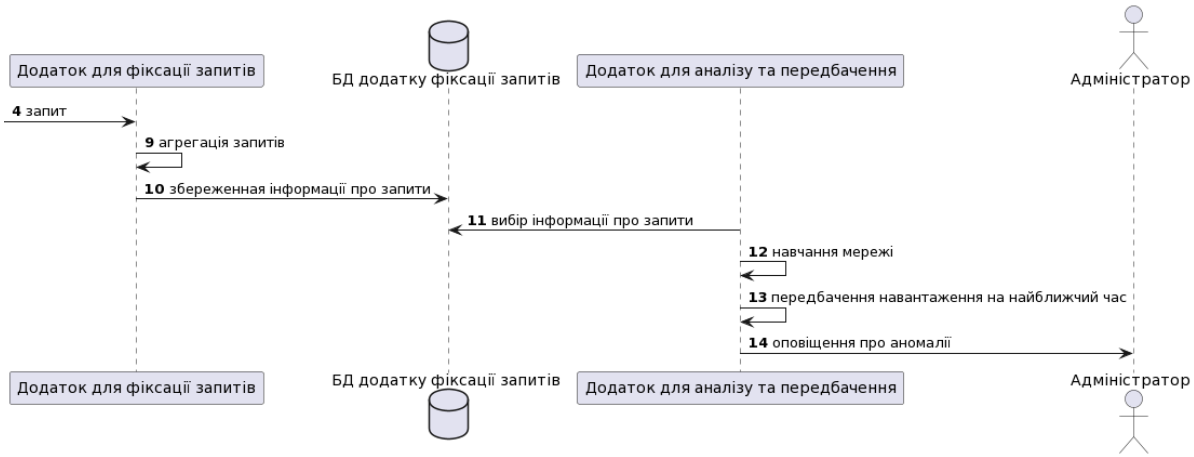

Рисунок 2.9 – Архітектура додатку для аналізу та прогнозування навантаження

Для реалізації додатку для аналізу та передбачення навантаження на сервери було обрано мову програмування Python. Python − високорівнева мова програмування загального призначення з динамічною строгою типізацією та автоматичним управлінням пам'яттю, орієнтована на підвищення продуктивності розробника, читання коду та його якості, а також на забезпечення переносимості написаних на ньому програм. Мова є повністю об'єктно-орієнтованою в тому плані, що все є об'єктами. Незвичайною особливістю мови є виділення блоків коду пробільними відступами. Синтаксис ядра мови мінімалістичний, рахунок чого практично рідко виникає необхідність звертатися до документації. Сама ж мова відома як інтерпретована і використовується в тому числі для написання скриптів. Недоліками мови є найчастіше нижча швидкість роботи і більш високе споживання пам'яті написаних ним програм порівняно з аналогічним кодом, написаним компільованими мовами, таких як C або C++.

Python є мультипарадигмальною мовою програмування, що підтримує імперативне, процедурне, структурне, об'єктно-орієнтоване програмування, метапрограмування та функціональне програмування. Завдання узагальненого програмування вирішуються рахунок динамічної типізації.

Аспектно-орієнтоване програмування частково підтримується через декоратори, повноцінніша підтримка забезпечується додатковими фреймворками. Такі методики як контрактне та логічне програмування можна реалізувати за допомогою бібліотек чи розширень. Основні архітектурні риси – динамічна типізація, автоматичне управління пам'яттю, повна інтроспекція, механізм обробки винятків, підтримка багатопоточних обчислень з глобальним блокуванням інтерпретатора (GIL), високорівневі структури даних. Підтримується розбиття програм на модулі, які можуть об'єднуватися в пакети.

Для того щоб організувати оточення проекту в одному місці, та щоб швидко приступити до розробки та в подальшому мати можливість легко розширювати як проект, так і команду розробників ми будемо використовувати Docker.

Docker – це відкрита платформа для розробки, доставки та запуску програм. Docker дозволяє нам відокремити свої програми від інфраструктури, щоб ви могли швидко доставити програмне забезпечення.

За допомогою Docker ми можем керувати інфраструктурою тими ж способами, якими керують програми. Скориставшись методологіями Docker для доставки, тестування та швидкого розгортання коду, ми можемо значно зменшити затримку між написанням коду та його запуском у виробництві.

Docker забезпечує можливість упаковки та запуску програми у слабко ізольованому середовищі (рисунок 2.10), званому контейнером. Ізоляція та безпека дозволяють одночасно запускати багато контейнерів на заданому хості. Контейнери мають невелику вагу, тому що їм не потрібно додаткового навантаження гіпервізора, а можуть працювати безпосередньо в ядрі хост-машини. Це означає, що ми можемо запустити більше контейнерів на певній комбінації апаратних засобів, ніж якщо ви використовували віртуальні машини [21]. Ми навіть можемо запускати

контейнери Docker в хост-машинах, які фактично є віртуальними машинами.

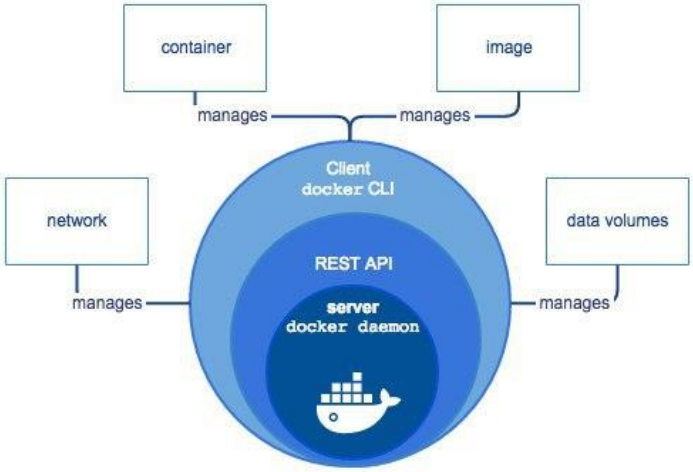

Рисунок 2.10 – Структура докеру

Для збереження різних версій коду та менеджменту стану кодової базими будемо використовувати Git. Git – це безкоштовна та з відкритим кодом розповсюджена система управління версіями, призначена для швидкого та ефективного управління всіма проектами від маленьких до дуже великих проектів.

3 РЕАЛІЗАЦІЯ АЛГОРИТМІВ ТА ПОРІВНЯЛЬНИЙ АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛЫДЖЕННЯ

3.1 Підготовка тестових даних одиничного запиту

Після того як данні будуть зібрані за допомогою нашого додатку для фіксації запиті, то ми будемо мати тестові данні для одного з API ендпоінтів, приклад яких зображено на рисунку 3.1. Фіксація забраних даних виконується кожну 1 хвилину.

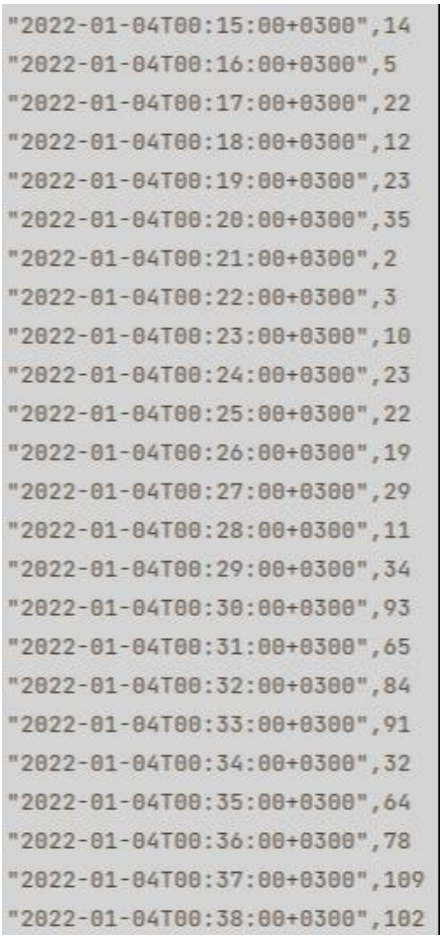

Рисунок 3.1 – Приклад даних

За допомогою простої програми на python візуалізуємо тестові данні. Код програми приведено на рисунку 3.2

```
import pandas
import matplotlib.pyplot as plt

if __name__ == '__main__':
    dataset = pandas.read_csv('/test/path/requests.csv', usecols=[1], engine='python')
    plt.plot(dataset)
    plt.show()
```

Рисунок 3.2 – Код для візуалізації тестових даних

Результатом роботи програми з рисунка 3.2 буде рисунок 3.3. На цьому рисунку по осі X відображено плин часу в хвилинах, а по осі У відображено кількість запитів отриманих сервером за хвилину.

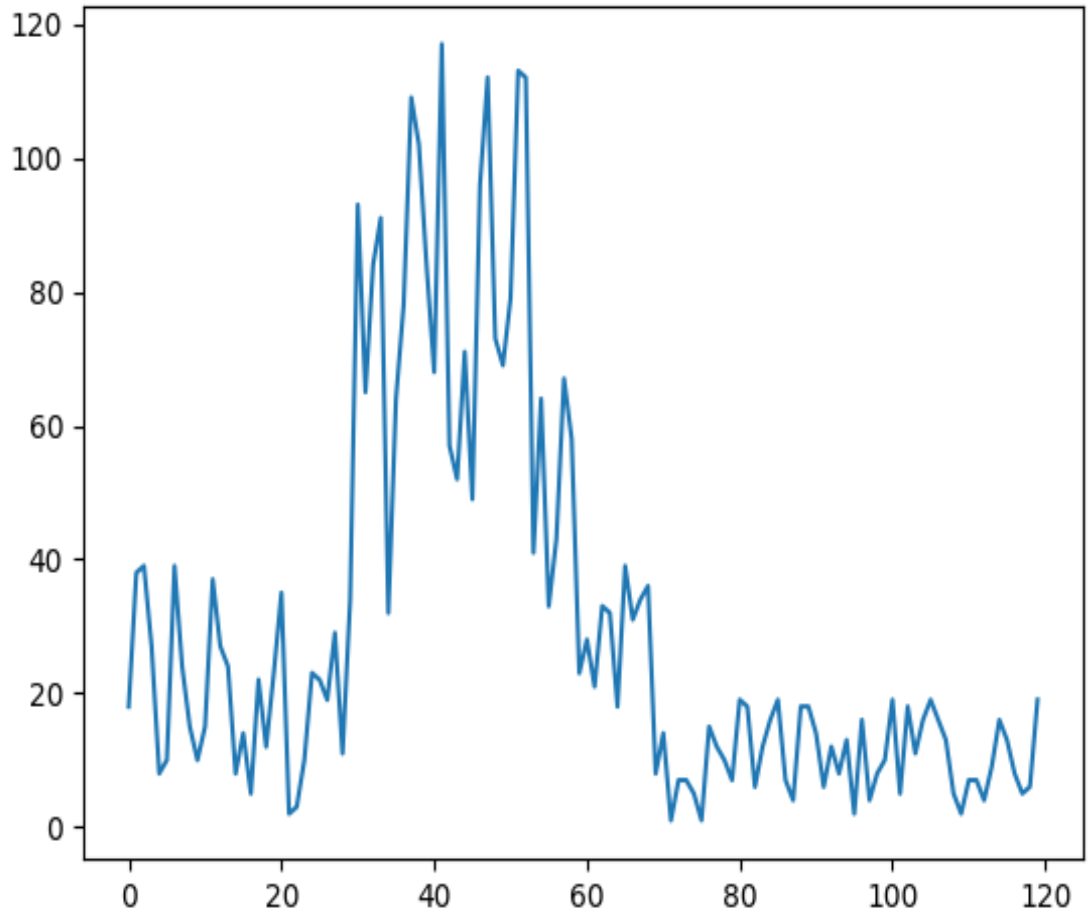

Рисунок 3.3 – Візуалізація тестових даних

3.2 Реалізація LSTM мережі для одиничного запиту

LSTM чутливі до масштабу вхідних даних, зокрема, коли використовуються сигмоподібні (за замовчуванням) або tanh функції активації. Може бути гарною практикою змінити масштаб даних до діапазону від 0 до 1, який також називається нормалізацією. Ми можемо легко нормалізувати набір даних за допомогою класу попередньої обробки MinMaxScaler з бібліотеки scikit-learn. Приклад коду зображено на рисунку 3.4.

```
# normalize the dataset
scaler = MinMaxScaler(feature_range=(0, 1))
dataset = scaler.fit_transform(dataset)
```

Рисунок 3.4 – Нормалізація даних

Після того, як ми змоделюємо наші дані та оцінимо навички нашої моделі на наборі навчальних даних, нам потрібно отримати уявлення про навички моделі на нових невидимих даних. Для звичайної задачі класифікації або регресії можна було б зробили це за допомогою перехресної перевірки. Для даних часових рядів важлива послідовність значень. Простий метод, який можна використати, − це розділити впорядкований набір даних на набори даних для підготовки та тестування.

Наведений нижче код на рисунку 3.5 обчислює індекс точки розщеплення та розділяє дані на навчальні набори даних із 67% спостережень, які ми можемо використовувати для навчання нашої моделі, залишаючи решту 33% для тестування моделі.

```
train_size = int(len(dataset) * 0.67)
test_size = len(dataset) - train_size
train, test = dataset[0:train_size, :], dataset[train_size:len(dataset), :]
```

Рисунок 3.5 – Розділення даних на тестову та тренувальну вибірку

Тепер ми можемо визначити функцію для створення нового набору даних. Реалізація функції приведена на рисунку 3.6. Функція приймає два аргументи: набір даних, який є масивом NumPy, який ми хочемо перетворити в набір даних, і look_back, який є кількістю попередніх кроків часу для використання як вхідних змінних для прогнозування наступного періоду часу − у цьому випадку за замовчуванням 1.

За замовчуванням буде створено набір даних, де X − кількість запитів у певний момент часу (t), а Y − кількість запитів у наступний раз (t + 1).

```
def create_dataset(dataset, look_back=1):
    dataX, dataY = [], []
    for i in range(len(dataset) - look_back - 1):
        a = dataset[i:(i + look_back), 0]
        dataX.append(a)
        dataY.append(dataset[i + look_back, 0])
    return numpy.array(dataX), numpy.array(dataY)
```

Рисунок 3.6 – Функцію для створення нового набору даних

Тепер використаємо цю функцію, щоб підготувати набори даних навчання та тестування моделі. Приклад коду зображено на рисунку 3.7.

```
look_back = 1
trainX, trainY = create_dataset(train, look_back)
testX, testY = create_dataset(test, look_back)
```

Рисунок 3.7 – Підготовка навчальної та тестової вибірки

Мережа LSTM очікує, що вхідні дані (X) будуть забезпечені конкретною структурою масиву у вигляді: [зразки, кроки часу, ознаки]. Наразі наші дані мають вигляд: [зразки, кроки часу], і ми формуємо проблему як один часовий крок для кожного зразка. Ми можемо перетворити підготовлені навчальні і тестові вхідні дані в очікувану структуру за допомогою numpy.reshape() так як це показано на рисунку 3.8.

```
trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))
```

Рисунок 3.8 – Підготовка навчальної та тестової вибірки

Тепер можна розробити та пристосувати нашу мережу LSTM для цієї проблеми. Мережа має видимий шар з 1 вхідним, прихований шар з 4 блоками або нейронами LSTM і вихідний шар, який робить передбачення єдиного значення. Для блоків LSTM використовується стандартна функція активації сигмовидної форми. Мережа навчається протягом 100 епох і використовується пакетний розмір 1. Приклад коду приведено на рисунку 3.9.

```
model = Sequential()
model.add(LSTM(4, input_shape=(1, look_back)))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='adam')
model.fit(trainX, trainY, epochs=100, batch_size=1, verbose=2)
```

Рисунок 3.9 – Реалізація моделі

Після того, як модель буде навчена, ми можемо оцінити продуктивність моделі на наборах даних для навчання та тестування. Це дасть нам точку порівняння для нових моделей.

Хочу зауважити, що ми інвертуємо передбачення перед обчисленням оцінок помилок, щоб гарантувати, що показники надані в тих самих одиницях, що й вихідні дані (кількість запитів за хвилину). Приклад коду зображено на рисунку 3.10.

```
# make predictions
trainPredict = model.predict(trainX)
testPredict = model.predict(testX)
# invert predictions
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform([trainY])
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform([testY])
# calculate root mean squared error
trainScore = math.sqrt(mean_squared_error(trainY[0], trainPredict[:, 0]))
print('Train Score: %.2f RMSE' % (trainScore))
testScore = math.sqrt(mean_squared_error(testY[0], testPredict[:, 0]))
print('Test Score: %.2f RMSE' % (testScore))
```

Рисунок 3.10 – Перевірка точності моделі

Тепер, ми можемо генерувати прогнози, використовуючи модель як для навчального набору даних, так і для тестового, щоб отримати візуальну індикацію навичок моделі.

Через те, як був підготовлений набір даних, ми повинні змістити передбачення так, щоб вони вирівнялися по осі x з вихідним набором даних. Після підготовки дані наносять на графік, показуючи вихідний набір даних синім кольором, прогнози для навчального набору даних помаранчевим, а прогнози для тестового набору даних зеленим. Код візуалізації результатів приведено на рисунку 3.11. Результат роботи коду візуалізації отриманих результатів приведено на рисунку 3.12. Видно, що модель виконала чудову роботу як на навчальній вибірці, так і тестовому наборів даних. Ми бачимо, що модель має середню похибку близько 20 запитів у наборі навчальних даних і близько 9 запитів у тестовому наборі даних.

```
trainPredictPlot = numpy.empty_like(dataset)
trainPredictPlot[:, :] = numpy.nan
trainPredictPlot[look_back:len(trainPredict) + look_back, :] = trainPredict
# shift test predictions for plotting
testPredictPlot = numpy.empty_like(dataset)
testPredictPlot[:, :] = numpy.nan
testPredictPlot[len(trainPredict) + (look_back * 2) + 1:len(dataset) - 1, :] = testPredict
# plot baseline and predictions
plt.plot(scaler.inverse_transform(dataset))
plt.plot(trainPredictPlot)
plt.plot(testPredictPlot)
plt.show()
```

Рисунок 3.11 – Код візуалізації результатів

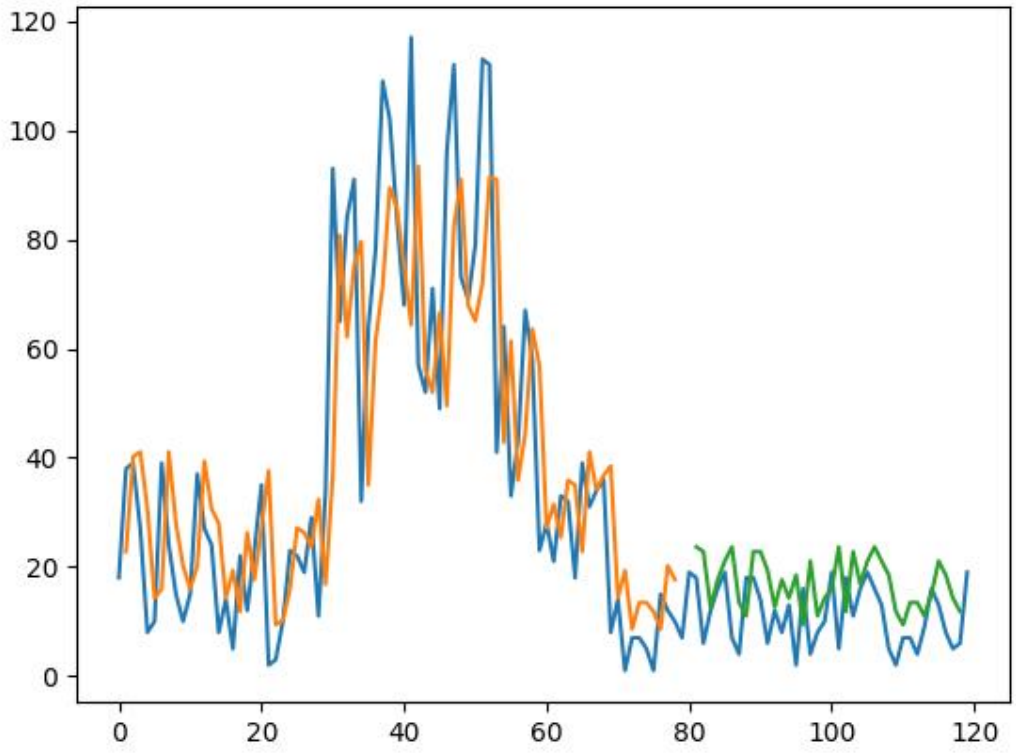

Рисунок 3.12 – Результат візуалізації

Ми також можемо сформулювати проблему так, щоб можна було використовувати кілька останніх кроків часу для прогнозування наступного кроку часу. Це називається вікном, а розмір вікна є параметром, який можна налаштувати для кожної проблеми. Наприклад, враховуючи поточний час (t), ми хочемо передбачити значення наступного разу в послідовності (t+1),

ми можемо використовувати поточний час (t), а також два попередні моменти (t-1 і t-2) як вхідні змінні.

Якщо формулювати як проблему регресії, то вхідні змінні – t-2, t-1, t, а вихідна – t+1.

Функція create_dataset(), яку ми створили раніше, дозволяє нам створити таке формулювання проблеми часових рядів, збільшивши аргумент look_back з 1 до 3.

В такій конфігурації ми бачимо, що модель має середню похибку близько 20 запитів у наборі навчальних даних і близько 6 запитів у тестовому наборі даних. Графічну візуалізацію роботи моделі можна побачити на рисунку 3.13.

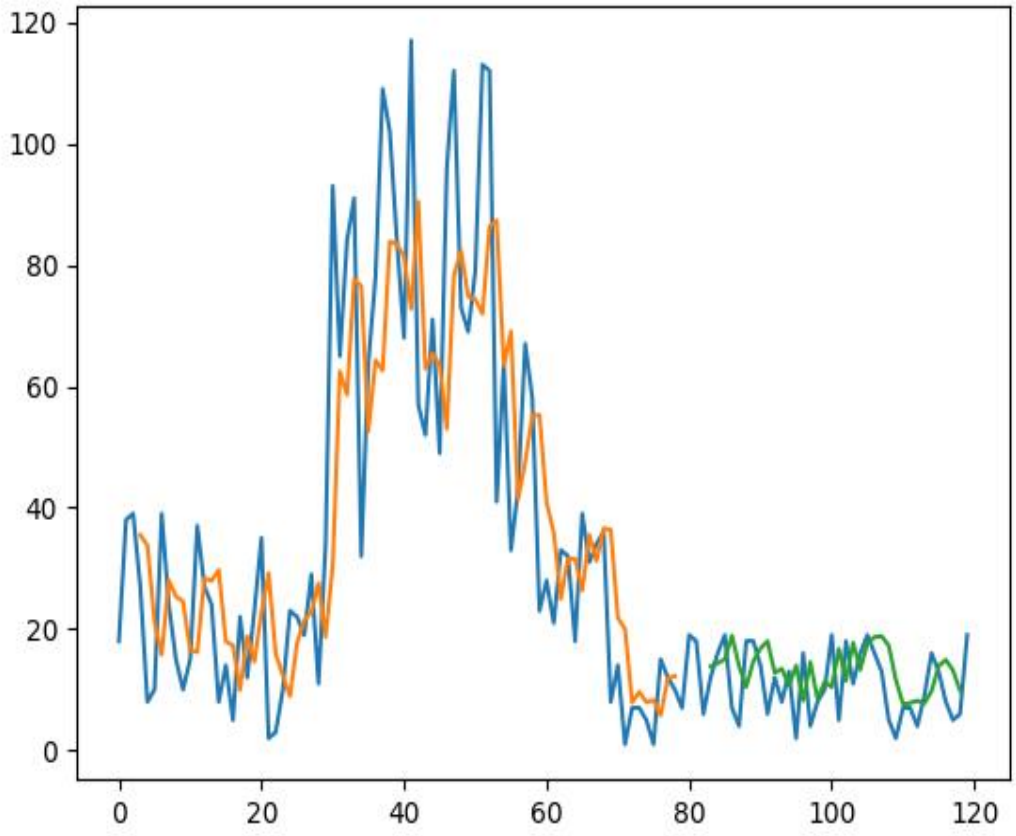

Рисунок 3.13 – Результат візуалізації

Мережа LSTM має пам’ять, яка здатна запам’ятовувати довгі послідовності. Зазвичай стан мережі скидається після кожного навчального пакету під час встановлення моделі, а також кожного виклику model.predict() або model.evaluate().

Ми можемо отримати кращий контроль над тим, коли внутрішній стан мережі LSTM очищається в Keras, роблячи шар LSTM «з статусом». Це означає, що він може створювати стан протягом усієї послідовності навчання і навіть підтримувати цей стан, якщо це необхідно для прогнозування.

Це вимагає, щоб навчальні дані не перемішувалися під час встановлення мережі. Він також вимагає явного скидання стану мережі після кожного впливу навчальних даних (епохи) за допомогою викликів model.reset_states() . Це означає, що ми повинні створити власний зовнішній цикл епох і всередині кожної епохи викликати model.fit() і model.reset_states(). Приклад коду зображено на рисунку 3.14.

```
for i in range(100):
    model.fit(trainX, trainY, epochs=1, batch_size=batch_size, verbose=2, shuffle=False)
    model.reset_states()
```

Рисунок 3.14 – Модифіковане навчання моделі

Коли створюється шар LSTM, параметр для визначення стану має бути встановлений True , і замість того, щоб вказувати вхідні розміри, ми повинні жорстко запрограмувати кількість зразків у пакеті, кількість часових кроків у вибірці та кількість функцій за час. крок, встановивши параметр batch_input_shape. Цей самий розмір необхідно використовувати при прогнозуванні. В такій конфігурації ми бачимо, що модель має середню похибку близько 15 запитів у наборі навчальних даних і близько 8 запитів у тестовому наборі даних. Графічну візуалізацію роботи моделі можна побачити на рисунку 3.15.

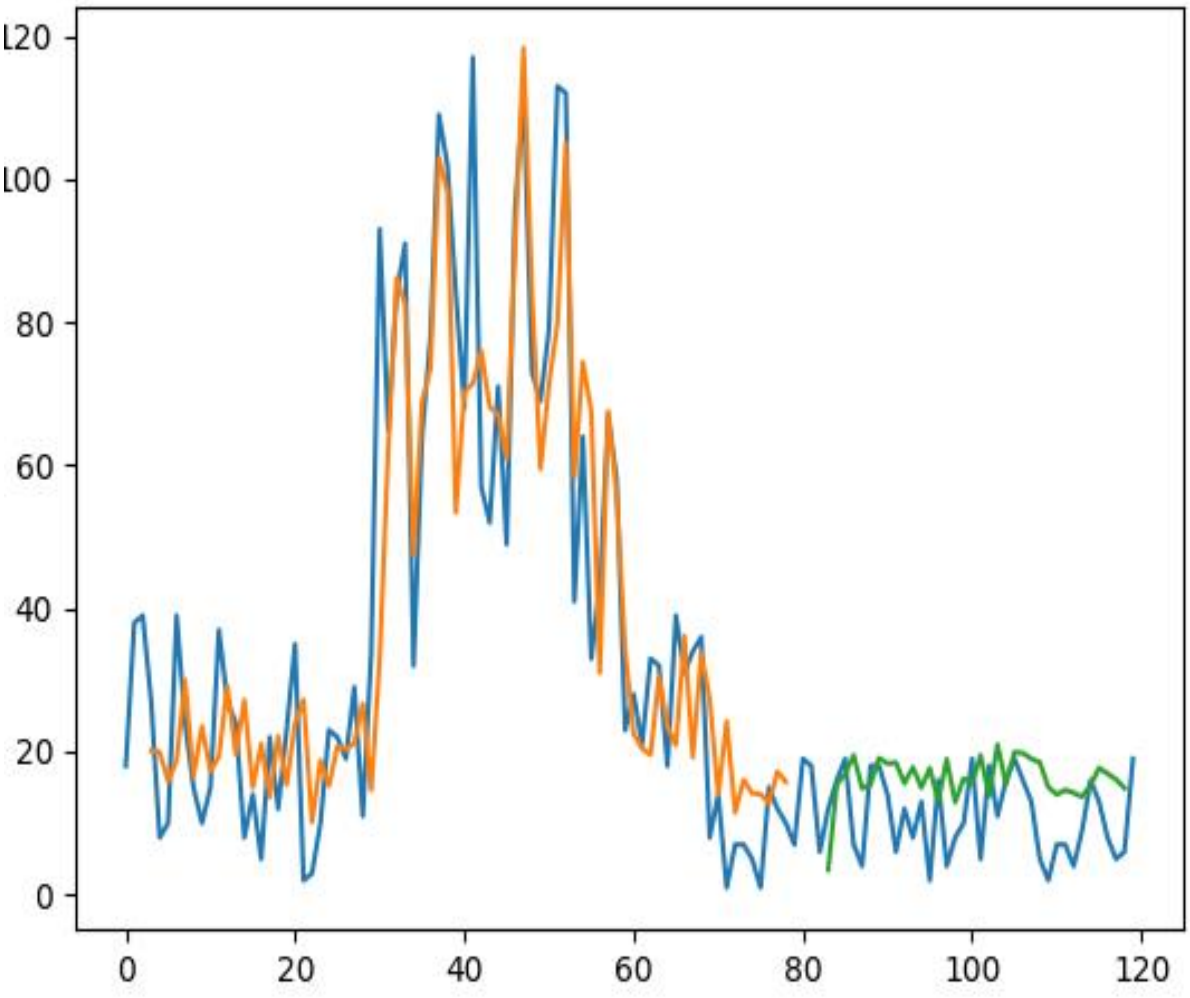

Рисунок 3.15 – Результат візуалізації

3.3 Підготовка тестових даних групи запитів

В цьому розділі ми розглянемо групу запитів. Фіксація забраних даних виконується кожну 1 хвилину. Ми маємо 5 різних запитів для яких проводилася фіксація. На рисунку 3.16 зображено приклад тестових даних, які будуть використовуватися в моделі. На рисунку 3.17 зображено візуалізація отриманих даних.

```
"2022-01-04T02:02:00+0300",82,3,49,22,8
"2022-01-04T02:03:00+0300",78,5,48,20,2
"2022-01-04T02:04:00+0300",68,4,54,23,1
"2022-01-04T02:05:00+0300",68,6,43,24,7
"2022-01-04T02:06:00+0300",80,10,41,15,2
"2022-01-04T02:07:00+0300",70,18,36,18,7
"2022-01-04T02:08:00+0300",65,7,36,19,4
"2022-01-04T02:09:00+0300",76,7,48,17,3
"2022-01-04T02:10:00+0300",70,18,40,16,6
"2022-01-04T02:11:00+0300",65,25,42,15,3
"2022-01-04T02:12:00+0300",64,2,33,24,3
"2022-01-04T02:13:00+0300",75,5,47,16,5
```

Рисунок 3.16 – Приклад даних для групи запитів

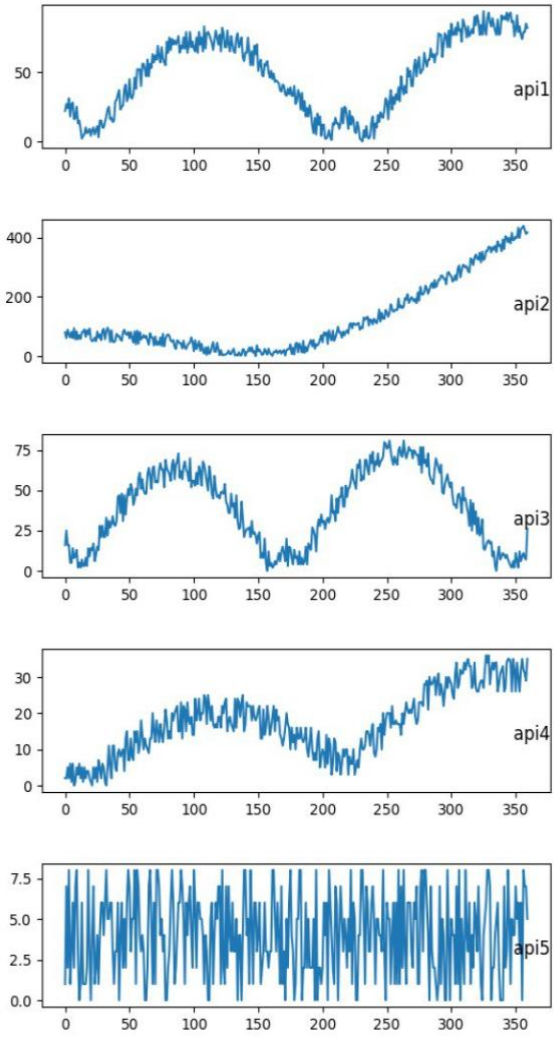

Рисунок 3.17 – Візуалізація групи запитів

3.4 Реалізація LSTM мережі для групи запитів

Реалізацію LSTM мережі для групи запитів почнемо з того що підготуємо данні: нормалізуємо, приведемо часові ряди до формату роботи з мережею, розділимо вибірку на навчальну та тестову. Код для реалізації описаного вище фрагменту приведено на рисунку 3.18.

```
values = dataset.values
values = values.astype('float32')
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = scaler.fit_transform(values)
n_h = 5
n_f = 2
reframed = series_to_supervised(scaled, n_h, 1)
print(reframed.shape)

values = reframed.values
n_train_hours = 300
train = values[:n_train_hours, :]
test = values[n_train_hours:, :]
n_obs = n_h * n_f
train_X, train_y = train[:, :n_obs], train[:, -n_f]
test_X, test_y = test[:, :n_obs], test[:, -n_f]
print(train_X.shape, len(train_X), train_y.shape)
train_X = train_X.reshape((train_X.shape[0], n_h, n_f))
test_X = test_X.reshape((test_X.shape[0], n_h, n_f))
```

Рисунок 3.18 – Приклад коду

Наведений приклад на рисунку 3.18 розбиває набір даних на набори навчальної і тестової вибірки, а потім розбиває набори навчальної і тестової

вибірки на вхідні та вихідні змінні. Нарешті, вхідні дані (X) змінюються в 3D-формат, який очікують LSTM, а саме [зразки, часові кроки, функції].

Тепер ми можемо створити та навчити нашу модель LSTM. Ми визначимо LSTM з 20 нейронами на першому, другому та третьому прихованих шарах та 1 нейроном у вихідному шарі для прогнозування навантаження. Модель буде розрахована на 150 епох навчання з розміром пакету 72. Також, ми будемо слідкувати за похибкою навчання шляхом завдання validation_data у функції fit(). Реалзізація моделі зображена на рисунку 3.19.

```
# design network
model = Sequential()
model.add(LSTM(20, input_shape=(train_X.shape[1], train_X.shape[2]), return_sequences=True))
model.add(LSTM(20, input_shape=(train_X.shape[1], train_X.shape[2]), return_sequences=True))
model.add(LSTM(20, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))
model.compile(loss='mean_squared_error', optimizer='rmsprop')
# fit network
history = model.fit(train_X, train_y, epochs=150, batch_size=72,
                    validation_data=(test_X, test_y), verbose=2,
                    shuffle=False)
```

Рисунок 3.19 – Приклад коду реалізації моделі

Після того як модель була навчена, ми можемо спрогнозувати весь набір тестових даних. Об’єднуємо прогноз із тестовим набором даних та інвертуємо масштабування. Ми також інвертуємо масштабування тестового набору даних із очікуваними числами. Візуалізація результату роботи передбачення моделі зображено на рисунку 3.20. Використовуючи прогнози та фактичні значення в початковій шкалі, ми можемо розрахувати оцінку помилки для моделі. У цьому випадку ми обчислюємо середню квадратичну помилку (RMSE), яка дає помилку в тих самих одиницях, що й сама змінна. В випадку з нашими тестовими даними групи запитів ми отримали похибку 8.8 запитів.

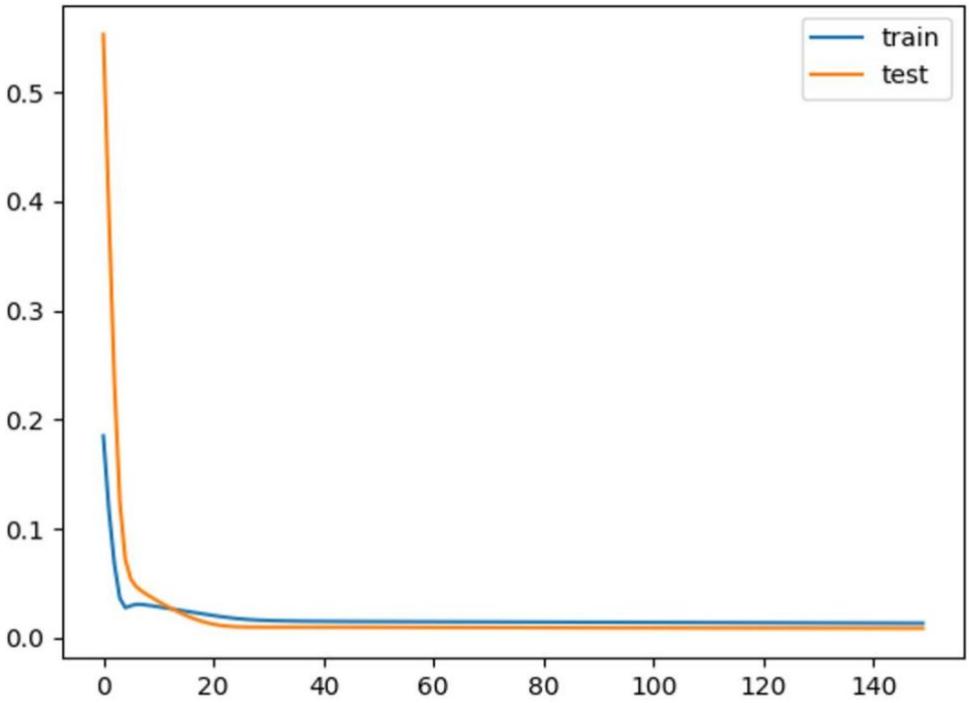

Рисунок 3.20 – Візуалізація результатів

3.5 Аналіз результатів дослідження

В ході проведеного дослідження було реалізовано декілька варіацій LSTM мережі для передбачення запитів:

1) LSTM мережа для одиничного запиту;
2) LSTM мережа з використанням методу вікна;
3) LSTM мережа з використанням пам’яті;
4) LSTM мережа для групи запитів.

В базовій реалізації мережі для одиночного запиту середньоквадратична помилка склала дев'ять запитів. В реалізації мережі з використанням методу вікна для одиночного запиту середньоквадратична помилка склала шість запитів. В реалізації мережі з використанням пам’яті для одиночного запиту середньоквадратична помилка склала вісім запитів. В реалізації мережі для групи запиті середньоквадратична помилка склала теж вісім запитів.

Аналізуючи отримані результати можна сказати, що розроблений метод є цілком працездатним та ефективним, але за рахунок того що він оброблює інформацію тільки одного запиту, то він не може враховувати неявні залежності між іншими запитами що в певних ситуаціях буде призводити до генерації дуже не точних прогнозів. Варіант з використанням пам’яті має схожі результати. Варіант з використанням методу вікна дав найкращий результат, але все ще він не може дати можливість відстежувати сховану залежність між іншими запитами. Використавши всі плюси реалізацій з одиничним запитом ми можемо сказати що найкращим вибором буде останній варіант реалізації мережі, тому що в такому варіанті реалізації робота мережі може відстежувати приховані залежності. Для отримання більш точних результатів потрібно зібрати більше даних.

Виходячи з аналізу отриманих результатів можна сказати що найкраще для вирішення проблеми передбачення навантаження мережі являється LSTM з використанням методу вікна для групи запитів.

ВИСНОВКИ

Результатом кваліфікаційної роботи є спроектована система моніторингу та аналізу мережевого трафіку, яка базується на використанні розробленого в роботі методу аналізу та прогнозування кількісної інформації про мережевий трафік.

Було проаналізовано готові рішення та виділено їх основні недоліки а саме труднощі при роботі з часовими даними. В ході проектування системи була описана архітектура системи, обрана СУБД PostgreSql та описана структура бази даних, а саме головні сутності та зв’язки між ними. Також порівняно різні типи нейронних мереж, було описано їх сильні та слабкі сторони, та в ході аналізу було обрано найбільш придатну для розв'язання завдання нейронну мережу LSTM. В архітектурі системи було описано взаємодію основних вузлів: балансувальника, проксі-сервера, головного серверу додатку, бази даних головного додатку, додаток для агрегації запитів, база даних додатку для агрегації запитів та додаток для моніторингу та аналізу запитів.

Для розробки також використовувався різноманітний інструментарій, а саме: Docker − для налаштування оточення; Pip − для управління залежностями; Git − для зберігання кодової бази проекту; PyCharm − для безпосередньої розробки та управління проектом.

Перспективним є реалізація додатку для агрегації запитів та синтез моделі рекурентної нейронної мережі з довгою короткочасною пам’яттю для створення додатку для прогнозування навантаження веб-орієнтованих високонавантажених систем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Mitchell T. For pioneering contributions and leadership in the methods and applications of machine learning. National Academy of Engineering, 2011. С. 34–42

2. Mitchell T. Machine Learning Definition Science/Engineering/Math. National Academy of Engineering, 1997. С. 23–55

3. A Survey on Big Data for Network Traffic Monitoring and Analysis. URL: https://ieeexplore.ieee.org/document/8789667 (дата звернення 10.04.2022).

4. Vrizlynn Vrizlynn L. L. Thing. IEEE 802.11 Network Anomaly Detection and Attack Classification: A Deep Learning Approach. URL: https://ieeexplore.ieee.org/document/7925567 (дата звернення 14.04.2022).

5. Shahraki A., Taherkordi A., Haugen Ø., Eliassen F. A survey and future directions on clustering: From WSNs to IoT and modern networking paradigms, 2020. С. 2242 – 2274.

6. Lane T., Brodley C. An application of machine learning to anomaly. Baltimore, USA, 1997. С. 366–380.

7. Хайкін С. Нейронні мережі. Повний курс, 2019. 1104 с.

8. Корнієнко О. О., Петров К. Е. Методи моніторингу та аналізу мережевого трафіку в веб-орієнтованих системах: матеріали Міжнар наук. конф., м. Харків, Баку, Жиліна, 27 квіт. 2022 р. С. 18.

9. Gerardus B. Recurrent neural network: Real Life Actions, 2018. 132 с.

10. Rogerson J. Theory, Concepts and Methods of Recurrent Neural Networks and Soft Computing, 2015. 304 с.

11. Gulli A. Deep Learning with Keras: Implementing deep learning models and neural networks with the power of Python, 2018. 318 с.

12. Bianchi F. Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis (SpringerBriefs in Computer Science), 2017. 81 с.

13. Demuth H. Neural Network Design, 2014. 800 c.

14. Rubens R. Virtual Firewall in Cloud, 2017. 90 c.

15. Lauret A. The Design of Web APIs, 2019. 400 c.

16. Robert M. Clean Architecture: A Craftsman's Guide to Software Structure and Design, 2017. 432 c.

17. McGuerty J. Network Field Survival Guide: The Way of the Packet, 2018. 210 c.

18. Soto I. Multimedia Networking Technologies, Protocols, & Architectures (Artech House Communications and Network Engineering), 2019. 300 c.

19. Forshaw J. Attacking Network Protocols: A Hacker's Guide to Capture, Analysis, and Exploitation, 2017. 336 c.

20. Hsu L. PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database, 2017. 314 c.

21. Edelman J. Network Programmability and Automation: Skills for the Next-Generation Network Engineer, 2018. 584 c.