Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра Комп'ютерних інтелектуальних технологій та систем
(повна назва)

АТЕСТАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Модель згорткової мережі для мультиагентної кооперації

Виконав:
студент II курсу, групи КІТм-19-1
Тельний М.А.

Спеціальність
123 – Комп'ютерна інженерія
(код і повна назва спеціальності)

Тип програми Освітньо-професійна
Освітня програма
Комп'ютерні інтелектуальні технології
(повна назва освітньої програми)

Керівник: проф. Аксак Н.Г.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри КІТС _____ Руденко О.Г.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра Комп’ютерних інтелектуальних технологій та систем

Рівень вищої освіти другий (магістерський)

Спеціальність 123 – Комп’ютерна інженерія

Тип програми Освітньо-професійна

Освітня програма Комп'ютерні інтелектуальні технології
(повна назва)

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

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Тельному Максиму Андрійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Модель згорткової мережі для мультиагентної кооперації

затверджена наказом по університету від “ 11 ” листопада 2020 р. № 1582 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 10 грудня 2020 р.

3. Вхідні дані до роботи Мультиагентна кооперація
Згорткова мережа
Штучні нейронні мережі
Python

4. Перелік питань, що потрібно опрацювати в роботі
Аналіз проблемної області і постановка задачі
Модель згорткової мережі для мультиагентної кооперації
Програмна реалізація моделі згоркової мережі для мультиагентної кооперації
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) Слайдів 10

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Огляд стану проблеми та постановка задачі	11.11 – 12.11	
2	Аналіз літератури за напрямком магістерської	12.11 – 23.11	
	роботи		
3	Вибір методів рішення для реалізації та їх	24.11 – 30.11	
	обґрунтування		
4	Розробка моделі згорткової мережі для	01.12 – 02.12	
	мультиагентної кооперації		
5	Оформлення пояснювальної записки	02.12 – 10.12	
6	Оформлення графічної частини	10.12 – 11.12	

Дата видачі завдання 11 листопада 2020 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Аксак Н.Г.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка атестаційної роботи: 80 с., 16 рис., 4 табл., 3 дод., 29 джерел.

МУЛЬТИАГЕНТНА СИСТЕМА, РОЗПОДІЛЕНИЙ ШТУЧНИЙ ІНТЕЛЛЕКТ, АГЕНТ, МУЛЬТИАГЕНТНА КООПЕРАЦІЯ, ЗГОРТКОВА МЕРЕЖА.

Метою атестаційної роботи є розробка моделі згорткової мережі для мультиагентної кооперації.

У ході виконання атестаційної роботи були розглянуті актуальність проблеми, зроблений огляд сучасної літератури за темою, визначені терміни які стосуються до тематики роботи та розлянуті роботи за тематикою атестаційної роботи.

Розроблена модель на мові Python за допомогою інструментів TensorFlow та Keras, яка з точністю 83% розпізнає зображення.

ABSTRACT

Master’s thesis: 80 pages, 16 figures, 4 tables, 3 appendices, 29 sources.

MULTI-AGENT SYSTEMS, DISTRIBUTED ARTIFICIAL INTELLIGENCE, AGENT, MULTI-AGENT COOPERATION, CONVOLUTIONAL NETWORK.

The major goal of this thesis is to develop a model of a convolutional network for multi-agent cooperation.

In the course of the attestation work the urgency of the problem was considered, the review of modern literature on the topic was made, the terms related to the subject of the work were determined and the works on the subject of the attestation work were divided.

Developed a model in Python using TensorFlow and Keras tools, which recognizes images with an accuracy of 83%.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

MAS – мультиагентна система (англ., Multi-agent system)

CNN – згорткова нейронна мережа (англ., Convolutional neural network)

GPU – графічний процесор (англ., Graphical processing unit)

DGN – графічне зготкове навчання, що засноване на глибокій Q мережі (англ., Graph convolutional reinforcement learning based on deep Q network)

MARL – мультиагентне навчання (англ., Multi-agent reinforcement learning)

RNN – рекурентна нейронна мережа (англ., Reccurent neural network)

GCN – графічна згорткова мережа (англ., Graph convolutional network)

RRL – реляційне навчання (англ., Relational reinforcement learning)

ВСТУП

В даний час штучні нейронні мережі являють собою напрямок який дуже бурхливо розвивається. Штучні нейронні мережі вже зараз активно використовуються для вирішення таких завдань як класифікація і кластеризація даних, прогнозування, апроксимація, стиснення і архівація інформації, розпізнавання образів та інші. Рішення останнього завдання, а точніше розпізнавання зображень, потрібно в багатьох практичних областях, наприклад, в деяких сферах військової діяльності, в криміналістиці, робототехніці, роботі органів правопорядку та інших. Один з найбільш відомих і потрибних для практичного застосування видів оптичного розпізнавання – це розпізнавання зашумлених або рукописних символів (цифр і букв). Для вирішення такого завдання можуть бути використані штучні нейронні мережі Хопфілда, Хеммінга, мережі адаптивної резонансної теорії, мережі двобічної асоціативної пам'яті, багатошаровий персептрон, когнітрон і неокогнітрон і деякі інші.

Незважаючи на існування великої кількості різних алгоритмів, на даний момент не створено ефективного програмного продукту для розпізнавання рукописних і зашумлених текстів, придатного для масового використання. Саме тому розробка нових ефективних і швидкодіючих алгоритмів оптичного розпізнавання зображень, зокрема, заснованих на застосуванні штучних нейронних мереж, до сих пір визнана актуальною.

Нейронні мережі та нейрокомп'ютери – галузь знань, досить популярна в даний час. Це проявляється, зокрема, у великій кількості публікацій, конференцій і різних застосувань.

Одна з підстав такої популярності – їх чудові здібності до навчання за спостережуваними прикладами і формування прийнятних висновків на базі неповної, зашумленной і неточною вхідної інформації. Роботи по нейронних мережах спочатку були розпочаті біологами. За допомогою нейромереж

дослідники прагнули вивчити властивості і особливості роботи головного мозку.

З розповсюдженням технологій прийняття рішень, заснованих на аналізі даних та використанні алгоритмів машинного навчання, велику роль починає відігравати коректне розпізнавання та класифікація об’єктів на зображенні, яке має одночасно забезпечувати необхідну швидкодію для того, щоб результат роботи системи був вчасним та актуальним. Оскільки класичні нейронні мережі погано придатні для роботи з зображеннями через нездатність враховувати просторове та відносне положення об’єктів, ускладненість рівнів, а також через труднощі зі знаходженням набору даних, достатньо великого для того, щоб навчити мережу класичної побудови тощо.

Згорткові нейронні мережі значно полегшують роботу з зображеннями з уведенням понять операції згортки та фільтрів, внаслідок чого структура мережі полегшується та залишаються лише важливі зв’язки між нейронами.

Одними з найбільш потрібних розробок в області комп'ютерного зору в останні роки стали згорткові нейронні мережі. Починаючи з 2012 року, коли конкурс ImageNet (щорічна міжнародна Олімпіада комп'ютерного зору) виграв алекс Крижевський з модифікацією згорткової нейронної мережі, переможцями цього конкурсу аж до цього року є мережі, побудовані на базі згорткових. Варто відзначити, що завдяки новому підходу до архітектури мережі показник помилки класифікації був знижений з 26% до 15% в 2012, в першу появу подібної мережі. У 2017 році результат мережі переможця складає 0.23%. На даний момент безліч компаній використовують технологію глибокого навчання в якості однієї з фундаментальних складових своїх послуг. Так, наприклад, Facebook використовує нейронні мережі для своїх алгоритмів автоматичної маркування, Google для пошуку фотографій, Amazon для рекомендацій своїх продуктів і Instagram для організації пошукової інфраструктури.

1 АКТУАЛЬНІСТЬ ПРОБЛЕМИ НА ОГЛЯД ЛІТЕРАТУРИ

1.1 Сфери та методи використання нейронних мереж

Штучні нейронні мережі міцно увійшли в наше життя і в даний час широко використовуються при вирішенні найрізноманітніших завдань і активно застосовуються там, де звичайні алгоритмічні рішення виявляються неефективними або зовсім неможливими. у числі завдань, вирішення яких довіряють штучних нейронних мереж, можна назвати наступні: розпізнавання текстів, гра на біржі, контекстна реклама в інтернеті, фільтрація спаму, перевірка проведення підозрілих операцій по банківських картах, системи безпеки і відеоспостереження – і це далеко не все.

Штучні нейронні мережі, подібно до біологічних, є обчислювальною системою з величезним числом паралельно функціонуючих простих процесорів з безліччю зв'язків. Незважаючи на те, що при побудові таких мереж зазвичай робиться ряд припущень і значних спрощень, що відрізняють їх від біологічних аналогів, штучні нейронні мережі демонструють дивовижну число властивостей, властивих мозку, – це навчання на основі досвіду, узагальнення, витяг істотних даних з надлишковою інформацією.

Нейронні мережі можуть міняти свою поведінку в залежності від стану навколишнього середовища. Після аналізу вхідних сигналів вони самоналагоджуються і навчаються, щоб забезпечити правильну реакцію. Навучена мережа може бути стійкою до деяких відхилень вхідних даних, що дозволяє їй правильно «бачити» образ, що містить різні перешкоди і спотворення.

Сьогодні існує велика кількість різних конфігурацій нейронних мереж з різними принципами функціонування, які орієнтовані на рішення самих різних завдань. уже сьогодні штучні нейронні мережі використовуються в багатьох областях, але перш ніж їх можна буде застосовувати там, де на

карту поставлені людські життя або значні матеріальні ресурси, повинні бути вирішені важливі питання, що стосуються надійності їх роботи. Тому рівень допустимих помилок слід визначати виходячи з природи самого завдання. Деякі проблеми з аналізом питань надійності виникають через допущення повної безпомилковості комп'ютерів, тоді як штучні нейронні мережі можуть бути неточні навіть при їх правильному функціонуванні. Насправді ж комп'ютери, як і люди, теж можуть помилятися. Перші – в силу різних технічних проблем або помилок в програмах, другі – через неуважність, втоми або непрофесіоналізм. отже, для особливо критичних завдань необхідно, щоб ці системи дублювали і страхували один одного.

Останнім часом робляться активні спроби об'єднання штучних нейронних мереж і експертних систем. у такій системі штучна нейронна мережа може реагувати на більшість простих випадків, а всі інші передаються для розгляду експертній системі. В результаті складні випадки приймаються на більш високому рівні, при цьому, можливо, зі збором додаткових даних або навіть із залученням експертів.

Нейромережеві прикладні пакети, що розробляються низкою компаній, дозволяють користувачам працювати з різними видами нейронних мереж і з різними способами їх навчання. Вони можуть бути як спеціалізованими (наприклад, для передбачення курсу акцій), так і досить універсальними.

Області застосування нейронних мереж досить разнообразні – це розпізнавання тексту й мови, семантичний пошук, експертні системи і системи підтримки прийняття рішень, передбачення курсів акцій, системи безпеки, аналіз текстів. розглянемо кілька особливо яскравих і цікавих прикладів використання нейронних мереж в різних областях.

Техніка та телекомунікації. В 1996 році фірмою AACo замовленням NASA був розроблений експериментальний автопілотіруемий гіперзвукової літа LoFL. Літак мав довжину всього 2,5 метри і вагу в 32 кілограми та був призначений для дослідження нових принципів пілотування. LoFL використовував нейронні мережі, що дозволяють автопілоту навчатися,

копіюючи прийоми пілотування льотчика. Одна з найважливіших задач в області телекомунікацій, яка полягає в знаходженні оптимального шляху пересилання трафіку між вузлами, може бути успішно вирішена за допомогою нейронних мереж. В даному випадку необхідно брати до уваги те, що, по-перше, запропоноване рішення має враховувати поточний стан мережі, якість зв'язку і наявність збійних ділянок, а по-друге, пошук оптимального рішення повинен здійснюватися в реальному часі.

Інформаційні технології. Визначення тематики текстових повідомлень – ще один приклад успішного використання штучних нейронних мереж. Так, сервер новин Convectis був обраний в 1997 році компанією «PointCast», що була лідером персоналізованої доставки новин в інтернеті, для автоматичної рубрикації повідомлень за категоріями. Нейромережевий продукт «SelectCast» від «Aptex Software» дозволяв визначати область інтересів користувачів інтернету і пропонував їм рекламу відповідної тематики.

Економіка та фінанси. Нейронні мережі активно застосовуються на фінансових ринках. Наприклад, американські банки використовують нейромережеві передбачення з 1990 року, і вже через два роки після їх впровадження, за свідченням журналів «The Economist», автоматичний дилинг показував прибутковість 25% річних.

Класифікація зображень і сигналів за допомогою нейромереж. Самим тривіальним завданням, яку навчилися вирішувати за допомогою нейронних мереж, стала класифікація зображень (рисунок 1.1).

Рисунок 1.1 – Класифікація зображень

Класифікації за допомогою CNN активно застосовуються в медицині: можна навчити нейронну мережу класифікації хвороб або симптомів, наприклад, для МРТ-діагностики (рисунок 1.2).

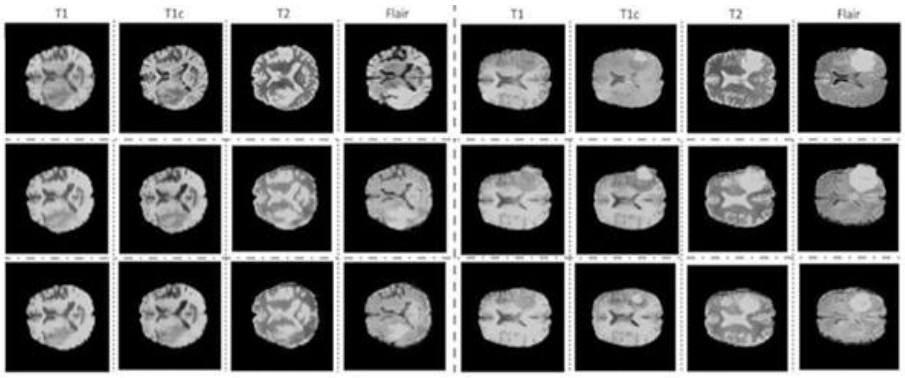

Рисунок 1.2 – МРТ

В агробізнесі розробляється і впроваджується методика аналізу і розпізнавання зображень, при якій дані отримують від відкритих супутників, таких як LSAT, і використовують для прогнозування майбутньої врожайності конкретних земель (рисунок 1.3).

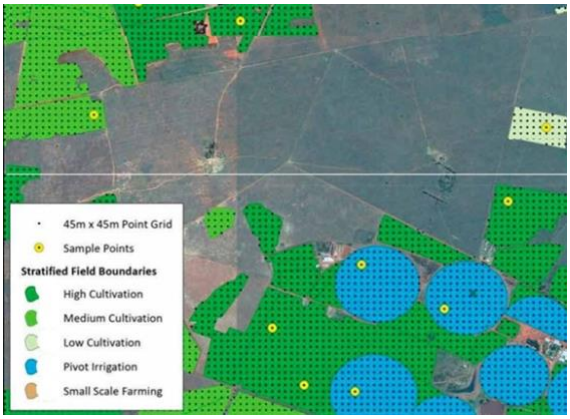

Рисунок 1.3 – Дані супутника

Розпізнавання об'єктів за допомогою нейронних мереж. Розпізнавання об'єктів на фото і відео за допомогою нейронних мереж (рисунок 1.4) застосовується в безпілотному транспорті, відеоспостереження, системи контролю доступу, системах "розумного будинку" і так далі.

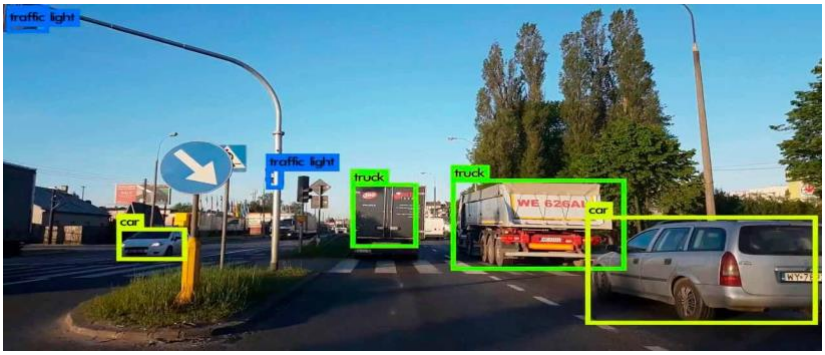

Рисунок 1.4 – Розпізнавання об'єктів на фото і відео

Зустрічається також масковане розпізнавання об'єктів з виділенням контуру, при якому ми також можемо отримувати чіткі контури об'єкта за допомогою згортальних нейронних мереж (рисунок 1.5).

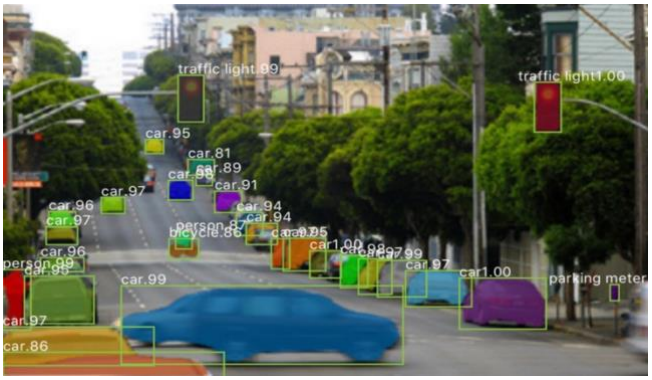

Рисунок 1.5 – Масковане розпізнавання об'єктів

Визначення марки і моделі конкретної техніки за допомогою машинного навчання.

Нейронні мережі для розпізнавання осіб і людей. Розпізнавання облич означає можливість виділяти особи на зображеннях (рисунок 1.6).

Рисунок 1.6 – Розпізнавання облич

Також нейронні мережі можна використовувати для виділення людей або окремих частин тіла людини на фото або відео, для побудови їх скелетів, поз, для відеоаналітики (рисунок 1.7).

Рисунок 1.7 – Виділення людей або окремих частин тіла

Тривимірна реконструкція осіб і об'єктів по фотографії за допомогою згортальних нейронних мереж. На даний момент існує кілька конкуруючих моделей, що дозволяють отримати тривимірні моделі особи (3DMM) всього по одній фотографії. Крім реконструкції осіб згорткові мережі застосовують також для реконструкції інших тривимірних об'єктів по фото (рисунок 1.8)

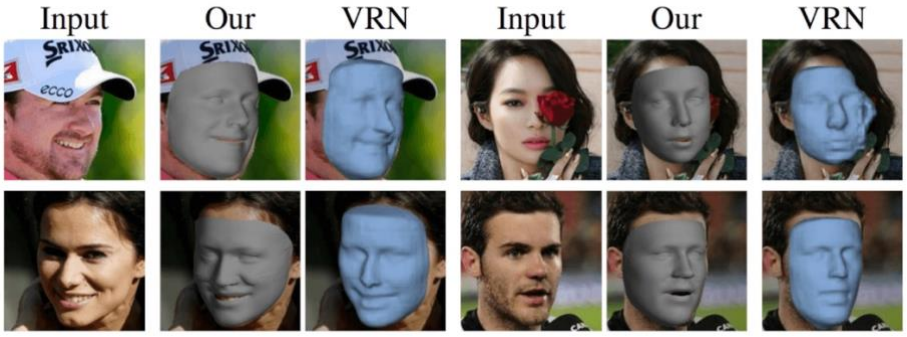

Рисунок 1.8 – Тривимірна реконструкція осіб

Розпізнавання мови і аналіз емоційної тональності тексту. Згорткові нейронні мережі можна застосовувати не тільки для вирішення завдань комп'ютерного зору. Наприклад, недавно Facebook AI Research виклала у відкритий доступ wav2letter ++ – свою технологію розпізнавання мови.

1.2 Перспективи розвитку і використання нейронних мереж

Потенціал у нейронних технологій величезний, але їх ефективне використання вимагає певного рівня знань і розуміння принципів їх дії. Нейронні мережі, на відміну від статистичних методів багатовимірного класифікаційного аналізу, базуються на паралельній обробці інформації і мають здатність до самонавчання, тобто отримувати обґрунтований результат на підставі даних, які не зустрічалися в процесі навчання. Ці властивості дозволяють нейронних мереж вирішувати складні (масштабні) завдання, які на сьогоднішній день вважаються важковирішуваними.

Основними перевагами нейронних мереж перед традиційними обчислювальними методами є:

- процес створення нейронної мережі відноситься до процесу навчання, ніж до програмування;
- гнучкість структури нейронних мереж дозволяє різними способами комбінувати прості складові нейрокомп'ютерів;
- нейронні мережі дозволяють створити ефективне програмне забезпечення для високопараллельних комп'ютерів;
- рішення задач в умовах невизначеності;
- стійкість до шумів у вхідних параметрах;
- адаптація до змін навколишнього середовища;
- потенційна відмовостійкість нейронних мереж.

Перевага використання нейронних мереж, як інструменту оцінки фінансово-економічного стану підприємства, полягає в тому, що взаємини між величинами заздалегідь не встановлюються, оскільки метод передбачає вивчення існуючих взаємозв'язків на готових моделях. Для нейронних мереж також не потрібно ніяких припущень щодо основного розподілу сукупності, а також, на відміну від багатьох традиційних статистичних методів, вони можуть працювати з неповними даними.

Нейронна мережа – паралельно розподілена структура обробки

інформації, що складається з нейронів, які з'єднані між собою зв'язками.

Нейронні мережі – потужний метод моделювання, що дозволяє відтворювати складні нелінійні залежності. Здатність до моделювання нелінійних процесів, роботи з зашумленими даними і адаптивність дають можливість застосовувати нейронні мережі для вирішення широкого класу економічних завдань.

Процес навчання в нейронних мережах стимулює бажані зразки активації і блокує небажані, грунтуючись на доступних даних. Для досягнення певного узагальнення знань в нейронної мережі розробляється алгоритм навчання.

Процес навчання намагається мінімізувати «дистанцію» між фактичними і бажаними виходами нейронної мережі. Протилежністю навчання з учителем є навчання без учителя. Коли використовується така парадигма, мається на увазі кілька зразків входу. Передбачається, що в процесі навчання нейронна мережа виявляє суттєві особливості входів. На відміну від навчання з учителем, тут не існує апріорного набору бажаних значень виходу. Нейронна мережа повинна розвинути власне уявлення стимулів входу без допомоги вчителя.

Це відкриває нові перспективи для систематизації численної експериментальної інформації в таких галузях знань, де традиційно важко приживається математичний формалізм, наприклад, в медицині, психології та історії.

1.3 Проблеми та недоліки згорткових мереж

За словами Джеффрі Хінтона, насправді прикро, що CNN працюють так добре, оскільки вони мають серйозні недоліки, від яких, на його думку, "важко буде позбутися". Нещодавно Хінтон висловив глибоку підозру щодо зворотного розвитку, сказавши, що він вважає, що це дуже неефективний спосіб навчання, оскільки він вимагає багато даних. У цій лекції Хінтон

вказує на деякі інші проблеми з CNN – погана поступальна незмінність та відсутність інформації про орієнтацію (або більш загально, що він називає "позою"). Інформація про позу стосується 3D-орієнтації щодо глядача, а також освітлення та кольору. Відомо, що CNN мають проблеми при обертанні об'єктів або при зміні умов освітлення.

Світові мережі використовують кілька шарів детекторів функцій. Кожен детектор функцій є локальним, тому детектори функцій повторюються в космосі. Об’єднання дає певну поступальну незмінність у набагато глибших шарах, але лише грубо. За словами Хінтона, психологія сприйняття фігури передбачає, що людський мозок досягає поступальної незмінності набагато кращим чином. Хінтон цього не обговорює, але ми знаємо, що, грубо кажучи, мозок має два окремі шляхи, шлях "що" та шлях "куди". Нейрони на шляху "що" реагують на певний тип подразника незалежно від того, де він знаходиться в полі зору. Нейрони на шляху “де” відповідають за кодування, де є речі. Як побічна примітка, передбачається, що шлях “де” має нижчу роздільну здатність, ніж шлях “що”.

Найголовніше, що тут слід визнати щодо шляху “що” – це те, що він не знає, де знаходяться об’єкти, не повідомивши про це шлях “де”. Якщо шлях людини, де “пошкоджено, він може визначити, чи присутній об’єкт, але не може відстежувати, де він знаходиться в полі зору і де він знаходиться відносно інших об’єктів. Це призводить до симультанагнозії, рідкісного неврологічного стану, коли пацієнти можуть сприймати лише один предмет одночасно.

Крайові детектори в першому шарі зорової кори (V1) не мають поступальної інваріантності – кожен детектор виявляє речі лише в невеликому зоровому полі. Те саме стосується CNN. Різниця між мозком та CNN відбувається на вищих рівнях. Хінтон стверджує, що мозок має модулі, які він називає "капсулами", які особливо добре справляються з різними типами зорових стимулів і кодують такі речі, як поза – наприклад, може бути один для автомобілів, а інший для облич. Мозок повинен мати механізм

"маршрутизації" зорової інформації низького рівня до того, що, на його думку, є найкращою капсулою для обробки. Зауважте, що якщо в мозку існує щось аналогічне капсулам, вони лежать у шляху “що”. Звичайно, у такій системі інформація «де» втрачається, але в більшості програм зору сьогодні це нормально.

За словами Хінтона, CNN здійснюють маршрутизацію шляхом об'єднання. Об’єднання було введено для зменшення надмірності подання та зменшення кількості параметрів, визнаючи, що точне розташування не є важливим для виявлення об’єкта. Однак, за словами Хінтона, інженери глибокого навчання насправді не замислювались над маршрутом конкретно. Пул робить маршрутизацію в дуже грубій формі, наприклад, максимальний пул просто вибирає нейрон з найвищою активацією, а не той, який, швидше за все, має відношення до поставленого завдання.

Ще однією відмінністю CNN від людського зору є те, що система людського зору накладає прямокутні координатні рамки на об'єкти. Деякі прості приклади, знайдені психологом Ірвінг Роком, такі:

Дуже грубо кажучи, квадрат і діамант виглядають як дуже різні фігури, тому що ми представляємо їх у прямокутних координатах. Якби вони знаходились у полярних координатах, вони б відрізнялися одним скалярним коефіцієнтом кутової фази, і їх числові подання були б дуже подібними. Так само нелегко сказати, на якому континенті справа фігура, вам доведеться спробувати обертати її подумки. Той факт, що мозок вбудовує речі в прямокутну систему координат, означає, що мозок легко керується лінійним перекладом, але обертання важко. Дослідження виявили, що розумова ротація вимагає часу, пропорційного величині необхідного обертання. CNN взагалі не можуть обробляти обертання - якщо вони тренуються на об’єктах в одній орієнтації, вони матимуть проблеми, коли орієнтацію змінять. За словами Хінтона, CNN взагалі не можуть виявляти "рукостискання", навіть в принципі. Іншими словами, CNN ніколи не могли відрізнити лівого взуття від правого, навіть якщо вони навчались на обох.

Подальше розглядаючи концепцію капсули та висловлюючись дуже гіпотетично, Хінтон припускає, що капсули можуть бути пов’язані з кортикальними мініколонками. Капсули можуть кодувати таку інформацію, як орієнтація, масштаб, швидкість та колір. Подібно нейронам у вихідному шарі CNN, капсула видає ймовірність присутності сутності, але додатково має прикріплені до неї метадані. Це дуже корисно, оскільки це може дозволити мозку зрозуміти, чи два об'єкти, такі як рот і ніс, є підкомпонентами основного об'єкта (обличчя). За словами Хінтона, мозок може мати досить високий розмірний "простір пози", приблизно 20 розмірів. Хінтон припускає, що легко визначити не випадкові пози у великих розмірах. Хінтон каже, що комп'ютерний зір повинен бути як зворотна графіка. Отже, в той час як графічний двигун множить матрицю обертання в рази на вектор, щоб отримати вигляд об’єкта в певній позі щодо глядача, система зору повинна мати зовнішній вигляд і повертати матрицю, що дає цю позу.

З точки зору точності класифікації система Хінтона працює приблизно так само, як CNN. Однак його система набагато повільніша, оскільки вона не має приємної властивості відповідати послідовності тензорних операцій (множення матриць), які мають CNN. Код Hintons був написаний на Matlab і не оптимізований для швидкості, тому майбутні реалізації, особливо з використанням графічних процесорів та іншого паралельного обладнання, можуть зробити їх цілком конкурентоспроможними з CNN.

1.4 Перспективи розвитку мультиагентних систем

Незважаючи на аспекти історії теорії та розвитку технологій MAS, вона все ще цілком вписується в класичну схему розвитку нових технологій і робить перші кроки від лабораторій до галузі, оскільки цей процес на практиці ніколи не є лінійним. На початку свого розвитку в 1990-х MAS став викликом у галузі програмного забезпечення і вимагав великих зусиль вчених та програмістів для перших успішних розробок – згадаймо сині

екрани Norton Commander у DOS та перші Інтернет-модеми.

Природно, не всі рішення виявилися успішними. Неважко передбачити, що процес розробки цього виду нової технології буде продовжувати бути вкрай нелінійним, і що це вимагатиме багаторазових спроб її трансформувати, і не всі з них були б успішними до того, як MAS-технологія стане справді зрілою, продуктивною та комерційною [3].

Розробники MAS зіткнуться з серйозними технічними, організаційними, комерційними та іншими проблемами через конкуренцію на ринку IT. Приклади таких проблем, наприклад, в управлінні ресурсами, включають:

- важко оцінити, наскільки ми далеко від "оптимального" зменшити пусковий струм;
- результати залежать від історії виникнення подій (немаркових процесів, чутливості до історії та ін.);
- «ефект метелика»: невеликий вхід, що веде до несподіваного великого виходу;
- реакцію системи можна несподівано уповільнити у разі переходу з одного атрактора на інший;
- у разі перезапуску системи результат планування можуть бути різними;
- важко повернути системні рішення (незворотність);
- взаємодія в режимі реального часу з користувачами стає більш досконалою;
- система може стати занадто "нервовою", планування та безліч обчислень із незначним впливом на оптимальність;
- системне рішення навряд чи можна пояснити користувачеві (втрата їдкості результатів);
- системне рішення навряд чи можна пояснити користувачеві (втрата їдкості результатів) [3].

Процес пошуку рішень у MAS розподілений та недетермінований за визначенням, якщо він базується на самоорганізації, що на практиці суперечить традиціям ієрархічного управління та, що більш важливо, традиціям проектування систем на принцип "зверху вниз", оскільки системи самоорганізації розробляються більше за принципом "знизу зверху" [3]. Однак тоді вирішення проблеми на будь-якому етапі може бути гармонізоване згори за відомим принципом Кауфмана - локальна взаємодія породжує глобальні структури, які впливають на початкові локальні взаємодії.

Також можливо сформувати гібридні моделі, в яких вихідне рішення знайдено традиційними пакетними методами, а потім воно модифікується залежно від обробки подій у MAS.

Комерційні та інженерні питання розробки MAS включають наступне:

- продаж інноваційних розробок MAS вимагає поглибленогозалучення експертів доменів, а не лише розробників. Більше того, це займає більші періоди часу (від 3 до 24 місяців);
- розробка важливих для бізнесу MAS-додатків (на основі досвіду великих проектів) вимагає збільшення зусиль та часу, приблизно в 3 – 5 разів більше, ніж очікувалося на початку;
- розробка зусиль для розробки та впровадження мультиагентної системи ("двигун") займає не більше 25% загального часу, а решта витрачається на інші питання, пов'язані з бухгалтерським обліком, базами даних, веб-інтерфейсом користувача, інтеграцією;
- розробка першої версії MAS має найбільш трудомісткий характер і займає від 3 до 6 місяців (мінімум), навіть маючи досвід використання MAS-технології [3];
- впровадження MAS часто забирає більше часу, ніж сама конструкція, оскільки вимагає виявлення та вивчення правил прийняття рішень та їх перевірки. Крім того, необхідна також інтеграція з існуючими інформаційними системами. Орієнтовне співвідношення затрат праці у % для

основних фаз проекту розробки MAS становить, в середньому, таке: проектування – 10%, розробка – 20%, тестування – 15%, доставка, впровадження та навчання – 35% та підтримка – 20%;

- розроблена система повинна «вижити» в умовах постійних помилок користувача, з неповними даними для проектування та отримання «поганих» даних;

- користувачі повинні мати можливість переробляти і доопрацьовувати рішення вручну, оскільки завжди є фактори, які не можна врахувати при автоматичному прийнятті рішень в системі.

Подолання негативних тенденцій у практичному використанні MAS, безумовно, вимагало б значних зусиль, але в разі успіху результати окупляться усіма вкладеннями, забезпечуючи на практиці переваги MAS-технологій. По суті, в даний час проводиться активна робота з усунення проблем MAS-технологій та подолання їх недосконалості [3].

Використовуючи ці результати, мультиагентні системи мають усі шанси досягти плато продуктивності на рівні галузевих стандартів, чого, мабуть, не можна очікувати раніше 2022 – 2027 років, і серед найбільш перспективних сфер їх використання можна передбачити наступне:

- аерокосмічна промисловість – колективна самоорганізуюча поведінка безпілотних космічних та повітряних апаратів, управління малими супутниковими групами, імітатори для пілотів та диспетчерів повітряного руху, космічна логістика тощо;

- мережі B2B виробничих та транспортних підприємств, стратегічне планування та управління виробничим виробництвом, логістика мережі (транспорт тощо) та управління ресурсами GRID, де агентурні рішення та технології вже використовуються, але все ще існує велике поле для використання мережі MAS-технологій;

- військові застосування – існує безліч непрямих ознак та свідчень діяльності щодо промислових розробок у цій галузі, але ця інформація є засекреченою;

- колективна робототехніка, автономні місії роботів. Фахівці з усього світу розглядають це як одне з найбільш перспективних напрямків для мультиагентних додатків;

- розумні мережі, віртуальні електростанції та інші програми в галузі виробництва електроенергії.

- охорона здоров'я – екологічні рішення для підтримки громадського здоров'я (проживання під впливом навколишнього середовища, особисте медичне обслуговування тощо);

- завдання в галузі Інтернет речей, де необхідно розвивати розподілені системи безпеки, сенсорні мережі, інтелектуальні простори тощо;

- мобільні додатки – за оцінками Gartner, до 40% майбутніх мобільних додатків протягом наступних 10 років будуть побудовані на основі MAS-технологій;

- віртуальні (накладені) мережі та однорангові (p2p) додатки (програмно-визначена мережа) – ця парадигма взаємодії розподілених додатків наразі користується величезним попитом.

1.5 Постановка задачі

Метою атестаційної роботи є розробка моделі згорткової мережі для мультиагентної кооперації.

Для досягнення даної мети необхідно вирішити наступні задачі:

- проаналізувати та розглянути сучасні роботи та публікації для визначення актуальності роботи;
- визначити сфери використання, переваги, нелодіки та перспективи розвитку нейронних мереж та мультиагентних систем;
- розглянути сучасні методи розробки нейронних мереж;
- провести аналіз моделей мультиагентної кооперації;
- реалізувати модель згоркової мережі для мультиагентної кооперації.

2 АНАЛІЗ ВИКОРИСТАННЯ АГЕНТНО-ОРІЄНТОВАНОГО МОДЕЛЮВАННЯ

2.1 Характеристика властивостей агента

Як і слід було очікувати від досить молодої галузі досліджень, ще немає універсального визначення агента. Однак визначення Вулдріджа та Дженнінгса все частіше приймається, і, мабуть, справедливо сказати, що більшість дослідників, коли їх питають про визначення говорять ті ж самі властивості, які були зазначені Вулдріджем та Дженнінгсом.

Агент – це комп'ютерна система, яка знаходиться в деякому навколишньому середовищі, і яка здатна до самостійних дій в цьому середовищі для досягнення своїх цілей проектування [1].

Вулдрідж розрізняє агента від розумного агента, від якого вимагається також реактивність, активність та соціальність з тих, про яких ми поговоримо нижче.

Спочатку зазначимо, що мова йде про програмних агентів. Кожного разу, коли ми (або будь-який інший з дослідників у цій галузі) кажуть «агент», ми справді маємо на увазі «агент програмного забезпечення». Типове словникове визначення агента, як суб'єкта, який має повноваження діяти від імені іншого (наприклад, агент з нерухомості) – це не те, що ми маємо на увазі.

Друга властивість (ситуальність) не дуже обмежує поняття агента оскільки практично все програмне забезпечення можна вважати таким, що знаходиться в оточенні. Однак, де агенти різняться, це тип середовища. Як правило, використовуються агенти де навколишнє середовище є складним; більш конкретно, типові середовища агента є динамічними, непередбачуваними та ненадійними. Ці середовища динамічні тим, що вони швидко змінюються. Під терміном «швидко» ми маємо на увазі, що агент не

може припустити, що це середовище залишатиметься статичним, поки він намагається досягти мети. Ці середовища непередбачувані, оскільки неможливо передбачити майбутні стани навколишнього середовища; часто це пов’язано з тим, що агент не може мати ідеальну та повну інформацію про навколишнє середовище, і тому, що довкілля модифікується не лише за межізнання та вплив агента. Нарешті, ці середовища в цьому ненадійні дії, які агент може виконувати, можуть зазнати невдачі з причин, які не входять до складу агента контроль. Наприклад, робот, який намагається підняти предмет, може зазнати невдачі в широкому діапазоні причини, включаючи надто важкий товар.

Агенти часто розташовані в динамічних середовищах, які швидко змінюються. Зокрема, це означає, що агент повинен реагувати на значні зміни у своєму середовищі. Наприклад, агент, який керує роботом, що грає у футбол, може складати плани на основі поточного положення м'яча та інших гравців, але він повинен бути готовим адаптуватися або відмовитись від своїх планів, якщо обстановка суттєво зміниться. Іншими словами, агентам потрібно своєчасно реагувати на зміни в навколишньому середовищі.

Ще однією ключовою властивістю агентів є те, що вони переслідують цілі з часом, тобто є ініціативними [1]. Однією з властивостей цілей є їх стійкість; це корисно тим, що робить агентів більш надійними: агент продовжуватиме намагатись досягти мети, незважаючи на невдачні спроби.

Незважаючи на те, що об’єкти можуть бути реагуючими і можуть розглядатися як ті, які мають неявну мету, вони є не ініціативними у тому сенсі, що має кілька цілей, і ці цілі є явними та наполегливими. Отже, активність – це ще одна властивість, яка відрізняє агентів від об’єктів.

Ключовим питанням в архітектурі агентів є збалансування реактивності та активності. З одного боку, агент повинен реактивним, тому на його плани та дії мають впливати зміни навколишнього середовища. З іншого боку, на плани та дії агента повинні впливати його цілі. Завдання полягає в тому, щоб збалансувати ці два (часто суперечливі) впливи: якщо агент занадто

реактивний, то він буде постійно коригувати свої плани і не досягати своїх цілей. Однак, якщо агент недостатньо реактивний, він витратить час на спроби дотримуватися планів, які вже не є актуальними чи застосованими.

Оскільки невдачі в діях і, загальніше, в планах, є можливим завданням середовищ, агенти повинні мати можливість відновлюватись після таких збоїв, тобто вони повинні бути міцними. Природним підходом до досягнення стійкості є гнучкість. Маючи діапазон шляхів досягнення заданої мети агент має альтернативи, які можна використовувати, якщо план не вдався. Ці дві властивості також є відмінними рисами агентів порівняно з об'єктами.

Нарешті, агентам майже завжди потрібно взаємодіяти з іншими агентами, тобто агенти є соціальними. Ця взаємодія часто на вищому рівні: замість того, щоб просто говорити, що агенти обмінювалися повідомленнями, взаємодія агента може бути сформульована з точки зору перформативів, таких як «інформувати», «запитувати», «погоджуватись» тощо. Вони мають стандартну семантику, яка визначена з точки зору їх впливу на психічний стан агента. Часто розглядається взаємодія агента з точки зору типів людської взаємодії, таких як переговори, координація, співпраця та колективна робота. На основі цих властивостей ми використовуємо наступне визначення.

Інтелектуальний агент – це частина програмного забезпечення, яка є ситуативною (існує в середовищі), автономною (незалежний, не контролюється зовні), реактивною (реагує на зміни у своєму навколишньому середовищі), проактивною (наполегливо переслідує цілі), ґнучкою (має кілька способів досягнення цілей), міцною (відновлюється після несправності), соціальною (взаємодіє з іншими агентами) [1].

Окрім цих властивостей, існує цілий ряд інших властивостей, які ми розглядаємо як менш центральні. Переслідуючи свої цілі, ми хочемо, щоб агенти були раціональними. Частино бути раціональним полягає в тому що агент не повинен робити «німих» речей, таких як одночасне проходження двох курсів дії, що конфліктують. Наприклад, плануючи витратити гроші на

відпочинок одночасно, як планувати витратити ті самі гроші на машину. Детальний аналіз того, що мається на увазі за "раціональним" можна знайти у праці Братмена.

Одне визначення агентів (сильний агент) також має ці різні властивості вимагає, щоб агенти розглядалися як такі, що мають психічні установки, такі як переконання, цілі та наміри [1]. Ця навмисна позиція має напрочуд прагматичне виправдання: оскільки система ускладнюється, її поведінку можна прогнозувати надійніше абстрагуючись від того, як він досягає своїх цілей, і замість цього міркувати про те, що це таке цілі та переконання. Наприклад, при спробі з'ясувати, чи предмет меблів буде підтримувати вагу людини, ми зможемо змоделювати напруження та розрахувати його несучу здатність, або ми могли б розглянути його дизайн і причину того, що метою стільця є сидіння і тому будь-який стілець повинен мати можливість витримати вагу людини. Колишня позиція є "фізичною позицією", остання – "дизайнерською позицією". «Навмисна позиція» – це розширення цього, яке застосовується до активних сутностей.

Хоча наявність агентів, які вчаться на своєму досвіді, для деяких людей може бути вкрай необхідним додатком, це може бути згубним для інших. Так само для програми, в яких моделювання людських емоцій може бути корисним, наприклад, інтерфейсні агенти або комп'ютерні ігри, але однаково, є багато додатків, у яких це не актуально. Існує ціла робота, присвячена мобільним агентам Однак між роботами над, напрочуд, мало перекривань розумні агенти та робота над мобільними агентами. Мобільність – це більше системний рівень з великою роботою, присвяченою таким питанням, як може бути запущена програма, зупинена, переміщена на іншу машину та перезапущена, та пов’язані з цим проблеми у безпеці [1].

2.2 Перспективи розвитку агентів

Важливо усвідомлювати, що, як і інші програмні технології, такі

об'єкти як агенти – це не магія. Вони є просто підходом до структурування та розробки програмного забезпечення який пропонує певні переваги, і який дуже добре підходить для певних типів програм (насправді одна точка зору вважає агентів еволюційним кроком вперед від об'єктів. Щоб зрозуміти, чому агенти корисні, нам потрібно зрозуміти які відмінні риси агентів перетворюються на властивості програмних систем, які розроблені та побудовані з використанням агентів. Корисність цих властивостей (таких як децентралізованість) залежить від програми, тому важливо також зрозуміти, як ці властивості програмної системи стосуються типів програм та областей застосування [1].

Мабуть, найголовнішою перевагою агентів є те, що вони зменшують зв'язок. Агенти є автономними, що може розглядатися як інкапсулюючий виклик. Тоді як об’єкт робить доступними методи, які запускаються зовні, агент не надає жодної контрольної точки зовнішнім суб’єктам. Коли повідомлення надісланє агенту, агент (будучи автономним) має контроль над тим, як він має справу з повідомленням.

Зв'язок зменшується не тільки завдяки інкапсуляції, що забезпечується автономністю, але і завдяки стійкості, реактивності та активності агентів. На агента можна покластися, щоб він наполегливо досягав своїх цілей, випробовуючи альтернативи, що відповідають змінам навколишнього середовищаа. Це означає, що коли агент бере мету то відповідальність за досягнення цієї мети лежить на цьому агенті. Постійного нагляду та перевірки немає потреби. Як аналогію розглядайте об’єкт як надійного працівника, якому не вистачає ініціативи та почуття відповідальності; нагляд за таким працівником вимагає значного спілкування. З іншого боку, агента можна розглядати як працівника, який має почуття відповідальності та проявляє ініціативу. Нагляд за таким працівником вимагає набагато менше спілкування, а отже, і менш зв’язку.

Зменшене сполучення може призвести до створення більш програмних систем, більш децентралізованих і більш змінних. Це призвело до

застосування агентів як архітектурний «клей» в цілому ряді програмних додатків. У цьому застосуванні агенти часто використовуються для «обгортання» застарілого програмного забезпечення.

Стверджувалося, що агенти добре підходять для розробки складних розподілених систем, оскільки вони забезпечують більш природну абстракцію та розкладання складних систем, що майже розкладаються [1].

Один дедалі важливіший клас систем, що демонструють децентралізацію, складність та розподіл – це відкриті системи: програмні системи, в яких розроблені різні частини і написані різними авторами, без спілкування. Прикладом є Всесвітня павутина, де автори веб-браузера та веб-сервера, ймовірно, ніколи не спілкувалися між собою. Не дивно, що стандарти відіграють ключову роль у забезпеченні програмного забезпечення, яке було розроблено самостійно для спільної роботи. Інтернет є простим прикладом відкритої системи, оскільки, по суті, цим займається транспортування статичних документів, на відміну від надання послуг, що змінюють стан серверів. Інші, більш складні, приклади таких систем включають семантичну павутину та обчислювальні мережі.

Окрім зменшеного зчеплення, агенти також чітко застосовуються в ситуаціях, коли навколишнє середовище є складним (динамічне, непередбачуване, ненадійне), в яких можливий збій і в якому необхідно зробити відновлення після відмови автономно. Надзвичайний приклад агентської системи, з якою потрібно було мати справу у таких ситуаціях був віддалений агент, який у травні 1999 році був під контролем NASA протягом двох днів, понад 96 500 000 кілометрів від Землі.

Будучи ініціативними та реактивними, агенти стають більш схожими на людей у тому, як вони мають справу проблемами. Це призвело до ряду програм, в яких програмні агенти використовуються як замінники людини в певних обмежених сферах. Однією із програм є використання програмного забезпечення агентів для заміни пілотів-людей у військових симуляціях. Інші, більш спокійні, програми включають розваги. Нещодавно комп'ютерна

гра Black & White використовувала агентів, спеціально заснованих на моделі Belief-Desire-Intention (BDI), яка широко використовується в середовищі агентів [1].

2.3 Цикл виконання агентів

Поняття дій, уявлень, подій, цілей, планів і переконань пов'язані з кожним інші через цикл виконання, який реалізує прийняття рішень агентом. Цикл виконання описує, як екземпляри цих концепцій взаємодіють під час виконання агентом. Наприклад, як сприйняття модифікує переконання, які в свою чергу впливають на вибір планів агента для досягнення своїх цілей.

Виконання агента слідує за циклом розуму-думки-дії, де мислиться частина циклу передбачає прийняття раціональних рішень. Цикл можна розглядати як такий, що складається з наступних кроків:

а) події обробляються для оновлення переконань та негайних дій;

б) цілі оновлюються: (тобто генеруються нові цілі, досягаються або ставляться неможливі цілі, визначаються пріоритети цілей);

в) плани вибираються з бібліотеки планів для досягнення цілей або обробки подій;

г) крок плану виконується в наступному плані, даючи нові події, (під) цілі, переконання зміни або дії.

Наприклад, розглянемо робота пожежної машини, який отримує об’єкт сприйняття, що містить інформацію про пожежу в певному місці. Оскільки агент не має відомих знань про пожежі, це сприйняття призводить до оновлення переконань агента знанням про нову пожежу. Це, в свою чергу, генерує мету загасити пожежу, яка в подальшому призводить до створення плану який обрається та виконується. Описаний цикл виконання є досить абстрактним. Звичайно, подробиці потрібно розробляти залежно від домену. Вилучення відповідної інформації з об’єкту сприйняття та належне оновлення переконань залежатиме від програми. Однак є загальні етапи, які є

частиною стандартного циклу виконання, що реалізує прийняття рішень BDIstyle.

Процес обробки події або спроби досягнення мети виконується такими кроками:

а) визначення відповідних планів з бібліотеки планів;

б) визначення підмножини відповідних планів, яка застосовується;

в) вибір один із застосовних планів;

г) виконання обраного плану.

План є актуальним, якщо в ньому зазначено, що він може досягти цілі, про яку йдеться. Відповідний план застосовується, якщо має сенс використовувати його в поточній ситуації. Це визначається за умови, що посилається на переконання агента. Цей стан відомий як умова контексту плану. Перевірка застосовності плану складається з перевірки чи істина умова контексту. Вибір плану з набору застосовних планів можна здійснити різними способами і це залежить від реалізації. Виконання плану може бути успішним, і в цьому випадку вважається, що (під) ціль була досягнута. Однак виконання плану також може провалитися. У цьому випадку, якщо агент намагається досягти мети, він розглядає альтернативні плани. У будь-якому випадку, якщо план не успішний – є інші плани для досягнення мети, тоді обирається та виконується альтернативний план.

2.4 Мультиагентні системи

Існує багато причин та переваг для вивчення та розробки систем з кількома агентами. Спочатку причиною було вивчення розподілених підходів до певного типу проблем. Насправді, незважаючи на те, що централізовані рішення, як правило, ефективніші, розподілені обчислення часом легше зрозуміти та розробити. Це має вирішальне значення, особливо коли проблема, яку потрібно вирішити, розповсюджується сама, наприклад, коли дані належать незалежним організаціям, які хочуть зберегти свою

інформацію в приватному та безпечному порядку з комерційних причин. Більше того, розподілений підхід описує важливу програмну інженерію системі, подібно до ефективності, насправді в деяких ситуаціях розподілений підхід прискорює вирішення проблем завдяки паралельному використанню ресурсів та надійності обидва вдосконалені, враховуючи те, що система стає стійкою до несправностей через надмірність.

Іншим цікавим аспектом розподіленої перспективи є можливість навчання нових підходів до вирішення певного типу проблем. Насправді розподіл може призвести до обчислювальних алгоритмів, які, можливо, не були б виявлені інакше і які часто є більш природним способом подання проблеми. Існуючий тип проблем, який експлуатується розподіленим підходом включає: маршрутизацію транспортних засобів серед незалежних диспетчерських центрів, планування виробництва, цифрові бібліотеки, збір багатоагентної інформації в Інтернеті, маршрутизація та розподіл пропускної здатності в багатопрофільних комп'ютерних мережах, електронна комерція та різні види планування, такі як планування роботи між багатьма компаніями, планування зустрічей, планування лікування пацієнтів у лікарнях, класі планування тощо, щоб назвати лише декілька [2].

Є важливі переваги у розробці систем, що складаються з декількох самозахоплених автономних агентів, що діють як особи, а не як частини цілої системи. Можливість агентів ефективного пошуку, фільтрації та обміну інформації, а також обміну знаннями, послугами, продуктами та досвідом з іншими агентами або навіть з людьми, дозволяє їм вирішувати проблеми, які неможливо вирішити самостійно. Здатність вести переговори, співпрацювати чи конкурувати з іншими агентами та міркувати про свої цілі або дії, які впливають на їх поведінку, дозволяє агентам мати змогу виконувати завдання які по суті вимагають взаємодії з іншими сутностями, як, наприклад, в електронній програмі пошуку інформації.

3 МОДЕЛЬ МУЛЬТИАГЕНТНОЇ КООПЕРАЦІЇ

3.1 Графічне згорткове навчання DGN

Кооперація є широко розповсюдженим явищем у природі від вірусів, бактерій та соціальних амеб до комашних товариств, соціальних тварин та людей [5]. За масштабами та масштабами співпраці людина перевершує всі інші види. Розвитку людської кооперації сприяє основний графік людських суспільств [6], де взаємна взаємодія між людьми абстрагується від їхніх стосунків.

Дуже важливо дати можливість агенту навчитися кооперуватися в мультиагентних середовищах для багатьох застосувань, наприклад, автономного водіння [7], управління світлофором [8], управління інтелектуальною сіткою [9] та управління кількома роботами [10]. Мультиагентне навчання (MARL), що сприяє спілкуванню [11], теорії середнього поля [12] та причинному впливу [13] були використані для співпраці між агентами. Однак спілкування між усіма агентами [11] ускладнює отримання цінної інформації для співпраці, тоді як спілкування лише з агентами поблизу [10] може стримувати діапазон співпраці. MeanField [12] фіксує взаємодію агентів середньою дією, але середня дія усуває різницю між агентами і тим самим спричиняє втрату важливої інформації, яка може допомогти кооперації. Причинний вплив [13] – це міра впливу дії, яка полягає у зміні політики агента за наявності дії іншого агента. Однак причинно-наслідковий вплив не пов'язаний безпосередньо з винагородою за довкілля і, отже, може не стимулювати співпрацю. Тим не менше, жодна з існуючих робіт не вивчає багатоагентну співпрацю з точки зору базового графіка, що потенційно може допомогти зрозуміти взаємну взаємодію агентів та сприяти їхній кооперації, як це відбувається в людській кооперації.

Розглянемо графічне згорткове навчання для мультиагентної

кооперації, де мультиагентне середовище моделюється як графік, кожен агент є вузлом, а кодування локального спостереження агента є особливістю вузла. Застосовуємо операції згортки до графіка агентів. Якщо декілько вузлів надати як ядро згортки [14], то графічна згортка здатна витягнути репрезентацію відношення між вузлами та згорнути елементи із сусідніх вузлів, як нейрон в згортковій нейронній мережі (CNN). Латентні риси, витягнуті з поступово зростаючих сприйнятливих полів, використовуються для вивчення політики співпраці. Градієнт агента не просто поширюється на себе, а й на інших агентів у його сприйнятливих полях, щоб підсилити вивчену кооперативну політику. Більше того, представлення відносин тимчасово регулюється, щоб допомогти агенту розробити послідовну політику співпраці.

Графічне згорткове навчання, а саме DGN, створюється на основі глибокої мережі Q, але його також можна реалізувати за допомогою градієнта політики або актора-критика. DGN навчається наскрізно, застосовуючи парадигму централізованого навчання та розподіленого виконання. Більше того, оскільки DGN ділить ваги між усіма агентами, його легко масштабувати, добре підходить для широкомасштабного MARL. DGN абстрагує взаємну взаємодію між агентами ядрами відношень, виділяє приховані риси за допомогою згортки та спонукає до послідовної співпраці шляхом регуляризації часових відносин. Покажємо ефективність навчання DGN у джунглях, бойових іграх та маршрутизації в мережах комутації пакетів. Продемонструємо, що агенти DGN здатні розробляти спільні та вдосконалені стратегії, а ефективність агентів DGN значно перевершує існуючі методи.

Згортання графіків значно покращує співпрацю агентів. На відміну від інших методів обміну параметрами, згортання графіка дозволяє градієнту одного агента перетікати до сусідів відповідно до їх внеску, сприяючи взаємодопомозі. Ядра взаємозв'язку допомагають створити політику в MARL, яка може краще узагальнити навколишнє середовище з набагато

більшою кількістю агентів, що дуже важливо для реальних додатків, оскільки ми можемо тренуватися з набагато меншою кількістю агентів і безпосередньо використовувати їх у великомасштабних середовищах. Тимчасова регуляризація, яка мінімізує розбіжність представлення відносин KL у послідовні періоди часу, стимулює співпрацю, допомагаючи агенту формувати довгострокову та послідовну політику у високодинамічному середовищі з великою кількістю рухомих агентів [15].

MADDPG [16] та COMA [17] є продовженням моделі актор-критик для мультиагентних середовищ, де MADDPG розроблений для змішаних кооперативних та конкурентних середовищ, а COMA пропонується вирішувати присвоєння мультиагентних кредитів у кооперативних умовах. Централізований критик, який бере за вхідні дані спостереження та дії всіх агентів, використовується в MADDPG та COMA, тоді як мережеві критики, які оновлюються за допомогою комунікацій, розглядаються в [18]. Однак усі ці три моделі повинні підготувати незалежну політичну мережу для кожного агента, який прагне вивчити політику, що спеціалізується на конкретних завданнях, легко переоцінюється за кількістю агентів і не масштабується.

Існує кілька моделей, які пропонуються для вивчення взаємодії між агентами шляхом спілкування. Ці моделі наскрізно піддаються зворотному розмноженню. CommNet [19] використовує постійний зв'язок для повноцінних завдань співпраці. На одному кроці зв'язку кожен агент надсилає свій прихований стан як повідомлення до каналу зв'язку, а потім усереднене повідомлення від інших агентів надходить на наступний рівень. BiCNet [20] використовує нейромережу, що відновлюється (RNN), як канал зв'язку для підключення політик та мереж цінностей кожного окремого агента. ATOC [21] та Tar-MAC [22] дозволяють агентам дізнатися, коли спілкуватися та кому надсилати повідомлення відповідно, використовуючи механізм уваги. Ці моделі спілкування доводять, що спілкування допомагає у співпраці. Однак повне спілкування є дорогим та неефективним, тоді як стримане спілкування обмежує коло співпраці.

Коли кількість агентів збільшується, навчання стає важким через прокляття розмірності та експоненціальне зростання агентних взаємодій. Замість того, щоб розглядати різні ефекти інших осіб на кожного агента, Mean-Field [12] апроксимує ефект інших осіб за їх середньою дією. Однак середня дія усуває різницю між цими агентами з точки зору спостереження та дії і тим самим спричиняє втрату важливої інформації, яка може допомогти у прийнятті спільних рішень. У роботі [13] агенти винагороджуються за те, що вони мали причинний вплив на дії інших агентів, де причинний вплив оцінювали за допомогою контрафактичних міркувань. Однак причинний вплив не пов'язаний безпосередньо з навколишнім середовищем і, отже, не може ефективно стимулювати співпрацю.

Жодна з існуючих робіт у MARL не вивчає основний графік мультиагентного середовища. Однак ми стверджуємо, що основний графік може значно сприяти співпраці між агентами, як це стосується людської співпраці [23].

3.2 Згорткові мережі з графами

Багато важливих реальних програм подаються у вигляді графіків, таких як соціальні мережі [24], мережі взаємодії білків [25] та 3D-хмара точок [26]. За останні пару років було створено декілька рамок для вилучення локально пов’язаних об’єктів з довільних графіків. Як правило, мета полягає в тому, щоб вивчити функцію функцій на графіках. Графік згорткової мережі (GCN) бере в якості вхідних даних матрицю функцій, яка узагальнює атрибути кожного вузла і виводить матрицю функцій на рівні вузла. Функція подібна до операції згортки в мережах CNN, де ядра обертаються по локальних регіонах вхідних даних для створення карт функцій.

Мережі взаємодії спрямовані на обґрунтування об’єктів, відносин та фізики в складних системах. Мережі взаємодії передбачають майбутні стани

та основні властивості, що схоже на спосіб мислення людини. Для моделювання взаємодій було запропоновано декілька рамок. У [27] фокусується на бінарних відносинах між сутностями. Модель обчислює ефект взаємодії та передбачає наступний стан, беручи до уваги взаємодію. VIN передбачає майбутні стани з необроблених візуальних спостережень. VAIN моделює мультиагентні відносини та передбачає майбутні стани з механізмом уваги.

Ідея навчання реляційному підкріпленню (RRL) полягає в поєднанні RL з реляційним навчанням, представляючи стани та політику, засновану на відносинах. Нейронні мережі можуть працювати на структурованих представленнях набору сутностей, нелокально обчислювати взаємодії на наборі векторів ознак та виконувати реляційні міркування за допомогою ітеративного передавання повідомлень [28]. Блок співвідношень, багатоголосова увага до продукту, вбудований у нейронні мережі для вивчення попарного представлення взаємодії.

3.3 Конволюційне навчання с графічним підкріпленням для мультиагентної кооперації

Побудуємо мультиагентне середовище як графік, де агенти в середовищі представлені вузлами графіка, і для кожного вузла є K ребер, пов'язані з його K найближчими сусідами (наприклад, з точки зору відстані чи інших показників, залежно від середовища). Це найближчі сусіди, що частіше взаємодіють і впливають один на одного. Більше того, у широкомасштабних мультиагентних середовищах враховувати вплив усіх агентів є дорогим і менш корисним, оскільки отримання великого обсягу інформації вимагає великої пропускної здатності та спричиняє велику обчислювальну складність, і агенти не можуть диференціювати інформацію із загальнодоступної інформації [21]. Крім того, оскільки згортка може поступово збільшувати сприйнятливе поле агента, сфера співпраці не

обмежується. Отже, ефективно розглядати лише K найближчих сусідів. На відміну від статичного графіка, що розглядається в GCN, графік багатоагентного середовища постійно змінюється з часом, коли агенти переміщуються або потрапляють/виходять із середовища. Отже, DGN повинен мати можливість адаптувати динаміку графіку та вчитися в міру розвитку мультиагентного середовища.

Розглянемо частково спостережуване середовище, де в кожному моменті часу кожен агент отримує локальне спостереження o_i^t , яке є властивістю вузла i на графіку, виконує дію a_i^t та отримує r_i^t винагороди. DGN складається з трьох типів модулів: кодера спостереження, згорткового рівня та мережі C, як показано на рисунку 3.1. Місцеве спостереження o_i^t кодується у вектор функції h_i^t за допомогою MLP для введення низьких розмірів або CNN для візуального введення. Згорнутий шар інтегрує вектори ознак у локальній області (включаючи вузол i та його сусідні) та генерує прихований вектор ознак h_i^t. Шляхом укладання більшої кількості звивин національних шарах сприйнятливе поле агента поступово розростається, де збирається більше інформації, і, отже, обсяг співпраці також може збільшуватися. Тобто за допомогою одного згорткового шару вузол i може безпосередньо отримувати вектори прихованих ознак від кодерів вузлів в одному стрибку (сусідів). Складаючи два шари, вузол може отримати вихід першого згорткового шару вузлів в одному стрибку, який містить інформацію з вузлів у двох стрибках. Однак більш згорнені шари не збільшать локальну область вузла , тобто вузол все ще спілкується лише зі своїми -сусідами. Ця помітна характеристика дуже важлива, оскільки ми розглядаємо децентралізоване виконання. DGN складається з трьох модулів: кодера, згорткового рівня та мережі Q. Всі агенти мають спільні ваги та градієнти, накопичені для оновлення ваг (рис. 3.1).

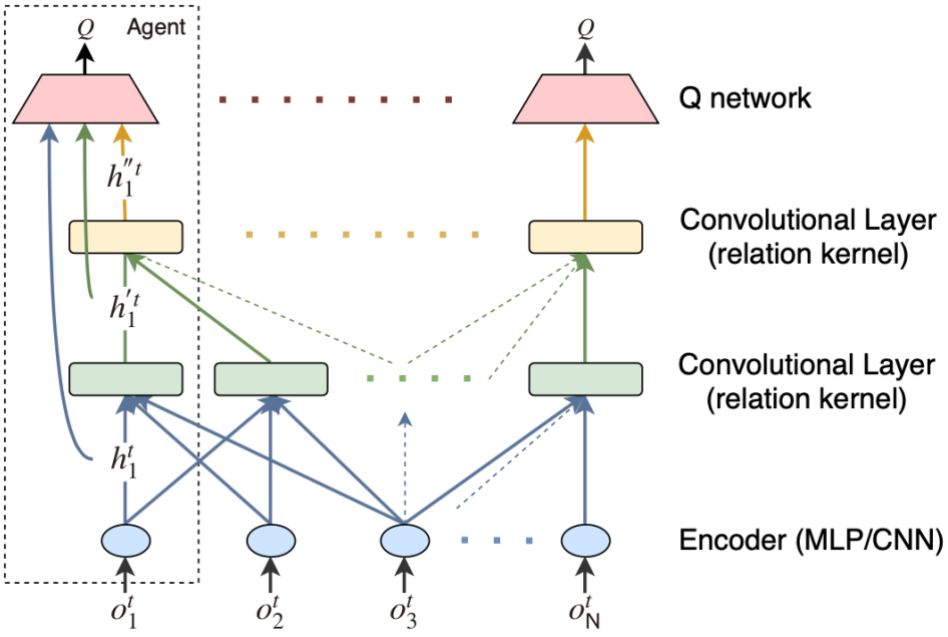

Рисунок 3.1 – Структура DGN

Оскільки кількість і положення агентів змінюються з плином часу, основний графік постійно змінюється, що призводить до труднощів до згортки графіка. Щоб вирішити цю проблему, об'єднуємо всі вектори функцій агентів в той час в матрицю функцій $\mathbf{F^t}$ з розміром $\mathbf{N \times L}$ в порядку індексу, де $\mathbf{N}$ - це кількість агентів, а - це довжина вектора функцій. Потім будуємо суміжну матрицю $\mathbf{C_i^t}$ з розміром $\mathbf{(K+1) \times N}$ для агента , де першим рядком є одногарячне представлення індексу вузла , а row, , . . , є одногарячим представленням індексу $\mathbf{j-1}$ найближчого сусіда. Потім можна отримати вектори функцій у локальному регіоні вузла $\mathbf{i}$ від $\mathbf{C_i^t \times F^t}$.

Для кожного агента функції всіх попередніх шарів об'єднуються та подаються в мережу, $\mathbf{Q}$ щоб зібрати та повторно використати представлення спостереження та особливості з різних рецептивних полів, які відповідно мають відмінні внески до стратегії, яка враховує співпрацю в різних сферах [27]. Мережа вибирає дію, яка максимізує значення з імовірністю 1 - ϵ або діє випадково з ймовірністю ε. Градієнт - втрат кожного агента буде поширюватися не тільки на себе та - сусідів, але й на інших агентів у його рецептивних полях. Тобто агент не лише зосереджується на максимізації

власної очікуваної винагороди, але також розглядає, як його політика впливає на інших агентів, а отже, агентам надається можливість навчитися співпраці. Більше того, кожен агент отримує кодування спостережень та намірів сусідніх агентів, що робить середовище більш стабільним з точки зору окремого агента [15].

У DGN усі агенти поділяють ваги. Однак це не заважає появі складних стратегій співпраці, як ми покажемо в експериментах. Розглянемо парадигму централізованого навчання та розподіленого виконання. Під час тренування на кожному кроці будемо зберігати кортеж (O, A, O′, R, C) у буфері відтворення , де = {o_1, ⋯, o_N}, = {a_1, ⋯, a_N}, O′ = {o'_1, ⋯, o'_N}, = {r_1, ⋯, r_N} і = {C_1, ⋯, C_N}. Потім відбиремо випадкову міні-партію зразків з і мінімізуємо втрати

$$L(\theta) = \frac{1}{S}\sum_{S}\frac{1}{N}\sum_{i=1}^{N}(y_i - Q(O_i, a_i; \theta))^2 \qquad (3.1)$$

,

де yi =ri+γmax$_{a'}$Q(O$_i'$,a$'_i$;θ′),Oi ⊆O позначає сукупність спостережень усіх агентів у рецептивних полях і, у - коефіцієнт знижки, а модель параметризована θ. Щоб зробити процес навчання більш стабільним, ми залишаємо C незмінним протягом двох послідовних кроків під час обчислення Q-втрати на тренуванні. Градієнти Q-втрат усіх агентів накопичуються для оновлення параметрів. Кожен агент не тільки мінімізує власні Q-втрати, але і Q-втрати інших агентів, з якими агент співпрацює. Потім ми м'яко оновлюємо цільову мережу як θ ′ = βθ + (1 - β) θ ′. Під час виконання кожен агент вимагає лише інформацію від своїх сусідів K (наприклад, за допомогою зв'язку), незалежно від кількості агентів [15]. Тому наша модель може легко масштабуватися і, отже, підходить для широкомасштабного MARL.

Ядра згортки інтегрують інформацію в рецепційне поле для вилучення прихованої ознаки. Одне з найважливіших властивостей полягає в тому, що

ядро має бути незалежним від порядку вхідних векторів ознак. Середня робота, як у CommNet, відповідає цій вимозі, але це призводить лише до незначного приросту продуктивності. BiCNet використовує ядро, яке можна вивчити, тобто RNN. Однак порядок введення векторів характеристик суттєво впливає на продуктивність, хоча ефект амортизується механізмом двонаправлення. Крім того, ядра згортки повинні мати можливість навчитися абстрагувати зв'язок між агентами, щоб інтегрувати їх вхідні функції [15].

Приймаючи ідею від RRL, будемо використовувати багатоголову увагу до продукту як згорткове ядро для обчислення взаємодій між сутностями. На відміну від RRL, приймаємо кожен агент, а не піксель, як сутність. Для кожного агента і існує набір об'єктів Ei (K сусідів і сам) у локальному регіоні. Вхідна особливість кожної сутності проектується на запит, подання ключа та значення кожною незалежною головою уваги. Відношення між i та j ∈ Ei обчислюється як (3.2)

$$a_{ij}^{m}=\frac{\exp\left(\tau * W_q^m h_i * (W_k^m h_j)^T\right.}{\sum_{e\epsilon\varepsilon_i}\exp\left(\tau * W_q^m h_i * (W_k^m h_e)^T\right.} \tag{3.2}$$

де τ - коефіцієнт масштабування. Значення всіх вхідних ознак зважуються відношенням і підсумовуються разом. Потім виходи для агента і об'єднуються, а потім подаються у функцію σ, тобто одношаровий MLP з нелінійністю ReLU, щоб отримати вихід згорткового шару,

$$h_i' = \sigma(Concat[\sum_{j\epsilon\varepsilon_i} a_{ij}^m W_v^m h_j, \forall m \in M] \tag{3.3}$$

Формула 3.3 ілюструє обчислення згорткового шару з ядром відношення.

Ілюстрація обчислення згорткового шару з ядром відношення для

агента з двома сусідами наведена на рисунку 3.2.

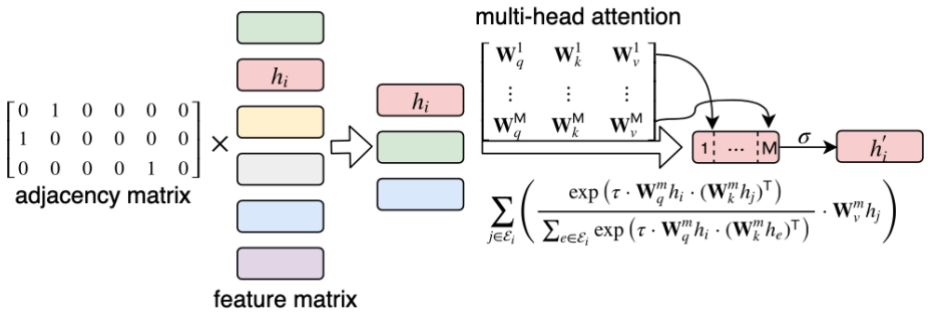

Рисунок 3.2 – Обчислення згорткового шару

Оскільки у ядрі відношення представлене як розподіл ваги уваги до стану оточуючих агентів, ми використовуємо розподіл ваги уваги в наступному стані як цільовий для поточного розподілу ваги уваги, щоб спонукати агента сформувати кон- представлення стійких відносин. Оскільки співвідношення в різних станах не повинно бути однаковим, але подібним, ми використовуємо розбіжність KL для обчислення відстані між розподілами ваги уваги в двох станах.

Регуляризація тимчасових відносин верхнього рівня в DGN допомагає агенту формувати довгострокову та послідовну політику дій у високодинамічному середовищі з великою кількістю рухомих агентів. Це додатково допоможе агентам сформувати кооперативну поведінку, оскільки для багатьох завдань співпраці потрібні довготривалі послідовні дії співпрацюючих агентів, щоб отримати остаточну винагороду.

3.4 Експериментальне моделювання

Для експериментів обрана мережева платформа MAGent. В навколишньому середовищі кожен агент відповідає одній сітці та має місцеве спостереження, яке містить квадратний вигляд із сітками 11 × 11 з центром у агента та його власні координати. Дискретні дії рухаються або атакують. Для

вивчення співпраці між агентами розглядаються два сценарії - «джунглі» та «битва». Крім того, створено середовище, яке імітує маршрутизацію в мережах комутації пакетів, щоб перевірити придатність нашої моделі в додатках для читання. Ці три сценарії проілюстровані на рисунку 3.3: джунглі (ліворуч), де агент отримує винагороду, харчуючись добре, але отримує вищу винагороду, атакуючи інших агентів; битва (в середині), де агенти співпрацюють для боротьби з більш могутніми ворогами; маршрутизація (праворуч), де агенти намагаються оптимізувати середню затримку пакетів даних, визначаючи лише наступний стрибок пакета на маршрутизаторі. В експериментах порівняємо DGN із незалежними DQN та MeanField Q-learning (MFQ) [12].

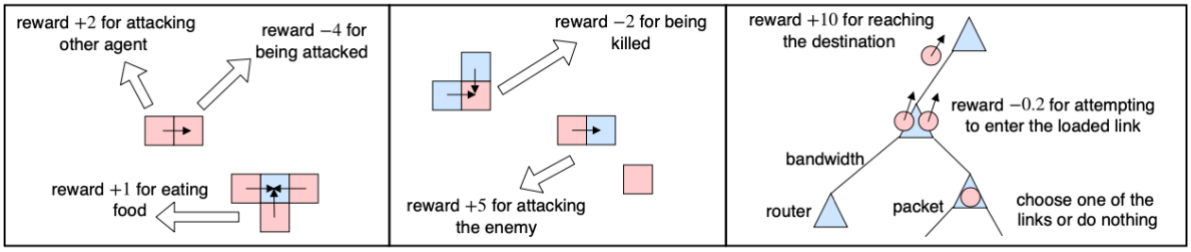

Рисунок 3.3 – Ілюстрація експериментальних сценаріїв

Джунглі. У полі є N агентів та L продуктів, де продукти стаціонарні. Агент отримує позитивну винагороду, вживаючи їжу, але отримує вищу винагороду, нападаючи на іншого агента. На кожному кроці часу кожен агент може перейти або атакувати одну з чотирьох сусідніх сіток. Винагорода становить 0 за пересування, +1 за напад (з'їдання) їжі, +2 за напад на іншого агента, –4 за напад і –0.01 за напад на порожню сітку (гальмування надмірних атак). Цей експеримент полягає у дослідженні того, чи можуть агенти навчитися стратегії спільного обміну ресурсами, а не нападати один на одного [15].

Були навучені всі моделі в умовах N = 20 та L = 12 для 2000 серій. На рисунку 3.4а показані їхні криві навчання, де DGN-M – це згортання графіка із середнім ядром замість ядра відношення, і кожна модель має три навчальні

прогони. У таблиці 3.1 наведено середню винагороду (усереднену за всіма агентами та кроками часу) та кількість атак між агентами (усереднену для всіх агентів) за 30 тестових запусків, кожна гра розгорнута зі 120 кроками.

DGN перевершує всі базові показники під час навчання та тестування за середньою винагородою та кількістю атак між агентами. Відзначається, що агенти DGN можуть правильно вибирати їжу, яка близька їй, і рідко шкодять одне одному, а їжа може бути раціонально розподілена оточуючими агентами. Більше того, атаки між агентами DGN набагато менше, ніж інші, тобто в 2 рази і 3 рази менше, ніж DGN-M та MFQ, відповідно. Прихована атака, запеклий конфлікт та вагання є характеристиками агентів CommNet та DQN, що підтверджує їх неспроможність навчитися співпраці. Хоча DGN-M і CommNet обидва використовують середню роботу, DGN-M істотно перевершує Comm-Net. Більше того, порівнюючи DGN з DGN-M, ми можемо зробити висновок, що ядро відношень, яке абстрагує представлення відносин між агентами, допомагає вивчити стратегію співпраці.

Таблиця 3.1 – Ілюстрація середньої винагороди

(N,L)		DGN	DGN-M	MFQ	CommNet	DQN
(20,12)	mean reward	0.70	0.66	0.62	0.30	0.24
	#attacks	0.91	1.89	2.74	5.44	7.35
(50,12)	mean reward	0.67	0.63	0.57	0.27	0.20
	#attacks	0.91	1.88	3.13	6.35	9.02

Застосовуємо навчену модель з N = 20 та L = 12 до сценарію N = 50 та L = 12. Більша щільність агента та нестача їжі ускладнюють моральну дилему. Незначне падіння середньої винагороди для всіх моделей пов’язано з тим, що їжі недостатньо для забезпечення кожного агента. DGN підтримує кількість атак, що означає, що агенти DGN все ще можуть раціонально розподіляти продукти, навіть коли їжі недостатньо для забезпечення кожного агента [15]. Однак агенти MFQ, CommNet та DQN частіше атакують один одного, коли більше агентів ділиться їжею.

Битва. Цей сценарій є цілком спільним завданням, коли N агенти

вчаться боротися проти L ворогів, які мають вищі здібності, ніж агенти. Діапазон руху або атаки агента - це чотири сусідні сітки, проте ворог може перейти до однієї з дванадцяти найближчих сіток або атакувати одну з восьми сусідніх сіток. Кожен агент/ворог має шість очок влучення (тобто, вбиваючись шістьма атаками) Нагорода складає +5 за напад на ворога, -2 за вбивство та -0.01 за атаку на порожню сітку. Після смерті агента/ворога баланс буде легко втрачено, а отже, ми додамо нового агента/ворога у випадковому місці, щоб підтримувати баланс. Таким чином, ми можемо зробити справедливе порівняння між різними методами з точки зору вбивств, смертей та співвідношення вбивств і смертей, крім винагороди за певні кроки часу. Попередньо навчена модель DQN, вбудована в MAgent, виконує роль ворога. Оскільки індивідуальний ворог набагато сильніший за окремого агента, агент повинен співпрацювати з іншими для вироблення узгодженої тактики боротьби з ворогами [15]. Більше того, оскільки точка враження ворога становить шість, агенти повинні постійно співпрацювати, щоб убити ворога. Тому завдання набагато складніше, ніж джунглі, з точки зору навчання співпраці.

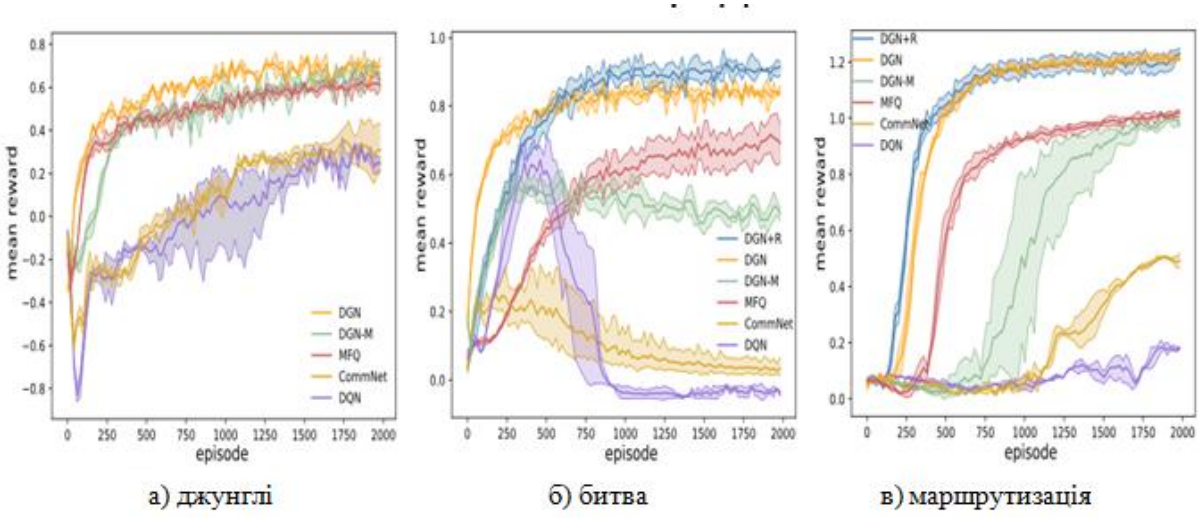

Рисунок 3.4 – Криві навчання всіх моделей з точки зору середньої винагороди

Були навчені всі моделі з налаштуванням N = 20 та L = 12 для 2000 серій. рисунок 3.4 показує криві навчання всіх моделей з точки зору середньої винагороди. DGN сходить до набагато вищої середньої винагороди, ніж інші базові лінії, і його крива навчання є стабільнішою. Для CommNet та DQN вони спочатку отримують відносно високу винагороду, але з часом вони сходяться до значно нижчої винагороди, ніж інші. Як спостерігалося в експерименті, на початку тренінгу DQN та CommNet вивчають такі неоптимальні політики, як збирання в групу в кутку, щоб уникнути нападу, оскільки така поведінка приносить відносно високу винагороду. Однак, оскільки розподіл винагороди нерівномірний, тобто на агентів з зовнішньої сторони групи легко нападати, навчаючись на досвіді з низьким рівнем винагороди, виробленому політикою субоптимуму, DQN та CommNet сходяться до більш пасивної політики, які призводять до значно нижчої винагороди. Ми оцінюємо DGN та базові лінії, проводячи 30 тестових ігор, кожна гра розгортається з 300 кроками. У таблиці 3.2 наведено середню винагороду, вбивства, смерті та коефіцієнт вбивства-смерті [15].

Таблиця 3.2 – Середня винагорода, вбивства, смерті та коефіцієнт вбивства-смерті

	DGN+R	DGN	DGN-M	MFQ	CommNet	DQN
mean reward	0.91	0.84	0.50	0.70	0.03	-0.03
#kills	220	208	121	193	7	2
#deaths	97	101	84	92	27	74
Kill-death ratio	2.27	2.06	1.44	2.09	0.26	0.03

Агенти DGN вивчають низку тактичних маневрів, таких як оточення та огородження одного флангу. Для одного ворога агенти DGN вчаться оточувати і атакувати його разом. Для групи ворогів агенти DGN вчаться рухатись і атакувати один із відкритих флангів ворога. Агенти CommNet приймають активну оборонну стратегію. Вони рідко починають атаки, а скоріше тікають або збираються разом, щоб уникнути нападу. Агенти DQN, керовані власними інтересами, не засвоюють раціональної політики.

Зазвичай їх збивають у кут і пасивно реагують на атаку противника. Агенти MFQ ефективно не співпрацюють між собою, оскільки серед агентів немає зворотного розповсюдження градієнтів для посилення співпраці.

У DGN ядра відношень можуть витягувати відносини високого порядку між агентами з поступово зростаючих сприйнятливих полів, які можна легко використати для отримання співпраці. Більше того, градієнтне зворотне розповсюдження від агента до інших агентів у рецептивному полі посилює співпрацю. Отже, DGN перевершує інші базові показники.

DGN з регуляризацією часових відносин, тобто DGN + R, досягає стабільно кращих показників порівняно з DGN, як показано на рисунку 3.4(b) та таблиці 3.2. В експерименті спостерігається, що агенти DGN + R дійсно поводяться більш послідовно та синхронно з кожним інше, тоді як агенти DGN частіше відволікаються на появу ворога чи друга поблизу та відмовляються від початкової запланованої траєкторії. Це призводить до меншої кількості випадків успішного формування оточення рухомого ворога, якому може знадобитися послідовна співпраця агентів для пересування по полю. Агенти DGN + R часто долають таку відволікання і демонструють більш довгострокову стратегію і націлюються, рухаючись більш синхронно, щоб переслідувати ворога, поки його не оточать і не знищать. З цього експерименту ми бачимо, що регуляризація часових відносин дійсно допомагає агентам формувати більш послідовну співпрацю [15].

Маршрутизація. Цей сценарій є абстракцією маршрутизації в мережах комутації пакетів, де протокол маршрутизації намагається оптимізувати середню затримку пакетів даних, приймаючи розподілене рішення на кожному маршрутизаторі (тобто, визначаючи лише наступний стрибок пакета на маршрутизаторі). Мережа складається з L маршрутизаторів. Кожен маршрутизатор випадковим чином підключений до постійної кількості маршрутизаторів (три в експерименті), а топологія мережі нерухома. Пропускна здатність кожної лінії збігається і встановлюється рівною 1. Існує N пакетів даних із випадковим розміром від 0 до 1, і кожному пакету

випадковим чином присвоюється маршрутизатор джерела та призначення. Якщо є кілька пакетів із розміром суми, більшим за 1, вони не можуть проходити за посиланням одночасно.

В експерименті пакети даних є агентами, і вони спрямовані на швидке досягнення пункту призначення, уникаючи при цьому заторів. На кожному кроці часу спостереження за пакетом - це власні атрибути (тобто поточне місцезнаходження, адресат та розмір даних), атрибути кабелів, підключених до його поточного розташування (тобто навантаження, довжина), та сусідні пакети даних (на підключений кабель або маршрутизатори). Потрібно кілька кроків часу, щоб пакет даних пройшов через кабель, лінійна функція довжини кабелю. Простір дій пакета - це вибір наступного стрибка. Якщо посилання на вибраний наступний стрибок буде перевантажено, пакет даних залишиться на поточному маршрутизаторі і буде покараний винагородою -0,2. Як тільки пакет даних прибуває в пункт призначення, він залишає систему і отримує винагороду +10, а інший пакет даних надходить у систему із випадковою ініціалізацією [15].

Були навчені всі моделі з налаштуванням N = 20 та L = 20 для 2000 серій. На рисунку 3.4(c) показано криві навчання з точки зору середньої винагороди. DGN та DGN + R сходяться до набагато вищої середньої винагороди та швидше, ніж вихідні показники.

DGN-M та MFQ мають подібну середню винагороду в кінці, хоча MFQ сходиться швидше, ніж DGN-M. Як і слід було очікувати, DQN працює гірше, що набагато нижче, ніж інші.

Оцінемо всі моделі, виконуючи 10 тестових ігор, кожна гра розгортається з 300 кроками. У таблиці 3.3 наведено середню винагороду, середню затримку пакетів даних та пропускну здатність, де затримка пакета вимірюється кроками часу, зробленими від джерела до пункту призначення, а пропускна здатність - це кількість доставлених пакетів за крок часу. Під час тестування, ми безпосередньо обчислюємо середню затримку на основі найкоротшого шляху кожного пакету, який становить 6.3. Зверніть увагу, що

ця затримка не враховує обмеження пропускної здатності (тобто пакети даних можуть одночасно проходити через будь-яке посилання). Таким чином, це ідеальний випадок для проблеми маршрутизації. Розглядаючи обмеження пропускної здатності, ми дозволяємо кожному пакету йти своїм найкоротшим шляхом, і якщо посилання перевантажене, пакет буде чекати біля маршрутизатора, поки посилання не буде розблоковано. Отримана затримка становить 8,7 що можна розглядати як практичне рішення.

Таблиця 3.3 – Середня винагорода, середня затримка пакетів даних та пропускна здатність

(N,L)		Floyd	Floyd with BL	DGN+R
(20,20)	mean reward			1.23
	delay	6.3	8.7	8.0
	throughput	3.17	2.30	2.50
(40,20)	mean reward			0.86
	delay	6.3	13.7	9.8
	throughput	6.34	2.91	4.08
(40,20) retrained	mean reward			0.94
	delay	6.3	13.7	10.2
	throughput	6.34	2.91	3.92

Таблиця 3.4 – Середня винагорода, середня затримка пакетів даних та пропускна здатність

(N,L)		DGN	DGN-M	MFQ	CommNet	DQN
(20,20)	mean	1.21	0.99	1.02	0.49	0.18
	reward	8.1	9.8	9.4	18.6	46.7
	delay	2.47	2.04	2.13	1.08	0.43
	throughput					
(40,20)	mean	0.83	0.70	0.78	0.39	0.12
	reward	10.0	12.7	11.8	23.5	83.6
	delay	4.00	3.15	3.39	1.70	0.49
	throughput					
(40,20) retrained	mean	0.90	0.78	0.76	0.35	0.05
	reward	10.5	12.2	12.8	21.2	112.2
	delay	3.81	3.27	3.12	1.86	0.35
	throughput					

Як показано у таблицях 3.3 та 3.4, продуктивність DGN-M, MFQ, CommNet та DQN гірша, ніж Floyd з BL. Однак затримка та пропускна здатність DGN набагато кращі за інші моделі, а також кращі за Floyd з BL. В експерименті спостерігається, що агенти DGN, як правило, вибирають найкоротший шлях до пункту призначення, і що ще цікавіше, вчаться вибирати різні шляхи, коли має відбутися затор. Агенти DQN не можуть навчитися найкоротшому шляху через короткозорість і легко спричиняють застійні явища на деяких зв'язках, не враховуючи впливу інших агентів. Обмін інформацією справді допомагає, оскільки DGN-M, MFQ та CommNet перевершують DQN. Однак вони не в змозі розробити складний протокол маршрутизації, як це робить DGN. DGN + R має трохи кращу продуктивність, ніж DGN. Це пов'язано з тим, що пакети даних з різними адресами рідко безперервно співпрацюють (обмінюючись багатьма посиланнями) на своїх шляхах.

Щоб дослідити, як модель трафіку впливає на продуктивність моделей, ми проводимо експерименти з більшим трафіком даних, тобто N = 40 і L = 20, де всі моделі безпосередньо застосовуються до налаштування без перекваліфікації. У таблицях 3.3 та 3.4 ми бачимо, що DGN + R та DGN все ще перевершують інші моделі, а Floyd - з BL. В умовах інтенсивнішого трафіку DGN + R і DGN набагато кращі, ніж Floyd з BL, а DGN-M і MFQ також кращі, ніж Floyd з BL. Причина полягає в тому, що стратегія Floyd з BL (тобто просто слідування найкоротшим шляхом) є сприятливою, коли трафік слабкий і затори рідкісні, в той час як це не працює добре, коли інтенсивний рух і легко виникають затори. Незважаючи на те, що трафік в 2 рази важчий, ніж раніше, затримка DGN + R та DGN збільшується лише приблизно на 20%, що робить пропускну здатність на 1,6 × вищою, ніж раніше. Ми також перекваліфікуємо всі моделі в цьому режимі. Цікаво, що, як показано у таблицях 3.3 та 3.4, DGN + R та DGN з перенавчанням мають дещо вищий показник, але довшу затримку та меншу пропускну здатність [15]. Причина полягає в тому, що агенти, навчені в умовах інтенсивного

руху, приділяють більше уваги уникненню заторів (зменшення штрафу), що може спонукати агентів піти на більш довгий шлях. Однак, коли рух транспорту слабкий, затори менш вірогідні, і агенти в основному зосереджуються на пошуку найкоротшого шляху. За допомогою експериментів можна побачити, що модель, що навчена меншою кількістю агентів, може цілком узагальнити обстановку з більшою кількістю агентів, що демонструє політику, яка бере вхідні дані інтегрованої функції сусідніх агентів на основі їх відносин, що добре масштабується з кількістю агентів.

4 РЕАЛІЗАЦІЯ МОДЕЛІ ЗГОРТКОВОЇ МЕРЕЖІ ДЛЯ МУЛЬТИАГЕНТНОЇ КООПЕРАЦІЇ

4.1 Аналіз інструментальних засобів

Для реалізації моделі згорткової мережі була обрана задача розпізнавання зображень у Python за допомогою TensorFlow та Keras. Одним з найпоширеніших застосувань TensorFlow та Keras є розпізнавання/класифікація зображень.

TensorFlow – це бібліотека з відкритим кодом, створена для Python командою Google Brain. TensorFlow компілює безліч різних алгоритмів та моделей, дозволяючи користувачеві реалізовувати глибокі нейронні мережі для використання в таких завданнях, як розпізнавання/класифікація зображень та обробка природної мови. TensorFlow – це потужний фреймворк, який функціонує шляхом реалізації серії обробних вузлів, кожен вузол являє собою математичну операцію, а всю серію вузлів називають "графіком".

З точки зору Keras, це API високого рівня (інтерфейс прикладного програмування), який може використовувати функції TensorFlow внизу (а також інші бібліотеки, такі як Theano). Keras був розроблений з урахуванням зручності та модульності як основних принципів. На практиці Keras максимально спрощує реалізацію багатьох потужних, але часто складних функцій TensorFlow, і він налаштований на роботу з Python без будь-яких серйозних змін або конфігурації.

Розпізнавання зображень відноситься до завдання введення зображення в нейронну мережу та надання йому певної мітки для цього зображення. Мітка, яку виводить мережа, буде відповідати заздалегідь визначеному класу. Може бути кілька класів, на яких зображення може бути позначене, або лише один. Якщо існує один клас, часто застосовується термін "розпізнавання", тоді як багатокласне завдання розпізнавання часто називають

"класифікацією".

Підмножиною класифікації зображень є виявлення об’єктів, де конкретні екземпляри об’єктів ідентифікуються як такі, що належать до певного класу, як тварини, машини чи люди.

Для того, щоб здійснити розпізнавання/класифікацію зображень, нейронна мережа повинна здійснити вилучення ознак. Ознаки – це елементи даних, про які ви дбаєте, які будуть передаватися через мережу. У конкретному випадку розпізнавання зображень ознаками є групи пікселів, такі як краї та точки об’єкта, які мережа буде аналізувати на наявність зразків.

Розпізнавання ознак (або вилучення ознак) – це процес витягування відповідних ознак із вхідного зображення, щоб ці ознаки можна було проаналізувати. Багато зображень містять анотації або метадані про зображення, що допомагає мережі знайти відповідні функції.

4.2 Метод навчання нейронних мереж для розпізнавання зображення

Перший шар нейронної мережі приймає всі пікселі зображення. Після того, як усі дані надходять у мережу, до зображення застосовуються різні фільтри, які формують зображення різних частин зображення. Це вилучення об’єктів, і воно створює «карти об’єктів» рисунок 4.1.

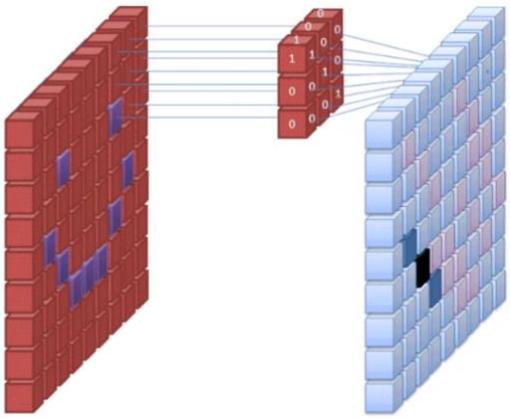

Рисунок 4.1 – Створення «карти об’єктів»

Цей процес вилучення об’єктів із зображення здійснюється за допомогою «згорткового шару», і згортка просто формує представлення частини зображення. Саме з цієї концепції згортки ми отримуємо термін Змовна нейронна мережа (CNN), тип нейронної мережі, який найчастіше використовується в класифікації/розпізнаванні зображень.

Якщо ви хочете наочно уявити, як працює створення функціональних карт, подумайте про те, щоб освітлити ліхтарик над зображенням у темній кімнаті. Проводячи промінь по зображенню, ви дізнаєтесь про особливості зображення. Фільтр – це те, що мережа використовує для формування подання зображення, і в цій метафорі світло від ліхтарика є фільтром.

Ширина променя вашого ліхтарика визначає, яку частину зображення ви одночасно досліджуєте, а нейронні мережі мають подібний параметр – розмір фільтра. Розмір фільтра впливає на те, яку частину зображення, скільки пікселів досліджують одночасно. Загальний розмір фільтра, що використовується в CNN, становить 3, і він охоплює як висоту, так і ширину, тому фільтр перевіряє область пікселів 3 x 3. Хоча розмір фільтра охоплює висоту та ширину фільтра, глибина фільтра також повинна бути вказана.

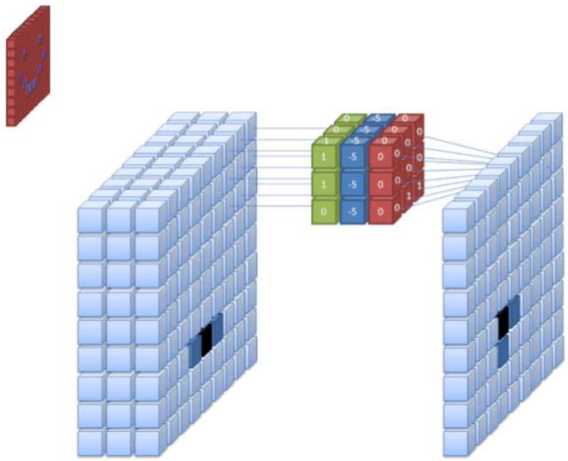

Рисунок 4.2 – Згорткова мережа з шаром функцій

Цифрові зображення відображаються як висота, ширина та деяке значення RGB, яке визначає кольори пікселя, тому "глибина", що відстежується – це кількість кольорових каналів, які має зображення.

Відтінки сірого (не кольорові) мають лише 1 кольоровий канал, тоді як кольорові зображення мають 3 канали глибини.

Все це означає, що для фільтра розміру 3, застосованого до повнокольорового зображення, розміри цього фільтра становитимуть 3 x 3 x 3. Для кожного пікселя, охопленого цим фільтром, мережа помножує значення фільтра на значення у самі пікселі, щоб отримати числове представлення цього пікселя. Потім цей процес робиться для всього зображення, щоб досягти повного представлення. Фільтр переміщується по решті зображення відповідно до параметра, який називається "крок", який визначає, на скільки пікселів повинен переміщуватися фільтр після обчислення значення в його поточному положенні. Звичайний розмір кроку для CNN становить 2.

Кінцевим результатом усього цього розрахунку є карта об’єктів. Цей процес зазвичай виконується за допомогою декількох фільтрів, що допомагає зберегти складність зображення.

Функції активації

Після створення карти функцій зображення значення, що представляють зображення, передаються через функцію активації або шар активації. Функція активації приймає значення, що представляють зображення, які мають лінійну форму (тобто просто перелік чисел) завдяки згортковому шару, і збільшує їх нелінійність, оскільки самі зображення нелінійні.

Типовою функцією активації, яка використовується для цього, є випрямлений лінійний блок (ReLU), хоча є деякі інші функції активації, які іноді використовуються.

Об’єднання шарів

Після активації даних вони надсилаються через шар об’єднання. Об'єднання "зменшених зразків" зображення, що означає, що воно бере інформацію, яка представляє зображення, і стискає, роблячи його меншим. Процес об'єднання робить мережу більш гнучкою та більш вправною у

розпізнаванні об'єктів/зображень на основі відповідних функцій.

Коли ми розглядаємо зображення, нас, як правило, не стосується вся інформація на задньому плані зображення, а лише ті функції, про які ми дбаємо, наприклад, про людей чи тварин.

Подібним чином, шар об'єднання в CNN абстрагує непотрібні частини зображення, зберігаючи лише ті частини зображення, які, на його думку, є доречними, як контролюється зазначеним розміром шару об'єднання.

Оскільки вона повинна приймати рішення щодо найбільш релевантних частин зображення, ми сподіваємось, що мережа вивчить лише ті частини зображення, які справді представляють об'єкт, про який йде мова. Це допомагає запобігти переобладнанню, коли мережа занадто добре вивчає аспекти навчальної справи та не узагальнює нові дані.

Існують різні способи об’єднання значень, але найчастіше використовується максимальне об’єднання. Максимальне об’єднання отримує максимальне значення пікселів в межах одного фільтра (в межах одного місця на зображенні). Це зменшує 3/4 частини інформації, припускаючи, що використовуються 2 x 2 фільтри.

Максимальні значення пікселів використовуються для того, щоб врахувати можливі спотворення зображення, а параметри / розмір зображення зменшуються для контролю надмірного встановлення. Є й інші типи об’єднання, такі як середнє об’єднання або об’єднання сум, але вони використовуються не так часто, оскільки максимальне об’єднання має тенденцію до кращої точності.

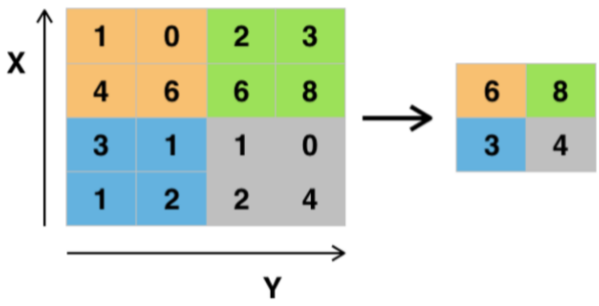

Рисунок 4.3 – Об’єднання шарів

Кінцеві шари CNN – це щільно з'єднані шари або штучна нейронна мережа (ANN). Основною функцією ANN є аналіз вхідних ознак та їх поєднання в різні атрибути, які допоможуть у класифікації. Ці шари по суті утворюють колекції нейронів, які представляють різні частини предмета, про який йде мова, а колекція нейронів може представляти дискети вух собаки або почервоніння яблука. Коли достатньо цих нейронів активується у відповідь на вхідне зображення, зображення класифікується як об’єкт.

Сплющення. Кінцеві шари нашого CNN, щільно зв’язані шари,

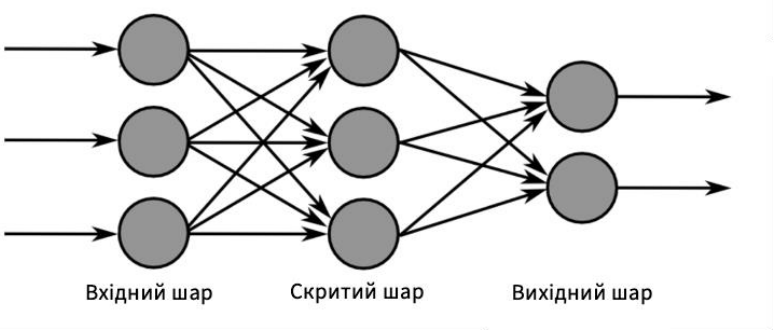

Рисунок 4.4 – Кінцеві шари

Помилка або різниця між обчисленими значеннями та очікуваним значенням у навчальному наборі обчислюється ANN. Потім мережа зазнає зворотного розмноження, де обчислюється вплив даного нейрона на нейрон у наступному шарі та коригується його вплив. Це робиться для оптимізації продуктивності моделі. Потім цей процес повторюється знову і знову. Це те, як мережа тренується на даних і вивчає асоціації між функціями введення та класами виводу.

Нейрони в середніх повністю зв’язаних шарах видаватимуть двійкові значення, що стосуються можливих класів. Якщо у вас чотири різні класи (скажімо, собака, машина, будинок і людина), нейрон матиме значення "1" для класу, який, на його думку, представляє зображення, і значення "0" для інших класів .

Остаточний повністю зв’язаний шар отримає вихідний рівень шару

перед ним і доставить ймовірність для кожного з класів, підсумовуючи до одного. Якщо в категорії "собака" є значення 0,75, це означає 75% впевненості, що зображення є собакою.

Класифікатор зображень вже підготовлений, і зображення можна передавати в CNN, який тепер видасть здогади про вміст цього зображення.

4.3 Етапи машинного навчання

Процес навчання моделі нейронної мережі є досить стандартним і може бути розбитий на чотири різні фази.

Підготовка даних. По-перше, потрібно зібрати дані та розмістити їх у формі, яку може навчити мережа. Це передбачає збір зображень та їх маркування. Навіть якщо ви завантажили набір даних, який підготував хтось інший, існує ймовірність попередньої обробки або підготовки, яку потрібно виконати, перш ніж використовувати його для навчання. Підготовка даних – це ціле мистецтво, яке включає вирішення таких проблем, як відсутні значення, пошкоджені дані, дані в неправильному форматі, неправильні мітки тощо.

Створення моделі. Створення моделі нейронної мережі передбачає вибір різних параметрів та гіперпараметрів. Треба прийняти рішення щодо кількості шарів, які слід використовувати у моделі, які розміри вхідного та вихідного шарів будуть, які функції активації будуть використовуватися, чи будете ви використовуватися відсів тощо.

Навчання моделі. Після того, як ми створили свою модель, ми просто створюємо екземпляр моделі та додаємо її до своїх навчальних даних. Найбільше врахування при навчанні моделі – це кількість часу, який модель займає для тренування. Ви можете вказати тривалість навчання для мережі, вказавши кількість епох, які потрібно тренувати. Чим довше ви тренуєте модель, тим більше її ефективність покращиться, але занадто багато епох тренувань і ви ризикуєте переобладнатись.

Вибір кількості епох, для яких потрібно тренуватися, це те, що ви відчуєте, і прийнято зберігати ваги мережі в перервах між тренувальними сесіями, так що вам не потрібно починати спочатку, коли ви досягли певного прогресу, навчаючи мережу.

Оцінка моделі. Існує кілька етапів оцінки моделі. Першим кроком у оцінці моделі є порівняння продуктивності моделі з валідаційним набором даних, набором даних, на якому модель не навчалась. Ви порівняєте ефективність моделі з цим набором перевірки та проаналізуєте її ефективність за допомогою різних показників.

Існують різні метрики для визначення продуктивності моделі нейронної мережі, але найпоширенішою метрикою є "точність", кількість правильно класифікованих зображень, поділена на загальну кількість зображень у вашому наборі даних.

Після того, як ви переконаєтесь у точності роботи моделі на наборі даних перевірки, ви, як правило, повертаєтесь назад і тренуєте мережу знову, використовуючи злегка доопрацьовані параметри, оскільки навряд чи ви будете задоволені роботою своєї мережі під час першого тренування. Ви будете продовжувати налаштовувати параметри своєї мережі, перекваліфікувати її та вимірювати її ефективність, доки вас не задовольнить точність мережі.

Нарешті, ви перевірите продуктивність мережі на наборі тестувань.

Хороша ідея зберігати пакет даних, яких мережа ніколи не бачила для тестування, оскільки всі налаштування параметрів, які ви виконуєте, у поєднанні з повторним тестуванням набору перевірок можуть означати, що ваша мережа вивчила деякі особливості набору перевірки, які не буде узагальнювати на дані, що не належать до вибірки.

Таким чином, метою набору тестувань є перевірка на наявність таких проблем, як переобладнання, та впевненість у тому, що ваша модель справді придатна для роботи в реальному світі.

4.4 Розпізнавання зображень за допомогою CNN

Розглянемо повний приклад розпізнавання зображень за допомогою Keras, від завантаження даних до оцінки. Для початку нам знадобиться набір даних для тренування. У цьому прикладі ми будемо використовувати відомий набір даних CIFAR-10.

CIFAR-10 – це великий набір зображень, що містить понад 60 000 зображень, що представляють 10 різних класів об’єктів, таких як коти, літаки та машини. Зображення мають повнокольоровий RGB, але вони досить маленькі, лише 32 x 32. Одна чудова річ у наборі даних CIFAR-10 полягає в тому, що він поставляється в упаковці з Keras, тому завантажити набір даних дуже легко, і зображення потрібні дуже мало попередньої обробки.

Перше, що нам слід зробити, це імпортувати необхідні бібліотеки. Наступним кроком потрібно підготувати дані. Нам потрібен ще один імпорт: набір даних. Тепер завантажимо набір даних. Можно зробити це, просто вказавши, до яких змінних ми хочемо завантажити дані, а потім скориставшись функцією load_data() (приклад 4.1):

```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

Приклад 4.1 – завантаження набору даних

Оскільки ми використовуємо попередньо розфасований набір даних, потрібно зробити дуже мало попередньої обробки. Одне, що ми хочемо зробити – це нормалізувати вхідні дані.

Якщо значення вхідних даних знаходяться в занадто широкому діапазоні, це може негативно вплинути на ефективність роботи мережі. У цьому випадку вхідними значеннями є пікселі на зображенні, значення яких становлять від 0 до 255.

Тож для того, щоб нормалізувати дані, ми можемо просто розділити

значення зображення на 255. Для цього нам спочатку потрібно зробити дані плаваючими, оскільки вони в даний час є цілими числами. Ми можемо зробити це за допомогою команди astype () Numpy, а потім оголосивши, який тип даних ми хочемо (приклад 4.2):

```
    # normalize the inputs from 0-255 to between 0 and
1 by dividing by 255
    X_train = X_train.astype('float32')
    X_test = X_test.astype('float32')
    X_train = X_train / 255.0
    X_test = X_test / 255.0
```

Приклад 4.2 – нормалізація даних

Інша справа, яку потрібно буде зробити, щоб підготувати дані до мережі – це одноразове кодування значень. Зображення не можуть використовуватися мережею, як вони є, їх потрібно кодувати спочатку, і найкраще використовувати одноразове кодування двійкова класифікація.

Тут ми ефективно робимо двійкову класифікацію, оскільки зображення або належить до одного класу, або ні, воно не може потрапити десь посередині. Команда Numpy to_categorical () використовується для кодування. Ось чому ми імпортували функцію np_utils з Keras, оскільки вона містить to_categorical ().

Також потрібно вказати кількість класів, що знаходяться в наборі даних, щоб ми знали, скільки нейронів стиснути до кінцевого шару:

```
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
class_num = y_test.shape[1]
```

Приклад 4.3 – кодування

Наступним кроком слід розробити модель. Перше, що потрібно зробити, це визначити формат, який ми хотіли б використовувати для моделі, Keras має кілька різних форматів або креслень для побудови моделей, але

Sequential є найбільш часто використовуваним, і з цієї причини ми імпортували його з Keras.

Перший шар нашої моделі – це згортковий шар. Він буде приймати входи і запускати на них згорткові фільтри.

Застосовуючи їх у Keras, ми повинні вказати кількість потрібних каналів/фільтрів (32), розмір потрібного фільтра (3 x 3 у цьому випадку), вхідну форму (при створенні першого шару) та активація та заповнення, які нам потрібні.

Як вже згадувалося, relu – це найпоширеніша активація, а padding = 'same' просто означає, що ми зовсім не змінюємо розмір зображення:

```
    model.add(Conv2D(32,(3,3),input_shape=X_train.shape
[1:],padding='same'))
    model.add(Activation('relu'))
```

Приклад 4.4 – створення шару

Тепер ми зробимо випадаючий шар, щоб запобігти переобладнанню, який функціонує шляхом випадкового усунення деяких зв’язків між шарами (0,2 означає, що він скидає 20% існуючих з’єднань):

```
model.add(Dropout(0.2))
```

Приклад 4.5 – створення випадаючого шару

Ми також можемо зробити тут пакетну нормалізацію. Пакетна нормалізація нормалізує заголовок входів на наступний рівень, гарантуючи, що мережа завжди створює активації з тим самим розподілом, який ми бажаємо.

Тепер з’являється ще один згортковий рівень, але розмір фільтра збільшується, тому мережа може вивчати більш складні уявлення:

```
model.add(Conv2D(64, (3, 3), padding='same')
model.add(Activation('relu'))
```

Приклад 4.6 – створення нового згорткового рівня

Ось шар об’єднання, як обговорювалося раніше, це допомагає зробити класифікатор зображень більш надійним, щоб він міг засвоїти відповідні шаблони. Існує також нормалізація відсіву та партії:

```
model.add(MaxPooling2D(pool_size=(2, 2))
model.add(Dropout(0.2))
model.add(BatchNormalization())
```

Приклад 4.7 – шар об’єднання

Це основний потік для першої половини реалізації CNN: згортання, активація, відмова, об’єднання. Додаючи згорткові шари, ви зазвичай збільшуєте їх кількість фільтрів, щоб модель могла вивчати більш складні подання.

Важливо не мати занадто багато шарів об’єднання, оскільки кожне об’єднання відкидає деякі дані. Об’єднання занадто часто призведе до того, що щільно пов’язаним шарам не буде майже нічого дізнатися про те, коли дані надходять до них. Точна кількість шарів об'єднання, які слід використовувати, буде залежати від завдання, яке виконується.

Повторимо ці шари, щоб дати вашій мережі більше уявлень для відпрацювання. Тепер використовуємо імпорт Dense і створюємо перший щільно пов’язаний шар. Потрібно вказати кількість нейронів у щільному шарі. Кількість нейронів у наступних шарах зменшується, з часом наближаючись до тієї ж кількості нейронів, що і класи в наборі даних. Обмеження ядра може регулювати дані під час вивчення, ще одна річ, яка допомагає запобігти перенапруженню.

```
    model.add(Dense(256, kernel_constraint=maxnorm(3)))
model.add(Activation('relu'))    model.add(Dropout(0.2))
model.add(BatchNormalization())     model.add(Dense(128,
kernel_constraint=maxnorm(3)))
```

```
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(BatchNormalization())
```

Приклад 4.8 – створення першого щільно пов’язаного шару

У цьому остаточному шарі передаємо кількість класів для кількості нейронів. Кожен нейрон представляє клас, і результатом цього шару буде вектор 10 нейронів, при цьому кожен нейрон зберігає певну ймовірність того, що розглянуте зображення належить класу, який він представляє.

Нарешті, функція активації softmax вибирає нейрон з найбільшою ймовірністю як вихідний результат, проголосувавши, що зображення належить до цього класу:

```
model.add(Dense(class_num))
model.add(Activation('softmax'))
```

Приклад 4.9 – функція активації softmax

Тепер, коли розроблена модель, яку хочемо використовувати, нам просто потрібно її скомпілювати. Давайте визначимо кількість епох, для яких ми хочемо тренуватися, а також оптимізатор, який ми будемо використовувати.

Оптимізатор – це те, що буде налаштовувати ваги у вашій мережі, щоб наблизитися до точки найменших втрат. Алгоритм Адама є одним із найбільш часто використовуваних оптимізаторів, оскільки він забезпечує чудову продуктивність для більшості проблем:

```
epochs = 25
optimizer = 'adam'
```

Приклад 4.10 – оптимізатор

Тепер складемо модель з обраними нами параметрами. Також визначимо метрику для використання.

```
    model.compile(loss='categorical_crossentropy',optim
izer=optimizer,
    metrics=['accuracy'])
```

Приклад 4.11 – складання моделі

Ми можемо роздрукувати резюме моделі, щоб побачити, як виглядає вся модель. Друк резюме дасть досить багато інформації (додаток В):

Тепер ми переходимо до навчання моделі. Для цього нам потрібно лише викликати функцію fit () на моделі та передати вибрані параметри.

```
    numpy.random.seed(seed) model.fit(X_train, y_train,
validation_data=(X_test,     y_test),     epochs=epochs,
batch_size=64)
```

Приклад 4.12 – навчання моделі

Тепер ми можемо оцінити модель і подивитися, як вона працює викликавши model.evaluate ():

```
    # Model evaluation scores = model.evaluate(X_test,
y_test,   verbose=0)   print   ("Accuracy:   %.2f%%"   %
(scores[1]*100))
```

Приклад 4.13 – оцінка моделі

Результат наведений у прикладі 4.14.

```
    Accuracy: 83.01%
```

Приклад 4.14 – результат система розпізнавання зображень CNN

ВИСНОВКИ

В ході виконання магістерської роботи був зроблений обзор та аналіз сучасних робот та публікацій за тематикою згоркових нейронних мереж в мультиагентній кооперації та в результаті проведених досліджень була розроблена модель згорткової мережі для мультиагентної кооперації.

Також були розглянуті сучасні методи розробки нейронних мереж, проведений аналіз моделей мультиагентної кооперації.

Розроблена модель на мові Python за допомогою інструментів TensorFlow та Keras. З їх допомогою довелося створити модель, яка розпізнає зображення з точністю 83%.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Padgham L., Winikoff M. Developing Intelligent Agent Systems: A Practical Guide. RMIT University, Melbourne, Australia: John Wiley & Sons, Ltd, 2004.

2. Dautenhahn K. Getting to know each other - artificial social intelligence for autonomous robots. Robotics and Autonomous Systems, 1995.

3. Gorodetsky V., Skoblev P., Marik V. System engineering view on multi-agent technology for industrial applications: barriers and prospects // Cybernetics and physics. 2020. №105.

4. J. E. Doran, S. Franklin, N. R. Jennings, T. J. Norman On Cooperation in Multi-Agent Systems // International Journal of Control. 2018. №91.

5. Melis, A. P. and Semmann, D. How is human cooperation different? Philosophical Transactions of the Royal Society of London B: Biological Sciences, 365(1553): 2663–2674, 2010.

6. Ohtsuki, H., Hauert, C., Lieberman, E., and Nowak, M. A. A simple rule for the evolution of cooperation on graphs and social networks. Nature, 441(7092):502, 2006.

7. Shalev-Shwartz, S., Shammah, S., and Shashua, A. Safe, multi-agent, reinforcement learning for autonomous driv- ing. arXiv preprint arXiv:1610.03295, 2016.

8. Wiering, M. Multi-agent reinforcement learning for traffic light control. In ICML’00, pp. 1151–1158, 2000.

9. Yang, Y., Hao, J., Sun, M., Wang, Z., Fan, C., and Strbac, G. Recurrent deep multiagent q-learning for autonomous brokers in smart grid. In IJCAI’18, pp. 569–575, 2018a.

10. Matignon, L., Jeanpierre, L., Mouaddib, A.-I., et al. Coordi- nated multi-robot exploration under communication con- straints using decentralized markov decision processes. In AAAI’12, 2012.

11. Sukhbaatar, S., Fergus, R., et al. Learning multiagent com- munication with backpropagation. In NeurIPS’16, pp. 2244–2252, 2016.

12. Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., and Wang, J. Mean field multi-agent reinforcement learning. In ICML’18, pp. 5571–5580, 2018b.

13. Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P. A., Strouse, D., Leibo, J. Z., and de Freitas, N. Intrinsic social motivation via causal influence in multi-agent rl. arXiv preprint arXiv:1810.08647, 2018.

14. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *NeurIPS’17*, pp. 5998–6008, 2017.

15. Jiechuan Jiang, Chen Dun, Zongqing Lu Graph Convolutional Reinforcement Learning for Multi-Agent Cooperation // Neural Networks. 2015. №228.

16. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In NeurIPS’17, pp. 6379–6390, 2017.

17. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In AAAI’18, 2018.

18. Zhang, K., Yang, Z., Liu, H., Zhang, T., and Bas¸ar, T. Fully decentralized multi-agent reinforcement learning with networked agents. In ICML’18, 2018.

19. Sukhbaatar, S., Fergus, R., et al. Learning multiagent com- munication with backpropagation. In NeurIPS’16, pp. 2244–2252, 2016.

20. Peng, P., Wen, Y., Yang, Y., Yuan, Q., Tang, Z., Long, H., and Wang, J. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learn- ing to play starcraft combat games. arXiv preprint arXiv:1703.10069, 2017.

21. Jiang, J. and Lu, Z. Learning attentional communication for multi-agent cooperation. NeurIPS’18, 2018.

22. Das, A., Gervet, T., Romoff, J., Batra, D., Parikh, D., Rab- bat, M., and

Pineau, J. Tarmac: Targeted multi-agent communication. arXiv preprint arXiv:1810.11187, 2018.

23. Ohtsuki, H., Hauert, C., Lieberman, E., and Nowak, M. A. A simple rule for the evolution of cooperation on graphs and social networks. Nature, 441(7092):502, 2006.

24. Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In ICLR'17, 2017.

25. Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bom- barell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints. In NeurIPS'15, pp. 2224–2232, 2015.

26. Charles, R. Q., Su, H., Kaichun, M., and Guibas, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In CVPR'17, pp. 77–85, 2017.

27. Battaglia, P., Pascanu, R., Lai, M., Rezende, D. J., et al. Interaction networks for learning about objects, relations and physics. In NeurIPS'16, pp. 4502–4510, 2016.

28. Zambaldi, V., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D., Lillicrap, T., Lockhart, E., et al. Relational deep reinforcement learn- ing. arXiv preprint arXiv:1806.01830, 2018.

29. Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In CVPR'17, pp. 2261–2269, 2017.