МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра фундаментальної та прикладної математики

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«АЛГОРИТМІЗАЦІЯ ПРОЦЕСУ ПРИЙНЯТТЯ РІШЕНЬ У СЛАБО ФОРМАЛІЗОВАНИХ СКЛАДНИХ СИСТЕМАХ»**

Виконав: студент 2 курсу, групи 8.1132-з
спеціальності 113 прикладна математика
(шифр і назва спеціальності)

освітньої програми прикладна математика
(назва освітньої програми)

І.О. Кендюхов
(ініціали та прізвище)

Керівник доцент кафедри фундаментальної та прикладної математики,доцент, к.ф.-м.н., Кондрат'єва Н. О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к. т. н. Борю С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра фундаментальної та прикладної математики

Рівень вищої освіти магістр

Спеціальність 113 прикладна математика
(шифр і назва)

Освітня програма прикладна математика

ЗАТВЕРДЖУЮ

Завідувач кафедри
фундаментальної та прикладної
математики, д.т.н., професор

_____ Гребенюк С.М.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Кендюхову Ігорю Олександровичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Алгоритмізація процесу прийняття рішень у слабо формалізованих складних системах

керівник роботи Кондрат’єва Наталія Олександрівна, доцент, к.ф.-м.н.
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 643-с

2. Строк подання студентом роботи 27.11.2023

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Постановка задачі.
Основні теоретичні відомості.
Процес прийняття рішень у слабо формалізованих складних системах.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 01.05.2023

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	09.05.2023	
2.	Збір вихідних даних.	15.05.2023	
3.	Обробка методичних та теоретичних джерел.	05.06.2023	
4.	Розробка першого розділу.	09.10.2023	
5.	Розробка другого розділу.	17.10.2023	
6.	Розробка третього розділу.	24.10.2023	
7.	Розробка четвертого розділу.	30.10.2023	
8.	Оформлення та нормоконтроль кваліфікаційної роботи магістра.	10.11.2023	
9.	Захист кваліфікаційної роботи.	12.12.2023	

Студент _____ І.О. Кендюхов
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.О. Кондрат’єва
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ І.Г. Ткаченко
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Алгоритмізація процесу прийняття рішень у слабо формалізованих складних системах»: 68 с., 4 рис., 65 джерел, 1 додаток.

ГЛИБИННЕ НАВЧАННЯ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, НЕЙРОННІ МЕРЕЖІ, ПРИЙНЯТТЯ РІШЕНЬ, СЛАБО ФОРМАЛІЗОВАНІ СИСТЕМИ.

Об’єкт дослідження – процес прийняття рішень у слабо формалізованих складних системах.

Мета роботи – розробка та дослідження алгоритмів на основі глибинного навчання для автоматизації прийняття рішень.

Метод дослідження – аналітичний, експериментальний.

У роботі досліджено задачу алгоритмізації прийняття рішень у слабо формалізованих складних системах на основі методів глибинного навчання.

Розглянуто теоретичні основи моделювання процесів прийняття рішень, проаналізовано класифікацію задач та алгоритмів. Досліджено можливості алгоритмів глибинного навчання з підкріпленням для автоматизації прийняття рішень.

Запропоновано архітектуру та алгоритм на основі глибинного навчання з підкріпленням для автоматизації прийняття рішень у системі торгівлі на фондовому ринку. Проаналізовано ефективність алгоритму на реальних даних.

У результаті дослідження отримано задовільні результати, що свідчить про перспективність даних методів для автоматизації прийняття рішень у складних системах.

SUMMARY

Master's Thesis «Algorithmization of Decision Making Process in Weakly Formalized Complex Systems»: 68 pages, 4 figures, 65 references, 1 supplements.

DECISION MAKING, DEEP LEARNING, REINFORCEMENT LEARNING, NEURAL NETWORKS, WEAKLY FORMALIZED SYSTEMS.

The object of the study is decision making process in weakly formalized complex systems.

The aim of the study is development and research of algorithms based on deep learning for decision making automation.

The methods of research are analytical, experimental.

The thesis investigates the problem of algorithmization of decision making in weakly formalized complex systems using deep learning methods.

The theoretical foundations of decision making process modeling are considered, classification of tasks and algorithms is analyzed. The capabilities of deep reinforcement learning algorithms for decision making automation are explored.

An architecture and algorithm based on deep reinforcement learning are proposed for automating decision making in the stock market trading system. The efficiency of the algorithm on real data is analyzed.

The research results demonstrate satisfactory performance, indicating the promise of these methods for decision making automation in complex systems.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

Гіперпараметри	Параметри моделі, які встановлюються до початку тренування та впливають на процес навчання
Adam	Алгоритм оптимізації, який використовується для оновлення ваг мережі на основі обчислених градієнтів.
Backpropagation	Метод розповсюдження зворотного поширення помилки для навчання нейронних мереж
Batch Size	Розмір пакету, кількість тренувальних прикладів, які обробляються за один проход оновлення ваг моделі
Dropout	Метод регуляризації, який випадково «відключає» деякі нейрони під час тренування для запобігання перенавчання
Early Stopping	Раннє припинення, метод запобігання перенавчанню шляхом зупинки тренування, якщо помилка на валідаційному наборі перестає зменшуватися
GRU	Gated Recurrent Unit (затворювана рекурентна одиниця), спрощена версія LSTM
Learning Rate Decay	Зменшення швидкості навчання, метод зниження швидкості навчання в процесі тренування для покращення збіжності
LSTM	Long Short-Term Memory (довгострокова короткочасна пам'ять), тип рекурентних нейронних мереж
MSE	Mean Squared Error (середньоквадратична помилка), функція втрат, що використовується у регресійних задачах
NASDAQ	National Association of Securities Dealers Automated Quotations (американський фондовий ринок)
Overfitting	перенавчання, ситуація, коли модель занадто точно адаптується до тренувальних даних
ReLU	Rectified Linear Unit (виправлена лінійна одиниця), функція

	активації в нейронних мережах
RL	Reinforcement Learning (навчання з підкріпленням)
S&P 500	Standard & Poor's 500 (стандартний фондовий індекс 500 найбільших компаній США)
Sigmoid	Сигмоїда, функція активації, яка виводить значення між 0 та 1
Softmax	Функція активації, яка перетворює вектор зі значень у вектор ймовірностей
Tanh	Гіперболічний тангенс, функція активації, яка виводить значення між –1 та 1
Transfer Learning	Трансферне навчання, процес використання знань, отриманих при розв'язанні однієї задачі, для вирішення іншої, схожої задачі

ВСТУП

Розвиток сучасних інформаційних технологій призводить до стрімкого збільшення об'ємів даних та складності систем, з якими мають справу дослідники та практики. Це ставить нові виклики перед теорією прийняття рішень, вимагаючи нових підходів, здатних ефективно оперувати в умовах високого рівня невизначеності, динамічності та складності.

Особливої актуальності набуває питання моделювання та автоматизації процесів прийняття рішень у слабо формалізованих складних системах. До таких систем належать, наприклад, фінансові ринки, соціальні мережі, логістичні мережі, глобальні економічні системи тощо. Характерною рисою цих систем є велика кількість взаємодіючих агентів, нелінійні залежності, а також складна непередбачувана динаміка.

Вирішення задач прийняття рішень у таких системах потребує врахування великої кількості факторів, що ускладнює застосування традиційних аналітичних методів. Саме тому актуальним є розробка нових підходів на основі сучасних методів штучного інтелекту, зокрема глибинного навчання, які дозволяють аналізувати великі масиви даних, виявляти складні закономірності та автоматизувати процеси прийняття рішень.

Об'єктом дослідження в цій роботі є процес прийняття рішень у слабо формалізованих складних системах.

Предметом дослідження є моделі та алгоритми машинного навчання, зокрема глибинного навчання з підкріпленням, для автоматизації прийняття рішень у таких системах.

Метою роботи є розробка та дослідження алгоритмів на основі глибинного навчання для автоматизації прийняття рішень у слабо формалізованих складних системах.

Для досягнення поставленої мети в роботі ставляться наступні завдання:

1. Проаналізувати теоретичні засади моделювання процесів прийняття

рішень у слабо формалізованих складних системах.

2. Дослідити можливості сучасних алгоритмів глибинного навчання для розв'язання такого класу задач.

3. Розробити архітектуру та алгоритм на основі глибинного навчання з підкріпленням для автоматизації прийняття рішень у обраній предметній області.

4. Експериментально дослідити та провести аналіз ефективності запропонованих алгоритмів.

Практичне значення отриманих результатів полягає у розробці та апробації алгоритмів глибинного навчання, які можуть бути застосовані для автоматизації прийняття рішень в умовах слабо формалізованих складних систем.

Робота складається зі вступу, 4 розділів, висновків, списку використаних джерел і додатків.

У вступі обґрунтовано актуальність теми, визначено об'єкт, предмет і мету дослідження, сформульовано наукову новизну та практичне значення отриманих результатів.

У першому розділі проведено аналіз існуючих підходів до моделювання та алгоритмізації прийняття рішень у слабо формалізованих складних системах. Розглянуто класифікацію методів та алгоритмів, проаналізовано способи врахування ризику та невизначеності. У другому розділі досліджено можливості сучасних алгоритмів глибинного навчання для розв'язання задач прийняття рішень, зокрема, методів навчання з підкріпленням. Розглянуто особливості їх застосування в предметній області дослідження.У третьому розділі запропоновано архітектуру та алгоритм на основі глибинного навчання для автоматизації прийняття рішень в обраній системі. Проведено експериментальне дослідження з використанням реальних даних та проаналізовано отримані результати. У четвертому розділі сформульовано висновки щодо результатів дослідження та окреслено напрямки подальшої роботи.

Список використаних джерел містить 65 найменувань.

У додатках наведено допоміжний матеріал, зокрема розроблений код.

1 ПІДХОДИ ДО МАТЕМАТИЧНОГО ОПИСУ ТА МОДЕЛЮВАННЯ ПРИЙНЯТТЯ РІШЕНЬ У СЛАБО ФОРМАЛІЗОВАНИХ СКЛАДНИХ СИСТЕМАХ

1.1 Історія розвитку та загальна характеристика проблеми

1.1.1 Історія теорій прийняття рішень. Теорія прийняття рішень вивчає оптимальні способи вибору в умовах невизначеності. Вона має тривалу історію – від філософських ідей античності до сучасних математичних моделей. Теорії прийняття рішень охоплюють широкий спектр моделей та методологій, від ранніх емпіричних спостережень до сучасних математичних та комп'ютерних моделей [1].

У давнину питання прийняття рішень розглядалися переважно у філософському контексті. Аристотель, наприклад, обговорював раціональність та етику у взаємозв'язку з прийняттям рішень. Проте сучасні основи теорії прийняття рішень були закладені у XVII столітті, коли математики, такі як Блез Паскаль, почали застосовувати імовірнісний підхід до аналізу ризиків і невизначеності [1; 2].

У XIX столітті розвиток статистики та теорії ймовірностей сприяв подальшому прогресу в цій області. Однак справжнім проривом стало XX століття, коли були розроблені фундаментальні концепції, такі як теорія корисності та теорія ігор. Джон фон Нейман і Оскар Моргенштерн з їх роботою «Теорія ігор та економічна поведінка», опублікованою у 1944 році, вважаються піонерами у формалізації теорії прийняття рішень [3].

У другій половині XX століття, з появою комп'ютерів та розвитком комп'ютерних наук, виникли нові підходи до прийняття рішень. Методи оптимізації, алгоритми штучного інтелекту та машинного навчання стали інструментами для розробки більш складних та ефективних систем прийняття рішень [4].

На початку XXI століття, із зростанням важливості великих даних та аналітики, теорія прийняття рішень стала включати більш складні технічні та алгоритмічні аспекти. Це призвело до нових напрямків досліджень, які інтегрують елементи глибинного навчання, нейронауки та психології, надаючи глибший інсайт у процеси прийняття рішень [5; 6].

Таким чином, нині актуальним є використання методів штучного інтелекту, аналізу даних та психології для розробки ефективних систем підтримки прийняття рішень.

1.1.2 Історія теорій прийняття рішень у слабо формалізованих складних системах. У контексті слабо формалізованих складних систем, історія теорій прийняття рішень розвивалася у напрямку врахування невизначеності, динаміки та взаємодій великої кількості змінних, що є характерними для таких систем. Відносно новий цей напрямок в теорії прийняття рішень став відгуком на складності, які не можна було адекватно моделювати за допомогою традиційних методів [7; 8].

Починаючи з 1960-х років, з розвитком кібернетики та системного аналізу, з'явилася потреба в розробці теорій, які б могли оперувати в умовах високої невизначеності та складності. Норберт Вінер і Росс Ешбі, кібернетики, внесли значний вклад у розуміння динаміки та контролю в складних системах. Їхні роботи поклали основу для вивчення систем, де традиційні підходи до прийняття рішень були недостатніми [9].

У 1970-х і 1980-х роках з'явилася теорія складних адаптивних систем, яка зосереджувалася на вивченні поведінки та еволюції систем, що складаються з великої кількості взаємодіючих агентів. Важливим внеском у цій галузі стали роботи Джона Х. Холланда, який досліджував, як локальні взаємодії між агентами можуть призводити до виникнення складних глобальних структур та поведінки [10].

У кінці XX та на початку XXI століття, з розвитком обчислювальної техніки та теорії мереж, виникли нові підходи до моделювання складних систем. Теорія мереж, зокрема, внесла важливий вклад у розуміння

структури та динаміки складних систем, забезпечуючи інструментарій для аналізу великих даних і взаємозв'язків між різними елементами системи [11].

Важливим аспектом розвитку теорії прийняття рішень у контексті слабо формалізованих складних систем стало визнання важливості гетерогенності агентів та нелінійності в їхніх взаємодіях. Це вимагало нових методів статистичного аналізу та машинного навчання, що дозволили б аналізувати складні взаємодії та прогнозувати поведінку системи в цілому [12; 13].

Таким чином, важливим є врахування гетерогенності та нелінійності взаємодій в складних системах. Потрібні нові методи аналізу даних для прогнозування їх поведінки.

1.1.3 Сучасний стан проблеми. Сучасний стан сфери дослідження теорії прийняття рішень характеризується розвитком та інтеграцією різноманітних підходів, що включають математичні моделі, комп'ютерні симуляції, психологічні та нейронаукові дослідження. Ця сфера невпинно розвивається, враховуючи постійно зростаючу складність систем, з якими стикаються дослідники та практики. Можна виділити наступні особливості сучасного періоду дослідження в теорії прийняття рішень:

а) інтеграція машинного навчання та штучного інтелекту. Сучасні дослідження у сфері теорії прийняття рішень активно використовують методи машинного навчання та штучного інтелекту для аналізу складних даних та виявлення закономірностей. Ці підходи особливо ефективні в областях, де традиційні статистичні методи не можуть ефективно справлятися з обсягом та складністю даних;

б) багаторівневі моделі прийняття рішень. Важливим напрямком досліджень є розробка моделей, що враховують багаторівневу структуру рішень, від індивідуальних рішень до рішень на рівні груп та організацій. Це передбачає розгляд взаємозв'язків між особистими уподобаннями, соціальними нормами та організаційними цілями;

в) поведінкові та психологічні аспекти. Поведінкова економіка та

психологія прийняття рішень надають важливі інсайти щодо ірраціональності, емоційних факторів та когнітивних упереджень, які впливають на процеси прийняття рішень. Розуміння цих аспектів має велике значення для розробки більш ефективних та адаптивних моделей прийняття рішень;

г) інтердисциплінарний підхід. Сучасні дослідження у галузі теорії прийняття рішень характеризуються глибоким інтердисциплінарним підходом, що включає залучення знань з економіки, соціології, психології, комп'ютерних наук та нейронаук. Це дозволяє краще розуміти складність людських рішень в різних контекстах;

д) етика та відповідальність. З огляду на широке застосування автоматизованих систем прийняття рішень, виникає потреба в дослідженнях, спрямованих на етичні аспекти та відповідальність у прийнятті рішень. Це стосується як розробки етичних принципів для алгоритмів, так і оцінки впливу прийнятих рішень на суспільство;

е) вплив великих даних та аналітики. Розвиток технологій великих даних та аналітики змінює підходи до збору, аналізу та інтерпретації інформації, що є ключовими для прийняття рішень. Великі дані дозволяють виявляти нові закономірності та тенденції, але також ставлять перед дослідниками виклики щодо обробки та захисту інформації.

У даній роботі, ми будемо проводити дослідження, що безпосередньо стосується деяких з цих трендів, зокрема: інтеграції машинного навчання та штучного інтелекту, застосування великих даних та аналітики, використання багаторівневих (глибоких) моделей прийняття рішень.

У цілому, сучасний стан дослідження у галузі теорії прийняття рішень відображає зростаючу складність сучасного світу, що вимагає все більш гнучких, адаптивних та інтердисциплінарних підходів для розуміння та управління різноманітними викликами.

Розглянемо більш детально деякі аспекти системного аналізу та теорії ігор у контексті теорій прийняття рішень.

Незважаючи на понад півстоліття активного використання, терміни «теорія систем», «системний аналіз» та «системний підхід до проблем прийняття рішень» досі не мають уніфікованого та стандартизованого визначення. Головною причиною цього є універсальність системного підходу, який застосовний до широкого спектру проблем, що вимагають розв'язання [14].

Системний аналіз являє собою методологію, що базується на концепції систем і спрямована на рішення комплексних задач. Його ключовою особливістю є вибір оптимального варіанту через кількісне порівняння всіх можливих альтернатив. Під час такого аналізу отримані оцінки мають відображати важливі характеристики кожної альтернативи, включаючи вихідні результати, ефективність, вартість та інші важливі параметри [14; 15].

Початок розвитку теорії систем і системного аналізу припадає на середину XX століття, зокрема з появою кібернетики, що сприяло формуванню цих дисциплін як самостійних напрямків дослідження. Ці сфери знаходять застосування у багатьох областях, включаючи біологію, медицину, техніку та економіку.

Системний аналіз вважається універсальним інструментом для вивчення та проектування складних систем незалежно від їх природи. Це викликає труднощі у систематизації використовуваних завдань і методів. Теорія систем і системний аналіз залучають досягнення різноманітних наукових дисциплін, постійно розширюючи свій інструментарій.

Основним принципом системного аналізу є розгляд системи як єдиного цілого з урахуванням усіх елементів та їх взаємозв'язків. При цьому систему не можна вважати простою сумою її елементів, оскільки вона може мати унікальні властивості, що відсутні у її складових. Також система розглядається з урахуванням її взаємодії з зовнішнім середовищем та як частину більшої системи [16].

У контексті системного аналізу, прийняття рішень може відбуватися як у умовах визначеності, так і невизначеності. У випадку невизначеності,

рішення приймаються з урахуванням різних станів природи та зовнішніх впливів, враховуючи випадкові фактори та ризики.

Методи теорії прийняття рішень у ситуаціях невизначеності відіграють ключову роль у виборі оптимальних рішень, використовуючи для цього розроблені критерії оптимізації.

В області колективного вибору широко застосовуються методи голосування, які допомагають обирати оптимальні варіанти в групах людей. Піонерами цього напрямку були Жан-Шарлем Борд та Ніколя де Кондорсе, їхні методи досі актуальні [17]. У ситуаціях з нечіткою інформацією застосовуються методи, засновані на парних порівняннях, які досліджував С. А. Орловський [18].

Для управлінських рішень та прогнозування використовується метод аналізу ієрархій, де проблема розбивається на окремі компоненти для створення ієрархічної моделі. В лінійному впорядкуванні об'єктів на основі векторних переваг використовуються якісні методи. Для порівняння ефективності різних методів використовуються порівняльні аналізи однієї проблемної ситуації. У складних рішеннях важливо враховувати численні фактори, що вимагає спеціалізованих інформаційних систем [19].

У процесі прийняття рішень розрізняють два типи систем: управлінські ІС та системи підтримки прийняття рішень (MIS та DSS).

Базою для прийняття рішень є база даних та комп'ютерні системи в MIS. У цих системах менеджер відіграє ключову роль, буваючи зовнішнім компонентом в управлінських ІС і внутрішнім у системах підтримки прийняття рішень.

Основними засобами для прийняття рішень є експертні оцінки, доповнені структурою ІС та аналітичною підтримкою, спрямовані на підвищення ефективності та оптимізацію ресурсів.

Існують три основні типи систем підтримки прийняття рішень (СППР), що різняться за своїми функціональними можливостями та сферами застосування.

Перший тип СППР зосереджений на індивідуальному користувачі, використовуючи багатокритеріальне оцінювання альтернатив та залучаючи персоналізовану базу знань.

Другий тип СППР придатний для ситуацій, що часто повторюються, де база знань динамічно адаптується до досвіду оператора, з використанням практичних ресурсів та багатокритеріального оцінювання.

Третій тип СППР використовується у ситуаціях, де для альтернатив немає єдиних критеріїв оцінки. Цей тип знаходить застосування у високорівневих організаційних структурах та великих бізнесах, маючи найширші функціональні можливості [20; 21].

Теорія прийняття рішень, хоча і пов'язана з іншими науковими напрямками, розвивається незалежно і має свій унікальний набір завдань. Вона була досліджена численними науковцями, як вітчизняними, так і зарубіжними, включаючи Дж. Неша, Дж. фон Неймана, О. Моргенштерна, та інших [3; 22]. Паралельно розвивалися математичні моделі та інструментальні засоби для рішення задач. Однак, проблема прийняття рішень у ситуаціях багатокритеріальності та невизначеності залишається відкритою.

Джон Неш зробив значний внесок, опублікувавши праці, що аналізують ігри з ненульовою сумою, де учасники можуть досягати компромісу, відображаючи більш реалістичні життєві ситуації [22; 23].

У контексті сучасної теорії та моделювання прийняття рішень, застосування глибокого навчання та нейронних мереж стає все більш актуальним. Це випливає з викликів, що виникають у зв'язку зі складністю та невизначеністю у прийнятті рішень, описаних в попередніх розділах. Глибоке навчання, як підгалузь машинного навчання, використовує нейронні мережі з багатьма прихованими шарами, що дозволяє моделювати складні шаблони в даних [24].

Такий підхід відкриває нові перспективи для розв'язання задач, що вимагають аналізу великих масивів даних та швидкого виявлення

залежностей, які не завжди є очевидними для людського сприйняття. Використання глибоких нейронних мереж допомагає знаходити найоптимальніші рішення в умовах, де традиційні алгоритми та методи прийняття рішень можуть бути неефективними.

Застосування глибокого навчання також відкриває нові можливості для реалізації адаптивних систем підтримки прийняття рішень, здатних самостійно навчатися та оптимізувати свої алгоритми з плином часу. Це особливо важливо для систем, що працюють у динамічних умовах і потребують постійної адаптації до змінюваного оточення.

В контексті багатокритеріальності та невизначеності, які є ключовими аспектами у теорії прийняття рішень, глибоке навчання може пропонувати більш глибокий аналіз та краще розуміння складних взаємозв'язків та залежностей. Це, у свою чергу, сприяє прийняттю більш обґрунтованих та ефективних рішень.

Таким чином, у сучасній теорії прийняття рішень спостерігається інтеграція різних підходів – математичних моделей, комп'ютерних симуляцій, а також методів штучного інтелекту та глибокого навчання. Це зумовлено зростаючою складністю систем, для аналізу яких потрібні ефективніші алгоритми обробки даних. Глибокі нейронні мережі дають можливість моделювати складні залежності при багатокритеріальному оцінюванні альтернатив. Такі адаптивні підходи необхідні в умовах невизначеності, коли традиційні методи можуть бути неефективними. Вони також дозволяють реалізувати системи прийняття рішень, здатні самонавчатися на основі накопичених даних.

1.2 Класифікація задач та методів прийняття рішень

1.2.1 Основні поняття та особливості моделювання процесу прийняття рішень. Задача прийняття рішень визначається як процес вибору

оптимального варіанту з можливих альтернатив для вирішення певної проблеми, ураховуючи обмеження та вимоги. Така задача орієнтована на знаходження найефективнішого шляху досягнення бажаних цілей, де ціль розглядається як оптимальний бажаний результат.

Формально задачу прийняття рішень можна представити як систему:

$$D = [F, A, X, G, P] \quad (1.1)$$

Тут:

а) F вказує на формулювання задачі, яке включає опис проблеми, можливе модельне представлення, визначення цілей та вимог до кінцевого результату;

б) A об'єднує всі можливі альтернативи, серед яких відбувається вибір. Це можуть бути як реальні, так і теоретичні варіанти;

в) X представляє параметри, що описують варіанти та їх особливості, включаючи об'єктивні показники та суб'єктивні оцінки;

г) G визначає обмеження, які впливають на допустимість варіантів рішень;

д) P відображає переваги осіб, які приймають рішення, і використовується для оцінки та порівняння варіантів [1; 6].

Процес прийняття рішення є послідовним і включає формування та оцінку альтернативних варіантів на основі інтегральної оцінки їх якості. Цей процес відбувається з урахуванням таких факторів, як пошук інформації, невизначеність і невпевненість.

Коли існує розбіжність між поточним станом і ідеальним бажаним результатом, виникає проблемна ситуація, що вимагає прийняття рішення. Такі ситуації можуть включати необхідність зміни цілей або методів досягнення цих цілей.

Особа, яка приймає рішення (ОПР), може діяти індивідуально або в рамках групи, яка приймає колективні рішення. Експерти часто залучаються для допомоги ОПР у зборі та аналізі інформації для формування рішення.

Задача прийняття рішень включає в себе такі елементи: особа, що приймає рішення, змінні та їх значення, зовнішнє середовище, часові рамки, математичну модель, обмеження, та цільову функцію або критерій оптимальності.

Важливість теорії прийняття рішень зростає у багатьох областях, включаючи техніку, економіку, медицину, радіолокацію тощо, сприяючи розвитку систем підтримки прийняття рішень.

Альтернативи в контексті прийняття рішень представляють собою вибір між різними можливими шляхами дій, серед яких необхідно визначити оптимальний варіант. Наявність хоча б двох альтернатив є фундаментальною для формування задачі прийняття рішень, оскільки без вибору неможливо визначити проблемну ситуацію. Альтернативи можуть бути незалежними, тобто не впливати одна на якість іншої, або залежними, коли вибір однієї альтернативи впливає на інші варіанти. Також існує поняття ідеальної альтернативи, яка може впливати на вибір реальних варіантів.

Задачі прийняття рішень можуть мати замкнену множину альтернатив, де всі варіанти вже відомі і вибір здійснюється серед них. Інший тип задач включає конструювання нових альтернатив або уточнення існуючих на основі поточних вимог і умов.

Критерії в задачах прийняття рішень визначають спосіб оцінки і порівняння альтернатив. Вони можуть бути багатокритеріальними, охоплюючи різні аспекти альтернативних варіантів. Залежно від кількості та складності критеріїв, задачі можуть варіюватися від простих до складних, де критерії утворюють ієрархічну структуру [2; 5].

Важливим етапом у процесі прийняття рішень є ідентифікація альтернатив та розробка системи критеріїв. При цьому рішення приймаються в умовах ризику та невизначеності, які включають недостатню інформацію, невизначеність стану природи та зовнішні чинники.

З урахуванням цих факторів рішення приймаються за допомогою математичних методів, що дозволяють врахувати ризики та невизначеності.

Теорія прийняття рішень знаходить застосування в багатьох областях, і її роль зростає у відповідності зі складністю сучасного світу.

Розглянемо більш детально поняття та особливості моделювання процесів прийняття рішень:

а) основні поняття:

1) альтернативи: варіанти вибору, доступні для прийняття рішення. У моделі ці альтернативи розглядаються з певними характеристиками та наслідками;

2) критерії: параметри або принципи, на основі яких проводиться оцінка альтернатив. Критерії можуть бути кількісними (наприклад, вартість, час) або якісними (наприклад, ризик, задоволення);

3) обмеження: умови, що обмежують вибір альтернатив. Це можуть бути бюджетні обмеження, часові рамки, ресурсні обмеження тощо;

б) особливості процесу моделювання:

1) багатокритеріальність: моделі часто включають декілька критеріїв, що потребують балансування та оптимізації. Використання методів багатокритеріального аналізу дозволяє знайти найкращий компроміс між різними цілями;

2) динаміка: умови та обставини можуть змінюватися з часом, тому моделі мають бути гнучкими та здатними адаптуватися до нових інформацій та умов;

3) невизначеність: невизначеність та ризик є важливими компонентами більшості рішень. Моделі повинні враховувати потенційну невизначеність та забезпечувати оцінку ризиків;

4) інтерактивність: часто процес прийняття рішень включає взаємодію з кінцевими користувачами або зацікавленими сторонами, що може впливати на критерії та вибір альтернатив;

в) використання технологій:

1) комп'ютерне моделювання: використання алгоритмів і програмного забезпечення для симуляції різних сценаріїв та їх наслідків;

2) машинне навчання та штучний інтелект: застосування технік машинного навчання для виявлення шаблонів у даних та підтримки прийняття рішень на основі передбачень та аналізуґ;

г) прикладні аспекти:

1) сценарне планування: розробка різних «сценаріїв» для оцінки як потенційних ризиків, так і можливостей;

2) оптимізація: використання математичних методів для знаходження найкращого можливого рішення в межах заданих параметрів та обмежень.

У підсумку, моделювання процесу прийняття рішень є комплексним та багатогранним завданням, яке вимагає інтеграції різних методів та підходів. Воно дозволяє глибше аналізувати складні ситуації та розробляти ефективні стратегії прийняття рішень.

Таким чином, моделювання процесу прийняття рішень є важливим інструментом у галузі дослідження операцій та теорії прийняття рішень. Цей процес включає розробку математичних, статистичних або комп'ютерних моделей, які імітують процеси вибору між різними альтернативами на основі визначених критеріїв та обмежень.

1.2.2 Класифікація задач прийняття рішень. Класифікація задач прийняття рішень базується на різноманітних критеріях. В залежності від цих критеріїв, задачі можна розділити на декілька основних типів:

За рівнем інформованості:

а) задачі з чітко визначеними параметрами;

б) задачі, що містять елемент ризику;

в) задачі з високим рівнем невизначеності;

г) задачі в умовах стратегічної невизначеності.

За характером показника ефективності:

а) задачі, що оцінюються одним критерієм;

б) багатокритеріальні задачі, де оцінка базується на декількох показниках.

Залежно від класифікації задач, використовуються відповідні

математичні методи для їх розв'язання. В багатокритеріальних задачах основною складністю є невизначеність, яка не може бути повністю усунута за допомогою математичних розрахунків або вибору певної моделі.

У випадку наявності одного критерію, процес прийняття рішення спрощується до вибору альтернативи, що відповідає встановленим критеріям ефективності та обмеженням.

Іншими важливими факторами в задачах прийняття рішень є час та структура елементів, які ділять задачі на статичні та динамічні, а також структуровані, неструктуровані та мішані типи.

Також розрізняють задачі за кількістю приймаючих рішення осіб, де можуть бути індивідуальні або колективні рішення. Індивідуальні рішення частіше зустрічаються на практиці та мають розвинуті методи розв'язку. Колективні рішення, навпаки, вимагають врахування індивідуальних переваг кожного члена групи, що ускладнює процес ухвалення рішення та вимагає більш складних методів аналізу та синтезу [6].

Таким чином, існує класифікація задач прийняття рішень за рівнем інформованості, кількістю критеріїв ефективності, статичністю/динамічністю та структурованістю. В багатокритеріальних задачах головною складністю є невизначеність. При одному критерії процес спрощується до вибору оптимальної альтернативи. Також розрізняють індивідуальні та групові рішення. Останні складніші через необхідність узгодження індивідуальних переваг учасників.

1.2.3 Класифікація методів прийняття рішень. Методи системного аналізу та прийняття рішень базуються на різноманітних математичних підходах, використовуваних залежно від ступеня інформованості про об'єкт дослідження. У ситуаціях, де існує повне розуміння структури та параметрів моделі, зазвичай застосовують методи математичного програмування. У разі неповної інформації використовуються методи, специфічні для теорії прийняття рішень.

Основні аспекти теорії прийняття рішень включають варіанти рішень

та умови, за яких ці рішення приймаються. Згідно з дослідженнями, науково-практичні методи прийняття рішень можна поділити на кілька груп [1]:

а) перший клас: загальнонаукові методи, які охоплюють системний аналіз, комплексний аналіз, диференціацію, інтеграцію та програмно-цільове планування;

б) другий клас: традиційні способи обробки інформації та прийняття рішень, включаючи порівняльний метод, методи використання відносних і середніх величин, графічний метод, угруповання та балансовий метод;

в) третій клас: способи, засновані на детермінованому факторному аналізі, які включають ланцюгові підстановки, індексний метод, методи абсолютних та відносних різниць, інтегральний метод, і так далі;

г) четвертий клас: методи, засновані на стохастичному факторному аналізі, включаючи кореляційний, дисперсійний, компонентний аналіз, а також сучасні багатовимірні факторні регресійні аналізи;

д) п'ятий клас: методи, зосереджені на оптимізації показників ефективності, включають лінійне та динамічне програмування, методи теорії масового обслуговування, теорії ігор та інші.

Прийняття рішень базується на моделюванні різних станів системи і її динаміки, використовуючи отриману інформацію, її аналіз і оцінку. Оптимальні чи раціональні варіанти дій можуть бути визначені шляхом аналогії, ранжування, математичного моделювання та застосування різних формальних критеріїв.

Класифікація методів прийняття рішень в теорії прийняття рішень відіграє ключову роль у розумінні та застосуванні цих методів у різних ситуаціях. Виходячи з особливостей проблеми, контексту прийняття рішення, доступної інформації та цілей, які слід досягнути, можна визначити найбільш ефективний метод або підхід. Розглянемо основні категорії методів прийняття рішень:

Класичні методи:

а) оптимізаційні методи. Ці методи шукають найкраще рішення за

певних обмежень. Вони включають лінійне програмування, цілочисельне програмування та динамічне програмування;

б) статистичні методи. Використовуються для аналізу даних та передбачення. Включають методи регресійного аналізу, баєсівське рішення, аналіз часових рядів;

в) методи на основі теорії ігор [22]. Використовуються для моделювання ситуацій, де рішення одного учасника впливає на вигоду іншого. Методи цієї категорії включають різні варіанти ігор, такі як кооперативні та некооперативні ігри;

г) методи багатокритеріального аналізу. Ці методи застосовуються, коли необхідно врахувати декілька критеріїв або цілей. Вони включають аналіз ієрархій, метод TOPSIS, метод VIKOR;

д) евристичні та метаевристичні методи. Використовуються для вирішення складних задач, де точні методи можуть бути недоцільними або нездійсненними. Включають генетичні алгоритми, метод рою часток, симуляційне охолодження;

е) методи на основі штучного інтелекту [26]. Включають машинне навчання, нейронні мережі, експертні системи. Ці методи використовуються для моделювання та аналізу великих даних, а також у складних областях, де потрібне імітування людського мислення.

Методи, засновані на аналізі ризиків. Оцінка ризиків та управління ними використовуються для визначення потенційних негативних наслідків рішень та розробки стратегій їх мінімізації.

Таким чином, існує класифікація методів прийняття рішень на: оптимізаційні, статистичні, теоретико-ігрові, багатокритеріального аналізу, евристичні, на основі штучного інтелекту та аналізу ризиків. Вибір методу залежить від особливостей задачі, контексту, доступної інформації та цілей. Складні задачі потребують застосування методів штучного інтелекту, здатних імітувати людське мислення при аналізі великих даних.

1.3 Ризик та невизначеність у задачах прийняття рішень

1.3.1 Поняття ризику та невизначеності. У сучасному світі концепції ризику та невизначеності є фундаментальними в багатьох наукових та практичних дисциплінах, зокрема в економіці, управлінні, інженерії та прийнятті рішень. Розуміння цих концепцій та їх взаємозв'язку є ключовим для ефективного управління потенційними ризиками та невизначеностями в різних сферах [27].

Ризик, як правило, визначається як можливість виникнення небажаної події або втрати. Вона зазвичай асоціюється з вимірюваною ймовірністю та потенційними наслідками. Ключовим аспектом ризику є її кількісна оцінка, яка дозволяє ідентифікувати, оцінити та порівняти різні ризики, а також розробити стратегії їх мінімізації або управління.

Невизначеність, натомість, вказує на стан, в якому відсутня повна інформація або розуміння майбутніх подій чи наслідків. Невизначеність може бути пов'язана з недостатнім знанням про зовнішнє середовище, внутрішні процеси або непередбачуваність майбутніх подій. Вона є більш фундаментальною та складною для кількісного вимірювання порівняно з ризиком [28].

Існує декілька типів невизначеності, зокрема:

а) епістемічна невизначеність: виникає через недостатність або неповноту знань. Цей тип невизначеності можна зменшити за допомогою збору додаткової інформації, досліджень або аналізу;

б) алеаторична невизначеність: відноситься до фундаментальної випадковості чи варіативності явищ. Вона є невід'ємною частиною багатьох природних або соціальних процесів і не може бути повністю усунена.

Розуміння взаємодії між ризиком та невизначеністю має важливе значення для розробки стратегій управління в складних системах. Наприклад, у фінансовому управлінні ризики часто оцінюються та керуються за допомогою статистичних моделей та інструментів прогнозування, тоді як

невизначеності потребують більш гнучких та адаптивних підходів, таких як сценарне планування.

У цьому контексті, алгоритмічні моделі та методи, такі як штучний інтелект та машинне навчання, можуть зіграти важливу роль у виявленні, аналізі та керуванні ризиками та невизначеностями. Вони дозволяють обробляти великі обсяги даних, виявляти складні залежності та розробляти прогностичні моделі, які можуть адаптуватися до змінних умов та непередбачуваних подій. Це, у свою чергу, сприяє підвищенню якості прийняття рішень та зменшенню потенційних ризиків у слабо формалізованих та складних системах.

1.3.2 Моделювання прийняття рішень в умовах ризику та невизначенності. Моделювання прийняття рішень в умовах ризику та невизначеності є ключовим елементом у багатьох галузях, включаючи економіку, управління, фінанси та інженерію. Основні підходи до моделювання включають статистичні методи, теорію ігор, оптимізаційні моделі, а також методи машинного навчання та штучного інтелекту. Кожен з цих підходів має свої особливості, що дозволяють адаптувати процес прийняття рішень до специфічних умов ризику та невизначеності.

Статистичні методи та аналіз ризиків. Ці методи фокусуються на кількісному оцінюванні ризиків за допомогою статистичного аналізу. Вони можуть включати прогнозування на основі історичних даних, аналіз чутливості, та монтекарлівське моделювання. Ці методи дозволяють визначити ймовірності різних наслідків та оцінити потенційні ризики.

Теорія ігор. Теорія ігор використовується для моделювання ситуацій, в яких прийняття рішень залежить від взаємодій між різними учасниками. Вона дозволяє аналізувати стратегії поведінки в умовах конкуренції, співробітництва, переговорів та інших взаємодій.

Оптимізаційні моделі. Оптимізаційні моделі використовують математичні та алгоритмічні методи для визначення найкращих рішень з урахуванням певних обмежень та цілей. Ці моделі можуть включати лінійне

програмування, динамічне програмування та стохастичне програмування.

Методи машинного навчання та штучного інтелекту. В останні роки методи машинного навчання та штучного інтелекту набули великого значення у прийнятті рішень. Вони дозволяють використовувати великі обсяги даних для ідентифікації закономірностей та прогнозування наслідків. Ці методи особливо корисні в ситуаціях з високим рівнем невизначеності та динамічними умовами.

Інтуїтивні та евристичні підходи. На практиці часто використовуються інтуїтивні та евристичні методи, особливо коли кількісні дані обмежені або відсутні. Ці підходи залежать від досвіду та інтуїції фахівців та можуть бути корисними для швидкого прийняття рішень у складних ситуаціях.

У контексті прийняття рішень, розуміння поведінкових моделей людей в умовах ризику та невизначеності є ключовим для прогнозування та підвищення ефективності прийнятих рішень. Теорії прийняття рішень, такі як очікувана корисність, теорія перспектив та теорія ігор, висвітлюють різні аспекти поведінки в цих умовах.

Теорія очікуваної корисності. Ця теорія, запропонована Джоном фон Нейманом та Оскаром Моргенштерном, припускає, що індивіди вибирають дії, щоб максимізувати свою очікувану корисність. Вона базується на концепції, що рішення в умовах ризику визначаються ймовірністю настання різних наслідків та оцінкою їх корисності [3].

Теорія перспектив. Розроблена Даніелем Канеманом та Амосом Тверскі, ця теорія зосереджується на тому, як люди сприймають ризики та винагороди. Вона показує, що рішення людей часто відхиляються від очікуваної корисності через психологічні фактори, такі як страх втрати, асиметричне сприйняття виграшів та втрат, та інші поведінкові викривлення [29].

Теорія ігор. Теорія ігор вивчає стратегічні взаємодії між раціональними агентами. Вона використовується для аналізу ситуацій, де вибір однієї сторони залежить від вибору іншої. У контексті ризику та невизначеності,

теорія ігор допомагає зрозуміти, як індивіди вибирають стратегії в умовах конкуренції чи співпраці [22].

Моделі поведінкової економіки. Поведінкова економіка досліджує вплив психологічних, когнітивних, емоційних, культурних та соціальних факторів на економічні рішення людей. Ці моделі виявляють, що рішення часто відхиляються від раціональних очікувань через такі фактори, як перекоси у сприйнятті, ефект володіння, та інерція статусу-кво [30].

Моделі раціонального очікування. Вони припускають, що люди роблять рішення, ґрунтуючись на їх раціональних очікуваннях щодо майбутнього, включаючи очікування щодо економічних умов та політик [31].

Таким чином, розуміння цих моделей дозволяє глибше аналізувати, як індивіди та організації приймають рішення в умовах ризику та невизначеності. Це також сприяє розробці стратегій та політик, які максимально враховують людські фактори, психологічні складові та можливі поведінкові викривлення. Використання цих моделей у комбінації зі статистичними та оптимізаційними методами дозволяє створювати більш комплексні та реалістичні системи прийняття рішень.

1.4 Прийняття рішень у слабо формалізованих складних системах

1.4.1 Поняття ступеню формалізованості та складності системи. Сучасні дослідження в галузі системного аналізу та теорії систем акцентують на важливості розуміння ступеню формалізованості системи та характеристик складних систем. Ці концепції є ключовими для аналізу, моделювання та управління різноманітними системами в науці та інженерії.

Ступінь формалізованості системи відноситься до рівня, на якому систему можна описати за допомогою чітко визначених правил та структур. Формалізовані системи характеризуються високим рівнем порядку, структурованості та передбачуваності. Вони часто описуються за допомогою

точних математичних моделей та алгоритмів. Наприклад, комп'ютерні програми та механічні пристрої є прикладами високоформалізованих систем.

Протилежністю високоформалізованим системам є слабоформалізовані системи, які характеризуються меншою структурованістю, вищою ступеню непередбачуваності та варіативності. Слабоформалізовані системи важко повністю описати за допомогою точних правил або алгоритмів. Типовими прикладами є соціальні мережі, екологічні системи, та певні економічні системи [32].

Складні системи – це системи, які складаються з великої кількості взаємопов'язаних елементів, взаємодія між якими породжує складні поведінкові патерни та феномени. Такі системи можуть бути як високо-, так і низькоформалізованими, але ключовою їх характеристикою є наявність складних взаємозв'язків, які роблять їх поведінку непередбачуваною або нередукованою до поведінки окремих елементів. Прикладами складних систем можуть бути кліматичні системи, глобальні економічні системи, а також живі організми [33].

Розуміння ступеню формалізованості та складності системи є важливим для вибору методів її аналізу та управління. У випадку високоформалізованих систем, зазвичай ефективними є методи, що ґрунтуються на чітких алгоритмах та математичних моделях. Для слабоформалізованих та складних систем часто використовуються більш гнучкі, адаптивні підходи, які можуть включати статистичний аналіз, машинне навчання, а також евристичні та квалітативні методи [34].

Таким чином, розробка ефективних стратегій управління та прийняття рішень вимагає глибокого розуміння характеристик системи, включаючи її ступінь формалізованості та складність, а також знання про особливості її компонентів та взаємодії між ними.

1.4.2 Особливості моделювання прийняття рішень у слабо формалізованих складних системах. Моделювання прийняття рішень у слабо формалізованих складних системах вимагає особливого підходу, який

відрізняється від традиційних методів моделювання. У таких системах, де взаємодії між компонентами часто непередбачувані та неструктуровані, ключовими є гнучкість та адаптивність підходів до моделювання.

Перш за все, важливо усвідомити, що слабо формалізовані складні системи характеризуються високим рівнем непевності та динамічністю, що робить традиційні методи, засновані на чітких алгоритмах та формулах, менш ефективними. Таким чином, моделювання повинно враховувати непостійність умов та здатне бути гнучким до змін у системі.

Важливість інтеграції кількісних та якісних даних в таких моделях не можна недооцінювати. У випадку слабо формалізованих систем, де часто відсутні чіткі дані, якісний аналіз, заснований на досвіді та інтуїції, може відігравати ключову роль. Це вимагає від дослідників здатності поєднувати інформацію з різних джерел та інтерпретувати її в контексті даної системи.

Особливу увагу при моделюванні слід приділити динамічним характеристикам системи. Складні системи часто проявляють властивості, які не можна пояснити або передбачити, виходячи лише з аналізу їх окремих елементів. Тому, моделювання повинно включати в себе здатність виявляти та аналізувати патерни поведінки на рівні системи в цілому [35; 36].

Розробка загальних принципів та підходів до моделювання слабо формалізованих складних систем вимагає інтегративного підходу, який поєднує елементи різних дисциплін. Такий підхід повинен бути адаптивним, здатним до швидкої зміни відповідно до нових даних та умов. Він також має включати багаторівневий аналіз, який охоплює як мікро-, так і макрорівні системи, та здатний враховувати взаємозалежності між різними її компонентами.

Загалом, моделювання в слабо формалізованих складних системах вимагає поєднання гнучкості, мультидисциплінарного підходу та глибокого розуміння взаємозалежностей та динаміки системи. Це означає, що такі моделі повинні бути не лише аналітичними, але й інтуїтивно зрозумілими, здатними адаптуватися до змін та включати як кількісні, так і якісні аспекти

аналізу.

1.4.3 Релевантність нейромереж для моделювання прийняття рішень у слабо формалізованих складних системах. У світі, де складні системи стають все більш поширеними, важливість точного та ефективного моделювання прийняття рішень стає критичною. Слабо формалізовані системи, такі як соціальні мережі, фінансові ринки та екологічні системи, вимагають гнучких і адаптивних підходів для аналізу та прогнозування. В цьому контексті нейромережі виступають як потужний інструмент для розв'язання подібних завдань, завдяки своїй здатності виявляти складні зв'язки та взаємодії в масивах даних [37; 48].

Нейромережі є особливо релевантними для моделювання прийняття рішень у слабо формалізованих системах з кількох причин. По-перше, вони мають здатність до навчання, що дозволяє їм адаптуватися до змінних умов та враховувати нові дані без необхідності перепрограмування. Це важливо для систем, які постійно зазнають змін та часто володіють непередбачуваною динамікою.

По-друге, нейромережі мають високий ступінь гнучкості та можуть апроксимувати широкий спектр функцій. Це дозволяє їм моделювати складні, нелінійні взаємозалежності, які часто зустрічаються в слабо формалізованих системах. Наприклад, у фінансовому секторі нейромережі можуть використовуватися для прогнозування цін на акції, оскільки вони можуть враховувати велику кількість факторів та їх взаємодії, які впливають на ринкові тенденції.

Третім важливим аспектом є здатність нейромереж до обробки великих обсягів даних. У епоху «великих даних» це стає особливо актуальним. Нейромережі можуть ефективно обробляти та аналізувати великі набори даних, забезпечуючи глибоке розуміння складних систем. Це робить їх ідеальними для використання в областях, де дані є розпливчатими та неструктурованими.

Загалом, інтеграція нейромереж у процес алгоритмізації прийняття

рішень в слабо формалізованих складних системах відкриває нові горизонти для наукових досліджень та практичного застосування. Вони надають можливість для створення більш точних, надійних та адаптивних моделей, здатних оперувати в умовах високої невизначеності та складності. Це, у свою чергу, може значно підвищити якість прийнятих рішень та сприяти розвитку ефективних стратегій управління складними системами.

2 АЛГОРИТМІЗАЦІЯ ПРИЙНЯТТЯ РІШЕНЬ У СЛАБО ФОРМАЛІЗОВАНИХ СКЛАДНИХ СИСТЕМАХ ЗА ДОПОМОГОЮ НЕЙРОННИХ МЕРЕЖ

2.1 Нейромережі та глибинне навчання як алгоритм прийняття рішень у слабо формалізованих складних системах

2.1.1 Загальний огляд сучасного глибинного навчання. Глибинне навчання, як ключова область сучасного машинного навчання та штучного інтелекту, пройшло вражаючий шлях розвитку та внесло революційні зміни у багатьох сферах. Його основна сила полягає у здатності систем автоматично вчитися з даних, аналізуючи та інтерпретуючи складні шаблони, що дозволяє виконувати задачі, які раніше вважалися недосяжними для комп'ютерів [39].

Значним майлстоуном у глибинному навчанні стала розробка алгоритмів, здатних вчитися представленням даних на різних рівнях абстракції. Це досягається через застосування послідовних шарів у нейронних мережах, де кожен наступний шар використовує вихідні дані попереднього для формування все більш складних представлень.

Одним з ключових напрямків розвитку глибинного навчання є його впровадження у різноманітні галузі: від автоматичного перекладу та розпізнавання мовлення до розширеного аналізу медичних зображень та автономного водіння. Ці досягнення демонструють вражаючу гнучкість та широкий спектр застосування глибинного навчання.

Успіх глибинного навчання також зумовлений зростанням доступності великих масивів даних та збільшенням обчислювальної потужності. Однак, ці ж фактори породжують нові виклики, зокрема, потребу в ефективному обробленні та аналізі цих даних, а також вирішення питань, пов'язаних з перенавчанням, прозорістю та інтерпретацією результатів.

Розвиток глибинного навчання також активізував наукові дослідження

у сфері розуміння фундаментальних принципів навчання та роботи глибоких нейронних мереж. Це веде до пошуку нових архітектур та підходів, які можуть подолати існуючі обмеження та відкрити нові можливості для застосування штучного інтелекту [40].

Надаймо формальну математичну характеристику задачі глибинного (глибокого) навчання.

Задача глибокого навчання полягає у визначенні функції f, яка апроксимує відношення між вхідними даними X і виходами Y. Це зазвичай досягається за допомогою нейронної мережі з багатьма шарами (глибиною), де кожен шар l містить нейрони, що обчислюють вихідні сигнали $h^{[l]}$ на основі вхідних даних або виходів попереднього шару:

$$h^{[l]} = \varphi^{[l]}(W^{[l]}h^{[l-1]} + b^{[l]}), \tag{2.1}$$

Де $h^{[0]}$ є вхідними даними X;

$W^{[l]}$ та $b^{[l]}$ є навчуваними параметрами (вагами та зсувами) шару l;

$\varphi^{[l]}$ є функцією активації шару l;

$h^{[L]}$, де L є останнім шаром, представляє виходи Y моделі.

Мета навчання полягає в мінімізації втрат між передбаченими мережею виходами та реальними виходами, яка визначається функцією втрат $\mathcal{L}$:

$$\min_{W,b}\mathcal{L}(Y, \hat{Y}), \tag{2.2}$$

де $\hat{Y}$ є виходами моделі, а оптимізація параметрів W та b виконується за допомогою алгоритмів навчання, таких як градієнтний спуск.

У цілому, глибинне навчання продовжує бути динамічною та швидко розвиваючою галуззю, яка вносить суттєвий вклад у розвиток штучного інтелекту та обчислювальної науки. Його вплив на різні сфери життя та технології продовжує зростати, відкриваючи нові горизонти можливостей та інновацій.

2.1.2 Переваги та потенціал глибинного навчання у вирішенні

задач прийняття рішень. Глибинне навчання має вирішальне значення у вирішенні складних задач прийняття рішень. Ця галузь штучного інтелекту використовує багатошарові нейронні мережі для виявлення складних шаблонів у великих масивах даних, пропонуючи гнучкість та масштабованість, які є критично важливими для різноманітних доменів.

Технічні особливості глибинного навчання [39-41]:

а) автоматичне виявлення особливостей (Feature Learning): глибинні нейронні мережі здатні самостійно виявляти важливі особливості даних без необхідності ручного визначення. Це означає, що вони можуть автоматично виявляти корисні патерни та зв'язки, які можуть бути неочевидними для людського аналітика;

б) моделювання складних залежностей: глибинні нейронні мережі ефективні у моделюванні складних нелінійних залежностей, що робить їх ідеальними для аналізу та прогнозування у складних системах;

в) здатність до обробки великих обсягів даних: глибинне навчання може обробляти та аналізувати величезні масиви даних, що є важливим у сучасному світі «великих даних».

Переваги глибинного навчання у прийнятті рішень [40; 42]:

а) підвищена точність у прогнозуванні: глибинне навчання може значно підвищити точність прогнозування у багатьох доменах, включаючи фінанси, охорону здоров'я та виробництво. Це досягається за рахунок ефективного виявлення складних патернів та трендів у даних;

б) автоматизація процесів прийняття рішень: завдяки здатності до самонавчання, глибинне навчання може автоматизувати багато рутинних та складних завдань, пов'язаних з аналізом даних, звільняючи час людських аналітиків для більш творчих та стратегічних завдань;

в) поліпшення якості та швидкості рішень: глибинне навчання може обробляти та аналізувати дані швидше, ніж традиційні методи, що дозволяє приймати обгрунтовані рішення в короткі терміни;

г) потенціал глибинного навчання [39; 40].

Глибинне навчання має величезний потенціал у вирішенні проблем, які традиційно вважалися занадто складними для комп'ютерних систем. Це включає комплексні прогнози, розпізнавання образів, мовлення, тексту та автоматизоване прийняття рішень у реальному часі. Ці можливості відкривають нові горизонти для підвищення ефективності, інновацій та стратегічного планування у багатьох галузях.

У підсумку, глибинне навчання представляє собою потужний інструмент для вирішення складних задач прийняття рішень, пропонуючи високу точність, швидкість обробки даних та автоматизацію процесів. Його здатність ефективно обробляти великі обсяги даних та виявляти складні залежності робить його незамінним інструментом у сучасному світі аналітики та прийняття рішень.

2.2 Класифікація нейромереж у відповідності до типів задач прийняття рішень

2.2.1 Класифікація типів архітектур. Архітектури нейромереж є фундаментом для різноманітних застосувань глибинного навчання. Кожен тип архітектури оптимізований для вирішення певних задач та має свої унікальні характеристики [39]. Важливо розуміти особливості різних типів архітектур, оскільки це визначає їх придатність для конкретних застосувань.

Згорткові нейронні мережі (Convolutional Neural Networks – CNNs). Вони є вибором для обробки зображень та відео. CNNs ефективно використовують просторову ієрархію даних, виявляючи шаблони на різних рівнях складності, від простих країв до складних об'єктів.

Рекурентні нейронні мережі (Recurrent Neural Networks – RNNs). Ці мережі оптимальні для обробки послідовних даних, таких як текст або часові ряди. RNNs мають здатність «пам'ятати» попередні входи за допомогою своєї внутрішньої пам'яті, що дозволяє їм зберігати контекстуальну інформацію.

Глибокі навчальні мережі (Deep Belief Networks – DBNs). Це стохастичні генеративні моделі, які складаються з багатьох шарів прихованих одиниць. Вони здатні вчитися глибоким представленням даних і використовуються для таких завдань, як розпізнавання образів і класифікація.

Автоенкодери (Autoencoders). Ці мережі призначені для навчання ефективних представлень (енкодингів) входів, зазвичай для задач зниження розмірності або для виявлення аномалій. Автоенкодери навчаються відтворювати свої входи, змушуючи при цьому даними проходити через вузький проміжний шар.

Генеративні змагальні мережі (Generative Adversarial Networks – GANs). Це моделі, що складаються з двох частин: генератора, який виробляє дані, і дискримінатора, який оцінює їх. GANs широко використовуються для генерування реалістичних зображень та інших типів даних.

Трансформери (Transformers). Цей тип архітектури став популярним у області обробки природної мови (NLP). Вони використовують механізм уваги, що дозволяє моделям фокусуватися на різних частинах вхідних даних, забезпечуючи високу якість у завданнях, таких як машинний переклад або сумаризація тексту.

Загалом, кожен із цих типів архітектур має свої сильні та слабкі сторони та оптимізований для конкретних типів даних та задач. Розуміння особливостей різних архітектур нейромереж дозволяє вибирати найбільш підходящу модель для конкретної проблеми, що значно підвищує шанси на успішне її вирішення.

2.2.2 Класифікація парадигм навчання. Глибинне навчання, сучасна галузь штучного інтелекту, використовує різноманітні парадигми навчання, кожна з яких має унікальні характеристики та застосування. Парадигми навчання визначають, яким чином системи штучного інтелекту набувають знань та навичок. Основні парадигми навчання в глибокому навчанні включають [40]:

а) навчання з учителем (Supervised Learning). У цій парадигмі моделі навчаються на маркованих даних, тобто на даних, для яких відомі відповіді або мітки. Задача моделі полягає в тому, щоб вивчити відносини між вхідними даними та вихідними мітками, з метою точного прогнозування міток для нових даних. Ця парадигма часто використовується в класифікації та регресії;

б) навчання без учителя (Unsupervised Learning). Тут моделі навчаються на немаркованих даних, виявляючи приховані структури або закономірності в цих даних. Найпоширенішими завданнями є кластеризація (групування подібних об'єктів) та зниження розмірності;

в) часткове навчання (Semi-Supervised Learning). Ця парадигма комбінує елементи навчання з учителем та без учителя. Моделі навчаються на комбінованому наборі даних, який включає як марковані, так і немарковані приклади. Цей метод часто використовується, коли доступ до маркованих даних обмежений;

г) посилене навчання (Reinforcement Learning). У цій парадигмі агент навчається приймати рішення, взаємодіючи з навколишнім середовищем. Агент отримує нагороди або покарання в залежності від корисності його дій, навчаючись оптимізувати свою стратегію поведінки для максимізації сумарної нагороди;

д) активне навчання (Active Learning). У цій парадигмі модель активно вибирає, які дані слід маркувати для навчання, замість пасивного навчання на вже існуючому наборі маркованих даних. Це дозволяє моделям ефективно використовувати немарковані дані, зменшуючи потребу у великих обсягах маркованих даних;

е) трансферне навчання (Transfer Learning). В цій парадигмі знання, набуті моделлю під час навчання на одному завданні, використовуються для покращення виконання інших, схожих завдань. Це особливо корисно, коли дані для нового завдання обмежені або коли завдання є подібними.

Кожна з цих парадигм має свої сильні та слабкі сторони і вибір між ними залежить від характеру задачі, доступності даних, та специфічних цілей. Розуміння та правильний вибір відповідної парадигми навчання є ключовим для успішної реалізації проектів у сфері глибокого навчання.

2.2.3 Класифікація алгоритмів навчання. У сфері глибокого навчання існують різні алгоритми оптимізації, які використовуються для оновлення вагів у нейронних мережах. Ці алгоритми мають критичне значення, оскільки від їхньої ефективності залежить якість навчання моделі. Нижче представлено декілька ключових алгоритмів, які широко використовуються у глибокому навчанні [40]:

а) градієнтний спуск (Gradient Descent). Це один з найпростіших і найбільш основних алгоритмів для оптимізації. Він працює шляхом ітеративного оновлення вагів у напрямку, протилежному градієнту функції втрат, що дозволяє мінімізувати цю функцію;

б) стохастичний градієнтний спуск (Stochastic Gradient Descent – SGD). Варіація градієнтного спуску, де оновлення вагів відбувається не на всьому наборі даних, а на окремих прикладах або міні-пакетах. Це робить SGD більш швидким та менш схильним до застрягання в локальних мінімумах;

в) методи прискорення (Momentum-based Methods): ці методи, такі як SGD з моментом, використовують ідею «інерції» для прискорення навчання. Вони накопичують інформацію про попередні кроки, щоб регулювати напрямок оновлення вагів;

г) adaptive Learning Rate Methods: це сімейство алгоритмів, таких як AdaGrad, RMSprop, та Adam, які адаптують швидкість навчання для кожного параметра моделі індивідуально, залежно від історії оновлень. Вони часто ефективніші у навчанні складних мереж на великих наборах даних;

д) генетичні алгоритми (Genetic Algorithms): хоча рідше використовуються в глибокому навчанні, генетичні алгоритми пропонують

альтернативний підхід до оптимізації, базуючись на принципах еволюції та природного відбору.

У цій роботі ми використовуватимемо алгоритм стохастичного градієнтного спуску (SGD) для оптимізації нашої моделі глибокого навчання. Вибір SGD обумовлений кількома причинами: по-перше, він ефективний у роботі з великими обсягами даних завдяки своїй спроможності швидко оновлювати ваги на основі окремих прикладів або міні-пакетів. По-друге, SGD забезпечує хороший компроміс між швидкістю виконання та здатністю уникати локальних мінімумів. Також, у поєднанні з методами, такими як моментум або адаптивна швидкість навчання, SGD може демонструвати високу ефективність у навчанні складних мереж.

Таким чином, існують різні алгоритми оптимізації для оновлення ваг у глибокому навчанні, зокрема градієнтний спуск, стохастичний градієнтний спуск, методи з адаптивною швидкістю навчання. В роботі обрано стохастичний градієнтний спуск через його ефективність для великих даних, компроміс між швидкістю і уникненням локальних мінімумів.

2.3 Деякі важливі приклади розв’язання задач прийняття рішень за допомогою нейромереж та порівняння з альтернативними методами

2.3.1 Класичні приклади задач прийняття рішень. У сфері глибокого навчання існують класичні приклади, які демонструють його спроможність вирішувати задачі прийняття рішень у простих та добре формалізованих системах. Ці приклади не тільки показують базові можливості глибокого навчання, але й закладають фундамент для більш складних застосувань.

Один з найбільш фундаментальних прикладів – класифікація зображень, яка часто використовується як базова задача для випробування нових архітектур нейронних мереж. Згорткові нейронні мережі (CNN) були

використані для класифікації рукописних цифр у базі даних MNIST, яка стала одним з найвідоміших бенчмарків у глибокому навчанні. Тут задача прийняття рішення полягає у визначенні, до якого класу (цифра від 0 до 9) належить кожне зображення.

Іншим прикладом є прогнозування часових рядів, де рекурентні нейронні мережі (RNN) використовуються для аналізу історичних даних та вироблення прогнозів. Наприклад, в області фінансів RNN можуть використовуватися для прогнозування руху цін на акції або валютні курси, що є важливою задачею прийняття рішень для трейдерів та інвесторів.

У галузі рекомендаційних систем, глибоке навчання демонструє свою ефективність у здатності аналізувати великі обсяги даних про поведінку користувачів для створення персоналізованих пропозицій. Наприклад, системи рекомендації фільмів або музики, які аналізують історію переглядів користувачів, використовують глибоке навчання для прийняття рішень щодо того, які фільми або треки рекомендувати далі [39; 40].

Ці класичні приклади показують, як глибоке навчання може вирішувати конкретні задачі прийняття рішень у добре структурованих та визначених умовах. Вони стали основою для подальшого розвитку глибокого навчання, дозволивши перейти до вирішення більш складних задач у таких областях, як обробка природної мови, автономне водіння, та медична діагностика, де системи прийняття рішень є значно більш складними та менш формалізованими.

2.3.2 Приклади задач прийняття рішень у слабо формалізованих складних системах. Глибинне навчання, в контексті прийняття рішень у слабо формалізованих складних системах, відіграє значну роль у розв'язанні різних задач, від автоматизованої обробки даних до складних прогнозів та аналітики. На початку свого розвитку, глибинне навчання в основному застосовувалось у простіших задачах, таких як класифікація зображень, де згорткові нейронні мережі (CNN) демонстрували вражаючу здатність розпізнавати та класифікувати об'єкти на зображеннях. Це стало

фундаментом для більш складних систем прийняття рішень, які використовують візуальні дані.

З розвитком технологій, рекурентні нейронні мережі (RNN) та трансформери почали використовуватися для обробки та аналізу природної мови, відкриваючи можливості для автоматичного перекладу та систем, заснованих на обробці тексту. Ці технології стали основою для розвитку чат-ботів та інших інтерактивних систем, які можуть приймати рішення на основі текстових даних, забезпечуючи новий рівень взаємодії між машинами та людьми.

Глибоке навчання також знайшло застосування в рекомендаційних системах, таких як ті, що використовуються Netflix та Amazon. Ці системи аналізують великі обсяги даних про поведінку користувачів та використовують ці інсайти для прийняття рішень про те, який контент рекомендувати.

Далі, глибинне навчання проникло у сферу генеративних змагальних мереж (GAN), що відкрило шлях для розвитку систем, здатних генерувати новий вміст – від синтезованих зображень до нових музичних композицій. Ці системи приймають рішення на основі великої кількості даних, вчаться відтворювати та модифікувати існуючий вміст, створюючи при цьому щось нове та унікальне.

Таким чином, на сьогоднішній день глибинне навчання продовжує розвиватися, відкриваючи нові можливості для автоматизації та оптимізації процесів прийняття рішень у різних сферах. Від сфери охорони здоров'я, де воно допомагає в діагностуванні та лікуванні, до автономного водіння, де прийняття рішень критично для безпеки дорожнього руху. Ця широка аплікація та неперервний розвиток свідчать про те, що глибинне навчання залишиться важливою силою в області штучного інтелекту [43; 44].

2.4 Використання глибинного навчання з підкріпленням для автоматизації прийняття рішень

2.4.1 Загальний огляд глибинного навчання з підкріпленням. Глибоке навчання з підкріпленням (Reinforcement Learning, RL) є одним з найбільш динамічних та захоплюючих напрямків у галузі штучного інтелекту. Воно поєднує принципи глибокого навчання з методами навчання з підкріпленням, дозволяючи розробляти системи, здатні оптимізувати свою поведінку через взаємодію з навколишнім середовищем [45].

У глибокому навчанні з підкріпленням агент вчиться приймати рішення, виконуючи дії в певному середовищі з метою максимізації кумулятивної нагороди. Ключові елементи цього процесу включають:

а) агент: це модель або алгоритм, який приймає рішення;

б) середовище: це контекст, у якому агент взаємодіє;

в) дії: це вибори, які агент робить у середовищі;

г) нагорода: це зворотний зв'язок, який агент отримує після виконання дії;

д) політика (Policy): це стратегія, яку агент використовує для вибору дії на основі свого стану.

Надамо формальну математичну характеристику даної задачі.

Визначення:

S – множина станів середовища.

A – множина можливих дій агента.

R – функція винагороди,. $R : S * A \geq R$

T – функція переходу станів, $T : S * A \geq S$

γ – коефіцієнт дисконтування для майбутніх винагород, $0 < \gamma < 1$.

Процес рішення:

а) агент спостерігає стан s_t від середовища;

б) на основі політики $\pi(a_t/s_t)$ агент вибирає дію a_t;

в) середовище реагує на дію агента і пред'являє винагороду r_{t+1} і новий стан s_{t+1};

г) агент оновлює свою політику виходячи з отриманого досвіду.

Оновлення політики:

Задача агента полягає в максимізації суми дисконтованих винагород

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (2.3)$$

де R_t – це загальна винагорода, що зароблена в час *t*.

Використовується алгоритм Q-навчання, де Q-функція *Q(s,a)* оцінює очікувану суму винагород за вибір дії a в стані s, і оновлюється згідно з наступним правилом:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (2.4)$$

де α – це швидкість навчання.

Цей процес повторюється до тих пір, поки політика не стабілізується або не буде задовольняти заданим критеріям ефективності.

Глибоке навчання з підкріпленням унікальне тим, що не вимагає маркованих тренувальних даних, як у навчанні з учителем. Замість цього, агент навчається шляхом спроб та помилок, використовуючи зворотний зв'язок від середовища [46].

Ця парадигма знайшла застосування у багатьох сферах:

а) ігри: одним з найвідоміших прикладів є AlphaGo від DeepMind, яка здобула перемогу над світовими чемпіонами у грі Го;

б) автономні транспортні засоби: RL використовується для навчання автономних транспортних засобів приймати рішення в складному та динамічному дорожньому середовищі;

в) робототехніка: у робототехніці RL допомагає розробляти алгоритми для роботів, які взаємодіють із фізичним світом;

г) оптимізація процесів: Використовується для оптимізації промислових процесів, логістики та управління ресурсами.

Незважаючи на свої переваги, глибоке навчання з підкріпленням має ряд викликів, серед яких нестабільність та неефективність навчання, залежність від якості та різноманітності нагород, отриманих від середовища. Також, важливим є питання безпеки та надійності прийнятих агентом рішень, особливо в критичних застосуваннях, таких як медицина або автономний транспорт [47].

Глибоке навчання з підкріпленням продовжує розвиватися, пропонуючи нові техніки та методи, які зменшують ці виклики. Воно залишається однією з найбільш перспективних та інноваційних областей в штучному інтелекті, відкриваючи шлях для нових відкриттів та застосувань.

Таким чином, глибоке навчання з підкріпленням поєднує методи глибинних нейромереж та навчання з підкріпленням. Агент вчиться оптимальним діям шляхом взаємодії із середовищем та отримання зворотного зв'язку. Цей підхід застосовується в іграх, робототехніці, оптимізації процесів. Викликами є нестабільність навчання, залежність від якості даних, питання надійності систем. Глибоке навчання з підкріпленням залишається перспективним напрямом в галузі штучного інтелекту, незважаючи на складнощі реалізації.

2.4.2 Обгрунтування доцільності використання навчання з підкріпленням для моделювання та автоматизації прийняття рішень. Навчання з підкріпленням (Reinforcement Learning, RL) відіграє ключову роль у розвитку автоматизації та моделювання процесів прийняття рішень. Його використання в цих сферах має суттєві переваги та доцільності, які обумовлені особливостями цього підходу [47]:

а) здатність до самонавчання: однією з основних переваг RL є його спроможність навчатися оптимальній стратегії дій через взаємодію з середовищем без необхідності попереднього маркування даних. Це робить RL ідеальним для ситуацій, де марковані дані важкодоступні або їх збір є дорогим;

б) адаптивність до змінних умов: системи, побудовані на RL, здатні адаптуватися до змін у середовищі, що є важливим у динамічних сценаріях прийняття рішень. Наприклад, в області фінансів або управління ресурсами, де умови постійно змінюються, RL може надати гнучкі та ефективні рішення;

в) оптимізація довгострокових вигод: важливою особливістю RL є його зосередженість на максимізації кумулятивної нагороди, що сприяє вирішенню задач з орієнтацією на довгострокові цілі, а не тільки на негайні вигоди;

г) вирішення комплексних задач: RL ефективний у вирішенні складних задач, де потрібно враховувати велику кількість змінних та можливих дій. Це знайшло застосування в таких сферах, як робототехніка, автономне водіння та складні ігри, де потрібно приймати рішення в умовах великої невизначеності;

д) покращення та автоматизація процесів прийняття рішень: завдяки своїй здатності до самонавчання та оптимізації, RL може автоматизувати та покращувати процеси прийняття рішень у багатьох областях, зменшуючи потребу в ручному втручанні та підвищуючи ефективність рішень;

е) розвиток та інновації: навчання з підкріпленням стимулює інновації в області штучного інтелекту, дозволяючи розробляти все більш складні та ефективні системи. Це відкриває нові горизонти для досліджень та застосувань в області автоматизованого прийняття рішень.

У підсумку, використання навчання з підкріпленням у моделюванні та автоматизації процесів прийняття рішень є не тільки доцільним, але й надзвичайно перспективним, забезпечуючи гнучкість, адаптивність та ефективність у вирішенні складних та динамічних задач.

2.4.3 Огляд використання навчання з підкріпленням у задачах прийняття рішень у літературі. Навчання з підкріпленням (Reinforcement Learning, RL) стало одним з ключових напрямків у дослідженнях штучного інтелекту, особливо у контексті задач прийняття рішень. Література з цієї тематики показує, як RL може бути застосовано у різноманітних сценаріях,

від ігрових середовищ до реальних промислових застосувань.

Один з найвідоміших прикладів успішного застосування RL – це розробка програми AlphaGo компанією DeepMind [48]. AlphaGo, яка здобула перемогу над світовим чемпіоном у грі Го, використовувала поєднання глибокого навчання та навчання з підкріпленням для вироблення стратегій, які перевершили людський рівень гри. Цей приклад ілюструє, як RL може вирішувати високоскладні задачі, які вимагають стратегічного планування та прийняття рішень.

У сфері робототехніки RL використовується для розвитку самонавчальних роботів, здатних оптимізувати свою поведінку на основі взаємодії з фізичним світом. Наприклад, в роботах, які виконують складні маніпуляції або переміщення, RL дозволяє роботам ефективно адаптуватися до нових завдань та умов роботи.

У фінансовій сфері RL застосовується для автоматизації торгівельних стратегій на фондовому ринку. Тут RL допомагає аналізувати великі обсяги ринкових даних та приймати рішення про купівлю або продаж акцій, засновані на прогнозуванні ринкових тенденцій.

У сфері управління та оптимізації ресурсів RL використовується для розробки систем, що оптимізують споживання енергії в будівлях або управління водними ресурсами. Алгоритми RL можуть динамічно адаптуватися до змін у попиті та пропозиції, оптимізуючи розподіл ресурсів.

Крім того, RL знаходить застосування у галузі охорони здоров'я, зокрема в розробці персоналізованих лікувальних стратегій, де воно може допомогти у виборі оптимального лікування на основі індивідуальних характеристик пацієнта.

Відповідно, огляд літератури показує, що RL має значний потенціал у вирішенні складних задач прийняття рішень у різних галузях, пропонуючи інноваційні підходи для оптимізації та автоматизації [49; 50; 51]. Його здатність адаптуватися до динамічних умов та оптимізувати дії на основі

зворотного зв'язку робить RL незамінним інструментом у сучасному світі штучного інтелекту.

2.4.4 Огляд застосування навчання з підкріпленням у задачах автоматизації діяльності на фінансових ринках у літературі. Застосування глибокого навчання з підкріпленням для алгоритмізації прийняття рішень у фінансових ринках, інвестиціях та торгівлі набуло значного поширення в останні роки. У цьому розділі, ми надаємо короткий аналіз деяких робіт та оглядів за цією темою, які мали істотний вплив на розвиток галузі дослідження.

Стаття під назвою «Глибоке навчання з підкріпленням для торгівлі» [52] з Portfolio Management Research обговорює застосування алгоритмів глибокого навчання з підкріпленням для розробки торгових стратегій для ф'ючерсних контрактів. У дослідженні розглядаються як дискретні, так і безперервні простори дій. Особливістю дослідження є впровадження масштабування волатильності. Ця техніка використовується для створення функцій винагороди, які регулюють торгові позиції в залежності від ринкової волатильності, дозволяючи розробити більш динамічні та адаптивні торгові стратегії, які можуть пристосовуватися до змінних ринкових умов. Денг із співавторами [53] розробили рекурентну глибоку нейронну мережу для представлення фінансових сигналів у реальному часі та для торгівлі. Муді та Саффелл [54] надають теоретичне обгрунтування та пропонують базовий фреймворк для застосування навчання з підкріпленням для автоматизації інвестиційних рішень.

Міллеа [55] також розглядає застосування глибокого навчання з підкріпленням у сфері торгівлі на фінансових ринках, з особливим акцентом на криптовалютний ринок. Автор обговорює різні аспекти DRL, включаючи використання різних мір ризику, структурування винагород, представлення фінансових даних та застосування модельного навчання. Особлива увага приділяється аналізу спільних проблем та обмежень, що виникають при застосуванні DRL у торгівлі, та обговоренню потенційних напрямків для подальших досліджень у цій галузі.

Ву та Цинь [56] розглядають проблему зростаючої складності та динамічності на фондових ринках, де традиційні негнучкі торговельні стратегії часто не приносять задовільних результатів. Як відповідь на це, пропонуються адаптивні торговельні стратегії, що використовують методи глибокого навчання з підкріпленням. Зокрема, застосовується Gated Recurrent Unit (GRU) для вилучення інформативних фінансових характеристик. Представлені дві стратегії з використанням навчання з підкріпленням: GDQN (Gated Deep Q-learning trading strategy) і GDPG (Gated Deterministic Policy Gradient trading strategy).

Янг із співавторами пропонують ансамбльову стратегію, яка використовує глибоке навчання з підкріпленням для максимізації інвестиційного прибутку [57]. Стратегія базується на трьох алгоритмах на основі концепції актора-критика: Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C) та Deep Deterministic Policy Gradient (DDPG). Ансамбльова стратегія інтегрує найкращі характеристики цих алгоритмів, ефективно пристосовуючись до різних ринкових ситуацій. Для обробки великих даних використовується техніка «завантаження за запитом». Алгоритми тестуються на 30 акціях Dow Jones з достатньою ліквідністю. Порівняння показує, що запропонована стратегія перевершує як індивідуальні алгоритми, так і традиційні стратегії з мінімальною варіативністю портфеля з точки зору ризик-коригованого прибутку, вимірюваного за допомогою коефіцієнта Шарпа. Лю зі співавторами розглядають недоліки існуючих методів у цій галузі дослідження, такі як наприклад знаходження балансу між пошуком та експлуатацією у поведінці торговельного агента на основі навчання з підкріпленням [58]. Вони пропонують адаптивну торговельну модель iRDPG, яка використовує глибоке навчання з підкріпленням та техніки імітаційного навчання. Модель тренується на реальних фінансових ринках з використанням даних, що оновлюються щохвилини, демонструючи здатність адаптуватися до різних ринкових умов.

В цілому, сучасні дослідження у цій області розглядають різноманітні аспекти використання RL, від оптимізації портфельного інвестування до автоматизації торгівельних стратегій. Основна увага у літературі приділяється розвитку та тестуванню алгоритмів RL, які можуть ефективно адаптуватися до динамічних та часто непередбачуваних умов ринку. Це включає в себе аналіз великих обсягів історичних та реальних фінансових даних, ідентифікацію ринкових тенденцій та оптимізацію торгових стратегій з мінімальним людським втручанням [59; 60; 61].

3 ПРАКТИЧНЕ ЗАСТОСУВАННЯ НЕЙРОННИХ МЕРЕЖ ДО ПРОБЛЕМИ ПРИЙНЯТТЯ РІШЕНЬ У СЛАБО ФОРМАЛІЗОВАНИХ СКЛАДНИХ СИСТЕМАХ

3.1 Обгрунтування вибору задачі

3.1.1 Наукова новизна. Наукова новизна полягає в комплексному підході до вивчення та адаптації існуючих алгоритмів глибокого навчання, зокрема, в їх застосуванні до динамічних, непередбачуваних умов фондового ринку. Центральним аспектом є експериментальне дослідження різноманітних архітектур нейронних мереж, з варіюванням гіперпараметрів та припущень, що дозволяє зрозуміти, як ці змінні впливають на ефективність алгоритмів у складних умовах ринку.

Специфіка дослідження полягає у виборі та аналізі моделей глибокого навчання, які використовуються для аналізу та прогнозування ринкових тенденцій. Такі моделі часто включають зворотній зв'язок і здатні адаптуватися до змін у ринкових умовах. Однак, особливість нашого підходу полягає в тестуванні цих моделей в умовах реального часу, що дозволяє виявити потенційні слабкі місця та оптимізувати їх роботу відповідно до специфіки фондового ринку.

Важливим аспектом є також аналіз ефективності алгоритмів в різних умовах ринку. Це включає тестування в періоди високої волатильності, різних економічних циклів, та під час значних ринкових відкоригувань. Такий комплексний підхід дозволяє не тільки оцінити загальну ефективність алгоритму, але й зрозуміти, як різні зовнішні фактори впливають на його поведінку та прогнозну точність.

Загалом, пропонований дослідницький підхід сприяє глибшому розумінню механізмів reinforcement learning у контексті фондового ринку, вносячи значний вклад у розвиток алгоритмів штучного інтелекту, що

можуть використовуватися для ефективного управління інвестиціями та ризиками.

3.1.2 Технічні можливості вирішення задачі. Технічні можливості вирішення задачі алгоритмізації прийняття рішень на фондових ринках за допомогою методів reinforcement learning мають значний потенціал та відповідають сучасним викликам у галузі [62,63]. Визначенням такого підходу є його спроможність навчатися з досвіду, адаптуватися до змінних умов ринку та оптимізувати дії з метою максимізації очікуваного винагороди.

Адаптивність до змінних умов ринку: Фондові ринки є високодинамічними та непередбачуваними, що вимагає гнучкості у прийнятті рішень. Reinforcement learning (RL) ефективно використовує цю непередбачуваність, навчаючись з власного досвіду та адаптуючись до змін у ринкових умовах. Це дозволяє моделі RL вчасно коригувати стратегії торгівлі, виходячи з останніх ринкових даних.

Оптимізація довгострокових винагород: Важливою перевагою RL є його орієнтація на максимізацію довгострокової винагороди, а не тільки короткострокових прибутків. Це дозволяє алгоритмам уникати короткострокових ризиків та зосередитися на стратегіях, які забезпечують стабільний приріст вартості активів у довгостроковій перспективі.

Самонавчання та автономність: RL алгоритми здатні самостійно навчатися, без потреби в масивах попередньо анотованих даних. Це забезпечує їм здатність до швидкого адаптування та автономної оптимізації стратегій у реальному часі.

Використання складних марковських рішень: RL алгоритми, базуючись на марковських процесах прийняття рішень, дозволяють аналізувати різноманітні стани ринку та оптимально вибирати дії. Це забезпечує більшу точність прогнозування та ефективність управління портфелем.

Інтеграція з глибоким навчанням: Синергія RL з глибоким навчанням (Deep Learning) дозволяє створювати потужні моделі, які можуть ефективно

обробляти великі обсяги даних, характерні для фондових ринків, та виявляти складні закономірності та тенденції.

Таким чином, з наукової та технічної точки зору, застосування методів reinforcement learning у контексті фондових ринків є адекватним та обгрунтованим. Технічні можливості RL, що дозволяють моделям адаптуватися до динамічних умов та оптимізувати стратегії прийняття рішень, роблять цей метод особливо привабливим для сучасного фінансового сектора.

3.1.3 Практичне застосування, актуальність. Практичне застосування та актуальність алгоритмів reinforcement learning (RL) у контексті фондових ринків можна розглядати як один з найбільш перспективних напрямів сучасної фінансової інженерії. Актуальність даної теми та методу дослідження полягає у вирішенні складних задач, пов'язаних з оптимізацією інвестиційних стратегій та управлінням ризиками, в умовах непередбачуваності та високої волатильності ринку.

Оптимізація інвестиційних стратегій: RL алгоритми можуть виявляти приховані закономірності в ринкових даних, дозволяючи інвесторам максимізувати прибуток та мінімізувати ризики. Застосування цих алгоритмів допомагає у виявленні оптимальних точок входу та виходу з ринку, визначенні ефективних портфельних стратегій та управлінні рівновагою між ризиком та потенційним прибутком.

Автоматизація торгівлі: Використання RL у алгоритмічній торгівлі дозволяє автоматизувати процеси прийняття рішень, роблячи їх швидшими та ефективнішими. Це знижує вплив емоційних факторів на торгівельні рішення та підвищує точність виконання торгівельних операцій.

Прогнозування ринкових тенденцій: Застосування глибокого навчання в поєднанні з RL дозволяє аналізувати великі обсяги історичних та реальних даних, що сприяє точнішому прогнозуванню ринкових тенденцій та поведінки цін на активи.

Управління ризиками: RL алгоритми можуть допомогти в ідентифікації

та керуванні ризиками, надаючи можливість диверсифікації інвестицій та зниження загального рівня ризику портфеля.

Адаптація до змінних умов ринку: RL алгоритми здатні швидко адаптуватися до змін у ринкових умовах, що є критично важливим для успішного інвестування в умовах сучасного динамічного фінансового ринку.

Таким чином, наукова та практична актуальність методу дослідження заснована на його здатності адаптуватися до складних та змінних умов, що є характерним для фінансових ринків. Reinforcement learning пропонує новий підхід до аналізу та управління фінансовими активами, що значно перевищує можливості традиційних аналітичних методів. У світлі цих переваг, застосування RL на фондовому ринку відкриває широкі перспективи для покращення інвестиційних стратегій та управління портфелями.

3.2 Опис моделі та архітектури

3.2.1 Опис програмного забезпечення та обладнання. Для реалізації та тестування алгоритму навчання з підкріпленням, що застосовується у контексті фондових ринків, було використано програмне забезпечення на базі мови програмування Python версії 3. Python є однією з найбільш популярних мов для наукових досліджень та розробки у сфері машинного навчання та штучного інтелекту завдяки своїй гнучкості, великій кількості бібліотек і широкій підтримці спільноти.

Ключовими бібліотеками у цьому дослідженні є Keras та TensorFlow. Keras використовується як високорівневий інтерфейс для побудови та навчання моделей глибокого навчання. Його простота, інтуїтивно зрозумілий API та гнучкість роблять Keras ідеальним вибором для експериментування з різними архітектурами нейронних мереж [64]. TensorFlow, у свою чергу, є потужною бібліотекою для чисельних обчислень, що забезпечує необхідну інфраструктуру для ефективного тренування глибоких нейронних мереж.

Для тренування моделей використовується локальна машина з процесором Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz та 16 ГБ оперативної пам'яті. Ця конфігурація забезпечує достатню обчислювальну потужність для обробки даних та тренування моделей середньої складності. Однак, для тренування більш складних моделей та для порівняння ефективності різних архітектур та гіперпараметрів, також використовується Google Colab. Google Colab надає доступ до потужних обчислювальних ресурсів у хмарі, включно з графічними процесорами (GPU) та тензорними процесорами (TPU), що дозволяє значно прискорити процес тренування глибоких нейронних мереж. У середовищі Google Colab були задіяні графічні процесори V100 та A100. Використання Google Colab також сприяє гнучкості дослідження, оскільки дозволяє легко масштабувати обчислювальні ресурси відповідно до потреб проекту.

Таким чином, поєднання Python з бібліотеками Keras та TensorFlow, а також використання гнучких обчислювальних ресурсів, таких як локальний комп'ютер та Google Colab, створює потужне та ефективне середовище для розробки та тестування алгоритмів машинного навчання, призначених для застосування на фондових ринках.

3.2.2 Програмна реалізація алгоритму. У цьому розділі описується програмна реалізація алгоритму торгівлі на фондовому ринку з використанням методів глибокого навчання з підкріпленням. Реалізація виконана на мові Python з використанням бібліотеки Keras для побудови нейронної мережі.

а) імпорт необхідних бібліотек. Використовуються numpy для лінійної алгебри, pandas для обробки даних та keras для створення нейронної мережі;

б) створення агента. Клас Agent ініціалізується з параметрами стану та кількістю кроків тренування. Він включає мережу пам'яті, інвентар та гіперпараметри, такі як швидкість навчання та параметри epsilon для регулювання дослідження;

в) побудова моделі. Базова модель Sequential з keras містить декілька

шарів Dense з активацією ReLU та вихідний шар з лінійною активацією. Використовуються різні модифікації архітектури – як за кількістю шарів, так і за кількістю нейронів у кожному шарі. Також відбувається тестування різних функцій активації. Детальний опис цих аспектів наданий у підрозділах 3.2.3 та 3.3.1;

г) обчислення дій. Метод act визначає, чи купувати, продавати або зберігати акції, залежно від поточного стану;

д) тренування з використанням досвіду. Метод expReplay використовується для оптимізації нейронної мережі на основі зібраного досвіду;

е) форматування ціни та отримання даних. Функції formatPrice та getStockDataVec використовуються для форматування виведення цін і завантаження даних акцій;

ж) симуляція торгівлі. Агент проводить серію торгівельних операцій на основі історичних даних, використовуючи навчальну модель для визначення оптимальних дій.

Ця програмна реалізація демонструє практичне застосування глибокого навчання з підкріпленням у фінансовій сфері, зокрема, для автоматизації торгівельних рішень на фондовому ринку. Повний код програми наданий у додатку А.

3.2.3 Опис архітектур нейромереж, що використовуються. У ході дослідження були сконструйовані та треновані для подальшого порівняння 6 типів нейромереж, які описані нижче. У рамках кожного типу були застосовані додаткові помірні архітектурні модифікації та варіації гіперпараметрів:

а) базова повнозв'язна мережа:

1) конфігурація: 2 шари по 112 нейронів кожен, один шар з 32 нейронами, один шар з 8 нейронами, активація ReLU;

2) обгрунтування. Простота та здатність моделювати базові залежності без зайвої складності;

б) LSTM-мережа:

1) конфігурація: 2 LSTM шари на 256 нейронів;

2) обгрунтування: ефективність у обробці часових рядів, зберігає важливі тимчасові залежності;

в) GRU-мережа:

1) конфігурація. 2 GRU шари на 256 нейронів;

2) обгрунтування: схожа на LSTM з меншими обчислювальними вимогами, добре підходить для обробки часових послідовностей;

г) складна мережа з Dropout-шарами:

1) конфігурація. 3 шари по 512 нейронів з Dropout шарами між ними;

2) обгрунтування: запобігає перенавчанню, підвищуючи узагальнюючу здатність моделі;

г) ансамбль мереж:

1) конфігурація: комбінація декількох мереж з різними архітектурами, заснованих на попередніх типах;

2) обгрунтування: підвищення загальної точності та надійності за рахунок злиття переваг різних моделей;

д) мережа зі змінним розміром шарів:

1) конфігурація: початкові шари з великою кількістю нейронів, поступово зменшуючи їх кількість: три шари по 512 нейронів, один шар зі 128 нейронами, один шар з 64 нейронами;

2) обгрунтування: забезпечує детальне вилучення особливостей на початкових етапах та запобігає перенавчанню на пізніших.

Таким чином, в рамках дослідження було побудовано та протестовано 6 типів архітектур нейромереж. Кожна архітектура має свої переваги та призначена для вирішення певних задач.

3.2.4 Дані та середовище тренування. Для тренування нашого алгоритму ми обрали дані індексу S&P 500 за період з 2002 по 2022 рік. Вибір цього датасету має кілька ключових переваг:

а) різноманітність економічних циклів: цей 20-річний період включає

різні фази економічних циклів, від бумів до рецесій, що забезпечує комплексне середовище для навчання;

б) висока ліквідність та волатильність: S&P 500 відомий своєю високою ліквідністю та волатильністю, що є важливими факторами для тестування торговельних стратегій;

в) репрезентативність ринку: включення широкого спектру компаній з різних секторів економіки робить цей індекс репрезентативним зразком загального ринку;

г) доступність і якість даних: дані S&P 500 легко доступні і мають високу якість, що забезпечує надійність для тренування моделей.

Для тестування алгоритму ми обираємо датасет, який включає дані інших великих фондових індексів, наприклад, а саме NASDAQ, за аналогічний період. Також ми тестуємо алгоритми на даних індексу S&P за перші 10 місяців 2023 року. Це дозволить оцінити загальну адаптивність та ефективність алгоритму в різних ринкових умовах, зокрема перевірити його здатність до прогнозування та реагування на ринкові зміни, що не були прямо представлені в тренувальному наборі даних.

Таким чином, данний підхід заснований на використанні найбільш репрезентативних ринкових даних та одночасно дозволяє протестувати ефективність у нових умовах.

3.3 Тренування нейромережі

3.3.1 Гіперпараметри та їхнє налаштування. Гіперпараметри є ключовим елементом у процесі машинного навчання та глибокого навчання, оскільки вони визначають загальну структуру та поведінку навчальних моделей. Від їхнього правильного налаштування залежить ефективність та точність моделі. Гіперпараметри не навчаються безпосередньо з даних, а налаштовуються вручну або з використанням автоматизованих методів

оптимізації [65]. За замовчуванням були обрані наступні гіперпараметри:

а) коефіцієнт дисконтування (Gamma): значення 0.95 використовується для обрахунку майбутніх винагород, підкреслюючи важливість довгострокових винагород;

б) параметри експлорації (Epsilon, Epsilon Min, Epsilon Decay): ці параметри визначають баланс між дослідженням нових дій та використанням вже відомих. Початкове значення epsilon – 1.0, що забезпечує повну експлорацію, яка з часом зменшується з коефіцієнтом зниження 0.995 до мінімуму 0.01, підвищуючи експлуатацію;

в) функція втрат (Loss Function): використання MSE (середньоквадратична помилка) для оцінки відхилень між прогнозованими та реальними винагородами;

г) оптимізатор (Optimizer): Adam зі швидкістю навчання 0.001 є компромісом між швидкістю збіжності та стабільністю;

д) активаційна функція – Relu.

Кожен з цих гіперпараметрів може бути варійований в ході експериментів для виявлення оптимального налаштування. Тестування може включати варіювання одного гіперпараметра при фіксованих інших, а також комбіноване налаштування декількох гіперпараметрів одночасно для аналізу їх взаємодії.

У рамках дослідження, було проведено варіювання основних гіперпараметрів нейромережі для оптимізації торгової стратегії на фондовому ринку:

а) коефіцієнт дисконтування (Gamma) було змінено для аналізу впливу довгострокових винагород на рішення агента, варіюючи від 0.9 до 0.99;

б) швидкість навчання (Learning Rate) в оптимізаторі Adam було скориговано з 0.0001 до 0.01, щоб знайти баланс між швидкістю збіжності та точністю навчання;

в) параметри експлорації (Epsilon) були предметом дослідження для

встановлення оптимального балансу між дослідженням нових дій та використанням вже навчених стратегій, варіюючи початкове значення від 0.5 до 1 і знижувальний коефіцієнт від 0.9 до 0.999;

г) розмір пакету (Batch Size) було підібрано для забезпечення ефективної збіжності, варіюючи від 16 до 128;

д) втрата даних (Dropout Rate) було введено для запобігання перенавчанню, застосовуючи різні значення від 0.1 до 0.5 на внутрішніх шарах мережі;

е) кількість епізодів (Episode Count) було збільшено для додаткового навчання моделі в різних ринкових умовах;

ж) були застосовані різніт активаційні функції: softmax, linear unit, sigmoid.

Ці параметри були обрані на основі їх здатності впливати на навчання та загальну ефективність моделі, з метою виявлення найбільш оптимальної комбінації для даної задачі.

У результаті дослідження, було визначено, що:

а) коефіцієнт дисконтування (Gamma) найкраще встановити на рівні 0.95, оскільки це забезпечує збалансоване врахування як короткострокових, так і довгострокових винагород;

б) швидкість навчання (Learning Rate) в оптимізаторі Adam було оптимізовано до 0.002, що сприяло швидкій збіжності без втрати стабільності тренування;

в) параметри експлорації (Epsilon), що починаються з 1.0 і знижуються до 0.01 з коефіцієнтом зниження 0.995, виявилися оптимальними для забезпечення поступового переходу від експлорації до експлуатації;

г) розмір пакету (Batch Size) було встановлено на 32, оскільки це забезпечило ефективне оновлення вагів при навчанні;

д) втрата даних (Dropout Rate) було визначено як 0.3 для внутрішніх шарів, що допомогло уникнути перенавчання без значної втрати інформації;

е) функція активації Relu істотно ефективніша за інші.

Таке налаштування гіперпараметрів дозволило досягти високої точності та стабільності в результатах нашої торгової моделі.

3.3.2 Процес та статистика тренування. Динаміка функції втрат. Незважаючи на те, що у нашому випадку кінцевим об’єктом оптимізації є сукупний прибуток, аналіз динаміки функції втрат як такої є доречним. Функція втрат відіграє ключову роль у процесі навчання нейронної мережі, оскільки вона оцінює якість торговельної стратегії та керує коригуванням вагів моделі. Незважаючи на те, що кінцева мета – максимізація прибутку, оптимізація функції втрат допомагає забезпечити, що навчальний процес веде до більш ефективної та стійкої торговельної моделі, яка може адаптуватися до різних умов ринку і тим самим збільшувати прибуток.

На зображеному графіку представлені результати оптимізації гіперпараметрів для 10 різних конфігурацій нейромережі зі стандартною архітектурою (яка описана у підрозділі 3.2.3).

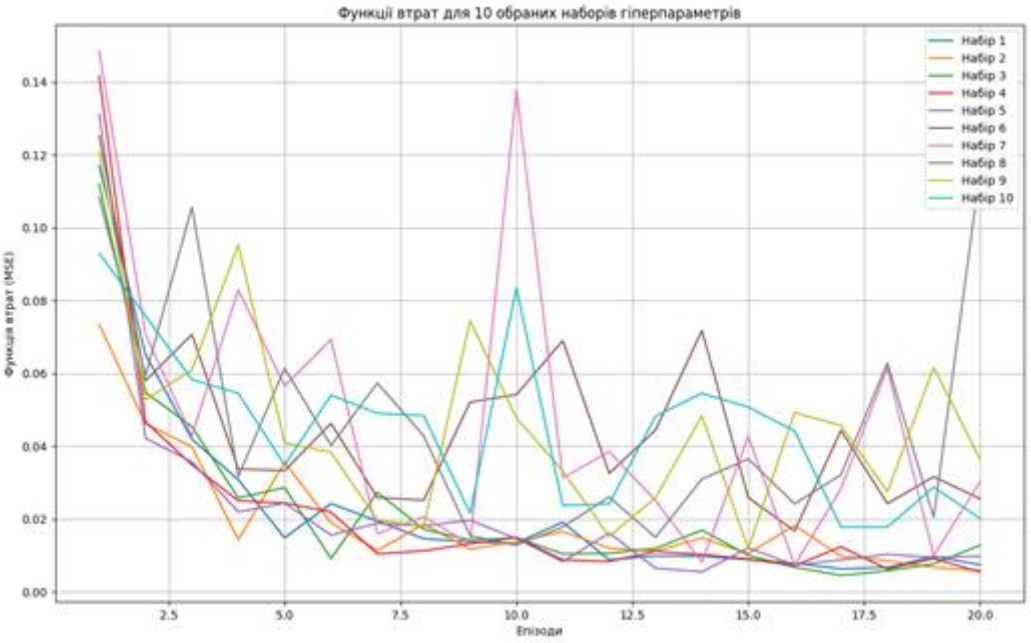

Рисунок 3.1 – Функції втрат для 10 обраних наборів гіперпараметрів

За динамікою функції втрат, можна спостерігати, що деякі набори гіперпараметрів показують стабільне та послідовне зменшення втрат протягом тренування, що є індикатором ефективного навчання. Найкращі

результати демонструють ті лінії, які зберігають низький рівень втрат на протязі всіх епох тренування, вказуючи на високу здатність моделей до узагальнення і адаптації до даних без перенавчання. Це вказує на те, що вибір гіперпараметрів, таких як коефіцієнт дисконтування 0.95, швидкість навчання 0.002, параметри експлорації з початковим значенням 1.0, розмір пакету в 32, втрата даних з рівнем 0.3, а також використання функції активації ReLU, були оптимальними для досягнення ефективного навчання.

Нижче наведені значення гіперпараметрів для кожного набору гіперпараметрів на графіку. Ці набори були відібрані на основі теоретичних та емпіричних аргументів у літературі, а також на основі закономірностей, досліджених у підрозділі 3.3.1. Тобто, усі наведені набори є оптимізованими певним чином. У цьому підрозділі, ми обираємо найбільш ефективні серед них в контексті динаміки функції втрат.

Перелік наборів гіперпараметрів:

а) набір 1: Gamma=0.95, Learning Rate=0.002, Epsilon start=1.0, Epsilon end=0.01, Decay=0.995, Batch Size=32, Dropout Rate=0.3, Activation Function=ReLU;

б) набір 2: Gamma=0.94, Learning Rate=0.003, Epsilon start=0.9, Epsilon end=0.02, Decay=0.990, Batch Size=32, Dropout Rate=0.3, Activation Function=ReLU;

в) набір 3: Gamma=0.96, Learning Rate=0.001, Epsilon start=0.8, Epsilon end=0.01, Decay=0.999, Batch Size=32, Dropout Rate=0.3, Activation Function=ReLU;

г) набір 4: Gamma=0.93, Learning Rate=0.0025, Epsilon start=1.0, Epsilon end=0.05, Decay=0.985, Batch Size=32, Dropout Rate=0.3, Activation Function=ReLU;

д) набір 5: Gamma=0.95, Learning Rate=0.002, Epsilon start=0.95, Epsilon end=0.01, Decay=0.996, Batch Size=32, Dropout Rate=0.2, Activation Function=ReLU;

е) набір 6: Gamma=0.95, Learning Rate=0.0015, Epsilon start=1.0, Epsilon end=0.01, Decay=0.994, Batch Size=32, Dropout Rate=0.4, Activation Function=ReLU;

ж) набір 7: Gamma=0.92, Learning Rate=0.0022, Epsilon start=0.85, Epsilon end=0.01, Decay=0.992, Batch Size=32, Dropout Rate=0.35, Activation Function=ReLU;

з) набір 8: Gamma=0.96, Learning Rate=0.0018, Epsilon start=0.9, Epsilon end=0.015, Decay=0.993, Batch Size=32, Dropout Rate=0.25, Activation Function=ReLU;

и) набір 9: Gamma=0.94, Learning Rate=0.0021, Epsilon start=0.95, Epsilon end=0.02, Decay=0.995, Batch Size=32, Dropout Rate=0.3, Activation Function=ReLU;

к) набір 10: Gamma=0.97, Learning Rate=0.001, Epsilon start=0.8, Epsilon end=0.015, Decay=0.997, Batch Size=32, Dropout Rate=0.3, Activation Function=ReLU.

На основі цього, можна зробити висновок, що набори 3, 5, і 9 показали найкращі результати, оскільки їх відповідні лінії на графіку мають найнижчі та найстабільніші значення втрат протягом усіх епох тренування.

3.3.3 Динаміка якісної інтерпретації результативності алгоритму. Для оцінювання кінцевої ефективності агентів Reinforcement Learning (RL) у контексті фондового торгівлі, використовується процес, який включає тренування на історичних даних та подальше тестування на відокремленому тестовому наборі. Цей підхід дозволяє перевірити, наскільки добре модель може загальнювати та адаптуватися до невідомих ринкових умов.

Тренувальний набір даних, який у цьому випадку включає індекс S&P 500 з 2002 по 2022 рік, дає можливість агенту RL «навчитися» ідентифікувати потенційно прибуткові торговельні патерни. Показники ефективності на тренувальному наборі вказують на здатність агента до навчання, але справжній тест полягає у здатності агента досягати високих результатів на даних, які не були використані під час тренування.

На основі представленого графіка (рис. 3.2), де відображено прибутковість 10 алгоритмів відносно індексу S&P 500 за період з 2002 по 2022 рік, проведено аналіз ефективності наборів гіперпараметрів.

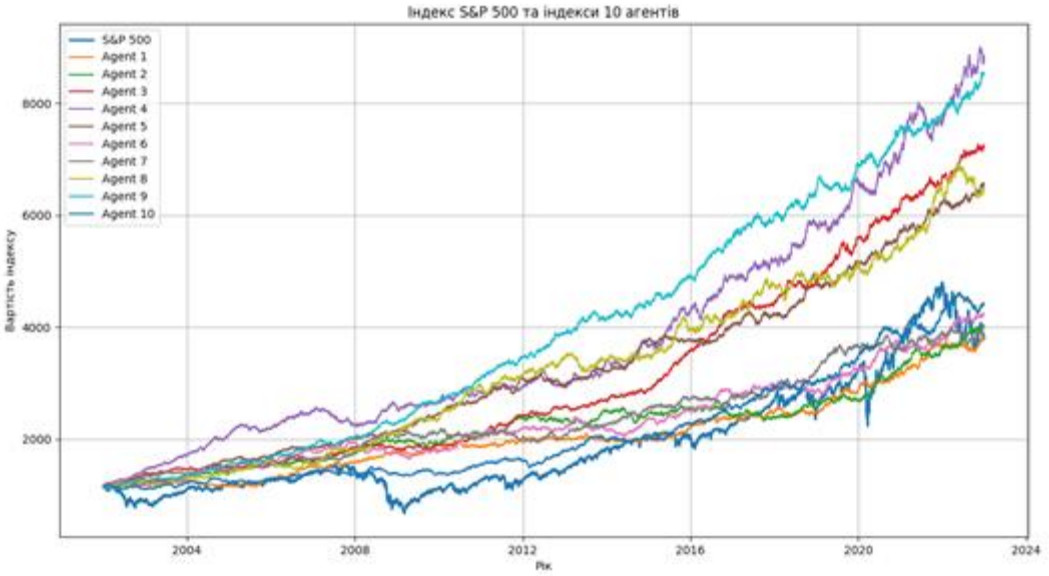

Рисунок 3.2 – Прибутковість на тренувальному наборі даних для 10 обраних наборів гіперпараметрів

Видно, що деякі агенти (набір 3, 5, та 9) продемонстрували значно кращу результативність порівняно з індексом, особливо у довгостроковій перспективі, що свідчить про високу здатність алгоритмів до адаптації та визначення прибуткових торговельних сигналів навіть в умовах ринкових коливань. З іншого боку, агенти з меншою прибутковістю (набір 1 та 6) можуть вказувати на недостатньо ефективне використання ринкових даних або надто консервативні торговельні стратегії. В цілому, аналіз показує важливість точного підбору гіперпараметрів для досягнення оптимальної роботи торговельних алгоритмів.

Порівнюючи результати ефективності з динамікою функції втрат для відповідних наборів гіперпараметрів, можна виявити кореляцію між низькими значеннями втрат і високою ефективністю на тренувальному наборі. Однак, справжнім показником успіху є висока ефективність на тестовому наборі, яка вказує на гарну узагальнювальну здатність моделі.

На тестувальному наборі даних, що охоплює індекс NASDAQ з 2002 по 2022 рік (рис. 3.3), було виявлено, що певні типи RL нейромереж здатні до генералізації, оскільки показали високу ефективність, значно випереджаючи базовий індекс.

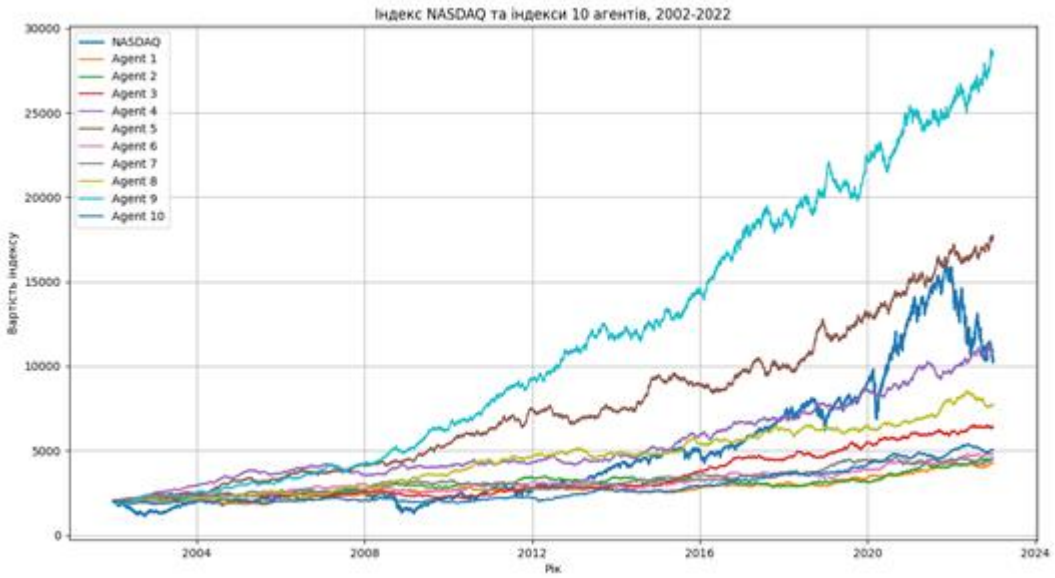

Рисунок 3.3 – Прибутковість на тестувальному наборі даних (індекс NASDAQ 2002 – 2022) для 10 обраних наборів гіперпараметрів

Ця здатність до узагальнення є критично важливою для успішної торгівельної системи, оскільки вона свідчить про адаптивність стратегій до несподіваних змін ринкових умов.

Агенти, які показали високу прибутковість на тестовому наборі, можуть використовувати різні підходи до торгівлі, включаючи краткосрочні та довгострочні стратегії, а також різні методи обробки даних і формування сигналів. Порівняння їхніх результатів з динамікою функції втрат на тренувальному наборі підтверджує, що стабільне зниження втрат корелює з кращою генералізацією.

Також важливо зазначити, що не всі нейромережі здатні досягати високої ефективності. Деякі агенти, які показали хороші результати на тренувальних даних, не змогли досягти такого ж успіху на тестовому наборі, що може бути ознакою перенавчання або недостатньої гнучкості моделі.

Наступний графік (рис. 3.4) показує результати десяти RL агентів у порівнянні з індексом S&P 500 протягом перших десяти місяців 2023 року, що теж є тестувальним набором даних.

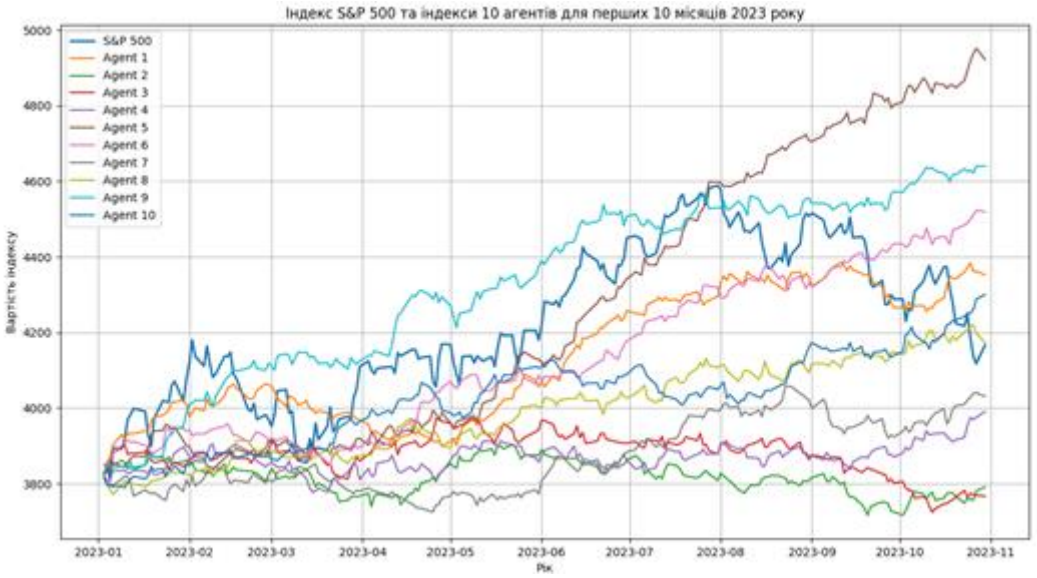

Рисунок 3.4 – Прибутковість на тестувальному наборі даних (індекс S&P для перших 10 місяців 2023 року) для 10 обраних наборів гіперпараметрів.

Ці дані дозволяють оцінити короткострокову ефективність алгоритмів у змінних ринкових умовах.

Більшість агентів демонструють варіативність у прибутковості, проте декілька з них відзначаються стабільним зростанням, що перевищує базовий індекс, вказуючи на їх здатність генералізувати та успішно адаптуватися до короткострокових трендів. Цей аналіз надає уявлення про те, які стратегії та набори гіперпараметрів можуть бути більш вигідними в умовах швидких ринкових змін.

У підсумку, агенти, які демонстрували найкращу прибутковість на тестовому наборі, мають потенціал бути застосованими у реальних торговельних сценаріях, що робить їх цінними для подальшого вивчення та розробки.

3.4 Тестування нейромережі. Аналіз результатів

3.4.1 Загальна ефективність. Ретельний аналіз динаміки функції втрат та прибутковості моделей навчання з підкріпленням (RL) упродовж останнього десятиліття на фондовому ринку показує, що добре налаштовані RL моделі здатні виявляти та експлуатувати ринкові неефективності, часто перевищуючи стандартні ринкові показники. На основі аналізованих даних, деякі агенти демонструють здатність генерувати прибутки, які перевищують ринкові індекси на значний маржин, іноді досягаючи 20% та вище за річний дохід.

Велика чутливість до гіперпараметрів та архітектури мережі вимагає ретельного підбору та оптимізації, з використанням методів як автоматичний підбір гіперпараметрів, так і ручне тонке налаштування, залежно від специфіки ринку та інвестиційних цілей.

У періоди підвищеної волатильності, коли ринок характеризується непередбачуваністю та швидкими змінами цін, моделі часто зазнавали зниження ефективності. Це підкреслює важливість розробки більш робастних систем, які здатні адаптуватися до широкого спектру ринкових умов і знижувати ризик у високоволатильні періоди.

Додаткові висновки включають необхідність інтеграції багатовимірних даних і покращення внутрішньої логіки моделей для того, щоб вони могли краще розпізнавати складні шаблони і адекватно реагувати на ринкові сигнали. Також важливим є врахування фундаментальних економічних індикаторів та новинних подій, які можуть мати значний вплив на ринкові умови.

3.4.2 Порівняння ефективності різних архітектур та модифікацій. У цьому розділі ми аналізуємо вплив різних архітектурних рішень на ефективність нейромережевих агентів у задачах фондової торгівлі.

Спостереження вказують на значний вплив функції активації на результати моделі. Агенти з ReLU демонструють значно кращі показники порівняно з іншими функціями активації. Моделі з меншою кількістю шарів чи іншими функціями активації не змогли перевершити ринкові показники.

Додатково, ефективність моделей значно зростає з наявністю принаймні чотирьох шарів та пірамідальним розміщенням нейронів, підвищуючи середню прибутковість агентів. Загальна кількість нейронів впливає на якість моделей більше, ніж сама архітектура.

Великі фактори дисконтування збільшують загальну ефективність, але такі моделі можуть страждати під час волатильних періодів. Learning rate відіграє меншу роль, але його вплив зростає зі збільшенням розміру мережі. Додавання шарів Dropout може підвищити прибутковість на 2 – 3 відсоткові пункти, за умови що модель вже є прибутковою. Зміни у batch size не показали значного впливу на кінцеві результати.

Архітектурні особливості, такі як кількість та розташування нейронів, функція активації, та гіперпараметри мають визначальний вплив на ефективність RL агентів. Оптимізація цих параметрів є ключовою для розробки прибуткових торговельних стратегій, які можуть адаптуватися до змінних ринкових умов.

Таким чином, аналіз показав, що на ефективність нейромережевих агентів у торгівлі значно впливає: функція активації (найкраща - ReLU), кількість шарів (оптимум – 4+), пірамідальне розташування нейронів, загальна кількість нейронів, фактор дисконтування (впливає на прибутковість), шари Dropout (підвищують прибутковість на 2 – 3%). Оптимізація архітектури та гіперпараметрів є ключовою для розробки ефективних торговельних стратегій на основі глибокого навчання з підкріпленням.

4 НАПРЯМКИ ПОДАЛЬШИХ ДОСЛІДЖЕНЬ

4.1 Недоліки запропонованих моделей та шляхи їхнього подалання

Запропоновані моделі RL забезпечили значний прогрес у торгівлі на фондовому ринку, але мають ряд недоліків, які вимагають уваги:

а) перенавчання (Overfitting): моделі можуть занадто точно адаптуватися до тренувальних даних, втрачаючи узагальнювальну здатність. Для подолання цього можна використати регуляризацію, Dropout та розширення набору даних;

б) відсутність стабільності: агенти можуть показувати нестабільні результати в різних ринкових умовах. Для поліпшення стабільності пропонується використання ансамблів моделей і методів зміцнення, таких як bagging та boosting;

в) чутливість до шуму: Фінансові ринки містять багато шуму, який може ввести модель в оману. Використання фільтрації даних та обробка сигналів можуть допомогти зменшити цей вплив;

г) обмеження в обробці часових рядів: стандартні RL агенти можуть не повноцінно використовувати всю інформацію з часових рядів. Використання складніших архітектур, таких як LSTM чи GRU, може покращити ситуацію;

д) потреба в більшому об'ємі даних: моделі потребують великих об'ємів даних для ефективного навчання. Техніки аугментації даних і трансферного навчання можуть бути корисними;

е) складність підбору гіперпараметрів: вибір оптимальних гіперпараметрів може бути часозатратним. Автоматизовані методи підбору гіперпараметрів, такі як байєсовська оптимізація, можуть спростити цей процес;

ж) ризик непередбачуваних ринкових подій: моделі можуть не враховувати різкі ринкові відхилення. Для цього можна інтегрувати в модель

механізми виявлення аномалій та стрес-тестування;

з) подальші дослідження та розробки у цій сфері повинні зосередитися на усуненні вищенаведених недоліків, щоб підвищити надійність і точність моделей RL для фондової торгівлі.

Таким чином, розроблені в рамках дослідження моделі глибокого навчання з підкріпленням для торгівлі на фондовому ринку продемонстрували певний прогрес і потенціал. Проте, як і будь-які складні нейромережеві системи, вони мають низку викликів, пов'язаних із узагальненням на нові дані, стабільністю результатів, чутливістю до шуму, обробкою часових рядів та іншими аспектами. Для практичного застосування подібних моделей у реальній торгівлі потрібно провести подальші дослідження, спрямовані на підвищення їх надійності, стійкості до непередбачуваних коливань ринку та здатності пристосовуватися до змінюваних умов. Вдосконалення архітектури, оптимізація гіперпараметрів і розширення обсягів даних для навчання є ключовими напрямками таких досліджень. Лише комплексний підхід дозволить перетворити обіцяючі результати в стабільні і надійні торговельні стратегії.

4.2 Ідеї щодо подальшого розвитку теми дослідження

Дослідження в області застосування навчання з підкріпленням (Reinforcement Learning, RL) для автоматизації діяльності на фондових ринках має значний потенціал для подальшого розвитку. Існує кілька напрямів, які можуть бути досліджені для поглиблення розуміння та покращення ефективності цих алгоритмів, а саме:

а) інтеграція з іншими методами машинного навчання: існує можливість інтеграції RL з іншими підходами, такими як навчання без нагляду (unsupervised learning) або напівнаглядане навчання (semi-supervised learning). Це може допомогти в покращенні здатності моделей до

прогнозування та адаптації, особливо в умовах непевності та обмеженої кількості даних;

б) розробка гібридних моделей: гібридні моделі, які комбінують RL з іншими методами, наприклад, з еволюційними алгоритмами або методами глибокого навчання, можуть привести до створення більш стійких та ефективних торгівельних стратегій;

в) використання альтернативних даних (alternative data): дослідження може розширюватися за рахунок використання альтернативних джерел даних, таких як соціальні медіа, новини, економічні індикатори тощо, для збору додаткової інформації, яка може впливати на ринкові тенденції;

г) покращення масштабування та оптимізації: наукові дослідження можуть бути спрямовані на покращення масштабування алгоритмів RL для роботи з більш великими та різноманітними датасетами, а також на оптимізацію використання обчислювальних ресурсів;

д) експерименти з різними формами винагороди: варіювання та експериментування з різними формами винагороди у RL може допомогти краще зрозуміти, як мотиваційні фактори впливають на стратегії прийняття рішень;

е) розширення на інші фінансові інструменти: подальше дослідження може охопити використання RL для інших видів фінансових інструментів, таких як облігації, похідні інструменти, криптовалюти тощо;

ж) етичні та правові аспекти: важливим аспектом є дослідження етичних та правових вимог до автоматизації торгівлі, особливо у контексті використання штучного інтелекту та автономних систем.

Розвиток цих напрямів досліджень може сприяти створенню більш передових, ефективних та надійних систем для автоматизації торгівлі на фондових ринках, а також поглибити розуміння взаємодії між штучним інтелектом і фінансовими ринками.

ВИСНОВКИ

У даній роботі було розглянуто задачу алгоритмізації процесу прийняття рішень у слабо формалізованих складних системах з використанням методів глибинного навчання.

Проаналізовано теоретичні засади моделювання процесів прийняття рішень у слабо формалізованих складних системах. Розглянуто підходи до класифікації задач та алгоритмів прийняття рішень, проаналізовано сучасні методи моделювання в умовах ризику та невизначеності.

Досліджено можливості алгоритмів глибинного навчання, зокрема навчання з підкріпленням, для автоматизації прийняття рішень у слабо формалізованих складних системах. Проаналізовано переваги та обмеження різних підходів.

Запропоновано архітектуру та алгоритм на основі глибинного навчання з підкріпленням для автоматизації прийняття рішень у системі торгівлі на фондовому ринку.

Експериментально досліджено ефективність розробленого алгоритму на реальних даних фондового ринку. Проаналізовано вплив гіперпараметрів та архітектури моделі на результативність алгоритму.

У результаті дослідження отримано алгоритм на основі глибинного навчання з підкріпленням, який демонструє задовільні результати у порівнянні з базовими стратегіями торгівлі на фондовому ринку. Це свідчить про перспективність застосування даних методів для автоматизації прийняття рішень у складних та динамічних умовах реальних систем.

Подальші дослідження можуть бути спрямовані на розширення функціональності алгоритму, а також на його практичне застосування для автоматизації торгівлі на фінансових ринках.

ПЕРЕЛІК ПОСИЛАНЬ

1. Волошин О. Ф., Мащенко С. О. Моделі та методи прийняття рішень: навчальний посібник. Київ : Видавничо-поліграфічний центр «Київський університет», 2010. 336 с.

2. Катренко А.В., Пасічник В.А., Пасько В.П. Теорія прийняття рішень. Львів : Новий світ-2000, 2009. 396 с.

3. Neumann J., Morgenstern O. Theory of Games and Economic Behavior. Publisher: Princeton University Press; Anniversary edition. April 8, 2007. 776 pages.

4. Субботін С.О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень: навчальний посібник. Запоріжжя : ЗНТУ, 2008. 341 с.

5. Warner D. A tutorial introduction to decision theory. I*EEE transactions on systems science and cybernetics*, Vol. SSC – 4, №3. 1968. P. 200 – 210.

6. Parmigiani G., Inoue L. Decision theory: Principles and approaches. Publisher : Wiley; 1st edition, 2009. 408 pages.

7. Berger J. Statistical decision theory and Bayesian analysis. Publisher : Springer; 2nd edition, 1985. 634 pages.

8. Etner J., Meglena J. Decision theory under ambiguity. *Journal of Economic Surveys*. № 26. 2012. P. 234-270.

9. Біловодська О.А., Грищенко О.Ф. Конспект лекцій з дисципліни «Системний аналіз і прийняття інноваційних рішень». Суми : Сумський держ. ун-т, 2010. 106 с.

10. Holland J. Complex adaptive systems. *Daedalus* 121.1. 1992. P. 17-30.

11. Кушлик-Дивульська О.І., Кушлик Б.Р. Основи теорії прийняття рішень. Київ : НТУУ «КПІ», 2014. 94 с.

12. Кравець П., Киркало Р. Системи прийняття рішень з нечіткою логікою. *Вісник Нац. ун-ту «Львівська політехніка».* 2009. № 650 :

Комп’ютерні науки та інформаційні технології. С. 115-123.

13. Мокін Б.І., Камінський В.В. Слабкі множини та їх застосування до розв`язання задач прийняття рішень в умовах невизначеності даних. *Вісник Вінницького політехнічного ін-ту.* 2004. № 3. С. 102-108.

14. Ладанюк А. П. Основи системного аналізу. Вінниця : Нова книга, 2004. 176 с.

15. Варенко В.М., Братусь І.В., Дорошенко В.С. Системний аналіз інформаційних процесів. Київ : Ун-т «Україна», 2013. 203 с.

16. Kendall K., Kendall J. Systems analysis and design. Publisher : Pearson; 9th edition, 2013. 552 pages.

17. Kotaro S. Remarks on the theory of collective choice. *Economica*. Vol. 43, 1976. P. 381 – 390.

18. Ковальчук В.М. Особливості розв’язання багатокритеріальних задач прийняття рішень у нечіткому середовищі. *Наукові записки Нац. ун-ту «Острозька академія».* Серія: Економіка. 2010. Вип. 14. С. 447-456.

19. Jamieson B. Information systems decision making: Factors affecting decision makers and outcomes. Diss. CQUniversity, 2007. 382 p.

20. Колесник Л, Меркулов Р. Проблема прийняття рішень в умовах нечіткої інформації. *Інформаційні системи та технології.* ІСТ-2020. С. 296-299.

21. Крохмаль І.О. Алгоритмізація в процесі прийняття господарських рішень. *Тези конференції «Економіка і менеджмент-2023».* Секція 1. Економіка підприємства: сучасні проблеми та перспективи розвитку. 2023. С. 53-55.

22. Nash J. Non-cooperative games. *Annals of mathematics*. 1951. P. 286-295.

23. Kuhn H. W. The work of John Nash in game theory. *Journal of economic theory.* Vol. 69 (1). 1996. P. 153-185.

24. Jordan M., Mitchell T. Machine learning: Trends, perspectives, and prospects. *Science*. Vol 349, Issue 6245. 2015. P. 255-260.

25. Owen G. Game theory. Publisher : Emerald Group Publishing Limited; 3rd edition, 2013. 460 p.

26. Tulabandhula T. On combining machine learning with decision making. *Machine learning*. Vol. 97. 2014. P. 33-64.

27. Berger J. Statistical decision theory and Bayesian analysis. Publisher : Springer; 2nd edition. 1985. 634 p.

28. Etner J., Jeleva M., Tallon J. Decision theory under ambiguity. *Journal of Economic Surveys*. Vol. 26 (2). 2012. P. 234-270.

29. Kahneman D., Tversky A. Prospect theory: An analysis of decision under risk. *Handbook of the fundamentals of financial decision making*. Part I. 2013. P. 99-127.

30. Mullainathan S., Thaler R. Behavioral economics. 2000. URL: https://www.nber.org/system/files/working_papers/w7948/w7948.pdf (дата звернення: 20.11.2023).

31. Sheffrin S. Rational expectations. Publisher : Cambridge University Press; 2nd edition. 1996. 200 p.

32. Bossomaier T., Green D. Complex systems. Publisher : Cambridge University Press; Illustrated edition, 2000. 420 p.

33. Рогоза М.Є., Султанахмед К.Р., Мусаєва Е.К. Нелінійні моделі та аналіз складних систем: навчальний посібник. Частина 1. Полтава : РВВ ПУЕТ, 2011. 631 с.

34. Соловйов В.М., Сердюк О.А., Данильчук Г.Б. Моделювання складних систем : навчально-методичний посібник для самостійного вивчення дисципліни; Міністерство освіти і науки України, Криворізький державний педагогічний університет, Черкаський національний університет імені Богдана Хмельницького. Черкаси : Видавець О. Ю. Вовчок, 2016. 204 с.

35. Ottino J. Complex systems. American Institute of Chemical Engineers. *AIChE Journal*. Vol. 49 (2). 2003. P. 292-298.

36. Ladyman J., Lambert J., Wiesner K.. What is a complex system? *European Journal for Philosophy of Science*. Vol. 3, 2013. P. 33-67.

37. Zhou, Zhi-Hua. Machine learning. Published by Tsinghua University Press. Springer Nature, 2021. 455 p.

38. Shrestha Y., Vaibhav K., Krogh G. Augmenting organizational decision-making with deep learning algorithms: Principles, promises, and challenges. *Journal of Business Research.* Vol 1 (23). 2021. P. 588-603.

39. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature.* Vol. 521 (7553). 2015. P. 436-444.

40. Goodfellow I,, Bengio Y., Courville A. *Deep learning.* MIT press, Massachusets Institute of Tehnology. 2016. 86 p.

41. Wang H. Towards Bayesian deep learning: A framework and some existing methods. *IEEE Transactions on Knowledge and Data Engineering.* Vol 28 (12). 2016. P. 3395-3408.

42. Osawa K. Practical deep learning with Bayesian principles. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/b53477c2821c1bf0da5d40e57b870d35-Paper.pdf (дата звернення: 01.11.2023).

43. DeepMind algorithm beats people at classic video games. *Of the 2018 international conference on big data and education.* Vol. 518. Nature 7540. 2018. P. 465-466.

44. Chivers Tom. How to train an all-purpose robot: DeepMind is tackling one of the hardest problems for AI. *IEEE Spectrum.* Vol. 58 (10). 2021. P. 34-41.

45. Li Y. Deep reinforcement learning: An overview. Arxiv preprint. URL: https://arxiv.org/abs/1810.06339 (дата звернення: 15.11.2023).

46. Barto G. Reinforcement learning. *Neural systems for control.* Academic Press, 1997. P. 7-30.

47. François-Lavet V. An introduction to deep reinforcement learning. *Foundations and Trends® in Machine Learning*. Vol. 11 (3-4). 2018. P. 219-354.

48. Holcomb S. Overview on deepmind and its alphago zero ai. *ICBDE.* 2018. P. 67-71

49. Fan J. A theoretical analysis of deep Q-learning. Proceedings of Machine Learning. Vol 120, 2020. URL:

http://proceedings.mlr.press/v120/yang20a/yang20a.pdf (дата звернення: 14.11.2023).

50. Gu Sh. Continuous deep q-learning with model-based acceleration. International conference on machine learning. PMLR, 2016. URL: https://www.researchgate.net/publication/301841627_Continuous_Deep_Q-Learning_with_Model-based_Acceleration (дата звернення: 14.11.2023).

51. Carta S. Ferreira A., Podda A. Multi-DQN: An ensemble of Deep Q-learning agents for stock market forecasting. *Expert systems with applications.* Vol. 164 2021. URL: https://www.researchgate.net/publication/343344950_Multi-DQN_an_Ensemble_of_Deep_Q-Learning_Agents_for_Stock_Market_Forecasting (дата звернення: 14.11.2023).

52. Zhang Z., Zohren S., Stephen R. Deep reinforcement learning for trading. *The Journal of Financial Data Science*. 2020. URl: https://arxiv.org/abs/1911.10107 (дата звернення: 18.11.2023).

53. Deng Y. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*. Vol. 28 (3). 2016. P. 653-664.

54. Moody J., Saffell M. Reinforcement learning for trading. Advances in Neural Information Processing Systems 11. 1998. URL: https://proceedings.neurips.cc/paper_files/paper/1998/hash/4e6cd95227cb0c280e99a195be5f6615-Abstract.html (дата звернення: 19.11.2023).

55. Millea A. Deep reinforcement learning for trading – A critical survey. Data 6.11. 2021. URL: https://www.mdpi.com/2306-5729/6/11/119 (дата звернення: 19.11.2023).

56. Wu X. Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences.* Vol. 538. 2020. P. 142-158.

57. Yang H., Liu X., Zhong S. Deep reinforcement learning for automated stock trading: An ensemble strategy. *Proceedings of the first ACM international conference on AI in finance*. 2020. Article №31. P. 1-8.

58. Liu Y., Liu Q., Zhao H. (2020). Adaptive Quantitative Trading: An

Imitative Deep Reinforcement Learning Approach. *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34 (02). P. 2128-2135.

59. Pricope Tidor-Vlad. Deep reinforcement learning in quantitative algorithmic trading. URL: https://arxiv.org/abs/2106.00123 (дата звернення: 20.11.2023).

60. Dang Q. Reinforcement learning in stock trading. *International conference on computer science, applied mathematics and applications.* Cham:Springer International Publishing, 2019. P. 311-322.

61. Lingze T., Khushi M. Reinforcement learning in financial markets. Vol. 4 (3). 2019. URL: https://www.mdpi.com/2306-5729/4/3/110 (дата звернення: 24.11.2023).

62. Li Y. Reinforcement learning applications. 2019. URL: https://arxiv.org/abs/1908.06973 (дата звернення: 28.11.2023).

63. Wang Q., Zhongli Zh. Reinforcement learning model, algorithms and its application. I*nternational Conference on Mechatronic Science, Electric Engineering and Computer (MEC). IEEE.* 2011. URL: https://ieeexplore.ieee.org/abstract/document/6025669 (дата звернення: 24.11.2023).

64. Ketkar N. Introduction to keras. *Deep learning with python: a hands-on introduction*. 2017. P. 97-111.

65. Feurer M., Hutter F. Hyperparameter optimization. Automated machine learning. URL: https://library.oapen.org/bitstream/handle/20.500.12657/23012/1/1007149.pdf#page=15 (дата звернення: 24.11.2023).

ДОДАТОК А

Програмний код, розроблений у рамках роботи

А.1 Загальний код алгоритму глибокого навчання з обробкою даних

```
import math
import sys
import numpy as np # імпорт бібліотеки для роботи з масивами
import pandas as pd # імпорт бібліотеки для роботи з даними та CSV-файлами

# Вхідні файли даних знаходяться у каталозі "../data/"
# Наприклад, запуск цього скрипту (натиснувши Shift+Enter) виведе файли з каталогу

import os # імпорт бібліотеки для роботи з файлами у системі
print(os.listdir("data"))
print(os.listdir("output"))

# Будь-які результати, записані у поточний каталог, зберігаються як вихідні дані

# Імпорт необхідних модулей та пакетів для машинного навчання
import keras
from keras.models import Sequential
from keras.models import load_model
from keras.layers import Dense
from keras.optimizers import Adam
import random
from collections import deque

class Agent:
```

```
# Клас агента, що навчається

def __init__(self, state_size, total_training_steps, is_eval=False, model_name=""):
    # ініціалізація агента

    self.state_size = state_size # нормалізовані попередні дні
    self.action_size = 3 # сидіти, купувати, продавати
    self.memory = deque (maxlen=10 000) # черга пам'яті довжиною 10 000
    self.inventory = [] # поточний інвентар
    self.model_name = model_name
    self.is_eval = is_eval
    self.total_training_steps = total_training_steps
    self.current_training_step = 0

    self.gamma = 0.95
    self.epsilon = 1.0
    self.epsilon_min = 0.01
    self.epsilon_decay = 0.995

    self.model = load_model("model_list/" + model_name) if is_eval else self._model()

def _model(self):
        # визначення моделі нейромережі
    model = Sequential()
    model.add(Dense(units=1024, input_dim=self.state_size, activation="relu"))
    model.add(Dense(units=512, activation="relu"))
    model.add(Dense(units=512, activation="relu"))
    model.add(Dense(units=128, activation="relu"))
    model.add(Dense(units=64, activation="relu"))
    model.add(Dense(units=32, activation="relu"))
    model.add(Dense(self.action_size, activation="linear"))
    model.compile(loss="mse", optimizer=Adam(lr=0.001))

    return model
```

```
    def act(self, state):
      # вибір дії на основі стану
      if not self.is_eval and random.random() <= self.epsilon:
        return random.randrange(self.action_size)

      options = self.model.predict(state)
      return np.argmax(options[0])

    def experience_replay(self, batch_size):
      # тренування моделі на основі пам'яті
      mini_batch = []
      l = len(self.memory)
      for i in range(l - batch_size + 1, l):
        mini_batch.append(self.memory[i])

      for state, action, reward, next_state, done in mini_batch:
        # оновлення Q-функції
        target = reward
        if not done:
          target = reward + self.gamma *
np.amax(self.model.predict(next_state)[0])

        target_f = self.model.predict(state)
        target_f[0][action] = target
        self.model.fit(state, target_f, epochs=1, verbose=0)
        self.current_training_step += 1
        progress = (self.current_training_step / self.total_training_steps) * 100
      #  print(f"Тренувальний прогрес: {progress:.2f}%")

      if self.epsilon > self.epsilon_min:
        self.epsilon *= self.epsilon_decay

# форматування ціни
def price_formatting(n):
      return ("-$" if n < 0 else "$") + "{0:.2f}".format(abs(n))

# отримання вектора даних акцій з файлу
def stock_data(key):
      vec = []
```

```
        lines = open("data/GSPC.csv", "r").read().splitlines()

        for line in lines[1:]:
            vec.append(float(line.split(",")[4]))

        return vec

    # сигмоїда
    def sigmoid_function(x):
        return 1 / (1 + math.exp(-x))

    # отримання стану на основі n попередніх днів на момент часу t
    def return_state(data, t, n):
        d = t - n + 1
        block = data[d:t + 1] if d >= 0 else -d * [data[0]] + data[0:t + 1] # заповнення
        res = []
        for i in range(n - 1):
            res.append(sigmoid(block[i + 1] - block[i]))

        return np.array([res])

    stock_name, window_size, episode_count = '^GSPC', 12, 1

    data = stock_data(stock_name)
    l = len(data) - 1
    batch_size = 32
    total_training_steps = episode_count * l
    agent = Agent(window_size, total_training_steps)

    for e in range(episode_count + 1):
        # тренування агента
      print("Епізод " + str(e) + "/" + str(episode_count))
      state = return_state(data, 0, window_size + 1)

      total_profit = 0
      agent.inventory = []

      it=0
```

```
        for t in range(l):
          action = agent.act(state)

          # сидіти
          next_state = getState(data, t + 1, window_size + 1)
          reward = 0

          it=it+1
          print(it)

          if action == 1: # купити
            agent.inventory.append(data[t])
            print("Купити: " + formatPrice(data[t]))

          elif action == 2 and len(agent.inventory) > 0: # продати
            bought_price = agent.inventory.pop(0)
            reward = max(data[t] - bought_price, 0)
            total_profit += data[t] - bought_price
            print("Продати: " + formatPrice(data[t]) + " | Прибуток: " +
price_formatting(data[t] - bought_price))

          done = True if t == l - 1 else False
          agent.memory.append((state, action, reward, next_state, done))
          state = next_state

          if done:
            print("--------------------------------")
            print("Загальний прибуток: " + price_formatting(total_profit))
            print("--------------------------------")

          if len(agent.memory) > batch_size:
            agent.experience_replay(batch_size)

        # збереження моделі
        if e % 10 == 0:
          agent.model.save("output/model_ep" + str(e))

      print("--------------------------------")
```

```
print("Загальний прибуток: " + price_formatting(total_profit))
print("-------------------------------")

agent.model.save("output/model_step" + str(e))
```

А.2 Код для побудови різних варіантів нейромереж з варіативними гіперпараметрами

Додання дропауту

```
def _model(self):
  model = Sequential()
  model.add(Dense(units=1024, input_dim=self.state_size,
activation="relu"))
  model.add(Dropout(0.2))
  model.add(Dense(units=512, activation="relu"))
  model.add(Dropout(0.2))
  model.add(Dense(units=256, activation="relu"))
  model.add(Dense(self.action_size, activation="linear"))
  model.compile(loss="mse", optimizer=Adam(lr=0.001))
  return model
```

Зміна функції активації та learning rate

```
def _model(self):
  model = Sequential()
  model.add(Dense(units=1024, input_dim=self.state_size,
activation="tanh"))
  model.add(Dense(units=512, activation="tanh"))
  model.add(Dense(units=256, activation="tanh"))
  model.add(Dense(self.action_size, activation="sigmoid"))
  model.compile(loss="mse", optimizer=Adam(lr=0.0005))
  return model
```

Зміна глибини нейромережі

```
def _model(self):
  model = Sequential()
```

```
    model.add(Dense(units=2048, input_dim=self.state_size,
activation="relu"))
    model.add(Dense(units=1024, activation="relu"))
    model.add(Dense(units=512, activation="relu"))
    model.add(Dense(units=128, activation="relu"))
    model.add(Dense(units=64, activation="relu"))
    model.add(Dense(self.action_size, activation="softmax"))
    model.compile(loss="mse", optimizer=Adam(lr=0.002))
    return model
```

Використання спрощеної архітектури

```
def _model(self):
    model = Sequential()
    model.add(Dense(units=256,                    input_dim=self.state_size,
activation="relu"))
    model.add(Dense(units=128, activation="relu"))
    model.add(Dense(self.action_size, activation="linear"))
    model.compile(loss="mse", optimizer=Adam(lr=0.01))
    return model
```

Додавання нестандартних функцій активації та регуляризації

```
def _model(self):
    model = Sequential()
    model.add(Dense(units=1024,                   input_dim=self.state_size,
activation="elu"))
    model.add(Dense(units=512, activation="elu"))
    model.add(Dense(units=256, activation="elu"))
    model.add(Dense(units=128, activation="elu"))
    model.add(Dense(self.action_size, activation="linear"))
    model.compile(loss="mse", optimizer=Adam(lr=0.0001))
    return model
```

Додавання більшої кількості нейронів

```
def _model(self):
    model = Sequential()
    model.add(Dense(units=2048,                   input_dim=self.state_size,
```

```
activation="relu"))
        model.add(Dense(units=2048, activation="relu"))
        model.add(Dense(units=1024, activation="relu"))
        model.add(Dense(units=512, activation="relu"))
        model.add(Dense(units=256, activation="relu"))
        model.add(Dense(self.action_size, activation="linear"))
        model.compile(loss="mse", optimizer=Adam(lr=0.0005))
        return model
```

Додавання batch normalisation

```
    def _model(self):
        model = Sequential()
        model.add(Dense(units=1024,                    input_dim=self.state_size,
activation="relu"))
        model.add(BatchNormalization())
        model.add(Dense(units=512, activation="relu"))
        model.add(BatchNormalization())
        model.add(Dense(units=256, activation="relu"))
        model.add(BatchNormalization())
        model.add(Dense(self.action_size, activation="linear"))
        model.compile(loss="mse", optimizer=Adam(lr=0.001))
        return model
```

Використання різних типів шарів:

```
    def _model(self):
        model = Sequential()
        model.add(Dense(units=512,                     input_dim=self.state_size,
activation="relu"))
        model.add(Dense(units=512, activation="relu"))
        model.add(LSTM(units=256, return_sequences=True))
        model.add(LSTM(units=256))
        model.add(Dense(self.action_size, activation="linear"))
        model.compile(loss="mse", optimizer=Adam(lr=0.001))
        return model
```

Використання residual connection

```
from keras.layers import add, Input, Dense
from keras.models import Model

def _model(self):
  inputs = Input(shape=(self.state_size,))
  x = Dense(units=512, activation="relu")(inputs)
  y = Dense(units=512, activation="relu")(x)
  y = add([y, x]) # Residual connection
  y = Dense(units=256, activation="relu")(y)
  predictions = Dense(self.action_size, activation="linear")(y)
  model = Model(inputs=inputs, outputs=predictions)
  model.compile(loss="mse", optimizer=Adam(lr=0.001))
  return model
```

Додавання згорткових шарів

```
def _model(self):
  model = Sequential()
  model.add(Conv1D(filters=64, kernel_size=3, activation="relu",
input_shape=(self.state_size, 1)))
  model.add(MaxPooling1D(pool_size=2))
  model.add(Flatten())
  model.add(Dense(128, activation="relu"))
  model.add(Dense(self.action_size, activation="linear"))
  model.compile(loss="mse", optimizer=Adam(lr=0.001))
  return model
```

Використання шарів RNN

```
def _model(self):
  model = Sequential()
  model.add(SimpleRNN(units=256, input_shape=(self.state_size, 1),
activation="relu"))
  model.add(Dense(units=128, activation="relu"))
  model.add(Dense(units=64, activation="relu"))
  model.add(Dense(self.action_size, activation="linear"))
  model.compile(loss="mse", optimizer=Adam(lr=0.001))
  return model
```