Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи короткострокового прогнозування часових рядів на основі обчислювального інтелекту

(тема)

Виконав:

студент II курсу, групи СПм-21-2

Понамарьов В.О.
(прізвище, ініціали)

Спеціальність

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

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

Освітня програма

Системне програмування
(повна назва освітньої програми)

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

Допускається до захисту

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

2023 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

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

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

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
"____" _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Понамарьову Владиславу Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Методи короткострокового прогнозування часових рядів на основі обчислювального інтелекту

затверджена наказом по університету від " 03 " квітня 2023 р. № 318 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 17 травня 2023 р.

3. Вхідні дані до роботи

1) документація мови програмування Python;
2) документація бібліотеки Keras;
3) інтегроване середовище: JetBrains PyCharm 2023;
4) набори даних SaveDnipro про індекс забруднення повітря.

4. Перелік питань, що потрібно опрацювати у роботі

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

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

Слайд-презентація – 16 слайдів

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	04.04.23-07.04.23	
2	Вибір та обґрунтування методики дослідження	08.04.23-13.04.23	
3	Вибір інструментальних засобів	14.04.23-18.04.23	
4	Розробка алгоритмічного забезпечення	19.04.23-25.04.23	
5	Проведення експериментів	26.04.23-03.05.23	
6	Оформлення матеріалів кваліфікаційної роботи	04.05.23-08.05.23	
7	Подання кваліфікаційної роботи керівникові та	10.05.23-11.05.23	
	її попередній захист		
8	Подання кваліфікаційної роботи на	12.05.23-16.05.23	
	рецензування		

Дата видачі завдання 03 квітня 2023 р.

Студент _____
(підпис)

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

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 88 с., 34 рис., 15 табл., 3 дод., 38 джерел.

ОБЧИСЛЮВАЛЬНИЙ ІНТЕЛЕКТ, ШТУЧНА НЕЙРОННА МЕРЕЖА, ЧАСОВІ РЯДИ, ПРОГНОЗУВАННЯ, MLP, LSTM, CNN, MAE, MAPE, KERAS, PYTHON.

Метою кваліфікаційної є дослідження ефективності використання методів обчислювального інтелекту для короткострокового прогнозування часових рядів.

У ході виконання кваліфікаційної роботи проведено аналіз існуючих методів обчислювального інтелекту для прогнозування часових рядів, їх переваг та недоліків. Були реалізовані моделі короткострокового прогнозування на основі штучних нейронних мереж, таких як багатошаровий персептрон, довга короткострокова пам’ять та згорткова нейронна мережа.

Дослідження проводилися із застосуванням даних щодо індексу забруднення повітря зібрані організацією SaveDnipro.

Розроблений програмний продукт дозволяє прогнозувати часові ряди за допомогою моделей MLP, LSTM, CNN та проводити порівняльний аналіз.

ABSTRACT

Master’s thesis: 88 pages, 34 figures, 15 tables, 3 appendices, 38 sources.

COMPUTATIONAL INTELLIGENCE, ARTIFICIAL NEURAL NETWORK, TIME SERIES, FORECASTING, MLP, LSTM, CNN, MAE, MAPE, KERAS, PYTHON.

The major goal of this thesis is research of computational intelligence methods for short-term time series forecasting.

In the process of the qualification work performed an analysis of existing methods of computational intelligence for time series forecasting, their advantages and disadvantages. There were research artificial neural networks, model multilayer perceptron, long short-term memory, and a convolutional neural network.

The research was conducted using air pollution index data collected by the SaveDnipro organization.

The developed software product allows forecasting time series using MLP, LSTM, and CNN models and comparing the results.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ГА – генетичний алгоритм

ШІС – штучка імунна система

ШНМ – штучна нейронна мережа

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

LSTM – довга короткострокова пам’ять (англ., Long Short-Term Memory)

MAE – середня абсолютна помилка (англ., Mean absolute error)

MAPE – середня абсолютна похибка у відсотках (англ., Mean absolute percentage error)

MLP – багатошаровий персептрон (англ., Multilayer Perceptron)

ВСТУП

Аналіз даних, представлених у вигляді часових рядів, є одним із ключових інструментів у сучасному світі, де дані відіграють все більш важливу роль у прийнятті рішень та плануванні процесів.

Поширення аналізу даних, представлених у вигляді часових рядів, відбувається у зв'язку з розвитком сучасних технологій, таких як машинне навчання та обчислювальний інтелект [1].

Основними завданнями аналізу часових рядів є прогнозування та аналіз трендів, сезонності та циклічності в даних [1]. Для досягнення цих цілей використовуються різні методи, такі як авторегресійна інтегрована середня ковзна (ARIMA), експоненційне згладжування, штучні нейронні мережі (ШНМ), генетичні алгоритми (ГА) та штучні імунні системи (ШІС). Ці методи дозволяють моделювати часові ряди та прогнозувати їх майбутні значення на основі історичних даних. Крім того, вони дозволяють виявляти приховані закономірності та залежності в даних, що може бути корисно для прийняття рішень та оптимізації процесів [1]. Найбільш поширеними серед засобів обчислювального інтелекту є штучні нейронні мережі, які надають потужний інструмент для прогнозування часових рядів, і за останні роки було розроблено велика кількість методів та алгоритмів, які дозволяють покращити точність та ефективність прогнозування [1].

Поширеним підходом є рекурентні нейронні мережі (RNN), які добре підходять для роботи з послідовностями даних, якими є часові ряди. Однак RNN мають проблему із загасанням градієнта, через що відбувається втрата інформації на довгих проміжках часу. Для вирішення цієї проблеми розроблено різні модифікації RNN, такі як довга короткострокова пам'ять (LSTM) та керовані рекурентні блоки (GRU).

Методи штучних імунних систем [2] використовуються для прогнозування часових рядів, шляхом моделювання поведінки штучної

імунної системи у відповідь на середовище, що змінюється. Наприклад, алгоритми, засновані на антигенних взаємодіях, можуть бути використані для пошуку шаблонів у часових рядах та прогнозування майбутніх значень на основі цих шаблонів [2]. Прикладом штучних імунних систем, які можуть використовуватися для прогнозування часових рядів, є алгоритм AIS-ARIMA. Цей алгоритм поєднує методи ШІС та класичну методику ARIMA для прогнозування часових рядів. AIS-ARIMA використовує ШІС для знаходження початкових параметрів моделі ARIMA, а потім застосовує метод ARIMA для прогнозування майбутніх значень.

Методи на основі генетичних алгоритмів можуть використовуватись для прогнозування часових рядів шляхом створення та оптимізації моделей прогнозування [2]. Генетичні алгоритми використовуються для визначення оптимальних параметрів моделі прогнозування часових рядів, таких як параметри авторегресійної моделі (AR) або ковзного середнього (MA). ГА можуть використовуватися для вибору оптимальної комбінації параметрів моделі, що призводить до покращення точності. Також ГА можуть використовуватися для створення ансамблів моделей прогнозування часових рядів [2]. Ансамбль складається з кількох індивідуальних моделей, які можуть працювати спільно для вирішення завдання прогнозування.

Для ефективного прогнозування часових рядів необхідно мати достатню кількість даних для навчання моделі. Якщо даних недостатньо, може використовуватися техніка ресемплінга [2], яка дозволяє збільшити кількість даних шляхом їх перевикористання або стиснення. Також важливо проводити аналіз якості даних та виявляти та виправляти аномальні викиди та помилки в даних, щоб підвищити точність моделі.

У роботі прогнозування часових рядів розглянуто в контексті прогнозування індексу забруднення повітря в Україні. В якості вихідних даних для навчання засобів обчислювального інтелекту використані дані, зібрані організацією SaveDnipro.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність завдання прогнозування

Прогнозування є важливим завданням у багатьох областях, включаючи фінанси, економіку, медицину, кліматологію, екологію, транспорт.

У фінансовій сфері прогнозування є невід'ємною частиною інвестиційного аналізу [1]. Воно допомагає інвесторам та трейдерам приймати обґрунтовані рішення щодо купівлі або продажу активів на фондовому ринку. Наприклад, прогнозування цін на акції, валюти чи товари може допомогти інвесторам визначити, коли краще купити чи продати активи. Також прогнозування рядів може допомогти в управлінні ризиками, оскільки інвестори можуть використати прогнози, щоб ухвалити рішення про те, скільки грошей слід інвестувати у кожен актив [1].

В економіці прогнозування є необхідним інструментом для планування бізнес-стратегії та бюджетування. Прогнози економічних показників, таких як інфляція чи безробіття, допомагають урядам та підприємствам приймати рішення про витрати та доходи, а також визначати, коли необхідно змінювати стратегію бізнесу. Наприклад, прогнози економічних показників можуть допомогти компаніям приймати рішення про розширення бізнесу або коли слід закривати частину підприємства [2].

У медицині прогнозування може бути використане для прогнозування захворювань, ефективності лікування. Наприклад, прогнозування поширення захворювань може допомогти керуючим організаціям у плануванні необхідної кількості медичних працівників та обладнання. Також воно може допомогти у вирішенні питань, пов'язаних з дистанційним моніторингом стану пацієнтів та запобіганням ускладненням [3].

У кліматології прогнозування може бути використане для визначення запобіжних заходів у разі несприятливих погодних умов, таких як урагани,

торнадо, посуха або повені. Наприклад, прогнози погоди можуть допомогти урядам та приватним організаціям приймати рішення про закупівлю продуктів харчування та ліків, організацію евакуації населення [4].

В екології прогнозування може бути використане для дослідження динаміки забруднення повітря, води та ґрунту, аналізу процесів зміни клімату та екологічних параметрів. Наприклад, прогнозування забруднення повітря дозволяє державним органам вживати запобіжних заходів для зменшення викидів забруднюючих речовин, а також організації екстрених заходів при надзвичайних ситуаціях. Також прогнозування клімату може допомогти у прийнятті рішень щодо сільськогосподарського виробництва та інших галузей, що залежать від погодних умов [5].

У транспорті прогнозування використовується для оптимізації розкладу та управління транспортними потоками. Наприклад, дані про трафік та використання громадського транспорту можуть бути застосовані для прогнозування кількості пасажирів у майбутньому та оптимізації маршрутів та розкладу громадського транспорту [6].

З розвитком технологій та появою великих даних, завдання прогнозування стає все більш важливим та актуальним. Вихідні дані для завдання прогнозування зазвичай представлені у вигляді часових рядів.

1.2 Основні поняття часових рядів

Часовий ряд – це послідовність значень, виміряних у різні моменти часу. Він складається з набору даних, де кожне значення відповідає певному моменту часу чи періоду. Часові ряди можуть бути використані для аналізу та прогнозування поведінки систем у часі. Вони можуть бути представлені у вигляді графіків або таблиць [7].

Аналіз часових рядів може бути корисним для прогнозування майбутніх значень, виявлення трендів, циклів, сезонних коливань, аномалій та інших характеристик ряду. Аналіз часових рядів включає застосування

статистичних методів для опису і розуміння закономірностей в даних, а також для прогнозування майбутніх значень.

Часові ряди можна розділити на детерміновані та недетерміновані (рисунок 1.1), а також за наявністю або відсутністю тенденцій і сезонних ефектів на стаціонарні та нестаціонарні (рисунок 1.2).

Детермінований часовий ряд – це часовий ряд, в якому значення в кожний момент часу залежать тільки від часу і не мають випадкової складової [8]. Таким чином, кожне значення часового ряду можна точно передбачити з урахуванням знання часу.

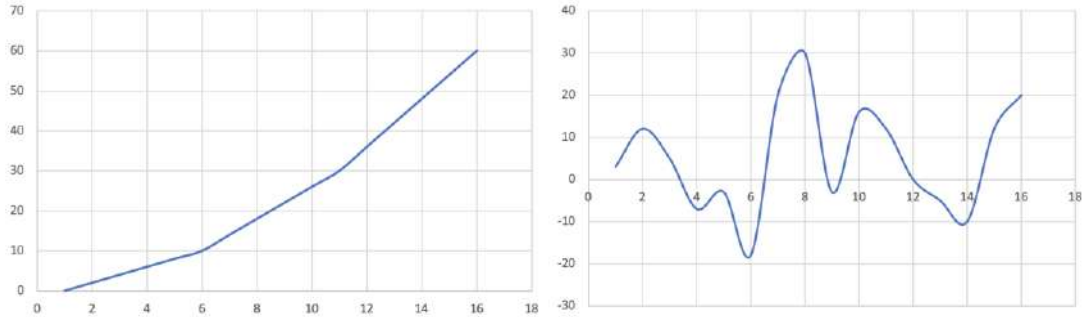

Рисунок 1.1 – Детермінований та недетермінований часовий ряд

Стаціонарний часовий ряд – це ряд, у якого статистичні характеристики не змінюються з часом [9]. Такі характеристики можуть включати середнє значення, дисперсію, автокореляцію і коваріацію. Якщо ці характеристики залишаються постійними у часі, можна припустити, що майбутні значення будуть схожі на минулі значення, що спрощує завдання прогнозування.

Стаціонарний часовий ряд може бути представлений у вигляді шуму (білий шум), коли значення ряду незалежні та однаково розподілені в часі, або у вигляді гармонійних коливань (синусоїд), коли амплітуда та частота залишаються постійними.

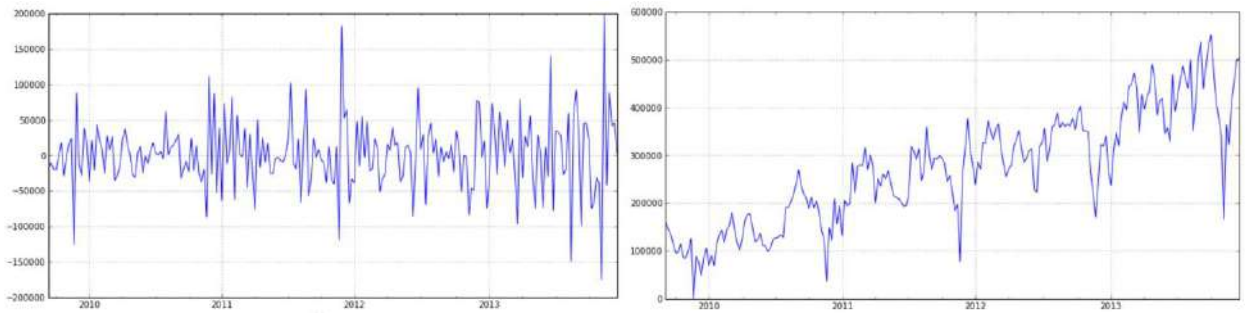

Рисунок 1.2 – Стаціонарний та нестаціонарний часовий ряд

Перевірка стаціонарності часового ряду включає аналіз його статистичних характеристик і графічний аналіз, такий як побудова часового графіка, гістограми і графіка автокореляції. Якщо статистичні характеристики ряду змінюються з часом, ряд вважається нестаціонарним і вимагає додаткової обробки, щоб стати стаціонарним.

Короткостроковість визначається значенням горизонту прогнозування. Горизонт прогнозування – це кількість часових періодів у майбутньому, на які проводиться прогноз. Горизонт прогнозування є важливим параметром при побудові моделей прогнозування часових рядів, тому що він визначає, як далеко в майбутнє робитимуться прогнози. При виборі горизонту прогнозування потрібно враховувати як мету прогнозування, так і характеристики часового ряду, такі як періодичність, тренди, сезонність [10].

Чим більший горизонт прогнозування, тим менш точними будуть прогнози, оскільки зі збільшенням горизонту прогнозування збільшується кількість невизначеності та шуму, що впливають на часовий ряд. Тому важливо підбирати оптимальний горизонт прогнозування, який дозволить отримати результати прогнозування з допустимою точністю.

Періодичність часового ряду означає, що в даних присутній патерн, що повторюється, або цикл з певною частотою [10], через що значення часового ряду змінюються у порядку і повертаються до вихідного значення через певний період. Період часового ряду може бути фіксованим (наприклад, денний, тижневий, місячний), так і змінним (наприклад, сезонне зростання

продажів у періоди свят може змінюватися від року до року). Визначення періодичності є важливим етапом в аналізі часових рядів, оскільки вона може допомогти у прогнозуванні майбутніх значень та виявленні трендів.

Тренд часового ряду – це загальна спрямованість зміни значень ряду протягом тривалого часу [11]. Тренд може бути висхідним (коли значення збільшуються) або низхідним (коли значення зменшуються). Тренд буває лінійним, коли зміна значень ряду відбувається поступово протягом усього періоду, або нелінійним, коли швидкість зміни значень змінюється з часом.

Визначення тренду може бути корисним для прогнозування майбутніх значень часового ряду та розуміння довгострокових змін у даних. Однак слід враховувати, що тренд не є єдиною складовою часового ряду і може вплинути на інші фактори, такі як сезонність.

Сезонність часового ряду – це циклічний патерн в вихідних даних, що повторюється в певний період часу [11]. Сезонність може бути як адитивною, так і мультиплікативною. Адитивна сезонність означає, що коливання ряду протягом року залишаються приблизно одному рівні, тоді як мультиплікативна сезонність означає, що коливання ряду зростають разом із збільшенням рівня ряду.

Прогнозування часових рядів полягає у створенні моделі для визначення майбутніх значень часового ряду на основі наявних даних та передбачення майбутніх даних до їх вимірювання. Прогнозування може здійснюватися за допомогою різних методів та моделей, залежно від типу даних та завдання, яке необхідно вирішити.

1.3 Класифікація методів та моделей прогнозування

Прогнозування – це процес оцінки ймовірних майбутніх подій або умов на основі наявних даних та аналізу їх тенденцій та патернів. Сучасні алгоритми машинного навчання та аналізу даних дозволяють більш точно та швидко прогнозувати різні параметри та події.

Точність прогнозування часових рядів засобами обчислювального інтелекту залежить від багатьох чинників, таких як вибір методу, наявність достатньої кількості даних, вибір ознак, доступність необхідних обчислювальних засобів [12]. Частина моделей та відповідних методів відноситься до окремих процедур процесу прогнозування. Частина методів представляє лише набір окремих прийомів, що відрізняються від базових або один від одного кількістю етапів отримання прогнозу та послідовністю їх застосування.

Усі методи прогнозування можна розділити на дві групи: інтуїтивні та формалізовані [13].

Інтуїтивне прогнозування застосовується тоді, коли об'єкт прогнозування або занадто простий, або навпаки настільки складний, що аналітично врахувати вплив зовнішніх факторів неможливо. Інтуїтивні методи прогнозування не передбачають розробку моделей прогнозування та відображають індивідуальні міркування фахівців щодо перспектив розвитку процесу, що прогнозується. Такі методи використовуються для аналізу процесів, розвиток яких або повністю або частково не піддається математичній формалізації, тобто для яких важко розробити адекватну модель. До подібних методів належать такі методи: експертні оцінки, історичні аналогії.

Серед формалізованих методів прогнозування поширеними є ті, що основані на використанні статистичних моделей, у яких функціональна залежність між майбутніми та фактичними значеннями часового ряду, а також зовнішніми факторами, задана аналітично [14]. До статистичних моделей належать такі групи як регресійні моделі, авторегресійні моделі, модель експоненційного згладжування.

Функціональна залежність між фактичними та майбутніми значеннями, задана структурно у моделях структурного аналізу [14]. До структурних моделей належать такі групи як, нейромережеві моделі, моделі ланцюгів Маркова, моделі на базі регресійних дерев.

1.4 Аналіз існуючих досліджень

Для формування задачі та вибору методів обчислювального інтелекту для подальшої реалізації та вдосконалення, проаналізовано існуючі дослідження у сфері прогнозування часових рядів. Розглянуті роботи присвячені використанню поширених моделей та методів прогнозування та, у випадку рішення вузькоспеціалізованих завдань, застосуванню особливих моделей прогнозування, які базуються одночасно на декількох окремих засобах.

1.4.1 Моніторинг забруднення повітря за допомогою ШНМ

Основною метою роботи було прогнозування погодинних рівнів п'яти забруднюючих речовин у шести точках у районі Більбао (Іспанія). З цією метою було побудовано двісті шістнадцять моделей на основі ШНМ [15].

Дані, що використовуються для прогнозування, були історичними записами транспортних, метеорологічних мереж і мереж спостерігання забруднення повітря, що існували у відповідному районі в 2000 році. Потім навчені моделі були протестовані на даних із тих самих мереж 2001 року.

На першому етапі, для кожного з двісті шістнадцяти випадків було побудовано сто моделей на основі різних типів нейронних мереж з використанням даних 2000 року. Остаточна ідентифікація кращої моделі ШНМ проводилася за критеріями достовірності щонайменше 95% кращих значень стосовно даних 2001 року.

Використання розглянутих моделей, заснованих на нейронних мережах, може забезпечити мережу забруднення повітря Більбао, спочатку призначену з метою діагностики, можливостями короткострокового прогнозування. Відповідно до результатів дослідження, отримані моделі штучних нейронних мереж показали кращу ефективність по відношенню до використання статистичних методів прогнозування.

1.4.2 Прогнозування рівнів озону за допомогою MLP

Метою розглянутого дослідження було розробка двоетапної методології для прогнозування рівнів концентрації тропосферного озону на кілька годин наперед, що поєднує метеорологічні дані, дані про якість повітря та дані про промислові викиди на трьох станціях моніторингу якості повітря в Сінині (Португальський регіон) [16]. Набори даних включають метеорологічні параметри, параметри якості повітря та промислових викидів з високою часовою роздільною здатністю, зібрані за період 2006–2009 років.

Найкращі предиктори концентрації тропосферного озону були визначені за допомогою методів класифікації та дерев регресії, а потім було прийнято багаторівневі моделі персептрону для прогнозування рівнів тропосферного озону для кожного місця моніторингу.

Навчені моделі MLP потенційно можуть бути реалізовані в інструменті реального часу для рекомендацій з охорони здоров'я та довкілля, що дозволить покращити місцеву систему управління якістю повітря. Якщо зважити на те, що кожна станція моніторингу має свою модель MLP, то просторово-часова модель, орієнтована на громадську охорону здоров'я, на основі стохастичних імітаційних моделей може бути побудована для прогнозування концентрацій забруднювачів повітря.

Розглянуте дослідження доводить перспективність використання архітектури MLP для багатокритеріального прогнозування у режимі реального часу.

1.4.3 Порівняння ARIMA та LSTM у прогнозуванні часових рядах

У роботі розглядається алгоритм на основі глибокого навчання для прогнозування часових рядів (довга короткострокова пам'ять), і порівняння його з традиційними методами, такими як авторегресійне інтегроване ковзне середнє [17].

Експериментальні дослідження проводилися на фінансових часових рядах великого обсягу.

Емпіричні дослідження показали, що алгоритми з урахуванням глибокого навчання, такі як LSTM, перевершують традиційні алгоритми, такі як модель ARIMA. Зокрема, середнє зниження частоти помилок, отримане за допомогою LSTM, становило близько 85% (у метриці MAPE) у порівнянні з ARIMA, що вказує на перевагу LSTM. Крім того, було помічено, що кількість періодів навчання не мала суттєвого впливи на продуктивність навченої моделі прогнозування.

1.4.4 Прогнозування часових рядів за допомогою CNN

У дослідженні [18] запропонована згорткова нейронна мережа яка самонавчається для прогнозування часових рядів. Модель CNN була навчена з використанням часових рядів різної природи походження. Після закінчення обробки потоку часових рядів CNN знову самонавчалася, використовуючи як вхідні дані раніше отримані правильні дані. Потім модель використовувалася для прогнозування даних часового ряду.

У роботі як зразки фінансових часових рядів було обрано найбільші наразі фінансові ринки Китаю – всі акції фондових ринків Шанхаю та Шеньчженю. Ціни закриття, обсягу та ринкової ціни всіх акцій з 2000 по 2009 рік використовувалися для навчання моделі згорткової нейронної мережі. Зібрано ціни закриття, обсягу та ринкову ціну всіх акцій з 2010 по 2015 рік у вигляді трьох різних типів тестового набору даних. Прогнозування даних відбувалося з використанням моделі, навченої у 2009 році. Всі типи даних часових рядів потребують нормалізації, щоб відповідати моделі згорткової нейронної мережі.

Оцінка прогнозу за допомогою CNN, що проводилася на фінансових часових рядах, показала, що запропонований метод працює краще, ніж традиційний метод смуг Болліджера.

1.4.5 Прогнозування часових рядів за допомогою ШІС

У роботі запропоновано комбінований метод прогнозування часових рядів за допомогою штучних імунних систем. Метою дослідження є розробка імунного алгоритму короткострокового прогнозування часових рядів на основі моделі клонального відбору, що використовує метод виведення по прецедентах (CBR).

У ході експериментальних досліджень було проведено короткострокове прогнозування рядів, що використовуються в M3-Competition, а також середньодобових показань температури.

Порівняльний аналіз результатів гібридного методу прогнозування часових рядів за допомогою штучних імунних систем та моделі на основі штучної імунної мережі показали, що використання клонального відбору дозволяє краще прогнозувати ряди з меншою передісторією [19].

Результат прогнозування значень середньодобових показань температури підтверджує перевагу використання підходів штучних імунних систем. Однак, при малій кількості значень часового ряду при отриманні прогнозу традиційні методи мають переваги перед гібридним методом прогнозування часових рядів на основі штучних імунних систем.

1.4.6 Прогнозування на основі нечітких часових рядів і ГА

У роботі досліджується гібридна модель FTSGA, заснована на нечітких часових рядах та генетичному алгоритмі. FTSGA зменшує похибку прогнозування, застосовуючи оператори генетичного алгоритму, такі як відбір, кросовер та мутацію, для вдосконалення процесу отримання прогнозних значень [20].

Для демонстрації ефективності моделей FTSGA вибрано дані щоденних цін закриття TAIEX, що охоплюють період з 1990 по 1999 рік, як набор даних для перевірки та порівняння їх ефективності з ефективністю

існуючих традиційних моделей. Дані з січня до жовтня кожного року використовувалися для оцінки, а дані за листопад та грудень – для прогнозування. Гібридна модель на нечітких часових рядах та генетичному алгоритмі у порівнянні з трьома різними традиційними моделями нечітких часових рядів має переваги в прогнозі набору даних фондових ринків TAIEX.

1.5 Постановка задачі

Метою роботи є дослідження методів короткострокового прогнозування часових рядів на основі засобів обчислювального інтелекту.

Як дані для дослідження методів короткострокового прогнозування часових рядів на основі обчислювального інтелекту вибрано індекс забруднення повітря в України, прогнозування якого дозволяє оцінити рівень забруднення та його ймовірний вплив на здоров'я людей та екологічну ситуацію. Це є важливим інструментом для прийняття рішень у галузі екології та охорони здоров'я, таких як розробка та реалізація заходів щодо зниження викидів шкідливих речовин в атмосферу, встановлення нормативів та контролю за забрудненням повітря, а також проведення моніторингу та аналізу якості повітря. Крім того, прогнозування індексу забруднення повітря дозволяє вживати заходів щодо захисту здоров'я населення, таких як рекомендації щодо використання захисних масок або регулювання фізичної активності.

Дані стосовно індексу забруднення повітря можуть буди представлені у вигляді часових рядів, для прогнозування яких доцільним є використання обчислювального інтелекту. На основі аналізу існуючих досліджень для вирішення задачі були обрані моделі штучних нейронних мереж, такі як багатошаровий персептрон, довга короткострокова пам'ять та згорткова нейронна мережа, оскільки вони показують найкращі результати при короткостроковому прогнозуванні часових рядів аналогічної природи.

2 ВИКОРИСТАННЯ ЗАСОБІВ ОБЧИСЛЮВАЛЬНОГО ІНТЕЛЕКТУ

2.1 Штучні нейронні мережі

Штучні нейронні мережі – це моделі, що використовуються у машинному навчанні для вирішення завдань класифікації, розпізнавання, регресії, кластеризації та прогнозування. ШНМ є математичними моделями, які імітують роботу нервової системи живих організмів, у яких велика кількість нейронів пов'язані між собою [21].

ШНМ складаються з шарів нейронів, які обробляють вхідні дані та передають результати на наступний шар штучних нейронів, поки не досягнуть останнього шару, який видає кінцевий результат (рисунок 2.1). Кожен нейрон в ШНМ має свої ваги та зміщення, які налаштовуються в процесі навчання, для забезпечення адаптації мережі до певних завдань [21].

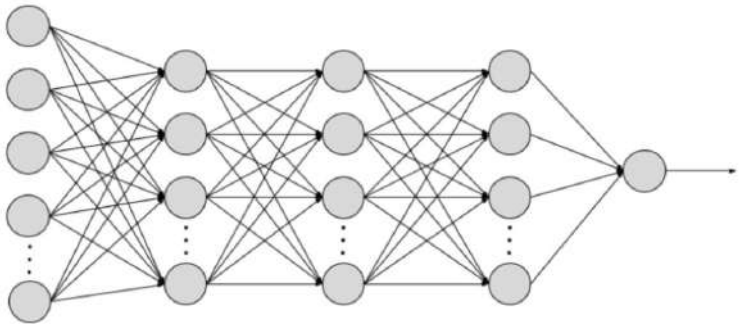

Рисунок 2.1 – Штучна нейронна мережа

ШНМ можуть бути навчені на великих обсягах даних, що робить їх придатними для виконання завдань, які зазвичай вважалися складними для машинної обробки. ШНМ знаходять широке застосування в обробці зображень та звуку, розпізнаванні мовлення, прогнозуванні часових рядів, медичній діагностиці, фінансах та інших галузях.

2.2 Навчання та використання штучних нейронних мереж

Одним із найпоширеніших алгоритмів навчання штучних нейронних мереж є алгоритм зворотного поширення помилки (Backpropagation) [22]. Його основний принцип полягає в тому, щоб налаштувати ваги зв'язків між нейронами в мережі таким чином, щоб мінімізувати помилку результату.

Наведено основні кроки алгоритму зворотного поширення помилки.

1) Пряме поширення. Передача вхідних значень через усі шари нейронів мережі та отримання вихідних значень.

2) Обчислення помилки. Обчислюється помилка між отриманим результатом та фактичним значенням. Для цього використовується функція втрат, яка може бути різною залежно від завдання.

3) Зворотна помилка поширюється через мережу, починаючи з останнього шару і до першого. Для кожного нейрона обчислюється величина помилки, яка пропорційна вкладу нейрона у загальну помилку.

4) Оновлення ваг. Використовуючи обчислену помилку та градієнт функції активації, оновлюються ваги зв'язків між нейронами. Це дозволяє знизити помилку у наступній ітерації навчання.

5) Повторення. Повторюються кроки 1-4 для кожного спостереження в навчальній вибірці, доки не буде досягнуто бажаної точності.

6) Регуляризація. Щоб уникнути перенавчання моделі, часто використовують регуляризацію.

7) Налаштування параметрів. У процесі навчання необхідно вибрати оптимальні параметри, такі як кількість шарів та нейронів у кожному шарі, а також тип функції активації. Ці параметри можуть значно впливати на якість навчання моделі.

8) Перевірка моделі. Після закінчення навчання необхідно перевірити якість моделі на тестовій вибірці, яка не використовувалася під час навчання. Це дозволяє оцінити узагальнюючу здатність моделі та виявити можливі проблеми.

9) Донавчання. Якщо модель показує недостатню точність на тестовій вибірці, можна продовжити навчання на навчальній вибірці або застосувати техніки донавчання, такі як fine-tuning.

10) Застосування моделі. Після успішної перевірки моделі її можна застосовувати для вирішення задачі, для якої вона була створена.

11) Оцінка продуктивності. Після застосування моделі можна оцінити її продуктивність у реальних умовах та порівняти з іншими моделями чи методами вирішення задачі.

12) Оновлення моделі. Якщо отримані нові дані або змінюються умови завдання, модель може знадобитися оновити. Це може включати в себе перенавчання на нових даних або зміну архітектури моделі.

13) Розгортання моделі. Після завершення навчання та перевірки моделі її можна розгорнути у продакшн-середовище для її застосування на практиці. Це може включати інтеграцію моделі з іншими системами та оптимізацію її продуктивності.

14) Моніторинг моделі. Після розгортання моделі необхідно її моніторити та періодично навчати на нових даних, щоб вона продовжувала надавати результати з допустимою точністю.

15) Оптимізація моделі. Для покращення продуктивності та ефективності моделі можна використовувати різні методи оптимізації, такі як квантизація, стиснення моделі або спеціалізовані апаратні прискорювачі.

В алгоритмі зворотного поширення помилки використовується градієнтний спуск для мінімізації помилки. Це означає, що навчання йде за градієнтом функції втрат у протилежному напрямку, щоб знайти локальний мінімум помилки. При цьому необхідно вибрати оптимальний розмір кроку, щоб не застрягти в локальному мінімумі та досягти глобального [23].

Для застосування алгоритму необхідно достатній обсяг даних та обчислювальні ресурси, а також оптимальні налаштування параметрів та моделей для конкретного завдання.

2.3 Проблема перенавчання штучних нейронних мереж

Перенавчання – це явище, коли модель штучної нейронної мережі стає надто спеціалізованою до навчальних даних, і починає показувати низьку точність на нових даних, які вона не обробляла раніше. Це відбувається, коли модель надмірно пристосувалася до шумів та зайвих деталей вхідних даних, які не мають спільної користі для вирішення задачі [24].

Існує кілька методів для запобігання перенавчанню такі як регуляризація, dropout, аугментація даних, рання зупинка, використання більш простих моделей [25].

Регуляризація – додавання штрафу у функцію втрат за використання великої кількості параметрів моделі. Наприклад, L1-регуляризація додає штраф за суму модулів усіх ваг, а L2-регуляризація – за суму квадратів ваг.

Dropout – випадкове виключення нейронів із мережі у процесі навчання. Це допомагає уникнути перенавчання шляхом запобігання надмірної залежності між нейронами.

Аугментація даних – це метод, який полягає у додаванні випадкових викривлень у навчальні дані для збільшення їх різноманітності та спільності. Це допомагає моделі узагальнювати знання нові дані, які можуть відрізнятися від навчальних даних.

Рання зупинка – це метод, який полягає у зупинці навчання, коли точність на тестових даних перестає покращуватись. Це дозволяє уникнути перенавчання шляхом зупинення навчання моделі, коли її узагальнююча здатність починає погіршуватися.

Використання простіших моделей – простіші моделі мають меншу ємність і менше схильні до перенавчання.

Запобігання перенавчанню є важливим аспектом навчання штучних нейронних мереж, і потребує ретельного аналізу та експериментів з різними методами та налаштуваннями моделі.

2.4 Багатошаровий персептрон

Багатошаровий персептрон (Multi-layer Perceptron, MLP) – це тип нейронної мережі, який складається з декількох шарів нейронів, включаючи вхідний, прихований та вихідний шари, які з'єднані між собою за допомогою зв’язків з певною вагою (рисунок 2.2) [26].

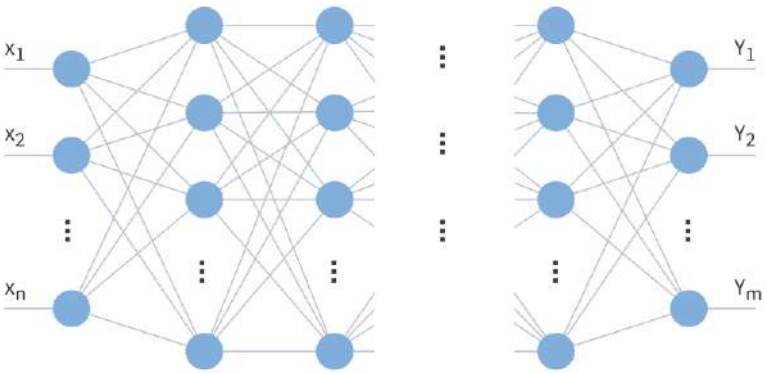

Рисунок 2.2 – Багатошаровий персептрон

На вхідний шар (рисунок 2.3) подаються дані, які є вектором значень. Кожне значення відповідає одній з ознак об'єкта, який необхідно класифікувати чи прогнозувати. Нейрони вхідного шару не виконують жодних обчислень, а передають значення на прихований шар.

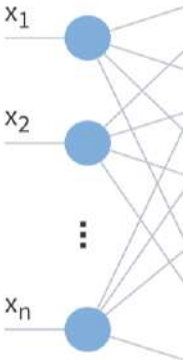

Рисунок 2.3 – Вхідний шар

Прихований шар (рисунок 2.4) виконує ряд обчислень, які дозволяють нейронній мережі визначити залежності між вхідними даними та цільовими значеннями. Кожен нейрон прихованого шару отримує вхідні дані від усіх нейронів попереднього шару і здійснює деякі обчислення з використанням своїх ваг та функції активації. Функція активації приймає на вхід зважену суму вхідних даних та ваги і повертає вихідне значення. Функція активації може бути, наприклад, сигмоїдою, гіперболічним тангенсом або ReLU.

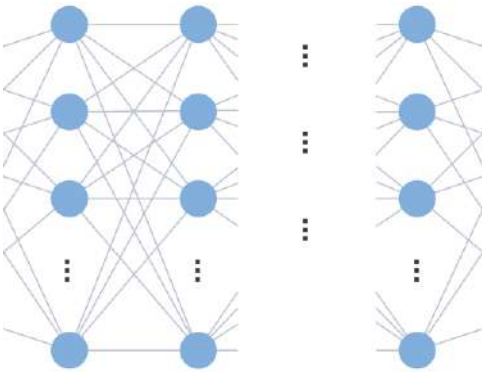

Рисунок 2.4 – Прихований шар

Вихідний шар (рисунок 2.5) приймає дані від прихованого шару та виконує фінальні обчислення. Функція активації вихідного шару залежить від типу завдання, яке потрібно розв'язати.

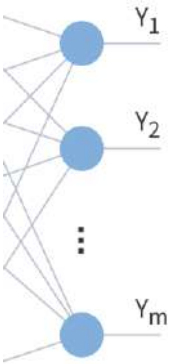

Рисунок 2.5 – Вихідний шар

Під час навчання MLP за допомогою алгоритму зворотного поширення помилки значення помилки на вихідному шарі порівнюються з фактичними значеннями, і ця помилка поширюється назад через нейронну мережу. Потім вага нейронної мережі оновлюється таким чином, щоб мінімізувати помилку. Цей процес повторюється для кожного об'єкта в навчальному наборі даних, поки ваги нейронної мережі не будуть оптимізовані [27].

Особливістю багатошарового персептрону є здатність автоматично отримувати ознаки з вхідних даних, що дозволяє мережі враховувати складні залежності між вхідними ознаками та цільовими змінними. Однак для успішного використання MLP необхідно правильно підібрати структурні параметри моделі, такі як кількість прихованих шарів, кількість нейронів у кожному шарі та функції активації [28].

Для поліпшення продуктивності та ефективності роботи багатошарового персептрону використовуються різні методи оптимізації, такі як проріджування та квантизація [28].

Проріджування дозволяє зменшити розмір MLP шляхом видалення нейронів, які не впливають на результати обчислень або дають малий внесок у загальну роботу мережі. Це може суттєво зменшити розмір моделі та прискорити її роботу без втрати точності пронозу.

Квантизація дозволяє зменшити кількість біт, що використовуються для зберігання ваг та активацій нейронів. Наприклад, використання 8-бітових замість 32-бітових чисел може суттєво знизити вимоги до пам'яті та прискорити обчислення на сучасних процесорах та графічних прискорювачах.

2.5 Довга короткострокова пам'ять

Довга короткострокова пам'ять (Long Short-Term Memory, LSTM) – це вид рекурентної нейронної мережі, здатний обробляти та зберігати послідовності даних з довгими залежностями між ними [29].

LSTM складається з вхідного шару, комірного шару забування, вхідного комірного шару, шару стану пам'яті та вихідного комірного шару (рисунок 2.6) [30].

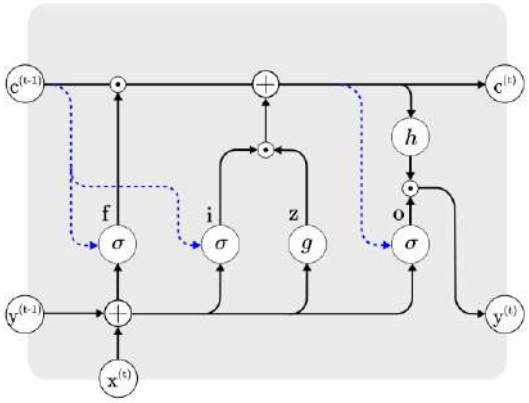

Рисунок 2.6 – Довга короткострокова пам'ять

Вхідний шар (рисунок 2.7) приймає на вхід поточні дані та передає їх далі по ланцюжку. Дані можуть бути будь-якого типу: слова в реченні, звуки в аудіозаписі, ціни на акції в торговій платформі або часові ряди.

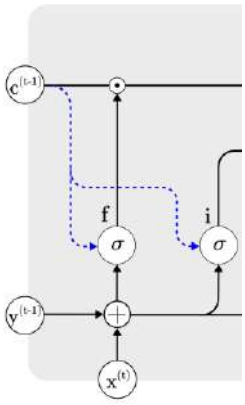

Рисунок 2.7 – Вхідний шар

Комірний шар забування (рисунок 2.8) вирішує, яка інформація має бути забута на основі попереднього стану блоку та поточного входу. Вхід і попередній стан блоку об'єднуються і проходять через сигмоїдну функцію,

яка повертає значення від 0 до 1. Значення близькі до 0 вказують, що інформацію слід забути, а значення, близькі до 1, вказують, що інформацію слід зберегти.

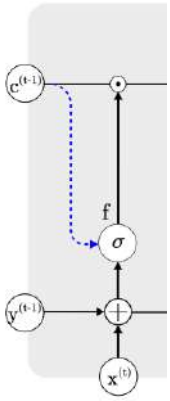

Рисунок 2.8 – Комірний шар забування

Вхідний комірний шар (рисунок 2.9) вирішує, яка інформація буде додана в блок, виходячи з попереднього стану блоку і поточного входу. Вхід та попередній стан блоку проходять через дві функції: сигмоїдну (щоб визначити, які значення слід оновити) та гіперболічний тангенс (щоб визначити, які значення слід додати до пам'яті блоку). Значення цих функцій перемножуються і додаються до попереднього стану блоку.

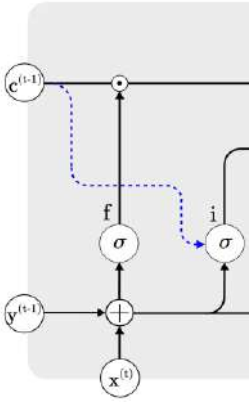

Рисунок 2.9 – Вхідний комірний шар

Стан пам'яті (рисунок 2.10). Це місце, де інформація зберігається протягом тривалого часу. Блок складається з кількох елементів пам'яті, кожен із яких може зберігати інформацію протягом тривалого часу.

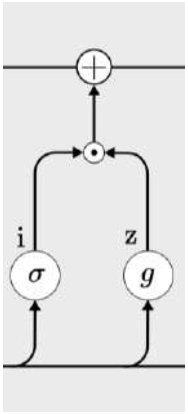

Рисунок 2.10 – Вхідний ворітний шар

Вихідний комірний шар (рисунок 2.11) визначає, яка інформація буде передана на вихід, виходячи з поточного стану блоку. Поточний стан блоку та поточний вхід об'єднуються та проходять через дві функції: сигмоїдну (щоб визначити, яка інформація буде передана на вихід) та гіперболічний тангенс (щоб визначити, яка інформація буде передана на вихід). Результати цих функцій перемножуються та передаються на вихід.

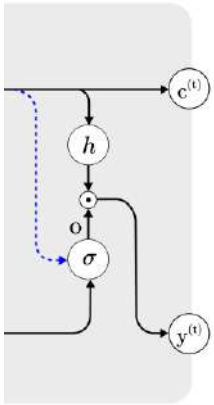

Рисунок 2.11 – Вхідний комірний шар

Однією з головних переваг LSTM є здатність зберігати та використовувати інформацію протягом тривалого часу та контролювати її потік. Крім того, LSTM може використовуватися у складі великих нейронних мереж і навчатися методом зворотного поширення помилки [31].

Одним із недоліків LSTM є висока обчислювальна складність, особливо для довгих послідовностей, які можуть вимагати багаторазового проходу через комірку LSTM. Для вирішення цієї проблеми запропоновані ефективніші алгоритми, такі як зменшення розмірності вхідних даних перед обробкою LSTM або використання інших типів рекурентних нейронних мереж, таких як GRU (Gated Recurrent Unit) [31].

2.6 Згорткова нейронна мережа

Згорткова нейронна мережа (Convolutional Neural Network, CNN) – це глибока нейронна мережа, яка використовується в комп'ютерному зорі для розпізнавання образів, класифікації зображень та інших завдань, пов'язаних із обробкою зображень [32].

CNN складається з згорткових шарів, шару пулінгу, повнозв'язного шару та функції активації (рисунок 2.12) [33].

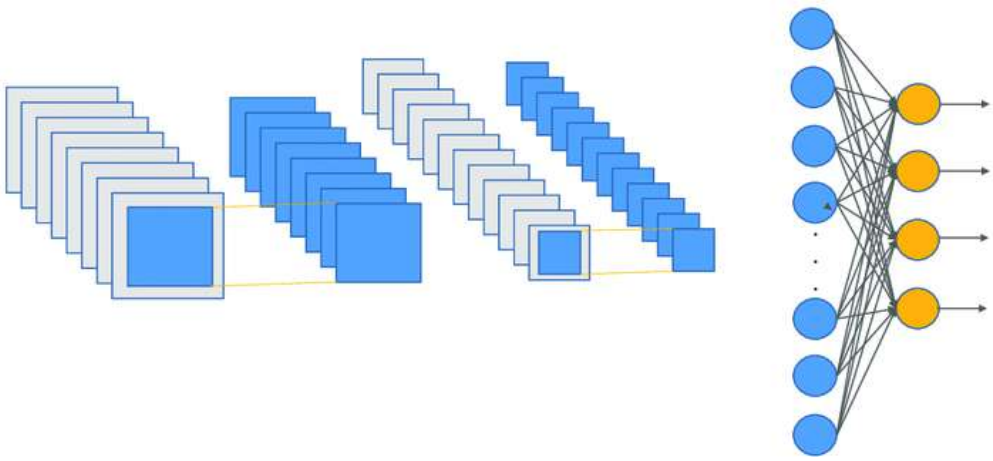

Рисунок 2.12 – Згорткова нейронна мережа

Перший шар згорткової мережі – це згортковий шар (рисунок 2.13). Він складається з декількох фільтрів, кожен з яких є невеликою матрицею. Фільтри застосовуються до вхідних даних, щоб одержати карту ознак, яка виділяє найбільш значущі ознаки даних. Кожен фільтр проходить по всім значенням даних, застосовуючи операцію згортки для отримання картки ознак.

Карта ознак є новими даними, які містить відображення того, як кожен фільтр співвідноситься з вихідним значенням. Кожне значення карти ознак являє собою суму значень у фільтрі та відповідних значень у вихідних даних. Таким чином, карта ознак містить інформацію про те, де на вхідних даних є найбільш значущі ознаки.

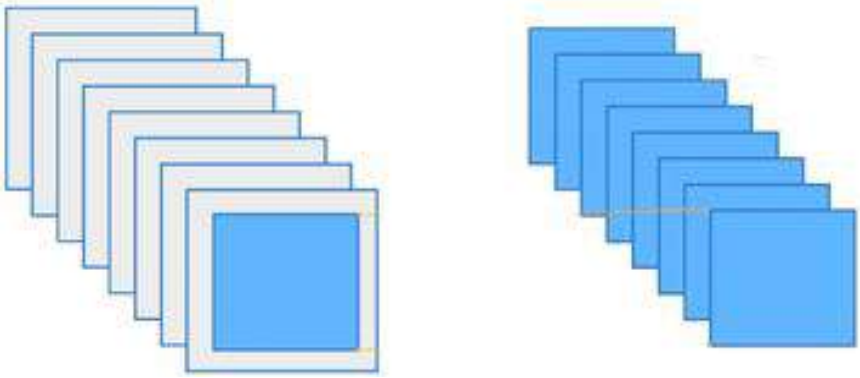

Рисунок 2.13 – Згортковий шар та шар пулінгу

Після кожного згорткового шару слідує шар пулінгу (рисунок 2.13). Його завдання – зменшити розмір карти ознак, знижуючи кількість параметрів для обробки та роблячи мережу більш стійкою до змін у позиції значень.

Існує кілька видів операцій пулінгу, але найпоширенішою є операція максимального пулінгу. Вона полягає у виборі максимального значення з певної області на карті ознак, щоб отримати нове значення на виході.

Після кількох шарів згортки і пулінгу дані передаються повнозв'язний шар (рисунок 2.14). Він складається з нейронів, кожен із яких з'єднаний з

усіма нейронами попереднього шару. Цей шар використовується для класифікації даних на основі ознак, виділених у попередніх шарах.

На виході повнозв'язкового шару застосовується функція активації, яка перетворює вихідні значення на ймовірність класів. Зазвичай для цієї мети використовується функція Softmax, яка перетворює значення ймовірності підсумовування до 1.

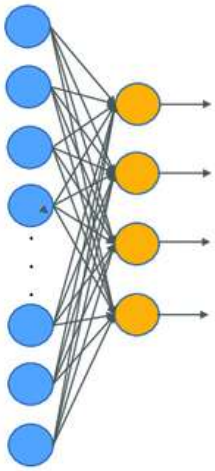

Рисунок 2.14 – Згорткова нейронна мережа

Навчання згорткової нейронної мережі відбувається шляхом подачі великої кількості даних на вхід мережі, а потім порівняння визначених класів із правильними класами (мітками) даних [34].

У процесі навчання ваги кожного шару згорткової мережі підлаштовуються таким чином, щоб мінімізувати помилку отримання результатів. Для цього використовується метод зворотного розповсюдження помилки, що дозволяє знаходити градієнти помилки по вагам зв’язків кожного нейрона в мережі. Ці градієнти використовуються для оновлення ваги таким чином, щоб мережа стала більш точною в передбаченні значень. У процесі навчання мережа згортки намагається вивчити найбільш значущі ознаки даних, які дозволяють правильно класифікувати значення [34].

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Використання можливостей Keras

Keras – це відкрита бібліотека з відкритим кодом для машинного навчання, написана мовою програмування Python. Keras надає простий та інтуїтивно зрозумілий інтерфейс для створення, тренування та оцінки глибоких нейронних мереж.

Бібліотека розроблена з упором на швидке прототипування і є однією з найпопулярніших бібліотек глибокого навчання серед дослідників та розробників. Вона надає високорівневі абстракції для створення різних типів нейронних мереж, таких як згорткові нейронні мережі (Convolutional Neural Networks, CNN), рекурентні нейронні мережі (Recurrent Neural Networks, RNN), рекурентні згорткові нейронні мережі (Recurrent Convolutional Neural Networks, RCNN), та багатошарові персептрони (Multilayer Perceptrons, MLP).

Keras пропонує широкий набір вбудованих функцій для обробки даних, регуляризації, оптимізації моделей, а також інструменти для візуалізації процесу навчання. Вона також є частиною бібліотеки глибокого навчання TensorFlow, що робить її найпопулярнішим вибором для розробки нейронних мереж в екосистемі TensorFlow.

Бібліотека дозволяє швидко створювати та тренувати складні нейронні мережі з мінімальною кількістю коду, що робить її доцільним вибором для початківців та досвідчених розробників, що працюють у галузі машинного навчання та глибокого навчання з використанням мови програмування Python.

Серед особливостей бібліотеки Keras слід зазначити модульність, підтримку багатьох бекендів, зручність використання, портативність, підтримку різних типів моделей.

Keras пропонує модульний підхід, що дозволяє легко поєднувати різні шари та моделі, створюючи складні архітектури нейронних мереж. Це полегшує експериментування з різними архітектурами та налаштування моделей під конкретні завдання.

Бібліотека є високорівневим інтерфейсом, яка може працювати з різними бекендами глибокого навчання, такими як TensorFlow, Theano і CNTK. Це дозволяє легко перемикатися між різними бекендами і вибирати найбільш підходящий для конкретного завдання.

Keras надає простий та інтуїтивно зрозумілий API, який спрощує процес створення, тренування та оцінки моделей машинного навчання. Наявні багато вбудованих функцій для автоматичної обробки даних, функції активації та втрат, оптимізатори та багато іншого. Бібліотека має активну спільноту розробників, що забезпечує наявність великої документації, посібників, прикладів коду та ресурсів підтримки.

Портативність моделей Keras полягає у можливості експорту в різні формати, такі як TensorFlow SavedModel, TensorFlow Lite, ONNX та інші, що полегшує інтеграцію моделей Keras в різні програми та платформи.

Keras підтримує різні типи моделей, такі як послідовні моделі, функціональні моделі та моделі з кількома входами/виходами. Це дозволяє створювати різноманітні архітектури нейронних мереж, включаючи нейронні згорткові мережі (Convolutional Neural Networks, CNN), рекурентні нейронні мережі (Recurrent Neural Networks, RNN), генеративні моделі (Generative Models).

3.1.1 Клас Sequential

Sequential – це клас який реалізує лінійну структуру у моделі глибокого навчання, що складається з одного шару, який йде слід за іншим. Кожен шар моделі Sequential має один вхід і один вихід, і інформація передається від одного шару до іншого послідовно.

Клас Sequential надає методи для створення, конфігурації, навчання та використання моделей глибокого навчання: add(), compile(), fit(), predict(), evaluate(), summary(), save() та load().

Метод add() дозволяє додавати шари моделі Sequential. Шари додаються послідовно, і порядок їх додавання визначає порядок обробки даних моделі. Приймає як аргумент об'єкт шару, який буде додано до моделі.

Метод compile() використовується для конфігурування моделі з функцією втрат (loss), оптимізатором (optimizer) та метриками (metrics) для навчання. Приймає як аргументи оптимізатор, функцію втрат та список метрик, які будуть використовуватися для оцінки продуктивності моделі.

Метод fit() використовується для навчання моделі даних. Приймає як аргументи вхідні дані (x) та цільові значення (y), а також інші параметри, такі як кількість епох (epochs), розмір пакету (batch_size) та частку даних для валідації (validation_split).

Метод predict() використовується для отримання прогнозованих нових даних. Приймає як аргумент вхідні дані (x) і повертає прогнозовані значення.

Метод summary() виводить інформацію про структуру моделі: кількість параметрів, розмірність вхідних та вихідних даних та кількість шарів.

Метод save() використовується збереження моделі на диск. Приймає як аргумент шлях до файлу, в який буде збережено модель.Метод load() використовується для завантаження збереженої моделі з диска. Приймає як аргумент шлях до файлу, з якого буде завантажена модель, і повертає об'єкт моделі.

Метод get_layer() дозволяє отримати посилання на шар моделі на певне ім'я. Призначений для отримання доступу до внутрішніх станів шару або здійснення інших операцій з ним.

Метод evaluate() використовується з метою оцінки продуктивності моделі на тестових даних. Приймає як аргументи вхідні дані (x) та цільові значення (y) та повертає значення функції втрат та метрики, зазначені при компіляції моделі.

3.1.2 Клас Dense

Клас Dense призначений для реалізації базових шарів для створення нейронних мереж. Базові шари являють собою повнозв'язковий шар (також відомий як шар з повною сполукою), де кожен нейрон вхідного шару з'єднаний з усіма нейронами вихідного шару.

Об’єкти класу Dense мають такі параметри:

- units – ціле число, обов'язковий параметр, що вказує на кількість нейронів у шарі. Наприклад, при units=64 буде створений шар із 64 нейронами;

- activation – функція активації, яка застосовуватиметься до виходів нейронів. Може бути однією з визначених функцій активації, таких як relu, sigmoid, tanh, softmax або функція активації. За замовчуванням None;

- use_bias – бульове значення, що вказує, чи використовуватиметься зміщення (bias) у шарі. Якщо True, то зміщення використовуватиметься, якщо False, то ні. За замовчуванням True;

- kernel_initializer – ініціалізація ваги (kernel) шару. Приймає такі значення, як glorot_uniform, glorot_normal, he_uniform, he_normal або користувацька власна реалізація ініціалізації ваг;

- bias_initializer – ініціалізація зміщення (bias) шару. Приймає значення zeros, ones або дозволяє використовувати користувацьку реалізацію зміщення шару. За замовчуванням використовується ініціалізація zeros;

- kernel_regularizer – регулювання ваги (kernel) зв’язків між нейронами шару. За замовчуванням дорівнює None;

- bias_regularizer – регулювання значень зміщення (bias) шару. За замовчуванням дорівнює None;

- activity_regularizer – регулювання виходу шару. За замовчуванням дорівнює None;

- kernel_constraint – обмеження ваги (kernel) шару. За замовчуванням дорівнює None;

- bias_constraint – обмеження на значення зміщення (bias) шару. За замовчуванням дорівнює None;

- input_dim – ціле число, що вказує розмірність вхідних даних. Використовується замість аргументу input_shape і задає розмірність вхідного тензора. Якщо не вказано, розмірність вхідних даних визначається автоматично на основі розмірності вхідних даних на наступному шарі;

- input_shape – кортеж або список цілих чисел, що вказує на розмірність вхідних даних, наприклад (batch_size, input_dim). Використовується лише для першого шару моделі. Якщо не вказано, розмірність вхідних даних визначається автоматично на основі розмірності вхідних даних на наступному шарі.

Параметри units і activation – це обов'язкові параметри для класу Dense, інші параметри опціональні і мають значення за замовчуванням.

3.1.3 Клас LSTM

Клас LSTM (Long Short-Term Memory) є реалізацією моделі LSTM, яка є одним із типів рекурентних нейронних мереж (RNNs). LSTM є модифікованим видом RNN, який здатний більш ефективно обробляти послідовні дані, зберігаючи інформацію про тривалі залежності в послідовності даних (прикладом таких даних є часові ряди).

Об’єкти класу LSTM мають такі параметри:

- units – ціле число, що вказує кількість блоків пам'яті LSTM (або кількість нейронів) у шарі;

- activation – функція активації, яка використовується для активації блоків пам'яті. Початково встановлено значення tanh, однак можна вказати іншу функцію активації, таку як sigmoid або relu;

- recurrent_activation – функція активації, що застосовується до рекурентних активацій. За замовчуванням встановлено значення sigmoid, але можна вказати іншу функцію активації;

- return_sequences – логічне значення, що вказує, чи LSTM шар повинен повертати послідовність на виході. Якщо значення дорівнює True, то LSTM шар повертатиме послідовність на виході, інакше – тільки останній вихідний елемент. За замовчуванням встановлено значення False;

- return_state – логічне значення, що вказує, чи LSTM шар повинен повертати стан на виході. Якщо значення дорівнює True, то LSTM шар повертатиме стан на виході, інакше – тільки вихідні дані. За замовчуванням встановлено значення False;

- dropout – частка вхідних нейронів, які будуть виключені випадково на кожній ітерації навчання для боротьби з перенавчанням. Приймає значення між 0 (за замовчуванням, dropout не використовується) та 1;

- recurrent_dropout – частка рекурентних нейронів, які будуть виключені для боротьби з перенавчанням.

Параметр timesteps відноситься до кількості часових кроків у вхідних даних, input_dim відноситься до розмірності вхідних даних на кожному часовому кроці, і output_dim відноситься до кількості класів вихідних даних (якщо використовується для класифікації задачі).

3.1.4 Клас Conv1D

Клас Conv1D – це реалізація одновимірної згорткової нейронної мережі (CNN) яка є високорівневим API для глибокого навчання.

Об’єкти класу Conv1D описуються такими параметрами:

- filters – кількість фільтрів (або ядер) згортки. Ціле число, що визначає кількість паралельних фільтрів, які будуть використані на вхідних даних. Чим більше фільтрів, тим більше різних ознак може бути вилучено;

- kernel_size – розмір вікна згортки. Ціле число чи кортеж цілих чисел, що визначають розмір вікна згортки. Для одновимірної згортки розмір вікна визначає кількість послідовних елементів у вхідних даних, на яких застосовується згортка;

- strides – кроки (крок згортки) по горизонталі. Ціле число або кортеж цілих чисел, що визначають кроки або зсув вікна згортки при його застосуванні на вхідних даних. Значення за промовчанням – 1, що означає, що вікно згортки зміщуватиметься на одну позицію за одну ітерацію;

- padding – режим заповнення. Рядок, що вказує режим заповнення даних перед застосуванням згортки. Може приймати значення valid (заповнення не використовується, дані обрізаються), same (заповнення таким чином, щоб вихідні дані мали ту саму ширину, що і вхідні), або causal (заповнення тільки з одного боку відповідно до принципу причинності). Значення за промовчанням – valid;

- activation – функція активації. Рядок або функція, що визначає функцію активації, яка застосовуватиметься після операції згортки. Наприклад, relu, sigmoid, tanh або вказана користувачем функція активації;

- input_shape – форма вхідних даних. Кортеж цілих чисел, що визначають форму вхідних даних. В одновимірному випадку це буде одне ціле число, що визначає розмір вхідної послідовності даних.

Conv1D використовується для обробки одновимірних даних, таких як часові ряди, сигнали або послідовності.

3.1.5 Клас MaxPooling1D

Клас MaxPooling1D представляє шар максимального пулінгу для одновимірних даних. Він використовується в нейронних мережах для зниження розмірності даних та отримання найбільш важливих ознак з часових послідовностей, таких як звукові сигнали або часові ряди.

Об’єкти класу MaxPooling1D мають такі параметри:

- pool_size – ціле число чи кортеж цілих чисел. Визначає розмір вікна пулінгу. Якщо це ціле число, вікно пулінга матиме однаковий розмір у всіх вимірах. Якщо це кортеж із двох цілих чисел, то вони визначатимуть розмір вікна пулінга вздовж першого та другого вимірів відповідно;

- strides – ціле число або кортеж цілих чисел. Визначає крок (крок переміщення вікна пулінга) вздовж кожного виміру. Якщо це ціле число, то крок буде однаковим у всіх вимірах. При використанні кортежу із двох цілих чисел, вони визначатимуть крок уздовж першого та другого вимірів відповідно. За замовченням strides дорівнює None, що означає використання значення pool_size для визначення кроку. Якщо встановити значення strides більше 1, то розмірність вихідних даних буде зменшуватися;

- padding – рядок, приймає одне із значень valid (за замовчуванням) або same. Визначає режим заповнення даних перед застосуванням пулінгу;

- data_format – рядок із значенням channels_last (за замовчуванням) або channels_first. Визначає порядок вимірів у вхідних даних. channels_last означає, що вимірювання каналів (ознак) знаходиться в останньому вимірі (осі -1), а channels_first означає, що вимірювання каналів знаходиться в першому вимірі (осі 1). Цей параметр важливий лише тоді, коли робота відбувається з багатоканальними даними.

Параметри дозволяють налаштувати операцію максимального пулінгу в одновимірних даних, що може бути корисним, наприклад, при обробці часових послідовностей у завданнях аналізу звуку, тексту та часових рядів.

3.1.6 Клас Flatten

Клас Flatten реалізує шар для згладжування (перетворення на одномірний вектор) багатовимірних тензорів. Цей шар зазвичай використовується в моделях нейронних мереж після згорткових шарів, щоб перетворити вихідні дані з згорткових шарів в одновимірний вектор, який потім може бути подано на повнозв'язкові шари для класифікації або регресії.

Клас Flatten у бібліотеці Keras має один параметр – data_format. Опціональний параметр, який вказує на формат вхідних даних. Він може набувати одного з двох значень: channels_last (за замовчуванням) або channels_first. channels_last означає, що вхідний тензор має формат

(batch_size, height, width, channels), а channels_first означає, що вхідний тензор має формат (batch_size, channels, height, width). Значення цього параметра впливає на те, як шар перетворюватиме вхідні дані на одномірний вектор. Якщо не вказано, значення за замовчуванням – channels_last.

Шар Flatten полегшує багатовимірні тензори в одновимірні, що дозволяє їх ефективно використовувати в повнозв'язкових шарах для подальшої обробки в нейронній мережі.

3.2 Алгоритм прогнозування

Для короткострокового прогнозування часових рядів обрано дані індексу забруднення повітря, зібрані організацією SaveDnipro. Дані отримуються за допомогою API та завантажуються до розробленого програмного забезпечення у ручному режимі і потім використовуються як для навчання моделей так і для прогнозу майбутніх значень. Для навчання моделей дані проходять підготовку, розбиваються на окремі вибірки вхідних та вихідних значень, залежно від величини горизонту прогнозування.

Після завантаження даних необхідно вказати горизонт прогнозування та параметри усіх моделей ШНМ, такі як кількість шарів, кількість нейронів у шарі, функцію активації, оптимізатор, функцію втрат та кількість епох.

Для підготовки вхідних даних для прогнозування використовується метод ковзного вікна. Метод ковзного вікна – це техніка обробки часових рядів та інших послідовностей даних, яка полягає в тому, щоб розбити вихідну послідовність на фрагменти (вікна) фіксованого розміру, що не перекриваються, і застосувати до кожного вікна певну функцію.

Для прогнозування індексу забруднення повітря методом ковзного вікна необхідно вибрати його розмір, а потім перейти по часовому ряду з кроком в одне значення, щоразу обчислюючи середнє значення індексу забруднення повітря у поточному вікні. Таким чином, виходить нова послідовність значень, яка буде ковзним середнім значенням (рисунок 3.1).

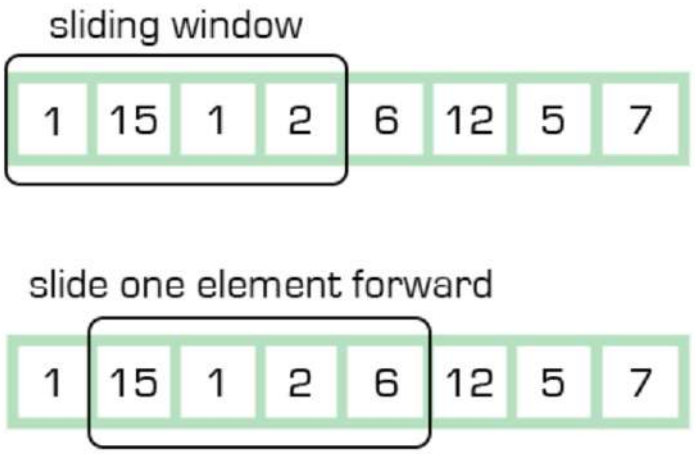

Рисунок 3.1 – Метод ковзного вікна

Дані для навчання та отримання прогнозу надходять до моделей багатошарового персептрону, довгої короткострокової пам’яті та згорткової нейронної мережі одночасно, що дозволяє отримати результату обчислювальних експериментів для усіх реалізованих моделей прогнозування одразу.

3.3 Підготовка даних

Для навчання штучної нейронної мережі вхідну послідовність даних необхідно розділити на кілька шаблонів введення та виведення для створення навчальної вибірки (лістинг 3.1).

Лістинг 3.1 – Підготовка даних

```
row_sequence, row_sequence_result = list(), list()
for i in range(len(sequence)):
    row_sequence_len = i + steps
    if row_sequence_len > len(sequence) - 1:
        break
    row_sequence_list, row_sequence_result_list =
sequence[i:row_sequence_len], sequence[row_sequence_len]
    row_sequence.append(row_sequence_list)
    row_sequence_result.append(row_sequence_result_list)
array(row_sequence), array(row_sequence_result)
```

Перші значення часового ряду використовуються як вхідні дані, наступні значення або кілька значень, залежно від горизонту прогнозування, використовуються як вихідні дані для прогнозування.

3.4 Багатошаровий персептрон

Для реалізації моделі багатошарового персептрону використовується бібліотека Keras в Python. Кінцева модель багатошарового персептрону складається з трьох шарів: двох прихованих шарів та одного вихідного шару. Модель створюється за допомогою класу Sequential. Цей клас дозволяє створювати послідовні нейронні мережі, де шари послідовно додаються один за одним (лістинг 3.2).

Лістинг 3.2 – Модель багатошарового персептрону

```
model = Sequential()
model.add(Dense(units=128, activation='relu', input_dim=64))
model.add(Dense(units=64, activation='relu'))
model.add(Dense(units=10, activation='softmax'))
model.compile(
 optimizer='adam',
 loss='mse',
 metrics=['accuracy']
)
model.summary()
model.fit(X_train, y_train, epochs=10, batch_size=32)
y_pred = model.predict(X_pred)
predicted_classes = y_pred.argmax(axis=-1)
print('Predicted classes:', predicted_classes)
```

Перший прихований шар створюється за допомогою функції model.add(), яка додає шари до моделі. Для цього використовується клас Dense (повнозв'язковий шар), який є шаром, в якому кожен нейрон пов'язаний з усіма нейронами попереднього шару. Параметри, що передаються Dense(), визначають кількість нейронів (128), функцію активації (ReLU) і розмірність вхідних даних (input_dim=64).

Другий прихований шар додається аналогічним чином з використанням model.add(Dense()), але вже з 64 нейронами та функцією активації ReLU.

Вихідний шар також додається за допомогою model.add(Dense()), але з іншою функцією активації – softmax.

Після додавання всіх шарів, модель компілюється за допомогою compile(), де вказуються оптимізатор, функція втрат і метрики для навчання та оцінки моделі.

У даному випадку оптимізатор – adam, який є поширеним вибором для навчання нейронних мереж. Функція втрат MSE використовується для вимірювання різниці між прогнозом та фактичними значеннями. А метрика accuracy обрана для оцінки якості моделі, вона показує частку прогнозних значень з найменшим відхиленням від реальних значень часового ряду.

Метод summary() виводить докладну інформацію про модель, включаючи кількість параметрів (ваг і зсувів), розмірність вхідних і вихідних даних кожного шару, та загальна кількість параметрів моделі, які корегуються у процесі навчання. Ця інформація корисна для аналізу структури моделі та оцінки її складності.

Після компіляції моделі відбувається навчання на тренувальних даних. Для цього використовується метод fit(), який виконує навчання моделі. Як вхідні дані передаються тренувальні дані (X_train) і мітки (y_train), а також вказується кількість епох навчання (epochs) та розмір пакету (batch_size), який визначає, скільки прикладів буде використовуватися для оновлення ваг моделі на кожному кроці.

Після навчання моделі її можна використовувати для прогнозу нових даних з використанням методу predict(). Передбачається, що вхідні дані для прогнозу зберігаються в змінній X_pred, і метод predict() повертає прогнозовані значення часового ряду. Можна використовувати функцію argmax() з бібліотеки NumPy для отримання індексу з найбільшою ймовірністю та визначення передбаченого значення.

3.5 Довга короткострокова пам'ять

Для реалізації моделі довгої короткострокової пам'яті використовується бібліотека Keras в Python, модель створюється за допомогою класу Sequential (лістинг 3.3).

Лістинг 3.3 – Модель довгої короткострокової пам'яті

```
model = Sequential()
model.add(LSTM(128, input_shape=(timesteps, features)))
model.add(Dense(num_classes, activation='softmax'))
model.compile(
 optimizer='adam', loss='mse', metrics=['accuracy']
)
model.summary()
model.fit(X_train, y_train, epochs=10, batch_size=32)
y_pred = model.predict(X_pred)
predicted_classes = y_pred.argmax(axis=-1)
print('Predicted classes:', predicted_classes)
```

Після створення моделі додається шар LSTM за допомогою функції model.add(LSTM()), вказуючи кількість нейронів (128), функцію активації (relu) та форму вхідних даних (кількість часових кроків та розмірність вхідних даних).

Після цього додається повнозв'язний шар за допомогою функції model.add(Dense()), вказуючи кількість вихідних нейронів (num_classes) та функцію активації (softmax).

Після створення моделі відбувається її компіляція за допомогою методу compile(), вказавши оптимізатор (adam), функцію втрат (mse) та метрику (accuracy) для оцінки продуктивності моделі під час навчання.

Після компіляції моделі відбувається її навчання, передаючи дані навчання в модель і вказуючи кількість епох і розмір пакету, де X_train – вхідні дані для навчання моделі, y_train – мітки для навчання моделі, epochs – кількість епох (повних проходів по навчальному набору даних), batch_size – розмір пакету даних, що передаються в модель за одну ітерацію навчання.

Після навчання моделі можна використовувати її для прогнозування, передаючи нові дані через функцію model.predict(), де X_pred – вхідні дані для прогнозування.

3.6 Згорткова нейронна мережа

Для реалізації моделі згорткової нейронної мережі використовується бібліотека Keras в Python (лістинг 3.4).

Лістинг 3.4 – Модель згортокової неронної мережі

```
model = Sequential()
model.add(
 Conv2D(
  32,
  (3, 3),
  activation='relu',
  input_shape=(64, 64, 3)
 )
)
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
model.compile(optimizer='adam', loss='mse',
metrics=['accuracy'])
model.summary()
model.fit(X_train, y_train, epochs=10, batch_size=32)
y_pred = model.predict(X_pred)
predicted_classes = y_pred.argmax(axis=-1)
print('Predicted classes:', predicted_classes)
```

Спочатку створюється об'єкт моделі Sequential, в якій шари додаються послідовно один за одним, модель з декількома шарами згортки, пулінгу і повнозв'язкових шарів.

За допомогою методу add() додаються такі шари моделі ШНМ. Згортковий шар (Conv2D) з 32 фільтрами, розміром вікна 3x3 та функцією активації (relu). Вхідний розмір даних вказується в аргументі input_shape,

який визначає розмірність вхідного тензора у форматі (висота, ширина, канали). Шар пулінгу (MaxPooling2D) із вікном 2x2 для зменшення розмірності карти ознак. Ще один згортковий шар з 64 фільтрами і вікном 3x3, за яким слідує шар пулінга з таким же вікном. Шар Flatten, який перетворює карту ознак на одномірний вектор. Повнозв'язковий шар Dense із 128 нейронами та функцією активації ReLU. Вихідний повнозв'язний шар з 10 нейронами та функцією активації Softmax.

За допомогою методу compile() задаються параметри компіляції моделі, такі як оптимізатор (optimizer), функція втрат (loss) і метрики (metrics). Використовується оптимізатор adam, функція втрат mse та метрика точності accuracy для оцінки продуктивності моделі.

Після компіляції модель навчається на наборі даних за допомогою методу fit(), передаючи вхідні дані та відповідні позначки класів. Модель буде навчатися цих даних, оптимізуючи параметри моделі для мінімізації функції втрат.

Після навчання моделі вона може бути використана для застосування на нових даних. Для цього використовується метод predict(), передаючи вхідні дані в модель та отримуючи результати прогнозу часового ряду для кожного вхідного прикладу.

Додатково можна зберегти навчену модель на диск за допомогою методу save(), щоб мати можливість завантажити її в майбутньому та використовувати.

4 АНАЛІЗ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

4.1 Досліджувані параметри моделей

У ході дослідження були створені моделі нейронних мереж такі як багатошаровий персептрон, довга короткострокова пам’ять та згорткові нейронні мережі із застосуванням різних параметрів.

Результати дослідження наведено у таблиці 1 додатка Б, з такими даними як row, predict_row, horizont, epoch, units, filters, kernel_size, pool_size MAPE та MAE.

Параметр row визначає розмір вибірки даних часового ряду, що використовується для навчання моделей штучних нейронних мереж. Чим більше значення цього параметру, тим складніше відбувається навчання та менший вплив ефекту перенавчання, що негативно впливає на результати прогнозування.

Параметр predict_row – розмір вибірки даних, що використовується для виконання прогнозування часового ряду.

Параметр horizont – розмір горизонту прогнозування, який визначає на скільки значень наперед відбуватиметесь прогнозування

Параметр epoch – кількість ітерацій навчання. Чим більше значення цього параметру, тим вище точність але велике значення параметру epoch може викликати ефект перенавчання моделі.

Параметр units – кількість нейронів у шарі. Чим більше значення нейронів у шарі, тим більший об’єм пам’яті у нейронної мережі.

Параметр filters (тільки для CNN) – кількість згорток на виході шару. Чим більше значення цього параметру, тим більше різних ознак може бути вилучено.

Параметр pool_size (тільки для CNN) – розмір вікна пулінгу. Може бути числом або кортежом чисел. Якщо ціле число, то вікно пулінгу має один

розмір у всіх вимірах. Якщо це кортеж із чисел, то вікно пулінгу визначається значенням цих чисел відповідно до вимірів.

Параметр kernel_size (тільки для CNN) – розмір згорткового ядра.

MAE (Mean Absolute Error) – середнє абсолютне відхилення між прогнозованими значеннями та фактичними значеннями цільової змінної. MAE обчислюється шляхом підсумовування абсолютних значень різниць між прогнозами моделі та фактичними значеннями, а потім поділу цієї суми на кількість спостережень.

MAPE (Mean Absolute Percentage Error) – відсоткове відхилення між прогнозними значеннями та фактичними значеннями на основі їх абсолютних значень. MAPE обчислюється шляхом знаходження абсолютних значень відсоткових різниць між прогнозами моделі та фактичними значеннями, а потім поділу цієї суми на кількість спостережень.

4.2 Базові конфігурації моделей

Для моделей штучних нейронних мереж, в якості базових конфігурацій визначені наступні значення гіперпараметрів.

Багатошаровий персептрон (MLP): units – 64, activation – relu, optimizer – adam, loss – mse.

Довга корокострокова пам’ять (LSTM): units – 64, activation – relu, optimizer – adam, loss – mse.

Згорткова нейронна мережа (CNN): fulters – 64, kernel_size – 2, pool_size – 2, activation – relu, optimizer – adam, loss – mse.

Глобальні гіперпараметри: epoch – 1000, horizont – 8.

Функція активації ReLU проста у обчисленні та має переваги, такі як простота та ефективність, здатність усувати проблему згасання градієнта, що може виникати при навчанні глибоких нейронних мереж, здатність моделювати нелінійні залежності та створювати розріджені активації, тобто такі, в яких велика кількість нейронів неактивні (мають нульові значення).

Функція активації залишає всі позитивні значення без змін та присвоює нуль негативним значенням.

Оптимізатор Adam – це метод стохастичної оптимізації, який оновлює ваги моделі, враховуючи не лише поточний градієнт, а й історію попередніх градієнтів, щоб пришвидшити сходимість до оптимального рішення.

Adam автоматично адаптує швидкість навчання (learning rate) для кожного параметра на основі градієнтів, що спостерігаються, що дозволяє більш ефективно навчати модель з різними типами даних і архітектурою нейронної мережі.

Оптимізатор Adam має такі переваги як простота використання, адаптивна швидкість навчання, врахування історії градієнтів, робота з великими наборами даних.

Функція втрат MSE використовується для визначення, як модель оцінює помилку і оновлює свої ваги у процесі навчання. MSE вимірює середнє значення квадратів різниць між прогнозами моделі та фактичними значеннями цільової змінної.

MSE обчислюється шляхом знаходження квадратів різниць між прогнозами моделі та фактичними значеннями, а потім поділу цієї суми на кількість спостережень.

Мінімізація функції втрат MSE у процесі навчання моделі допомагає моделі знаходити оптимальні значення параметрів, щоб досягти відповідності між прогнозами та справжніми значеннями цільових змінних.

4.3 Залежність результатів від параметрів прогнозування

Для дослідження впливу параметрів розміру вибірки даних для навчання (row), розміру вибірки даних для прогнозування (predict_row) та горизонту прогнозування (horizont), використовувались моделі штучних нейронних мереж у базовій конфігурації.

Таблиця 4.1 – Залежність результатів від розміру вибірки даних для навчання моделі багатошарового персептрону

№	Параметри			Помилка	
	row	predict_row	horizont	MAE	MAPE
1	16	8	8	0,07	2
2	32	8	8	0,28	6
3	64	8	8	0,69	7
4	128	8	8	0,3	14
5	256	8	8	1,6	21
6	512	8	8	1,37	19
7	1024	8	8	1,32	15
8	2048	8	8	1,32	26

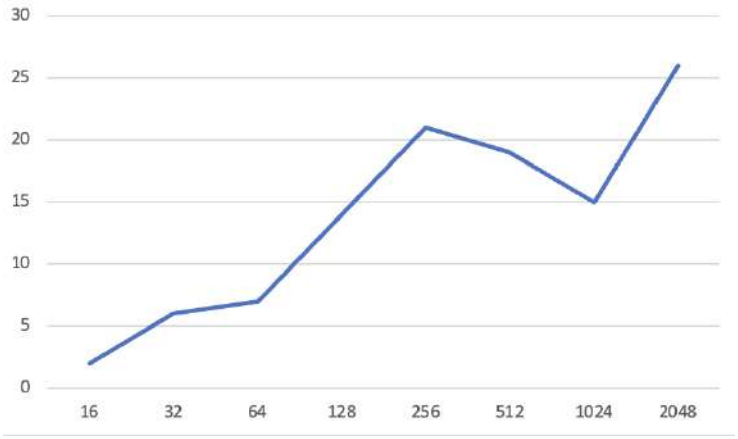

Рисунок 4.1 – MAPE в залежності від розміру вибірки даних для навчання багатошарового персептрону

При збільшені розміру вибірки даних для навчання багатошарового персептрону помилка при прогнозуванні часових рядів, з використанням моделі багатошарового персептрону, збільшується (таблиця 4.1). Найкращі результати модель MLP показала при значенні 16, найгірші при значенні 2048 (рисунок 4.1).

Таблиця 4.2 – Залежність результатів від розміру вибірки даних для навчання моделі довгої короткострокової пам’яті

№	Параметри			Помилка	
	row	predict_row	horizont	MAE	MAPE
1	16	8	8	0,03	1
2	32	8	8	0,34	7
3	64	8	8	0,17	2
4	128	8	8	0,1	3
5	256	8	8	0,14	2
6	512	8	8	0,31	4
7	1024	8	8	0,12	2
8	2048	8	8	0,24	4

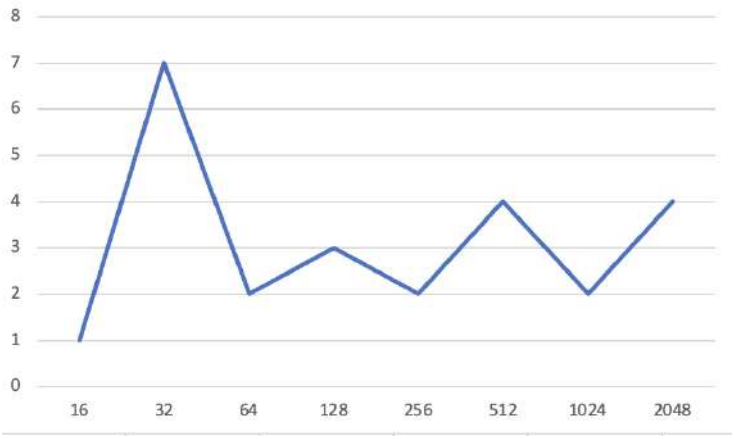

Рисунок 4.2 – MAPE в залежності від розміру вибірки даних для навчання моделі довгої короткострокової пам’яті

Модель довгої короткострокової пам’яті, при збільшені розміру вибірки даних для навчання показує результати майже на одному рівні, помилка коливає від 2 до 4 відсотків (таблиця 4.2). Найкращі результати модель довгої короткострокової пам’яті показала при значенні 16, найгірші при значеннях 512 та 2028 (рисунок 4.2).

Таблиця 4.3 – Залежність результатів від розміру вибірки даних для навчання моделі згорткової нейронної мережі

№	Параметри			Помилка	
	row	predict_row	horizont	MAE	MAPE
1	16	8	8	0,05	1
2	32	8	8	0,26	5
3	64	8	8	0,57	6
4	128	8	8	0,38	11
5	256	8	8	1	13
6	512	8	8	1,24	17
7	1024	8	8	1,32	15
8	2048	8	8	1,4	30

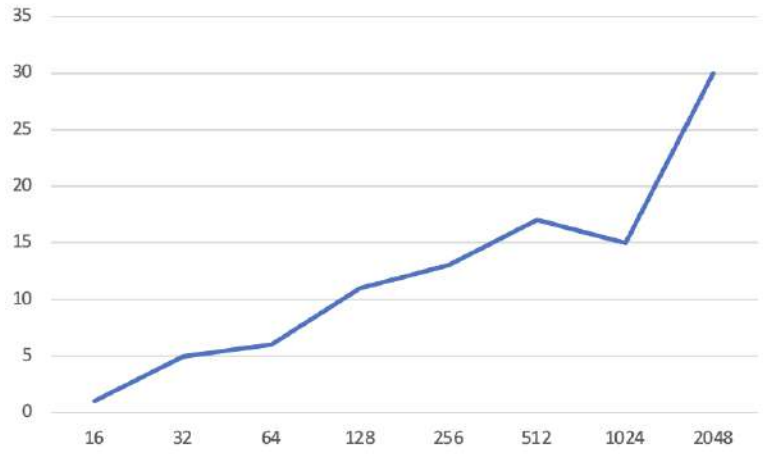

Рисунок 4.3 – MAPE в залежності від розміру вибірки даних для навчання згорткової нейронної мережі

Помилка при прогнозуванні часових рядів з використанням моделі згорткової нейронної мережі, при збільшені розміру вибірки даних для навчання, зростає (таблиця 4.3). Найкращі результати модель CNN показала при значенні 16, найгірші при значенні 2028 (рисунок 4.3).

Таблиця 4.4 – Залежність результатів від розміру вибірки даних для прогнозування моделі багатошарового персептрону

№	Параметри			Помилка	
	row	predict_row	horizont	MAE	MAPE
1	64	4	8	1,12	11
2	64	8	8	0,86	8
3	64	12	8	0,67	7
4	64	16	8	0,13	1
5	64	20	8	0,08	1
6	64	24	8	0,05	1
7	64	28	8	0,11	1
8	64	32	8	0,38	1

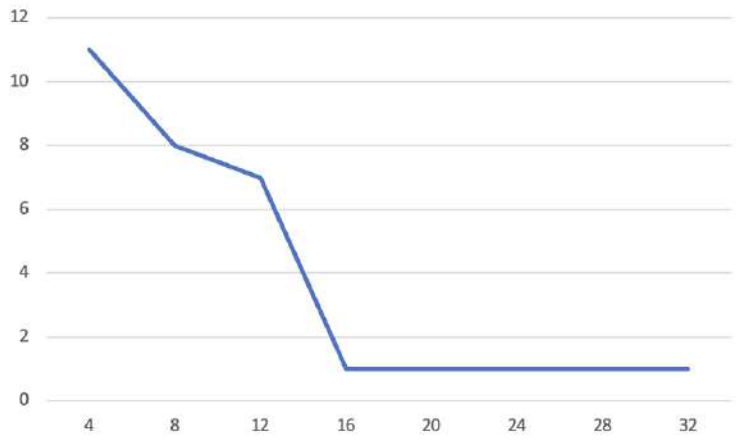

Рисунок 4.4 – Залежність MAPE від розміру вибірки для прогнозування при використанні MLP

При збільшені розміру вибірки даних для прогнозування помилка при прогнозуванні, з використанням моделі багатошарового персептрону, зменшується (таблиця 4.4). Найкращі результати прогнозу модель MLP показала при значеннях 16, 20, 24, 28 та 32, найгірші при значенні 4, помилка – 11 відсотків (рисунок 4.4).

Таблиця 4.5 – Залежність результатів від розміру вибірки даних для прогнозування моделі довгої короткострокової пам’яті

№	Параметри			Помилка	
	row	predict_row	horizont	MAE	MAPE
1	64	4	8	0,6	7
2	64	8	8	0,24	3
3	64	12	8	0,19	2
4	64	16	8	0,11	1
5	64	20	8	0,16	2
6	64	24	8	0,04	1
7	64	28	8	0,11	1
8	64	32	8	0,46	5

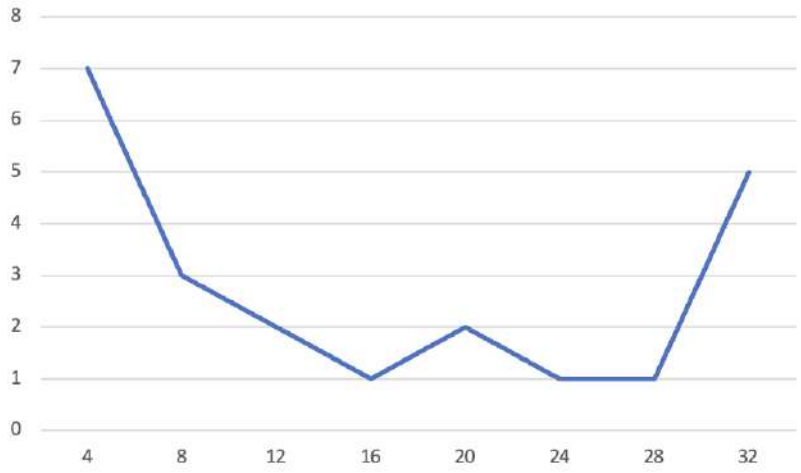

Рисунок 4.5 – Залежність MAPE від розміру вибірки для прогнозування при використанні LSTM

Модель довгої короткострокової пам’яті, при збільшені параметру розміру вибірки даних для прогнозування має помилку на рівні від 1 до 7 відсотків (таблиця 4.5). Найкращі результати прогнозу модель LSTM показала при значенні 16, 24 та 28, найгірші при значенні 4 (рисунок 4.5).

Таблиця 4.6 – Залежність результатів від розміру вибірки даних для прогнозування моделі згорткової нейронної мережі

№	Параметри			Помилка	
	row	predict_row	horizont	MAE	MAPE
1	64	4	8	1,58	19
2	64	8	8	1,01	11
3	64	12	8	0,15	2
4	64	16	8	0,1	1
5	64	20	8	0,04	1
6	64	24	8	0,04	1
7	64	28	8	0,05	1
8	64	32	8	0,04	1

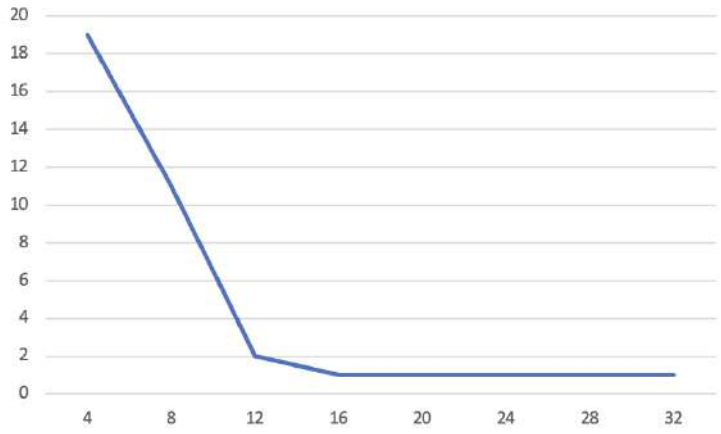

Рисунок 4.6 – Залежність MAPE від розміру вибірки для прогнозування з використанням CNN

Помилка при прогнозуванні з використанням моделі згорткової нейронної мережі при збільшені параметру розміру вибірки даних для прогнозування зменшується (таблиця 4.6). Найкращі результати прогнозу модель CNN показала при значеннях 16, 20, 24, 28 та 32, найгірші при значенні 4 (рисунок 4.6).

Таблиця 4.7 – Залежність результатів від горизонту прогнозування для моделі багатошарового персептрону

№	Параметри			Помилка	
	row	predict_row	horizont	MAE	MAPE
1	64	8	2	0,13	2
2	64	8	4	0,46	6
3	64	8	6	0,34	5
4	64	8	8	0,86	9
5	64	8	10	0,87	9
6	64	8	12	0,73	8
7	64	8	14	0,78	10
8	64	8	16	0,71	8

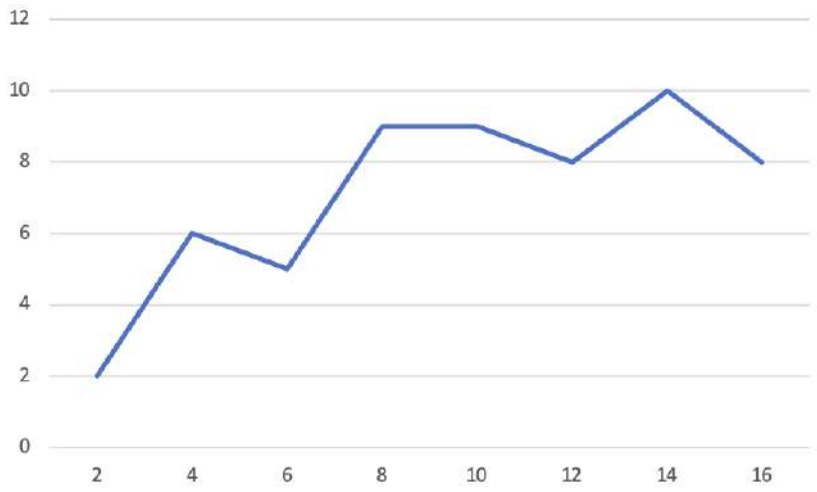

Рисунок 4.7 – Залежність MAPE від горизонту прогнозування при використанні MLP

При збільшені горизонту прогнозування помилка при отриманні результатів, з використанням моделі багатошарового персептрону, збільшується (таблиця 4.7). Найкращі отримані результати прогнозу модель MLP показала при значенні 2, найгірші при значенні 14 (рисунок 4.7).

Таблиця 4.8 – Залежність результатів від горизонту прогнозування для моделі довгої короткострокової пам’яті

№	Параметри			Помилка	
	row	predict_row	horizont	MAE	MAPE
1	64	8	2	0,08	1
2	64	8	4	0,09	1
3	64	8	6	0,23	3
4	64	8	8	0,23	3
5	64	8	10	0,21	3
6	64	8	12	0,22	3
7	64	8	14	0,28	4
8	64	8	16	0,23	3

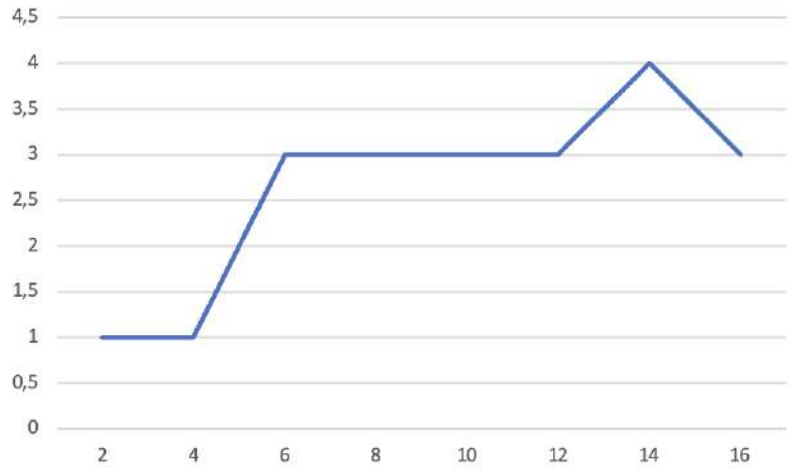

Рисунок 4.8 – Залежність MAPE від горизонту прогнозування при використанні LSTM

Модель довгої короткострокової пам’яті, при збільшені горизонту прогнозування, показує майже незмінні результати, помилка коливає від 1 до 4 відсотків (таблиця 4.8). Найкращі отримані результати прогнозу модель LSTM показала при значенні 2, найгірші при значенні 14. (рисунок 4.8).

Таблиця 4.9 – Залежність результатів від горизонту прогнозування для моделі згорткової нейронної мережі

№	Параметри			Помилка	
	row	predict_row	horizont	MAE	MAPE
1	64	8	2	0,19	3
2	64	8	4	0,54	7
3	64	8	6	0,79	11
4	64	8	8	1,17	14
5	64	8	10	0,83	11
6	64	8	12	0,99	11
7	64	8	14	0,64	9
8	64	8	16	0,57	7

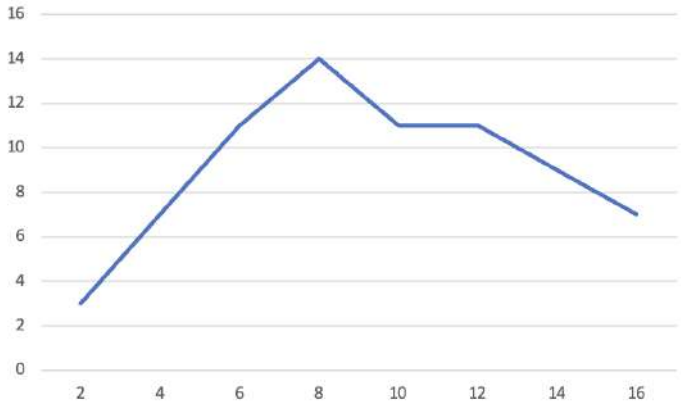

Рисунок 4.9 – Залежність MAPE від горизонту прогнозування при використанні CNN

Помилка при отриманні результатів прогнозування, з використанням моделі згорткової нейронної мережі, при збільшені горизонту прогнозування при значеннях від 2 до 8 зростає з 3 до 14 відсотків, а при значеннях від 8 до 16 зменшується з 14 до 7 відсотків (таблиця 4.9). Найкращі отримані результати прогнозу модель CNN показала при значенні – 2, найгірші при значенні – 8 (рисунок 4.9).

Середнє значення абсолютної помилки у відсотках багатошарового персептрону становить – 8%, довгої короткострокової пам’яті – 3%, згорткової нейронної мережі – 9%.

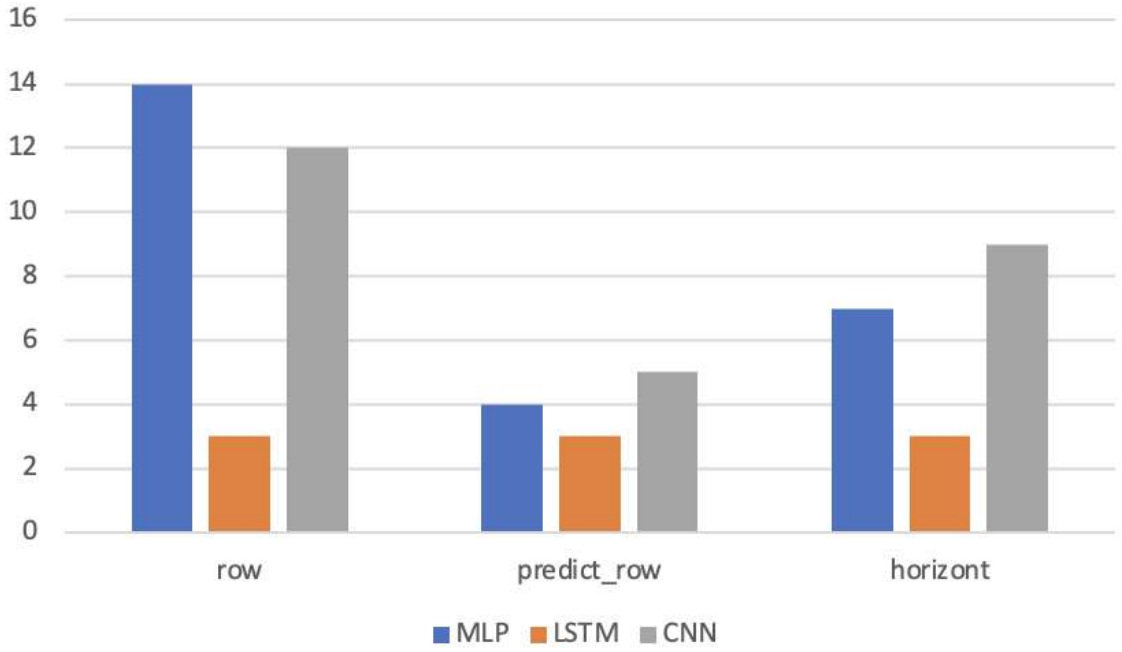

Рисунок 4.10 – Значення MAPE в залежності від параметрів прогнозування

Модель довгої короткострокової пам’яті показує більш високу стійкість при зміні параметрів прогнозування, таких як розмір вибірки даних для навчання (row), розмір вибірки даних для прогнозування (predict_row) та горизонт прогнозування (horizont) (рисунок 4.10).

4.4 Вплив гіперпараметрів на результати прогнозування

Для дослідження впливу гіперпараметрів (параметрів, що визначають структуру моделей) на результати прогнозування використовувалися часові ряди з розміром вибірки даних для навчання 64, розміром вибірки даних для прогнозування 8 та горизонтом прогнозування 8 значень.

Таблиця 4.10 – Вплив кількості нейронів у шарі на результати прогнозування для моделі багатошарового персептрону

№	Гіперпараметри		Помилка	
	units	epochs	MAE	MAPE
1	16	1000	1,18	13
2	32	1000	1,12	11
3	48	1000	1	10
4	64	1000	0,66	6
5	80	1000	0,59	6
6	96	1000	0,7	7
7	112	1000	0,41	4
8	128	1000	0,35	3

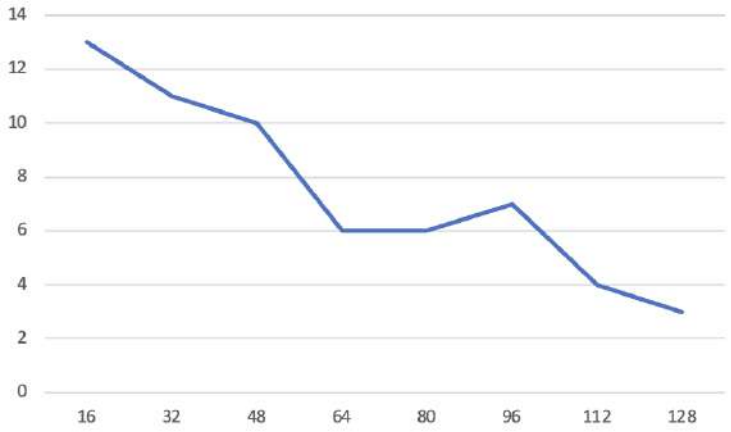

Рисунок 4.11 – Значення MAPE в залежності від кількості нейронів у моделі багатошарового персептрону

При збільшені кількості нейронів помилка при прогнозуванні з використанням моделі штучної нейронної мережі багатошарового персептрону зменшується (таблиця 4.10). Найкращі результати MLP показала при значенні 128, найгірші при значенні 16. При значеннях від 64 до 96 помилка залишалася на одному рівні 6-7 відсотків (рисунок 4.11).

Таблиця 4.11 – Вплив кількості нейронів у шарі на результати прогнозування для моделі довгої короткострокової пам’яті

№	Гіперпараметри		Помилка	
	units	Epochs	MAE	MAPE
1	16	1000	0,6	6
2	32	1000	0,17	2
3	48	1000	0,14	2
4	64	1000	0,24	3
5	80	1000	0,2	3
6	96	1000	0,14	2
7	112	1000	0,12	1
8	128	1000	0,36	5

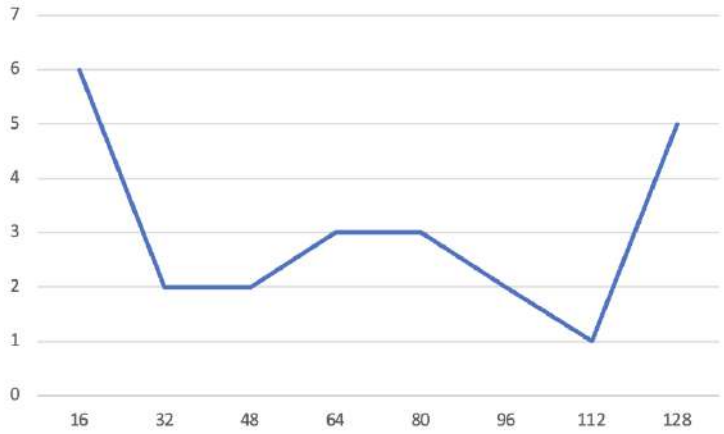

Рисунок 4.12 – Значення MAPE в залежності від кількості нейронів у шарі у моделі довгої короткострокової пам’яті

Модель штучної нейронної мережі на основі довгої короткострокової пам’яті при збільшені кількості нейронів має помилку на рівні від 1 до 6 відсотків (таблиця 4.11). Найкращі результати прогнозу штучна нейронна мережа моделі LSTM показала при 112 нейронах, найгірші при 16 нейронів у шарі (рисунок 4.12).

Таблиця 4.12 – Вплив кількості згорток на виході шару на результати прогнозування для згорткової нейронної мережі

№	Гіперпараметри		Помилка	
	filters	epochs	MAE	MAPE
1	16	1000	0,88	11
2	32	1000	0,81	9
3	48	1000	1,05	13
4	64	1000	0,7	8
5	80	1000	0,96	12
6	96	1000	0,94	11
7	112	1000	0,98	12
8	128	1000	0,96	11

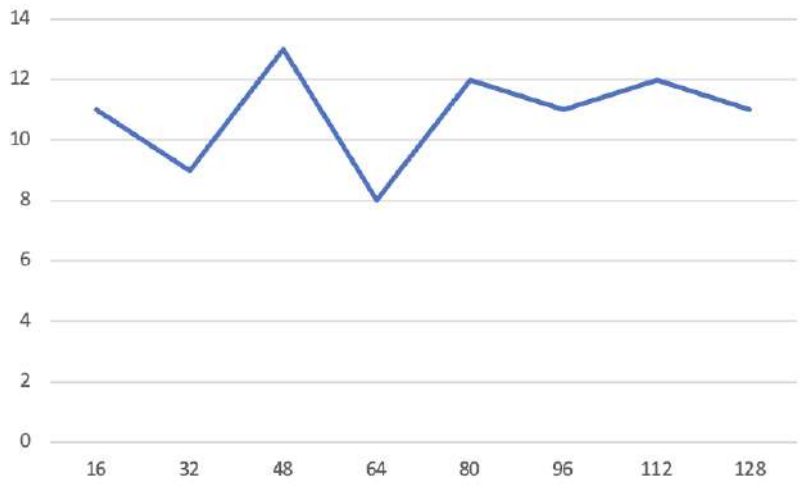

Рисунок 4.13 – Значення MAPE в залежності від кількості нейронів у шарі

Модель згорткової нейронної мережі, при збільшені гіперпараметру кількості згорток на виході шару, має помилку на рівні від 8 до 13 відсотків (таблиця 4.12). Найкращі результати прогнозу часового ряду модель CNN показала при 64 згортках на виході шару, помилка складає 8%. Найгірші результати при 48 згортках, помилка досягає 13% (рисунок 4.13).

Таблиця 4.13 – Вплив кількісті ітерацій навчання на помилку прогнозування для моделі багатошарового персептрону

№	Гіперпараметри		Помилка	
	units	epochs	MAE	MAPE
1	64	10	2,16	25
2	64	100	1,57	19
3	64	400	1,45	17
4	64	800	0,87	9
5	64	1000	0,59	6
6	64	1200	1,01	10
7	64	1600	0,4	5
8	64	2000	0,15	2

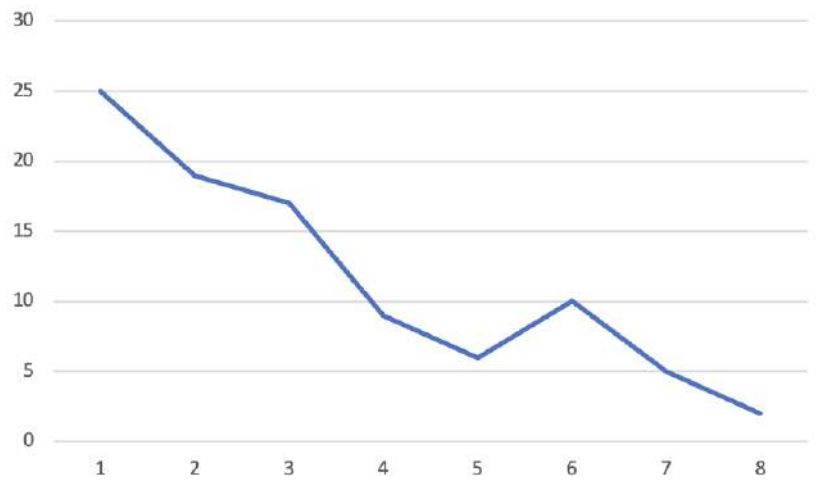

Рисунок 4.14 – Значення MAPE в залежності від кількісті ітерацій навчання для моделі багатошарового персептрону

При збільшені кількості ітерацій навчання помилка при прогнозуванні часових рядів, з використанням ШНМ моделі багатошарового персептрону, зменшується (таблиця 4.13). Найкращі результати, при збільшені кількості ітерацій, модель MLP показала при значенні 2000, найгірші при 16 ітераціях навчання (рисунок 4.14).

Таблиця 4.14 – Вплив кількісті ітерацій навчання на помилки прогнозування для моделі довгої короткострокової пам’яті

№	Гіперпараметри		Помилка	
	units	epochs	MAE	MAPE
1	64	10	2,02	21
2	64	100	1,5	17
3	64	400	0,83	9
4	64	800	0,4	5
5	64	1000	0,25	3
6	64	1200	0,22	3
7	64	1600	0,08	1
8	64	2000	0,08	1

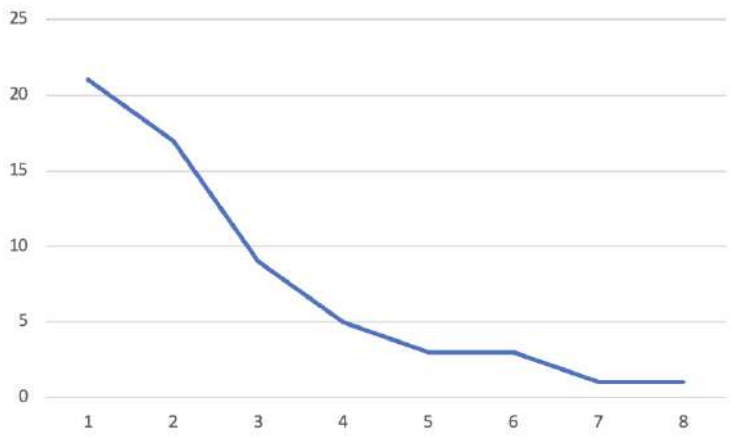

Рисунок 4.15 – Значення MAPE в залежності від ітерацій навчання для моделі довгої короткострокової пам’яті

Помилка при отриманні результатів прогнозування часових рядів з використанням моделі довгої короткострокової пам’яті, при збільшені кількості ітерацій навчання, зменшується (таблиця 4.14). Найкращі отримані результати прогнозу часових рядів модель LSTM показала при 1600-2000 ітераціях навчання, найгірші при 10 (рисунок 4.15).

Таблиця 4.15 – Вплив кількісті ітерацій навчання штучної нейронної мережі на результати прогнозування для моделі CNN

№	Гіперпараметри		Помилка	
	filters	epochs	MAE	MAPE
1	64	10	2,45	26
2	64	100	1,62	18
3	64	400	1,35	15
4	64	800	1,02	12
5	64	1000	1,07	13
6	64	1200	0,39	5
7	64	1600	0,58	9
8	64	2000	0,13	2

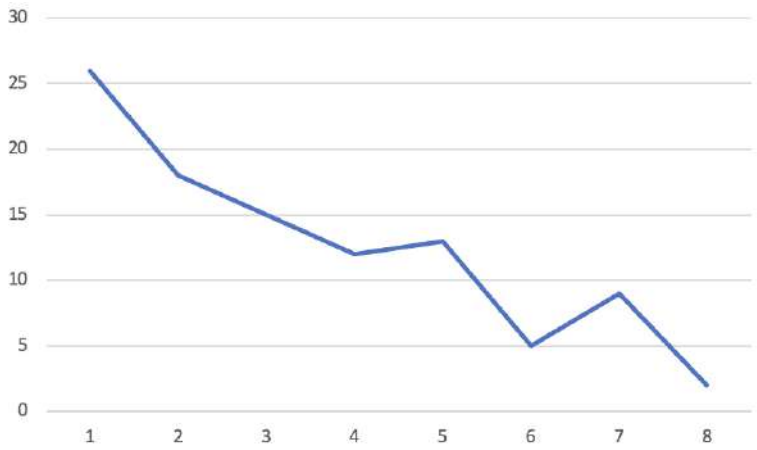

Рисунок 4.16 – Значення MAPE в залежності від кількісті ітерацій навчання для моделі згорткової нейронної мережі

При збільшені кількості ітерацій навчання помилка при отриманні результатів, з використанням ШНМ згорткової нейронної мережі, зменшується (таблиця 4.15). Найкращі результати прогнозування, при збільшені кількості ітерацій, модель CNN показала при значенні 2000, помилка дорівнює 2%, найгірші при 16 (помилка 26%) (рисунок 4.16).

Середнє значення абсолютної помилки у відсотках (MAPE) при прогнозуванні часових рядів для багатошарового персептрону становить 10%, довгої короткострокової пам’яті – 6%, згорткової нейронної мережі – 12%.

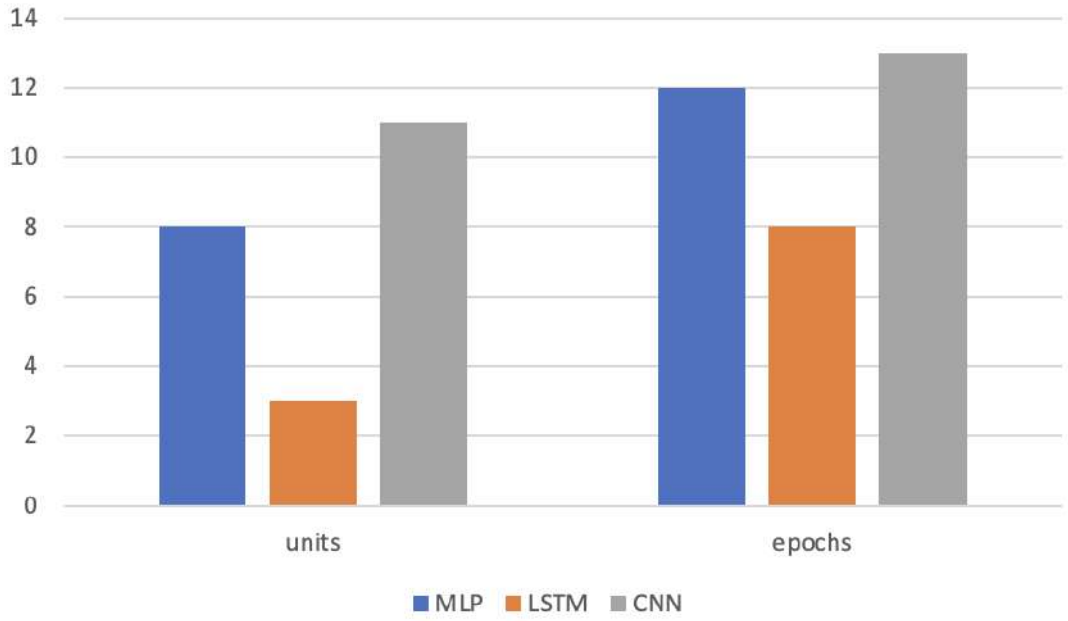

Рисунок 4.17 – Значення MAPE в залежності від гіперпараметрів

ШНМ моделі довгої короткострокової пам’яті показує кращі середні результати, ніж багатошаровий персептрон та згорткова нейронна мережа, при зміні таких гіперпараметрів як кількість блоків пам’яті моделі (units) та кількість ітерацій навчання (epochs) (рисунок 4.17).

Штучна нейронна мережа багатошарового персептрону більше підходить для прогнозування з великою кількістю нейронів у шарі та вимагає більшу кількість ітерацій навчання.

При використанні згорткової нейронної мережі для прогнозування часових рядів гіперпараметри майже не впливають на точність прогнозу, тому можна використовувати менше нейронів та епох навчання для підвищення швидкості та зменшення обсягу необхідної пам’яті.

ВИСНОВКИ

У роботі досліджено використання засобів обчислювального інтелекту для прогнозування часових рядів індексу забруднення повітря, що зібрані організацією SaveDnipro.

За результатами проведених досліджень найкращі результати показала модель довгої короткострокової пам’яті з середньою абсолютної помилкою у відсотках 4%, в той час як модель згорткової нейронної мережі – 10%, а багатошаровий персептрон – 9%. При зміні параметрів, таких як розмір вибірки для навчання, розмір вибірки для прогнозування та горизонт прогнозування, середнє значення абсолютної помилки у відсотках для LSTM становить 3%, MLP – 8%, CNN – 9%. При зміні гіперпараметрів таких як кількість нейронів у шарі та кількість ітерацій навчання, середнє значення абсолютної помилки у відсотках для довгої короткострокової пам’яті становить 6%, багаташорового персептрону – 10%, згорткової нейронної мережі – 12%.

Моделі багатошарового персептрону та довгої короткострокової пам’яті були досліджені при прогнозуванні часових рядів M3-Competition. MAPE досліджуваних моделей на 2-6% нижче при порівнянні з використанням традиційних моделей [36].

Результати роботи представлені на міжнародних науково-технічних конференціях «Проблеми інформатизації» [37] та «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління» [38].

Дослідженні моделі доцільно використовувати для короткострокового прогнозування часових рядів при правильному підборі параметрів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Palit, A. Computational Intelligence in Time Series Forecasting [Текст] / A. Palit, D. Popovic. – 2006. – 372 p.

2. Chen, S. Computational Intelligence in Economics and Finance [Текст] / S. Chen, P. Wang. – 2007. – 53 p.

3. Piccialli, F. Artificial intelligence and healthcare: Forecasting of medical bookings through multi-source time-series fusion [Текст] / F. Piccialli, F. Giampaolo, P. Edoardo // Information. – 2021. – Vol. 74. – P. 1–16.

4. Kousari, M. Introducing an operational method to forecast long-term regional drought based on the application of artificial intelligence capabilities [Текст] / M. Kousari, M. Hosseini, H. Ahani // Theoretical and Applied Climatology. – 2015. – Vol. 127. – P. 361–380.

5. Mostafa, M. A neuro-computational intelligence analysis of the ecological footprint of nations [Текст] / M. Mostafa, R. Nataraajan // Computational Statistics & Data Analysis. – 2009. – Vol. 53. – P. 3516–3531.

6. Afan, H. Past, present and prospect of an Artificial Intelligence (AI) based model for sediment transport prediction [Текст] / H. Afan, A. Shafie, W. Mohtar // Journal of Hydrology. – 2016. – Vol. 541. – P. 902–913.

7. Chatfield, C. Time-Series Forecasting [Текст] / C. Chatfield. – Chapman and Hall/CRC, 2000. – 280 p.

8. Li, S. Deterministic fuzzy time series model for forecasting enrollments [Текст] / S. Li, Y. Cheng // Computers & Mathematics with Applications. – 2007. – Vol. 53. – P. 1904–1920.

9. Cheng, C. Time series forecasting for nonlinear and non-stationary processes: a review and comparative study [Текст] / C. Cheng, A. Ngasoongsong, O. Beyca // IIE Transactions. – 2015. – Vol. 47. – P. 1053–1071.

10. Mohammed, A. S. Green Hybrid Models Based on Clonal Selection and Case-based Reasoning for Short-term Time Series Forecasting [Текст] /

A. S. Mohammed, H. Ivashchenko, T. Filimonchuk // Journal of Green Engineering. – 2020. – Vol. 10. – P. 2139–2154.

11. Derbentsev, V. Machine learning approaches for financial time series forecasting [Текст] / V. Derbentsev, A. Matviychuk, N. Datsenko // Proceedings of the Selected Papers of the Special Edition of International Conference on Monitoring, Modeling & Management of Emergent Economy. – 2020. – № 2713. – P. 434–450.

12. Korablev, N. M. Parallel immune algorithm of short-term forecasting based on model of clonal selection [Текст] / N. M. Korablev, G. S. Ivaschenko // Radio Electronics, Computer Science, Control. – 2014. – № 2. – P. 73–78.

13. Armstrong, J. S. Forecasting for Marketing [Текст] // Quantitative Methods in Marketing. – 1999. – P. 92–119.

14. Yang, J. Power System Short-term Load Forecasting: Thesis for Ph.d degree [Текст] // Elektrotechnik und Informationstechnik. – P. 139.

15. Berastegi, G. From diagnosis to prognosis for forecasting air pollution using neural networks: Air pollution monitoring in Bilbao [Текст] / G. Berastegi, A. Elias, A. Barona // Environmental Modelling & Software. – 2008. – Vol. 23. – P. 622–637.

16. Durao, R. Forecasting O3 levels in industrial area surroundings up to 24 h in advance, combining classification trees and MLP models [Текст] / R. Durao, M. Mendes, J. Pereira // Atmospheric Pollution Research. – 2016. – Vol. 7. – P. 961–970.

17. Namini, S. A Comparison of ARIMA and LSTM in Forecasting Time Series [Текст] / S. Namini, N. Tavakoli, A. Namin // IEEE International Conference on Machine Learning and Applications. – 2018. – Vol. 17.

18. Zeng, Z. Self CNN-based time series stream forecasting [Текст] / Z. Zeng, H. Xiao, X. Zhang // Electronics Letters. – 2016. – Vol. 52. – P. 1857–1858.

19. Ivaschenko, G. S. Time series forecasting on the basis of the case-based reasoning using the models of artificial immune systems [Текст] /

Heorhii S. Ivaschenko, N. M. Korablev // System technologies. – 2014. – № 6. – P. 43–51.

20. Cai, Q. A Novel Stock Forecasting Model based on Fuzzy Time Series and Genetic Algorithm [Текст] / Q. Cai, D. Zhang, B. Wu // Procedia Computer Science. – 2013. – Vol. 18. – P. 1155–1162.

21. Gurney, K. An Introduction to Neural Networks // K. Gurney., 1997. – 234 p.

22. Lachtermacher, G. Back propagation in time-series forecasting [Текст] / G. Lachtermacher, D. Fuller // Journal of forecasting. – 1995. – P. 381–393.

23. Wong, F. Time series forecasting using backpropagation neural networks [Текст] // Neurocomputing. – 1991. – Vol. 2. – P. 147–159.

24. Zhao, R. Improving Neural Network Quantization without Retraining using Outlier Channel Splitting [Текст] / R. Zhao, Y. Hu, J. Dotzel // International Conference on Machine Learning. – 2019. – Vol. 97. – P. 7543–7552.

25. Wang, Z. RFPruning: A retraining-free pruning method for accelerating convolutional neural networks [Текст] / Z. Wang, X, Xie, G. Shi // Applied Soft Computing. – 2021. – Vol. 113.

26. Ahmed, N. An Empirical Comparison of Machine Learning Models for Time Series Forecasting [Текст] / N. Ahmed, A. Atiya, N. Gayar // Econometric Reviews. – 2010. – Vol. 29. – P. 594–621.

27. Voyant, C. Uncertainties in global radiation time series forecasting using machine learning: The multilayer perceptron case [Текст] / C. Voyant, G. Notton, C. Darras // Energy. – 2017. – Vol. 125. – P. 248–257.

28. Ebrahimpour, R. Mixture of MLP-experts for trend forecasting of time series: A case study of the Tehran stock exchange [Текст] / R. Ebrahimpour, H. Nikoo, S. Masoudnia // International Journal of Forecasting. – 2011. – Vol. 27. – P. 804–816.

29. Abbasimehr, H. Improving time series forecasting using LSTM and attention models [Текст] / H. Abbasimehr, R. Paki // Journal of Ambient Intelligence and Humanized Computing. – 2021. – Vol. 13. – P. 673–691.

30. Chimmula, V. Time series forecasting of COVID-19 transmission in Canada using LSTM networks [Текст] / V. Chimmula, L. Zhang // Chaos, Solitons & Fractals. – 2020. – Vol. 135.

31. Cao, J. Financial time series forecasting model based on CEEMDAN and LSTM [Текст] / J. Cao, Z. Li, J. Li // Physica A: Statistical Mechanics and its Applications. – 2019. – Vol. 519. – P. 127-139.

32. Kirisci, M. A New CNN-Based Model for Financial Time Series: TAIEX and FTSE Stocks Forecasting [Текст] / M. Kirisci, O. Yolcu // Neural Processing Letters. – 2022. – Vol. 54. – P. 3357–3374.

33. Livieris, I. A CNN–LSTM model for gold price time-series forecasting [Текст] / I. Livieris, E. Pintelas, P. Pintelas // Neural Computing and Applications. – 2020. – Vol. 32. – P. 17351–17360.

34. Rick, R. Energy forecasting model based on CNN-LSTM-AE for many time series with unequal lengths [Текст] / R. Rick, L. Berton // Engineering Applications of Artificial Intelligence. – 2022. – Vol. 113.

35. Mehtab, S. Analysis and Forecasting of Financial Time Series Using CNN and LSTM-Based Deep Learning Models [Текст] / S. Mehtab, J. Sen // Advances in Distributed Computing. – 2022. – Vol. 302. – P. 405–423.

36. Іващенко, Г. С. Короткострокове прогнозування нестаціонарних часових рядів з використанням моделей MLP та LSTM [Текст] / Г. С. Іващенко, В. О. Понамарьов, В. О. Холєв // Системи управління, навігації та зв'язку. – 2023. – № 1(71). – С. 91–95.

37. Іващенко, Г. С. Порівняльний аналіз методів прогнозування часових рядів на основі обчислювального інтелекту [Текст] / Г. С. Іващенко, В. О. Понамарьов // Проблеми інформатизації. – 2021. – № 1. – С. 97.

38. Іващенко, Г. С. Порівняльний аналіз моделей штучних нейронних мереж для прогнозування часових рядів [Текст] / Г. С. Іващенко, В. О. Понамарьов // Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління. – 2023. – № 2. – С. 68.