Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Моделі глибокого навчання для прогнозування часових рядів сейсмологічних даних

(тема)

Виконав:

студент II курсу, групи СПм-22-5

Тимошенко Д.О.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

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

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Іващенко Г.С.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

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

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Тимошенко Дар’ї Олександрівні
(прізвище, ім’я, по батькові)

1. Тема роботи Моделі глибокого навчання для прогнозування часових рядів сейсмологічних даних

затверджена наказом по університету від “ 01 ” квітня 2024 р. № 257 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи

1) документація мови програмування Python 3;
2) операційна система Windows 10;
3) середовище розробки Jupyter Notebook;
4) набір даних про сейсмічну активність вулканів.

4. Перелік питань, що потрібно опрацювати у роботі

1) аналіз предметної області;
2) аналіз існуючих досліджень;
3) аналіз методів прогнозування часових рядів;
4) програмна реалізація;
5) аналіз результатів;
6) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Слайд-презентація – 18 слайдів

6. Консультанти розділів роботи (заповнюється 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	Термін виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	02.04.24-08.04.24	
2	Вибір методики дослідження	09.04.24-16.04.24	
3	Вибір інструментальних засобів	17.04.24-22.04.24	
4	Розробка алгоритмічного забезпечення	23.04.24-06.05.24	
5	Проведення експериментів	07.05.24-23.05.24	
6	Оформлення матеріалів кваліфікаційної роботи	24.05.24-03.06.24	
7	Подання кваліфікаційної роботи керівникові	04.06.24-07.06.24	
	та її попередній захист		
8	Подання кваліфікаційної роботи на	08.06.24-12.06.24	
	рецензування		

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Іващенко Г.С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 75 с., 24 рис., 11 табл., 3 дод., 44 джерела.

ЧАСОВІ РЯДИ, ШТУЧНІ НЕЙРОННІ МЕРЕЖІ, ГЛИБОКІ НЕЙРОНІ МЕРЕЖІ, CNN, LSTM, ПРОГНОЗУВАННЯ, PYTHON, MSE, PANDAS, NUMPY.

Метою кваліфікаційної роботи є дослідження та розробка моделей глибокого навчання для прогнозування часових рядів сейсмологічних даних.

У ході виконання кваліфікаційної роботи було проведено аналіз існуючих методів прогнозування сейсмічної активності. Було вивчено декілька різних архітектур глибоких нейронних мереж, а саме моделей LSTM та CNN.

Дослідження проводилися з використанням актуальних даних щодо сейсмічної активності вулканів, що були зібрані Італійським національним інститутом геофізики та вулканології.

Розроблений програмний продукт дозволяє виконувати прогнозування сейсмічної активності вулканів для найпершого виявлення передвісника, що надасть інформацію про час майбутніх виверженьвулканів.

ABSTRACT

Master’s thesis: 75 pages, 24 figures, 11 tables, 3 appendices, 44 sources.

TIME SERIES, ARTIFICIAL NEURAL NETWORKS, DEEP NEURAL NETWORKS, CNN, LSTM, FORECASTING, PYTHON, MSE, PANDAS, NUMPY.

The major goal of this thesis is research and development of deep learning models for forecasting time series of seismological data.

In the process of the qualification work an analysis of the existing methods of forecasting seismic activity was carried out. Several different deep neural network architectures have been studied, namely LSTM and CNN models.

The research was carried out using current data on the seismic activity of volcanoes, which were collected by the Italian National Institute of Geophysics and Volcanology.

The developed software product allows forecasting the seismic activity of volcanoes for the earliest detection of a harbinger that will provide information about the time of future volcanic eruptions.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШНМ – штучна нейронна мережа

AMSGrad – оцінка адаптивного моменту з квадратичним градієнтним усередненням (англ., Adaptive Moment Estimation with Squared Gradient Averaging)

ARIMA – авторегресійне інтегроване ковзне середнє (англ., autoregressive integrated moving average)

CNN – згорткова нейронна мережа (англ., Convolutional Neural Network)

DNN – глибока нейронна мережа (англ., Deep Neural Network)

FCNN – повністю зв’язана нейронна мережа (англ., Fully Connected Neural Network)

GAN – генеративно-змагальна мережа (англ., Generative Adversarial Network)

LDA – лінійний дискримінантний аналіз (англ., Linear Discriminant Analysis)

LSTM – нейронна мережа довгої короткострокової пам'яті (англ., Long Short-Term Memory)

LTA – довгострокове середнє (англ., Long-Term Average)

MLP – багатошаровий персептрон (англ., Multilayer Perceptron)

MSE – середньоквадратична помилка (англ., Mean Squared Error)

STA – короткострокове середнє (англ., (Short-Term Average)

SVM – машина опорних векторів (англ., Support Vector Machine)

TCN – часова згорткова мережа (англ., Temporal Convolutional Network)

ВСТУП

На сьогоднішній день прогнозування часу до виверження вулканів є актуальним завданням, від точності результатів якого залежить велика кількість людських життів.

Наразі більшість запропонованих моделей прогнозування виверження вулканів зосереджено на певній формі оцінки сейсмічної небезпеки. Такі моделі виявилися дуже корисними для оцінки ініціюючих землетрусів, але вони часто зазнають невдачі при прогнозуванні майбутніх подій, через неможливість ідентифікування безлічі змінних, які спричиняють землетрус. Закономірності сейсмічності важко інтерпретувати. У випадку активних вулканів сучасні підходи передбачають виверження на кілька хвилин наперед, проте не дають змоги зробити довгострокові прогнози.

Через недостатню точність поширених методів прогнозування виверження вулканів актуальним є використання нових підходів, що використовують джерела інформації, які зазвичай надають дані у вигляді часових рядів. Відповідно, аналіз часових рядів може бути доцільним при вирішенні завдань прогнозування сейсмічної активності. Для прогнозування часових рядів перспективним є використання глибоких нейронних мереж, які мають більшу точність при вирішенні проблем, у яких застосовуються статистичні моделі навчання та де важко отримати точне математичне формулювання.

У даній роботі прогнозування часових рядів розглянуто в контексті прогнозування сейсмічної активності вулканів. В якості навчального матеріалу для нейронних мереж використані дані, що були зібрані Італійським національним інститутом геофізики та вулканології впродовж 24-годинного моніторингу сейсмічної та активної вулканічної активності по всій країні.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність завдання прогнозування виверження вулканів

Виверження вулканів є одними з найбільш руйнівних та непередбачуваних природних катаклізмів, які мають величезний фізичний, психологічний та економічний вплив на населення в усьому світі. При своєчасному попередженні райони навколо вулкана можуть бути безпечно евакуйовані перед їх знищенням. На сьогодні актуальною проблемою залишається обробка даних, що можуть дати можливість виявити сейсмічну активність вулканів на ранніх її етапах. Щоб отримати краще наближення або додаткову інформацію про те, де і коли може відбутися виверження, необхідно застосовувати прогнозування на основі вже наявних даних про сейсмічну активність вулканів.

Прогнозуванням вважається процес вироблення нового знання про майбутнє на основі накопичених даних та знань про минуле й сьогодення [1].

В основі прогнозування лежать три взаємодоповнюючи джерела інформації про майбутнє:

- оцінка перспектив розвитку майбутнього стану прогнозованого явища на основі досвіду [2], найчастіше за допомогою аналогії з достатньо подібними явищами та процесами;
- екстраполяція тенденцій [2], закономірності розвитку яких у минулому та сьогоденні добре відомі;
- моделювання майбутнього стану того чи іншого явища чи процесу. Суть моделювання полягає в представлені спрощеного аналогу реального об’єкта, який ураховує лише суттєві властивості об’єкта, необхідні для розв’язування конкретного завдання [2].

Моделювання майбутнього стану виверження вулкану полягає в заміні реального процесу певною математичною конструкцією, що відтворює

основні риси цього явища за допомогою абстракції від другорядних аспектів. Для цього використовуються дані у вигляді часових рядів, на основі яких будується математична модель ряду, що містить сейсмологічні дані. Ця модель дозволяє пояснити поведінку ряду та здійснити прогноз на майбутні періоди, використовуючи минулі значення, зібрані у часовому ряді.

1.2 Часові ряди

Часовий ряд – це послідовність спостережень за певним параметром у різні моменти часу [3]. Як приклад часових рядів можна навести дані про продаж продукції протягом певного дня або дані про температуру на певній території в різні часові рамки.

Ряд складається з двох частин – сигналу та шуму (рисунок 1.1). Сигнал – це математично залежні змінні [3], які можна проаналізувати та прибрати з часового ряду. Шум складається з випадкових значень, частіше за все нормального розподілу, та візуально його відображення схоже на графічне відображення звукового шуму [3]. Реальні дані є поєднанням шуму та сигналу і найчастіше за все низька якість моделі залежить саме від неможливості відтворити шум.

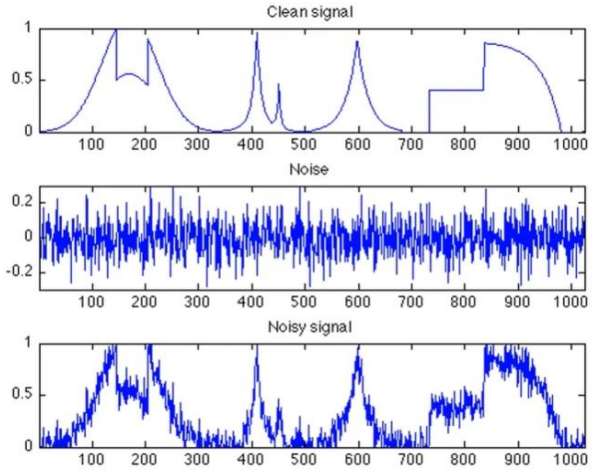

Рисунок 1.1 – Чистий сигнал, шум та зашумлений сигнал

Основними складовими часового ряду є тренд і сезонний компонент.

Тренд є систематичним компонентом часового ряду, що може змінюватися в часі [4]. Трендом називають невипадкову функцію, що формується під дією загальних або довгострокових тенденцій, що впливають на часовий ряд. Прикладом тенденції може виступати, наприклад, фактор росту досліджуваного ринку.

Сезонна складова часового ряду є періодично повторюваним компонентом часового ряду [4]. Властивість сезонності означає, що через приблизно рівні проміжки часу форма кривої, що описує поведінку залежної змінної, повторює свої характерні риси. Приклад такого ряду – ряди продажу декількох товарів, підданих сезонним коливанням. Ряд можна вважати несезонним, якщо при розгляді його зовнішнього вигляду не можна зробити припущень про повторюваність форми кривої через рівні проміжки часу.

Часові ряди також можна розділити на стаціонарні та нестаціонарні (рисунок 1.2). Стаціонарний часовий ряд – це ряд, статистичні властивості якого залишаються постійними з часом [5]. Це означає, що середнє, дисперсія та автокореляція ряду не змінюються на різних проміжках часу. Стаціонарний ряд не має довгострокового тренду, і його коливання мають постійну амплітуду і частоту.

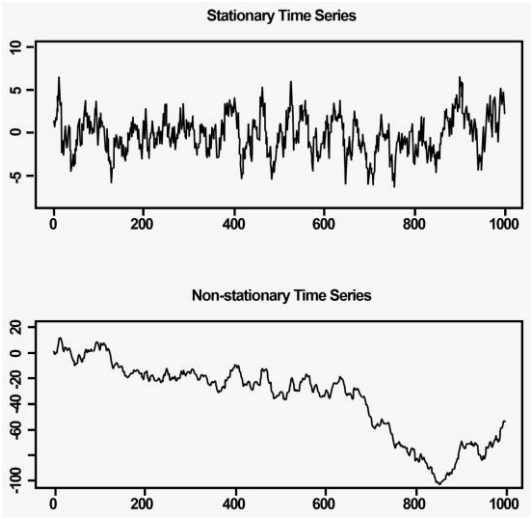

Рисунок 1.2 – Стаціонарний та нестаціонарний ряд

Стаціонарний часовий ряд легше прогнозувати, оскільки тренд є константою, на відміну від нестаціонарного часового ряду. Тренд нестаціонарного ряду може змінити подія, передбачити яку дуже складно. Як приклад, акції Твіттера, які впали через висловлювання Ілона Маска.

1.2.1 Методи прогнозування часових рядів

Вибір методу, який використовується для аналізу даних і прогнозування, залежить від проблеми, яка має бути вирішена, і характеру її даних. Прогнозування часових рядів використовує різноманітні статистичні методи та методи, засновані на машинному навчанні.

Класичні методи прогнозування базуються на статистичних моделях, які фіксують закономірності та зв’язки в історичних даних і екстраполюють їх на майбутнє [6]. Одними з найпоширеніших класичних методів є експоненціальне згладжування, ARIMA та моделі простору станів:

- експоненціального згладжування [6] використовують зважені середні значення минулих спостережень для прогнозування майбутніх значень, тоді як моделі ARIMA [7] використовують авторегресію та інтегроване ковзне середнє для врахування залежності та випадковості даних;
- моделі простору станів [8] є більш гнучкими та можуть працювати з кількома джерелами варіацій, такими як тенденції, цикли та сезонність.

Класичні методи є відносно простими, інтерпретованими та надійними, але вони можуть бути недієвими, якщо дані шумні, нелінійні або мають складну динаміку.

Методи машинного навчання базуються на алгоритмах, які навчаються на історичних даних і узагальнюють нові ситуації. Одними з найпоширеніших методів машинного навчання є регресія, дерева рішень, випадкові ліси та нейронні мережі:

- методи регресії використовують математичні функції для моделювання залежностей між вхідними та вихідними змінними та

прогнозування майбутніх значен [9], тоді як дерева рішень і випадкові ліси використовують правила та гілки, щоб розділити дані на менші підмножини та зробити прогнози;

- нейронні мережі складаються з шарів штучних нейронів [9], які можуть вивчати складні та нелінійні зв’язки в даних.

Методи машинного навчання є потужнішими, гнучкішими та адаптивнішими, ніж класичні методи, але вони також можуть вимагати більше даних, обчислень та налаштування, та вони не завжди можуть бути легко інтерпретованими.

Гібридні методи засновані на поєднанні або інтеграції різних методів для використання їх переваг і компенсації недоліків. Деякі з найпоширеніших гібридних методів – це ансамблеві методи, методи вилучення ознак і наскрізні методи:

- ансамблеві методи використовують кілька моделей і агрегують їхні прогнози [10], щоб зменшити дисперсію та підвищити точність прогнозів;

- методи вилучення ознак використовують один метод для перетворення даних у більш відповідний формат, а інший – для прогнозування на основі перетворених даних [10];

- наскрізні методи використовують єдину структуру, яка може обробляти всі аспекти проблеми прогнозування [10], такі як попередня обробка даних, побудова моделі та генерація прогнозів.

Змішані методи є більш інноваційними, універсальними та ефективними, ніж окремі методи, але вони також можуть бути складнішими та дорожчими у застосуванні.

1.3 Аналіз існуючих досліджень

Існує багато досліджень у сфері прогнозування сейсмічної активності вулканів, де моделі глибокого навчання знаходять широке застосування. Ці дослідження сприяють розвитку точних і швидких методів прогнозування

вулканічних вивержень, що є критичним для збереження життя та майна людей, що проживають у вулканічних районах.

Моделі машинного навчання використовувалися для ідентифікації та картографування теплових викидів від активних і охолоджуючих потоків лави [11], а також для оцінки площі покриття та загального об’єму лавових полів або відкладень [11]. Методи машинного та глибокого навчання були прийняті для виявлення та характеристики основних компонентів вулканічних шлейфів під час вибухових вивержень [12]. Крім того, алгоритми глибокого навчання були застосовані для автоматичного розпізнавання від тонких до інтенсивних теплових аномалій, використовуючи просторові відносини вулканічних особливостей [12]. Результати цих досліджень показують, що методи глибокого навчання, порівняно з традиційними методами машинного навчання, досягають кращої точності в автоматичному аналізі та розпізнаванні специфічних особливостей у контекстах прогнозування виверження вулканів.

1.3.1 Моніторинг тектонічних землетрусів вулканів на основі двох систем машинного навчання

У дослідженні [13] розроблено багатостанційний підхід до моніторингу вулканічних тектонічних землетрусів, заснований на методах перенесення. В досліджені застосовано дві системи машинного навчання – рекурентну нейронну мережу на основі довгих клітинок короткочасної пам’яті (RNN–LSTM) і часову згорткову мережу (TCN) – обидві навчені з основним набором даних і каталогом, що належать вулкану Десепшн-Айленд (Антарктида), зібраних з червня по грудень 2017.

Системи були навчені за підходом, заснованим на мультикореляції, тобто були обрані лише сейсмічні сліди, виявлені одночасно на різних сейсмічних станціях (продуктивність систем істотно покращилася). Виявлено, що система на основі RNN показала найвищу надійність у визначенні сейсмічної активності, за винятком випадків з низькою достовірністю у

виявленні сейсмічних слідів, тобто тих, які були лише частково подібними до базових зразків сейсмічних даних. Навпаки, мережа на базі TCN була здатна виявляти більшу кількість подій, однак, багато з цих подій були лише частково подібні до основних подій базової лінії (рисунок 1.3). Події базової лінії – це еталонні або стандартні події, які використовуються для порівняння та оцінки результатів виявлення. Вони являють собою типові або характерні приклади подій, які система повинна розпізнавати.

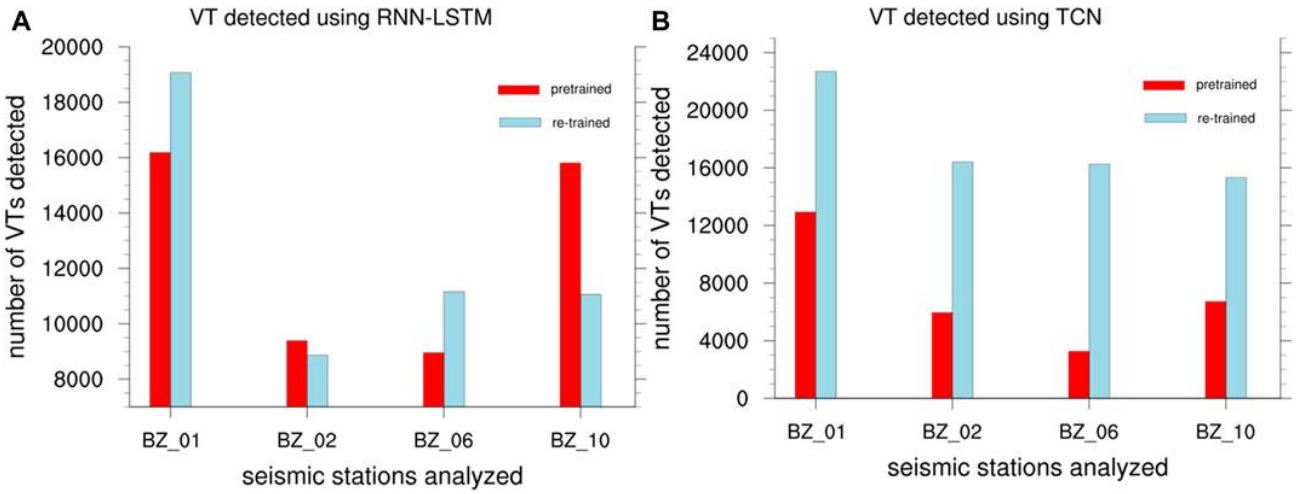

Рисунок 1.3 – Загальна кількість вулкано-тектонічних землетрусів, виявлених RNN–LSTM і TCN до та після процесу повторного навчання

Разом ці два підходи пропонують додаткові інструменти для моніторингу вулканів.

Також в досліджені описуються виявлені переваги порівняно з класичним алгоритмом короткочасного середнього над довготривалим (STA/LTA). Під час дослідження було обрано один день, коли відбулося кілька сотень землетрусів, і проаналізовано результати в погодинному часовому масштабі (рисунок 1.4). Враховуючи, що TCN завжди виявляв більшу кількість подій, ніж RNN–LSTM, було припущено, що вулкано-тектонічні землетруси, виявлені RNN–LSTM, були підмножиною тих, які виявив TCN, і вибрані лише ці події для аналізу. У вибраний день RNN–LSTM не розпізнавав

жодного землетруса до повторного навчання; після повторного навчання всі розпізнані вулкано-тектонічні землетруси раніше були класифіковані як події виверження.

Системи RNN–LSTM та TCN автоматично виявляють вулкано-тектонічні землетруси у сейсмічному сліді – кривої, записаної одним сейсмографом під час вимірювання руху ґрунту, без пошуку оптимальних налаштувань параметрів, що робить їх портативним, масштабованим та економічним інструментом із відносно низькою обчислювальною вартістю.

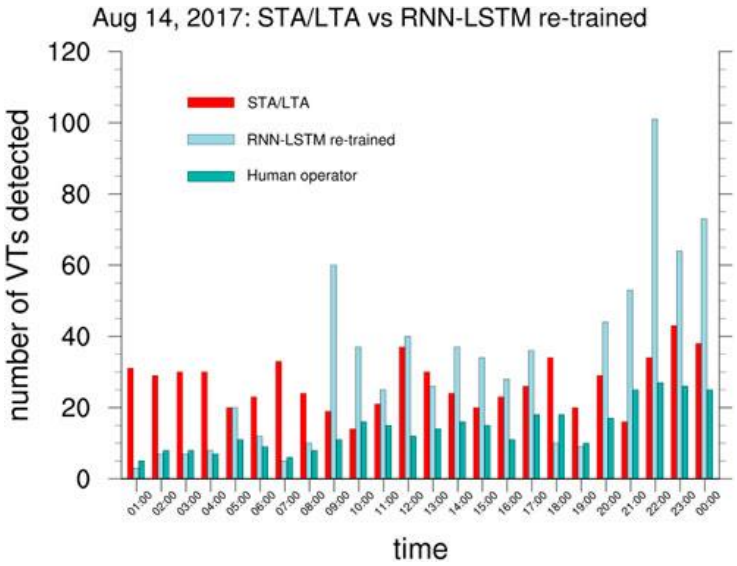

Рисунок 1.4 – Кількість вулканотектонічних землетрусів, виявлених щогодини 14 серпня 2017 року за допомогою алгоритму STA/LTA і повторно навченої нейронної мережі RNN-LSTM

Серед загальної кількості подій, визначених як землетруси, ті, які демонструють вірогідність виверження вулкану більше 80% після повторного навчання, можна було вважати точно класифікованими. Ці два описані в досліджені підходи пропонують додаткові інструменти для моніторингу вулканів: RNN-LSTM для дрібнозернистих сейсмічних каталогів і TCN для крупнозернистих сейсмічних каталогів.

1.3.2 Методи глибокого навчання на основі LSTM для прогнозування землетрусів з використанням іоносферних даних

У статті [14] аналізується зв’язок між землетрусами та даними TEC для виявлення землетрусів. TEC – загальний вміст електронів – це основний параметр для дослідження структури іоносфери. В цьому досліджені обговорюється мінливість іоносфери під час помірних і сильних землетрусів різної сили протягом 2012-2019 років. Запропоновані моделі використовують моделі глибокого навчання на основі LSTM (довгокороткочасної пам’яті) для класифікації днів річних землетрусів шляхом аналізу значень TEC за останні дні. Моделі прогнозування на основі LSTM порівнюються з класифікатором SVM, LDA і класифікатором Random Forest, щоб оцінити запропоновані моделі на основі прогнозу землетрусів (рисунок 1.5).

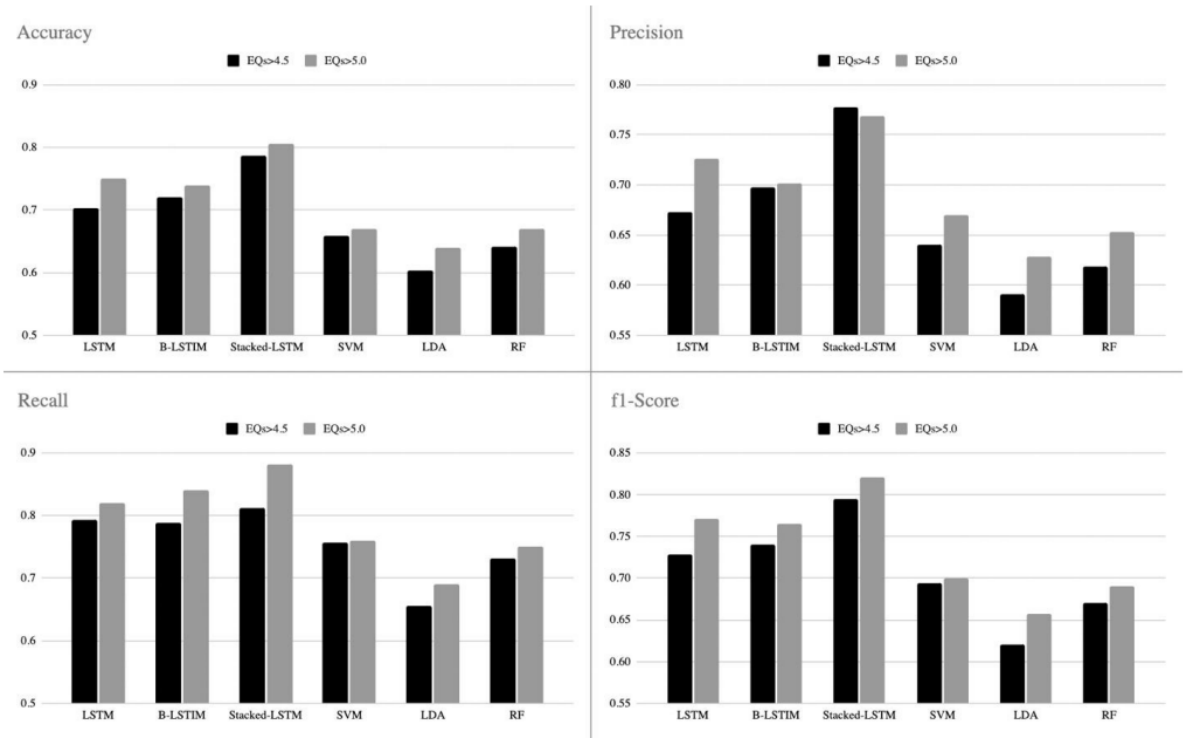

Рисунок 1.5 – Показники продуктивності моделей LSTM та інших моделей прогнозування на основі двох наборів даних

Результати показують, що запропоновані моделі LSTM забезпечують виявлення землетрусів із рівнем точності приблизно 0,82 у двох наборах тестів землетрусів, включаючи помірні та сильні землетруси, і можуть бути використані як інструмент для виявлення землетрусів на основі попередніх днів.

1.3.3 Дослідження прогнозування виверження вулкану за допомогою муографії з використанням згорткової нейронної мережі

В досліджені [15] демонструється можливість муографії для прогнозування виверження за допомогою згорткової нейронної мережі CNN. Муографія – це новий метод візуалізації внутрішніх структур активних вулканів за допомогою космічних мюонів високої енергії, що надходять майже горизонтально. У цьому дослідженні сім послідовних муографічних зображень вводяться в CNN для обчислення ймовірності вивержень на восьмий день (рисунок 1.6).

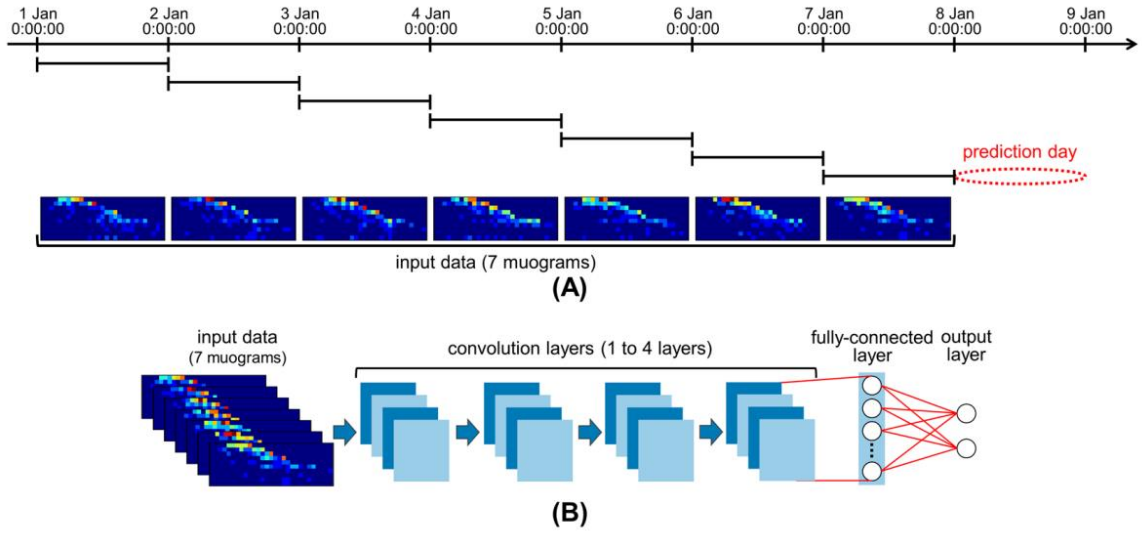

Рисунок 1.6 – Зв'язок між вхідними даними для моделі CNN (A) та умовою прогнозу для виверження, конфігурація моделі CNN (B)

Модель CNN було навчено з використанням налаштування гіперпараметрів через алгоритм оптимізації Байєса. Використовувані дані були отримані на вулкані Сакураджіма, Японія. Ефективність прогнозування досягла значення 0,726 для площі під кривою характеристик приймача, що демонструє значну кореляцію між муографічними зображеннями та подіями виверження.

1.4 Постановка задачі

Метою роботи є дослідження методів прогнозування сейсмічної активності вулканів за допомогою моделей глибокого навчання. При аналізі архітектур моделей deep learning було обрано дві моделі для навчання – LSTM та CNN. LSTM здатна ефективно працювати у моделюванні складних довгострокових залежностей даних, що є особливо важливим для прогнозування вулканічних подій на довгий період часу. Натомість CNN може виявляти конкретні зміни або структури у даних, що можуть вказувати на наближення вулканічного виверження.

Для досягнення мети роботи необхідно виконати ряд завдань, а саме:

- проаналізувати доступні часові ряди з даними о сейсмологічної активності вулкану, що вміщає в собі 9000 файлів з показанням десятьох датчиків навколо вулкану та часу до його виверження. Великий обсяг даних доцільний для моделей глибокого навчання, оскільки велика кількість даних сприяє підвищенню точності та ефективності прогнозів;
- створити моделі глибокого навчання штучних нейронних мереж на основі LSTM та CNN;
- навчити моделі на наявних наборах даних;
- розробити програмне забезпечення для проведення експериментальних досліджень, що включає створення інструментів для аналізу часових рядів із даними про сейсмологічну активність вулкана;
- порівняти та проаналізувати отримані результати.

2 ВИКОРИСТАННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

2.1 Штучні нейронні мережі

Штучна нейронна мережа – це математична модель, що складається з взаємопов'язаних вузлів, організованих у шари [16]. Кожен вузол виконує просту математичну операцію на своїх входах і передає результат на наступний рівень. Ці операції, як правило, є лінійними або нелінійними перетвореннями, що дозволяє мережі вивчати складні моделі та зв'язки в даних [16]. Нейронні мережі використовуються в машинному навчанні для таких завдань, як класифікація, регресія та кластеризація.

Нейронні мережі складаються з набору вузлів [17]. Вузли розподіляються принаймні на три шари: вхідний шар, «прихований» шар, вихідний шар. Нейронні мережі можуть мати більше одного прихованого шару, крім вхідного та вихідного [17]. У кожному вузлі мережі виконується обчислення, яке передбачає лінійну комбінацію вхідних сигналів з відповідними вагами та застосування функції активації до отриманого результату. Функція активації визначає вихідний сигнал вузла в залежності від вхідних даних та параметрів вузла [18]. Вона забезпечує нелінійність у роботі мережі, що є важливим для вирішення складних завдань. Найпоширенішими функціями активації є sigmoid, tanh та ReLU.

Немає обмежень щодо кількості вузлів і шарів, які може мати нейронна мережа, і ці вузли можуть взаємодіяти практично будь-яким способом [19]. Через це перелік типів нейронних мереж постійно розширюється. Одна з категорій, на яку можна поділити нейронні мережі – це глибокі і неглибокі ШНМ [20]:

- неглибокі нейронні мережі зазвичай мають лише один прихований шар;
- глибокі нейронні мережі мають кілька прихованих шарів.

Неглибокі нейронні мережі швидші [21], ніж глибокі нейронні мережі, але вони мають обмежену здатність вирішувати складні завдання, такі як розпізнавання образів у високорозмірних даних (наприклад, розпізнавання облич на фотографіях), обробка природної мови (наприклад, машинний переклад та аналіз тексту), а також прогнозування складних часових рядів (наприклад, передбачення погоди або фінансових ринків).

2.2 Особливості глибоких нейронних мереж

Глибоке навчання – це тип машинного навчання та штучного інтелекту, який імітує спосіб отримання людьми певних типів знань [22]. Deep Learning навчається на основі великих обсягів даних з використанням зворотного поширення помилки. Нейронні мережі самостійно виявляють закономірності, підлаштовуючись під тренувальні дані [23]. Це дає їм змогу ефективно обробляти нові дані, навіть якщо вони відрізняються від тих, на яких проходило навчання.

Моделі глибокого навчання можна навчити виконувати завдання класифікації та розпізнавати шаблони на фотографіях, тексті, аудіо та інших різних даних. Вони також використовуються для автоматизації завдань, які зазвичай потребують людського інтелекту, таких як опис зображень або транскрибування аудіофайлів.

Моделі штучних нейронних мереж глибокого навчання мають кілька рівнів взаємопов’язаних вузлів [24], причому кожен рівень базується на попередньому для вдосконалення й оптимізації прогнозів і класифікацій. Глибоке навчання виконує нелінійні перетворення своїх вхідних даних і використовує отримані результати для побудови моделі, яка відображає закономірності в даних. Ітерації тривають, доки результат не досягне прийнятного рівня точності [25].

В додатку Б наведена таблиця з порівняльною характеристикою між deep learning та традиційним машинним навчанням. Алгоритм глибокого

навчання є одним з найпопулярніших серед машинного навчання [26] та може бути вирішенням в таких випадках, коли:

- люди-експерти недоступні;
- люди не можуть пояснити рішення, прийняті за допомогою свого досвіду, наприклад: розуміння мови, медичні рішення та розпізнавання мовлення;
- рішення проблеми оновлюється з часом, наприклад: прогноз цін, перевага акцій, прогноз погоди та відстеження;
- рішення потребують адаптації на основі конкретних випадків, наприклад: персоналізація, біометрія;
- проблема значно перевищує людські здібності до міркування, наприклад аналіз настроїв, зіставлення реклами з Facebook, обчислення рейтингів веб-сторінок [27].

У глибокому навчанні застосовуються різні типи нейронних мереж [28]:

- персептрон – найпростіша форма, яка використовується для бінарної класифікації;
- згорткові нейронні мережі (CNN) – ефективні в обробці зображень, виділяючи ознаки [29], як-от грані та текстури;
- рекурентні нейронні мережі (RNN) – підходять для послідовних даних [29], з пам’яттю для врахування попередніх станів;
- глибокі нейронні мережі (DNN) – багатошарові мережі, що виявляють складні патерни в даних [29];
- спеціалізовані мережі (наприклад, GAN і LSTM) – генеративно-змагальні мережі (GAN) для створення даних і LSTM для обробки послідовних даних із довгостроковою залежністю [30].

2.3 Базова архітектура штучної нейронної мережі

Персептрон – це тип штучної нейронної мережі, яка є фундаментальною концепцією машинного навчання [31]. Він використовується для класифікації

вхідних даних за допомогою лінійної моделі. У персептрона є вхідний шар, один або кілька прихованих шарів нейронів та вихідний шар. Він складається з таких основних компонентів (рисунок 2.1) як:

- вхідний рівень: вхідний рівень складається з одного або кількох вхідних нейронів [31], які отримують вхідні сигнали із зовнішнього світу або з інших рівнів нейронної мережі;

- вагові коефіцієнти: кожен вхідний нейрон пов’язаний із ваговим коефіцієнтом, який представляє силу зв’язку між вхідним нейроном і вихідним нейроном [31];

- зміщення: до вхідного рівня додається зсув, щоб надати персептрону додаткову гнучкість у моделюванні складних шаблонів у вхідних даних [31];

- функція активації: функція активації визначає вихід персептрона на основі зваженої суми входів і члена зсуву [31]. Загальні функції активації, що використовуються в персептронах, включають функцію кроку, функцію сигмоіда та функцію ReLU;

- вихід: вихід персептрона є одним двійковим значенням, 0 або 1, яке вказує на клас або категорію, до якої належать вхідні дані.

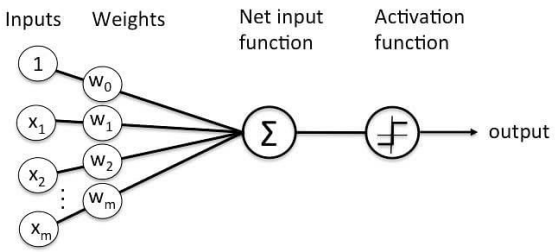

Рисунок 2.1 – Проста архітектура персептрона

Персептрон зазвичай навчається за допомогою алгоритму навчання під наглядом, такого як алгоритм навчання персептрона [32] або зворотне поширення. Під час навчання ваги та зміщення персептрона налаштовуються, щоб мінімізувати похибку між прогнозованими значеннями та справжнім результатом для даного набору навчальних прикладів.

Існує два типи моделей перспетрону – одношаровий та багатошаровий.

Одношарова модель персептрона – це один із найпростіших типів штучних нейронних мереж, що складається з прямої мережі та включає порогову передачу всередині моделі [33]. Порогова передача означає, що для кожного нейрона використовується порогова функція активації, яка приймає рішення про активацію нейрона на основі вхідного сигналу. Якщо сума зважених вхідних значень перевищує певний поріг, нейрон активується (видає 1), інакше він не активується (видає 0). Основною метою моделі одношарового персептрона є аналіз лінійно роздільних об’єктів із бінарними результатами. Одношаровий персептрон може вивчати лише лінійно роздільні шаблони.

Багатошарова модель персептрона в основному схожа на одношарову модель персептрона, але має більше прихованих шарів. Багатошарова модель персептрона має більшу обчислювальну потужність і може обробляти лінійні та нелінійні шаблони [33]. Крім того, вона також реалізує логічні вентилі, такі як AND, OR, XOR, XNOR та NOR.

Персептрон – це найпростіша нейронна мережа, яка становила основу до більш складних нейронних мереж, таких як довга короткострокова пам'ять та згорткова нейронна мережа, які сьогодні використовуються в глибокому навчанні.

2.4 Мережі довготривалої короткочасної пам’яті

Довга короткострокова пам'ять (LSTM) – це тип архітектури рекурентної нейронної мережі (RNN), призначеної для вирішення проблеми зникнення градієнта та фіксації довготривалих залежностей у послідовних даних [34]. Проблема зниклого градієнта виникає у глибоких нейронних мережах під час навчання методом зворотного поширення помилки. Під час цього процесу градієнт помилки (вектор похідних функції втрати за вагами мережі) передається від виходу мережі до її входу. У деяких випадках, коли мережа дуже глибока, градієнт може зменшуватись (зникає) пропорційно до

глибини мережі, тобто градієнт стає дуже малим або навіть нульовим. Це ускладнює навчання глибоких мереж, оскільки ваги далеких від виходу шарів майже не оновлюються, і вони залишаються недостатньо навченими. LSTM вирішує цю проблему за рахунок своєї архітектури, яка дозволяє зберігати та використовувати довгострокові залежності в даних, навіть коли вони знаходяться на великому віддаленні в часі [34]. На відміну від традиційних нейронних мереж, LSTM включає з'єднання зворотного зв'язку [35], що дозволяє обробляти цілі послідовності даних, а не лише окремі точки даних. Це робить її дуже ефективною для розуміння та прогнозування моделей у послідовних даних, таких як часові ряди, текст і мовлення.

Архітектура мережі LSTM [36] складається з трьох частин, як показано на рисунку 2.2, кожна частина виконує окрему функцію.

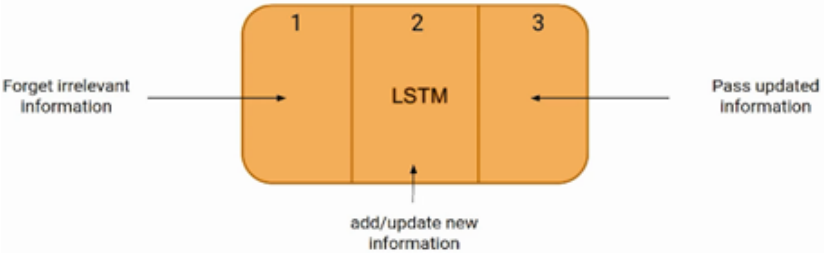

Рисунок 2.2 – Архітектура LSTM

Перша частина вибирає, чи слід запам'ятовувати інформацію, що надходить із попередньої позначки часу, чи вона є нерелевантною та її можна забути. У другій частині клітинка намагається дізнатися нову інформацію з вхідних даних цієї клітинки [37]. У третій частині, клітинка передає оновлену інформацію з поточної мітки часу до наступної мітки часу. Цей один цикл LSTM вважається одноразовим кроком [38].

Ці три частини блоку LSTM відомі як ворота (рисунок 2.3). Вони контролюють потік інформації в комірку пам'яті або комірку LSTM і з неї. Перші ворота називаються Forget gate, другі ворота відомі як Input gate, а останні – Output gate [38]. Блок LSTM, який складається з цих трьох воріт і

комірки пам’яті або комірки LSTM, можна розглядати як шар нейронів у традиційній нейронній мережі прямого зв’язку [38], де кожен нейрон має прихований шар і поточний стан.

LSTM має прихований стан, який служить посередником між станом клітини та зовнішнім світом. Він може вибірково запам'ятовувати або забувати інформацію про стан комірки та створювати вихідні дані.

Прихований стан відомий як короткострокова пам'ять [38], а стан комірки відомий як довгострокова пам'ять. Стан комірки містить інформацію разом із усіма часовими мітками.

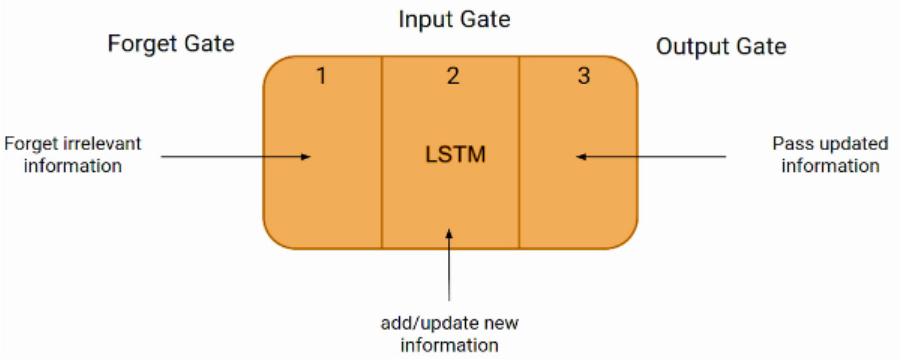

Рисунок 2.3 – Ворота LSTM

Модель LSTM була розроблена Hochreiter і Schmidhuber [39], вона вирішує проблему зниклого градієнту, спричинену традиційними RNNs і алгоритмами машинного навчання. Модель LSTM можна реалізувати на Python за допомогою бібліотеки Keras.

2.5 Згорткові нейронні мережі

Згорткова нейронна мережа (ConvNet/CNN) – це алгоритм глибокого навчання, який спеціалізується на обробці даних, що мають сіткову топологію. Попередня обробка, необхідна в ConvNet, набагато менша порівняно з іншими алгоритмами класифікації [40].

Архітектура ConvNet (рисунок 2.4) аналогічна структурі підключення нейронів у людському мозку [41] та була натхненна організацією зорової кори. Окремі нейрони реагують на стимули лише в обмеженій області поля зору, відомої як рецептивне поле [41]. Набір таких полів перекривається, щоб охопити всю візуальну область.

CNN зазвичай має три рівні: згортковий рівень, рівень об’єднання та повністю зв’язаний рівень [42].

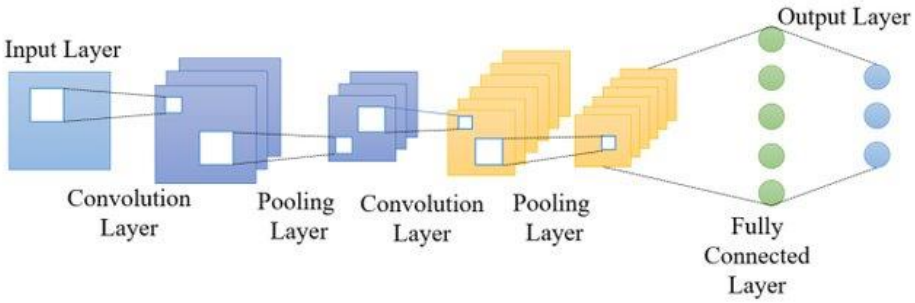

Рисунок 2.4 – Базова архітектура CNN

Рівень згортки є основним будівельним блоком CNN. Цей рівень виконує скалярний добутку між вхідними даними і фільтром (ядром), що дозволяє вилучати важливі ознаки з вхідних даних [42].

Рівень об’єднання замінює вихідні дані мережі в певних місцях шляхом отримання сумарної статистики найближчих виходів [42]. Це допомагає зменшити просторовий розмір представлення, що зменшує необхідну кількість обчислень і ваги. Операція об’єднання обробляється для кожного фрагмента представлення окремо.

Нейрони в повністю підключеному рівні мають повний зв’язок з усіма нейронами в попередньому та наступному шарі, як це видно у звичайному FCNN [42]. Ось чому його можна обчислити шляхом множення матриці з наступним ефектом зміщення. Цей рівень допомагає відобразити представлення між вхідним і вихідним сигналами [42].

2.6 Перенавчання нейронних мереж

Глибокі нейронні мережі стали популярними в різних програмах машинного навчання, таких як розпізнавання зображень, розпізнавання мови та обробка природньої мови. Однак вони схильні до перенавчання, яке відбувається, коли модель навчена занадто добре відповідати навчальним даним і через це не може опрацьовувати належиним чином нові дані, тобто втрачає можливість до узагальнення. Для запобігання перенавчанню глибоких нейронних мереж використовується Dropout.

Dropout або «Метод проріджування» – це техніка регулярізації, яка випадковим чином вилучає (тобто встановлює на нуль) ваги деяких нейронів в нейронній мережі під час навчання [43]. Ідея проріджування полягає в тому, щоб змушувати мережу вивчати надлишкові представлення вхідних даних. Через випадкове вилучення нейронів мережа стає менш чутливою до питомої ваги окремих нейронів і більш стійкою до зашумлених вхідних даних. Dropout можна розглядати як навчання ансамблю з кількох нейронних мереж, у кожній з яких випадково випадають різні набори нейронів.

Dropout реалізується на етапі навчання нейронної мережі. Під час навчання кожен нейрон у мережі або зберігається з ймовірністю p, або випадає з ймовірністю 1-p [43]. Імовірність p є гіперпараметром, який можна налаштувати для досягнення бажаного рівня регуляризації. Як правило, p встановлюється між 0,2 і 0,5. Під час прямого проходу навчання мережа обчислює вихід, використовуючи лише збережені нейрони. Під час зворотного проходу градієнти обчислюються лише для збережених нейронів. Ваги нейронів, які були обнулені, не оновлюються під час навчання. Це еквівалентно тимчасовому видаленню нейронів із мережі [43]. Під час тестування відключення вимикається, і для обчислення результату використовується повна мережа. Однак, щоб гарантувати, що очікуване значення виходу є однаковим під час навчання та тестування, ваги утриманих нейронів масштабуються за ймовірністю утримання.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Опис набору даних

Для роботи, з відкритого ресурсу Kaggle були взяті дані, представлені Італійським національним інститутом геофізики та вулканології. Дані були зібрані 10 сейсмічними датчиками впродовж 24-годинного моніторингу сейсмічної та активної вулканічної активності по всій Італії. Дані розділені на два набори: навчальні та тестові та складаються з двох файлів – train/train.csv та [train|test]/*.csv.

Файл train.csv описує метадані для навчальних файлів. Файл складається з двох полів:

- segment_id – ідентифікаційний код для сегмента даних. Збігається з назвою пов’язаного файлу даних;
- time_to_eruption – цільове значення, час до наступного виверження.

Файли [train|test]/*.csv – файли даних. Кожен файл містить десять хвилин журналів з десяти різних сейсмічних датчиків, розташованих навколо вулкана. Показання були нормалізовані в кожному сегменті, частково для забезпечення того, щоб показання потрапляли в діапазон значень int16.

Файл submision.csv містить заповнений segment_id та пустий time_to_eruption – використовується для тестування нейронної мережі.

3.2 Аналіз часового ряду

Головною метою розкладання часових рядів є розкладання рівнів ряду на складові компоненти, такі як тренд і сезонність, з метою врахування їх при прогнозуванні, тобто декомпозиція часового ряду.

Оскільки значень в ряді дуже багато, для візуалізації ряду були взяті дані першого рядка файлу (рисунок 3.1).

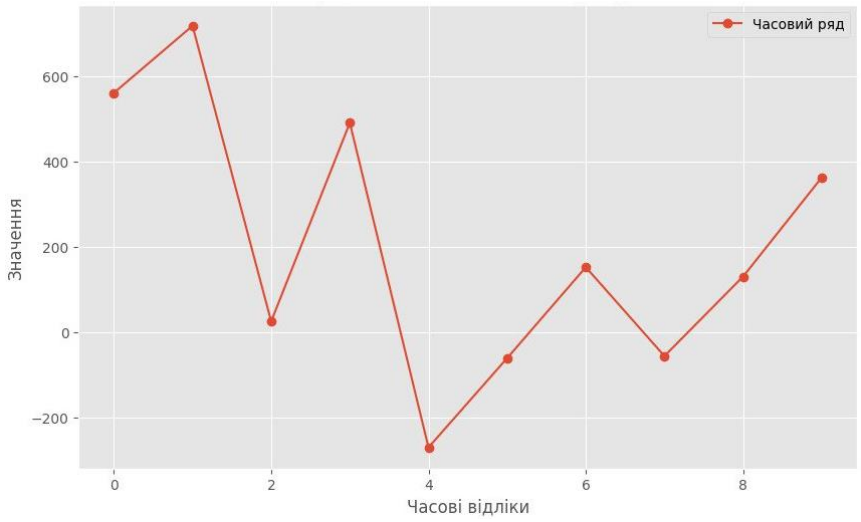

Рисунок 3.1 – Графік значень часового ряду

З наведеного рисунку видно, що в ряді наявний тренд. Для більшої наочності було виділено тренд та сезонність окремо. Аналізуючи часовий ряд перш за все виявляють тенденцію (тренд), яка визначає основний напрямок розвитку явища за великий проміжок часу (рисунок 3.2). Для побудови трендової складової використано метод експоненційного згладжування. Експоненційне згладжування – це метод аналізу часових рядів, який дозволяє зменшити випадкові коливання даних та виділити трендову складову.

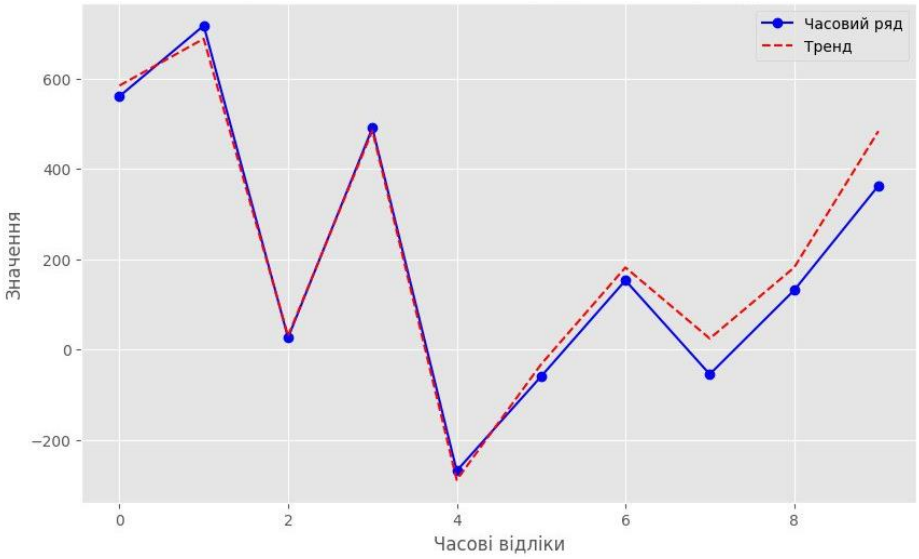

Рисунок 3.2 – Графік значень часового ряду та його тренду

Далі виділено сезонну складову (рисунок 3.3):

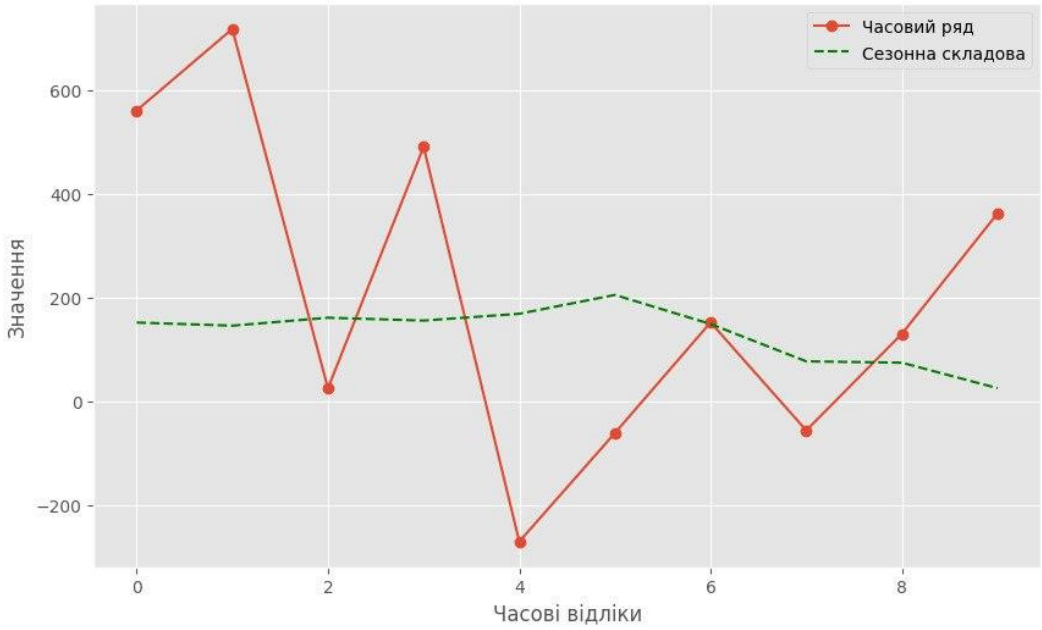

Рисунок 3.3 – Графік значень часового ряду з виділеною сезонною складовою

Виходячи з отриманих результатів вже можна сказати, що ряд є нестаціонарним і присутній тренд і сезонність.

Для отримання кінцевого результату проведено тест Дікі-Фуллера. Результатом такого тесту є значення p-value, яке дозволяє остаточно зробити висновок про стаціонарність ряду. У випадку, коли p-value < 0.05 – ряд є стаціонарним, в іншому випадку – нестаціонарним. В результаті тесту отримано значення p-value, що дорівнює 0.109, що підтверджує припущення про нестаціонарність ряду.

3.3 Аналіз даних

Щоб подивитися структуру набору даних файлу train.csv він був завантажений до датафрейму df train за допомогою бібліотеки pandas. Метод head(10) повернув перші 10 рядків (рисунок 3.4). Кожен рядок представляє сегмент даних про вулкан та час до його виверження.

	segment_id	time_to_eruption
0	1136037770	12262005
1	1969647810	32739612
2	1895879680	14965999
3	2068207140	26469720
4	192955606	31072429
5	1640671020	22264866
6	288840453	34952168
7	1162441568	9504818
8	1791400073	11719563
9	2059516238	11629084

Рисунок 3.4 – Структура файлу tran.csv

На першому етапі проведено аналіз описової статистики для стовпців segment_id і time_to_eruption. Ця статистика включає кількість числових значень, їх мінімальні та максимальні значення, а також стандартне відхилення та квартилі.

Таблиця 3.2 – Описова статистика для набору даних

	segment_id	time_to_eruption
count	4.431000e+03	4.431000e+03
mean	1.074694e+09	2.284891e+07
std	6.161966e+08	1.348439e+07
min	5.131810e+05	6.250000e+03
25%	5.527934e+08	1.127016e+07
50%	1.066153e+09	2.246559e+07
75%	1.606350e+09	3.434356e+07
max	2.146939e+09	4.904609e+07

Середнє значення часу до вибуху становить приблизно 2.284891e+07 секунд (приблизно 264 дні), що вказує на те, що в середньому вулканічний вибух відбувається через довгий час після початкового спостереження.

Стандартне відхилення для часу до вибуху вулкану становить 1.348439e+07 секунд, що свідчить про значні коливання в даних.

Щоб перевірити датафрейм на кількість пропущених значень використано функцію isnull().sum(). В результаті отримано segment_id = 0 та time_to_eruption = 0, що означає, що в датафреймі відсутні пропущені значення в обох стовпцях.

Використовуючи функцію hist() побудовано гістрограму розподілу значень стовпця time_to_eruption з датафрейму df_train (рисунок 3.5).

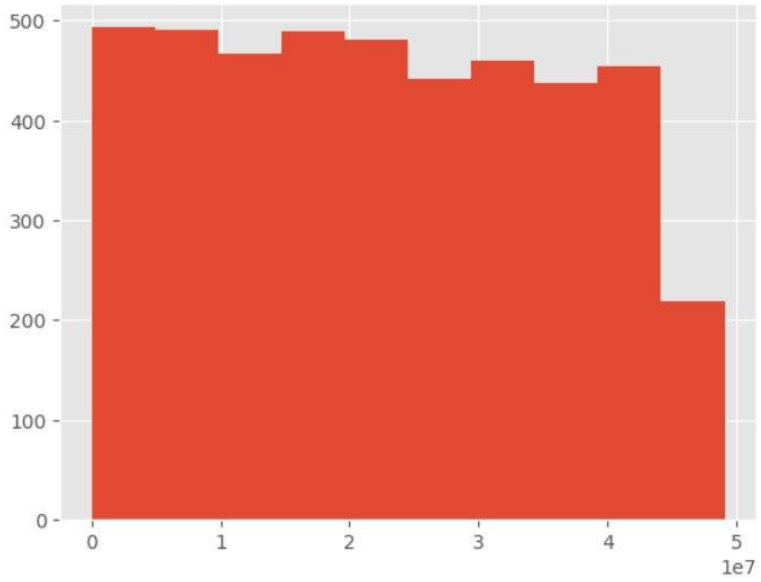

Рисунок 3.5 – Гістограма розподілу значень стовпця time_to_eruption

Проаналізувавши гістограму та порахувавши кількість значень у кожному біні гістограми отримано результат – найбільш розповсюдженим є діапазон значень приблизно від 29430152.2 до 34334135.9, оскільки цей бін має найбільшу кількість спостережень – 460. Використовуючи функцію max() знайдено максимальне значення часу до виверження вулкану – 490460087.

Візуалізовано розподіл часу до вибуху вулкану з використанням гістограми та кривої нормального розподілу. Створено гістограму розподілу часу до вибуху вулкану (рисунок 3.6), fit=norm додає криву нормального розподілу, щоб порівняти з фактичним розподілом даних.

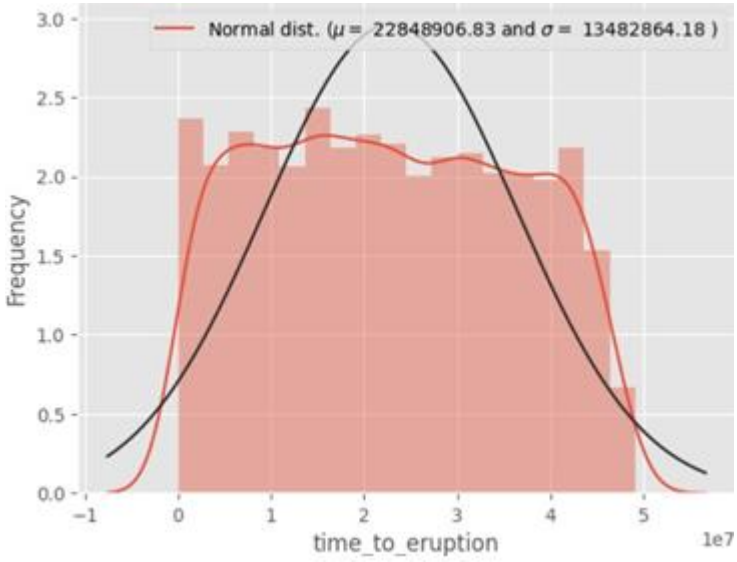

Рисунок 3.6 – Гістограма розподілу часу до вибуху вулкана

Далі сгенеровано QQ-plot для порівняння фактичного розподілу даних з нормальним розподілом (рисунок 3.7). Розподіл часу до вибуху вулкану має середнє значення приблизно 22 848 906.83 одиниць, а стандартне відхилення становить близько 13 482 864.18 одиниць. Це вказує на розкид значень навколо середнього значення. Це може бути зумовлено різноманітністю вулканічних систем, геологічними умовами або іншими факторами, які можуть впливати на час до вибуху.

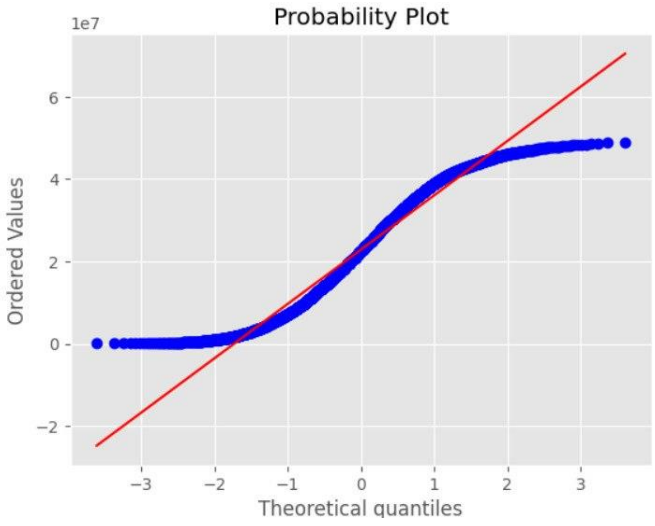

Рисунок 3.7 – QQ-графік

Розглянуто окремий файл з даними, що містять в собі значення сенсорів (sensor_1 до sensor_10) впродовж їх 10 хвилинної роботи (рисунок 3.8).

	sensor_1	sensor_2	sensor_3	sensor_4	sensor_5	sensor_6	sensor_7	sensor_8	sensor_9	sensor_10
0	614.0	653.0	35.0	478.0	-314.0	0.0	217.0	121.0	243.0	632.0
1	561.0	718.0	26.0	491.0	-269.0	-60.0	153.0	-55.0	131.0	363.0
2	496.0	846.0	18.0	423.0	-166.0	-160.0	30.0	40.0	40.0	81.0
3	429.0	1000.0	14.0	415.0	-33.0	-223.0	-87.0	-131.0	-60.0	-203.0
4	368.0	1098.0	19.0	419.0	108.0	-95.0	-219.0	-55.0	-152.0	-456.0
5	324.0	1073.0	26.0	368.0	256.0	-41.0	-345.0	232.0	-236.0	-655.0
6	289.0	949.0	46.0	317.0	355.0	-313.0	-381.0	286.0	-280.0	-813.0
7	259.0	770.0	15.0	283.0	369.0	-255.0	-280.0	602.0	-307.0	-932.0
8	240.0	472.0	9.0	207.0	327.0	-75.0	-71.0	458.0	-328.0	-1018.0
9	239.0	22.0	-21.0	43.0	265.0	-45.0	106.0	537.0	-363.0	-1066.0

Рисунок 3.8 – Фрагмент зібраних даних щодо значень сенсорів

Для подальшого аналізу даних (рисунок 3.9) створена описова статистика для стовпців sensor_1-sensor_10.

	sensor_1	sensor_2	sensor_3	sensor_4	sensor_5	sensor_6	sensor_7	sensor_8	sensor_9	sensor_10
count	60001.000000	60001.000000	60001.000000	60001.000000	60001.000000	60001.000000	60001.000000	60001.000000	59533.000000	60001.000000
mean	-2.566141	-3.353361	-0.018533	-2.012000	1.721038	2.547641	-0.832786	-0.176447	1.026372	-116.737804
std	290.689446	519.035022	231.049818	264.341298	256.266425	314.794419	415.310810	339.036140	293.248579	1042.540568
min	-1234.000000	-5323.000000	-1279.000000	-1573.000000	-1837.000000	-1437.000000	-3553.000000	-3787.000000	-1935.000000	-5174.000000
25%	-185.000000	-318.000000	-148.000000	-170.000000	-151.000000	-211.000000	-239.000000	-216.000000	-182.000000	-723.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	182.000000	311.000000	148.000000	162.000000	156.000000	212.000000	238.000000	218.000000	187.000000	585.000000
max	1442.000000	3630.000000	1229.000000	1678.000000	2382.000000	1585.000000	4132.000000	3020.000000	2053.000000	4925.000000

Рисунок 3.9 – Фрагмент даних для описової статистики

Розподіл даних для всіх сенсорів асиметричний, з середнім значенням близьким до нуля, але значними відхиленнями.

Створено теплову карту кореляційної матриці даних використовуючи бібліотеку seaborn (рисунок 3.10). Кореляція – це статистичний показник, який вказує на силу та напрям зв'язку між двома змінними. Зазвичай кореляція

вимірюється у діапазоні від -1 до 1. Значення близьке до 1 вказує на дуже сильну позитивну кореляцію, тобто коли одна змінна зростає, інша змінна також зростає. Значення близьке до -1 вказує на дуже сильну негативну кореляцію, коли одна змінна зростає, інша змінна зменшується. Значення кореляції близьке до 0 означає відсутність кореляції, тобто зміни в одній змінній не впливають на зміни в іншій змінній.

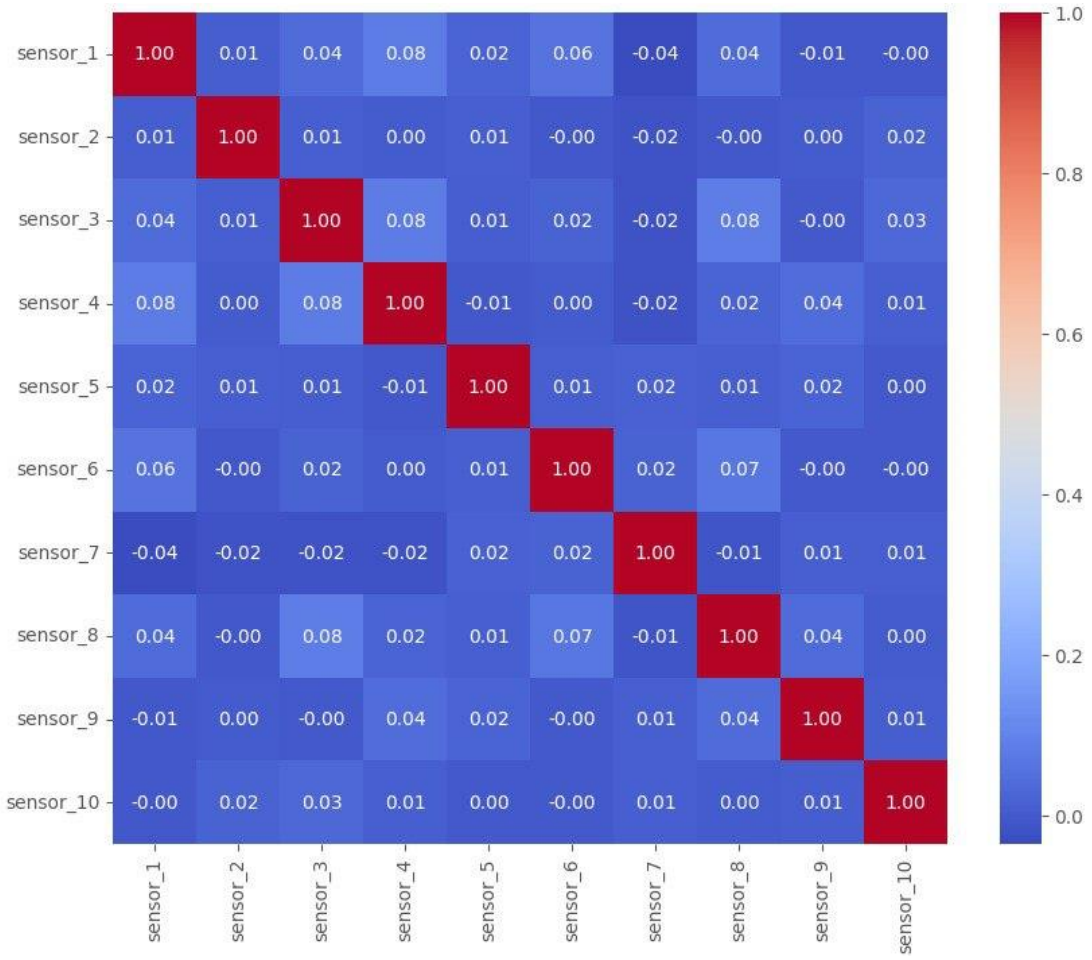

Рисунок 3.10 – Теплова карта

Sensor_1 має слабку позитивну кореляцію з Sensor_4, що вказує на те, що зі зростанням значень Sensor_1 зростають і значення Sensor_4, але ця кореляція є дуже слабкою. Також Sensor_6 має помірну позитивну кореляцію з Sensor_8, що вказує на певну залежність між цими двома сенсорами. Sensor_7 має слабку негативну кореляцію з Sensor_2, що вказуває на зворотню

залежність між ними. Багато з цих кореляцій є слабкими, що свідчить про те, що дані від цих сенсорів досить незалежні одне від одного.

3.4 Створення датасету та моделей глибоких нейронних мереж

Для створення датасету для навчання моделі був визначен клас TimeEruptionDataset (лістинг 3.1).

Лістинг 3.1 – Реалізація класу для створення датасету для навчання моделі

```
class TimeEruptionDataset (torch.utils.data.Dataset):
def __init__(self, df):
self.segment = df.segment_id.values
self.time = df.time_to_eruption.values
def __len__(self):
return self.segment.shape[0]
def __getitem__(self, index):
df = pd.read_csv(f"../input/predict-volcanic-eruptions-ingv-
oe/train/{self.segment[index]}.csv")
df = df.fillna(0)
df = df[:60000].values.reshape(6000, 100)
label = self.time[index]
return df, label
```

В цьому класі зчитано CSV-файл з даними для певного сегменту з індексом «index», обмежено кількість рядків у DataFrame до 60000 та перетворено у форму (6000, 100) – 6000 рядків по 100 значень у кожному рядку для полегшення обробки даних та прискорення навчання. В дослідженні пропуски даних можуть означати відсутність активності чи події, тобто якщо датчик не повідомляє про будь-які зміни, це інтерпретовано як відсутність подій, тому пропущені значення в DataFrame заповнено значенням 0. Кожен екземпляр TimeEruptionDataset класу дозволяє отримувати доступ до певного сегменту та відповідного часу до виверження вулкану для цього сегменту.

Для обробки та аналізу даних використано бібліотеку Pandas. Робота Pandas з даними будується поверх бібліотеки NumPy, яка є інструментом нижчого рівня. Pandas найчастіше використовується для маніпулювання

даними, а об’єкти NumPy в основному використовуються для створення масивів або матриць, які можна застосовувати до моделей глибокого навчання. Також для реалізації нейронних мереж було використано бібліотеку Keras. Keras – це відкрита бібліотека, написана мовою Python, що забезпечує взаємодію зі штучними нейронними мережами.

Було створено два класи для реалізації нейронних мереж. Клас LstmNet визначає нейронну мережу LSTM. В фрагменті коду (лістинг 3.2) відбувається ініціалізація шарів нейронної мережі.

Лістинг 3.2 – Ініціалізація шарів нейронної мережі LSTM

```
class LstmNet (pl.LightningModule):
    def init(self):
        super().init()
        self.lstm_net = nn.LSTM(100 , 200 , bidirectional=False,
batch_first=True)
        self.linearr = nn.Linear(400, 400)
        self.linear_aux_out = nn.Linear(400, 1)
        self.critrion = nn.MSELoss(reduction='mean')
        self.dropout = nn.Dropout(0.3)
```

lstm_net являє собою одношарову нейронну рекурентну мережу LSTM з 100 вхідними ознаками і 200 прихованими блоками. Цей шар немає двонаправлених зв'язків.

Далі йдуть 2 повнозв'язних шара Linear. Перший шар відображає 400 вхідних ознак у 400 вихідних ознак, а другий приймає на вхід 400 ознак та видає 1 вихідну ознаку.

Dropout є шаром регуляризації, в даному випадку з ймовірністю відключення, що дорівнює 30%. Також, у коді ініціалізується функція втрат MSELoss, яка реалізує середньоквадратичну функцію втрат.

MSE – це метрика, яка використовується для вимірювання середньої квадратичної різниці між прогнозованими значеннями і фактичними значеннями в регресійних моделях. MSE обчислюється як середнє арифметичне квадратів відхилень між прогнозами моделі і відповідними

фактичними значеннями. Вище MSE вказує на більшу похибку моделі. Середньоквадратична помилка розраховується за формулою:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 , \quad (3.1)$$

де n – кількість прикладів в наборі даних, y_i – фактичне значення для і-го прикладу, $\hat{y}_i$ – передбачене значення для і-го прикладу.

Другий клас CnnNet визначає нейронну мережу CNN (лістинг 3.3). Шари Conv1D виконують згорткові операції на вхідних даних. in_channels визначає кількість вхідних каналів, out_channels визначає кількість вихідних каналів, kernel_size – розмір ядра згортки, а padding – кількість нульових відступів, доданих навколо входу по кожному виміру.

Шари MaxPooling1D виконують підвибірку за допомогою максимального значення в кожному згортковому вікні. Вони допомагають зменшити кількість параметрів та обчислювальну складність моделі, зберігаючи важливу інформацію. Далі йдуть 2 повнозв'язних шара Linear.

Лістинг 3.3 – Ініціалізація шарів CNN

```
class CnnNet(pl.LightningModule):
  def __init__(self, n_t, n_f):
super(ConvNet, self).__init__()
        self.conv1 = nn.Conv1d(6000, 200, kernel_size=3,
padding=1)
        self.conv2 = nn.Conv1d(200, 200, kernel_size=3,
padding=1)
        self.conv3 = nn.Conv1d(200 ,200, kernel_size=3,
padding=1)
        self.pool = nn.MaxPool1d(kernel_size=2)
        self.fc1 = nn.Linear(200*25, 400)
        self.fc2 = nn.Linear(400, 1)
        self.criterion = nn.MSELoss(reduction='mean')
        self.dropout = nn.Dropout(0.3)
```

Dropout шар реалізує метод регуляризації, випадково відключаючи певний відсоток нейронів під час навчання, у цьому випадку 30%.

Далі визначається функція прямого проходу forward pass (лістинг 3.4). Вхідні дані x пропускаються через перший згортковий шар (conv1), після чого використовується функція активації ReLU. Це дозволяє моделі вивчити нелінійність у данних.

Функція активації ReLU представлена наступною формулою:

$$y = f(x) = \max(0, x), \tag{3.2}$$

де y – елемент поточного рівня, x – елемент з вхідних даних. ReLU повертає 0 для всіх негативних входів і сам вхід для позитивних значень. Це означає, що ReLU обнуляє всі негативні значення та пропускає позитивні значення без змін.

Аналогічно, вихід від першого згорткового шару подається на вхід до другого згорткового шару (conv2), і знову використовується функція активації ReLU. Результат згорткових шарів піддається пулінгу – зменшенню розмірності. Після пулінгу результат подається на третій згортковий шар (conv3), після чого застосовується функція активації ReLU.

Лістинг 3.4 – Визначення функції forward

```
def forward(self, x):
        x = torch.relu(self.conv1(x))
        x = torch.relu(self.conv2(x))
        x = self.pool(x)
        x = torch.relu(self.conv3(x))
        x = self.pool(x)
        x = x.view(x.size(0), -1) # Flatten the output of conv
layers
        x = torch.relu(self.fc1(x))
        x = self.dropout(x)
        x = self.fc2(x)
        return x
```

Ще один шар пулінгу використовується для зменшення розмірності вихідних даних. Після згорткових шарів результат необхідно перетворити у

вектор. x.view(x.size(0), -1) робить розгортання вихідних даних для подальшого подання їх на повністю зв'язаний шар. Перетворений вектор подається на перший повністю зв'язаний шар (fc1), після чого використовується функція активації ReLU. Далі застосовується шар dropout, який допомагає уникнути перенавчання. Результат подається на другий повністю зв'язаний шар (fc2), який не має функції активації, що використовується для виведення кінцевих прогнозів. Значення x, яке є виходом останнього повністю зв'язаного шару, повертається як вихід функції forward.

Метод validation_step обчислює втрати під час валідації (лістинг 3.5). Він приймає партію даних (batch) та її індекс (batch_idx). Дані конвертуються у тип float та переміщаються на пристрій. Модель робить прогнозування для вхідних даних. Обчислюється втрата між прогнозуванням та очікуваними значеннями. Втрата записується у словник logs та повертається.

Лістинг 3.5 – Обчислення втрат

```
def validation_step(self, batch, batch_idx):
        x, y = [i.float().to(DEVICE) for i in batch]
        x_pred = self(x)
        loss = self.criterion(x_pred, y.reshape(-1,1))
        logs = {
        'val_loss': loss
          }
         return logs
```

Також для кожної з моделей використовується оптимізатор Adam (лістинг 3.6), який допомагає прискорити навчання та поліпшити збіжність моделі. Adam (torch.optim.Adam) оновлює параметри нейронної мережі під час навчання. Одним з параметрів є amsgrad, який вказує на використання методу адаптивного ковзного середнього для коригування швидкості навчання. Цей метод використовується для покращення стійкості та збіжності оптимізатора. AMSGrad зберігає максимальне значення квадрату експоненціального рухомого середнього градієнту на кожному кроці оптимізації. Це дозволяє

уникнути проблеми з падінням норми градієнту, яка може виникнути в оригінальному методі Adam.

Лістинг 3.6 – Налаштування оптимізатора Adam

```
def configure_optimizers(self):
        self.optimizer = torch.optim.Adam(self.parameters(),
lr=LR, betas= (0.9,0.999), weight_decay= 5e-7, amsgrad=True) #,
betas= (0.9,0.999), weight_decay= 5e-7, amsgrad=True
        self.scheduler =
torch.optim.lr_scheduler.StepLR(self.optimizer, step_size=3,
gamma=0.6)
        return [self.optimizer], [self.scheduler]
```

Далі створено об'єкт Trainer (лістинг 3.7) з двома параметрами навчання: max_epochs вказує на максимальну кількість епох для навчання, а accelerator='gpu' вказує, що прискорювач, який буде використовуватися для навчання – це графічний процесор. GPU мають велику кількість ядер, які можуть виконувати паралельні обчислення. Це робить їх ефективнішими для обробки великих обсягів даних та виконання складних обчислень, таких як навчання нейронних мереж.

Лістинг 3.7 – Процес навчання моделі

```
net = CnnNet()
torch.backends.cudnn.benchmark = True
trainer = pl.Trainer(max_epochs=EPOCHS, accelerator='gpu')
trainer.fit(net)
```

trainer.fit(net) починає процес навчання моделі net. trainer.fit використовує дані для навчання моделі, яка передається в якості аргументу net. Вона автоматично виконує кроки навчання на протязі вказаної кількості епох (max_epochs), обчислює втрати, здійснює зворотнє поширення та оновлення ваг моделі, щоб зменшити ці втрати.

4 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

4.1 Параметри та базові налаштування нейронних мереж

У ході дослідження було створено нейронні мережі на основі CNN та LSTM архітектур із застосуванням різних гіперпараметрів. Гіперпараметри – це параметри, які визначають архітектуру та умови навчання моделі, а не її ваги, які змінюються під час тренування. Визначення правильних значень для гіперпараметрів має значний вплив на результати навчання моделі, її швидкість навчання та загальну ефективність.

При навчанні нейронних мереж використовувались наступні параметри:

- batch_size – це параметр, який визначає розмір партії даних, які модель оброблює за один прохід навчання. Збільшення цього параметра ускладнює навчання та зменшує ефект перенавчання, що погіршує результати валідації;
- lr – це параметр, який визначає швидкість навчання (learning rate) , він вказує на те, наскільки швидко ваги моделі оновлюються під час тренування;
- epochs – це параметр, який визначає кількість епох, або проходів, протягом яких модель буде навчатися на всьому тренувальному наборі даних. Одна епоха навчання передбачає обробку усіх даних;
- input_size – це параметр, який визначає кількість вхідних ознак, очікуваних на вході до LSTM шару, вказує розмірність вхідних даних для кожного часового кроку;
- hidden_size – параметр, що визначає розмір прихованих станів та вихідних векторів LSTM шару;
- bidirectional – це параметр, що вказує, чи буде LSTM шар двонаправленим. Якщо bidirectional=True, LSTM навчатиметься та враховуватиме інформацію як у прямому, так і у зворотному напрямку по часовій осі. У цьому випадку розмірність вихідного тензора буде подвоєна, тобто hidden_size*2 для кожного часового кроку. В дослідженні bidirectional =

False, тому LSTM буде односпрямованим, і розмір вихідного тензора залишиться hidden_size;

- kernel_size – параметр, який вказує на розмір згорткового ядра в мережі CNN;
- poolsize – визначається розмір вікна до виконання операції підвибірки (пулінгу);
- padding – використовується для керування розмірами вихідних даних та обробки крайових ефектів;
- mse (Mean Squared Error) – середньоквадратична помилка, яка використовується у якості критерію оцінки роботи при начанні. Чим кращі результати показує нейронна мережа, тим менше значення середньоквадратичної помилки.

В якості базових конфігурацій штучних нейронних мереж були визначені наступні набори гіперпараметрів:

- LSTM: input_size – 100, hidden_size – 200, activation – LeakyReLU, dropout – 0.3.
- CNN: activation – ReLU; dropout – 0,3; kernelsize – 3, poolsize – 2; padding – 1.
- глобальні параметри: epochs – 15, batch_size – 5, lr – 0.0001.

4.2 Проведені дослідження

4.2.1 Залежність точності прогнозування від ступеню навчання

Щоб дослідити вплив параметрів batch_size, epochs та lr на прогнозування нейронних мереж, використовувалися нейронні мережі у базовій конфігурації (конфігурація описана у розділі 4.1).

Проведено серію експериментів для виявлення оптимальних налаштувань нейронної мережі. Першим досліджено параметр epochs та проведено 9 експериментів (таблиця 4.1).

Таблиця 4.1 – Вплив параметру epochs на результати прогнозування

№	Параметри			Значення Mean Squared Error	
	epochs	batch_size	lr	LSTM	CNN
1	5	5	0.0001	3.25	2.78
2	10	5	0.0001	1.79	2.03
3	15	5	0.0001	1.33	1.53
4	20	5	0.0001	1.45	1.87
5	25	5	0.0001	1.56	1.75
6	30	5	0.0001	1.58	1.83
7	35	5	0.0001	1.67	1.94
8	40	5	0.0001	1.98	2.55
9	45	5	0.0001	2.89	3.67

Побудовано графік за результатами експериментів дослідження впливу параметру epochs на точність прогнозування за допомогою штучних нейронних мереж (рисунок 4.1).

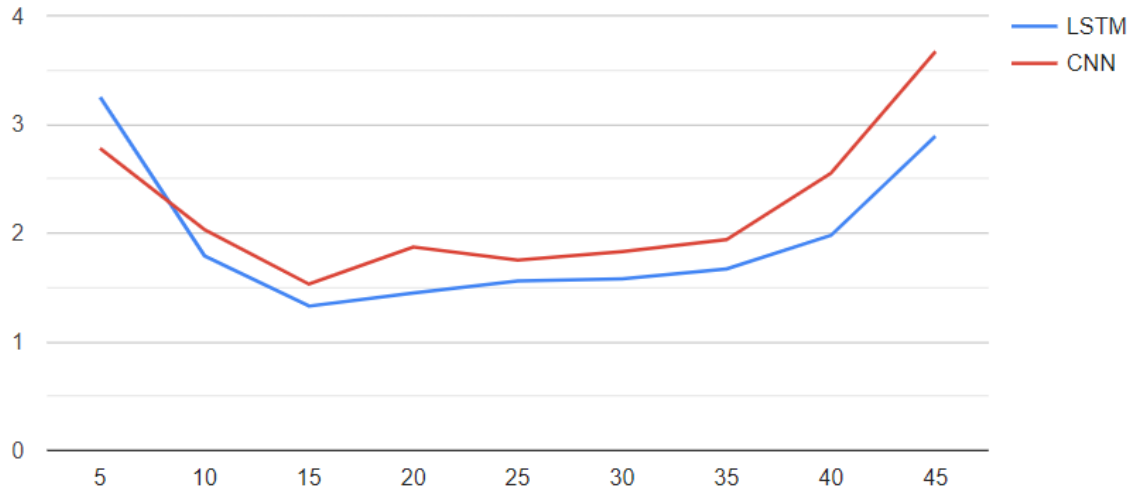

Рисунок 4.1 – Візуалізація залежності MSE від зміни параметру epochs

MSE зростає зі збільшенням числа епох для обох моделей. Це вказує на те, що модель стає менш здатною до узагальнення даних та схильною до

перенавчання зі збільшенням числа епох. Найкращий результат MSE нейронні мережі показують при epochs 15.

Досліджено вплив параметра lr та проведено 6 експериментів (таблиця 4.2). Параметр epochs обраний як найкращий результат з попереднього дослідження.

Таблиця 4.2 – Вплив параметру lr на результати прогнозування

№	Параметри			Значення Mean Squared Error	
	epochs	batch_size	lr	LSTM	CNN
1	15	5	0.000001	2.25	2.06
2	15	5	0.00001	1.78	1.89
3	15	5	0.0001	1.33	1.53
4	15	5	0.001	1.52	1.67
5	15	5	0.01	2.10	2.65
6	15	5	0.1	2.80	2.94

Побудовано графік за результатами дослідження впливу параметру ls на точність прогнозування за нейронних мереж (рисунок 4.2).

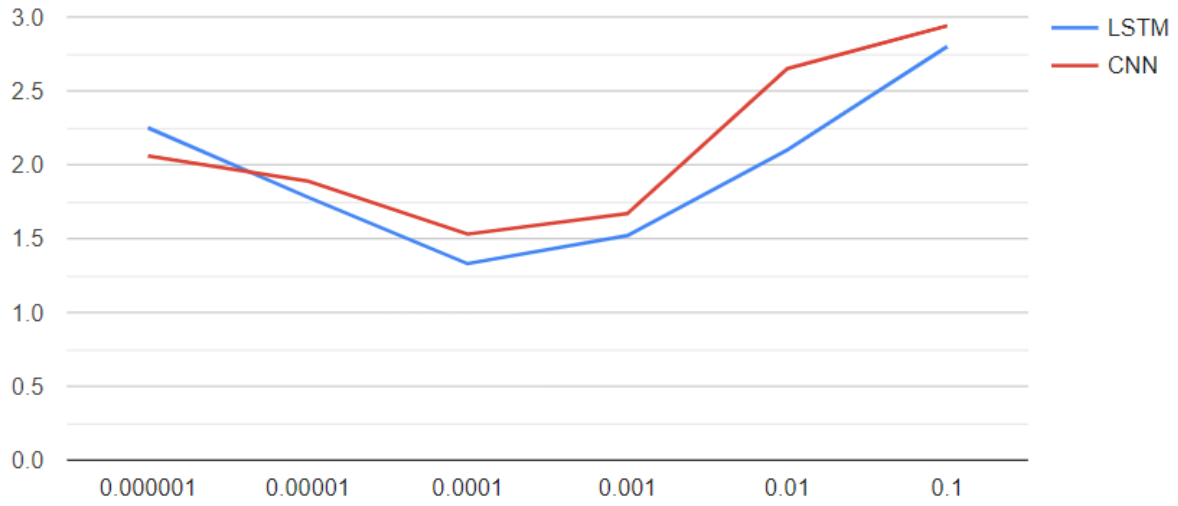

Рисунок 4.2 – Графік залежності MSE від зміни параметру lr

Зі збільшенням значення lr, MSE спочатку зменшується, але потім знову починає зростати. Це може вказувати на те, що при вищих значеннях швидкості навчання модель стає більш схильною до перенавчання.Для цього набору даних оптимальним виявляється значення lr 0.0001, у якому досягається найменше значення MSE для обох моделей. Значення MSE для моделей LSTM і CNN становлять 1.33 і 1.53 відповідно.

Третім досліджено параметр batch_size та проведено 3 експерименти. Параметри epochs та lr обрані як найкращий результат попередніх експериментів.

Таблиця 4.3 – Вплив параметру batch_size на результати прогнозування

№	Параметри			Значення Mean Squared Error	
	epochs	batch_size	lr	LSTM	CNN
1	15	1	0.0001	1.62	1.70
2	15	5	0.0001	1.33	1.53
3	15	10	0.0001	1.45	1.67

Побудована діаграма, в якій візуалізована залежність Mean Squared Error від параметру batch_size (рисунок 4.3).

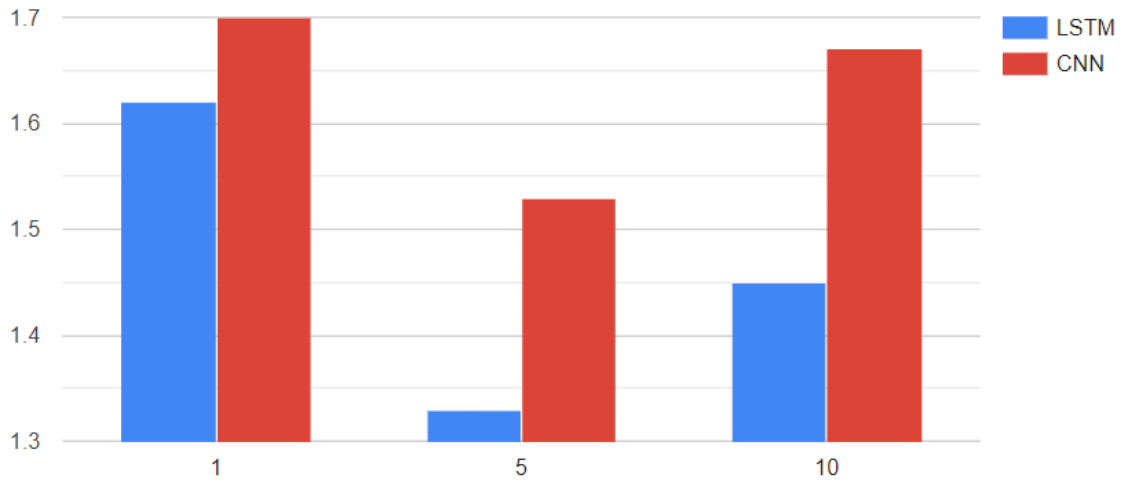

Рисунок 4.3 – Візуалізація залежності MSE від зміни параметру batch_size

При batch size рівному 10, моделі LSTM та CNN показують найвищі значення MSE. При зменшенні batch size до 5, значення MSE значно зменшується для обох моделей. При подальшому зменшенні batch size до 1, значення MSE для обох моделей трохи збільшується. Найкращий результат MSE нейронні мережі показують при batch size 5.

4.2.2 Вплив гіперпараметрів на навчання моделі та результати прогнозування

Для дослідження впливу гіперпараметрів на навчання моделей та результати прогнозування були використані моделі з базовою конфігурацією, що описана в розділі 4.4.

Досліджено вплив параметрів hidden_size та dropout на MSE та час навчання для нейронної мережі LSTM (таблиця 4.4).

Таблиця 4.4 – Налаштування гіперпараметрів моделі для мережі LSTM

№	hidden_size	dropout	MSE	Час навчання
1	100	0,2	1.14	9
2	200	0,2	1.13	8
3	300	0,2	1.25	15
4	400	0,2	1.26	15
5	100	0,3	1.12	8
6	200	0,3	1.12	9
7	300	0,3	1.15	14
8	400	0,3	1.18	11
9	100	0.4	1.36	14
10	200	0.4	1.51	14
11	300	0.4	1.23	18
12	400	0.4	1.56	19

Збільшення розміру прихованого шару hidden_size дозволяє моделі більш точно апроксимувати складніші функції, що може призвести до зниження значення MSE. Проте в таблиці для hidden_size 300 та 400, MSE не зменшується далі, а зростає, що свідчить про те, що модель починає перенавчатися на тренувальних даних. Таким чином, для цього набору даних оптимальним може бути значення hidden_size в діапазоні 100–200. Для значень dropout 0.4 MSE знову зростає, що пов'язано з тим, що велике значення dropout призводить до втрати важливої інформації під час навчання.

Виходячи з результатів представлених в таблиці оптимальними параметрами для моделі є hidden_size=100 та dropout=0,3, з якими модель демонструє найменше значення MSE.

Далі досліджено вплив параметрів kernel_size та dropout на MSE та час навчання для нейронної мережі CNN (таблиця 4.5).

Таблиця 4.5 – Вплив гіперпараметрів на час навчання та точність прогнозу за допомогою штучної нейронної мережі на основі CNN

№	dropout	kernel_size	MSE	Час навчання, хвилин
1	0.2	2	1.73	11
2	0.2	3	1.68	12
3	0.2	4	1.65	13
4	0.2	5	1.67	14
5	0.3	2	1.50	11
6	0.3	3	1.62	12
7	0.3	4	1.58	13
8	0.3	5	1.60	14
9	0.4	2	1.82	11
10	0.4	3	1.93	12
11	0.4	4	1.90	13
12	0.4	5	1.92	14

При значенні dropout 0.2 зменшується значення MSE при збільшенні значень kernel_size від 2 до 4, але потім MSE знову зростає при kernel_size 5. Це може вказувати на те, що для цього діапазону значень kernel_size модель CNN краще пристосовувалася до великих значень dropout, але з більшим значенням kernel_size вона починає втрачати узагальнюючу здатність і виникає перенавчання.

Виходячи з результатів представлених в таблиці, оптимальними параметрами для моделі є kernel_size=3 та dropout=0,3, з якими модель демонструє найменше значення MSE.

Зміна гіперпараметрів в нейронних мережах LSTM та CNN значно впливає на їхню ефективність та продуктивність. Наприклад, встановлення більших значень dropout допомогає запобігти перенавчанню, але при цьому може призвести до втрати важливої інформації під час навчання. Зміна значень kernel_size може впливати на здатність моделі виявляти локальні та глобальні залежності в даних. Параметр hidden_size визначає розмірність внутрішніх представлень даних та впливає на здатність моделі до вивчення складних залежностей у часовому ряді.

4.3 Візуальний аналіз якості прогнозування

Для візуального аналізу якості прогнозування вибрано конфігурації нейронних мереж, у яких за результатами експериментів було найменше значення MSE.

Вибрано наступні конфігурації:

- LSTM (Dropout = 0.3, Activation = LeakyReLU, hidden_size = 100). Досягнуте значення MSE дорівнює 1.12;
- CNN (KernelSize = 3, PoolSize = 2, Dropout = 0.3, Activation = ReLU). Досягнуте значення MSE дорівнює 1.50.

Побудовано 2 таблиці (додаток Б) з візуалізацією метрик MAE та MAPE для LSTM та CNN, обчислених на основі реальних даних та тих, що були

спрогнозовані глибокими нейронними мережами. Фрагменти обчислених метрик для мереж LSTM (таблиця 4.6) та CNN (таблиця 4.7) представлені нижче у таблицях.

Таблиця 4.6 – Обчислення метрик для LSTM

№	Реальне значення	Спрогнозоване значення	AE	APE
1	29684540	25175888	4508652	15.19%
2	24569825	22971505	1598320	6.505%
3	12262005	12643378	381373	3.11%
4	32739612	32578563	161049	0.492%
5	14965999	15765654	799655	5.34%

Середня абсолютна похибка для LSTM при обчислені дорівнює 1502387.5 та середня абсолютна похибка у відсотках дорівнює 6.9815%.

Таблиця 4.7 – Обчислення метрик для CNN

№	Реальне значення	Спрогнозоване значення	AE	APE
1	29684540	28194180	1490360	5.01%
2	24569825	26971505	2401680	9.77%
3	12262005	12251564	10441	0.0852%
4	32739612	31367327	1372285	4.19%
5	14965999	20474824	5508825	36.81%

Середня абсолютна похибка для CNN дорівнює 2182185.4, а середня абсолютна похибка у відсотках дорівнює 8.919%.

На графіку (рисунок 4.4) наведені фрагменти часових рядів, що відповідають реальним даним та тим, що були спрогнозовані нейронними мережами CNN та LSTM. Час до виверження вулкану виміряється в секундах.

Проаналізувавши результати прогнозування, наведені на рисунку 4.4, слід зазначити, що прогнози LSTM слідують трендам реальних даних, але

відрізняються згладженими коливаннями. В точках, де реальні дані мають спади, модель LSTM показує точніші прогнози, але відстає на піках. CNN показує більші відхилення на піках та спадах реальних даних.

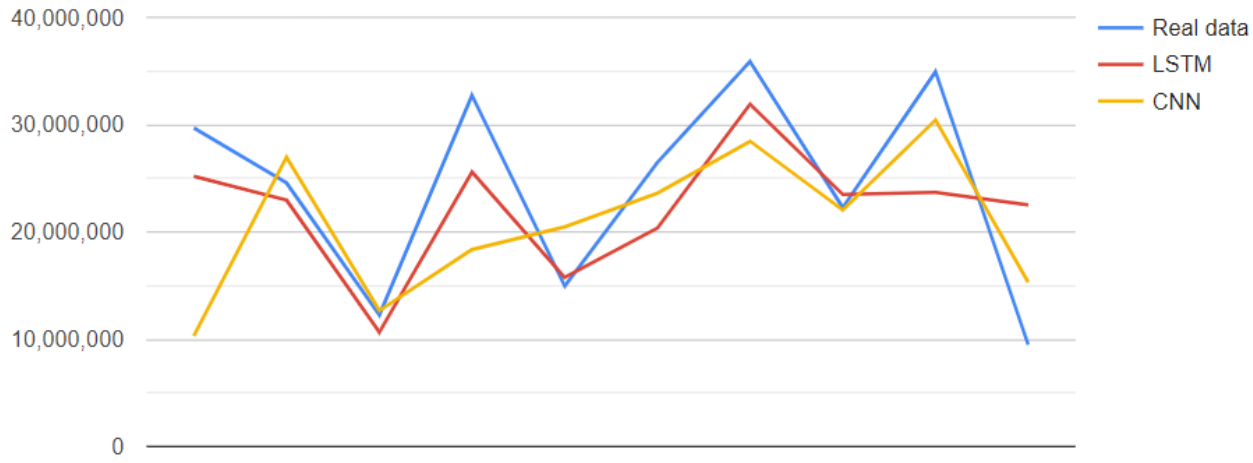

Рисунок 4.4 – Графік візуалізації реальних даних та спрогнозованих CNN та LSTM

Можна зробити висновок, що найкращі результати показує LSTM мережа. LSTM модель іноді недооцінює чи переоцінює час до виверження, але у середньому зберігає форму лінії реальних даних. Модель CNN демонструє меншу точність порівняно з LSTM. Обидва підходи LSTM і CNN демонструють здатність вловлювати загальні тенденції.

ВИСНОВКИ

У роботі досліджена і проаналізована проблема прогнозування сейсмічної активності вулканів з використанням моделей глибокого навчання. В якості вихідних даних використовувалися часові ряди, що були зібрані Італійським національним інститутом геофізики та вулканології впродовж 24-годинного моніторингу сейсмічності та вулканічної активності по всій країні.

Було проаналізовано більше 30 існуючих досліджень в області прогнозування сейсмічної активності вулканів та проведено більше 40 експериментів щодо використання штучних нейронних мереж для прогнозування виверженьь [44]. В результаті проведеного дослідження вибрано моделі глибокого навчання LSTM та CNN для прогнозування сейсмічної активності вулканів. Для дослідження і аналізу даних використовувалася мова програмування Python та середовище розробки PyCharm.

Дослідження включає аналіз даних про сейсмічну активність вулканів, створення та навчання моделей глибокого навчання на цих даних, а також аналіз отриманих результатів. Результати вказують на перевагу моделі LSTM. В порівнянні з CNN, LSTM забезпечує кращу точність у відтворенні реальних даних, проте обидва підходи демонструють здатність відображати загальні тенденції в даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bowerman, B. Forecasting, Time Series, and Regression (4th ed.) [Текст] / B. Bowerman, R. O'Connell, A. Koehler // Cengage Learning, 2005. – 720 p.

2. Nielsen, A. Practical Time Series Analysis: Prediction with Statistics and Machine Learning 1st Edition [Текст] /A. Nielsen // O'Reilly, 2019. – 506 p.

3. Аналіз та прогнозування часових рядів [Електронний ресурс] – Режим доступу : www/ URL: https://drukarnia.com.ua/articles/analiz-ta-prognozuvannya-chasovikh-ryadiv-H6gg0/ – 18.03.2024 р. – Загол. з екрану.

4. Brockwell, P. Introduction to Time Series and Forecasting (3rd ed.) [Текст] / P. Brockwell, R. Davis // Springer, 2016. – .428 p.

5. Chatfield, C. The Analysis of Time Series: An Introduction (7th ed.) [Текст] / C.Chatfield // Chapman and Hall/CRC, 2019. – 414 p.

6. Gelman, A. Regression and Other Stories [Текст] / A. Gelman, J. Hill, A. Vehtari // Cambridge University Press, 2020. – 546 p.

7. Hyndman, R. Forecasting: Principles and Practice (2nd ed.) [Текст] / R. J. Hyndman, G. Athanasopoulos // OTexts, 2018. – 512 p.

8. Shmueli, G. To Explain or to Predict? [Текст] / G. Shmueli // Statistical Science. – 2010. – Vol. 25, No. 3. – P. 289–310.

9. Kourentzes, N. Improving forecasting by estimating time series structural components across multiple frequencies [Текст] / N. Kourentzes, F. Petropoulos, J. Trapero // International Journal of Forecasting. – 2019. – Vol. 35, No 4. – P. 1225–1238.

10. Berberich, D. Hybrid Methods for Time Series Forecasting [Текст] / D. Berberich // Springer. – 2021. – 320 p.

11. Rouet-Leduc, B. Machine learning predicts laboratory earthquakes. [Текст] / B. Rouet-Leduc, C. Hulbert, N. Lubbers, K. Barros, C. Humphreys, P. Johnson // Geophys. Res. Lett. – 2017. – Vol. 44, No 1. – P. 9276 – 9282.

12. DeVries, PMR. Deep learning of aftershock patterns following large

earthquakes. [Текст] / PMR DeVries, F. Viégas, M. Wattenberg, BJ. Meade // Nature. – 2018. – Vol. 56. – P. 632–634.

13. Abri, R. LSTM-Based Deep Learning Methods for Prediction of Earthquakes Using Ionospheric Data [Текст] / R. Abri, H. Artuner // Journal of Science. – 2022. – P. 1417–1428.

14. Nomura, Y. Pilot study of eruption forecasting with muography using convolutional neural network [Текст] / Y. Nomura, M. Nemoto, N. Hayashi, S. Hanaoka, M. Murata, T. Yoshikawa, Y. Masutani, E. Maeda, O. Abe, K. Hiroyuki, M. Tanaka // Scientific Reports. – 2020. – P. 1–13.

15. Nielsen, M. Neural Networks and Deep Learning [Текст] / M. Nielsen // Determination Press. – 2015. – 293 p.

16. Goodfellow, I. Deep Learning [Текст] / I. Goodfellow, Y. Bengio, A. Courville // MIT Press. – 2016. – 800 p.

17. Neural Network Principles and Applications [Електронний ресурс] – Режим доступу : www/ URL: https://www.intechopen.com/chapters/63906 – 18.03.2024 р. – Загол. з екрану.

18. What is a neural network? [Електронний ресурс] – Режим доступу : www/ URL: https://www. cloudflare.com/learning/ai/what-is-neural-network – 18.03.2024 р. – Загол. з екрану

19. Géron, A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow [Текст] / A. Géron // O'Reilly Media. – 2019. – 510 p.

20. Tariq, U. Neural Networks for Beginners: An Easy Guide to Understanding Artificial Intelligence, Machine Learning, and Deep Learning [Текст] / U. Tariq // Independently published. – 2020. – 174 p.

21. Karpathy, A. Deep Learning for Beginners: Concepts, Techniques, and Tools [Текст] / A. Karpathy // Independently published. – 2021. – 231 p.

22. Alzubaidi, L. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions [Текст] / L. Alzubaidi, J. Zhang, A. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. Fadhel, M. Amidie, L. Farhan // Journal of Big

Data. – 2021. – P. 2–74.

23. LeCun, Y. Deep learning [Текст] / Y. LeCun, Y. Bengio, G. Hinton // nature. – 2015. – Vol. 521, No 7553. – P. 436–444.

24. Shrestha, A. Review of deep learning algorithms and architectures [Текст] / A. Shrestha, A. Mahmood // IEEE access. – 2019. – Vol. 7. – P. 53040-53065.

25. Najafabadi, M. Deep learning applications and challenges in big data analytics [Текст] / M. Najafabadi //Journal of big data. – 2015. – Vol. 2. – P. 1–21.

26. Aggarwal, C. Neural Networks and Deep Learning: A Textbook [Текст] / C. Aggarwal // Springer. – 2018. – 512 p.

27. Zhang, A. Dive into Deep Learning [Текст] / A. Zhang, Z. Lipton, M. Li, A. Smola // CRC Press. – 2023. – 574 p.

28. Patterson, J. Deep Learning: A Practitioner's Approach [Текст] / J. Patterson, A. Gibson // O'Reilly Media. – 2017. – 536 p.

29. Deep Learning [Електронний ресурс] – Режим доступу : www/ URL: https://www.techtarget.com/searchenterpriseai/definition/deep-learning-deep-neural-network – 18.03.2024 р. – Загол. з екрану.

30. Що таке персептрон? [Електронний ресурс] – Режим доступу : www/ URL: https://towardsdatascience.com/what-the-hell-is-perceptron-626217814f53– 18.03.2024 р. – Загол. з екрану.

31. What is Perceptron: A Beginners Guide for Perceptron [Електронний ресурс] – Режим доступу : www/ URL: https://www.simplilearn.com/ tutorials/deep-learning-tutorial/perceptron – 18.03.2024 р. – Загол. з екрану.

32. Rashid, T. Make Your Own Neural Network [Текст] / T. Rashid // Createspace Independent Pub. – 2016. – 222 p.

33. Brownlee, J. Long Short-Term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning [Текст] / J. Brownlee // Machine Learning Mastery. – 2017. – 246 p.

34. Howard, J. Practical Deep Learning for Coders: Building Real World AI Applications with LSTM Networks [Текст] / J. Howard, S. Gugger // O'Reilly Media. – 2020. – 622 p.

35. What is LSTM? [Електронний ресурс] – Режим доступу : www/ URL: https://www.analyticsvidhya.com/blog /2021/03/introduction-to-long-short-term-memory-lstm/ – 18.03.2024 р. – Загол. з екрану.

36. Zhang, Y. Hands-On LSTM for Time Series Forecasting: A Practical Guide to Implementing Long Short-Term Memory Models with Python [Текст] / Y. Zhang // Packt Publishing. – 2019. – 106 p.

37. Li, S. Time Series Prediction Using LSTM Deep Neural Networks [Текст] / S. Li, Y. Zhang, Q. Li, W. Li // IEEE International Conference on Big Data. – 2016. – P. 265–274

38. Hochreiter, S. Long Short-Term Memory [Текст] / S. Hochreiter, J. Schmidhuber // Neural Computation. – 1997. – Vol. 9, No. 8. – P. 1735–1780.

39. Brownlee, J. Deep Learning for Computer Vision: Image Classification, Object Detection, and Face Recognition in Python [Текст] / J. Brownlee // Machine Learning Mastery. – 2019. – 563 p.

40. A Comprehensive Guide to Convolutional Neural Networks – the ELI5 way [Електронний ресурс] – Режим доступу : www/ URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53 – 18.03.2024 р. – Загол. з екрану.

41. Rashid, T. Learning TensorFlow: A Guide to Building Deep Learning Systems / T. Rashid // O'Reilly Media, Inc. – 2017. – 228 p.

42. Convolutional Neural Networks, Explained [Електронний ресурс] – Режим доступу : www/ URL: https://towardsdatascience.com/convolutional-neural-networks-explained9cc5188c 4939 – 18.03.2024 р. – Загол. з екрану.

43. Dropout: A Powerful Regularization Technique for Deep Neural Networks [Електронний ресурс] – Режим доступу : www/ URL: https://www.linkedin.com/dropout-powerful – 18.03.2024 р. – Загол. з екрану.

44. 1. Іващенко, Г. С. Моделі глибокого навчання для прогнозування часових рядів [Текст] / Г. С. Іващенко, Д. О. Тимошенко, О. В. Близнюк, О. М. Кононенко // Системи управління, навігації та зв’язку. – 2024. – №1. – С. 82–87.