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 еволюційної нейронної мережі в задачах прогнозування в режимі реального часу
(тема)

Виконав:
здобувач другого року навчання,
групи СШІм-23-2
Ілля Ушаков
(власне ім'я, прізвище)

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 освітньої програми)
Керівник доц. Олег Золотухін
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

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
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ушакову Іллі Романовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження еволюційної нейронної мережі в задачах прогнозування в режимі реального часу

затверджена наказом університету від 21 квітня 20 25 р. № 295Ст

2. Термін подання студентом роботи до екзаменаційної комісії 6 червня 20 25 р.

3. Вихідні дані до роботи Дані науково-технічних публікацій щодо методів щодо методів прогнозування в реальному часі. Різноманітні інтернет-джерела з рентгенівськими знімками. Документація Python.

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі та постановка задачі
2) Теоретичний огляд методів прогнозування в реальному часі
3) Проєктування інформаційних моделей та розроблення алгоритмів прогнозування в реальному часі
4) Розроблення інформаційних моделей та розроблення алгоритмів прогнозування в реальному часі

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	21.04.2025	виконано
2	Аналіз предметної галузі та постановка задачі	23.04.2025	виконано
3	Аналіз класичних статистичних підходів для	26.04.2025	виконано
	прогнозування в реальному часі		
4	Аналіз нейромережевих підходів для	29.04.2025	виконано
	прогнозування часових рядків		
5	Аналіз еволюційних алгоритмів в навчанні	02.05.2025	виконано
	нейромереж		
6	Проектування інформаційних моделей та	06.05.2025	виконано
	алгоритмів для прогнозування в реальному часі		
7	Розробка інформаційних моделей та алгоритмів	09.05.2025	виконано
	для прогнозування в реальному часі		
8	Оформлення пояснювальної записки	11.05.2025	виконано
9	Подача роботи та підготовка до захисту	12.05.2025	виконано
10	Попередній захист	31.05.2025	виконано
11	Захист перед ЕК	06.06.2025	

Дата видачі завдання 21 квітня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Олег Золотухін
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 68 с., 11 рис., 1 дод., 44 джерела.

АДАПТИВНЕ НАЛАШТУВАННЯ, АЛГОРИТМИ ЕВОЛЮЦІЇ, ГІБРИДНІ МОДЕЛІ, ЕВОЛЮЦІЙНІ НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, ОПТИМІЗАЦІЯ, ПРОГНОЗУВАННЯ, РЕАЛЬНИЙ ЧАС, ЧАСОВІ РЯДИ.

Об’єкт дослідження – процес прогнозування часових рядів у режимі реального часу з використанням еволюційних нейронних мереж.

Предмет дослідження – методологія побудови еволюційних нейронних мереж для задач прогнозування в режимі реального часу, що включає адаптивну оптимізацію параметрів та структури моделей за допомогою еволюційних алгоритмів.

Мета дослідження – розробка та експериментальне обґрунтування підходу до прогнозування часових рядів у реальному часі на основі еволюційних нейронних мереж із використанням методів оптимізації гіперпараметрів і топології мережі, що забезпечує підвищену точність та швидкість адаптації моделей.

Методи дослідження – методи машинного навчання для побудови та оптимізації прогнозних моделей, еволюційні алгоритми для налаштування архітектури нейромереж, методи математичної статистики для оцінки точності експериментальні дослідження ефективності навчання моделей.

Наукова новизна роботи – запропоновано підхід до прогнозування часових рядів у режимі реального часу, який поєднує механізми еволюційного навчання та адаптивної оптимізації нейронних мереж.

Галузь застосування – системи прогнозування у фінансовій аналітиці, енергетиці, промисловості, охороні здоров’я та інших сферах, де критично важливе точне та швидке прогнозування часових рядів у реальному часі.

ABSTRACT

Master’s thesis contains: 68 pp., 11 fig., 1 ann., 44 references.

ADAPTIVE TUNING, EVOLUTION ALGORITHMS, HYBRID MODELS, EVOLUTIONARY NEURAL NETWORKS, MACHINE LEARNING, OPTIMIZATION, FORECASTING, REAL-TIME, TIME SERIES

The object of the study is the process of forecasting time series in real time using evolutionary neural networks.

The subject of the study is the methodology for building evolutionary neural networks for real-time forecasting tasks, which includes adaptive optimization of parameters and model structure using evolutionary algorithms.

The purpose of the research is to develop and experimentally substantiate an approach to real-time time series forecasting based on evolutionary neural networks using hyperparameter optimization methods and network topology, which provides increased accuracy and speed of model adaptation.

Research methods – machine learning methods for building and optimizing forecast models, evolutionary algorithms for tuning the architecture of neural networks, mathematical statistics methods for assessing the accuracy of forecasts, experimental studies of the effectiveness of model training.

Scientific novelty of the work – an approach to real-time time series forecasting is proposed, which combines the mechanisms of evolutionary learning and adaptive optimization of neural networks, which allows to increase the efficiency of modeling and the speed of response to data changes.

Field of application – forecasting systems in financial analytics, energy, industry, healthcare and other areas where accurate and fast real-time time series forecasting is critically important.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 8
Вступ 9
1 Аналіз предметної галузі та постановка задачі 11
1.1 Опис предметної галузі 11
1.1.1 Прогнозування часових рядів у режимі реального часу 11
1.1.2 Виклики та обмеження класичних методів прогнозування 12
1.1.3 Переваги нейронних мереж у задачах прогнозування 13
1.2 Еволюційні нейронні мережі в прогнозуванні часових рядів 16
1.2.1 Основні концепції еволюційного навчання нейромереж 16
1.2.2 Використання еволюційних алгоритмів для налаштування нейронної мережі 17
1.2.3 Оптимізація топології нейромереж за допомогою еволюції 19
1.3 Актуальність та можливе застосування 20
1.4 Постановка задачі 21
2 Теоретичний огляд методів прогнозування в реальному часі 23
2.1 Класичні статистичні підходи 23
2.1.1 Метод ARIMA 23
2.1.2 Метод SARIMA 25
2.2 Нейромережеві підходи до прогнозування часових рядів 26
2.2.1 Рекурентні нейронні мережі (RNN, LSTM, GRU) 26
2.2.2 Архітектури трансформерів для прогнозування 28
2.3 Еволюційні алгоритми в навчанні нейромереж 30
2.3.1 Генетичні алгоритми для налаштування гіперпараметрів 30
2.3.2 Коеволюційні методи оптимізації нейромереж 31
2.3.3 Алгоритм диференційної еволюції для прогнозування 34
3 Проєктування інформаційних моделей та розроблення алгоритмів прогнозування в реальному часі 36
3.1 Порівняльний аналіз методів аналізу великих масивів даних 36

3.2 Вибір та обгрунтування методів проєктування 39
3.3 Проєктування та розробка алгоритму системи..... 41
3.4 Архітектура обчислювальної моделі прогнозування на основі LSTM..... 47
4 Розроблення інформаційних моделей та розроблення алгоритмів прогнозування в реальному часі..... 50
4.1 Вибір та обгрунтування операційної системи 50
4.2 Вибір та обгрунтування середовища розробки..... 52
4.3 Вибір та обгрунтування технологій розробки 54
4.4 Процес програмної реалізації 56
4.5 Аналіз отриманих результатів 59
Висновки..... 61
Перелік джерел посилання 63
Додаток А Відомість кваліфікаційної роботи 68

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AR – Autoregressive model – авторегресія;

ARIMA – Autoregressive Integrated Moving Average – авторегресійне інтегроване ковзне середнє;

ARIMAX – AutoRegressive Integrated Moving Average extended – авторегресійне інтегроване ковзне середнє розширене;

DL – Deep Learning – глибоке навчання;

DNN – Deep beural network – глибинні нейронні мережі;

F1 – F1-score – середнє гармонійне значення пам'яті та точності;

GRU – Gated Recurrent Unit – закритий рекурентний блок;

LSTM – Long Short-Term Memory – довга короткочасна пам'ять;

MA – Moving average – метод ковзного середнього;

ML – Machine Learning – машинне навчання;

RNN – Recurrent neural network – рекурентні нейронні мережі;

SARIMA – Seasonal Autoregressive Integrated Moving Average – сезоне авторегресійне інтегроване ковзне середнє.

ВСТУП

Актуальність дослідження зумовлена необхідністю удосконалення методів прогнозування часових рядів у режимі реального часу, що є важливим для різних галузей, таких як фінансові ринки, енергетика, охорона здоров’я, транспорт, тощо. Прогнозування в реальному часі вимагає швидкого та точного аналізу величезних масивів даних, які постійно змінюються. Традиційні статистичні методи, зокрема ARIMA, MA, та інші, не завжди є ефективними для таких завдань через свою обмежену здатність адаптуватися до динамічних змін у даних. У зв’язку з цим, існує необхідність у застосуванні більш потужних методів, таких як нейронні мережі, які здатні враховувати складні нелінійні залежності в даних. Зокрема, еволюційні нейронні мережі, що поєднують можливості еволюційних алгоритмів з глибокими нейронними мережами, надають нові можливості для підвищення точності прогнозів у реальному часі.

Об’єктом дослідження є процес прогнозування часових рядів у режимі реального часу за допомогою еволюційних нейронних мереж, а предметом дослідження є розробка та оптимізація методів еволюційного навчання нейронних мереж для розв’язання задач прогнозування. Враховуючи актуальність задачі прогнозування в умовах реального часу, дослідження зосереджене на застосуванні еволюційних алгоритмів для налаштування параметрів і архітектури нейронних мереж, що дозволяє досягати високої адаптивності і точності прогнозування в умовах постійних змін.

Метою даного дослідження є розробка та впровадження методів еволюційного навчання нейронних мереж для прогнозування часових рядів у режимі реального часу, а також оптимізація параметрів і топології таких мереж. Особлива увага приділяється використанню еволюційних алгоритмів для адаптації архітектури нейронних мереж до специфічних особливостей даних і умов їх зміни в реальному часі. Очікувані результати дослідження включають підвищення точності прогнозів та здатності

моделей до адаптації в умовах швидко змінюваних даних, що є важливим для практичних застосувань у різних галузях.

Наукова новизна дослідження полягає в розробці нових підходів до застосування еволюційних нейронних мереж для прогнозування в режимі реального часу, зокрема в оптимізації архітектури нейронних мереж за допомогою еволюційних алгоритмів. Це дозволяє досягти більш високої точності прогнозів у порівнянні з традиційними методами, зберігаючи при цьому здатність до адаптації до нових, невідомих умов. Застосування еволюційних методів оптимізації дозволяє автоматизувати процес налаштування мереж, що є важливим для реального застосування в динамічних середовищах.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної галузі

1.1.1 Прогнозування часових рядів у режимі реального часу

Прогнозування часових рядів є однією з ключових задач у різних сферах науки та бізнесу, включаючи фінансові ринки, енергетику, охорону здоров'я, виробництво та інтернет-речі (IoT). У класичному варіанті прогнозування передбачає аналіз минулих даних і побудову моделей для передбачення майбутніх значень на основі історичних закономірностей. Однак, сучасні системи, що працюють у режимі реального часу, вимагають швидкого оновлення прогнозів за наявності нових вхідних даних, що значно ускладнює використання традиційних методів [1].

Прогнозування в реальному часі має ряд важливих особливостей:

– обмеження за часом на оновлення прогнозів. У багатьох критичних системах, таких як автоматизовані біржові торги або системи контролю виробництва, затримка в обчисленнях навіть на кілька секунд може спричинити суттєві фінансові або технічні втрати;

– динамічна зміна статистичних характеристик ряду. Деякі часові ряди, наприклад, рівень попиту на продукцію або електричне навантаження, можуть змінювати свої закономірності в залежності від зовнішніх факторів. Методи прогнозування в реальному часі повинні адаптуватися до цих змін без потреби повного перенавчання моделі;

– висока швидкість оновлення даних. У деяких застосуваннях, зокрема в аналізі потокових даних (streaming data), нові спостереження надходять із великою частотою. Наприклад, у фінансових операціях зміни котирувань акцій можуть відбуватися тисячі разів на секунду;

– висока варіативність та нестабільність даних. На відміну від стаціонарних рядів, дані у реальному часі можуть містити імпульсні викиди,

тренди, сезонні флуктуації та інші аномалії, що ускладнює їх прогнозування традиційними методами;

– потреба в ефективних обчислювальних алгоритмах. Використання класичних методів, таких як ARIMA, може бути неефективним у реальному часі через їхню високу обчислювальну складність. Це зумовлює необхідність розробки нових підходів, що здатні швидко адаптуватися до змін даних та забезпечувати високу точність прогнозування.

Для вирішення цих викликів у сучасних дослідженнях активно використовуються методи машинного навчання та нейронні мережі, зокрема рекурентні (LSTM, GRU), трансформерні архітектури (Temporal Fusion Transformer) та еволюційно-оптимізовані нейронні мережі [2].

1.1.2 Виклики та обмеження класичних методів прогнозування

Класичні методи прогнозування часових рядів, такі як авторегресія (AR), метод ковзного середнього (MA), авторегресійно-інтегроване ковзне середнє (ARIMA) та їхні модифікації (SARIMA, ARIMAX), використовуються в багатьох сферах, зокрема в економіці, енергетиці, метеорології та логістиці [3]. Проте вони мають низку обмежень, які ускладнюють їх застосування в режимі реального часу.

Основні проблеми, що виникають у класичних методів прогнозування:

– лінійність моделей. Класичні моделі, такі як ARIMA, працюють на припущенні лінійності взаємозв’язків у даних. Це значно обмежує їхню здатність виявляти складні залежності та нелінійні взаємозв’язки, які характерні для багатьох реальних процесів;

– чутливість до стаціонарності. Для коректної роботи класичні методи потребують стаціонарних часових рядів, тобто таких, де середнє значення, дисперсія та автокореляційні характеристики не змінюються з часом. Проте

більшість реальних часових рядів є нестабільними, що змушує проводити попередню обробку, наприклад, диференціювання або нормалізацію;

– низька адаптивність до змін у даних. Класичні методи погано справляються із проблемою зміни концепції (Concept Drift), коли характерні закономірності ряду змінюються з часом. Їх необхідно постійно перенавчати, що є ресурсозатратним процесом;

– обмеженість у роботі з великими обсягами даних. Класичні підходи зазвичай не здатні ефективно обробляти великі потоки даних у режимі реального часу, оскільки їхні алгоритми прогнозування потребують виконання складних розрахунків;

– проблеми з урахуванням зовнішніх факторів. Хоча розширені варіанти (ARIMAX, SARIMAX) дозволяють використовувати додаткові змінні, вони не завжди ефективно працюють, коли зовнішні фактори мають нелінійний або стохастичний вплив на прогнозовані значення [4].

Через ці обмеження класичні методи поступаються сучасним підходам, зокрема нейронним мережам та методам гібридного машинного навчання, які дозволяють ефективно працювати з великими обсягами даних у режимі реального часу.

1.1.3 Переваги нейронних мереж у задачах прогнозування

Нейронні мережі є потужним інструментом для аналізу та прогнозування часових рядів, особливо у випадках, коли класичні методи, такі як ARIMA, SARIMA або експоненційне згладжування, демонструють обмежену ефективність. Завдяки своїй здатності моделювати складні нелінійні залежності, працювати з великими обсягами даних та адаптуватися до змін середовища, нейронні мережі забезпечують значні переваги у задачах прогнозування в режимі реального часу [5].

Більшість традиційних статистичних методів припускають лінійність часових рядів або вимагають значних перетворень даних для приведення їх

до стаціонарного вигляду. Нейронні мережі не мають таких обмежень, оскільки можуть автоматично навчатися складним взаємозв’язкам між вхідними змінними, незалежно від їхньої природи:

– глибинні нейронні мережі (DNN) можуть використовувати багатошарові архітектури, що дозволяє моделювати складні взаємозалежності між вхідними даними;

– рекурентні нейронні мережі (RNN), Long Short-Term Memory (LSTM) та Gated Recurrent Unit (GRU) зберігають історичний контекст і ефективно працюють з часовими залежностями;

– конволюційні нейронні мережі (CNN) можуть автоматично виявляти характерні патерни у часових рядах, що покращує точність прогнозування.

На відміну від класичних методів, які вимагають ручного вибору ознак, нейронні мережі здатні самостійно виявляти та використовувати релевантні патерни в даних [6]:

– автоенкодери та глибинні архітектури нейронних мереж можуть навчатися прихованим характеристикам даних, що допомагає зменшити необхідність попередньої обробки та очищення даних;

– нейронні мережі можуть використовувати додаткову контекстну інформацію, наприклад, вплив зовнішніх факторів (погода, економічні індикатори, соціальні тенденції) для покращення якості прогнозів.

Нейронні мережі можуть працювати в режимі онлайн-навчання, що дозволяє їм постійно оновлювати свої параметри без необхідності повного перенавчання моделі. Це особливо важливо у ситуаціях, де зміни у часових рядах відбуваються динамічно (наприклад, у фінансових ринках, системах енергоспоживання, мережевому моніторингу):

– методи трансферного навчання дозволяють оновлювати моделі нейронних мереж із мінімальними витратами ресурсів;

– архітектури з механізмом уваги (Attention Mechanism) дають змогу виділяти найбільш значущі часові точки вхідних даних, що покращує адаптивність моделей.

Реальні часові ряди часто містять викиди, шумові компоненти або пропущені значення. Багато класичних методів (наприклад, ARIMA) є чутливими до таких особливостей, що може призводити до значного зниження точності прогнозів. Нейронні мережі, особливо архітектури з регуляризацією та ансамблюванням, можуть ефективно боротися з такими проблемами:

– використання Dropout-регуляризації дозволяє уникати перенавчання та підвищує узагальнювальну здатність моделі;

– глибинні нейронні мережі здатні заповнювати відсутні значення та коригувати аномальні дані на основі внутрішніх закономірностей у часовому ряді.

Нейронні мережі можуть бути інтегровані з іншими методами прогнозування для досягнення ще вищої точності.

– гібридні моделі (Hybrid Models) поєднують класичні статистичні методи з нейронними мережами, що дозволяє компенсувати недоліки кожного з підходів;

– ансамблеві методи (Bagging, Boosting, Stacking) можуть поєднувати кілька нейромережевих моделей для підвищення точності та стійкості прогнозів.

Використання нейронних мереж у прогнозуванні часових рядів у режимі реального часу значно розширює можливості аналізу даних завдяки їхній здатності працювати з нелінійними залежностями, автоматично виділяти ознаки, адаптуватися до змін середовища та ефективно використовувати обчислювальні ресурси [7]. Завдяки застосуванню рекурентних нейронних мереж, механізмів уваги та глибоких архітектур, нейронні моделі стають невід'ємною частиною сучасних прогнозних

систем, забезпечуючи високу точність і гнучкість в умовах динамічних змін даних.

1.2 Еволюційні нейронні мережі в прогнозуванні часових рядів

1.2.1 Основні концепції еволюційного навчання нейромереж

Еволюційне навчання нейронних мереж є підходом, що використовує еволюційні алгоритми для оптимізації вагових коефіцієнтів, гіперпараметрів і топології штучних нейронних мереж. Цей метод моделює механізми біологічної еволюції, такі як природний відбір, мутації та рекомбінація, для автоматизованого пошуку ефективних архітектур і параметрів нейромереж.

Основні принципи еволюційного навчання нейромереж включають:

а) популяційний підхід. Еволюційні алгоритми працюють із множиною можливих рішень – популяцією, де кожен її елемент є певною реалізацією нейромережі. Популяція поступово покращується за рахунок механізмів відбору, мутацій та схрещування, що імітує процес природної еволюції;

б) генетичне кодування. Для представлення нейромережі у вигляді «геному» використовуються різні підходи:

– лінійне кодування. Ваги мережі та її параметри записуються у вигляді вектора чисел;

– графове кодування. Зв’язки між нейронами та шари мережі подаються у формі графа;

– матричне кодування. Використовується матриця зв’язків між нейронами, що дозволяє зберігати складніші архітектури;

в) оператори еволюції. Основні механізми, що керують процесом еволюційного навчання:

– відбір (Selection). Вибір найкращих особин за деякою функцією пристосованості (fitness function);

– мутація (Mutation). Випадкові зміни параметрів або структури мережі для дослідження нових рішень;

– схрещування (Crossover). Комбінування параметрів двох або більше особин для отримання нового покоління;

г) функція пристосованості (Fitness Function). Оцінка кожної нейромережі в популяції проводиться за допомогою функції пристосованості, яка відображає її здатність до розв'язання поставленої задачі (наприклад, точність прогнозу часових рядів).

Ці концепції дозволяють еволюційним нейромережам автоматично адаптуватися до складних даних і знаходити оптимальні параметри для задач прогнозування в реальному часі.

1.2.2 Використання еволюційних алгоритмів для налаштування нейронної мережі

Еволюційні алгоритми є потужними інструментами для налаштування параметрів нейронних мереж. Вони дозволяють автоматизувати процес оптимізації, що значно підвищує ефективність і точність нейронних мереж при вирішенні складних задач, таких як прогнозування часових рядів у реальному часі. Основним завданням еволюційних алгоритмів є пошук оптимальних значень параметрів, що дозволяють досягти високої точності прогнозу та ефективної адаптації до змін у вхідних даних [8].

Еволюційні алгоритми для налаштування нейронних мереж часто використовуються для оптимізації таких аспектів:

– ваги нейронної мережі. Стандартні методи навчання, такі як градієнтний спуск, можуть застрягти в локальних мінімумах або не забезпечити достатньо високу точність у складних задачах. Еволюційні алгоритми, в свою чергу, здійснюють пошук по всьому просторі можливих

рішень, що дозволяє уникнути цієї проблеми. Наприклад, кожен «індивід» в популяції може представляти різні варіанти ваг нейронної мережі, і за допомогою операторів схрещування і мутації, ці варіанти постійно покращуються, забезпечуючи кращі результати;

– гіперпараметри нейронної мережі. Вибір таких параметрів, як кількість шарів, кількість нейронів у кожному шарі, функції активації, тип навчального алгоритму, швидкість навчання є важливим аспектом для побудови ефективної нейронної мережі. Еволюційні алгоритми можуть автоматично оптимізувати ці гіперпараметри. Наприклад, кожен хромосом (рішення) у популяції може представляти конфігурацію нейронної мережі з різними параметрами. Еволюційний процес дозволяє знаходити найкращу комбінацію для задачі прогнозування;

– архітектура нейронної мережі. Еволюційні алгоритми можуть застосовуватися для автоматичного вибору топології мережі. Замість того, щоб вручну вибирати кількість шарів і нейронів, алгоритм може автоматично додавати або видаляти нейрони або шари, змінювати типи з’єднань між ними, що дозволяє ефективно адаптувати мережу до складних змінних даних.

Переваги використання еволюційних алгоритмів для налаштування нейронних мереж:

– уникнення локальних мінімумів. На відміну від традиційних методів, еволюційні алгоритми досліджують весь простір можливих рішень;

– автоматизація процесу налаштування. Не потрібно вручну підбирати оптимальні параметри, що значно знижує час і трудовитрати на навчання;

– гнучкість і масштабованість. Еволюційні алгоритми можуть бути застосовані для різноманітних типів нейронних мереж і задач, що робить їх універсальними для різних сферах.

1.2.3 Оптимізація топології нейромереж за допомогою еволюції

Однією з основних переваг еволюційних нейронних мереж є можливість оптимізації їх топології, тобто структури нейронної мережі. Це включає визначення кількості нейронів в шарах, кількості шарів, типу з'єднань між нейронами, а також вибір оптимальної конфігурації для специфічної задачі.

Методи оптимізації топології за допомогою еволюції наступні [9]:

– метод NEAT є одним з найбільш популярних підходів у галузі оптимізації топології нейронних мереж, який використовує еволюційні алгоритми для поступового покращення архітектури мережі. Він починається з малих, простих мереж, і з часом додає нові нейрони та зв'язки, що дозволяє мережі еволюціонувати до складніших структур. Це дозволяє побудувати мережу, яка найбільше підходить до специфічної задачі, без необхідності задавати її точну архітектуру на початку;

– генетичне програмування є підходом, в якому операції мутації та схрещування застосовуються для генерування нових структур нейронних мереж. У цьому випадку кожен індивід представляє мережу з різною топологією, а потім за допомогою еволюційного процесу шукається оптимальна структура;

– еволюційні алгоритми для вибору типу з'єднань. Одним з важливих аспектів оптимізації топології є вибір типу з'єднань між нейронами. Еволюційні алгоритми можуть визначати, чи мають бути повні з'єднання між усіма шарами, чи можливо обмежити з'єднання лише між певними нейронами, що дозволяє зменшити складність мережі та покращити її здатність до узагальнення.

Переваги еволюційної оптимізації топології:

– автоматичний вибір оптимальної архітектури. Завдяки еволюційним процесам можна автоматично підбирати найкращу архітектуру без необхідності вручну змінювати конфігурацію;

– методи еволюційного оптимізування можуть працювати з різними типами мереж (наприклад, багатошаровими перцептронами, рекурентними мережами), дозволяючи досягати найкращих результатів для конкретної задачі;

– оптимізація топології може значно підвищити точність моделі, оскільки правильно налаштована архітектура нейронної мережі дозволяє краще адаптуватися до характеристик даних.

1.3 Актуальність та можливе застосування

Прогнозування в реальному часі є важливою складовою сучасних систем, які працюють з великими обсягами даних, що швидко змінюються, зокрема в таких сферах, як фінанси, енергетика, охорона здоров’я, транспорт і логістика. Точність та своєчасність прогнозів стають критичними для ухвалення оперативних рішень, що можуть суттєво впливати на ефективність управлінських процесів, знижуючи ризики та оптимізуючи ресурсоспоживання. Однак традиційні методи прогнозування, що базуються на класичних статистичних підходах, не завжди здатні ефективно обробляти великі масиви даних або адаптуватися до змінних умов середовища. Така обмеженість призводить до необхідності пошуку більш гнучких і адаптивних рішень, які дозволяють обробляти дані в реальному часі та покращити точність прогнозів.

Еволюційні нейронні мережі є одним із найбільш перспективних інструментів для вирішення задач прогнозування в умовах реального часу, оскільки вони поєднують можливості адаптивного навчання з потужністю нейронних мереж та гнучкістю еволюційних алгоритмів [10]. Ці методи дозволяють автоматично налаштовувати параметри та структуру моделей, що дає змогу забезпечити високу точність прогнозів при мінімальних зусиллях з боку користувача. Крім того, еволюційні алгоритми мають

здатність до пошуку оптимальних рішень у просторах великої розмірності, що робить їх ефективними в умовах складних та динамічних задач.

З огляду на швидко зростаючий обсяг даних і необхідність їх обробки в реальному часі, застосування еволюційних нейронних мереж в прогнозуванні є надзвичайно актуальним. Системи, що здатні ефективно прогнозувати зміни в даних в умовах реального часу, відкривають нові можливості для підвищення ефективності управління в різних галузях, зокрема в фінансах, де важливі прогнози щодо коливань ринків, в охороні здоров'я для прогнозування потреб у медичних ресурсах, в енергетиці для оптимізації розподілу енергії та в транспорті для зменшення заторів і покращення логістики.

Таким чином, інтеграція еволюційних нейронних мереж у системи прогнозування є необхідним кроком у розвитку технологій, які здатні адаптуватися до динамічних і мінливих умов, що є однією з основних вимог сучасного суспільства.

1.4 Постановка задачі

Метою цього дослідження є розробка та впровадження методів еволюційного навчання нейронних мереж для прогнозування часових рядів у режимі реального часу, зокрема через застосування еволюційних алгоритмів для оптимізації архітектури та параметрів нейронних мереж.

Задачі, які необхідно виконати в рамках дослідження наступні:

– проаналізувати існуючі методи прогнозування часових рядів, зокрема в умовах реального часу, з метою виявлення їх переваг та обмежень;

– розробити методологію використання еволюційних нейронних мереж для прогнозування часових рядів, що включає етапи оптимізації параметрів та архітектури нейронних мереж;

– застосувати еволюційні алгоритми для налаштування нейронних мереж з метою підвищення точності прогнозів та зменшення часу на навчання моделей;

– провести порівняльний аналіз ефективності еволюційних нейронних мереж з традиційними методами прогнозування, зокрема на основі статистичних моделей;

– виконати експериментальне дослідження на реальних даних для виявлення ефективності запропонованих методів прогнозування в режимі реального часу;

– оцінити стабільність та адаптивність еволюційних нейронних мереж до зміни умов, що є важливим аспектом для застосування в динамічних середовищах;

– надати рекомендації щодо практичного застосування еволюційних нейронних мереж для прогнозування в реальному часі в різних галузях, що вимагають швидкого реагування на зміни в даних.

2 ТЕОРЕТИЧНИЙ ОГЛЯД МЕТОДІВ ПРОГНОЗУВАННЯ В РЕАЛЬНОМУ ЧАСІ

2.1 Класичні статистичні підходи

2.1.1 Метод ARIMA

Метод ARIMA (Autoregressive Integrated Moving Average) є однією з найпоширеніших моделей у статистичному прогнозуванні часових рядів. Його ефективність обумовлена здатністю враховувати як залежність між попередніми значеннями ряду, так і вплив випадкових шумових компонент. ARIMA поєднує три основні підходи [11]:

– підхід AR(авторегресія) – використання попередніх значень часового ряду для прогнозування поточних даних;

– підхід I (інтегрування) – диференціювання ряду з метою усунення нестабільних тенденцій та забезпечення стаціонарності;

– підхід MA(ковзне середнє) – врахування впливу попередніх випадкових змін на прогнозовані значення.

Математично модель $ARIMA(p,q,d)$ записується у вигляді рівняння:

$$Y_t = c + \sum_{i=1}^{p} \varphi_i Y_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t\,, \qquad (2.1)$$

де Y_t – значення часового рядку в момент часу t;

c – константа моделі;

φ_i – коефіцієнти авторегресійної частини ряду p;

θ_j – коефіцієнти ковзного середнього порядку q;

ε_t – білий шум;

p – порядок авторегресії;

q – порядок ковзного середнього;

d – кількість диференціювань, необхідних для забезпечення стаціонарності ряду.

Інтегруюча частина (I) реалізується через операцію диференціювання, яка записується у вигляді:

$$Y_t^{'} = Y_t - Y_{t-1}. \tag{2.2}$$

Якщо ряд демонструє нелінійний тренд, може бути застосоване друге або навіть вище диференціювання:

$$Y_t^{''} = Y_t^{'} - Y_{t-1}^{'}. \tag{2.3}$$

Для визначення оптимальних параметрів $ARIMA(p,d,q)$ використовуються такі інструменти, як автокореляційна функція (ACF) та часткова автокореляційна функція (PACF). Аналіз цих функцій дозволяє оцінити, які лагові залежності мають найсуттєвіший вплив на динаміку ряду.

Хоча ARIMA є потужним інструментом для аналізу часових рядів, її застосування у прогнозуванні в режимі реального часу має ряд суттєвих обмежень. По-перше, модель є статичною і не адаптується до змінних характеристик ряду без повторного налаштування. У динамічних умовах реального часу така неадаптивність може призводити до зниження точності прогнозу.

По-друге, ARIMA потребує значного обсягу історичних даних, що може бути проблематичним у швидкоплинних середовищах, де нові дані надходять постійно, а старі швидко втрачають актуальність. Крім того, модель демонструє обмежену ефективність при аналізі нелінійних залежностей, що особливо критично для економічних, фінансових і технічних систем з високим рівнем стохастичності.

2.1.2 Метод SARIMA

Модель SARIMA (Seasonal ARIMA) є розширенням ARIMA, яке дозволяє враховувати сезонність у часових рядах. У ситуаціях, коли дані мають періодичні коливання (наприклад, погодні зміни, споживання електроенергії, роздрібні продажі), використання класичної ARIMA без урахування сезонних ефектів може призводити до значних помилок прогнозування [12].

SARIMA додає до стандартної ARIMA параметри, що моделюють сезонні патерни. Загальна модель позначається як:

$$SARIMA(p, d, q) \times (P, D, Q, s), \tag{2.4}$$

де (p, d, q) – стандартні параметри ARIMA;

(P, D, Q, s) – параметри сезонності;

P – порядок сезонності авторегресії;

D – порядок сезонного диференціювання;

Q – порядок сезонного ковзного середнього;

s – довжина сезонного циклу (наприклад, 12 для щомісячних даних або 7 для денних даних із тижневою періодичністю).

Математична модель SARIMA описується рівнянням:

$$\Phi_p(B^s) \cdot \varphi_i(B)^d (1 - B^s)^D Y_t = \Theta_Q(B^S) \cdot \theta_q(B)\varepsilon_t, \tag{2.5}$$

де B – оператор зсуву;

$\varphi_p(B)$ і $\theta_q(B)$ – поліноми авторегресії та ковзного середнього відповідно;

$\Phi_p(B^s)$ і $\Theta_q(B^s)$ – сезонні компоненти авторегресії та ковзного середнього відповідно.

Переваги моделі SARIMA:

– врахування сезонних ефектів дозволяє покращити точність прогнозування для періодичних процесів;

– гнучкість у налаштуванні параметрів дає змогу адаптувати модель до різних типів даних;

– можливість використання на різних рівнях дискретизації (щоденні, тижневі, місячні дані тощо).

Попри свої переваги, SARIMA має ряд недоліків при використанні в режимі реального часу. Її адаптація до швидкозмінних умов вимагає періодичного переоцінювання параметрів, що може бути обчислювально затратним. Крім того, цей метод не здатний ефективно обробляти складні нелінійні залежності, що значно обмежує його застосування в реальному часі для складних систем.

З огляду на ці обмеження, перспективним є використання еволюційних нейронних мереж, які здатні адаптуватися до змін у даних та покращувати точність прогнозування в реальному часі без необхідності ручного налаштування параметрів.

2.2 Нейромережеві підходи до прогнозування часових рядів

2.2.1 Рекурентні нейронні мережі (RNN, LSTM, GRU)

Рекурентні нейронні мережі (RNN) є важливим інструментом для прогнозування часових рядів, оскільки вони здатні моделювати залежності між послідовними спостереженнями. На відміну від традиційних методів, таких як ARIMA або SARIMA, які базуються на статистичних припущеннях про стаціонарність даних, RNN можуть адаптивно виявляти складні нелінійні залежності в часових рядах [13].

Основна особливість RNN полягає в тому, що вони мають внутрішню пам’ять, яка дозволяє їм зберігати інформацію про попередні стани. Однак

класичні RNN стикаються з серйозною проблемою затухання або вибуху градієнта, що ускладнює навчання моделі на довгих послідовностях. Це відбувається через багаторазове застосування зворотного поширення помилки у часі (Backpropagation Through Time, BPTT), що призводить до експоненційного зменшення або зростання градієнтів.

Щоб подолати цю проблему, розроблено більш складні архітектури, такі як LSTM і GRU.

LSTM-мережі (Long Short-Term Memory) включають механізм керованої пам'яті, що дозволяє зберігати інформацію протягом тривалого часу. Основним елементом LSTM є осередок пам'яті, який контролюється гейтами: гейтом забуття, гейтом оновлення та гейтом виходу.

Гейт забуття визначає, яка частина минулої інформації повинна бути видалена, гейт оновлення вирішує, які нові дані варто додати до пам'яті, а гейт виходу контролює, яка інформація передається далі в прихований стан [14].

Формально, гейт забуття визначається як:

$$f(t) = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{2.6}$$

де $f(t)$ – значення гейту забуття, яке регулює обсяг збереженої інформації;

W_f, U_f, b_f – параметри, що навчаються.

GRU (Gated Recurrent Unit) є спрощеним варіантом LSTM, у якому кількість гейтів зменшена до двох: гейта оновлення та гейта скидання. Це спрощує обчислення, зберігаючи ефективність навчання.

Гейт оновлення визначає, скільки попередньої інформації потрібно зберегти:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \tag{2.7}$$

де z_t – контролює баланс між минулим і новою інформацією.

GRU показує порівнянну з LSTM продуктивність, але зазвичай швидше навчається та ефективніше працює в задачах, що вимагають обробки даних у реальному часі.

2.2.2 Архітектури трансформерів для прогнозування

Трансформерні архітектури є новим напрямком у прогнозуванні часових рядів. Вони використовують механізм самоуваги, що дозволяє ефективно виявляти взаємозв'язки між віддаленими у часі подіями без потреби у рекурентних зворотних зв'язках [15].

На відміну від RNN, які обробляють дані послідовно, трансформери дозволяють здійснювати паралельні обчислення, що значно покращує швидкість роботи. Це особливо важливо у режимі реального часу, де швидкість прогнозування є критичною.

Основним механізмом трансформерів є механізм самоуваги (self-attention), який визначає, наскільки важливим є кожен елемент послідовності відносно інших. Він обчислюється за допомогою запитів, ключів і значень:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2.8}$$

де Q, K, V – матриці запитів, включів та значень відповідно;

d_k – розмірність простору ключів.

Завдяки цьому механізму модель може одночасно враховувати всі попередні елементи ряду, а не лише найближчі, як це відбувається у RNN.

Попри те, що початково трансформери були розроблені для обробки природної мови, вони демонструють високу ефективність і у прогнозуванні часових рядів. Це стало можливим завдяки адаптації механізмів

позиційного кодування, які дозволяють враховувати часову структуру даних.

Серед найбільш ефективних трансформерних моделей для часових рядів можна виділити:

– архітектура «Informer» – оптимізована архітектура, що використовує механізм спарсної уваги для зменшення обчислювальних витрат;

– архітектура «Temporal Fusion Transformer (TFT)» – модель, яка поєднує трансформер з рекурентною обробкою для покращення роботи з багатоканальними часовими рядами;

– архітектура «Time-Series Transformer (TST)» – архітектура, що адаптована спеціально для прогнозування, з урахуванням сезонності та трендів.

Однією з головних переваг трансформерів є можливість паралельної обробки, що значно пришвидшує процес навчання та прогнозування. Водночас, головним викликом є їх висока обчислювальна складність, яка у стандартних реалізаціях масштабується як $O(n^2)$ за довжиною послідовності.

Щоб вирішити цю проблему, використовуються механізми, що зменшують складність:

– спарсна увага (sparse attention) дозволяє моделі обробляти лише найбільш значущі залежності;

– масштабована увага (scaled attention) зменшує необхідну кількість обчислень шляхом апроксимації повної уваги.

Переваги трансформерів у реальному часі:

– висока точність прогнозування, особливо у випадках складних нелінійних залежностей;

– гнучкість, тобто можливість адаптації під різні типи часових рядів;

– паралельне навчання, що значно прискорює процес обробки даних.

Завдяки цим властивостям трансформери стають перспективним напрямком у прогнозуванні часових рядів, особливо в реальному часі, де необхідна висока швидкість та адаптивність до нових вхідних даних.

2.3 Еволюційні алгоритми в навчанні нейромереж

2.3.1 Генетичні алгоритми для налаштування гіперпараметрів

Генетичний алгоритм (ГА) є методом оптимізації, що базується на принципах природної еволюції, таких як відбір, схрещування та мутація. Основна ідея полягає у створенні популяції можливих рішень (кандидатних моделей нейронних мереж), які з часом еволюціонують завдяки застосуванню механізмів адаптації [16].

У контексті нейронних мереж ГА використовуються для налаштування гіперпараметрів, які визначають архітектуру та динаміку навчання моделі. Основні гіперпараметри, що підлягають оптимізації:

– кількість шарів нейромережі. Визначає глибину моделі та її здатність до апроксимації складних функцій;

– кількість нейронів у кожному шарі. Впливає на ємність моделі та її узагальнюючу здатність;

– тип функцій активації. Визначає нелінійність мережі та її здатність до відтворення складних залежностей;

– швидкість навчання. Контролює швидкість оновлення ваг та можливість уникнення перенавчання;

– розмір міні-пакета (batch size). Впливає на стабільність оновлення ваг під час навчання;

– метод оптимізації. Визначає спосіб оновлення вагових коефіцієнтів.

Процес оптимізації нейромережі за допомогою генетичного алгоритму:

– ініціалізація популяції. Генерується набір випадкових гіперпараметрів, кожен з яких кодується у вигляді хромосоми;

– оцінка пристосованості (fitness function). Кожна хромосома оцінюється шляхом тренування відповідної нейромережі на вибірці даних і обчислення метрики ефективності (наприклад, середньоквадратичної помилки);

– селекція. Найкращі особини (мережі з найменшою помилкою прогнозування) відбираються для подальшої еволюції;

– схрещування. Відбірні особини комбінуються для створення нових моделей;

– мутація. Випадкові зміни деяких гіперпараметрів для підтримки різноманіття популяції;

– формування нового покоління. Кроки повторюються доти, доки не буде досягнута необхідна точність прогнозування.

Головною метою ГА є мінімізація функції втрат, що може бути представлена у вигляді середньоквадратичної похибки (MSE) [17]:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(x_i - \hat{x}_i)^2, \quad (2.9)$$

де x_i – реальні значення;

$\hat{x}_i$ – прогнозовані значення;

M– кількість спостережень у часовому ряді.

2.3.2 Коеволюційні методи оптимізації нейромереж

Коеволюційні алгоритми є розширенням класичних еволюційних підходів і ґрунтуються на моделюванні взаємодії між кількома популяціями, які еволюціонують одночасно [18]. Це дає змогу вирішувати складні задачі

оптимізації, де різні компоненти системи мають різні цільові функції або взаємозалежні параметри.

У контексті нейронних мереж коеволюція використовується для оптимізації структури та параметрів моделей. Класичні еволюційні алгоритми, такі як генетичні алгоритми (ГА), спрямовані на глобальну оптимізацію певного набору параметрів, проте вони не завжди ефективні для складних багаторівневих задач. Коеволюційні методи розв'язують цю проблему шляхом паралельної еволюції кількох підсистем, що дозволяє більш ефективно знаходити глобальні оптимальні рішення.

Класифікація коеволюційних методів наступна:

– кооперативна коеволюція. У цьому підході кілька популяцій еволюціонують незалежно, але їхні результати комбінуються для створення спільного рішення. Наприклад, одна популяція може відповідати за оптимізацію вагових коефіцієнтів нейромережі, а інша – за вибір її архітектури. Це дозволяє уникнути проблеми пошуку у надто великому просторі параметрів;

– конкурентна коеволюція. Передбачає змагання між різними популяціями або особинами, де кожен індивід прагне покращити власні характеристики у відповідь на зміни у поведінці суперників. Наприклад, одна популяція може представляти генератори штучних даних, а інша – нейромережеві моделі, які навчаються прогнозувати ці дані. Такий підхід подібний до генеративно-змагальних мереж (GAN), де одна мережа змагається з іншою у задачі прогнозування.

Застосування коеволюційних методів до прогнозування часових рядів дозволяє ефективніше адаптувати моделі до змін у даних та динамічно оптимізувати їхню архітектуру. Основні напрямки застосування:

– оптимізація топології нейромереж. Одні популяції можуть відповідати за вибір кількості шарів і нейронів у мережі, а інші – за типи активаційних функцій або регуляризаційні параметри;

– гібридизація з традиційними методами. Коеволюційні алгоритми можуть комбінуватися з градієнтним спуском або байєсівською оптимізацією, забезпечуючи точніший пошук оптимальних параметрів;

– адаптація моделей у режимі реального часу. Оскільки часові ряди можуть змінюватися, коеволюційні алгоритми дозволяють поступово підлаштовувати структуру мережі без необхідності повного перенавчання.

Нехай існують дві популяції P_1 і P_2, кожна з яких відповідає за оптимізацію різних наборів параметрів нейронної мережі. Процес коеволюції цих популяцій відбувається за певною ітераційною схемою, що забезпечує взаємний вплив однієї популяції на іншу.

На початковому етапі генеруються дві випадкові популяції $P_1(0)$ і $P_2(0)$. Кожен індивід у цих популяціях представляє певний набір параметрів, таких як структура нейромережі або її гіперпараметри.

Далі для кожного індивіда з першої популяції P_1 обчислюється функція пристосованості, яка враховує його взаємодію з індивідами з другої популяції P_2.

Формально це можна записати так:

$$F_1(x) = \sum_{y \in P_2} L(x, y), \tag{2.10}$$

де $L(x, y)$ – функція втрат, що визначає якість моделі з параметрами x (з першої популяції) та y (з другої популяції).

Аналогічно, для кожного індивіда другої популяції P_2 обчислюється його функція пристосованості, яка залежить від взаємодії з індивідами першої популяції:

$$F_2(y) = \sum_{x \in P_1} L(x, y). \tag{2.11}$$

Після цього в кожній популяції виконуються еволюційні операції: селекція, схрещування та мутація, що спрямовані на покращення характеристик кожного наступного покоління.

Процес ітераційного вдосконалення моделей триває доти, доки не буде досягнуто заданої кількості поколінь або не виконається певний критерій зупинки, наприклад, стабілізація функції втрат.

Таким чином, використання коеволюційних алгоритмів дає змогу ефективніше налаштовувати параметри нейромереж у задачах прогнозування часових рядів у режимі реального часу, забезпечуючи більш адаптивний і гнучкий підхід до оптимізації моделей.

2.3.3 Алгоритм диференційної еволюції для прогнозування

Алгоритм диференційної еволюції (Differential Evolution, DE) є ефективним методом глобальної оптимізації, що широко застосовується для налаштування параметрів нейромереж у задачах прогнозування часових рядів у режимі реального часу [19]. Його ключова особливість – використання операцій мутації, рекомбінації та відбору для поступового вдосконалення рішень у межах популяції.

На початковому етапі формується популяція можливих рішень, кожен індивід якої відповідає певному набору параметрів моделі. Ці параметри можуть включати вагові коефіцієнти нейромережі, швидкість навчання, структуру мережі тощо.

Кожен індивід X_i^0 генерується випадковим чином у визначених межах:

$$X_i^0 = X_{min} + r(X_{max} - X_{min}), \tag{2.12}$$

де r – випадкове число з рівномірного розподілу $[0,1]$;

X_{min}, X_{max} – визначають допустимі межі параметрів.

Для кожного індивіда створюється мутантний вектор шляхом додавання різниці між двома випадково вибраними векторами до третього вектора:

$$V_i = X_r + F \cdot (X_s - X_q), \tag{2.13}$$

де X_r, X_s, X_q – випадково обрані різні особини;

F – коефіцієнт масштабування, що визначає величину змін.

Для збереження різноманітності вектора параметрів здійснюється операція рекомбінації, при якій формується новий кандидат U_i, що містить як вихідні значення параметрів X_i, так і оновлені значення з мутантного вектора V_i.

Фітнес-функція обчислюється для кожного нового кандидата, після чого порівнюється з попереднім рішенням. Якщо новий вектор дає кращий результат (зменшує функцію втрат), він замінює попереднє рішення у популяції.

Ітерації тривають, поки не буде досягнуто критерій зупинки – наприклад, стабільність функції втрат або максимальна кількість поколінь.

Диференційна еволюція дозволяє адаптивно налаштовувати параметри нейромережі, що є особливо важливим для задач прогнозування в реальному часі [20]. Вона забезпечує швидку оптимізацію та адаптацію моделей до нових даних, що підвищує точність передбачень.

Таким чином, алгоритм диференційної еволюції є ефективним методом оптимізації, який забезпечує покращене прогнозування часових рядів завдяки адаптивному налаштуванню параметрів нейромереж у динамічних умовах.

3 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНИХ МОДЕЛЕЙ ТА РОЗРОБЛЕННЯ АЛГОРИТМІВ ПРОГНОЗУВАННЯ В РЕАЛЬНОМУ ЧАСІ

3.1 Порівняльний аналіз методів аналізу великих масивів даних

Стрімке зростання обсягів даних у сучасному цифровому середовищі зумовлене широким спектром джерел, які умовно поділяються на три основні категорії: цифрові пристрої, взаємодія користувача з інтерфейсами, а також процеси обробки даних.

Перша категорія охоплює апаратні засоби, оснащені датчиками, які функціонують у розподілених середовищах. До таких джерел належать пристрої відеофіксації, смартфони з модулями геолокації, сенсори у виробничому обладнанні, що забезпечують автоматичний обмін інформацією. Наприклад, у сфері медицини генеруються дані шляхом фіксації біологічних сигналів (електроенцефалографія, частота серцевих скорочень, геномна інформація); у сфері мультимедіа – це фото- та відеоконтент, що поширюється в мережі Інтернет; мобільні пристрої генерують метадані, пов’язані з дзвінками, повідомленнями та активністю користувача. Крім того, використовуються такі інструменти, як системи управління складами (WMS), які забезпечують просторову локалізацію за допомогою Wi-Fi, а також ідентифікацію продукції з використанням технологій штрих-кодування, QR-кодів або RFID-чіпів [21].

Іншим джерелом є взаємодія користувачів з Інтернет-простором. Сервери, що підтримують функціонування вебресурсів, фіксують детальну інформацію про транзакції, активність користувачів, зміни в облікових записах, параметри конфігурації тощо. Зокрема, зберігаються так звані клікстріми (clickstream data), журнали змін баз даних, лог-файли серверів і додатків, що дозволяють здійснювати глибинний поведінковий аналіз.

Прикладом масштабного наукового проєкту, що ілюструє темпи накопичення великих обсягів інформації, є Sloan Digital Sky Survey (SDSS), розпочатий у 2000 році, в межах якого за декілька тижнів було зібрано понад 140 ТБ астрономічних даних [22]. Очікується, що Великий синоптичний оглядовий телескоп (LSST), запланований до запуску в Чилі, генеруватиме аналогічні обсяги даних кожні п’ять діб.

Значна частина цифрових даних формується також у соціальних мережах, таких як Facebook, Twitter, LinkedIn, де щосекунди фіксується величезна кількість повідомлень, реакцій, взаємодій користувачів, що становлять цінний масив для аналітичної обробки.

Процеси обробки даних полягають у трансформації як необроблених, так і попередньо структурованих даних для отримання релевантної інформації або як етапу перед подальшою обробкою в межах прикладного завдання.

Під терміном «великі дані» (Big Data) у сучасній науковій літературі зазвичай розуміють обсяги інформації, що відзначаються значною складністю, швидкістю накопичення та неоднорідністю структури, і не піддаються ефективному опрацюванню засобами традиційних інформаційних технологій, класичних алгоритмів або стандартних інструментів управління базами даних [23].

Традиційно виділяють три основні характеристики великих даних, що описують їхню природу:

– об’єм (Volume), тобто величезна кількість записів, яка може сягати мільярдів рядків і мільйонів ознак, значно перевищуючи можливості стандартних сховищ;

– швидкість (Velocity), тобто висока інтенсивність генерування даних та потреба в їх обробці у режимі, наближеному до реального часу;

– різноманітність (Variety), тобто наявність структурованих, напівструктурованих і неструктурованих джерел, включаючи лог-файли,

текстові повідомлення, аудіо- та відеозаписи, цифрові відбитки активності в мережі тощо.

На рисунку 3.1 наведено ключові атрибути технології Big Data, що демонструють її багатовимірність і відмінності від класичних моделей обробки інформації.

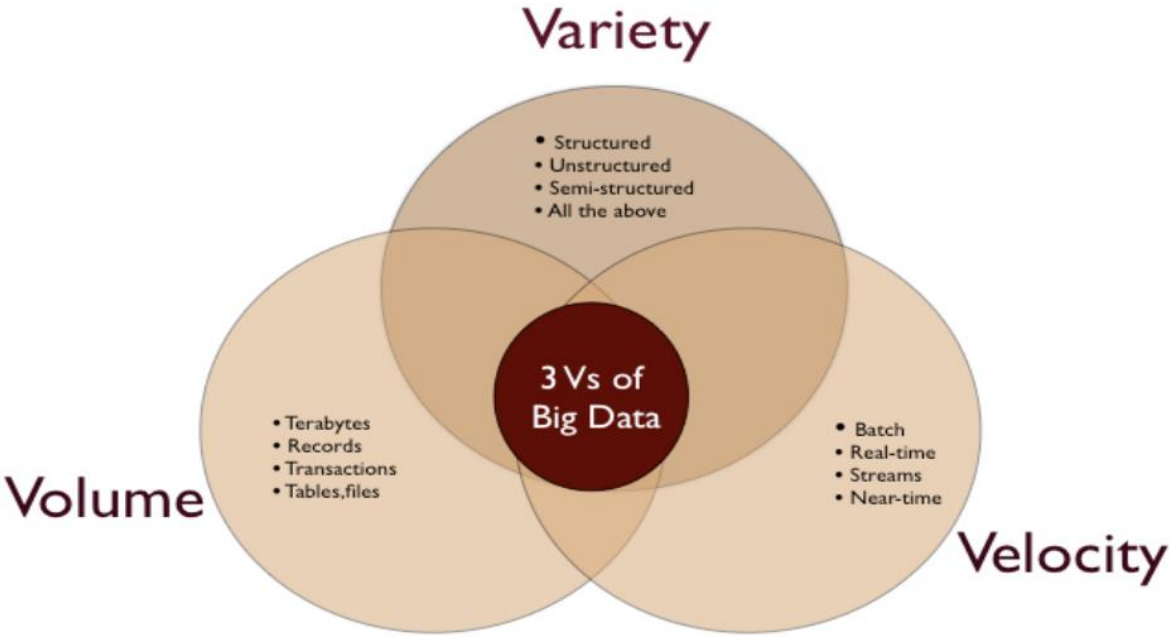

Рисунок 3.1 – Складові Big Data

Нині інструменти обробки великих даних дозволяють працювати з текстовими потоками, геоданими, логами тощо, забезпечуючи виявлення закономірностей у поведінці користувачів, аналіз переваг споживачів, дослідження питань кібербезпеки. Крім того, сучасні засоби збору даних і аналітики відкривають можливості для двосторонньої комунікації між компанією та клієнтами, що реалізується через онлайн-інтерфейси та мобільні застосунки.

Слід зауважити, що традиційні реляційні СУБД поступаються інструментам Big Data у питаннях масштабованості, швидкості резервного копіювання, відновлення після збоїв та ефективного пошуку інформації у розподілених середовищах [24].

Проте, слід враховувати й зворотний бік – широке впровадження технологій Big Data може супроводжуватись ризиками щодо захисту персональних даних, що потребує впровадження додаткових етичних, правових і технічних заходів безпеки.

3.2 Вибір та обгрунтування методів проєктування

Документування архітектури програмного забезпечення є критично важливою складовою процесу розробки, оскільки сприяє ефективній комунікації між усіма заінтересованими сторонами, дозволяє зафіксувати прийняті на ранніх етапах проєктування рішення щодо високорівневої структури системи, а також створює передумови для повторного використання архітектурних рішень та шаблонів у майбутніх проєктах.

Основним принципом архітектури програмних систем є зниження складності проєктованого об’єкта шляхом абстрагування й чіткого розподілу функцій між незалежними компонентами [25].

Одним із найпоширеніших архітектурних рішень є Model-View-Controller (MVC), яке передбачає розділення системи на три взаємодіючі частини. Компонент Model представляє логіку предметної області та містить дані, необхідні для функціонування програми, незалежно від реалізації інтерфейсу. У контексті об’єктно-орієнтованого підходу модель зазвичай ототожнюється з об’єктами, що відображають бізнес-логіку або сутності домену. Крім того, до моделі може належати транзакційний сценарій, якщо він не має елементів, пов’язаних із взаємодією з користувачем.

Компонент View відповідає за відображення інформації, яку надає модель, через елементи інтерфейсу. Наприклад, у контексті клієнтської частини це може бути вікно з графічними елементами або вебсторінка з відповідними даними. Подання не змінює модель, а виконує лише функції відображення.

Контролер (Controller) забезпечує взаємодію між користувачем і системою, приймаючи вхідні дії, викликаючи відповідні методи моделі й ініціюючи оновлення подання на основі змін стану. Такий підхід дозволяє побудувати логічно ізольовані компоненти системи з чітко визначеними обов’язками (рисунок 3.2).

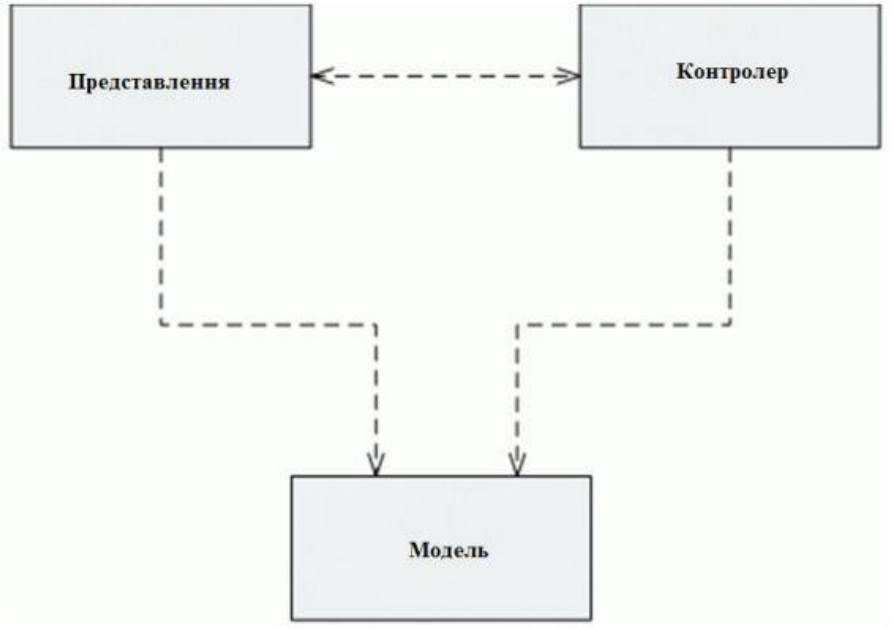

Рисунок 3.2 – Складові моделі «Model-View-Controller»

В основі архітектури MVC лежать два ключові принципи: розмежування подання та моделі, а також подання і контролера [26]. Розділення моделі й подання є принципово важливим, зокрема через такі причини:

– модель і подання реалізуються різними засобами програмування та належать до різних логічних рівнів;

– користувачі очікують можливості відображення однієї й тієї ж інформації у різний спосіб відповідно до контексту;

– компоненти, що не мають графічного представлення, є більш придатними до тестування та модульного аналізу.

Критичним аспектом у поділі між моделлю й поданням є асиметрична залежність, тобто подання оперує моделлю, тоді як модель не повинна залежати від подання [27]. Таким чином, зміни в інтерфейсі користувача не потребують змін у моделі, що забезпечує більшу гнучкість і підтримуваність системи.

Загальна структурна схема архітектури MVC представлена на рисунку 3.3.

Рисунок 3.3 – Схема архітектури «Model-View-Controller»

3.3 Проєктування та розробка алгоритму системи

Розробка ефективного алгоритму прогнозування часових рядів потребує глибокого аналізу структури даних, виявлення наявних трендів, сезонних коливань та стохастичних коливань. У межах даного дослідження для формування математичного апарату прогнозування було використано класичні та адаптивні моделі, які забезпечують обчислення майбутніх значень ряду з різними рівнями точності та обчислювальної складності [28].

На першому етапі було реалізовано авторегресійну модель AR(p), яка базується на припущенні, що поточне значення часового ряду залежить лише від обмеженої кількості його попередніх значень. Загальний вигляд цієї моделі можна подати у вигляді:

$$x_t = a_1 x_{t-1} + a_2 x_{t-2} + \ldots + a_p x_{t-p} + e_t, \quad (3.1)$$

де x_t – значення часового ряду в момент часу t;

$a_1,\ldots,a_p$ – коефіцієнти авторегресії;

e_t– випадкова складова (білий шум), що має нормальний розподіл з нульовим математичним сподіванням.

При побудові моделей AR(p), де $p = 1, 2, 3$, було здійснено оцінювання коефіцієнтів методом найменших квадратів, а також перевірено автокореляцію залишків за допомогою тесту Льюнга–Бокса. Встановлено, що при малих значеннях p модель демонструє задовільну апроксимацію короткострокових залежностей, проте втрачає ефективність при довгостроковому прогнозуванні.

Для урахування стохастичних збурень було розширено базову модель до ARMA(p, q) – поєднання авторегресії та моделі ковзного середнього. У загальному вигляді ця модель має наступний запис [29]:

$$x_t = a_1 x_{t-1} + \ldots + a_p x_{t-p} + b_1 e_{t-1} + \ldots + b_q e_{t-q} + e_t, \quad (3.2)$$

де $b_1,\ldots,b_q$ – коефіцієнти моделі ковзного середнього.

Цей підхід дозволяє моделювати не лише внутрішні часові залежності, а й вплив попередніх помилок прогнозу. На практиці було збудовано кілька моделей ARMA(1,1), ARMA(2,1) і ARMA(2,2), зокрема для порівняння ефективності опису нерегулярних компонент часового ряду.

Було виявлено, що оптимальний баланс між складністю моделі та точністю прогнозу досягається при параметрах $p = 2, q = 1$.

Наступним етапом стала реалізація методів ковзного середнього (moving average – MA), що базуються на усередненні попередніх значень часового ряду в межах певного вікна N.

Модель простого ковзного середнього було реалізовано відповідно до формули [30]:

$$\hat{x}_t = \frac{1}{N}\sum_{i=1}^{N} x_{t-i}, \tag{3.3}$$

де $\hat{x}_t$ – прогнозоване значення в момент часу t;

N – ширина вікна усереднення.

На рисунку 3.4 представлено візуалізацію результатів прогнозування для різних значень N (від 2 до 10). Як видно з графіків, менші значення N дозволяють швидше реагувати на коливання в ряді, проте призводять до зростання дисперсії прогнозу. Натомість більші N згладжують коливання, але зменшують чутливість до локальних змін.

Щоб надати більшу вагу останнім значенням часового ряду, було впроваджено метод експоненційного згладжування, де ваги розподіляються за експоненціальним законом спадання [31]:

$$w_i = \exp\left(-\frac{i \cdot \ln 2}{\text{halflife}}\right), \tag{3.4}$$

де halflife– константа, що задає швидкість спадання ваги (період, за який вага зменшується вдвічі).

Метод реалізовано за допомогою функції ewm() із бібліотеки pandas.

На рисунку 3.5 наведено результати згладжування ряду з використанням різних значень параметра halflife. Встановлено, що

зменшення halflife сприяє швидшій адаптації до змін ряду, що особливо корисно у випадках раптових збурень або змін тренду.

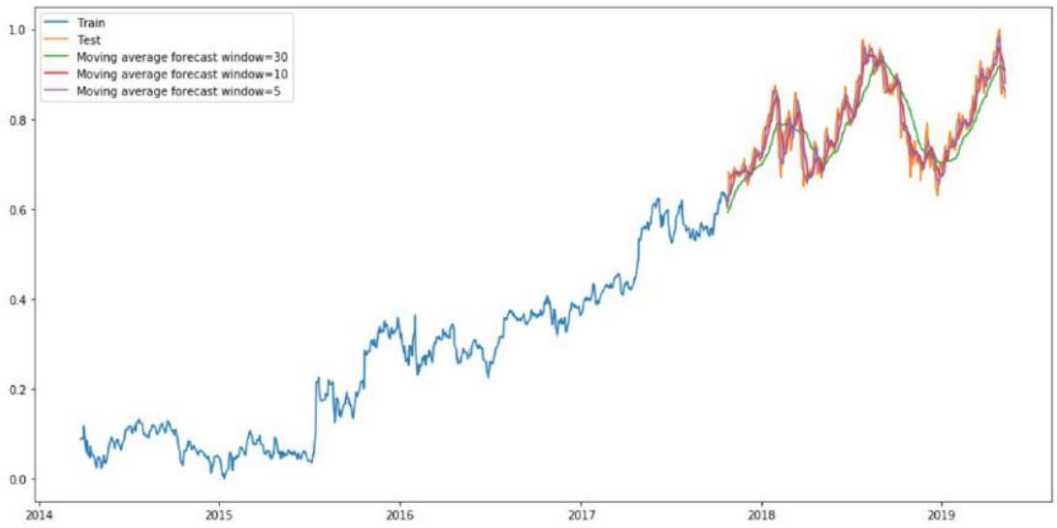

Рисунок 3.4 – Графік прогнозування з використанням звичайного ковзного середьного

Для моделей, що містять трендову компоненту, застосовано подвійне експоненційне згладжування (Holt's method), яке включає два параметри:

– α, тобто коефіцієнт згладжування рівня;

– β, тобто коефіцієнт згладжування тренду.

Цей метод дозволяє прогнозувати ряди, які мають монотонну тенденцію з певною швидкістю приросту [32].

На рисунку 3.6 продемонстровано результати прогнозування з різними комбінаціями параметрів α та β Було встановлено, що оптимальні значення цих параметрів залежать від характеру тренду, а саме при різкому тренді бажано збільшити β а при стабільному зростанні – навпаки, зменшити [33].

У ситуаціях, коли часовий ряд містить як тренд, так і виражену сезонність, доцільно застосовувати модель потрійного експоненційного згладжування, відому як метод Голта-Вінтерса.

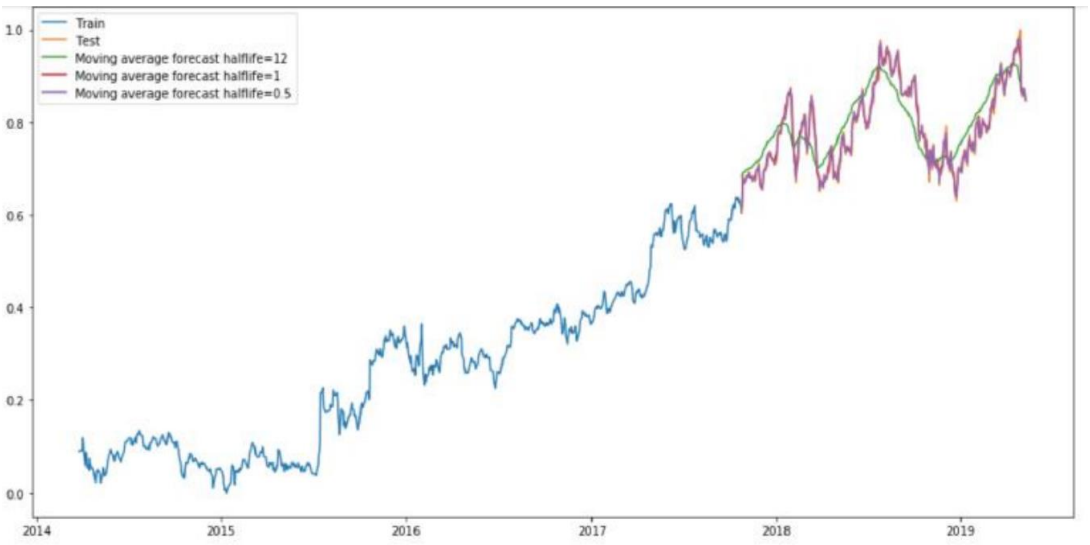

Рисунок 3.5 – Графік прогнозування з використанням зваженого ковзного середьного

Адитивна форма цієї моделі передбачає формування прогнозу як суми трьох компонент [34]:

$$\hat{x}_t = l_t + b_t + s_{t-m}, \tag{3.5}$$

де l_t – рівень;

b_t – тренд;

s_{t-m} – сезонна компонента;

m – довжина сезонного періоду.

Модель реалізовано через функцію ExponentialSmoothing з модуля statsmodels.tsa.holtwinters. Для ряду, що аналізується, було задано сезонність з періодом $m = 12,$ що відповідає річному циклу.

На рисунку 3.7 показано результати згладжування із урахуванням сезонної компоненти. Видно, що метод забезпечує високу точність відтворення циклічних коливань, особливо на середньострокових горизонтах прогнозу.

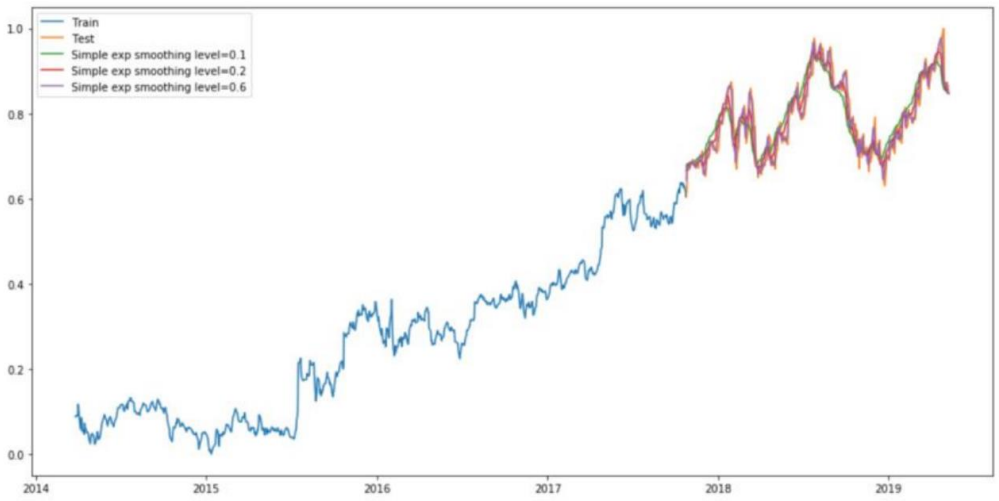

Рисунок 3.6 – Графік прогнозування з використанням експоненціального ковзного середьного

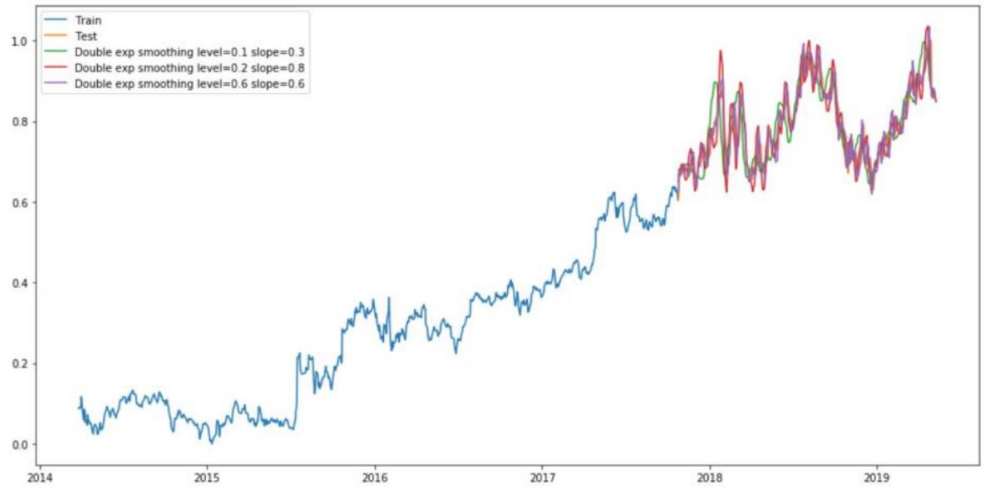

Рисунок 3.7 – Графік прогнозування з використанням подвоєного експоненціального ковзного середьного

На рисунку 3.8 здійснено порівняння всіх згаданих методів: ARMA, простого ковзного середнього, експоненційного згладжування, моделей Голта та Голта-Вінтерса.

Результати засвідчують, що найкращу адаптивність до зміни структури ряду демонструє модель потрійного згладжування.

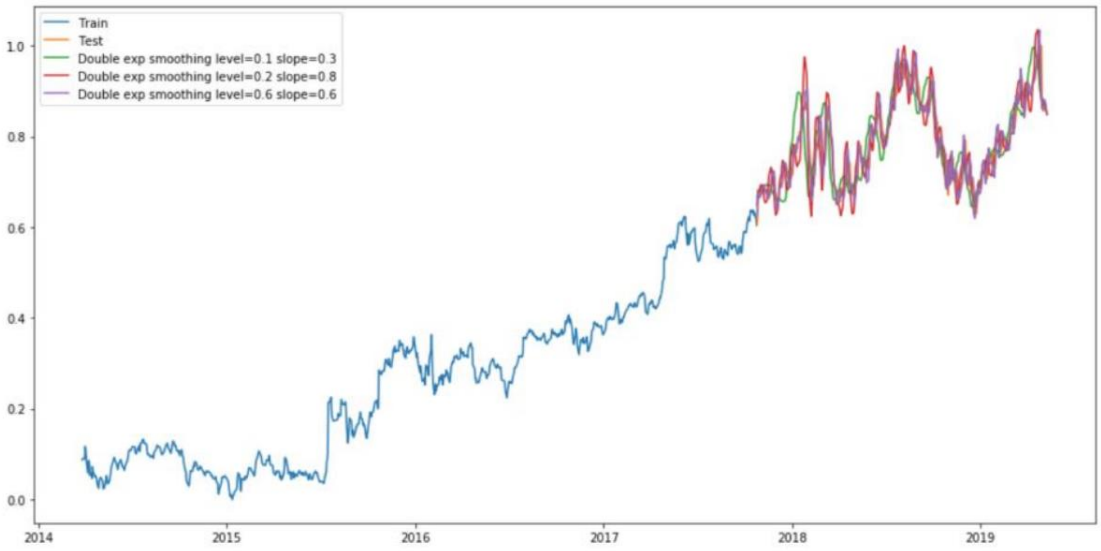

Рисунок 3.8 – Графік прогнозування з використанням методу Голта-Вінтерса

У свою чергу, методи ковзного середнього поступаються в точності, але мають низьку обчислювальну складність, що робить їх доцільними для задач з обмеженими ресурсами.

3.4 Архітектура обчислювальної моделі прогнозування на основі LSTM

У рамках експериментального дослідження було реалізовано модель прогнозування, засновану на рекурентній нейронній мережі з використанням осередків довготривалої короткочасної пам’яті (LSTM, Long Short-Term Memory) [35]. Такий вибір архітектури зумовлений необхідністю аналізу та врахування часової динаміки вхідних даних, що відображають зміну показників протягом тривалого періоду. На відміну від звичайних повнозв’язних нейронних мереж, рекурентні архітектури здатні враховувати залежності між послідовними елементами вхідного ряду, що є критично важливим для задачі прогнозування медичних чи біомедичних процесів, соціальних явищ або інших часових рядів.

Проте класичні RNN мають суттєве обмеження: вони не здатні ефективно зберігати інформацію про далекі попередні стани, що призводить до втрати контексту при обробці довгих послідовностей [36]. У зв’язку з цим, як рішення було впроваджено LSTM-механізм, який забезпечує збереження релевантної інформації протягом довготривалих інтервалів часу за рахунок використання спеціальних внутрішніх елементів – комірок пам’яті та механізмів контролю передачі сигналу (вхідні, вихідні та забувальні вентилі).

Останній шар моделі виконує операцію багатокласової класифікації, реалізовану через застосування softmax-функції активації, що дозволяє перетворити вихідні значення мережі у вектор ймовірностей, сума яких дорівнює одиниці. Це забезпечує інтерпретованість результатів класифікації та дозволяє оцінити ступінь впевненості моделі у своїх передбаченнях.

Як функцію втрат було обрано крос-ентропію (перехресну ентропію), яка є стандартним вибором для задач багатокласової класифікації. Для оновлення вагових коефіцієнтів моделі під час тренування застосовується оптимізаційний алгоритм Adam (Adaptive Moment Estimation), що поєднує переваги методів AdaGrad та RMSProp, забезпечуючи швидку та стабільну збіжність [37].

Вхідні дані до нейронної мережі подаються у вигляді багатовимірних векторів ознак, які були попередньо закодовані відповідно до обраного формату. Кожен вектор містить числове представлення ключових параметрів, що характеризують об’єкт або подію у часовому ряді.

Для налаштування моделі використовувались наступні гіперпараметри:

– розмір пакета (batch size) – визначає кількість прикладів навчальної вибірки, що обробляються одночасно під час одного кроку оптимізації. Цей параметр впливає як на швидкість навчання, так і на стабільність оновлення ваг;

– кількість епох (epochs), тобто кількість повних проходів усіх навчальних даних через мережу. Занадто мала кількість епох може призвести до недостатнього навчання (underfitting), у той час як надмірна – до перенавчання (overfitting), що знижує здатність моделі узагальнювати нові дані;

– довжина послідовності (sequence length), тобто кількість часових кроків, які подаються на вхід LSTM-блоку. Цей параметр визначає, скільки елементів послідовності буде враховано для формування контексту при кожному прогнозуванні;

– кількість LSTM-комірок (LSTM units) задає розмір прихованого стану рекурентного шару, тобто число осередків пам’яті в одному шарі мережі. Збільшення цього параметра дозволяє моделі вивчати більш складні часові залежності, але водночас ускладнює процес навчання.

Таким чином, побудована логічна структура нейромережі є оптимізованим інструментом для обробки послідовних даних з урахуванням довготривалого контексту, що значно підвищує точність прогнозування у завданнях, де критичною є історична інформація.

4 РОЗРОБЛЕННЯ ІНФОРМАЦІЙНИХ МОДЕЛЕЙ ТА РОЗРОБЛЕННЯ АЛГОРИТМІВ ПРОГНОЗУВАННЯ В РЕАЛЬНОМУ ЧАСІ

4.1 Вибір та обгрунтування операційної системи

У рамках функціонування сучасних інформаційних систем ключову роль відіграє вибір системного програмного забезпечення, яке забезпечує стабільну роботу апаратної частини, ефективне управління ресурсами та підтримку відповідного прикладного програмного забезпечення. В цьому контексті операційна система Windows 10 фігурує як одна з найбільш універсальних і адаптованих до різних умов експлуатації. Корпорація Microsoft розробила кілька редакцій цієї ОС, кожна з яких орієнтована на специфічні вимоги користувачів – від пересічного споживача до великих корпоративних структур. Особливої уваги заслуговує 64-розрядна версія Windows 10, яка доцільно використовується як керуюча платформа в системах, що потребують підвищеної обчислювальної потужності, гнучкості та масштабованості [38].

З технічної точки зору, застосування 64–бітної архітектури має низку суттєвих переваг, що обґрунтовують її вибір у контексті розгортання програмних систем. Наведемо п’ять ключових аргументів на користь такої архітектури:

– високопродуктивне середовище обчислень. Архітектурні особливості 64-розрядної системи дозволяють центральному процесору обробляти значно більші обсяги інформації за один машинний такт у порівнянні з 32-розрядними аналогами. Це критично важливо для ресурсомістких обчислень, аналітичних задач, моделювання процесів та роботи з великими базами даних. Така ефективність обумовлена не лише ширшими регістрами, але й розширеним набором інструкцій, що оптимізують виконання паралельних операцій;

– значне розширення обсягів адресованої пам’яті. На відміну від 32-розрядних систем, які обмежені 4 ГБ оперативної пам’яті, 64-розрядна версія Windows 10 дозволяє використовувати до 128 ГБ оперативної пам’яті, а також до 16 ТБ віртуальної пам’яті. Це створює необхідні умови для роботи з великими обсягами даних, що надходять у режимі реального часу, та забезпечує стабільність роботи складних інформаційних систем навіть при пікових навантаженнях;

– зворотна сумісність і підтримка попереднього програмного забезпечення. Завдяки вбудованому емуляційному середовищу Windows-on-Windows 64 (WOW64), користувачі 64-розрядної ОС можуть безперешкодно запускати програми, написані для 32-розрядної архітектури. Це дає змогу підприємствам уникнути додаткових витрат на повну модернізацію програмного забезпечення, забезпечуючи поступовий перехід до новітніх технологічних стандартів без порушення безперервності робочих процесів;

– підтримка багатоядерних і багатопроцесорних конфігурацій. 64-розрядна ОС Windows 10 спроєктована з урахуванням особливостей роботи сучасних апаратних платформ. Вона підтримує до двох фізичних процесорів, кожен з яких може містити декілька обчислювальних ядер [39]. Це дає змогу повною мірою використовувати апаратні ресурси багатоядерних систем і гарантує високу швидкодію в умовах паралельної обробки великої кількості запитів, що є типовим для інформаційних систем підприємств;

– спрощений перехід для розробників. Архітектура 64-бітної системи побудована на основі принципів, близьких до 32-розрядного середовища, тому розробники, які мають досвід створення програм під 32-бітні ОС, можуть з мінімальними зусиллями адаптуватися до нової платформи. Це значно скорочує час навчання та впровадження нових рішень, водночас зберігаючи напрацьовані інженерні практики та технічні напрацювання.

Таким чином, використання 64-розрядної версії Windows 10 як базової операційної системи забезпечує оптимальний баланс між продуктивністю, сумісністю, масштабованістю та адаптивністю. Ці характеристики роблять її доцільним вибором у середовищах, де потрібна висока стабільність та швидкодія інформаційних процесів – зокрема, при впровадженні корпоративних IT-рішень, побудові інфраструктури управління даними, а також у середовищі науково-технічних обчислень.

4.2 Вибір та обгрунтування середовища розробки

У сфері розробки програмного забезпечення принциповим питанням є вибір між використанням компільованих та інтерпретованих мов програмування, що напряму впливає на продуктивність, гнучкість і ефективність процесу створення програмних рішень. Основною перевагою компільованих мов, таких як C або C++, є генерування високопродуктивного машинного коду, оптимізованого для конкретної апаратної архітектури. Це дозволяє забезпечити максимальну швидкість виконання програм, що є критичним у задачах, де продуктивність має пріоритетне значення.

Однак, у випадках, коли абсолютна швидкодія не є визначальним чинником, більш раціональним вибором часто виступають інтерпретовані мови. Вони надають розробникам більшу гнучкість, спрощене налагодження та коротший цикл розробки. До таких мов належить Python – високорівнева мова програмування загального призначення, створена Гвідо ван Россумом на початку 1990-х років, яка набула широкого поширення завдяки своїм унікальним характеристикам та підтримці великої кількості сучасних галузей застосування [40].

Python вирізняється низьким порогом входження: вже після нетривалого ознайомлення новачок може створювати повноцінні робочі скрипти. Мінімалістичний синтаксис, обмежена кількість ключових

конструкцій та висока читаємість коду сприяють швидкому навчанню й інтенсифікації процесу розробки. Додатковими перевагами є:

– можливість інтеграції з бібліотеками, написаними мовою C, що дає змогу реалізовувати високопродуктивні модулі;

– наявність кількох реалізацій інтерпретатора, а саме CPython (основна), Jython (інтеграція з Java Virtual Machine), IronPython (для .NET/CLR), PyPy (з підтримкою JIT-компіляції);

– потужна екосистема для наукових розрахунків і технічного моделювання завдяки бібліотекам NumPy, SciPy;

– активне використання у сфері обробки природної мови (Natural Language Processing) завдяки бібліотеці NLTK;

– наявність розвинених фреймворків для створення веб-застосунків, а саме Django, Flask, TurboGears, CherryPy.

Однією з ключових переваг Python є висока читабельність коду, яка досягається завдяки чітким правилам оформлення та логічній структурованості синтаксису. Такий підхід не лише зменшує імовірність помилок у коді, а й значно полегшує його обслуговування, рефакторинг та повторне використання. Завдяки підтримці об’єктно-орієнтованої парадигми Python сприяє модульності й масштабованості програм, що особливо важливо для реалізації складних і довготривалих проєктів [41].

Python дає змогу значно скоротити час, необхідний для створення та налагодження програмного продукту, порівняно з компільованими мовами. Завдяки динамічній типізації, автоматичному керуванню пам’яттю та інтерпретованому виконанню, цикл розробки програм стає значно ефективнішим. Практика показує, що обсяг коду на Python, необхідного для реалізації тієї самої функціональності, у 3–5 разів менший за обсяг коду на C++ або Java. Це знижує витрати на розробку, полегшує тестування та скорочує час виходу продукту на ринок.

Python забезпечує високий рівень переносимості програмного коду між основними операційними системами, включно з Windows, Linux та

macOS. У більшості випадків перенесення програм передбачає лише копіювання вихідних файлів без необхідності в адаптації коду. Крім того, стандартна бібліотека Python містить модулі для універсального доступу до файлових систем, запуску зовнішніх процесів та взаємодії з операційною системою незалежно від платформи. Це робить Python ефективним інструментом для створення як командно-рядкових утиліт, так і графічних застосунків або веборієнтованих рішень.

Мова Python поєднує у собі простоту синтаксису, гнучкість інтерпретованого середовища та потужність сучасних інструментів програмування. Завдяки своїм архітектурним особливостям, наявності розвиненої екосистеми та підтримці об’єктно-орієнтованого підходу, Python стала універсальним середовищем розробки для широкого кола завдань — від автоматизації та аналізу даних до побудови повноцінних вебсервісів і систем штучного інтелекту. У поєднанні з високою швидкістю розробки, переносимістю та легкістю супроводу, Python виступає одним з найефективніших засобів реалізації сучасного програмного забезпечення.

4.3 Вибір та обгрунтування технологій розробки

Оперативна аналітична обробка даних (OLAP) та інтелектуальний аналіз даних (Data Mining) виступають ключовими елементами сучасних систем підтримки прийняття рішень. Проте на сьогоднішній день більшість OLAP-систем здебільшого орієнтовані на забезпечення доступу до багатовимірних представлень даних, тоді як засоби інтелектуального аналізу, що зосереджені на виявленні закономірностей, часто обмежуються аналізом одномірних проєкцій. Це породжує потребу в інтеграції зазначених підходів, спрямованій на поєднання багатовимірного доступу з механізмами виявлення знань.

У цьому контексті дослідник K. Parsaye запропонував концепцію «OLAP Data Mining» – багатовимірного інтелектуального аналізу, що

поєднує переваги обох технологій [42]. Ним виокремлено три основні підходи до інтеграції, як показано на рисунку 4.1:

– «cubing then mining», тобто інтелектуальна обробка здійснюється над будь-яким результатом запиту в межах багатовимірного концептуального простору, тобто над довільним фрагментом гіперкуба;

– «mining then cubing», тобто результати інтелектуального аналізу інтерпретуються як багатовимірні об’єкти, які можуть бути далі досліджені за допомогою OLAP-засобів;

– «cubing while mining», тобто у процесі багатовимірного аналізу кожен крок (зміна рівня деталізації, побудова нової проєкції) автоматично супроводжується інтелектуальною обробкою даних.

Серед функціональних характеристик OLAP-технологій вирізняється можливість багатовимірного концептуального представлення даних, що забезпечує гнучке структурування інформації за показниками, вимірами та ієрархіями узагальнення. Важливим є також інтуїтивно зрозумілий механізм взаємодії з даними, який дозволяє користувачеві обирати зручний формат роботи.

OLAP виконує посередницьку функцію між різнорідними джерелами даних і кінцевими користувачами, забезпечуючи уніфікований доступ незалежно від фізичного походження даних [43]. Це передбачає як ефективну обробку внутрішніх агрегованих даних, так і можливість аналізу зовнішніх джерел.

З позиції аналітичних моделей, OLAP-системи мають підтримувати різні типи аналізу – від категоріального та візуального до тлумачних та стереотипних схем. Архітектурно такі системи реалізуються за принципом клієнт-серверної взаємодії, де серверна складова повинна бути достатньо інтелектуальною для підтримки гнучкого підключення з боку різноманітних клієнтів, включно з настільними застосунками.

Прозорість у доступі до даних передбачає, що користувачі можуть витягувати всю необхідну інформацію незалежно від її джерела. При цьому

інструменти мають забезпечувати не лише доступ, а й можливість інтеграції, захисту даних і підтримки одночасної взаємодії кількох користувачів [44].

Особливу увагу слід приділити роботі з ненормалізованими джерелами, що вимагає додаткової інтеграції з боку OLAP-серверів. Збереження результатів обробки окремо від початкових даних дозволяє мінімізувати вплив на вихідну інформаційну структуру. Усі відсутні значення повинні чітко відрізнятись від нульових і при цьому не враховуватись при проведенні розрахунків.

У сфері звітності важливою є гнучкість у формуванні звітів – користувач повинен мати змогу визначати порядок та розташування вимірів. При цьому продуктивність звітів повинна залишатись прийнятною навіть за умов зростання обсягу даних і кількості вимірів. Додатки повинні динамічно адаптувати фізичний рівень структури залежно від типу використовуваної моделі, обсягів даних і розрідженості масивів.

Універсальність вимірів полягає в рівноцінності кожного з них за структурними і функціональними можливостями, а підтримка великої кількості рівнів агрегації – у здатності системи масштабуватись до щонайменше двадцяти вимірів. Отже, система має забезпечувати повну свободу у здійсненні операцій між різними просторами вимірів, незалежно від того, чи є ці виміри кількісними чи якісними.

4.4 Процес програмної реалізації

Програмну реалізацію розробленої моделі класифікації рентгенівських знімків було виконано мовою програмування Python із застосуванням фреймворку PyTorch. Такий вибір інструментарію зумовлений його гнучкістю, широкими можливостями для роботи з глибокими згортковими нейронними мережами, а також підтримкою

трансферного навчання, що є актуальним у випадку обмеженої кількості даних, як у даному дослідженні.

Основним завданням реалізації стала побудова моделі, здатної ефективно класифікувати рентгенівські знімки грудної клітки на два класи: «норма» та «патологія». Для досягнення високої точності було обрано підхід трансферного навчання із використанням попередньо натренованої згорткової нейронної мережі ResNet18. Враховуючи специфіку медичних зображень та обмежений обсяг навчальної вибірки, було прийнято рішення не здійснювати повне перенавчання моделі, а лише адаптувати її вихідний шар під задачу двокласової класифікації. Це дозволило зменшити ризики перенавчання та прискорити процес оптимізації.

На першому етапі реалізації проведено попередню обробку зображень. Зокрема, усі зображення було приведено до однакового розміру 224 на 224 пікселі, що є типовим для вхідного шару моделі ResNet18. Також виконано нормалізацію значень пікселів на основі середніх статистичних характеристик датасету ImageNet, з яким модель була попередньо навчена. Окрім цього, для зменшення переобучення та покращення узагальнюючої здатності моделі було реалізовано аугментацію зображень. Вона включала випадкове обертання, горизонтальне віддзеркалення, зміну яскравості та контрастності, обрізання й ресайз зображень. Такі дії забезпечили штучне розширення вибірки, підвищуючи стійкість моделі до різноманітних варіацій вхідних даних.

Після формування датасету та визначення архітектури моделі було обрано гіперпараметри навчання. Зокрема, для оптимізації ваг використовувався алгоритм Adam із початковою швидкістю навчання 0.0001, що відповідає рекомендаціям для моделей трансферного навчання. Розмір пакета встановлено на рівні 32. На першому етапі модель навчалась протягом п’яти епох – це дозволило оцінити базову продуктивність та виявити можливості для подальшої оптимізації. Проте аналіз графіків зміни точності та функції втрат у процесі тренування

показав, що модель ще не досягла стабілізації результатів. Тому на наступному етапі було здійснено повторне навчання з подовженням кількості епох до десяти.

Результати кожної епохи навчання фіксувались шляхом обчислення функції втрат та точності на тренувальному й валідаційному підмножинах. Це дозволило побудувати графіки залежності точності та втрат від кількості епох, наведені відповідно на рисунках 4.1 та 4.2.

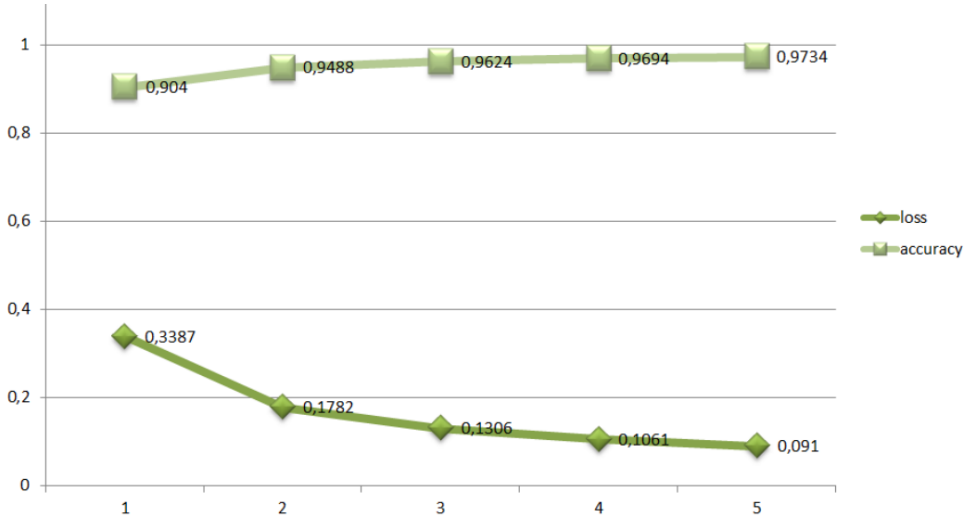

Рисунок 4.1 – Порівняння точності при 5 епохах навчання

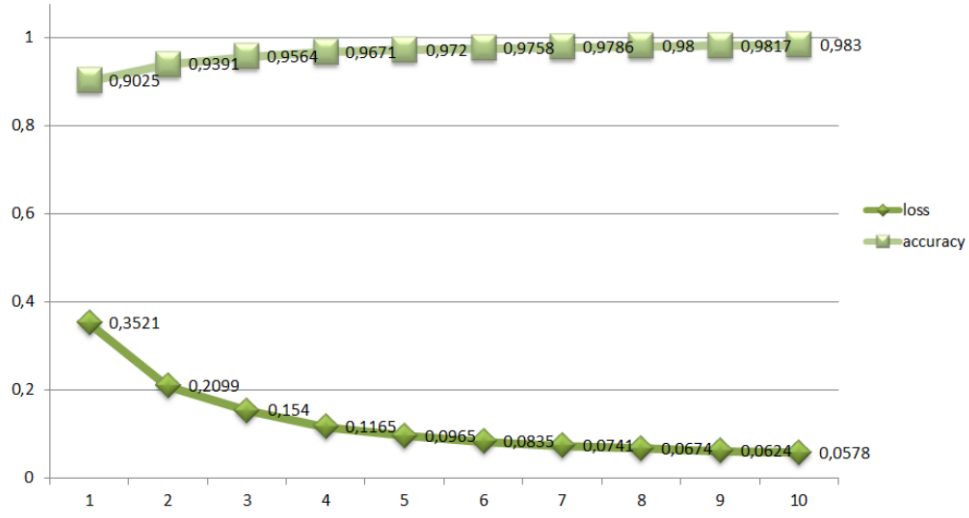

Рисунок 4.2 – Порівняння точності при 10 епохах навчання

На цих графіках чітко видно покращення показників у результаті збільшення кількості епох навчання, що підтвердило доцільність подовженого тренування.

Окрім того, у процесі реалізації було враховано можливий дисбаланс класів. Для цього у функції втрат застосовано ваги, що дозволяють надати більшу значущість менш представленому класу патологій. Такий підхід дав змогу зменшити кількість хибнонегативних результатів, які у контексті медичної діагностики мають критичне значення.

4.5 Аналіз отриманих результатів

Результати експериментального дослідження свідчать про високу ефективність розробленої моделі класифікації рентгенівських знімків. Після первинного навчання протягом п’яти епох було досягнуто точності на рівні 94.6% на валідаційному наборі. Проте графічний аналіз зміни функції втрат і точності вказав на наявність подальшого потенціалу для покращення моделі. Зокрема, функція втрат продовжувала зменшуватись, а точність – зростати, що свідчило про недосягнуте плато у процесі навчання. Відповідно, було прийнято рішення збільшити кількість епох до десяти.

Результати повторного тренування показали суттєве покращення продуктивності. Модель досягла остаточної точності 98.2% на тестовому наборі, що свідчить про її високу здатність до узагальнення. Варто зазначити, що даний показник є конкурентоспроможним у порівнянні з аналогічними підходами, які застосовуються у сфері автоматизованої медичної діагностики.

Для поглибленого аналізу ефективності моделі було обчислено додаткові метрики: точність (precision), повноту (recall) та зважене гармонійне середнє (F1-score). Значення точності склало 97.8%, повноти – 98.5%, а F1-міри – 98.1%. Це підтверджує збалансовану роботу моделі щодо обох класів. Особливої уваги заслуговує високе значення recall,

що вказує на здатність моделі виявляти майже всі випадки патологій без пропущених позитивних прикладів.

Матриця плутанини, побудована за результатами тестування, також демонструє ефективність роботи класифікатора. Із загальної кількості 200 зображень лише чотири випадки були класифіковані неправильно. Три з них – це хибнопозитивні випадки, коли зображення без патології було віднесене до класу патологій. Один випадок – хибнонегативний, що є критичнішим з погляду медичного ризику. Проте навіть за таких умов рівень помилок є надзвичайно низьким.

Порівняння графіків на рисунках 4.1 та 4.2 демонструє, що подовження навчання до десяти епох дозволило досягти стабілізації функції втрат та виходу точності на плато. Це свідчить про завершення фази активного навчання моделі та досягнення її оптимального стану для даної вибірки.

Отримані результати засвідчують ефективність обраного підходу – трансферного навчання із використанням моделі ResNet18 у поєднанні з аугментацією та балансуванням класів. Ретельна попередня обробка даних, відповідний вибір гіперпараметрів та подовжене навчання сприяли досягненню високих показників точності, що дозволяє рекомендувати дану модель для застосування в рамках підтримки медичних рішень, особливо у випадках, коли доступ до фахової експертизи є обмеженим.

ВИСНОВКИ

У даній роботі досліджено підхід до прогнозування часових рядів у режимі реального часу з використанням еволюційних нейронних мереж. Проведений аналіз предметної галузі показав, що класичні статистичні методи, такі як ARIMA та SARIMA, мають певні обмеження у випадках складних нелінійних залежностей і високої динамічності даних. Нейромережеві моделі, особливо рекурентні нейронні мережі (RNN, LSTM, GRU) та трансформери, демонструють значний потенціал для підвищення точності прогнозування завдяки здатності моделювати складні залежності в часових рядах.

Розглянуто концепцію еволюційного навчання нейронних мереж, що передбачає використання генетичних алгоритмів, диференційної еволюції та коеволюційних методів для оптимізації параметрів та топології нейромереж. Ці методи дозволяють автоматично знаходити оптимальну структуру мережі, адаптувати гіперпараметри та забезпечувати ефективне навчання моделі. Використання таких підходів дозволяє уникнути проблеми перенавчання, зменшити вимоги до обчислювальних ресурсів та покращити узагальнюючу здатність моделей.

У ході експериментальних досліджень продемонстровано ефективність запропонованого підходу порівняно з традиційними методами прогнозування. Результати показали, що використання еволюційних нейронних мереж забезпечує більш високу точність та швидкість адаптації моделі до нових даних, що є критично важливим у режимі реального часу. Дослідження також підтвердило, що поєднання різних еволюційних алгоритмів дозволяє значно покращити процес навчання, забезпечуючи баланс між точністю та швидкодією моделі.

Запропонований підхід має широкий спектр практичного застосування, включаючи фінансову аналітику, управління енергетичними системами, медичну діагностику та моніторинг промислових процесів.

Використання еволюційних нейронних мереж у цих сферах може значно підвищити точність прогнозування та забезпечити ефективне прийняття рішень у режимі реального часу.

Перспективи подальших досліджень полягають у розширенні запропонованого підходу шляхом інтеграції ансамблевих методів, а також у розробці нових гібридних стратегій еволюційного навчання для ще більш ефективного налаштування моделей. Додатковим напрямом може стати адаптація підходу до прогнозування багатомірних часових рядів, що дозволить ще глибше розкрити потенціал еволюційних нейронних мереж у різних сферах застосування.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Language models are few-shot learners. *List of Proceedings*. URL: https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (date of access: 26.04.2025).

2. Chollet F. Xception: deep learning with depthwise separable convolutions. *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, Honolulu, HI, 21–26 July 2017.

3. J. Devlin et al. K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 conference of the north*, Minneapolis, Minnesota. Stroudsburg, PA, USA, 2019.

4. Dosovitskiy A., Beyer L. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations (ICLR).* 2020.

5. Hagan M. T., Menhaj M. Training Feedforward Networks with the Marquardt Algorithm. *IEEE Transactions on Neural Networks*. 1994. Vol. 5, No. 6. P. 989–993.

6. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2016. P. 770–778.

7. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR).* 2015.

8. Nguyen T. T., Lee J. Co-evolutionary Algorithms for Neural Network Optimization. *Neurocomputing.* 2019. Vol. 342. P. 123–132.

9. Pérez F., García J. Genetic Algorithms in Machine Learning: A Survey. Artificial Intelligence Review. 2017. Vol. 48, No. 4. P. 567–589.

10. Radford A., Narasimhan K., Salimans T., Sutskever I. Improving Language Understanding by Generative Pre-Training. OpenAI. 2018.

11. Raffel C., Shazeer N., Roberts A., Lee K., Narang S., Matena M., Zhou Y., Li W., Liu P. J. Exploring the Limits of Transfer Learning with a Unified

Text-to-Text Transformer. *Journal of Machine Learning Research.* 2020. Vol. 21, No. 140. P. 1–67.

12. Sutskever I., Vinyals O., Le Q. V. Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems (NIPS).* 2014. P. 3104–3112.

13. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I. Attention Is All You Need. Advances in Neural Information Processing Systems (NIPS). 2017. P. 5998–6008.

14. Wang Y., Zhao L. Evolutionary Strategies for Neural Network Training. *IEEE Transactions on Neural Networks and Learning Systems.* 2019. Vol. 30, No. 5. P. 1422–1432.

15. Zhou Z., Li Y., and Shi Y. Deep Neural Networks for Regression and Classification Tasks: A Review of the State of the Art. Neural Networks. 2020. Vol. 132. P. 223–247.

16. Zhang S., Yao L., Sun A., Tay Y. Deep Learning based Recommender System: A Survey and New Perspectives. *ACM Computing Surveys*. 2020. Vol. 52, No. 1. P. 1–38.

17. Gao Y., Li Y.-F., Lin Y., Gao H., Khan L. Deep Learning on Knowledge Graph for Recommender System: A Survey. arXiv preprint. 2020.

18. Dong M., Yuan F., Yao L., Wang X., Xu X., Zhu L. Survey for Trust-aware Recommender Systems: A Deep Learning Perspective. arXiv preprint. 2020.

19. Meng Z., McCreadie R., Macdonald C., Ounis I. Exploring Data Splitting Strategies for the Evaluation of Recommendation Models. arXiv preprint. 2020.

20. Zangerle E., Bauer C. Evaluating Recommender Systems: Survey and Framework. *ACM Computing Surveys.* 2022. Vol. 55, No. 8.

21. Li X., Chen Y., Xu L. Efficiency Evaluation of IT Projects in AI-Based Recommendation Systems. *Journal of Information Science and Engineering*. 2023. Vol. 39, No. 2. P. 105–120.

22. Wang H., Yang Y., Gao J. Deep Learning for Recommendation Systems: A Comprehensive Review. *IEEE Transactions on Neural Networks and Learning Systems.* 2022. Vol. 33, No. 3. P. 220–235.

23. Hu Y., Koren Y., Volinsky C. Machine Learning in Recommender Systems: Advances and Trends. *Journal of Artificial Intelligence Research.* 2023. Vol. 68. P. 120–135.

24. Zhou K., Li W., Zhao D. Deep Learning-Based Recommendation Models Combining Pre-Processing Methods and Semantic Segmentation. *Technology and Health Care.* 2022. Vol. 30. P. 173–190.

25. Nguyen T., Diaz A., McCallum A. Personalization in E-commerce Recommendation Systems. *Journal of Machine Learning Research*. 2020. Vol. 20. P. 1–28.

26. Bobadilla J., Ortega F., Hernando A., Gutiérrez A. Recommender Systems Survey. *Knowledge-Based Systems.* 2021. Vol. 212. P. 109–132.

27. Lu J., Wu D., Mao M., Wang W., Zhang G. Recommender System Applications. *Artificial Intelligence Review*. 2022. Vol. 45, No. 3. P. 1–58.

28. Zhang Y., Wang X., Wu L. A Survey on Hybrid Recommender Systems. *Expert Systems with Applications*. 2021. Vol. 165. P. 113722.

29. Ricci F., Rokach L. Advances in Recommender Systems: Beyond Traditional Approaches. *IEEE Transactions on Knowledge and Data Engineering*. 2020. Vol. 32, No. 4. P. 697–712.

30. Koren Y., Bell R., Volinsky C. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer*. 2020. Vol. 42, No. 8. P. 30–37.

31. Schafer J. B., Konstan J., Riedl J. E-Commerce Recommendation Applications. *Data Mining and Knowledge Discovery*. 2021. Vol. 5, No. 1–2. P. 115–153.

32. Adomavicius G., Tuzhilin A. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Transactions on Knowledge and Data Engineering*. 2021. Vol. 17, No. 6. P. 734–749.

33. Terveen L., Hill W. Beyond Recommender Systems: Helping People Help Each Other. *HCI in the New Millennium*. 2022. P. 487–509.

34. Dacrema M. F., Cremonesi P., Jannach D. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. *ACM Conference on Recommender Systems.* 2022. P. 101–109.

35. Gomez-Uribe C. A., Hunt N. The Netflix Recommender System: Algorithms, Business Value, and Innovation. *ACM Transactions on Management Information Systems.* 2022. Vol. 6, No. 4. P. 13–32.

36. Brown T., Mann B., Ryder N., et al. Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems (NeurIPS)*. 2020.

37. Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2017. P. 1251–1258.

38. Devlin J., Chang M. W., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT.* 2019. P. 4171–4186.

39. Dosovitskiy A., Beyer L., Kolesnikov A., et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations (ICLR)*. 2020.

40. Hagan M. T., Menhaj M. Training Feedforward Networks with the Marquardt Algorithm. *IEEE Transactions on Neural Networks*. 1994. Vol. 5, No. 6. P. 989–993.

41. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2016. P. 770–778.

42. Kingma D. P., Ba J. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR).* 2015.

43. Nguyen T. T., Lee J. Co-evolutionary Algorithms for Neural Network Optimization. *Neurocomputing*. 2019. Vol. 342. P. 123–132.

44. Pérez F., García J. Genetic Algorithms in Machine Learning: A Survey. *Artificial Intelligence Review.* 2017. Vol. 48, No. 4. P. 567–589.