Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження моделей навчання з підкріпленням для створення анімації рухів персонажів
(тема)

Виконав:
студент 2 курсу, групи СШІм-21-2
Волошинова Г.С.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

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

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник доц. каф. ШІ Чала Л.Е.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

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

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Волошиновій Ганні Сергіївні
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження моделей навчання з підкріпленням для створення анімації рухів персонажів

затверджена наказом університету від 31 березня 20 23 р. № 306Ст
2. Термін подання студентом роботи до екзаменаційної комісії 18 травня 20 23 р.
3. Вихідні дані до роботи Науково-технічні публікації, дані Інтернет-джерел з інформацією про проведені дослідження у галузі створення анімацій рухів персонажів за допомогою моделей навчання з підкріпленням

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної області
2) Методи і алгоритми машинного навчання і глибокого навчання для створення анімацій
3) Проведення експериментального дослідження
4) Аналіз результатів проведення дослідження

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри)_____

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

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

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз предметної галузі	04.03.2023	виконано
2	Аналіз методів створення анімацій	15.04.2023	виконано
3	Аналіз методів і алгоритмів глибокого навчання	20.04.2023	виконано
4	Експериментальне дослідження моделей	28.04.2023	виконано
5	Написання пояснювальної записки	05.05.2023	виконано
6	Попередній захист	13.05.2023	виконано
7	Захист перед ЕК	18.05.2023	

Дата видачі завдання 3 квітня 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. каф. ШІ Чала Л.Е.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 61 с., 15 рис., 4 табл., 1 дод., 26 джерела.

АНІМАЦІЯ, ГЛИБОКЕ НАВЧАННЯ, ЗАХОПЛЕННЯ РУХУ, ІМІТАЦІЙНЕ НАВЧАННЯ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, ПРОЦЕДУРНА АНІМАЦІЯ, UNITY, UNITY ML AGENTS

Об'єкт дослідження – моделі навчання з підкріпленням та глибокого навчання з підкріпленням.

Предмет дослідження – методи створення моделей навчання з підкріпленням та глибокого навчання з підкріпленням.

Мета роботи – застосування моделей навчання з підкріплення для створення анімацій рухів персонажів.

Методи дослідження – аналіз існуючих підходів для створення анімацій рухів персонажів, розробка архітектури та алгоритму навчання моделей навчання з підкріпленням, програмна реалізація та проведення комп’ютерного експерименту, обробка та аналіз отриманих результатів, порівняння ефективності метода з існуючими методами. Під час виконання кваліфікаційної роботи проведений теоретичний аналіз літературних джерел щодо методів моделювання штучних нейронних мереж та алгоритмів їх навчання, наукових публікацій щодо створення анімацій та моделей навчання з підкріплення інших алгоритмів глибокого навчання.

ABSTRACT

Explanatory note: 61 p., 15 fig., 4 tabl., 1 ann., 26 sources.

ANIMATION, DEEP LEARNING, IMITATION LEARNING, MOTION CAPTURE PROCEDURAL ANIMATION, REINFORCEMENT LEARNIG , UNITY, ML AGENTS

Object of research – models of reinforcement learning and deep learning with reinforcement.

The subject of research – methods of creating models of reinforcement learning and deep learning with reinforcement.

Purpose – to apply reinforcement learning models to create character motion animations.

Research methods – analysis of existing approaches of creating character animations, development of architecture and algorithm for training reinforcement learning models, software implementation and computer experiment, processing and analysis of the results, comparison of the method's effectiveness with existing methods. During the certification work, a theoretical analysis of literature sources on methods for modeling artificial neural networks and their training algorithms, scientific publications on creating animations and reinforcement learning models, and other deep learning algorithms was conducted.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

2D – двовимірний;

3D – тривимірний ;

ШІ – Штучний інтелект;

ВН – behavioral cloning – поведінкове клонування;

DLR – deep reinforcement learning – глибоке навчання з підкріпленням;

GAIL – generative adversarial imitation learning – генеративне змагальне імітаційне навчання;

GAN – generative Adversarial Network – генеративна змагальна мережа;

GRU – gated recurrent unit – вентильний рекурентний вузол;

IL – imitation learning – (Імітаційне навчання);

LSTM – long short term memory – довга короткочасна пам'ять;

ML – machine learning – машинне навчання;

PPO – proximal policy optimization – проксимальна оптимізація стратегії;

RL – reinforcement learning – навчання з підкріпленням;

RNN – recurrent neural network – рекурентна нейронна мережа.

ВСТУП

Анімація – це захоплюючий та універсальний вид мистецтва, який зачаровує глядачів вже понад століття. Від перших днів мальованих мультфільмів до новітніх комп'ютерних зображень (CGI) анімація зазнала значних змін у своїх техніках і стилях. Сьогодні анімація є не лише засобом розваги, але й потужним інструментом в освіті, рекламі та інших галузях. Успіх таких анімаційних фільмів і серіалів, як «Крижане серце» Діснея та «Історія іграшок» Піксара, сприяв зростанню популярності анімації, а технологічний прогрес відкрив нові можливості для аніматорів. Однак, незважаючи на широке розповсюдження, анімація залишається складною і відповідальною сферою, яка вимагає поєднання художніх і технічних навичок.

Однак анімація також використовується в ігровій індустрії для створення реалістичних 3D-світів. Анімація у відеоіграх – це процес створення та відображення серії зображень або кадрів у швидкій послідовності для імітації руху та оживлення персонажів і об'єктів в ігровому середовищі.- Після створення анімації її зазвичай інтегрують в ігровий рушій, який відповідає за керування тим, коли і як анімація відображається. Це може включати такі речі, як запуск анімації на основі введення гравцем, об'єднання декількох анімацій для створення більш складних рухів, а також регулювання часу і швидкості анімації, щоб вона виглядала більш реалістично. Загалом, анімація відіграє вирішальну роль в оживленні світів відеоігор, роблячи їх більш захоплюючими та цікавими для гравців.

Програми комп’ютерної графіки і віртуальної реальності, від фільмів до відеоігор, широко використовують віртуальних персонажів, тобто цифрові зображення людей, тварин чи інших живих істот. Протягом тривалого часу стандарти анімації прагнули до реалізму та контролю над стилем руху за допомогою повністю кінематичних персонажів, часто розроблених

художниками вручну спеціально для кожної ситуації. У великих анімаційних студіях та ігрових галузях анімація персонажів створюється за допомогою техніки захоплення руху на основі маркерів, яка вимагає від актора одягати костюм для захоплення руху. Щоб зафіксувати рух актора, у кімнаті потрібно бути кілька камер. Після цього дані захоплення руху застосовуються до 3D-віртуального персонажа через процес монтажу. Такий спосіб анімації персонажів вимагає величезних витрат часу і ресурсів.

Штучний інтелект вже досяг значного прогресу у створенні 3D-анімації, і очікується, що він продовжить вдосконалюватися в майбутньому. ШІ може автоматизувати багато трудомістких завдань у процесі 3D-анімації, таких як монтаж і захоплення руху. Це може дозволити аніматорам працювати швидше та ефективніше, що потенційно знизить виробничі витрати. ШІ може аналізувати фізику реального світу і використовувати ці знання для створення більш реалістичної анімації. Наприклад, ШІ може допомогти створити більш природні рухи персонажів або точніше імітувати поведінку рідин та інших матеріалів. З розвитком інструментів штучного інтелекту 3D-анімація може стати більш доступною для неспеціалістів. Наприклад, людина без великого досвіду в анімації зможе використовувати інструмент зі штучним інтелектом для створення базової анімації або візуальних ефектів. ШІ може відкрити нові творчі можливості в 3D-анімації, дозволяючи аніматорам експериментувати з новими техніками або створювати нові типи контенту.

Загалом, потенційний вплив штучного інтелекту на 3D-анімацію, ймовірно, буде значним, і ми можемо очікувати на подальші інновації в цій галузі в міру розвитку технології штучного інтелекту.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальний огляд анімації і видів створення анімації

Анімація – це техніка створення ілюзії руху та змін стану об’єкта шляхом швидкого показу послідовності статичних зображень, які мінімально відрізняються одне від одного. Це форма візуальної розповіді, яка може оживляти персонажів, об'єкти та середовище.

Існує кілька різних типів анімації, зокрема:

– традиційна анімація: найстаріша і найпоширеніша форма анімації, де кожен кадр намальований вручну художником [1]. У традиційній анімації зображення малюються або розфарбовуються вручну на прозорих целулоїдних аркушах, які потім фотографуються і демонструються на плівці. Наприклад, фонова сцена, яка не рухається, буде базовим шаром, а персонажі та інші рухомі об'єкти будуть намальовані на прозорих аркушах, розміщених зверху. Таким чином, їх можна анімувати без того, щоб фон або інші персонажі та реквізит рухалися одночасно. Традиційна анімація була використана для створення деяких з найбільш знакових анімаційних фільмів і телевізійних шоу. Хоча традиційна анімація є трудомістким процесом, вона залишається улюбленою і шанованою формою анімації, яка продовжує використовуватися сьогодні як у 2D, так і в 3D анімації.

– комп'ютерна анімація: сьогодні багато анімації створюється за допомогою комп'ютерної графіки (CGI). Комп'ютерна анімація може бути дуже деталізованою 3D-анімацією, тоді як 2D-анімація (яка може мати вигляд традиційної анімації) може використовуватися зі стилістичних міркувань, низької пропускної здатності або для швидшого рендерингу в реальному часі [2]. Такий тип анімації може використовуватися для широкого спектру застосувань, включаючи фільми, відеоігри та рекламу. Хоча комп'ютерна анімація може бути трудомісткою і технічно складною,

вона забезпечує високий ступінь контролю і гнучкості у створенні реалістичного або стилізованого анімаційного контенту.

– анімація стоп-моушн передбачає маніпуляції з фізичними об'єктами та створення серії фотографій, з невеликими змінами об'єктів між кожним знімком. Коли зображення відтворюються послідовно, здається, що об'єкти рухаються самі по собі. Існує багато різних типів стоп-моушн анімації, які зазвичай називаються за матеріалами, що використовується для створення анімації [3]. Комп'ютерне програмне забезпечення широко доступне для створення цього типу анімації; традиційна стоп-моушн анімація зазвичай дешевша, але більш трудомістка у виготовленні, ніж сучасна комп'ютерна анімація.

– захоплення руху (Motion capture): це техніка, що використовується для захоплення рухів людей або інших об'єктів реального світу та використання цих даних для анімації цифрових персонажів або об'єктів. Ця техніка допомагає підвищити реалістичність 3D-анімації. Захоплення руху використовується у відеоіграх, фільмах і телевізійних шоу для створення реалістичної та природної анімації. Актори одягають костюми зі спеціальними датчиками і розігрують сцену. Анімаційне програмне забезпечення для захоплення рухів переводить рухи з датчиків у цифрову версію [4]. Існує також програмне забезпечення, яке може захоплювати вираз обличчя, щоб допомогти точніше передати емоції та нюанси актора. Хоча анімація захоплення руху може бути потужним інструментом для створення реалістичної анімації, вона має певні обмеження. Наприклад, вона може не захопити певні тонкі нюанси або рухи, які важливі для передачі емоцій або рис характеру. Крім того, налаштування та збір даних про рух може бути дорогим і трудомістким процесом.

1.2 Процедурна анімація

Процедурна анімація – це техніка створення анімації за допомогою комп'ютерних алгоритмів і правил, а не традиційної покадрової анімації чи захоплення руху. Ця техніка часто використовується в комп'ютерній графіці, відеоіграх і симуляторах для створення складних і реалістичних анімацій [5].

Процес процедурної анімації зазвичай передбачає створення набору правил і алгоритмів, які визначають поведінку і рух об'єктів або персонажів, що анімуються. Ці правила можуть бути засновані на фізиці, штучному інтелекті або інших математичних моделях, що імітують реальний світ. Після визначення правил програма може автоматично генерувати анімацію на основі вхідних параметрів, таких як положення і швидкість об'єктів або персонажів. Це забезпечує високий ступінь контролю та гнучкості у створенні складних і динамічних анімацій.

Процедурна анімація може використовуватися для створення широкого спектру ефектів, включаючи природні явища, такі як вода і вогонь, а також анімацію персонажів, таких як ходьба і біг. Вона також корисна для створення анімації в реальному часі, наприклад, у відеоіграх або інтерактивних симуляторах.

Деякі яскраві приклади процедурної анімації включають динамічні погодні ефекти в таких іграх, як Grand Theft Auto V і The Legend of Zelda: Breath of the Wild, а також динамічну анімацію натовпу в таких іграх, як FIFA і NBA 2K.

Хоча процедурна анімація може бути потужним інструментом для створення складних і реалістичних анімацій, вона вимагає глибокого розуміння програмування і математики. Також може бути складно створити алгоритми, які точно імітують реальний світ і створюють анімацію, що виглядає природно і правдоподібно.

1.3 Motion matching

Motion matching – це техніка, яка використовується в розробці відеоігор та інших додатків у реальному часі для створення більш реалістичної та динамічної анімації персонажів. Ця технологія покликана покращити традиційну анімацію захоплення руху, дозволяючи персонажам рухатися більш плавно і природно, з більшою реакцією на дії гравця.

Процес зіставлення рухів зазвичай передбачає створення великої бази даних з даними захоплення руху для різних рухів, таких як ходьба, біг, стрибки та повороти. Потім ці дані зберігаються в системі, яка може швидко і точно зіставити поточний рух персонажа з найближчим доступним рухом у базі даних. Коли гравець вводить команду руху, система зіставлення рухів шукає в базі даних найбільш відповідний рух і плавно поєднує його з поточною анімацією. Це створює більш природну і чуйну анімацію, яка адаптується до дій гравця в реальному часі. Логіка цього метода представлена на рисунку 1.1.

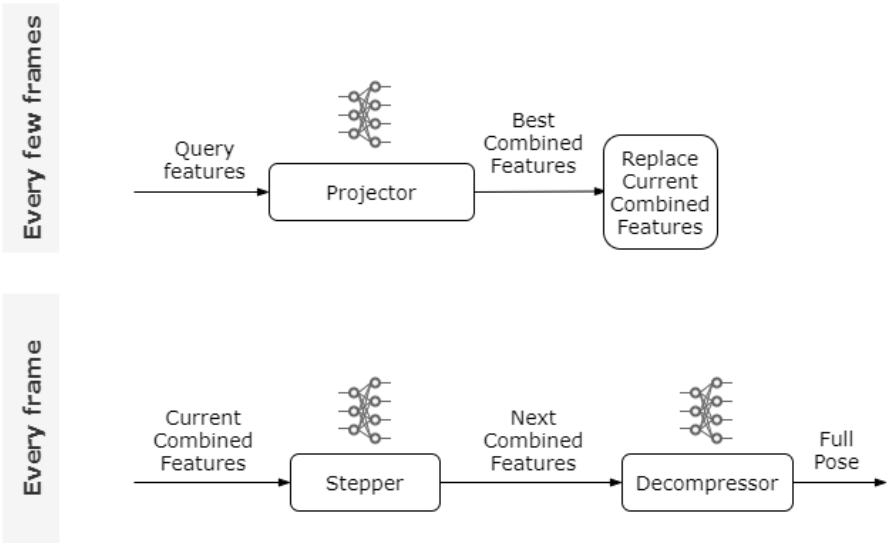


Рисунок 1.1 – Логіка узгодження руху (Motion matching)

Підбір рухів був використаний у кількох відеоіграх, включаючи NBA

2K, Assassin's Creed і Red Dead Redemption 2. Він також використовується в інших додатках, таких як робототехніка і віртуальна реальність.

Хоча зіставлення рухів може створювати більш реалістичні та динамічні анімації, воно вимагає значних обчислювальних потужностей і ємності для зберігання та доступу до бази даних захоплених рухів. Також може бути складно створити достатньо велику і різноманітну базу даних, щоб охопити всі можливі рухи і взаємодії персонажів.

1.4 Проблеми пов’язані з процесом створення анімації персонажів

Створення анімації персонажа – це складне і відповідальне завдання, що вимагає багатьох технічних і творчих навичок. Хоча технологічний прогрес спростив створення реалістичної та динамічної анімації, все ще існує кілька сучасних проблем, з якими стикаються аніматори при створенні анімації.

Однією з найбільших проблем в анімації персонажів є створення природних і правдоподібних рухів. Навіть за допомогою технологій захоплення руху або процедурної анімації може бути важко створити анімацію, яка точно імітує фізику та рухи реальних об'єктів і персонажів. Це вимагає глибокого розуміння анатомії, фізики та руху, а також вміння втілювати ці концепції в реалістичну анімацію.

Ще одним викликом в анімації персонажів є створення анімації, яка реагує на дії гравця або інші зовнішні фактори. В іграх або інтерактивних проектах персонажам може знадобитися реагувати на зміну оточення або інших персонажів, і анімація повинна мати можливість адаптуватися в реальному часі. Для цього потрібна надійна анімаційна система, яка може безперешкодно поєднувати кілька анімацій і реагувати на зовнішні події.

Крім того, створення анімації персонажів може бути тривалим і трудомістким процесом. Аніматори повинні створювати та вдосконалювати анімацію кадр за кадром, що може зайняти багато годин або навіть днів для

складних анімацій. Це може бути особливо складно, коли ви працюєте в стислі терміни або з обмеженими ресурсами.

Нарешті, зростає потреба в різноманітності та інклюзивності анімації персонажів. Оскільки відеоіграми та іншими формами медіа користується все більше різноманітних аудиторій, зростає попит на персонажів, які представляють ширший спектр ідентичностей та досвіду. Це вимагає від аніматорів бути чутливими до культурних відмінностей і створювати анімацію, яка відображає різноманітні погляди та досвід.

1.3 Постановка завдання дослідження

Незважаючи на те, що сучасні технології полегшили створення анімації, аніматори все ще стикаються з багатьма проблемами при створенні природної, чутливої та різноманітної анімації, яка залучає та розважає аудиторію.

Отже, предметом дослідження цієї роботи є методи створення моделей навчання з підкріпленням та глибокого навчання з підкріпленням. Метою роботи – дослідження і використання моделей навчання з підкріпленням та глибокого навчання з підкріпленням для створення анімацій рухів персонажів.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

– провести аналіз предметної області;

– дослідити та проаналізувати існуючі методи створення анімацій рухів персонажів;

– дослідити та проаналізувати існуючі моделі машинного та глибокого навчання;

– провести експериментальне дослідження методів, використовуючи розроблені алгоритми;

– проаналізувати та порівняти отримані результати.

2 ОГЛЯД МОДЕЛЕЙ ВИКОРИСТОВУВАНИХ ДЛЯ СТВОРЕННЯ АНІМАЦІЙ

2.1 Використання систем штучного інтелекту

Останніми роками штучний інтелект все частіше використовується в анімації з метою автоматизації процесу виробництва анімації та підвищення ефективності та якості створення анімації. Методи на основі систем штучного інтелекту можна використовувати для широкого спектру анімаційних завдань, таких як прогнозування руху, синтез руху, моделювання персонажів, анімація обличчя та синхронізація губ.

Однією з головних переваг використання штучного інтелекту в анімації є можливість генерувати реалістичні та природні рухи і поведінку, чого може бути важко і довго досягти вручну. Методи на основі штучного інтелекту також можуть зменшити кількість ручної праці, необхідної для виробництва анімації, що може заощадити час і знизити витрати.

Існує багато різних моделей і методів, які можна використовувати в анімації, зокрема моделі глибокого навчання, такі як RNN, GAN і VAE, а також інші типи нейронних мереж, такі як CNN, мережі з автокодуванням, мережі, що базуються на увазі, і мережі навчання з підкріпленням[6].

Використання систем штучного інтелекту в анімації все ще перебуває на ранніх стадіях, і є багато проблем і обмежень, які необхідно вирішити, наприклад, потреба у великих обсягах високоякісних навчальних даних, потенційна можливість упередженості навчальних даних і складність досягнення художнього контролю і творчості за допомогою автоматизованих методів. Однак, за умови подальшого розвитку та досліджень, штучний інтелект має потенціал трансформувати анімаційну індустрію та революціонізувати спосіб створення анімаційного контенту.

Звісно, є певні переваги та недоліки кожної моделі, що використовується в анімації. Однак вибір моделі і її архітектури залежить

від конкретних потреб і вимог анімаційного проекту.

2.2 Нейронні мережі

Нейронні мережі, також відомі як штучні нейронні мережі, є підмножиною алгоритмів машинного навчання, натхненних структурою і функціями людського мозку. Вони складаються з взаємопов'язаних вузлів, відомих як штучні нейрони, які організовані в шари. Ці шари обробляють і перетворюють вхідні дані, дозволяючи мережі вчитися на основі закономірностей у даних і робити прогнози або класифікації [7].

Нейронні мережі складаються з трьох типів шарів: вхідні шари, приховані шари та вихідні шари. Вхідні шари отримують дані у вигляді векторів або матриць, тоді як вихідні шари виробляють кінцевий результат роботи мережі. Приховані шари, які розташовані між вхідними та вихідними шарами, обробляють вхідні дані за допомогою серії зважених обчислень і застосовують функцію активації для отримання результату, який передається наступному шару. Ваги між вузлами мережі спочатку задаються випадковими значеннями, але в процесі навчання вони коригуються, щоб мінімізувати похибку між виходом мережі та фактичним виходом. Цей процес називається навчанням, і він включає в себе подачу в мережу великої кількості маркованих навчальних даних і коригування ваг на основі помилок, вироблених мережею.

Нейронні мережі можна використовувати для широкого спектру завдань, таких як розпізнавання зображень, розпізнавання мови, обробка природної мови і навіть анімація [8]. Вони особливо корисні для завдань, що включають великі обсяги складних даних, де традиційні підходи до програмування можуть бути недостатніми. Використання нейронних мереж для анімації передбачає навчання моделей машинного навчання для створення або маніпулювання анімованим контентом, таким як персонажі, істоти або середовища. Ці моделі можна навчити виконувати широкий

спектр завдань, пов'язаних з анімацією, включаючи захоплення руху, зіставлення рухів, генерацію персонажів і прогнозування анімації.

2.3 Генеративні змагальні мережі

Генеративна змагальна мережа (GAN) – це модель машинного навчання, в якій дві нейронні мережі змагаються одна з одною, використовуючи методи глибокого навчання, щоб стати більш точними у своїх прогнозах. GAN зазвичай працюють без нагляду і використовують для навчання кооперативну гру з нульовою сумою, де виграш однієї людини дорівнює програшу іншої [9].

GAN складаються з двох нейронних мереж: мережі-генератора та мережі-дискримінатора, які навчаються змагально генерувати нові дані з тією ж статистикою, що і навчальна вибірка, наприклад, генеруючи нові фотографії, які виглядають принаймні зовні достовірними для людини-спостерігача, маючи багато реалістичних характеристик. На рисунку 2.1 зображено приклад архітектури цієї нейронної мережі.

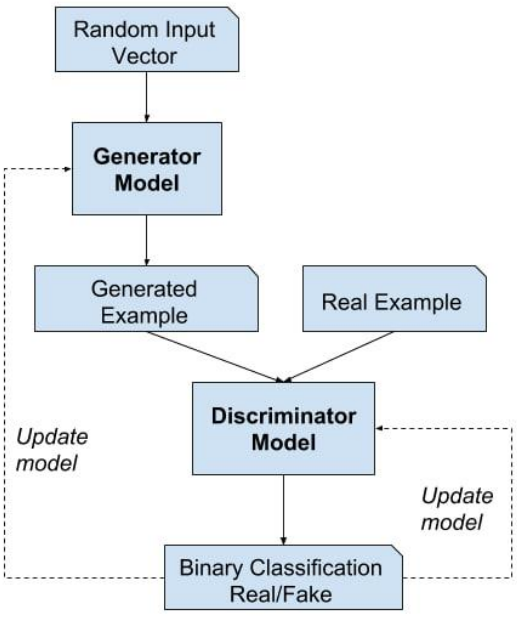


Рисунок 2.1 – Приклад архітектури моделі генеративної змагальної мережі.

В анімації персонажів GAN можна використовувати для створення нових анімацій, навчаючись на існуючих анімаціях, а потім генеруючи нові кадри, які відповідають подібному шаблону. Генераторна мережа навчається генерувати нові анімації, приймаючи випадковий шум на вході і створюючи нову анімацію на виході. Дискримінаторна мережа вчиться розрізняти згенеровані анімації від набору реальних анімацій з навчального набору даних. Під час навчання генераторна мережа намагається створювати анімації, які неможливо відрізнити від реальних анімацій, тоді як дискримінантна мережа намагається правильно ідентифікувати реальні анімації від згенерованих анімацій. З часом обидві мережі вчаться і покращують свою роботу: мережа-генератор створює більш реалістичні анімації, а мережа-дискримінатор стає більш точною у розрізненні реальних і згенерованих анімацій.

Однією з переваг використання GAN в анімації є те, що вони можуть генерувати дуже різноманітну та унікальну анімацію, яка може бути неможливою за допомогою традиційних методів анімації. GAN також можуть навчитися генерувати анімацію в широкому діапазоні стилів і рухів, що робить їх дуже універсальними.

Однак є й певні виклики, пов'язані з використанням цієї моделі в анімації. Одна з них полягає в тому, що вони можуть створювати артефакти або помилки в згенерованій анімації, що може вплинути на якість кінцевого результату. GAN також можуть бути дорогими в обчислювальному плані і вимагати значних витрат часу та ресурсів на навчання та створення анімації.

Переваги цієї моделі:

– може генерувати високореалістичну та різноманітну анімацію;

– може навчатися на великих наборах даних існуючих анімацій, що дозволяє використовувати широкий спектр стилів і рухів;

– можна створювати нові та унікальні анімації, які можуть бути неможливими за допомогою традиційних методів анімації.

Недоліки:

– іноді може створювати артефакти або помилки у згенерованій анімації.

– може вимагати значних обчислювальних ресурсів і часу для навчання та створення анімації.

– не завжди дають узгоджені результати, що ускладнює контроль за виходом.

2.4 Рекурентні нейронні мережі

2.4.1 Загальний огляд рекурентних нейронних мереж

Рекурентні нейронні мережі (RNN) – це тип нейронних мереж, які можуть обробляти послідовні дані, що робить їх особливо корисними для обробки часових рядів даних, таких як мова, текст і, у випадку анімації, дані про рух. На відміну від нейронних мереж прямого поширення, які обробляють вхідні дані за один прохід, RNN обробляють вхідну послідовність крок за кроком, зберігаючи внутрішній стан, який відображає контекст послідовності на даний момент.

Внутрішній стан RNN оновлюється на кожному кроці шляхом поєднання поточного вхідного сигналу з попереднім станом за допомогою набору вивчених ваг. Це дозволяє мережі зберігати інформацію про попередні часові кроки і включати її в поточний прогноз. Цей процес відомий як рекурентні обчислення, і він дозволяє RNN моделювати складні часові залежності в послідовних даних [10].

Однією з проблем звичайних RNN є те, що вони можуть страждати від проблеми зникаючого градієнта, коли градієнти, які використовуються для навчання, багаторазово перемножуються, в результаті чого вони стають дуже малими і фактично зникають. Для вирішення цієї проблеми було розроблено кілька варіантів RNN, включаючи мережі з довгою

короткочасною пам'яттю (LSTM) та рекурентні блоки зі стробуванням (GRU). Ці моделі використовують механізми вентилювання для управління потоком інформації через мережу, що дозволяє їм навчатися і зберігати інформацію протягом більш тривалих періодів часу. Порівняння моделей зображено на рисунку 2.2.

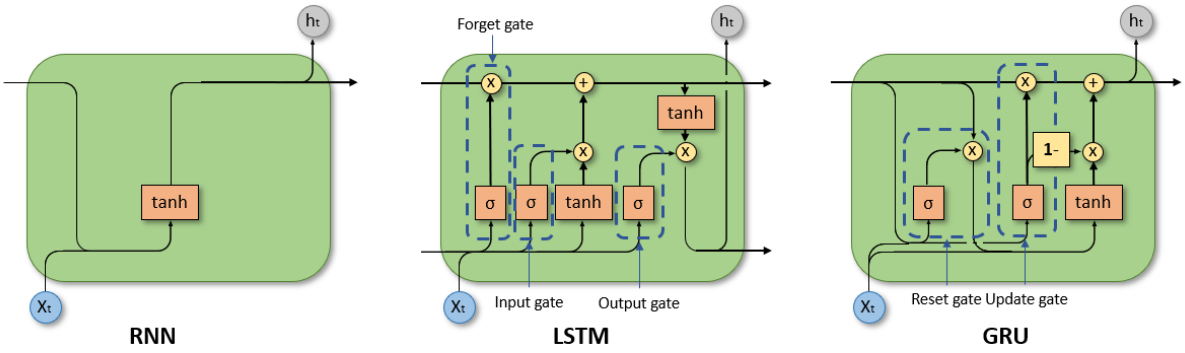


Рисунок 2.2 – RNN, LSTM та GRU.

У контексті анімації RNN використовуються для генерації послідовностей руху персонажів на основі попередніх даних про рух. Наприклад, RNN можна навчити на даних захоплення руху, щоб вивчити основні моделі руху певного персонажа, а потім генерувати нові послідовності рухів на основі цих моделей. RNN також використовуються в поєднанні з іншими моделями, такими як GAN, для створення більш реалістичних і різноманітних послідовностей рухів.

Загалом, RNN є потужним інструментом для моделювання послідовних даних і показали багатообіцяючі результати в галузі анімації.

Переваги:

– можливість моделювати складні часові залежності: ШНМ добре підходять для моделювання послідовних даних, що робить їх ідеальними для анімації персонажів, які рухаються в часі. Вони можуть навчитися вловлювати тонкі, мінливі в часі патерни в даних про рух, які інші методи можуть пропустити;

– гнучкість: RNN можна навчати на широкому спектрі даних про рух, включаючи дані захоплення руху, анімацію ключових кадрів або їх комбінацію. Їх також можна використовувати для створення нових послідовностей руху, що робить їх корисними для вивчення нових стилів анімації;

– ефективність: RNN можна навчати на відносно невеликих наборах даних, що може бути значною перевагою для аніматорів, яким потрібно швидко створювати кастомні анімації.

Недоліки:

– обчислювальна складність: Навчання RNN може бути дуже дорогим, особливо якщо вони навчаються на великих наборах даних. Це може ускладнити ефективне використання RNN для аніматорів з обмеженими обчислювальними ресурсами;

– складність у навчанні: RNN може бути складно навчати, особливо якщо дані про рух складні або містять багато шуму. Це може ускладнити досягнення якісних результатів і вимагати багато експериментів, щоб отримати правильний результат;

– обмежена здатність до узагальнення: RNN схильні до надмірної адаптації, що означає, що їм важко узагальнювати нові дані про рух, які відрізняються від тих, на яких вони були навчені. Це може обмежити їхню корисність у певних анімаційних програмах.

Загалом, RNN є потужним інструментом для анімації персонажів, але вони потребують ретельного налаштування та експериментів для досягнення найкращих результатів. Аніматори, які розглядають можливість використання RNN для анімації, повинні бути готові витратити час і зусилля на процес навчання, але результати можуть бути того варті.

2.4.2 LSTM

Мережі LSTM (Long Short-Term Memory – довга короткочасна

пам'ять) можна також використовувати для анімації персонажів. LSTM – це тип RNN, спеціально розроблений для уникнення проблеми зникаючого градієнта, яка виникає у стандартних RNN. Це робить його добре придатним для моделювання довгострокових залежностей у часових рядах даних, що є ключовою вимогою для анімації персонажів.

LSTM-мережі використовуються для різноманітних завдань анімації персонажів, включаючи прогнозування руху, синтез руху та перенацілювання руху. Наприклад, LSTM-мережі використовуються для прогнозування майбутнього руху персонажа на основі даних про його минулі рухи, для генерації нових послідовностей рухів на основі заданих вхідних даних і для перенесення руху одного персонажа на іншого.

Однією з переваг LSTM-мереж є те, що вони можуть вловлювати як короткострокові, так і довгострокові залежності в даних про рух, що важливо для створення плавного і реалістичного руху. Однак, як і інші типи нейронних мереж, LSTM-мережі можуть вимагати великих обсягів маркованих навчальних даних і можуть бути обчислювально дорогими для навчання.

Загалом, LSTM-мережі є потужним інструментом для анімації персонажів і успішно використовуються в різних додатках.

2.4.3 GRU

Мережі GRU (Gated Recurrent Unit) можна використовувати для анімації персонажів, так само як і мережі LSTM. GRU – це тип RNN, який розроблений для уникнення проблеми зникаючого градієнта, що виникає у стандартних RNN, а також підходить для моделювання довгострокових залежностей у часових рядах даних.

GRU-мережі використовувалися для різних додатків анімації персонажів, таких як прогнозування руху, синтез руху та перенацілювання руху. Наприклад, GRU-мережі використовувалися для генерації нових

послідовностей рухів на основі заданих вхідних даних, передачі руху від одного персонажа до іншого, а також для прогнозування майбутнього руху персонажа на основі даних про минулі рухи.

Однією з переваг GRU-мереж є те, що вони простіші, можуть бути швидшими і вимагають менше параметрів, ніж LSTM-мережі. Ця перевага може зробити їх більш практичними для певних додатків анімації персонажів. Однак, як і інші нейромережеві моделі, GRU-мережі все ще можуть вимагати великої кількості маркованих навчальних даних і можуть бути обчислювально дорогими для навчання.

Таким чином, GRU-мережі є підходящим варіантом для завдань анімації персонажів і успішно застосовуються в різних додатках. Однак вибір архітектури RNN залежить від конкретних потреб і вимог анімаційного проекту.

2.4.4 Порівняння RNN і GAN

RNN добре підходять для послідовних даних, таких як дані захоплення руху, анімації ключових кадрів та інших типів даних, що змінюються в часі, тоді як GAN можуть генерувати нові дані на основі випадкових вхідних даних, що робить їх ідеальними для створення нових і нових послідовностей руху.

RNN можуть фіксувати закономірності та залежності в даних про рух, що змінюються в часі, тоді як GAN можуть вивчати розподіл навчальних даних і генерувати нові вибірки даних на основі цього розподілу.

Вимоги до навчання: RNN потребують великих обсягів маркованих навчальних даних, а процес навчання може бути дорогим в обчислювальному плані. GAN потребують менше маркованих навчальних даних, а процес навчання можна прискорити за допомогою таких методів, як навчання з переносом.

RNN широко використовуються в прогнозуванні та синтезі руху, де

вони можуть генерувати нові послідовності руху на основі заданих вхідних даних. GAN широко застосовуються для створення нових, унікальних і креативних даних про рух, а також можуть використовуватися для перенесення стилю між різними наборами даних про рух.

RNN можуть генерувати реалістичні та плавні послідовності рухів, але вони можуть мати проблеми з генеруванням нових або унікальних рухів. GAN можуть генерувати нові та креативні дані руху, але не завжди можуть генерувати рух, який є настільки ж плавним або реалістичним, як у RNN.

Отже, можна сказати, що і RNN, і GAN мають свої сильні та слабкі сторони, коли мова йде про анімацію персонажів. RNN добре підходять для таких завдань, як прогнозування та синтез руху, тоді як GAN краще підходять для створення нових та креативних даних про рух. Обираючи між цими двома методами, аніматори повинні враховувати конкретні вимоги свого анімаційного проекту.

2.5 Варіаційні автокодери

Варіаційні автокодери (Variational Autoencoders, VAE) – це тип генеративних нейромережевих моделей, які також використовуються для анімації персонажів. VAE схожі на GAN тим, що їх можна навчити генерувати нові дані, подібні до наявного набору даних, але вони працюють інакше, ніж GAN.

VAE – це моделі латентних змінних [11]. Такі моделі ґрунтуються на ідеї, що дані, згенеровані моделлю, можуть бути параметризовані деякими змінними, які генеруватимуть певні специфічні характеристики даної точки даних. Ці змінні називаються латентними змінними.

Одна з ключових ідей VAE полягає в тому, що замість того, щоб намагатися явно побудувати латентний простір (простір латентних змінних) і робити вибірки з нього, щоб знайти вибірки, які дійсно можуть генерувати належні результати, будується мережа типу «кодер-декодер», яка розділена

на дві частини. Кодер вчиться генерувати розподіл залежно від вхідних вибірок X, з якого ми можемо вибрати латентну змінну, яка з високою ймовірністю генерує вибірок X. Іншими словами, ми вивчаємо набір параметрів θ1, які генерують розподіл Q(X,θ1), з якого ми можемо вибрати латентну змінну z, що максимізує P(X|z). Частина декодера вчиться генерувати вихід, який належить до реального розподілу даних, якщо на вхід подано латентну змінну z. Це зображено на рисунку 2.3.

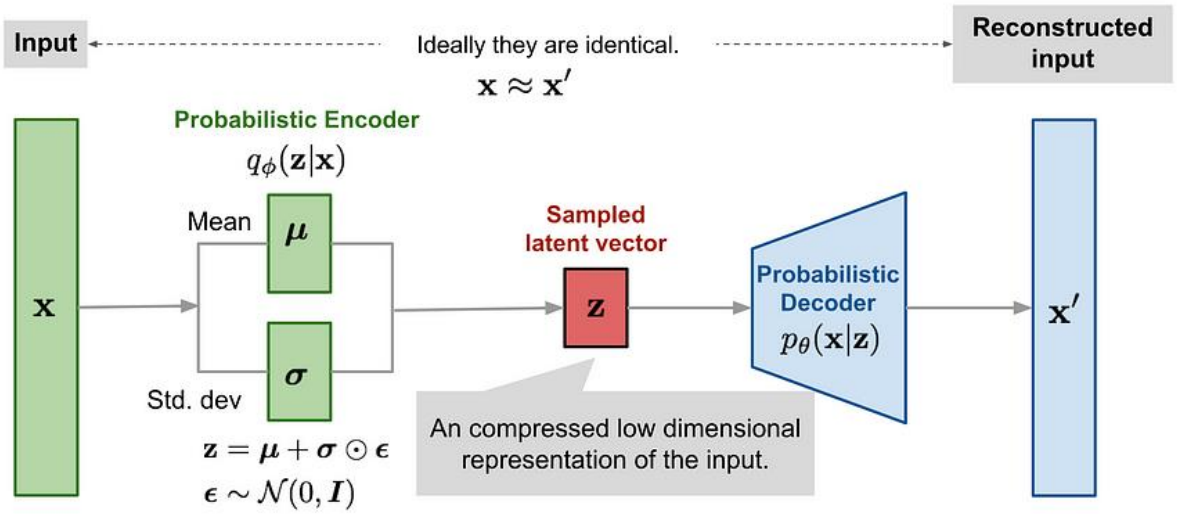


Рисунок 2.3 – Глобальна архітектура варіаційного автокодера

Однією з переваг використання VAE для анімації є те, що вони можуть бути використані для створення нових послідовностей руху, які мають схожі характеристики з вхідними даними, але також вводять варіації, які не присутні в оригінальному наборі даних. Це може бути корисним для створення більш різноманітної та цікавої анімації.

VAE використовуються в різних додатках для анімації персонажів, таких як синтез руху, прогнозування руху та перенацілювання руху. Наприклад, VAE використовувалися для синтезу нових послідовностей руху, схожих на існуючі, але з варіаціями, які можуть зробити анімацію більш природною і плавною. VAE також використовувалися для

прогнозування майбутнього руху персонажа на основі даних про минулі рухи і для перенесення руху одного персонажа на іншого.

Потенційним недоліком використання VAE для анімації є те, що вони іноді можуть генерувати послідовності рухів, які є менш реалістичними, ніж ті, що генеруються іншими методами, такими як GAN. Це пов'язано з тим, що VAE оптимізовані для максимізації вірогідності згенерованих даних, а не для оптимізації реалістичності чи якості. Крім того, VAE можуть вимагати великої кількості навчальних даних і можуть бути дорогими в обчислювальному плані для навчання.

Переваги використання варіаційних автокодерів (VAE) для анімації полягають у наступному:

– можливість генерувати нові та різноманітні послідовності руху: VAE можна навчати на великому наборі даних послідовностей руху, а потім використовувати для генерування нових і різноманітних послідовностей руху, подібних до тих, що містяться в навчальному наборі даних.

– можливість інтерполяції між послідовностями рухів: VAE можна використовувати для інтерполяції між двома різними послідовностями руху, створюючи плавний і природний перехід між ними;

– стійкість до шуму та помилок у вхідних даних: VAE розроблені таким чином, щоб бути стійкими до шуму та помилок у вхідних даних, що робить їх більш надійними, ніж інші методи синтезу руху;

– можливість контролювати стиль і характеристики руху: ШНМ можна використовувати для генерації послідовностей руху з певними стилями та характеристиками, регулюючи параметри ШНМ під час процесу генерації.

Недоліки використання VAE для цілей анімації включають

– складність інтерпретації латентного простору: Прихований простір ШНМ може бути складним для інтерпретації, що може ускладнити розуміння того, як ШНМ генерує послідовності рухів.

– обмежена здатність фіксувати довгострокові залежності: ШНМ

призначені для фіксації короткочасних залежностей у вхідних даних, і можуть мати труднощі з фіксацією довгострокових залежностей у послідовності рухів;

– обмежена здатність моделювати складні взаємодії між об'єктами: ШНМ може намагатися моделювати складні взаємодії між об'єктами в сцені, що може обмежувати їхню корисність для деяких завдань анімації;

– потреба у великих обсягах навчальних даних: Для створення високоякісних послідовностей руху VAE потребують великої кількості навчальних даних, що може зробити їх навчання більш трудомістким і дорогим, ніж інші методи синтезу руху.

2.6 Імітаційне навчання

Імітаційне навчання – це ще одна техніка, яку можна використовувати для анімації. При імітаційному навчанні агент вчиться виконувати завдання, імітуючи поведінку експерта-демонстратора. Цей підхід часто використовується в сценаріях, де важко або непрактично визначити явну функцію винагороди, яка керує поведінкою агента, наприклад, у складних середовищах з багатьма можливими діями та результатами. При імітаційному навчанні агенту надається набір експертних демонстрацій, які показують, як слід виконувати завдання. Потім агент вчиться зіставляти спостережувані входи (наприклад, зображення або показання датчиків) з відповідними виходами (наприклад, діями), мінімізуючи різницю між власною поведінкою та поведінкою експерта [12].

Однією з головних переваг імітаційного навчання є його здатність використовувати наявні знання та досвід для швидкого засвоєння нових завдань. Воно також може бути використане для генерування високоякісної поведінки, яку важко визначити або оптимізувати за допомогою традиційних методів навчання з підкріпленням. Однак імітаційне навчання

також має певні обмеження. Наприклад, для його ефективності потрібна велика кількість високоякісних демонстраційних даних, які може бути важко або дорого отримати. Крім того, воно може не справлятися зі складним і непередбачуваним середовищем або генерувати поведінку, яка виходить за рамки поведінки експерта-демонстратора.

У контексті анімації імітаційне навчання можна використовувати для навчання віртуальних персонажів імітувати рухи людських акторів або дані захоплення руху. Цей підхід може заощадити час і зусилля порівняно з традиційними методами анімації, оскільки усуває необхідність для аніматорів вручну створювати кожен кадр анімації.

Імітаційне навчання можна також поєднувати з іншими методами, такими як навчання з підкріпленням, щоб ще більше підвищити реалістичність і складність анімації персонажів. Наприклад, агент, навчений за допомогою імітаційного навчання, може бути точно налаштований за допомогою навчання з підкріпленням, щоб генерувати більш різноманітні та реалістичні рухи у відповідь на зміну умов навколишнього середовища.

Однією з головних переваг імітаційного навчання в анімації є його здатність вловлювати нюанси і тонкощі людських рухів, що призводить до більш природної і реалістичної анімації персонажів. Крім того, імітаційне навчання вимагає менше обчислювальних ресурсів і навчальних даних порівняно з іншими методами, такими як глибоке навчання з підкріпленням.

Незважаючи на численні переваги, імітаційне навчання має деякі обмеження. Наприклад, він може зіткнутися з труднощами, коли має справу з дуже складним або непередбачуваним середовищем, або коли намагається створити рухи, які виходять за рамки поведінки досвідченого демонстратора. Крім того, якість отриманих анімацій сильно залежить від якості вхідних даних і досвіду експерта-демонстратора, що може бути потенційним джерелом помилок або упередженості.

2.7 Навчання з підкріпленням

Навчання з підкріпленням – це тип техніки машинного навчання, в якому агент вчиться приймати рішення, взаємодіючи з навколишнім середовищем. Мета полягає в тому, щоб агент вивчив політику, яка максимізує кумулятивний сигнал винагороди з плином часу. У задачі навчання з підкріпленням агент виконує дії в середовищі і отримує зворотній зв'язок у вигляді винагороди або покарання на основі результатів цих дій [13].

Навчання з підкріпленням відрізняється від інших методів машинного навчання тим, що зосереджується на послідовному прийнятті рішень, де дії агента на одному часовому кроці впливають на винагороду та спостереження, які він отримує на наступному часовому кроці. Це створює цикл зворотного зв'язку, який дозволяє агенту вчитися на власному досвіді та вдосконалювати свою політику з часом. На рисунку 2.4 зображено архітектуру моделі навчання з підкріпленням.

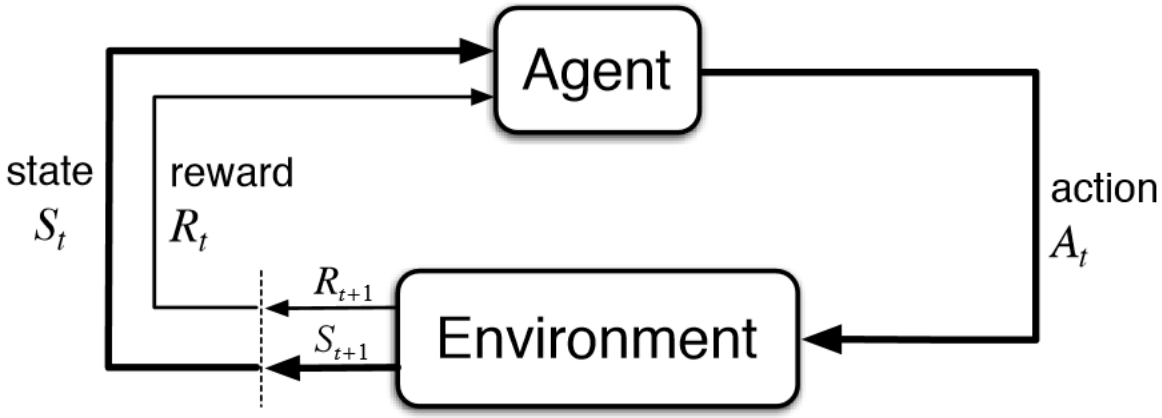


Рисунок 2.4 – Глобальна архітектура моделі навчання з підкріпленням

Навчання з підкріпленням успішно застосовується в широкому спектрі застосувань, включаючи ігри, робототехніку, фінанси та охорону здоров'я. Він також виявилося перспективним як інструмент для створення

анімацій [14].

У контексті анімації навчання з підкріпленням можна використовувати для створення більш реалістичних і природних рухів і поведінки персонажів.

Одним із способів застосування навчання з підкріпленням в анімації є навчання агента, щоб він навчився взаємодіяти з навколишнім середовищем у спосіб, який відповідає бажаній анімаційній поведінці. Це може включати в себе визначення набору функцій винагороди, які стимулюють агента виконувати певні дії або поведінку, наприклад, ходити, бігати або стрибати.

Інший підхід полягає у використанні навчання з підкріпленням для навчання агента виконувати складні завдання, такі як маніпулювання об'єктами або виконання акробатичних трюків, шляхом розбиття завдання на менші, більш керовані підзадачі. Це може включати використання ієрархічних методів навчання з підкріпленням, коли агент вчиться виконувати дії високого рівня, які складаються з менших дій нижчого рівня.

Однією з переваг використання навчання з підкріпленням для анімації є те, що воно дозволяє персонажам адаптуватися до нових ситуацій і середовищ, а також вчитися на власному досвіді. Це може зробити анімацію більш реалістичною та динамічною, оскільки персонажі реагують на зміни в навколишньому середовищі або взаємодіють з іншими персонажами в несподіваний спосіб.

Однак навчання з підкріпленням також має певні обмеження та проблеми, коли йдеться про анімацію, такі як потреба у великих обсягах навчальних даних, складність визначення відповідних функцій винагороди, а також потенційна можливість для агента застрягти в локальних оптимумах або демонструвати небажану поведінку. Тим не менш, навчання з підкріпленням є перспективним інструментом для створення більш складних і реалістичних анімацій.

2.8 Глибоке навчання з підкріпленням

Глибоке навчання з підкріпленням – це тип навчання з підкріпленням, який поєднує методи глибокого навчання з навчанням з підкріпленням, щоб дозволити агентам навчатися на основі сирих сенсорних даних, таких як зображення або аудіо. У глибокому навчанні з підкріпленням глибокі нейронні мережі використовуються для апроксимації політики або функції цінності, що дозволяє агенту вивчати більш складні та абстрактні уявлення про навколишнє середовище.

Однією з ключових переваг глибокого навчання з підкріпленням є те, що він може працювати з високовимірними, безперервними просторами станів, які є поширеними в багатьох реальних додатках, таких як робототехніка або ігри. Глибоке навчання з підкріпленням також може навчатися на основі необроблених сенсорних даних, усуваючи потребу в ручних функціях або експертних знаннях. На рисунку 2.5 зображена глобальна архітектура моделі глибокого навчання з підкріпленням.

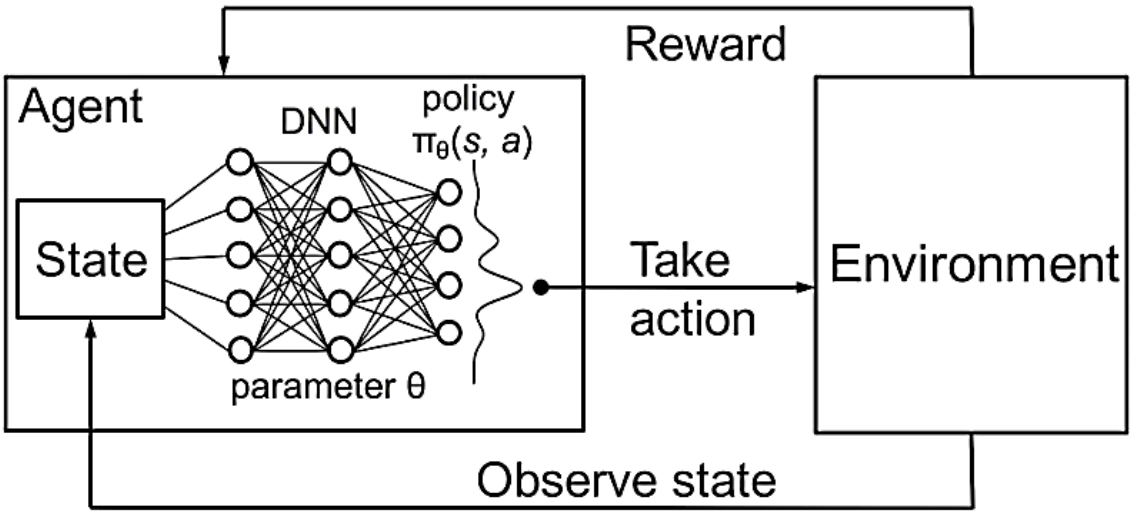


Рисунок 2.5 – Глобальна архітектура моделі глибокого навчання з підкріпленням

Глибоке навчання з підкріпленням успішно застосовується в

широкому спектрі застосувань, включаючи автономне водіння, ігри та робототехніку. Наприклад, глибоке навчання з підкріпленням використовується для навчання роботів виконувати складні завдання, такі як захоплення об'єктів або навігація в захаращеному середовищі.

Однак, глибоке навчання з підкріпленням може бути обчислювально інтенсивним і вимагає великих обсягів навчальних даних, що робить його складним для використання в деяких контекстах. Крім того, навчання агентів глибокого навчання з підкріпленням може зайняти багато часу і вимагати спеціалізованого обладнання або обчислювальних ресурсів. Незважаючи на ці труднощі, глибоке навчання з підкріпленням виявився дуже перспективним інструментом для створення більш складної анімації, а також для вирішення широкого кола інших складних завдань.

Глибоке навчання з підкріпленням продемонструвало великий потенціал для створення анімації персонажів. Використовуючи глибоке навчання з підкріпленням, аніматори можуть навчити агентів вчитися на основі сирих сенсорних даних і генерувати більш складні та різноманітні рухи.

Одним із прикладів використання глибокого навчання з підкріпленням для анімації є створення віртуальних акторів у відеоіграх і фільмах. Глибоке навчання з підкріпленням можна використовувати для навчання агентів генерувати реалістичні та різноманітні рухи персонажів. Це може призвести до більш захоплюючого та цікавого досвіду для користувачів.

Ще одне застосування глибокого навчання з підкріпленням – створення роботизованих систем. Глибоке навчання з підкріпленням можна використовувати для навчання роботів навчатися та адаптуватися до навколишнього середовища, дозволяючи їм виконувати складні завдання більш природними та плавними рухами. Наприклад, за допомогою глибокого навчання з підкріпленням можна навчити роботів хапати об'єкти, орієнтуватися в захаращеному середовищі та виконувати інші складні

завдання.

Однією з ключових переваг моделей цього типу в анімації є здатність працювати з високорозмірними та безперервними просторами станів. Це дозволяє створювати більш реалістичні та нюансовані анімації, оскільки агенти можуть навчатися на більш складних вхідних даних і генерувати більш детальні та природні рухи.

Однак, в будь-якому застосуванні глибокого навчання з підкріпленням, існують певні проблеми, які необхідно вирішити, наприклад, потреба у великих обсягах навчальних даних і обчислювальних ресурсах, необхідних для навчання. Крім того, використання моделей цього типу в анімації вимагає спеціальних знань і досвіду, що ускладнює його застосування для нефахівців. Незважаючи на ці виклики, глибоке навчання з підкріпленням має великі перспективи для створення більш складної анімації без артефактів, де рухи подібні до людських, в майбутньому.

3 ВИКОРИСТАННЯ МОДЕЛЕЙ НАВЧАННЯ З ПІДКРІПЛЕННЯМ ДЛЯ СТВОРЕННЯ АНІМАЦІЙ

3.1 Огляд існуючих рішень

Останніми роками в галузі анімації персонажів відбулися значні прориви, завдяки глибоким методам навчання з підкріпленням. Значно зросла кількість змодельованих персонажів, які виконують широкий спектр складних завдань. Багато з них обрали для цих задач локомоції методи градієнта політики.

Спочатку метод DeepLoco запропонував систему для додавання імітаційного члена до функції винагороди [15]. Запропонована ними система демонструє завдання локомоції за допомогою цілей розміщення ніг, які обчислюються високорівневими та низькорівневими контролерами. Вони тренували обидва рівні політики, використовуючи глибоке навчання з підкріпленням. Вони продемонстрували результати на двоногій 3D моделі, яка зображена на рисунку 3.1.

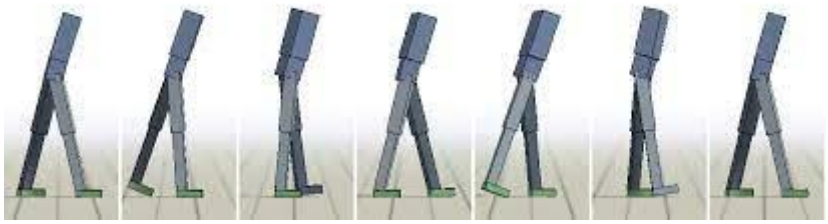


Рисунок 3.1 – Демонстрація роботи методу DeepLoco

Пізніше DeepMimic вдосконалили їхню роботу, запропонувавши методи глибокого навчання з підкріпленням, які можна використовувати для того, щоб змусити агента імітувати велику кількість еталонних відеокліпів руху. Їхній метод можна застосовувати як для надзвичайно динамічних дій, так і для рухів з ключовими кадрами. Вони поєднали

імітацію цілей руху з метою завдання. Їхній метод зберігає якість методів захоплення руху, роблячи його більш зручним у використанні [16]. Різноманітність анімацій, які були створені цим методом, зображено на рисунку 3.2.

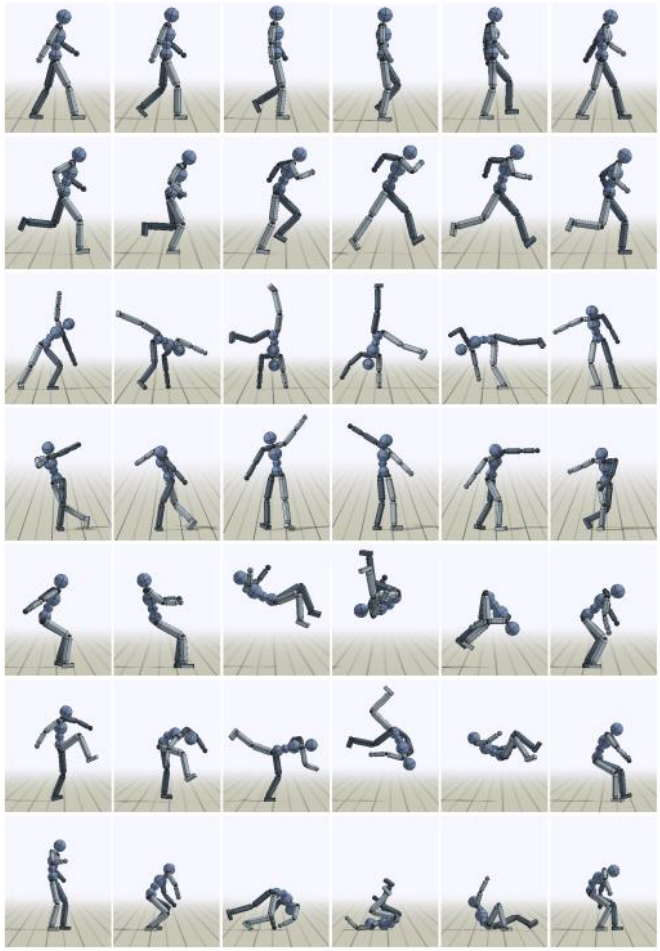


Рисунок 3.2 – Демонстрація роботи методу DeepMimic

У статті Self-imitation пропонується система навчання на основі самоімітації, яка дозволяє створювати стабільні контролери рухів для швидкого навчання [17]. Їхній підхід в основному поєднує дві нещодавні роботи з безперервного керування – FDI-MCTS та DeepMimic . Однак їхня робота спрямована на подолання обмежень цих двох методів. Ці обмеження включають високу вартість часу виконання методу FDI-MCTS та залежність

від даних і складність вибірки методу DeepMimic [18]. У запропонованій ними системі вони спочатку генерують еталонний рух за допомогою методу онлайн-пошуку по дереву, що дозволяє їм використовувати навчання з підкріпленням з RSI (Ініціалізація еталонного стану) для пошуку нейромережевого контролера для імітації еталонного руху [19]. Підхід, який запропоновано в цій роботі, подібно до попередніх досліджень, поєднує глибоке навчання з підкріпленням та імітаційне навчання, роблячи реалізацію набагато простішою та зручнішою у використанні.

3.2 Огляд інструментів дослідження

Серед інструментів, що були використанні при дослідженні ігровий рушій Unity 2021.3.15f1, Unity ML-AgentsToolkit для навчання агента в симульованому середовищі та анімації та 3D моделі Mixamo.

Unity – це крос-платформний ігровий рушій, розроблений Unity Technologies. Рушій можна використовувати для створення тривимірних (3D) та двовимірних (2D) ігор, а також інтерактивних симуляцій та іншого досвіду. Unity підтримує багато платформ, включаючи Windows, macOS, iOS, Android, Linux та багато інших. Він має візуальний редактор, який дозволяє розробникам створювати сцени, додавати об'єкти та маніпулювати ними в реальному часі. Unity також має потужну систему сценаріїв на основі C#, яка дозволяє розробникам створювати кастомну поведінку та взаємодію для своїх об'єктів. Крім того, Unity надає ряд ресурсів, таких як моделі, текстури та анімації, які можна використовувати для прискорення процесу розробки.

Unity Machine Learning Agents Toolkit (ML-Agents Toolkit) – це проект з відкритим вихідним кодом, який дозволяє використовувати ігри та симуляції в якості середовища для навчання інтелектуальних агентів. Агенти можуть навчатися за допомогою навчання з підкріпленням, імітаційного навчання, нейроеволюції та інших методів машинного

навчання за допомогою простого у використанні Python API.

Інструментарій Unity ML-Agents включає ряд функцій та інструментів, які полегшують розробникам навчання та розгортання ML-моделей у своїх проектах Unity. Сюди входить підтримка популярних бібліотек ML, таких як TensorFlow та PyTorch, а також готові середовища та приклади, які допоможуть розробникам швидко розпочати роботу.

Деякі з ключових особливостей Unity ML-Agents включають;

– підтримка широкого спектру алгоритмів ML, включаючи навчання з підкріпленням, імітаційне навчання та інші;

– гнучкі робочі процеси навчання та оцінювання з можливістю навчання локально або в хмарі;

– інтеграція з фізикою та анімаційними системами Unity, що дозволяє агентам взаємодіяти з ігровим середовищем та контролювати його;

– інструменти візуалізації в реальному часі для моніторингу поведінки та продуктивності агентів під час навчання.

Загалом, Unity ML-Agents надає розробникам ігор потужну та гнучку платформу для впровадження технологій штучного інтелекту та машинного навчання у свої ігри та симулятори, що дозволяє створювати нові та інноваційні ігрові можливості. Інструментарій Unity ML Agents включає декілька алгоритмів для навчання агентів, зокрема:

– Проксимальна оптимізація стратегії;
– Глибокі Q-мережі;
– Градієнт глибокої детермінованої стратегії з подвійною затримкою;
– М'який актор-критик;
– Curiosity-Driven Exploration;
– Стратегії еволюції;
– Навчання з глибоким підкріпленням на основі.

Ці алгоритми розроблені таким чином, щоб бути гнучкими і масштабованими, що дозволяє легко налаштовувати їх відповідно до конкретних вимог конкретного завдання або середовища.

Блок-схему цього інструмента зображено на рисунку 3.3.

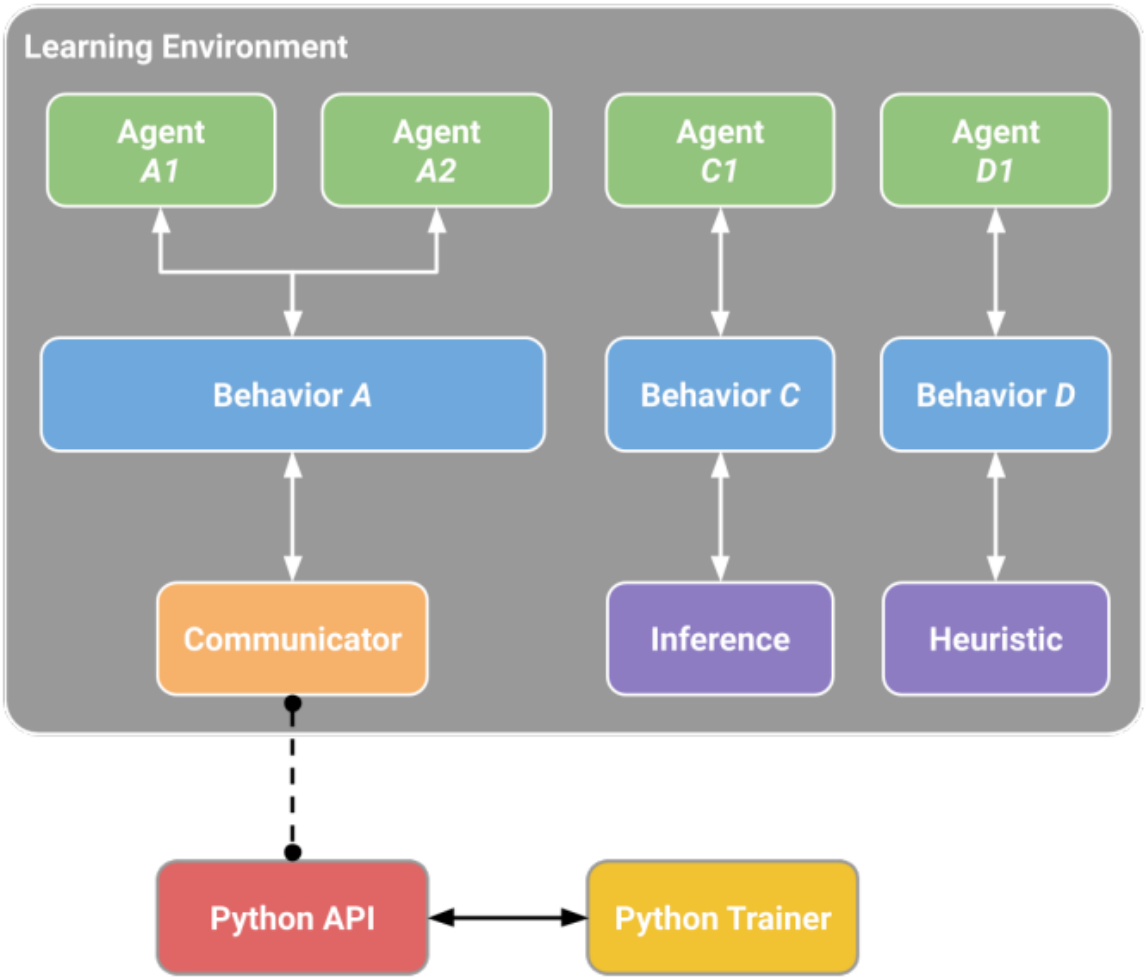


Рисунок 3.3 – Блок-схема ML-Agents Toolkit

Mixamo – це онлайн-платформа, яка надає готові 3D-моделі персонажів, анімацію та дані захоплення руху для використання в розробці ігор та інших цифрових медіа-проектів. Вона пропонує величезну бібліотеку моделей людей та істот з попередньо створеною анімацією та реквізитом, що дозволяє розробникам легко створювати та анімувати персонажів без необхідності мати значний досвід у 3D-моделюванні та анімації. Mixamo також дозволяє робити ретаргетинг анімації, що дозволяє користувачам легко переносити анімацію з одного персонажа на іншого, та функцію

автоматичного оснащення, яка автоматично створює скелет і скін для 3D-моделі, спрощуючи процес налаштування. На рисунку 3.4 зображено бібліотеку Mixamo.

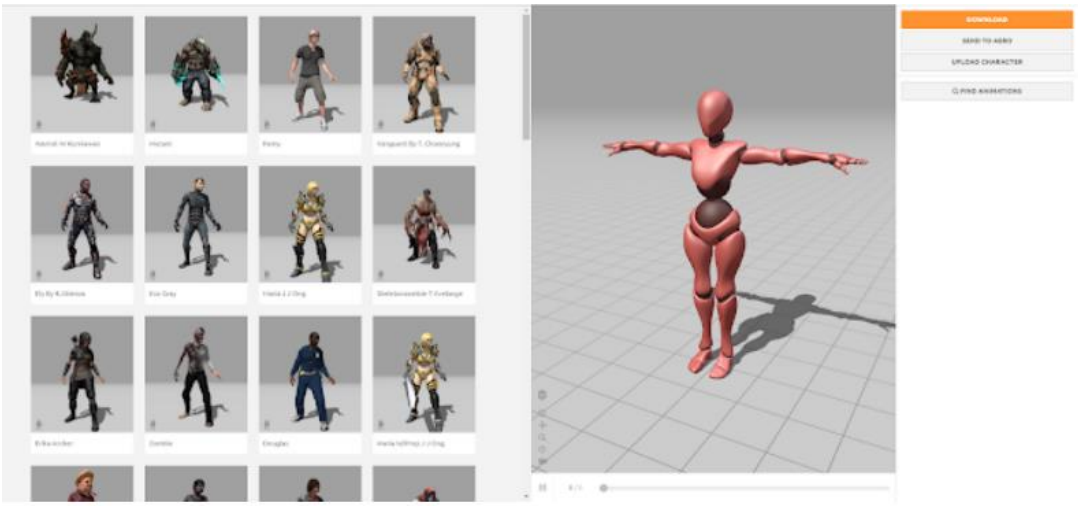


Рисунок 3.4 – Інструментарій Mixamo

При навчанні моделі було використано навчання з підкріпленням та імітаційне навчання. .Процес роботи з Unity ML-Agents можна розділити на три етапи. Створення середовища, навчання нейронної мережі, а потім вбудовування нейронної мережі в інше середовище. Спочатку були створенні навчальне середовище та модель персонажа за допомогою середовища прикладу, що надається Unity ML-Agents. Середовище складається з академії, мозку та агентів. Академія – це те, що дозволяє використовувати інші навчальні мережі за допомогою python API [19]. Навчання проводилося у дві різні сесії. Перша була проведена за допомогою найсучаснішого алгоритму навчання з підкріпленням, проксимальної оптимізації стратегії (Proximal Policy Optimization, PPO). У другій сесії було використано алгоритми імітаційного навчання, генеративне змагальне імітаційне навчання (Generative Adversarial Imitation Learning, GAIL) та поведінкове клонування(Behavioral Cloning, BC), у поєднанні з проксимальною оптимізацією стратегії.

3.3 Алгоритми, використані при створенні моделі

Інструментарій ML-Agents надає доступ до двох найсучасніших алгоритмів навчання з підкріпленням та двох алгоритмів імітаційного навчання. Було вирішено використовувати алгоритм навчання з підкріпленням, проксимальна оптимізація стратегії, у поєднанні з двома алгоритмами навчання з імітацією, генеративне змагальне навчання та поведінкове клонування.

При навчанні з підкріпленням агент знаходить стратегію, яка максимізує винагороду [20]. Серед численних алгоритмів навчання з підкріпленням, алгоритм проксимальної оптимізації стратегії виявився більш стабільним та універсальним. Він був представлений у 2017 році дослідниками з OpenAI. Цей алгоритм призначений для усунення деяких обмежень попередніх алгоритмів навчання з підкріпленням, таких як нестабільність і чутливість до гіперпараметрів. Проксимальна оптимізація стратегії стала популярним алгоритмом для навчання агентів у різноманітних задачах, включаючи ігри, робототехніку та симуляції. Було показано, що він досягає найсучаснішої продуктивності в декількох тестових середовищах, таких як ігровий набір Atari і набір управління OpenAI Gym.

Тому було вирішено використати самей цей алгоритм як алгоритм навчання з підкріпленням. За даними OpenAI, він працює порівняно краще, ніж інші сучасні алгоритми [21]. Окрім цього, ще однією причиною, чому був обраний цей алгоритм, є те, що його набагато простіше реалізовувати та налаштовувати, що робить його простим у використанні, зберігаючи при цьому хорошу продуктивність. Проксимальна оптимізація стратегії – це те, що дозволяє навчати агента в складних умовах, коли він намагається рухатися до місця призначення (цільового об'єкта). У цьому процесі він вчиться ходити і бігати. Навіть повертатися обличчям до цілі. В інших алгоритмах навчання з підкріпленням для отримання хорошого результату

доводиться докладати значних зусиль для налаштування гіперпараметрів. Налаштування гіперпараметрів не може бути простішим, ніж в цьому алгоритмі [10]. У новому варіанті алгоритму використовується нова цільова функція, яка зазвичай не зустрічається в інших алгоритмах:

$$L^{CLIP}(\theta) = E_t[\min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)], \qquad (3.1)$$

де ϵ є гіперпараметром, який, зазвичай, дорівнюється 0.1 або 0.2;

θ є параметром стратегії;

E_t – емпіричне очікування за часовими кроками;

r_t – відношення ймовірності при новій та старій стратегіях;

A_t – оцінена перевага в момент часу t.

Для того, щоб зробити оновлення області довіри, задача знаходить спосіб його реалізації. Він сумісний зі стохастичним градієнтним спуском. Крім того, за рахунок видалення розходження Кульбака-Лейбнера алгоритм спрощується. У тестах цей алгоритм показав найкращу продуктивність на завданнях безперервного управління і майже відповідає продуктивності ACER (Actor-Critic with Experience Replay), незважаючи на те, що він набагато простіший у реалізації.

В імітаційному навчанні передбачається, що можна отримати набір експертних даних. Набір даних містить набір траєкторій та факторів. Використовуючи ці набори даних, стратегія може бути вивчена шляхом прямого відображення фактора на траєкторію. Цей процес формування стратегії може бути реалізований за допомогою алгоритму, який називається поведінкове клонування[22]. Поведінкове клонування є найпростішою формою імітаційного навчання. За допомогою нього агент вчиться імітувати поведінку людини або експерта-демонстратора, навчаючи модель передбачати дії на основі вхідних даних. Він передбачає збір даних від експерта і використання їх для навчання моделі, щоб зіставити спостереження з діями. Навчену модель можна використовувати для

прогнозування в нових ситуаціях. Поведінкове клонування є формою контрольованого навчання і часто використовується в умовах, коли важко визначити функцію винагороди або коли необхідно вивчити складну поведінку у кваліфікованого демонстратора. У цьому алгоритмі збирається набір демонстрацій, зроблених експертом. Це представлено у вигляді набору траєкторій. Таким чином, для застосування необхідне представлення стратегії π_θ. Для порівняння подібності між продемонстрованою дією та стратегією агента – цільова функція L має бути обрана. Нарешті, параметри політики θ повинні бути повернуті. Цей алгоритм вважається ефективним, оскільки він здатний імітувати експерта без прямого контакту з навколишнім середовищем. Але помилка, допущена агентом, може легко перевести його в стан, в якому експерт ніколи не бував, а агент ніколи не тренувався. У таких станах поведінка агента є невизначеною, що може призвести до катастрофічних невдач. Це зображено на рисунку 3.4.

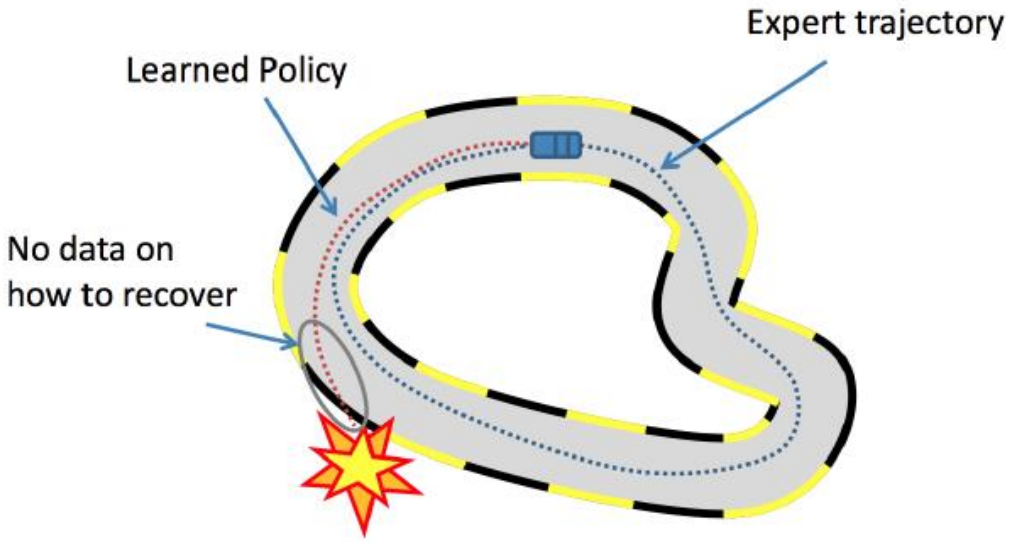


Рисунок 3.4 – Робота алгоритму поведінкове клонування

Алгоритм генеративного змагального імітаційного навчання – це швидкий і прямий підхід до проведення аналогії між генеративними змагальними мережами та імітаційним навчанням, спрямований на

підвищення продуктивності за рахунок імітації складної поведінки у великих, багатовимірних середовищах [23].

У GAIL мережа-генератор генерує зразки стратегії експерта, а мережа-дискримінатор розрізняє ці згенеровані зразки та демонстрації експерта. Мережа-генератор навчається генерувати зразки, які, на думку дискримінатора, не відрізняються від демонстрацій експерта. Це дозволяє генератору вивчити політику, яка імітує поведінку експерта. На відміну від традиційного імітаційного навчання, яке спирається на вже існуючий набір даних з демонстраціями експертів, GAIL може навчатися на основі взаємодії з навколишнім середовищем і вдосконалювати вивчену політику за допомогою ітеративного навчання. GAIL був застосований до низки завдань, включаючи робототехніку та автономне водіння, і показав свою ефективність у створенні високоякісних політик, які перевершують традиційні методи імітаційного навчання. Схема роботи алгоритму зображена на рисунку 3.5.

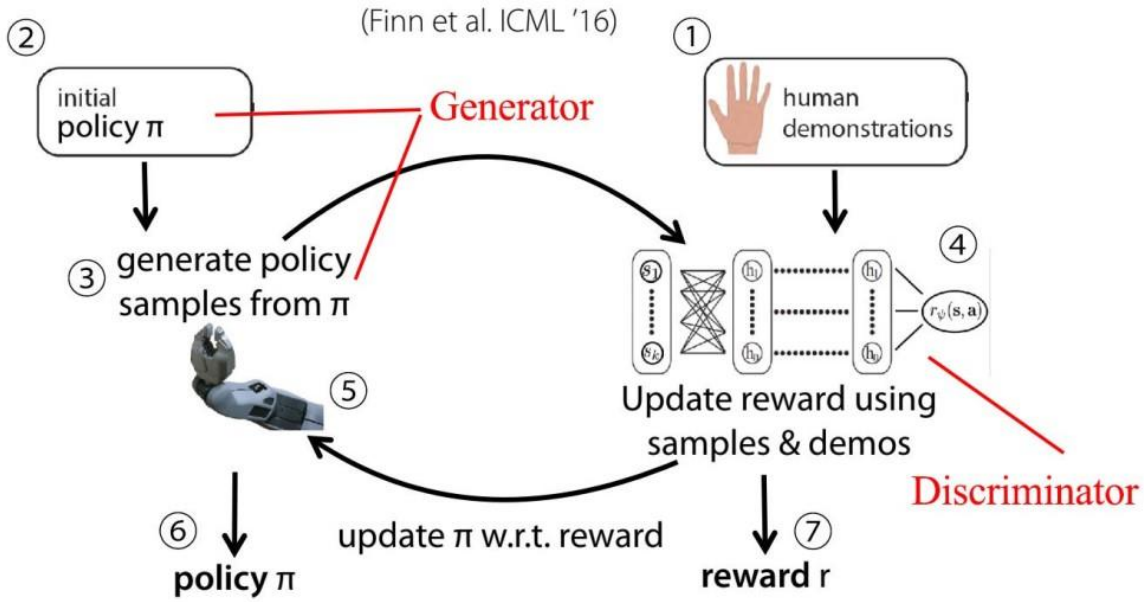


Рисунок 3.5 – Робота алгоритму генеративного змагального імітаційного навчання

У рамках GAIL використовується вторинна нейронна мережа, яка називається дискримінатором, що розрізняє спостереження/дії між демонстрацією та агентом. Дискримінатор винагороджує агента, якщо спостереження/дії агента наближаються до демонстрації. З кожним кроком навчання агент намагається максимізувати винагороду, а дискримінатор все краще розрізняє агента та демонстрацію [24]. Таким чином, агент вивчає політику, яка виконує стани та дії, подібні до демонстрації, що призводить до поведінки, подібної до людської. Алгоритм GAIL представлений наступним виразом:

$$E_{\pi}\left[\log\left(D(s,a)\right)\right] + E_{\pi E}[log1 - D(s,a)] - \lambda H(\pi), \qquad (3.2)$$

де *D* – дискримінатор, який приймає на вхід траєкторію, утворену станом *s* та відповідною йому дією a;

log(D(s,a)) – функція, що повертає неперервне значення між 0 та 1.

Чим більше це значення наближається до 1, тим більше вхідна траєкторія нагадує траєкторію експерта. Таким чином, *D* можна використовувати як сигнал винагороди для навчання *G* генерувати траєкторії, подібні до траєкторій експерта. Якщо, наприклад, *G* є алгоритмом градієнта стратегії, параметри якого задаються параметром θ, то оновлення градієнта можна обчислити як:

$$\nabla_{\theta} log\pi_{\theta}(s,a)Q(s,a), \qquad (3.3)$$

$$Q(s,a) = \log\left(D(s,a)\right). \qquad (3.4)$$

Як і в стандартному GAN, кожна ітерація навчання ділиться на дві частини, які повторюються протягом певної кількості кроків або епох:

– навчити дискримінатор D, використовуючи експертні та згенеровані траєкторії за формулою (3.3);

– оновлення політики G з кроком градієнта, обчисленим за формулою (3.4).

Експертні траєкторії можуть бути згенеровані людиною-експертом, алгоритмом або політикою, яка вже опанувала цільову задачу.

3.4 Винагорода

Інструментарій агентів ML визначає сигнали винагороди за модульним принципом. Він надає два типи сигналів винагороди. Зовнішній тип винагороди – це винагорода, яка визначається середовищем і відповідає досягненню мети. З іншого боку, внутрішні сигнали винагороди визначаються поза середовищем, щоб заохочувати поведінку агента певним чином і допомагати вивченню зовнішньої винагороди [25]. Отже, функція винагороди r_T визначається наступним чином:

$$r_T = r_E + r_I, \tag{3.5}$$

де r_T – це загальна миттєва винагорода на будь-якому часовому кроці t, яка є сумою зовнішньої винагороди r_E та внутрішньої винагороди r_I.

Зовнішня винагорода – це винагорода за виконання завдання для агента. Тут агент отримує позитивну винагороду, коли він відповідає цільовій швидкості і дивиться в напрямку цілі або торкається цільового об'єкта і отримує негативну винагороду за дотик до землі. Сигнал винагороди виражається наступною нотацією:

$$r_E = (r_S \times r_l) + r_t + r_p, \tag{3.6}$$

де r_s обчислює нормалізоване значення різниці між швидкістю руху до цілі та середньою швидкістю агента. Винагорода наближається до 1, якщо швидкість ідеально збігається, і наближається до 0, якщо відхиляється. Те

саме стосується r_l, де винагорода наближається до 1, якщо агент ідеально дивиться в напрямку цілі, і наближається до 0, коли він відхиляється. Вирази для r_s та r_l наступні:

$$r_s = [1 - (\frac{v_{goal} - v_{avg}}{v_{max}})^2]^2, \quad (3.7)$$

$$r_l = ((\sum_{i=1}^{n} C_i H_i) + 1) \times 0.5, \quad (3.8)$$

де v_{goal} – цільова швидкість руху;

v_{avg} – фактична швидкість агента;

v_{max} – максимальна цільова швидкість руху відповідно.

У рівнянні (3.8), C_i та H_i представляють компоненти прямих векторів куба орієнтації та частини тіла голови відповідно. Запис $\sum_{i=1}^{n} C_i H_i$ представляє точковий добуток двох прямих векторів.

Далі, r_t та r_p визначаються наступним чином:

$$r_t = 1, r_p = -1, \quad (3.9)$$

У рівнянні (3.9) штраф за контакт з землею дається тільки тоді, коли інші частини тіла, окрім ніг, торкаються землі.

Внутрішня винагорода являє собою імітацію об'єктивної винагороди. Для внутрішньої винагороди було використано внутрішню винагороду генеративного змагального імітаційного навчання. Винагорода генеративного змагального імітаційного навчання вводить в процес навчання упередженість до того, хто вижив, за допомогою методу дискримінатора-актор-критика (Discriminator-Actor-Critic, DAC). Функція винагороди дискримінатора визначається наступним чином [26]:

$$r_I = r(s_T, a_T) + \sum_{t=T+1}^{\infty} \gamma^{t-T} r(s_a), \quad (3.10)$$

де $r(s_T, a_T)$ представляє винагороду для кінцевих станів. З незміщеними функціями винагороди DAC, вивчена винагорода $r(s_a)$ додається до доходу для кінцевих станів замість просто $r(s_T, a_T)$ [14]. Для цього дискримінатор GAIL може розрізняти, чи є досягнення певного стану бажаною поведінкою з точки зору експерта, і відповідно до цього давати винагороду.

3.5 Тренування

Навчання мереж було проведено в Unity з використанням ML-агентів. Агенти навчалися в двох різних навчальних сесіях з двома різними конфігураціями навчання. У першій сесії агенти навчались лише за допомогою алгоритму проксимальної оптимізації стратегії, а в другій сесії разом з проксимальною оптимізацією стратегією було використано поведінкове клонування і генеративне змагальне імітаційне навчання. Навчання відбувається епізодично, де агент ініціалізується на початку кожного епізоду з випадковою орієнтацією, а цільова швидкість ходьби також є випадковою для кожного епізоду. Епізод завершується або після закінчення фіксованого часу, або при виконанні умови завершення, тобто коли агент падає на землю. Гіперпараметри для обох навчальних сесій ідентичні: розмір батча 2048, розмір буфера 20480 та швидкість навчання 3×10^{-4}. Гіперпараметри наведено в таблиці 3.1.

Таблиця 3.1 – Гіперпараметри для проксимальної оптимізації стратегії

Гіперпараметри	Значення
Розмір батча	2048
Розмір буфера	20480
Швидкість навчання	3×10^{-4}
Бета	5×10^{-3}
Епсілон	0.2
Лямбда	0.95

Продовження таблиці 3.1

Кількість епох	3

Що стосується мереж, то вони повністю з'єднані з 3 шарами та 512 прихованими елементами. Для першої навчальної сесії використовувався лише сигнал зовнішньої винагороди з гамою 0,995 та силою 1,0, де гама – це коефіцієнт дисконтування для майбутніх винагород, отриманих з навколишнього середовища, а сила – це коефіцієнт, на який множиться дана винагорода. На другому тренуванні до конфігурації навчання додаються сигнали внутрішньої винагороди поведінкового клонування і генеративного навчання з підкріпленням, які мають низьку силу 0,1 і однаковий набір демонстрацій. Параметри внутрішньої винагороди генеративного навчання з підкріпленням наведені в таблиці 3.2.

Таблиця 3.2 – Параметри для генеративного навчання з підкріпленням

Параметри	Значення
Сила	0.1
Гамма	0.99
Розмір кодування	128
Швидкість навчання	$3.0 \times e^{-4}$

3.6 Практичні експерименти

Як було зазначено раніше, всі тренування та експерименти проводилися в ігровому рушію Unity. Також було імпортовано моделі персонажів та еталонну анімацію з Mixamo.

Було використано три моделі персонажів для навчання. Одна з них – WalkerRagdoll з Unity, друга – The Robot Kyle model з Unity assets store, а третя – Y Bot з Mixamo. Моделі були гуманоїдними, оскільки спочатку мета була досягти двоногого пересування. Отже, можна сказати, що будь-яка

гуманоїдна модель персонажа може бути використана з методом, який був запропонований в цій роботі. Портрети моделей персонажів можна знайти на рисунку 3.6, а персонажів, що виконують завдання – на рисунку 3.7. Після імпорту моделі персонажа з Mixamo, регдолл для персонажа був створений за допомогою майстра регдоллів Unity. Довелося вручну конфігурувати деякі колайдери та конфігурації з'єднання, щоб вони відповідали WalkerRagdoll'y Unity. Масу та опір жорсткого тіла було відповідно відрегульовано для всіх частин тіла.

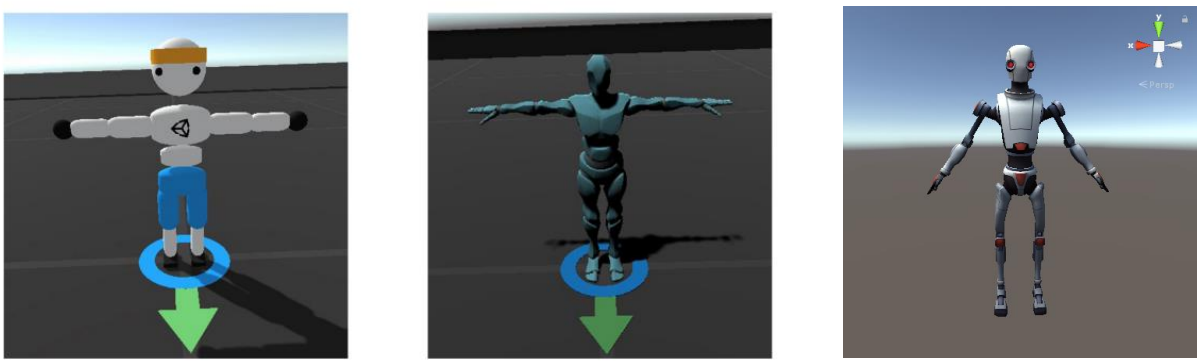
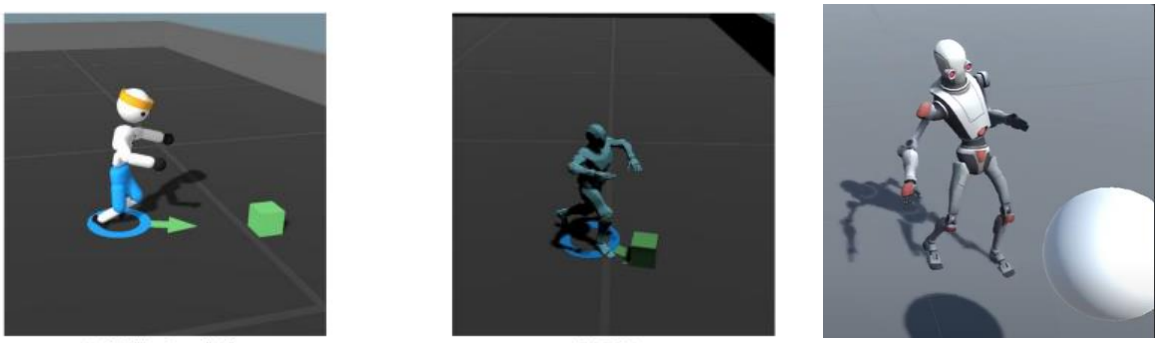


Рисунок 3.6 – Обрані гуманоїдні моделі



(a) WalkerRagdoll (b) Y Bot

Рисунок 3.7 – Результати практичного експерименту

Навчальне середовище складається з платформи як основи та об'єкта, куба або кулі, як мішені. Модель персонажа розміщується в центрі

платформи, а об'єкт-мішень випадковим чином з'являється на платформі. Рандомізатор швидкості ходи рандомізує швидкість ходи від 0.1 до 10 для кожного епізоду. Таким чином, агент навчається ходити з будь-якою обраною швидкістю від 0,1 до 10. Агент разом з платформою та цільовим об'єктом дублюється багато разів, так що середовище містить 10 незалежних агентів з однаковими параметрами поведінки. Це значно прискорює процес навчання. Для виконання завдання було використано всі моделі персонажів в одних і тих самих ситуаціях середовища.

За допомогою еталонних анімацій, імпортованих з Mixamo, ми налаштували евристичну функцію для запису демонстрацій агента. Демонстрації були записані за допомогою записувача демонстрацій Unity. Ми записали демонстрацію експерта тривалістю 5-7 хвилин, яка складалася з 3534 кроків. Ця демонстрація була використана як еталон для алгоритмів генеративного змагального навчання і поведінкового клонування.

3.7 Результати практичних експериментів

В ході роботи було проаналізовано результати навчання в TensorBoard, інструментарії візуалізації TensorFlow. Це набір веб-додатків для перевірки та розуміння навчальних прогонів та графіків. TensorBoard забезпечив нас візуалізацією та інструментами, необхідними для результатів навчання. Відстежувались та візуалізувалися графіки моделі кумулятивної винагороди, тривалості епізоду, втрати стратегії, втрати вартості та всі графіки стратегії.

В першій тренувальній сесії навчалися лише моделі Y Bot і The Robot Kyle, де використовувався лише алгоритм проксимальної оптимізації стратегії. У таблиці 3.3 наведено дані про кумулятивну винагороду та тривалість епізодів для завершених кроків. З таблиці видно, що для максимальної кількості кроків 30000000 середня кумулятивна винагорода становить 488.3, а довжина епізоду – 169.4.

Таблиця 3.3 – Результати першої сесії

Кроки	Кумулятивна винагорода	Довжина епізоду
15M	395.3	148.5
30M	488.3	169.4

В таблиці 3.4 надані результати другої тренувальної сесії, де генеративне змагальне імітаційне навчання та поведінкове клонування були використані разом з проксимальною оптимізацією стратегії. Для тренувальної сесії було використано дві моделі.

Таблиця 3.4 – Результати другої тренувальної сесії

Параметри	Результати
Кроки	30M
Кумулятивна винагорода	657.8
Довжина епізода	214.4

В таблиці 3.4 можна побачити, що після 30000000 максимальних кроків середня кумулятивна винагорода становить 657.8, а довжина епізоду становить 214.4. Незважаючи на те, що тривалість епізоду більша на другому тренуванні, сукупна винагорода також вища, ніж на першому тренуванні.

На рисунку 3.8 показано графічне представлення середньої кумулятивної винагороди та тривалості епізоду. Тут графік кумулятивної винагороди представляє середню кумулятивну винагороду за епізод двох тренувальних сесій.

Перша тренувальна сесія позначена червоним кольором, а друга тренувальна сесія (PPO + GAIL + BC) – синім кольором. Під час успішного тренування графік повинен зростати.

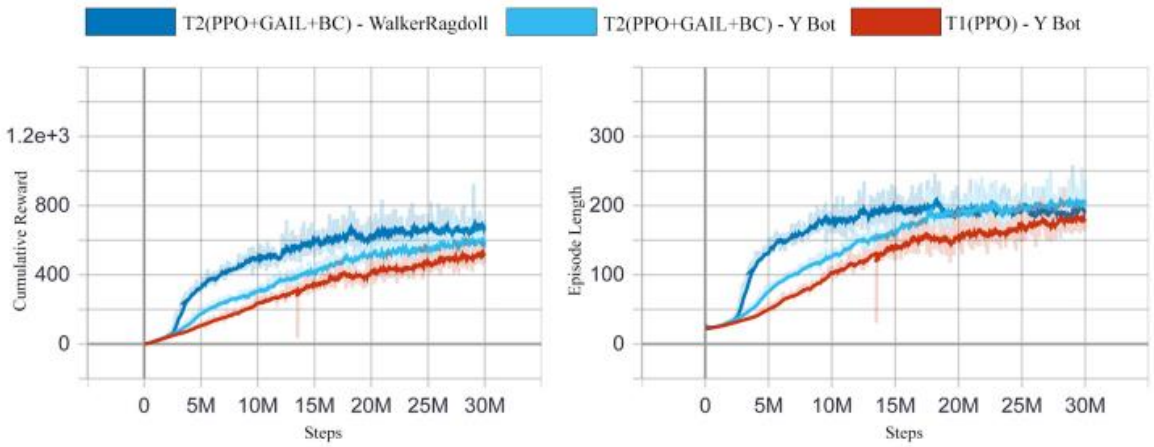


Рисунок 3.8 – Результати тренувальних сесій

На графіку кумулятивної винагороди видно, що друга тренувальна сесія генерує більше середньої винагороди, оскільки вона є більш зростаючою. Графік «Довжина епізоду » показує середню довжину кожного епізоду. Можна побачити, що довжина епізоду також більша для другого тренування.

На рисунку 3.9 та рисунку 3.10 показано графіки функцій втрат. Графік втрат GAIL згенеровано лише для другого тренування, оскільки для першого тренування він не використовувався.

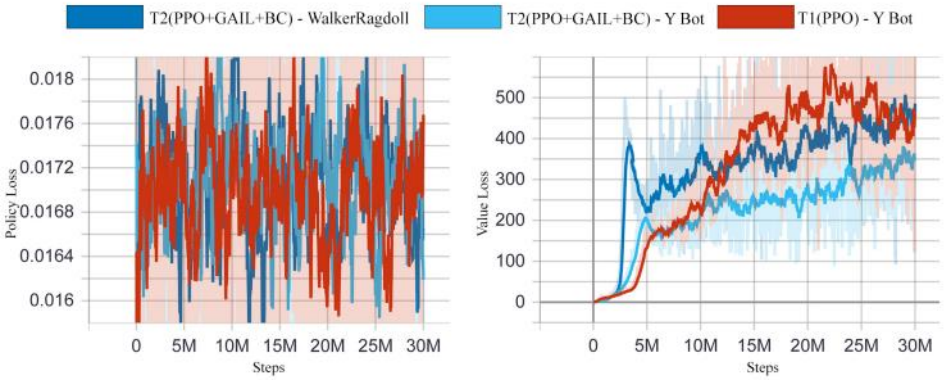


Рисунок 3.9 – Графіки функції втрат стратегії

Цей графік показує середню величину втрат дискримінатора

генеративного змагального імітаційного навчання, що свідчить про ефективність моделі в імітації демонстраційних даних.

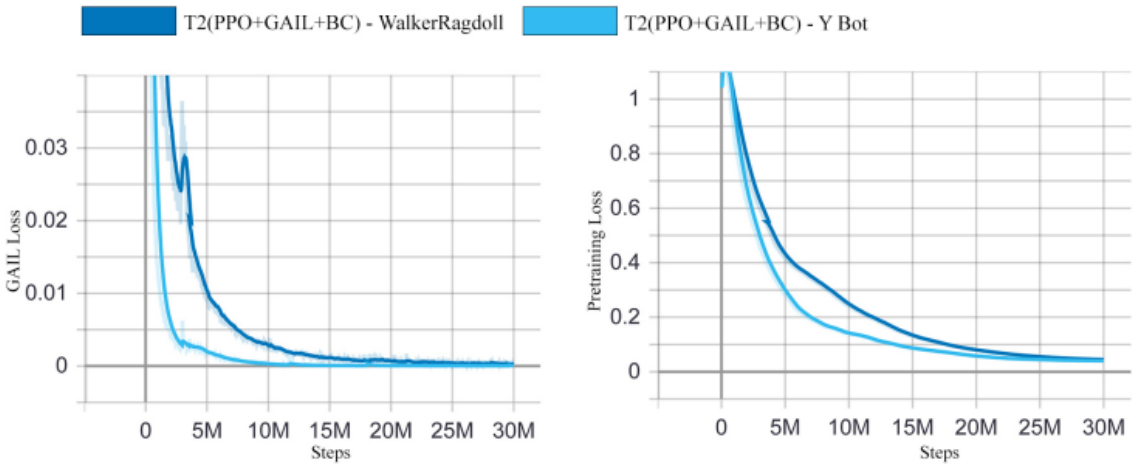


Рисунок 3.10 – Графіки функції втрат GAIL

Середнє значення функції втрат стратегії представлено графіком втрат стратегії, який показує, як часто змінюється політика. При успішному навчанні вона повинна зменшуватися в кінці, і на графіку показано, що на другому навчанні вона зменшується. Графік втрат до навчання генерується для поведінкового клонування, який представляє середню величину втрат при поведінковому клонуванні. І подібно до втрат GAIL, цей графік показує, наскільки добре модель імітує демонстраційні дані. Нарешті, графік втрат цінності, що представляє середню втрату оновлення функції цінності, повинен зростати під час навчання і зменшуватися, коли винагорода стабілізується. Цей графік відображає ефективність моделі в прогнозуванні вартості кожного стану.

ВИСНОВКИ

У результаті виконання завдань кваліфікаційної роботи були упорядковані та вдосконалені знання, практичні уміння і навички, набуті в Харківському Національному університеті радіоелектроніки на кафедрі штучного інтелекту.

Виконано аналіз традиційних видів анімації, багатьох моделей глибокого навчання для створення анімацій. Крім того, були розглянуті проблеми складності та різноманітності анімації.

Для проведення практичного дослідження був проведений аналіз предметної області, а саме застосування методів штучного інтелекту і методів машинного навчання для створення анімацій. Потім були проаналізовані методи і алгоритми навчання з підкріпленням та імітаційного навчання. Після проведеної підготовчої роботи методи були проведені практичні експерименти.

Був запропонований підхід для анімації персонажів, який поєднує алгоритми навчання з підкріпленням та імітаційного навчання. Спочатку модель навчилась лише за допомогою навчання з підкріпленням. Хоча цей сеанс навчання був швидшим, результати не були схожими на поведінку людини. Потім ми навчили агентів комбінацією алгоритмів глибокого навчання з підкріпленням та імітаційного навчання. Результати були набагато більш схожими на людські, навіть незважаючи на те, що час навчання збільшився.

Цей метод анімації персонажів доводить, що можна отримати анімацію персонажів, схожу на людську, без необхідності використовувати дорогі студії для анімації рухів і великі бази даних. Однак є проблеми пов’язані з налаштуваннями регдолла. Запропонований метод, працює з персонажами, які мають схожу гуманоїдну кісткову структуру, і налаштування регдолла також має бути точно таким же.

При правильному налаштуванні алгоритму глибокого навчання з

підкріпленням, проксимальної оптимізації стратегії, у поєднанні з імітаційним навчанням та використанням інструментарію Unity ML-Agents, генерувати анімацію персонажів у реальному часі буде простіше, ніж будь-коли, скоротивши час виробництва, витрати та ресурси до мінімуму. З подальшими дослідженнями та майбутніми розробками запропонований підхід дозволить створювати ще кращі анімації, що зробить його порівнянним або навіть кращим за сучасні методи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Thomas, Frank; Johnston, Ollie. Disney Animation: The Illusion of Life. Abbeville Press, 1981. 678 p.

2. Laybourne, Kit. The Animation Book: A Complete Guide to Animated Filmmaking – from Flip-books to Sound Cartoons to 3-D Animation. New York: Three Rivers Press, 1998. 834 p.

3. Culhane, Shamus. Animation: Script to Screen. St. Martin's Press, 1990. 320 p.

4. Sito, Tom. Moving Innovation: A History of Computer Animation. Massachusetts: MIT Press, 2013. 212 p.

5. Greg James, “Operations for Hardware-Accelerated Procedural Texture Animation” in “Game Programming Gems 2” ed. Mark DeLoura, Charles River Media, 2001. 497 p.

6. Hodgins, J. K., & Pollard, N. S. Adapting simulated behaviors for new characters. In Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, 2002. 143-150 p.

7. Miller R. K. Neural network. Future Technology Surveys, 1991.

8. Czarnowski I., Jędrzejowicz P. Population learning metaheuristic for neural network training. Neural networks and soft computing. Heidelberg, 2003. P. 161–166. URL: https://doi.org/10.1007/978-3-7908-1902-1_21 (дата звернення: 18.04.2023).

9. GAN（Generative adversarial nets）. Journal of japan society for fuzzy theory and intelligent informatics. 2017. Vol. 29, no. 5. P. 177. URL: https://doi.org/10.3156/jsoft.29.5_177_2 (дата звернення: 11.05.2023).

10. Tealab A. Time series forecasting using artificial neural networks methodologies: a systematic review. Future computing and informatics journal. 2018. Vol. 3, no. 2. P. 334–340. URL: https://doi.org/10.1016/j.fcij.2018.10.003 (дата звернення: 11.05.2023).

11. Variational autoencoder / L. Pinheiro Cinelli et al. Variational methods

for machine learning with applications to deep networks. Cham, 2021. P. 111–149. URL: https://doi.org/10.1007/978-3-030-70679-1_5 (дата звернення: 23.04.2023).

12. Duan A. A structured prediction approach to robot imitation learning : doctoral thesis. 2021. URL: http://hdl.handle.net/11567/1050082 (дата звернення: 11.05.2023). 13. Sanghi N. Markov decision processes. Deep reinforcement learning with python. Berkeley, CA, 2021. P. 19–48. URL: https://doi.org/10.1007/978-1-4842-6809-4_2 (дата звернення: 14.04.2023).

14. Zhang H., Zhang S. Multi-Agent reinforcement learning. Deep reinforcement learning. Singapore, 2020. P. 335–346. URL: https://doi.org/10.1007/978-981-15-4095-0_11 (дата звернення: 07.05.2023).

15. Neural state machine for character-scene interactions / S. Starke et al. ACM transactions on graphics. 2019. Vol. 38, no. 6. P. 1–14. URL: https://doi.org/10.1145/3355089.3356505 (дата звернення: 11.05.2023).

16. Coros S., Beaudoin P., van de Panne M. Robust task-based control policies for physics-based characters. ACM transactions on graphics. 2009. Vol. 28, no. 5. P. 1–9. URL: https://doi.org/10.1145/1618452.1618516 (дата звернення: 04.05.2023).

17. Even-Dar E. Reinforcement learning. Encyclopedia of algorithms. Boston, MA, 2008. P. 771–774. URL: https://doi.org/10.1007/978-0-387-30162-4_341 (дата звернення: 11.05.2023).

18. DeepMimic / X. B. Peng et al. ACM transactions on graphics. 2018. Vol. 37, no. 4. P. 1–14. URL: https://doi.org/10.1145/3197517.3201311 (дата звернення: 18.04.2023).

19. Locomotion skills for simulated quadrupeds / S. Coros et al. ACM transactions on graphics. 2011. Vol. 30, no. 4. P. 1–12. URL: https://doi.org/10.1145/2010324.1964954 (дата звернення: 11.05.2023).

20. Chapter 5: proximal algorithms. MM optimization algorithms.

Philadelphia, PA, 2016. P. 123–149. URL: https://doi.org/10.1137/1.9781611974409.ch5 (дата звернення: 11.05.2023).

21 Dai T., Liu H., Anthony Bharath A. Episodic Self-Imitation Learning with Hindsight. Electronics. 2020. Vol. 9, no. 10. P. 1742. URL: https://doi.org/10.3390/electronics9101742 (дата звернення: 11.05.2023).

22 A platform for building mobile virtual humans / A. W. Feng et al. Intelligent virtual agents. Cham, 2015. P. 310–319. URL: https://doi.org/10.1007/978-3-319-21996-7_33 (дата звернення: 01.05.2023).

23. Minton S. Journal of artificial intelligence research (journal of artificial intelligence). Morgan Kaufmann Pub, 1996. 600 p.

24. Neumann G., Osa T., Pajarinen J. An algorithmic perspective on imitation learning. Now Publishers Inc, 2018. 196 p.

25. Generative adversarial network for imitation learning from single demonstration / T. N. Duc et al. Baghdad science journal. 2021. Vol. 18, no. 4(Suppl.). P. 1350. URL: https://doi.org/10.21123/bsj.2021.18.4(suppl.).1350 (дата звернення: 02.05.2023).

26 Demonstration actor critic / G. Liu et al. Neurocomputing. 2021. Vol. 434. P. 194–202. URL: https://doi.org/10.1016/j.neucom.2020.12.116 (дата звернення: 23.04.2023).

ДОДАТОК А

Відомість кваліфікаційної роботи

Позначення	Найменування	Дод. відомості
	Текстові документи	
1.	Пояснювальна записка	_61_ с.
	Інші документи	
2.	Презентаційні матеріали	_20_ плакатів

.	.	Прізвище та ініціали.	Підп.	Дата			
Розробив		Волошинова Г.С.			Дослідження моделей навчання з підкріпленням для створення анімації рухів персонажів	Шифр групи	Код напр./спец.
Перевірив		Чала Л.Е.				СШІм-21-2	122
Н.контр.		Малєєва І.А.				ХНУРЕ кафедра ШІ	
Затв.		Філатов В.О.					