УДК 004.8

А. М. Тарновський; С. М. Захарченко, канд. техн. наук, проф.

СУЧАСНІ ТЕХНОЛОГІЇ З РЕАЛІЗАЦІЇ ІНТЕЛЕКТУАЛЬНИХ АГЕНТІВ З ВИКОРИСТАННЯМ 3D-СИМУЛЯТОРІВ

У роботі проведено комплексний огляд сучасних методів і технологій створення інтелектуальних агентів, здатних автономно діяти у віртуальних тривимірних середовищах. Розглянуто основні підходи машинного навчання, зокрема навчання з підкріпленням (RL) та його глибокі модифікації (DRL), що забезпечують формування адаптивної поведінки агентів у складних сценаріях і динамічних середовищах. Детально проаналізовано алгоритми PPO та A3C, а також методи з участю людини (RLHF, ReQueST), які дозволяють поєднати ефективність нейронних мереж із експертною оцінкою, підвищуючи безпеку і стабільність навчання. Значну увагу приділено сучасним симуляційним платформам (Unity ML-Agents, Habitat, AI2-THOR, Webots, CoppeliaSim), які забезпечують масштабоване, безпечне та контрольоване середовище для навчання, перевірки й тестування моделей у режимі, наближеному до реальності. Окремо розглянуто роль трансформерних архітектур і великих мовних та мультимодальних моделей (LLM, LMM), що відкривають можливості для побудови гнучких агентів із контекстно-залежною поведінкою, здатних інтегрувати обробку мовних, візуальних і просторових даних для прийняття рішень. Проаналізовано сучасні підходи до роботи з 3D-поданнями, включаючи хмари точок та воксельні сітки, а також ефективність архітектур PointNet/PointNet++ і VoxelNet для обробки просторової інформації з високою точністю. Наведено перспективи розвитку поєднання тривимірних симуляторів із мультимодальними моделями для досягнення більшої узгодженості між віртуальними й реальними середовищами, підвищення точності виконання завдань, оптимізації ресурсів і покращення переносимості навичок у практичні застосування. Огляд демонструє, що поєднання сучасних нейронних архітектур, потужних симуляторів і методів оптимізації навчання формує основу для створення автономних систем нового покоління, здатних ефективно адаптуватися до складних, непередбачуваних і швидкозмінних умов.

Ключові слова: *інтелектуальний агент, навчання з підкріпленням, глибоке навчання, глибока нейронна мережа, трансформерна архітектура, великі мультимодальні моделі, PointNet/PointNet++, VoxelNet, 3D-симулятор.*

Вступ

Штучний інтелект (ШІ) та нейронні мережі стали однією з найпопулярніших тем, що широко обговорюються останнім часом. Хоча часто ці терміни вживаються як тотожні, насправді вони не є взаємозамінними. Нейронні мережі належать до методів машинного навчання, які є однією зі складових штучного інтелекту [1]. У той час як традиційні методи штучного інтелекту засновані на правилах «якщо-то», машинне навчання спрямоване на ітеративне знаходження відповідності між наборами даних без необхідності явно кодувати будь-які правила [1], [2].

Сьогодні ми стаємо свідками нової фази розвитку штучного інтелекту, що характеризується переходом від спеціалізованих систем, призначених для вирішення вузьких завдань, до все більш складних архітектур, здатних діяти або приймати рішення автономно. Особливо значущим досягненням у цьому напрямі стали ШІ-агенти [3]. На відміну від традиційних систем ШІ, інтелектуальні агенти мають здатність автономно сприймати, приймати рішення та діяти, адаптуючи свою поведінку до змін у навколишньому середовищі та накопиченого досвіду. Унікальність агентів полягає в їх здатності постійно вдосконалюватись завдяки машинному навчанню.

Обчислювальні потужності сучасних комп'ютерів надають можливість використовувати для навчання агентів віртуальні тривимірні середовища, які імітують реальні або вигадані фізичні умови. Це створює інструментарій для експериментування, перевірки гіпотез, аналізу сценаріїв у ситуаціях, які є складними або неможливими для реалізації чи спостереження у дійсності. До того ж такий підхід є гнучким та добре масштабованим, вимагає менших

фінансових витрат тощо. Саме тому стратегія "симуляція перед реальністю" набуває усе більшого поширення у навчанні та дослідженні агентів.

Метою роботи є огляд та узагальнення сучасних методів та технологій в області нейронних мереж та 3D-симуляторів, що можуть бути використані для створення інтелектуальних агентів, здатних автономно діяти у тривимірній реальності.

Методи машинного навчання інтелектуальних агентів

Одним з основних методів машинного навчання інтелектуальних агентів є навчання з підкріпленням (Reinforcement Learning, RL) – метод навчання на основі спроб та помилок, у процесі якого агент навчається оптимальній поведінці через взаємодію з навколишнім середовищем, отримуючи зворотній зв'язок у формі винагороджень або штрафів та адаптуючи свої дії для максимізації винагородження у довгостроковій перспективі [4], [5]. Навчання з підкріпленням застосовується для конкретної задачі, для якої немає єдиної вірної відповіді, проте є бажаний кінцевий результат [6]. Воно є універсальним засобом для прийняття рішень у багатьох випадках, де агент знаходиться в ситуації, коли він має вибір дій [5]. Це, можливо, найбільш наближений до притаманного людині спосіб навчання.

В основі RL лежить Марковський процес прийняття рішень [7]: результат кожної дії частково залежить від попередніх дій та поточного стану, а частково є випадковим. Вибір дії, яку треба зробити далі для отримання максимального сукупного винагородження, робиться через вибір між подальшим дослідженням середовища для отримання нових винагороджень за дії у цьому стані та вибору серед відомих для цього стану дій з високим винагородженням. Таким чином, агент має дотримуватися балансу між використанням досвіду та дослідженням невідомих станів. Вірний баланс приведе агента до знаходження оптимальної політики, що дає максимальну винагороду. Політикою є стратегія, якої дотримується агент під час вибору дій у кожному стані. Якщо агент продовжить використовувати лише минулий досвід, він з високою імовірністю виробить неоптимальну політику. З іншого боку, якщо агент продовжить дослідження без використання досвіду, він може ніколи не знайти гарну політику.

Подальшим розвитком RL є глибоке навчання з підкріпленням (Deep Reinforcement Learning, DRL). В основі DRL лежать дві основні концепції: навчання з підкріпленням, яке фокусується на вивченні оптимальної політики через взаємодію з навколишнім середовищем, та глибоке навчання, яке використовує глибокі штучні нейронні мережі для узагальнення та подання складних закономірностей чи відносин у даних [4], [5]. Глибока нейронна мережа (Deep Neural Network, DNN) (багатошарова нейронна мережа з кількома прихованими шарами) використовується для апроксимації функцій, необхідних для навчання, таких як функція цінності (передбачення майбутніх винагород) або політика. Глибока нейронна мережа надає можливість агентам навчатися на складних, багаторозмірних даних, таких як, наприклад, зображення або показання сенсорів [4].

Серед різних методів DRL слід відзначити алгоритми Proximal Policy Optimization (PPO) та Asynchronous Advantage Actor-Critic (A3C), що сьогодні набули широкої популярності. Алгоритм PPO був запропонований у 2017 році John Schulman зі співавторами з Open AI [8]. Алгоритм заснований на обмеженні оновлення політики для того, щоб гарантувати, що нова політика не буде суттєво відхилятися від старої. Це запобігає різкій зміні стратегії агенту, забезпечуючи гарний баланс між стабільністю навчання та простотою реалізації. Перевагами PPO є стабільність процесу навчання, висока продуктивність та масштабованість. Він часто використовується у складних візуальних середовищах, де важлива надійність [9].

Алгоритм A3C був запропонований у 2016 році вченими з Google Deep Mind на чолі з Volodymyr Mnih [10]. Особливістю алгоритму є те, що він одночасно здійснює оптимізацію політики, відому як крок актора, та оцінювання політики, відому як крок критика. Актор відповідає за вибір дій, які агент повинен виконати у цьому стані оточуючого середовища, тобто формує політику поведінки. Критик оцінює, наскільки вдала чи невдала вибрана дія

актора з точки зору досягнення кінцевої мети. Алгоритм A3C працює з використанням кількох асинхронних потоків агентів зі спільною політикою. Таких підхід дозволяє ефективніше досліджувати середовище і прискорити навчання агентів [11]. Алгоритми PPO та A3C добре масштабуються на великі сценарії, що містять тисячі варіантів взаємодій.

Деякі підходи передбачають залучення людини у процес навчання (Human-In-The-Loop, HITL), що дозволяє поєднувати переваги навчання з підкріпленням з людською інтуїцією та оцінкою ризиків. Одним з таких методів є навчання з підкріпленням на основі зворотного зв'язку з людиною (Reinforcement Learning From Human Feedback, RLHF) [12], при якому зворотній зв'язок з людиною використовується або для формування значення винагородження, або для оцінювання політики (наскільки обрана дія краща або гірша порівняно з поточною очікуваною поведінкою). Таким чином RLHF використовує людський фактор для покращення здатності агентів оцінювати складні ситуації, дозволяючи їм адаптуватися та оптимізуватися на основі складності реальних умов, а не покладатися виключно на статичні дані навчання. Цей метод навчання ідеально підходить для завдань зі складними, нечітко визначеними або важко визначеними цілями. RLHF відіграв ключову роль у розвитку генеративних моделей штучного інтелекту [12].

В умовах, коли агент повинен уникати небезпечної поведінки як підчас навчання, так і під час розгортання може бути використана модель навчання ReQueST (Reinforcement Learning via Questionable State Tweaks) [13]. За класичного RLHF для вивчення небезпечних станів агенту спочатку необхідно побувати в цих станах, щоб користувач міг надати зворотний зв'язок. За відсутності відповідного симулятора це ускладнює або унеможливлює навчання у випадках критичних систем, де помилка агента може мати значні наслідки, наприклад, у медичних застосуваннях, безпілотному управлінні або технічному обслуговуванні складного обладнання. На відміну від класичного RLHF ReQueST працює у два етапи. Спочатку агенту пропонується досліджувати широкий спектр станів, основуючись на штучно згенерованій гіпотетичній поведінці. Користувач надає зворотний зв'язок з цієї гіпотетичної поведінки для вивчення агентом моделі винагородження. Після того, як агент успішно навчиться передбачати винагороди та небезпечні стани, відбувається навчання з підкріпленням для оптимізації сформованої на попередньому етапі функції винагородження. В [14] було показано, що використання ReQueST для навчання агентів у 3D-середовищах дозволяє на порядок скоротити випадки небезпечної поведінки порівняно з традиційним RL.

За обмеженої кількості навчальних вибірок використовуються методи (Few-ShotLearning, FSL) або (Zero-ShotLearning, ZSL) [15] – парадигм, за яких агент здатен виконувати нові завдання або адаптувати поведінку з мінімальним (FSL) або взагалі без (ZSL) попереднього навчання на цільовому середовищі. Це досягається за рахунок переносу попередньо накопичених узагальнених знань на нові контексти. Обидві парадигми дозволяють агентам навчатися та узагальнювати знання на основі дуже обмеженої кількості інформації, що є ключовим для застосування штучного інтелекту в реальних сценаріях, де збирання великих наборів даних неможливе або недоцільне. Методи FSL та ZSL широко використовуються у задачах розпізнавання об'єктів та класифікації зображень.

Симуляційні платформи

Реалізація навчання у віртуальних просторах неможлива без відповідного середовища моделювання – симулятора, що забезпечує змодельовану дійсність, зокрема фізику, візуальні елементи, логіку взаємодій та зворотний зв'язок для агентів. Різноманіття доступних симуляторів дозволяє моделювати широкий спектр сценаріїв: від простих задач навігації до складної маніпуляції об'єктами у віртуальному середовищі. Як приклади подібних симуляторів можна відзначити Unity ML-Agents, Habitat та AI2-THOR, які часто згадуються у різних публікаціях.

Unity ML-Agents є плагіном для ігрового двигуна Unity, що дозволяє створювати візуально насичені сцени з фізикою об'єктів та гнучким налаштуванням винагороди для

навчання агентів при моделюванні поведінки ігрових персонажів [16]. Платформа моделювання Habitat є ефективним 3D-симулятором, який підтримує фотореалістичні 3D-моделі приміщень для навчання домашніх роботів-помічників, моделювання їх поведінки та проведення експериментів з ними [17], [18]. Аналогічно Habitat, AI2-THOR також є інтерактивним середовищем, яке імітує домашній простір з його фізикою та об'єктами, з якими можна взаємодіяти для навчання вирішенню побутових задач, таких як прибирання, пошук предметів, взаємодія з інтер'єром [18], [19]. Завдяки реалістичності платформи Habitat та AI2-THOR дозволяють точно переносити у реальний світ навички, закріплені у симуляції.

Однією з проблем симуляторів є труднощі масштабування кількості сцен до великих значень (тисяч або десятків тисяч), оскільки сцени, як правило, створюються або вручну, або отримуються за допомогою 3D-сканування реальних структур. Подолати ці труднощі дозволяє фреймворк ProcTHOR [18] для AI2-THOR, що розширює можливості AI2-THOR за рахунок підтримки процедурної генерації інтерактивних приміщень для навчання (рис. 1).

Оскільки змодельовані середовища забезпечують різну ступінь реалізму, у реальному світі агент не завжди буде діяти так само, як він це робив під час навчання. Для більш точного моделювання фізики можуть бути використані такі симулятори, як, наприклад, Webots або CoppeliaSim, які є потужними інструментами для розробки та перевірки різноманітних застосунків робототехніки. Webots [20] – це open-source платформа, яка надає широкі можливості для створення віртуальних роботів, моделювання сенсорів, маніпуляторів і фізичної взаємодії з об'єктами. Її застосовують у дослідженнях та тестуванні автономних систем. В [21] запропонований фреймворк, названий Deepbots, що поєднує бібліотеку для розробки та тестування алгоритмів навчання з підкріпленням OpenAI Gym з симулятором Webots для надання можливості використання DRL в реальних сценаріях робототехніки.

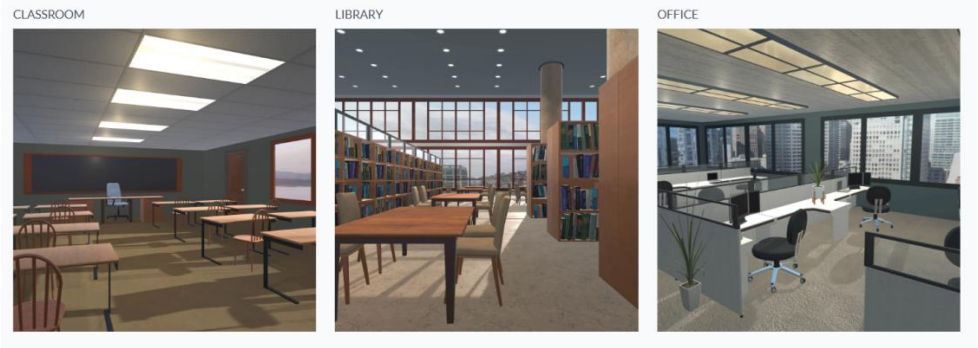

Рис. 1. Приклади середовищ для навчання згенерованих ProcTHOR

CoppeliaSim (раніше відомий як V-REP) є кросплатформовим програмним симулятором з розподіленою архітектурою управління та інтегрованим середовищем розробки, що надає інструменти для ефективного проєктування та моделювання роботів [22]. CoppeliaSim пропонує користувачам багату бібліотеку моделей роботів та комплексних фізичних двигунів, що дозволяє створювати, налаштовувати та програмувати різні моделі роботів, а також взаємодіяти з ними у реальному часі. Симулятор CoppeliaSim може бути використаний як середовище для навчання агентів із застосуванням DRL та оцінювання їх поведінки. Приклади інтегрування DRL з CoppeliaSim розглядаються в [23] та [24].

Прикладами середовищ для навчання та дослідження мультимодальних агентів (агентів, здатних обробляти текст, зображень, аудіо та відео) є симулятори TEACh (Task-driven Embodied Agents via Conversational Human feedback) та BEHAVIOR (Benchmark for Everyday Household Activities in Virtual, Interactive, and ecOlogical enviRonments). У TEACh [25] агент навчається, отримуючи діалогові інструкції на природній мові, та поступово уточнює свою поведінку відповідно до зворотного зв'язку від людини. BEHAVIOR [26], у свою чергу, надає багатий набір завдань побутового характеру з високою деталізацією середовища, де агент виконує інструкції, маніпулює об'єктами і формує текстовий звіт. У таких умовах

алгоритми з підкріпленням поєднуються з послідовними моделями типу sequence-to-sequence (seq2seq), які перекладають текстові інструкції у послідовність дій. Seq2seq-моделі, що походять з обробки природної мови, використовують архітектуру кодер-декодер і забезпечують трансформацію вхідної послідовності (текстової команди) у відповідний план дій у просторі.

Нейромережеві архітектури для навчання на візуальних даних

Для обробки візуальної інформації, яку сприймає агент під час взаємодії з середовищем, можуть застосовуватися згорткові нейронні мережі (Convolutional Neural Networks, CNN). Вони дозволяють виділяти візуальні ознаки і формувати узагальнене представлення сцени [27]. У складніших сценаріях, де потрібно враховувати історію взаємодій, використовуються рекурентні нейронні мережі (Recurrent Neural Networks, RNN).

RNN здатні зберігати внутрішній стан, що дає змогу агенту враховувати попередні спостереження під час прийняття рішень. Проте RNN запам'ятовують тільки самі останні вхідні дані, а тому мають обмеження у ситуаціях, коли для поточного рішення важливо враховувати події, що стались багато кроків тому [28]. Ці обмеження проявляються у вигляді затухання або вибуху градієнтів під час навчання на довгих послідовностях: затухання градієнту характеризує втрату можливості навчатися на наявних даних, вибух градієнту – перенавчання, коли є гарна адаптація до навчальних даних, але спостерігається втрата здатності узагальнювати закономірності у них.

Для подолання проблеми затухання градієнту Sepp Hochreiter та Jürgen Schmidhuber запропонували архітектуру нейронної мережі з довгою короткостроковою пам'яттю (Long Short-Term Memory, LSTM) [29]. LSTM є рекурентною нейромережею з прихованим шаром, що утворюється комірками пам'яті зі шлюзами"вхід", "забуття" та "вихід", які надають можливість вирішувати яку нову інформацію запам'ятати, яку інформацію залишити чи відкинути, яку передати далі. Це дозволяє зберігати корисну інформацію протягом тривалого часу і, як наслідок, ефективно опрацьовувати довгі послідовності. Наприклад, у системі ProcTHOR агент може зберігати інформацію про вже відвідані кімнати, що покращує ефективність навігації.

Іншим рішенням до подолання обмежень, характерних для RNN, стали трансформери (Transformer). Трансформерна архітектура була вперше представлена Ashish Vaswani зі співавторами [30] як модель без рекурентності для обробки логічно зв'язаних послідовностей даних, перш за все таких як текст. Основною інновацією стало використання механізму self-attention, який дозволяє моделі звертати увагу на всі елементи вхідної послідовності незалежно від їхнього порядку, що критично важливо для довготривалих залежностей та паралельної обробки даних. Однією з трансформерних архітектур є Decision Transformer [31]. Ця нейромережева архітектура інтерпретує завдання навчання з підкріпленням як задачу послідовного передбачення: агент отримує як вхідні дані історію станів, дій та відповідних винагород і вчиться генерувати наступну дію аналогічно до задач мовного моделювання.

На відміну від класичних RL-алгоритмів, трансформери не вимагають явно заданої функції значущості або складного механізму обчислення Q-функції (функції, що визначає очікувану цінність виконання дії у даному стані), а навпаки – оперують з історією у вигляді контексту. Завдяки цьому такі моделі демонструють гарну здатність до узагальнення і можуть працювати в режимі offline RL, тобто навчатись на фіксованих наборах даних без взаємодії із середовищем у реальному часі. Це особливо корисно в умовах, де симуляція є високовартісною або доступ до середовища обмежений.

Трансформерна архітектура лежить в основі великих мовних моделей (Large Language Models, LLM). LLM є типом глибоких нейронних мереж, що попередньо навчені на величезних об'ємах даних для розуміння, узагальнення та прогнозування нового контенту. LLM відіграли вирішальну роль у задачах обробки природньої мови [32], [33]. Найчастіше

вони реалізуються як діалоговий агент, з яким можна спілкуватися у розмовній формі у форматі питання-відповідь. Особливістю LLM є те, що вони не вимагають даних для навчання, специфічних для конкретної задачі, і можуть узагальнювати інформацію, на якій вони були навчені, для виконання широкого спектру завдань. Найбільш відомою LLM є GPT (Generative Pre-trained Transformer) від OpenAI, що налаштована за допомогою RLHF на виконання інструкцій в режимі діалогу [34].

Агенти на основі LLM зазвичай складаються з чотирьох ключових компонентів: планування, пам'яті, сприйняття та дії [33]. Планування та пам'ять складають основу «мозку» агента, керованого LLM, який взаємодіє з навколишнім середовищем через компоненти сприйняття та дії для досягнення поставленої мети. Планування забезпечує розбиття складних завдань на підзавдання та визначає їх послідовність для досягнення кінцевої мети. Ця послідовність може коригуватися на основі зворотного зв'язку від середовища або людини. Пам'ять дозволяє зберігати історію дій, рішень та спостережень агента, що дозволяє агенту використати попередній досвід для більш ефективного вирішення задач. Сприйняття забезпечує отримання інформації у текстовій, візуальній або аудіо формах для оптимізації планування. Дія відповідає за виконання конкретних дій у взаємодії із зовнішнім середовищем з врахуванням плану.

Одним з прикладів реалізації LLM є PORTAL – фреймворк, в якому агент формує свої дії, ґрунтуючись на текстовому описі завдання, без класичної функції винагороди [35]. Така модель дозволяє агенту інтерпретувати семантику завдання в умовах тисячі різних сценаріїв. Крім цього, LLM також використовуються для генерації самих сцен, об'єктів і навіть текстових описів середовища, що суттєво полегшує масштабування симуляції та створення нових навчальних середовищ [36]. Архітектуру фреймворку PORTAL, що демонструє інтеграцію генерації дерева поведінки на основі LLM через взаємодію з середовищем, а також реакцією на зміни в ньому, зображено на рис. 2.

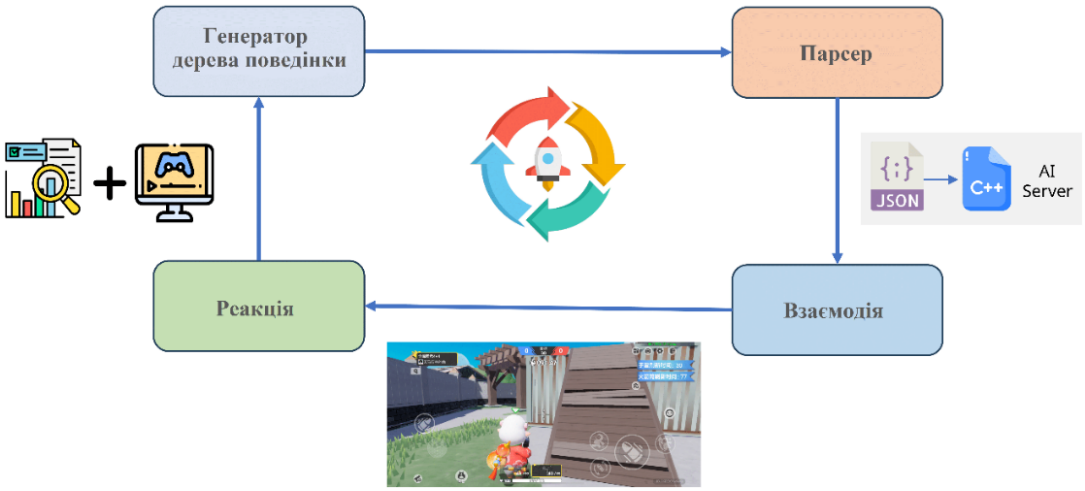

Рис. 2. Архітектура фреймворку PORTAL

Останні досягнення в області LLM призвели до розробки великих мультимодальних моделей (Large Multimodal Models, LMM), які дозволяють вирішувати задачі обробки та аналізу даних різних модальностей – тексту, зображень, аудіо та відео. Структурно LMM зазвичай складаються з п'яти компонентів [37]: кодувальники (кодери) модальностей (зображення, відео, аудіо тощо), проектори вхідних даних для вирівнювання подання різних модальностей, основа у вигляді LLM для семантичного розуміння та міркувань, проектори вихідних даних для отримання інструкцій щодо генерації контенту, генератори контенту різних модальностей (декодери).

Сучасні мультимодальні агенти, які поєднують обробку зображень, просторових даних та природної мови, демонструють потенціал до реалізації більш гнучкої, контекстно-залежної

поведінки. На відміну від традиційних агентів, такі системи не обмежуються лише сенсорною інформацією, а здатні інтерпретувати завдання, формулювання і запити у вербальній формі. Зокрема, агенти можуть не лише виконувати конкретні дії, але й пояснювати власний план, адаптуватися до змін або коментувати процес навчання. Такі підходи активно досліджуються в межах напрямку embodied AI, де інтелектуальний агент поєднує сенсорно-моторну систему з мовною та планувальною складовими.

Для реалізації таких можливостей широко використовуються заздалегідь навчені візуальні кодери. Масштабні LLM, інтегровані з такими кодерами, дозволяють агентам кодувати логіку дій на основі узагальнених знань. Віртуальні 3D-простори, у свою чергу, виступають як інтерактивне середовище для узгодження логіки, сенсорики та динаміки, де zero-shot агенти можуть проявляти когнітивну гнучкість і швидко адаптуватися до нових завдань.

Іншим підходом є використання CLIP-подібних архітектур. CLIP (Contrastive Language–Image Pre-training) є нейромережевою моделлю, розробленою OpenAI. Вона складається з двох основних компонентів:кодувальника зображень та кодувальника тексту. Як перший зазвичай використовується згорткова нейронна мережа Другий реалізується на основі трансформера. CLIP навчається одночасно на зображеннях і текстах, формуючи спільний простір ознак [38]. Завдяки цьому агент здатен знаходити відповідності між візуальним контекстом та мовними інструкціями, що критично важливо для zero-shot дій без прямого тренування на конкретному середовищі [38], [39].

Не зважаючи на те, що сьогодні досягнений значний прогрес у використанні згорткових мереж та методів глибокого навчання для обробки зображень, для вирішення задачі розуміння тривимірних сцен останнім часом замість методів, що використовують набори зображень, отриманих з різних ракурсів, більшого поширення набувають методи, що напряму працюють з тривимірними представленнями об'єктів реального чи віртуального світу, такими як хмара точок та воксельна модель.

Хмарою точок (Points Cloud) [40] є набір точок у просторі, що подають тривимірні об'єкти або поверхні. Кожна точка визначається тривимірними координатами. Крім того, для точок можуть задаватися додаткові параметри, такі як яскравість або колір. Більшість методів тривимірного сканування отримують саме хмару точок.

Воксельна (від англ. Volumetric Pixel – об'ємний піксель) модель або воксельна сітка (voxel grid) – підхід, при якому тривимірний об'єкт вбудовується у тривимірну матрицю (сітку). Весь тривимірний простір представляється як рівномірна матриця, комірки якої є тривимірними об'ємними пікселями (вокселями). Воксель можна розглядати як тривимірну базову кубічну одиницю, яка може бути використана для представлення 3D моделей. Розташування вокселя визначається не координатами у просторі, а його позицією у матриці. Воксель також може мати ряд атрибутів, наприклад, колір. Воксельна сітка може отримуватися з хмари точок [40], [41].

Першою нейронною мережею для розпізнавання хмари точок стала PointNet [42], що працювала з хмарами точок без будь-якої попередньої обробки. PointNet безпосередньо приймає хмару точок як вихідні дані і повертає або мітки класів для всього набору вихідних даних, або мітки сегментів (частин) для кожної точки вихідних даних. Це досить проста архітектура, що використовує геометрію хмари, оскільки координати точок входять у вектори ознак, проте працює не з локальною околицею точки, а з усією хмарою у цілому. Розвитком мережі PointNet стала PointNet++ [43], яка вдосконалила PointNet за рахунок застосування ієрархічного навчання, що дозволило мережі захоплювати локальні структури у різних масштабах. Цей підхід значно покращив здатність обробляти різні щільності точок та захоплювати дрібні геометричні деталі [40], [44]. PointNet++ має широку сферу застосувань і використовується в класифікації 3D-об'єктів, сегментації 3D-деталей та 3D-сцен. Після значного успіху мереж PointNet та PointNet++ було запропоновано ряд подібних архітектур [44].

Зазвичай точки у хмарі розташовані нерівномірно, а тому їх щільність не є однаковою і залежить від місяця у хмарі. Це значно підвищує трудомісткість обчислень за допомогою

PointNet++. Щоб вирішити цю проблему в [45] була запропонована структура мережі Octree-Grouping-PointNet++ (OG-PointNet++), що здійснює групування точок хмари відповідно до їх щільності з використанням незбалансованого октодерева (деревоподібна структура, кожен вузол якої має вісім нащадків): кожен вузол в октодереві охоплює кількість точок, що є сумою тих, що охоплені усіма його дочірніми вузлами. Вузли (групи) з максимальною глибиною розташовані більш щільно і їх особливості мають бути витягнуті першими. Тому дані точок цих вузлів вводяться на перший шар мережі PointNet++. Локальні ознаки, витягнуті з них, використовуються як атрибути батьківського вузла. Дані точок цих батьківських вузлів та вузлів з такою самою глибиною, як у них, вводяться на наступний шар PointNet++ для навчання та отримання ознак і так далі, поки дані точок дочірніх вузлів кореневого вузла (вузлів з глибиною 1), не будуть введені для обробки на останній шар PointNet++. Вилучені ознаки використовуються для класифікації та сегментації.

Іншим підходом до вирішення проблеми складності обчислень через невпорядкованість точок у хмарах є використання вокселів. Методи на основі вокселів перетворюють дані хмар точок на набір рівновіддалених тривимірних вокселів, що усуває невпорядкованість [41], [46]. За такого підходу область простору, що цікавить, охоплюється прямокутним паралелепіпедом. Далі цей паралелепіпед розбивається на значно менші прямокутні паралелепіпеди однакового розміру – вокселі (рис. 3). У результаті всередині кожного вокселя буде знаходитися деякий набір точок. Точки всередині кожного вокселя перетворюються на один вектор ознак, наприклад, з використанням PointNet++. Після цього можна застосувати звичайні згорткові мережі та вирішувати будь-які завдання.

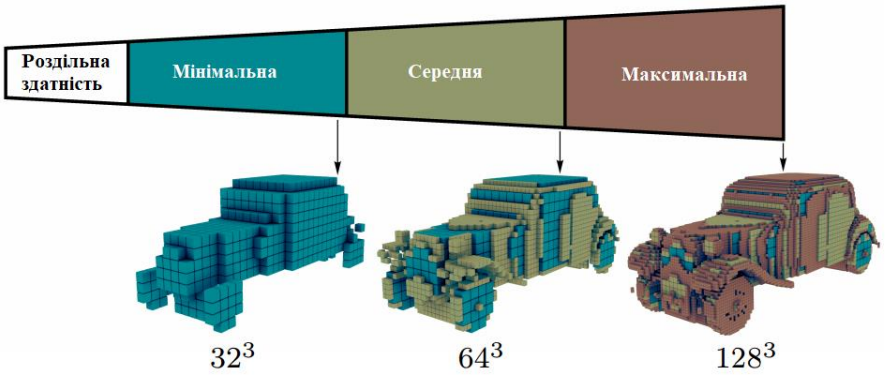

Рис. 3. Приклад розбиття об'єкта на вокселі

В [47] була запропонована нейронна мережа VoxelNet для виявлення об'єктів, що приймає на вхід хмару точок і перетворює її у воксельну сітку. VoxelNet ділить хмару точок на однаково розташовані тривимірні вокселі і перетворює групу точок у кожному вокселі в єдине подання ознак, що забезпечується введенням у мережу шару кодування ознак вокселів (Voxel Feature Encoding,VFE). Додавання декількох шарів VFE дозволяє вивчати складні ознаки для того, щоб характеризувати локальні 3D-форми. Далі за допомогою 3D-згортки додатково збираються локальні воксельні ознаки, що трансформує хмару точок в об'ємне високорозмірне подання. Нарешті, RPN обробляє об'ємне подання та дає результат виявлення.

Висновки

Навчання з підкріпленням та його глибокі модифікації залишаються одним із найбільш ефективних підходів до формування автономної поведінки агентів у складних середовищах. Висока обчислювальна складність та значні вимоги до обсягів навчальних даних визначають необхідність подальшого пошуку методів, здатних зменшити витрати ресурсів за збереження якості результату.

Трансформерні архітектури і великі мультимодальні моделі забезпечують гнучку,

контекстно-залежну поведінку агентів. Подальший розвиток цього напряму доцільно спрямувати на оптимізацію моделей для роботи у режимі реального часу та інтеграцію з ефективними методами обробки просторових даних.

Поєднання воксельних представлень із трансформерними та CLIP-подібними архітектурами створює основу для семантично керованої взаємодії агентів з об'єктами у тривимірних середовищах. Такий підхід може підвищити точність виконання завдань і покращити переносимість отриманих навичок у реальний світ.

Сучасні 3D-симулятори є ключовим інструментом для масштабованого, безпечного та контрольованого навчання агентів. Їхнє вдосконалення сприятиме ефективнішій перевірці та адаптації поведінкових стратегій перед впровадженням у реальні умови, що дозволить знизити ризики у критично важливих або високовартісних застосуваннях.

СПИСОК ЛІТЕРАТУРИ

1. Introduction to machine learning, neural networks, and deep learning / R. Y. Choi et al. *Translational Vision Science & Technology February*. 2020. Vol. 9 (2), Article14. DOI: 10.1167/tvst.9.2.14.

2. Di Franco G., Santurro M. Machine learning, artificial neural networks and social research. *Qual Quant*. 2021. Vol. 55. P. 1007–1025. DOI: 10.1007/s11135-020-01037-y.

3. Krishnan N. AI Agents: evolution, architecture, and real-world applications. *Arxiv.Org*. 2025. URL: https://arxiv.org/html/2503.12687v1.

4. Terven J. Deep reinforcement learning: a chronological overview and methods. *AI*. 2025. Vol. 6 (3), Article46. DOI: 10.3390/ai6030046.

5. Deep learning, reinforcement learning, and world models / Y. Matsuo et al. *Neural Networks*. 2022. Vol. 152. P. 267–275. ISSN 0893-6080.

6. Goel A., Goel A. K., Kumar A. The role of artificial neural network and machine learning in utilizing spatial information. *Spatial Information Research*. 2023. Vol. 31 (3). P. 275–285. DOI: 10.1007/s41324-022-00494-x.

7. Ghasemi M., Ebrahimi D. Introduction to reinforcement learning. *arXiv*. 2024. https://doi.org/10.48550/arXiv.2408.07712.

8. Proximal policy optimization algorithms / J. Schulman et al. *arXiv*. 2017. https://doi.org/10.48550/arXiv.1707.06347.

9. Proximal policy optimization via enhanced exploration efficiency / J. Zhang et al. *Information Sciences*. 2022. Vol. 609. P. 750–765. DOI: 10.1016/j.ins.2022.07.111.

10. Asynchronous methods for deep reinforcement learning / V. Mnih et al. *arXiv*. 2016. https://doi.org/10.48550/arXiv.1602.01783.

11. Towards understanding asynchronous advantage actor-critic: convergence and linear speedup / H. Shen et al. *IEEE Transactions on Signal Processing*. 2023. Vol. 71. P. 2579–2594. DOI:10.1109/TSP.2023.3268475.

12. Human-in-the-loop reinforcement learning: a survey and position on requirements, challenges, and opportunities / C. O. Retzlaff et al. *Journal of Artificial Intelligence Research*. 2024. Vol. 79, P. 359–415. DOI: 10.1613/jair.1.15348.

13. Learning human objectives by evaluating hypothetical behavior / S. Reddy et al. *Proceedings of the 37 th International Conference on Machine Learning, Online*. 2020. PMLR 119. P. 8020–8029.

14. Safe deep RL in 3D environments using human feedback / M. Rahtz et al. *arXiv*. 2022. https://doi.org/10.48550/arXiv.2201.08102.

15. Zero-shot and few-shot learning with knowledge graphs: a comprehensive survey / J. Chen et al. *Proceedings of the IEEE*. 2023. Vol. 111 (6). P. 653–685.

16. Multi-agent system-based framework for an intelligent management of competency building / F. Outay et al. *Smart Learn. Environ*. 2024. Vol. 11, Article 41. DOI: 10.1186/s40561-024-00328-3.

17. Habitat 2.0: training home assistants to rearrange their habitat / A. Szot et al. *Proceedings of the 35th International Conference on Neural Information Processing System (NIPS 21)*. 2021. P. 251–266.

18. ProcTHOR: large-scale embodied AI using procedural generation / M. Deitke et al. *Advances in Neural Information Processing Systems*. 2022. Vol. 35. P. 5982–5994.

19. AI2-THOR: An interactive 3D environment for visual AI / E. Kolve et al. *ArXiv*. 2017. https://doi.org/10.48550/arXiv.1712.05474.

20. Fisher J. C. A new Python API for Webots robotics simulations. *Proceedings of the 21th Python In Science Conference (SCIPY 2022)*. 2022. P. 147–151.

21. Deepbots: A Webots-based deep reinforcement learning framework for robotics / M. Kirtas et al. *Artificial Intelligence Applications and Innovations (AIAI 2020)*. 2020. P. 64–75. DOI: 10.1007/978-3-030-49186-4_6.

22. The application of CoppeliaSim EDU in robot course teaching / Y. Zhang et al. *Journal of Education and Educational Research*. 2024. Vol. 7 (1). P. 167–170. DOI: 10.54097/as82qn67.

23. RL STaR Platform: reinforcement learning for simulation based training of robots / T. Blum et al. *ArXiv*. 2020. https://doi.org/10.48550/arXiv.2009.09595.

24. A deep reinforcement learning framework for control of robotic manipulators in simulated environments /

C. Calderón-Cordova et al. *IEEE Access*. 2024. Vol. 12. P. 103133–103161. DOI: 10.1109/ACCESS.2024.3432741.

25. TEACh: Task-driven Embodied Agents that Chat / A. Padmakumar et al. *Proceedings of the AAAI Conference on Artificial Intelligence.* 2022. Vol. 36 (2). P. 2017–2025. DOI: 10.1609/aaai.v36i2.20097.

26. Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments / S. Srivastava et al. *Proceedings of the 5th Conference on Robot Learning (CoRL 2021)*. 2021. P. 477–490.

27. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions / L. Alzubaidi et al. *Journal of Big Data*. 2021. №8, Article 53. DOI:10.1186/s40537-021-00444-8.

28. Mienye I. D., Swart T. G., Obaido G. Recurrent neural networks: a comprehensive review of architectures, variants, and applications. *Information*. 2024. Vol. 15 (9), Article 517. DOI:10.3390/info15090517.

29. Hochreiter S., Schmidhuber J. Long Short-Term Memory. *Neural Computation.* 1997. Vol. 9 (8). P. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

30. Attention is all you need / A. Vaswani et al. *Advances in Neural Information Processing Systems*. 2017. Vol. 30. P. 5998–6008.

31. Decision transformer: reinforcement learning via sequence modeling / L. Chen et al. *Advances in Neural Information Processing Systems.* 2021. Vol. 34. P. 15084–15097.

32. Ferrara E. Gen AI against humanity: nefarious applications of generative artificial intelligence and large language models. *Journal of Computational Social Science*. 2024. Vol. 7. P. 549–569. DOI: 10.1007/s42001-024-00250-1.

33. A survey on large language model based autonomous agents / L. Wang et al. *Frontiers of Computer Science*. 2024. Vol. 18, Article 186345. DOI:10.1007/s11704-024-40231-1.

34. Kaufmann T., Bengs V., Hüllermeier E. Reinforcement learning from human feedback for cyber-physical systems: on the potential of self-supervised pretraining. *International Conference on Machine Learning For Cyber-Physical Systems*. ML4CPS 2023, 2024. Vol 18, Springer, Cham. P. 11–18. DOI:10.1007/978-3-031-47062-2_2.

35. Agents play thousands of 3D video games. / Z. Xu et al. *arXiv.* 2025. https://doi.org/10.48550/arXiv.2503.13356.

36. Katara P., Xian Z., Fragkiadaki K. Gen2sim: Scaling up robot learning in simulation with generative models. *IEEE International Conference on Robotics and Automation*. 2024. P. 6672–6679. DOI: 10.1109/ICRA57147.2024.10610566.

37. From Large Language Models to Large Multimodal Models: a literature review / D. Huang et al. *Applied Sciences*. 2024. Vol. 14, Article 5068. DOI:10.3390/app14125068.

38. Tu W., Deng W., Gedeon T. A closer look at the robustness of Contrastive Language-Image Pre-Training (CLIP). *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS 2023)*. 2023. P. 13678–13691.

39. Transductive zero-shot and few-shot CLIP / S. Martin et al. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024. P. 28816–28826. DOI: 10.1109/CVPR52733.2024.02722.

40. Wang Z. 3D Representation methods: a survey. *arXiv*. 2024. https://doi.org/10.48550/arXiv.2410.06475.

41. Xu Y., Tong X., Stilla U. Voxel-based representation of 3D point clouds: Methods, applications, and its potential use in the construction industry. *Automation in Construction*. 2021. Vol. 126, Article 103675. DOI: 10.1016/j.autcon.2021.103675.

42. PointNet: deep learning on point sets for 3D classification and segmentation / C. R. Qi et al. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017. P. 652–660.

43. Pointnet++: Deep hierarchical feature learning on point sets in a metric space / C. R. Qi et al. *Advances in neural information processing systems.* 2017. Vol. 30. P. 5099–5108. DOI:10.48550/arXiv.1706.02413.

44. PointNet++ network architecture with individual point level and global features on centroid for ALS point cloud classification / Y. Chen et al. *Remote Sensing*. 2021. Vol. 13 (3), Article 472. DOI: 10.3390/rs13030472.

45. Using deep learning in semantic classification for point cloud data / X. Yao et al. *IEEE Access.* 2019. Vol. 7. P. 37121–37130. DOI: 10.1109/ACCESS.2019.2905546.

46. 3D point cloud object detection method based on multi-scale dynamic sparse voxelization / J. Wang et al. *Sensor*. 2024. Vol. 24 (6), Article 1804. DOI:10.3390/s24061804.

47. Zhou Y., Tuzel O. Voxelnet: end-to-end learning for point cloud based 3D object detection Proceedings of the *IEEE conference on computer vision and pattern recognition*. 2018. P. 4490–4499. DOI: 10.1109/CVPR.2018.00472.

Стаття надійшла до редакції 22.07.2025.
Стаття пройшла рецензування 25.07.2025.

Тарновський Артем Миколайович – аспірант кафедри обчислювальної техніки.
Вінницький національний технічний університет.

Захарченко Сергій Михайлович – канд. техн. наук, професор кафедри обчислювальної техніки, e-mail: zakharchenko.sergii@vntu.edu.ua.
Вінницький національний аграрний університет.