Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження властивостей нейронних мереж з випадковою ініціалізацією
(тема)

Виконав:
студент 2 курсу, групи СШІм-20-1
Пуйда Б. Д.
(прізвище, ініціали)

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

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

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник доц. Узлов Д.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

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

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

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

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

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

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

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Пуйді Богдану Дмитровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження властивостей нейронних мереж з випадковою ініціалізацією

затверджена наказом університету від 08 листопада 20 21 р. № 1659Ст
2. Термін подання студентом роботи до екзаменаційної комісії 08 грудня 20 21 р.
3. Вихідні дані до роботи Науково-технічні публікації, дані відомих наукових проектів, щодо моделювання коннектомів, дані статей, результати експериментальних досліджень

4. Перелік питань, що потрібно опрацювати в роботі 1. Аналіз предметної галузі та постановка задачі 2. Проектування системи 3. Програмна реалізація системи

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок1 – Модель штучного нейрона, Рисунок 2 – Приклади функцій активацій, Рисунок 3 – Проста одношарова штучна нейронна мережа, Рисунок 4 – Схематична ілюстрація методів NAS, Рисунок 5 – Ілюстрація різних просторів пошуку, Рисунок 6 – Схема генетичного алгоритму, Рисунок 7 – Приклад оптимальних по Парето рішень, Рисунок 8 – Методологія прямого кодування, Рисунок 9 – Схема мутації, Рисунок 10 – Схема кросоверу, Рисунок 11 – Типовий сценарій RL проблеми, Рисунок 12 – Огляд схеми пошуку нейронних мереж з випадковими вагами, Рисунок 13 – Оператори для пошуку простору мережевих топологій, Рисунок 14 – Розвиток нейромережевих топологій з плином часу, Рисунок 15 – Покажчики ефективності агентів, Рисунок 16 – Мережі-чемпіони для безперервних завдань управління, Рисунок 17 – Класифікаційна точність на MNIST, Рисунок 18 – Мережа для класифікації MNIST (1849 з'єднань)

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

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	доц. Узлов Д. Ю.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на дипломну роботу	01.09.2021	виконано
2	Аналіз предметної галузі і постановка завдання	02.09.2021 – 15.09.2021	виконано
3	Дослідження методів та технологій	16.09.2021 – 25.09.2021	виконано
4	Аналіз існуючих реалізацій	26.09.2021 – 05.10.2021	виконано
5	Розробка методу рішення поставленої проблеми	06.10.2021 – 15.10.2021	виконано
6	Створення імітаційної моделі	16.10.2021 – 25.10.2021	виконано
7	Тестування і опрацювання імітаційної моделі	26.10.2021 – 08.11.2021	виконано
8	Оформлення пояснювальної записки	09.11.2021 – 18.11.2021	виконано
9	Оформлення графічних матеріалів	19.12.2021 – 30.11.2021	виконано
10	Попередній захист	01.12.2021	виконано
11	Захист перед ЕК	08.12.2021	виконано

Дата видачі завдання 01 вересня 20 21 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Записка пояснювальна: 86 с., 18 рис., 2 табл., 2 дод., 34 джерела.

АЛГОРИТМІЧНА ІНФОРМАЦІЙНА ТЕОРІЯ, ГЕНЕТИЧНИЙ АЛГОРИТМ, ГЛИБОКЕ НАВЧАННЯ, НАВЧАННЯ З ПІДКРІПЛЕННЯМ, NEURAL ARCHITECTURE SEARCH, PYTHON, WEIGHT AGNOSTIC NEURAL NETWORKS

У данній кваліфікаційній роботі розглядається застосування генетичного алгоритму до архітектури нейронної мережі, з метою визначення ступеню впливу архітектури на якість вирішення задач як класифікації, так і задач з підкріпленням. Результати багатьох досліджень в задачах пов’язаних з комп’ютерним зором та обробкою дискретних послідовностей показали, що деякі типи мереж, навіть без тренування ваг, можуть кодувати успішні рішення для деяких задач. Суть даного алгоритму полягає в поступовому нарощенні складності простої нейронної мережі з випадковою кількістю дуг та функцій активацій. Для оцінки цих мереж, ваги заповнюються єдиним спільним параметром, відібраним з рівномірного випадкового розподілу, і вимірюються очікувані показники. Демонструється, що цей метод може знайти мінімальні архітектури нейронних мереж, які можуть виконувати декілька навчальних завдань з підкріпленням без тренування. У навчанні з вчителем також можна знайти мережеві архітектури, які досягають набагато вищої точності, ніж алгоритми з випадковою точністю, використовуючи випадкові ваги.

Проводиться порівняльний аналіз точності моделей нейроних мереж для поставлених задач.

Результатом кваліфікаційної роботи є розроблений фреймворк здобуття оптимальної нейронної мережі для заданої задачі та можливість візуалізації та аналізу здобутої мережі.

РЕФЕРАТ

Записка пояснительная: 86 с., 18 рис., 2 табл., 2 прил., 34 источника.

АЛГОРИТМИЧЕСКАЯ ИНФОРМАЦИОННАЯ ТЕОРИЯ, ГЕНЕТИЧЕСКИЙ АЛГОРИТМ, ГЛУБКОЕ ОБУЧЕНИЕ, ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ, NEURAL ARCHITECTURE SEARCH, PYTHON, WEIGHT AGNOSTIC NEURAL NETWORKS

В данной квалификационной работе рассматривается применение генетического алгоритма к архитектуре нейронной сети с целью определения степени влияния архитектуры на качество решения задач как классификации, так и задач с подкреплением. Результаты многих исследований в задачах, связанных с компьютерным зрением и обработкой дискретных последовательностей, показали, что некоторые типы сетей, даже без тренировки весов, могут кодировать успешные решения для некоторых задач. Сущность данного алгоритма заключается в постепенном наращивании сложности простой нейронной сети со случайным количеством дуг и функций активаций. Для оценки этих сетей веса заполняются единственным общим параметром, отобранным из равномерного случайного распределения, и измеряются ожидаемые показатели. Демонстрируется, что этот метод может найти минимальные архитектуры нейронных сетей, которые могут выполнять несколько задач с подкреплением без тренировки. В обучении с учителем можно найти сетевые архитектуры, которые достигают гораздо более высокой точности, чем алгоритмы со случайной точностью, используя случайные веса. Результатом квалификационной работы является фреймворк получения оптимальной нейронной сети для заданной задачи и возможность визуализации и анализа полученной сети.

ABSTRACT

Explanatory note: 86 p., 18 fig., 2 tabl., 2 ann., 34 source.

ALGORITHMIC INFORMATION THEORY, DEEP LEARNING, GENETIC ALGORYTHM, NEURAL ARCHITECTURE SEARCH, REINFORCEMENT LEARNING, PYTHON, WEIGHT AGNOSTIC NEURAL NETWORKS

This thesis considers the application of a genetic algorithm to the architecture of a neural network, in order to determine the degree of influence of architecture on the quality of solving both classification and reinforcement problems. Many studies in computer vision and discrete sequence processing have shown that some types of networks, even without weight training, can encode successful solutions for some tasks. The essence of this algorithm is to gradually increase the complexity of a simple neural network with a random number of arcs and activation functions. To evaluate these networks, the weights are filled with a single common parameter selected from a uniform random distribution, and the expected performance is measured. It is demonstrated that this method can find minimal neural network architectures that can perform multiple training reinforcement tasks without training. In supervised learning, one can also find network architectures that achieve that much higher than chance accuracy using random weights.

A comparative analysis of the accuracy of neural network models for the tasks is also made. The result of the thesis is the developed framework for obtaining the optimal neural network for a given task and the ability to visualize and analyze the obtained network.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ СИМВОЛІВ ОДИНИЦЬ СКОРОЧЕНЬ І ТЕРМІНІВ

ГА – генетичний алгоритм;

ШНМ – штучна нейронна мережа;

AIT – algoritmic information theory – алгоритмічна інформаційна теорія;

ANN – artificial neural network – штучна нейронна мережа;

MDL – minimum description length – принцип мінімальної довжини опису;

NEAT – neuroevolution of augmenting topologies – нейроеволюція наростаючої топології;

ReLU – rectified linear unit – випрямлена лінійна одиниця;

TWEANNs – topology and weight evolving artificial neural networks – штучні нейронні мереж із топологією і вагами, що розвиваються;

WANN – weight agnostic neural networks – еволюційний алгоритм розвитку нейронних мереж з випадковими вагами.

ВСТУП

Разом із розвитком інформаційних технологій, в останні роки особливою проблемою є збір та обробка великої кількості інформації з метою покращення процесів, що відбуваються у складних інформаційних системах. Машинне навчання, зокрема глибоке навчання, досягло значних успіхів за останні роки, і на нього покладається постійно зростаюча кількість дисциплін. Однак цей успіх залежить від людського втручання на багатьох етапах (попередня обробка даних, розробка ознак (features), вибір моделі, оптимізація гіперпараметрів тощо). Оскільки складність цих завдань часто виходить за межі неспеціалістів, швидке зростання додатків машинного навчання створило попит на готові або багаторазові методи, які можна використовувати легко і без знання експертів.

Розглянемо чим AutoML (Автоматизоване машинне навчання) відрізняється від ML. У ML дослідники даних спочатку починають з постановки проблеми та набору даних. Дані аналізуються та очищаються, визначається показник продуктивності, а потім експериментуються з кількома моделями, які можуть працювати на наборі даних, відповідно до людської інтуїції. Перш ніж буже отримана прийнятна модель, є багато робот по настройці параметрів. AutoML – це спроба автоматизувати якомога більшу частину цього конвеєру. Хоча деякі з цих кроків автоматизувати дотсатньо просто, так звану тонку настройку моделі формалізувати та автоматизувати складно, наприклад вибір правильної архітектури / моделі тощо. Тобто AutoML – це спроба створити єдину систему, яка може усунути необхідність людського втручання на кожному кроці тренувального процесу.

Завданням AutoML є завдання забезпечення універсальними навчальними машинами, які можуть вчитися та робити прогнози без втручання людини. Загальна проблема охоплює широкий спектр

труднощів, які неможливо вирішити усі відразу. Назвемо лише декілька: парсинг даних та їх форматування, попередня обробка та feature engineering, виявлення та обробка упереджених (biased) даних, неоднорідних, дрейфуючих, мультимодальних даних, узгодження алгоритмів із проблемами (що може включати контрольоване, неконтрольоване або навчання з підкріпленням чи з іншими налаштуваннями), отримання нових даних (активне навчання, навчання за запитами, причинно-наслідкові експерименти), управління великими обсягами даних, включаючи створення належних масштабованих та стратифікованих навчальних, валідаційних та тестових наборів, вибір алгоритмів, які задовольняють обмеженням ресурсів під час навчання та виконання, можливість генерувати та повторно використовувати робочі процеси та генерувати пояснювальні звіти.

Окрім мотивації автоматизованого пошуку параметрів без навчання є ще одна, яка випливає з теорії нейронауки. Штучні нейронні мережі пережили революцію, каталізатором якої стали контрольовані алгоритми навчання. Однак, на відміну від молодих тварин (включаючи людей), навчання таких мереж вимагає величезної кількості позначених прикладів, що призводить до переконання, що тварини повинні покладатися переважно на навчання без нагляду. Стверджується, що поведінка більшості тварин не є результатом розумних алгоритмів навчання – під наглядом чи без нагляду – а закодована в геномі. Зокрема, тварини народжуються з високоструктурованими мозковими зв’язками, що дозволяє їм дуже швидко вчитися. Оскільки схема підключення є занадто складною, щоб бути чітко визначеною в геномі, її необхідно стиснути через «геномне вузьке місце». Геномне вузьке місце передбачає шлях до ШНМ, здатних до швидкого навчання.

У цій роботі пропонується провести пошук нейронних архітектур, починаючи з простих і повільно нарощувати складність, для того, щоб

автоматично знайти алгоритм, який є у хорошому сенсі упередженим для вирішення задачі ML, і водночас є нелійним алгоритмом з можливістю інтерпретування його архітектури. При цьому відпадає необхідність тренувати модель (зменшуються вимоги до hardware ресурсів) та вирішувати різні проблеми методу обратного поширення похибки.

Таким чином, беручи до уваги усе, що зазначено вище, актуальною для дослідження є тема генерації нейронних мереж з випадковими вагами.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Штучні нейронні мережі

Штучна нейронна мережа є концептуальною моделлю біологічної нейронної мережі і складається з пов'язаних між собою різним чином шарів штучних нейронів, які організовують загальну активну структуру і функціонально впливають на роботу один одного. У більшості архітектур штучних нейромереж активність нейрона визначається перетворенням зовнішнього сумарного впливу інших нейронів на даний нейрон. З моменту свого зародження технології штучних нейронних мереж розвивалися досить відокремлено від класичних методів, нерідко докорінно змінюючи уявлення про предмет і проблематику теорії машинного навчання і розпізнавання об’єктів, залишаючи значний вплив на теоретичний, термінологічний і методологічний апарати цих дисциплін. Через деякий час після розвитку базових моделей штучних нейронних мереж, відбувся значний поділ науки про нейромережі на види топологій архітектури мереж і методи навчання мереж. У більшості архітектур штучних нейронних мереж функції активації нейронів фіксовані, а ваги синапсів є параметрами мережі. Деякі входи нейронів є зовнішніми входами сукупної мережі, а деякі виходи нейронів - виходами сукупної мережі. Завдання нейромережі полягає в перетворенні вхідного вектора у вихідний вектор, що здійснюється вагою і топологією мережі. Штучний нейрон характеризується своїм поточним станом. Тут можна провести аналогію з нервовими клітинами головного мозку, які можуть бути пошкоджені або неправильно працювати[21]. Він володіє групою синапсів – односпрямованих вхідних зв'язків, з'єднаних з виходами інших нейронів, присутня така частина, як аксон – вихідний зв'язок штучного нейрона, з

якого сигнал (збудження або гальмування) надходить на синапси наступних нейронів. Загальний вигляд штучного нейрона наведено на рисунку 1.1.

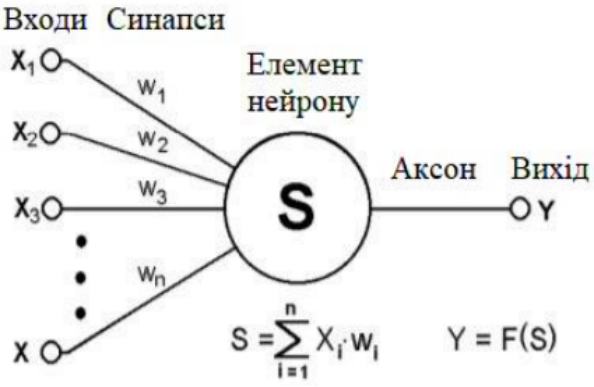

Рисунок 1.1 – Модель штучного нейрона

Штучний нейрон спочатку імітує властивості, що дані біологічному нейрону. Тут безліч вхідних сигналів, позначених x_1, x_2, … , x_N, надходить на штучний нейрон. Ці вхідні сигнали, в сукупності позначаються вектором X, приходять до синапсів біологічного нейрона. Кожен синапс характеризується величиною синаптичного зв'язку або його вагою.

Для реалізації нелінійності при активації нейрона, його активність, крім різних видів суматорів і систем ваг на входах, визначається функцією одного аргументу – функцією активації. Нейрон в цілому реалізує скалярну функцію векторного аргументу, а вихідний сигнал нейрона визначається видом функції активації і може бути дійсним або цілим. Функція активації застосовується до зваженої суми постсинаптичних сигналів на вході нейрона.

Таким чином, активність нейрона повністю визначається його параметрами і його функцією активації. Існує безліч передавальних функцій, що застосовуються на практиці при розробці нейронних мереж, деякі з них служать для реалізації нелінійності системи. Вибір тієї чи іншої

функції активації часто залежить від умов завдання і структури мережі. Деякі з розглянутих нижче функцій застосовуються тільки в застарілих системах або в навчанні, але вважаються класичними і згадуються щоразу при вивченні штучних нейронних мереж. Порогова функція Хевісайда [8] є найпростішою кусочно-лінійною передавальною функцією. Ця функція використовувалася в класичному персептроні, в даний час використовується в основному з метою навчання теорії нейронних мереж.

При використанні нескладної кусочнолінійної функції сигнал на виході нейрона лінійно пов'язаний зі зваженою сумою сигналів на його вході. В даний момент лінійна функція на практиці також використовується дуже рідко. Сигмоїдальна функція активації [10]. Сигмоїд є монотонно зростаючою всюди диференційованою S-образною нелінійною функцією з насиченням. Забезпечує посилення слабких сигналів і запобігає насиченню сильних сигналів. Є однією з найбільш поширених передавальних функцій, часто використовується в нейронних мережах і сьогодні. Введення сигмоїдальних функцій було зумовлено недостатньою гнучкістю класифікаторів на основі порогових передавальних функцій і дозволило перейти від жорсткої однорозрядної логіки до більш гнучкої поведінки і адаптивної параметризації нейронних мереж. Прикладом сигмоїдальной функції є логістична передавальна функція [3]. Функція гіперболічного тангенса відрізняється від логістичної кривої тим, що її область значень лежить в інтервалі (-1; 1), що в деяких випадках може спростити завдання навчання нейромереж. У неглибоких нейромережах використовуються нелінійні функції активації. Часто зустрічаються різновиди сигмоїдальних і тангенціальних функцій є нелінійними, але на практиці при навчанні глибоких нейромереж такі функції можуть привести до проблем із загасанням або збільшенням градієнтів.

Функція ReLU є випрямленою лінійною функцією і на даний момент вважається набагато більш простим і ефективним з точки зору

обчислювальної складності варіантом передавальної функції. Похідна цієї функції дорівнює або 0, або 1, від чого її застосування запобігає розростанню і загасанню градієнтів, і призводить до зменшення ваг, що позитивно позначається на обчислювальної здатності нейромережі. Передавальна функція ReLU є одним з останніх успіхів в області методів налаштувань глибоких нейронних мереж. Сьогодні існує сімейство різних модифікацій ReLU, які вирішують проблеми надійності цієї функції при проходженні через нейрон великих градієнтів: Leaky ReLU, Parametric ReLU, Randomized ReLU. Приклади активаційних функцій наведено на рисунку 1.2.

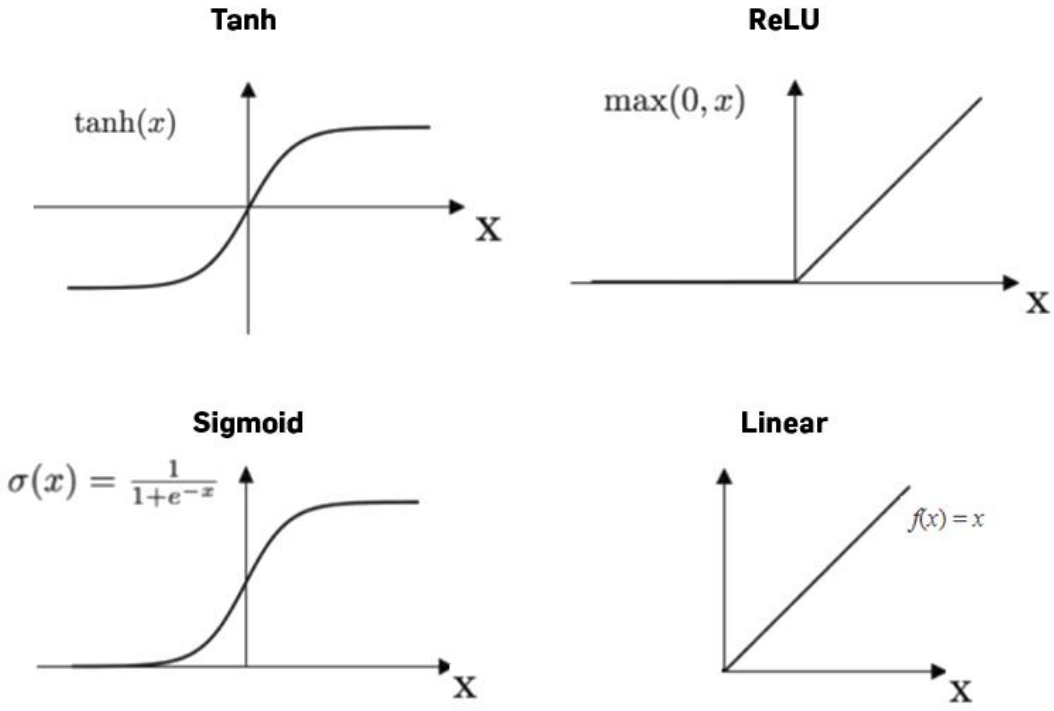

Рисунок 1.2 – Приклади функцій активації

Можливість простого розпізнавання доступна одному нейрону, проте з'єднання тих же нейронів в мережі – запорука кращого результату. Одношаровою є мережа, що складається з сукупності нейронів, утворення яких формує шар.

Ліві вершини схеми зображеного нейрону служать для розподілу сигналів, що подаються на вхід. За цими вершинами не закріплено жодних обчислень, тому вони не вважаються шаром і позначені колами, щоб відрізняти їх від обчислювальних нейронів, що позначаються квадратами. Існує сполучення окремою вагою кожен елементу з множини входів X з кожним штучним нейроном. За нейроном закріплена робота подачі зваженої суми входів в мережу. З метою спільності штучні та біологічні мережі мають відсутні з'єднання. Існують з'єднання між виходами і входами елементів в самому шарі. Ваги представлені елементами матриці *W*. Матриця має n рядків і m стовпців, де n – число входів, а m – число нейронів. Наприклад, *w12* – це вага, що зв'язує перший вхід з другим нейроном. Таким чином, обчислення вихідного вектора *Y*, зводиться до матричному множенню *Y = XW*. Приклад простої одношарової нейронної мережі наведено на рисунку 1.3.

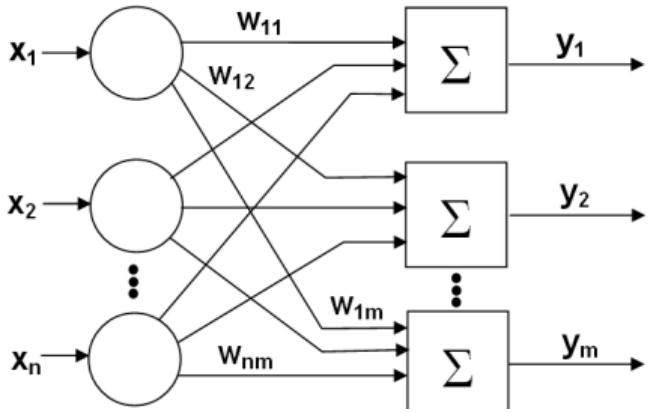

Рисунок 1.3 – Проста одношарова штучна нейронна мережа

1.1.1 Багатошарові нейронні мережі

Відмінності обчислювальних процесів в нейронних мережах часто обумовлені способом взаємозв'язків нейронів. За сукупністю критеріїв на сьогоднішній день багатошарові архітектури можна розділити на статичні і динамічні. Кожен з класів архітектур нейронних мереж може включати безліч підкласів, реалізуючи різні підходи, нижче будуть наведені основні з них. До статичних архітектур відносять мережі прямого поширення, в яких реалізована однонаправлений зв'язок між шарами, відсутні динамічні елементи і зворотній зв'язок, а вихід навченої нейромережі однозначно визначається входом і не залежить від попередніх станів мережі.

Статичні штучні нейронні мережі прямого поширення:

– персептрон;
– нейронна мережа Кохонена;
– когнітрон та неокогнітрон;
– сучасна згорткова нейронна мережа.

На противагу статичним архітектурам, існують динамічні архітектури штучних нейромереж, що реалізують рекурентну структуру з використанням зворотніх зв'язків, завдяки чому стан мережі в кожний момент часу залежить від попереднього стану. Рекурентні нейромережі як правило базуються на багатошаровому персептроні. Елементарний персептрон організовується на основі сенсорних даних на вході – S-елементів, асоціативних елементів – A-елементів, і реагуючих елементів на виході – R-елементів. Набір S-елементів, пов'язаний з A-елементом утворює асоціацію, і A-елемент активізується після того як досягнуто певне число сигналів від S-елементів. A-елемент передає зважений сигнал на R-елемент обрахунку суми, і в залежності від 19 того, чи перевищує зважена сума деякий поріг, R-елемент видає результат роботи перцептрону. Багатошаровий персептрон організовується з додатковими прихованими

шарами А-елементів, розташованими між S-елементами і R-елементами. Принципова складність завдань, що вирішуються багатошаровим персептроном, є найвищою для класу персептронів. Навчання елементарного і багатошарового персептрона полягає в зміні вагових коефіцієнт зв’язків А – R. Персептрон здатний працювати в режимі розпізнавання або узагальнення. Персептрон В одношаровому персептроні вхідні елементи безпосередньо пов'язані з вихідними за допомогою системи ваг, зв'язки S – A організовані за принципом однозначної відповідності. Одношаровий персептрон є окремим випадком класичного елементарного персептрону, найпростішою мережею прямого поширення – лінійним класифікатором, і має безліч принципових обмежень, таких як неможливість реалізації функції XOR. Когніторон був розроблений на основі будови біологічної зорової кори, має ієрархічну принципово багатошарову архітектуру. Нейрони між шарами когнітрону пов'язані тільки локально, і кожен шар реалізує різні 20 рівні узагальнення: вхідні шари сприймають прості образи, такі як лінії, великі однорідні ділянки, їх орієнтацію і локалізацію в просторі вхідних даних, в той час як глибокі шари сприймають складніші абстрактні структури, незалежні від локалізації та інших простих ознак образу.

Когнітрон організовується з ієрархічно пов'язаних збудливих і гальмуючих шарів. Співвідношення збуджуючих і гальмуючих сигналів на вході нейрона визначає його стан збудження. Існують спрощені моделі когнітрону, що будуються з одновимірних шарів, але спочатку когнітрон конструювався як каскад двовимірних шарів [11]. Пресинаптичний простір сигналів визначає виходи попереднього шару, постсинаптичний простір – входи наступного шару або площини. Нейрон когнітрону сприймає не весь постсинаптичний простір сигналів, а тільки його частину, реалізується принцип локальної зв'язності. Область пресинаптичного простору сигналів, що утворюють постсинаптичний простір сигналів, що впливають

на стан даного нейрона, називається його локальним рецептивних полем. Рецептивні поля близьких один до одного постсинаптичних нейронів, звані зонами конкуренції, перекриваються, тому активність даного пресинаптичного нейрона позначається на все більше поширення області постсинаптичних нейронів наступних шарів ієрархії. Розміри зон конкуренції обумовлюють кількість сприймання в просторі області ознак. Неокогнітрон є прямим розвитком ідеї, що лежать в основі когнітрону і точніше моделює структуру зорової кори головного мозку і є класифікатором, здатним до кращого розпізнавання об’єктів. Кожен шар неокогнітрона складається з площини простих S-нейронів і площини складних С-нейронів, що також організують локальну зв'язність [13]. Локальне рецептивне поле на площині S-нейронів наступного шару формується пресинаптичними сигналами площини С-нейронів попереднього шару. Локальні ознаки способу сприймаються S-нейронами, а спотворення локальних ознак компенсуються С-нейронами. В результаті цього процесу кожен шар після вхідного має своїм входом все більш узагальнену картину, утворену С-нейронами попередніх шарів[24]. З кожним рівнем глибини первинні прості ознаки визначаються у більш складних з’єднаннях. Площину S-нейронів можна розглядати як один нейрон, ваги якого визначають ядро згортки, що застосовуються до попереднього шару у всіх можливих позиціях. Всі С-нейрони реагують на образ, відповідний ядру згортки, в їх рецептивному полі, тому він визначається інваріантно до його локалізації. Сучасні глибокі згорткові нейронні мережі засновані на ідеях, що лежать в основі неокогнітрона, і сьогодні застосовуються для вирішення широкого кола завдань: від промислових, корпоративних та дослідницьких до повсякденно побутових, які вирішуються мобільними пристроями.

1.2 Поняття задачі пошуку нейронних мереж

Пошук нейронної архітектури (NAS) – це область дослідження, яка з’явилася завдяки різним зусиллям з автоматизації процесу архітектурного проектування нейронних мереж. NAS змогла створити безліч сучасних мереж [1], крім того досягнення в цій галузі запропонували методи, які не вимагають великої кількості ресурсів [2]. Процедуру NAS можна розділити на декілька компонент, кожен з яких сприяє траєкторії, а також результату пошуку. Найбільш відособленими компонентами є простір пошуку, метод оптимізації та метод оцінки кандидата. Простір пошуку визначає мережі, які можна дослідити для створення остаточної архітектури. Стверджується, що це найважливіший компонент при розробці процедур NAS, оскільки він може значно зменшити складність пошуку та, отже, обчислювальні вимоги. Вибір якісного простору пошуку може бути реалізован навіть випадковим пошуком для створення високопродуктивних архітектур. Тим не менше, визначення таких просторів має два застереження. По-перше, для цього потрібні попередні знання про набір даних, а це означає, що він не підходить для нових доменів. По-друге, він вводить упередженість, оскільки виключає різні архітектури, які ще не досліджені людьми. Метод оптимізації детермінує, як досліджувати пошуковий простір, що може сильно вплинути на ефективність пошуку, а також на ефективність остаточної запропонованої архітектури. Будучи проблемою оптимізації, NAS демонструє класичний компроміс між розвідкою та експлуатацією. Таким чином, вибір відповідної стратегії оптимізації може забезпечити достатнє вивчення обраного простору пошуку, тоді як запропонована архітектура буде максимально наближена до загального оптимуму. Метод оцінки кандидатів відповідає за порівняння проміжних результатів та допомогу стратегії оптимізації вибирати між різними варіантами на етапі пошуку. Оскільки оцінка властивостей архітектур глибокого навчання може

дорого коштувати через необхідну підготовку (тренування наприклад), використовуються різні методи для прискорення процесу. Схематична ілюстрація методів NAS наведена на рисунку 1.4.

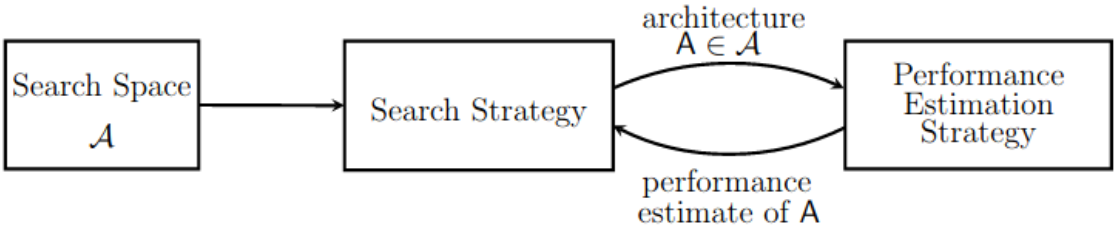

Рисунок 1.4 – Схематична ілюстрація методів NAS

На цьому рисунку показано, що з існуючого простору пошуку нейронних мереж метод оптимізації (або стратегія пошуку) обирає мережу та передає її до стратегії або методу оцінки кандидата. Метод оцінки повертає оцінку стратегії пошуку.

1.2.1 Простір пошуку

Простір пошуку визначає, які архітектури нейронних мереж метод NAS може винайти. Розгляньмо деякі приклади таких просторів. Відносно простий простір – це простір ланцюгово-струкурованих нейронних мереж, проілюстрований на рисунку 1.5. Така архітектура A може бути записана як

$$\mathrm{A} = L_n \circ \ldots L_1 \circ L_0 \quad , \tag{1.1}$$

де A – архітектура мережі;

n – кількість шарів;

L_i – і-тий шар;

∘ – абстрактний оператор.

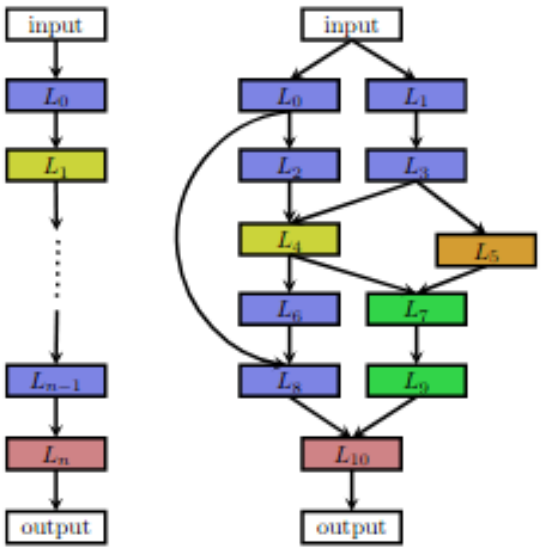

Рисунок 1.5 – Ілюстрація різних просторів пошуку

Кожен вузол на рисунку відповідає шару нейронної мережі, наприклад, згортковому. Різні типи шарів візуалізуються різними кольорами. Ребро означає, що шар отримує вихідні дані з попереднього слою. Зліва зображен елемент ланцюгово-структурованого простору. Зправа: елемент більш складного простору пошуку з додатковими типами шарів та кількома гілками які переходять з нижніх шарів одразу до найверхніх (skip-connections). Ми будемо розглядати прості повнозв’язні мережі з skip-connections. Вибір простору пошуку значною мірою визначає складність проблеми оптимізації: навіть у випадку простору пошуку, заснованого на одній комірці з фіксованою макроархітектурою, проблема оптимізації залишається неперервною і відносно високоюрозмірною (оскільки більш складні моделі, як правило, працюють краще, що призводить до більшої кількості варіантів дизайну). Зазначимо, що архітектури в багатьох

просторах пошуку можуть бути записані як вектори фіксованої довжини; наприклад, простір пошуку для кожної з двох клітинок можна записати як 40-вимірний простір пошуку з категоріальними вимірами, кожен з яких вибирає між невеликою кількістю різних будівельних блоків та входів. Необмежені пошукові простори можуть обмежуватися наявністю (потенційно дуже великої, але скінченної) кількості шарів, що знову породжує простори пошуку фіксованого розміру з (потенційно багатьма) умовними розмірами.

1.2.2 Стратегії пошуку

Багато різних стратегій пошуку можуть бути використані для дослідження простору нейронних архітектур, включаючи випадковий пошук, байєсівську оптимізацію, еволюційні методи, навчання з підкріпленням (RL) та методи на основі градієнта. Байєсівська оптимізація забезпечує техніку, засновану на теоремі Байєса, для пошуку глобального екстремуму функції. Вона працює шляхом побудови імовірнісної моделі цільової функції, так званої сурогатної функції, яку потім ефективно оптимізує за допомогою функції придбання, перш ніж вибірки кандидатів будуть обрані для оцінки реальної цільової функції. Байєсова оптимізація відзначила кілька ранніх успіхів у NAS з 2013 року, що призвело до створення сучасних архітектур комп'ютерного зору. Байєсова оптимізація – один з найпопулярніших методів оптимізації гіперпараметрів, але вона не застосовувався до NAS з багатьма групами, оскільки типові набори інструментів BO засновані на гауссових процесах і зосереджені на проблемах безперервної оптимізації низьких розмірів.

Щоб поставити NAS як проблему навчання з підкрірленням (RL), генерація нейронної архітектури може розглядатися як дія агента, при чому простір дій агента ідентичний простору пошуку даних. Винагорода агента

базується на оцінці ефективності навченої архітектури на валідаційних даних. Різні підходи RL відрізняються тим, як вони представляють політику агента та як її оптимізують: наприклад використовують політику рекуррентної нейронної мережі (RNN) для послідовного вибіру рядка, який, у свою чергу, кодує нейронну архітектуру.

Альтернативою використанню RL є нейроеволюційні підходи, що використовують еволюційні алгоритми для оптимізації нейронної архітектури. Перший такий підхід до проектування нейронних мереж датується майже трьома десятиліттями назад: Miller et al. [12] використовували генетичні алгоритми для пропонування архітектур та використовували зворотне розповсюдження похибки для оптимізації їх ваг. З тих пір у багатьох нейроеволюційних підходах використовуються генетичні алгоритми для оптимізації як архітектури, так і її ваг; однак при масштабуванні до сучасних нейронних архітектур з мільйонами ваг для завдань з вчителем, методи оптимізації ваг, засновані на SGD, в даний час перевершують еволюційні. Нещодавні нейроеволюційні підходи знову застосовують градієнтні методи для оптимізації ваг і використовують виключно еволюційні алгоритми для оптимізації нейронних архітектур. Еволюційні алгоритми розмножують сукупність моделей, тобто набір мереж (вони можлуть бути і натренованими); на кожному етапі еволюції відбирається принаймні одна модель із популяції, яка служить батьком для генерування нащадків шляхом застосування до нього мутації. У контексті NAS мутації – це локальні операції, такі як додавання або видалення шару, зміна гіперпараметрів шару, додавання skip-connections, а також зміна тренувальних гіперпараметрів. Після навчання нащадків оцінюється їх придатність (наприклад, показники ефективності на наборі), і вони додаються до популяції.

Хоча наведені вище методи використовують дискретний простір пошуку, в деяких роботах пропонують безперервну релаксацію задачі, щоб

забезпечити пряму оптимізацію на основі градієнта: замість фіксації однієї операції O_i для виконання на певному рівні можна обчислити опуклу комбінацію з набору операцій $\{o_1, \ldots, o_m\}$ або конкретніше, з урахуванням шару x, вихід шару y обчислюється як

$$y = \sum_{i=1}^{m} \alpha_i o_i(x), \alpha_i \geq 0, \sum_{i=1}^{m} \alpha_i = 1 \quad , \qquad (1.2)$$

де y – архітектура мережі;

α_i – опуклі коефіцієнти;

O_i – деяка операція;

m – кількість операцій.

Опуклі коефіцієнти α_i ефективно параметризують архітектуру мережі. Потім в цьому методі оптимізують як ваги мережі, так і архітектуру мережі, чергуючи кроки градієнтного спуску на навчальних даних для ваг та на даних валідації для архітектурних параметрів, таких як α. Врешті-решт, дискретна архітектура отримується шляхом вибору операції i^* з $i^* = \arg\max_i$ для кожного шару.

1.2.3 Стратегії оцінки кандидатів

Описані стратегії пошуку спрямовані на пошук нейронної архітектури A, яка максимізує певні показники продуктивності, такі як точність валідаційних даних. Для того, щоб управляти процесом пошуку, ці стратегії повинні оцінити ефективність даної архітектури A, яку вони розглядають. Найпростіший спосіб зробити це – навчити A на тренувальних даних та оцінити її результати на основі даних валідації. Розгляньмо показнки нейронної мережі: точність та втрату. Епоха –

прямий і зворотний прохід по всім тренувальним прикладам. Розмір серії (batch) – кількість тренувальних прикладів для однієї ітерації прямого і зворотного проходів. Кількість ітерацій – кількість проходів: кожен прохід використовує приклади (batch). Тобто маючи 1000 прикладів, batch = 500, нам буде потрібно зробити дві ітерації, щоб завершити одну епоху. З математичної точки зору, навчання нейронних мереж – багатопараметрична задача нелінійної оптимізації. Процес навчання мереж описують за допомогою графіку кривої, де горизонтальна вісь відображає кількість навчальних зразків, що отримує мережі, а вертикальна вісь відображає значення помилки, що отримано при класифікації вхідних даних.

Чим більше даних використовується для навчання, тим більше помилок видасть архітектура мережі, що відповідатимуть навчальним даним. У той же час навчальні дані стають більш схожими на справжній розподіл даних, які слід зафіксувати за навчальними даними. У певний момент часу помилка на навчальному та тестовому наборах повинна бути приблизно однаковою. Крива процесу навчання для встановлення є індикатором правильності роботи мережі, якщо велика кількість навчальних даних без будь-яких змін, що в свою чергу покращує ефективність мережі. Побудувавши криву навчального набору, можна оцінити, чи здатна архітектурна модель мережі відповідати даним для потрібної класифікації помилки. Значення помилки при тестовому наборі ніколи не повинне бути значно нижчим, ніж значення помилки навчального набору. Навчання кожної архітектури, яку слід оцінювати з нуля, часто призводить до великих обчислювальних вимог до GPU. Це обмеженя веде до розробки методів для прискорення оцінки ефективності, які зараз будуть описані.

Ми можемо апроксимувати метрику точності нейрнної мережі, тренуя її на невеликій кількості епох або на підмножині тренувального датасету, або спрощуючи подану архітектуру. Хоча ці наближення з proxy

точністю зменшують обчислювальні витрати, вони також вводять упередження в оцінку, оскільки результативність занижується. Це може не завдавати стратегії великої шкоди, поки вона покладається лише на ранжування різних архітектур, а відносне ранжування залишається стабільним. Однак нещодавні результати вказують на те, що цей відносний рейтинг може різко змінитися, коли різниця між дешевими наближеннями та «повною» оцінкою занадто велика.

Інший можливий спосіб оцінки продуктивності архітектури ґрунтується на екстраполяції кривої навчання. Пропонується екстраполювати початкові криві навчання та припинити ті, які, як передбачається, будуть неефективними, щоб пришвидшити процес пошуку архітектури. Також розглядають архітектурні гіперпараметри для прогнозування того, які часткові криві навчання є найбільш перспективними. Навчання сурогатної моделі для прогнозування ефективності нових архітектур також можливе, якщо не використовувати екстраполяцію кривої навчання, але підтримувати прогнозування ефективності на основі архітектурних властивостей та екстраполювати на архітектури з більшими розмірами. Основною проблемою для прогнозування результатів нейронних архітектур є те, що для прискорення процесу пошуку потрібно робити хороші прогнози у відносно великому просторі пошуку на основі порівняно невеликої кількості оцінок.

Ще один підхід до прискорення оцінки продуктивності – це ініціалізація ваг нових архітектур на основі ваг інших архітектур, які вже навчались раніше. Один із способів досягти цього дозволяє змінювати архітектуру, залишаючи функції активації мережі без змін. Це дозволяє послідовно збільшувати пропускну здатність мереж і зберігати високу продуктивність, не вимагаючи навчання з нуля. Постійне навчання для кількох епох може також використовувати додаткову ємність, введену мережевими морфізмами. Перевага цих підходів полягає в тому, що вони

дозволяють простори пошуку без властивого їм верху, що обмежується розмірами архітектури; з іншого боку, суворі мережеві морфізми можуть лише збільшити архітектури і, отже, можуть призвести до надто складних архітектур.

Незважаючи на те, що NAS досягнув вражаючих показників, поки що він дає мало уявлень про те, чому конкретні архітектури працюють добре, і наскільки подібними будуть архітектури, отримані в незалежних прогонах. Визначення загальних блоків, надання розуміння того, чому ці блоки важливі для високих показників, та дослідження, чи ці блоки можуть узагальнювати різні проблеми, є ключовим напряком подальшого розвитку NAS.

1.3 Нейронні мережі з випадковими вагами

1.3.1 Мотивація до методу побудови нейронних мереж з випадковими вагами

У біології зрілонародженими видами вважаються ті, чиї молоді особини вже володіють певними здібностями з моменту народження. Існують дані, які показують, що ящірки [19] та змії [14] після вилуплення вже мають поведінку, яка дозволяє їм врятуватися від хижаків. Незабаром після вилуплення качки можуть плавати і їсти, а індики можуть візуально розпізнавати хижаків. На відміну від цього, коли ми навчаємо штучних агентів виконувати завдання, ми, як правило, вибираємо архітектуру нейронної мережі, яку ми вважаємо придатною для кодування політики для завдання, і знаходимо ваги цієї політики за допомогою алгоритму навчання. В цьому методі розробляються нейронні мережі з архітектурами, які здатні виконувати певне завдання, навіть якщо параметри їх ваги випадково відбираються. Використовуючи такі архітектури нейронних

мереж, агенти вже можуть добре працювати в своєму середовищі без необхідності вивчати ваги. Крім того, ми можемо значно спростити алгоритм NAS, переходячи одразу до стадії евалюації мережі на валідаційних даних.

Десятиліття досліджень нейронних мереж забезпечили будівельні блоки з сильними індуктивними упередженнями для різних сфер застосування. Наприклад, згорткові мережі [7] особливо підходять для обробки зображень [17]. Недавня робота [13] продемонструвала, що навіть випадково ініціалізовані CNN можуть бути ефективно використані для таких завдань обробки зображень, як superresolution (підвищення якості зображення), inpainting (фарбування чорно-білого зображення), style transfer (передача стилю). У роботі [13] показали, що випадково ініціалізований LSTM з вивченим лінійним вихідним шаром може передбачати часові ряди, тоді як традиційні резервуарні RNN [19] виходять з ладу. Останні розробки в мережах self-attention [22] та капсульних нейронних мереж [23] розширюють набір інструментальних елементів для створення архітектур з потужними індуктивними упередженнями для різних завдань. У цій роботі я буду шукати повнозв’язні (для простоти) архітектури з потужними індуктивними упередженнями, які вже можуть виконувати різні завдання із випадковими вагами. Це досягається шляхом присвоєння єдиного спільного параметра ваги кожному мережевому вузлу та оцінки мережі в широкому діапазоні цього єдиного параметра ваги (оцінюємо мережу декілка раз). Замість оптимізації ваг фіксованої мережі, виконується оптимізація для архітектур, які працюють добре в широкому діапазоні ваг. Перед тим як зробити постановку задачі, розгляньмо детальніше деякі евристики з суміжних областей для того, щоб покращити алгоритм, та чітко розуміти його властивості та отриманий результат.

1.3.2 Нейроеволюційний алгоритм NEAT

Еволюційні алгоритми – це евристичний підхід до розв’язання задач, які неможливо легко розв’язати за поліноміальний час, наприклад класичні NP-Hard задачі, та будь-що інше, на що для вичерпної обробки знадобиться занадто багато часу. Вони зазвичай застосовуються до комбінаторних проблем; однак генетичні алгоритми часто використовуються в поєднанні з іншими методами, що дає швидкий спосіб знайти субоптимальне початкове місце для відпрацювання іншого алгоритму. Передумова еволюційного алгоритму (який далі називатиметься ГА) досить проста, вона схожа на процес природного відбору. ГА містить чотири загальних етапи: ініціалізація групи агентів, відбір, відбір признаків та завершення еволюції, це зображено на Рисунку 1.6. Кожен із цих кроків приблизно відповідає певному етапу природного відбору та забезпечує прості способи формалізації та інтепретації цієї категорії алгоритмів.

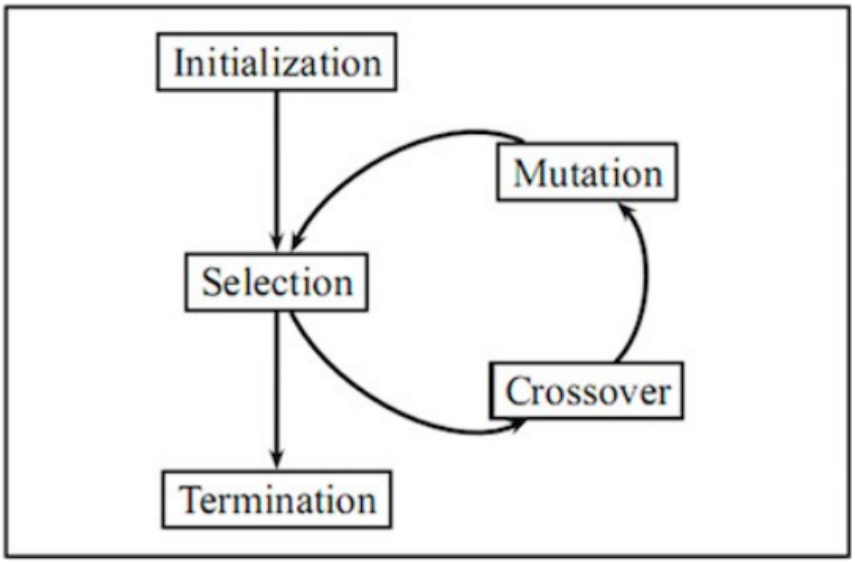

Рисунок 1.6 – Схема генетичного алгоритму

Формально задача еволюції нейронних архітектур ставиться так: ми хочемо знайти найкращу комбінацію елементів, яка максимізує певну функцію фітнесу; буде прийнято рішення про зупинку алгоритму, коли ми або запустимо алгоритм для деякої максимальної кількості ітерацій, тобто вичерпаємо апаратний ресурс, або ми досягнемо певного порогу придатності.

Для того, щоб розпочати наш алгоритм, ми повинні спочатку створити початкову сукупність рішень. Населення буде містити довільну кількість можливих рішень проблеми. Ініціалізація часто створюється випадковим чином (в рамках обмежень проблеми) або, якщо відомі деякі попередні знання про завдання, приблизно зосереджується навколо того, що вважається ідеальним. Важливо, щоб популяція охоплювала широкий спектр рішень, оскільки вона по суті являє собою генофонд, тож якщо ми хочемо дослідити безліч різних можливостей протягом еволюції, ми повинні прагнути мати багато різних генів.

Як тільки популяція створена, члени популяції тепер повинні оцінюватися відповідно до функції придатності. Фітнес-функція – це функція, яка приймає характеристики члена та виводить числове представлення того, наскільки життєздатним є рішення. Формалізація функції фітнесу часто може бути дуже складним процесом, і важливо знайти хорошу функцію, яка точно передає сутність проблеми. Потім обчислюється підготовленість усіх членів та вибирається частина членів, які отримали найбільший бал. Безліч цільових функцій ГА також можна застосувати для використання декількох функцій фітнесу. Це дещо ускладнює процес, оскільки замість того, щоб змогти визначити одну оптимальну точку, ми, натомість, отримуємо набір оптимальних точок при використанні декількох функцій фітнесу. Сукупність оптимальних рішень називається кордоном Парето і містить елементи, однаково оптимальні в тому сенсі, що жодне рішення не домінує над будь-яким іншим рішенням

на кордоні. Потім використовується евристичний алгоритм для звуження набору єдиного рішення на основі контексту задачі чи іншої метрики. Приклад оптимальних по Парето рішень зображен на рисунку 1.7.

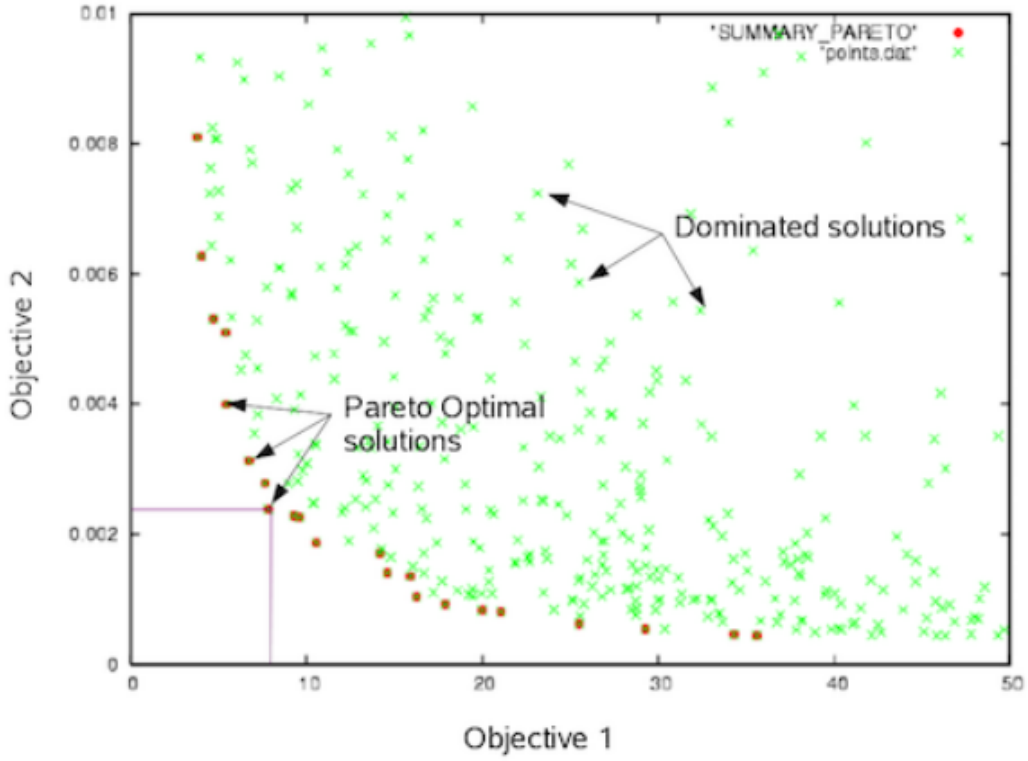

Рисунок 1.7 – Приклад оптимальних по Парето рішень

Розгляньмо застосування генетичних операторів. Цей крок насправді включає два підкроки: кросовер та мутацію. Після вибору найкращих членів, ці члени тепер використовуються для створення наступного покоління в алгоритмі. Використовуючи характеристики вибраних батьків, створюються нові діти, які є сумішшю якостей батьків. Зробити це часто може бути важко залежно від типу даних. Тепер ми повинні ввести новий генетичний матеріал у покоління. Якщо ми не зробимо цього найважливішого кроку, ми дуже швидко застрягнем у локальних екстремумах і не отримаємо оптимальних результатів. Цей крок є

мутацією, і ми робимо це досить просто, а саме змінюючи невелику частину дітей таким чином, що вони вже не ідеально відображають підмножини генів батьків. Мутація, як правило, відбувається імовірнісно, тому шанс дитини отримати мутацію, а також тяжкість мутації регулюються розподілом ймовірностей.

Зрештою, алгоритм повинен закінчитися. Є два випадки, коли це зазвичай трапляється: або алгоритм досяг якогось максимального часу виконання, або алгоритм досяг певного порогу продуктивності. На цьому етапі вибирається та повертається остаточне рішення. До NEAT було кілька спроб застосовувати алгоритм еволюційного пошуку до топології мереж, які були дещо успішними, проте вони виявили низку проблем. Тож розгляньмо деякі специфічні рішення NEAT для застування ГА до пошуку нейроархітектур, які роблять цей алгоритм актальим і сьогодні.

У біології ми маємо генотип і фенотип. Генотип - це генетичне уявлення про істоту, а фенотип – це актуалізоване фізичне уявлення про істоту. Ця ідея була застосована до кодування архітекур. Питання кодування випливає з питання про те, як ми хочемо представляти архітектури в нашому алгоритмі. Спосіб, яким ми кодуємо своїх індивідів, визначає шлях до того, як наш алгоритм буде обробляти ключові еволюційні процеси: відбір, мутацію та кросовер (також відомий як рекомбінація). Будь-яке кодування потрапляє в одну з двох категорій, пряму чи непряму. Пряме кодування чітко вказує все про індивід (архітектуру). Якщо він представляє нейронну мережу, це означає, що кожен ген буде безпосередньо пов'язаний з яким-небудь вузлом, з'єднанням або властивістю мережі. Це може бути двійкове кодування одиницями та нулями, кодування графом (зв’язування різних вузлів зваженими зв’язками) або щось ще більш складне. Справа в тому, що між генотипом і фенотипом завжди буде прямий зв’язок, який є очевидним. Непряме кодування – це прямо протилежна річ. Замість прямї вказівки того, як може виглядати

структура, непряме кодування, як правило, визначає правила або параметри процесів для створення особистості. Як результат, непряме кодування набагато компактніше. Недолік такого підходу полягає в тому, що встановлення правил непрямого кодування може призвести до значного упередження в просторі пошуку, тому набагато складніше створити непряме кодування без значних знань про те, як кодування буде використовуватися у контексі поставленої задачі. Через це алгоритм NEAT обирає методологію прямого кодування. Їх представлення трохи складніше, ніж простий графік або двійкове кодування, проте зрозуміти це все одно просто: у ньому просто є два списки генів, серія вузлів і серія з'єднань. Візуально це показано на рисунку 1.8.

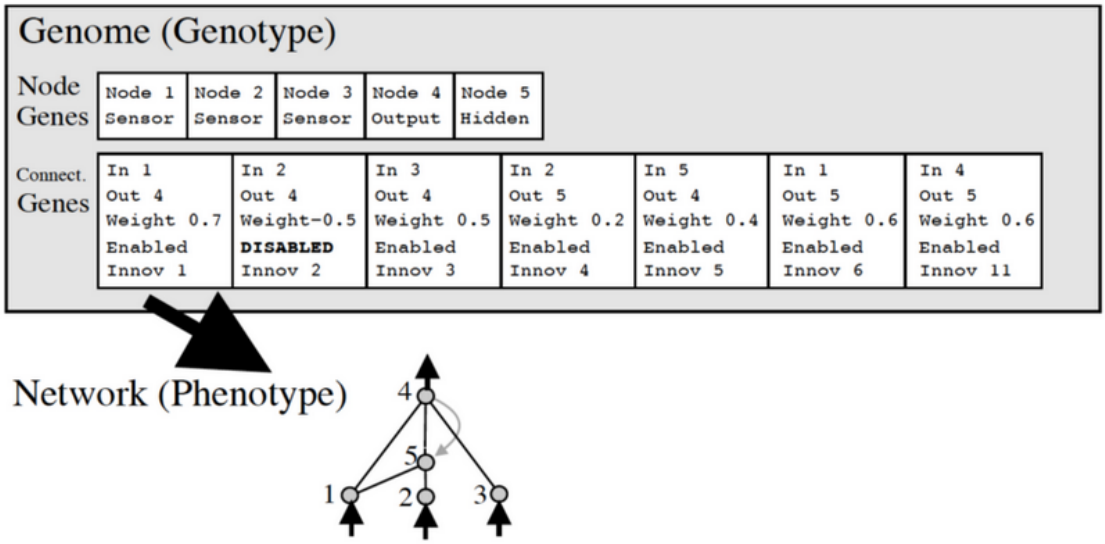

Рисунок 1.8 – Методологія прямого кодування

Приховані вузли можна додавати або видаляти. Що стосується вузлів підключення, вони вказують, де з’єднання входить і виходить з нього, вагу такого з’єднання, чи вмикається зв’язок, та номер інновації.

У NEAT мутація може або змінюватии існуючі з'єднання, або додати нову структуру до мережі. Якщо між початковим і кінцевим вузлом додається нове з'єднання (вузол), йому випадково присвоюється вага. Якщо додається новий вузол, він розміщується між двома вузлами, які вже підключені. Попереднє з'єднання буде відключено (хоча все ще буде присутнє в геномі). Попередній початковий вузол буде пов'язан з новим вузлом з вагою старого з'єднання, а новий вузол буде пов'язан з попереднім кінцевим вузлом з вагою 1. Це було встановлено, щоб допомогти пом'якшити проблеми з новими структурними доповненнями, алгоритм мутації зображен на Рисунку 1.9.

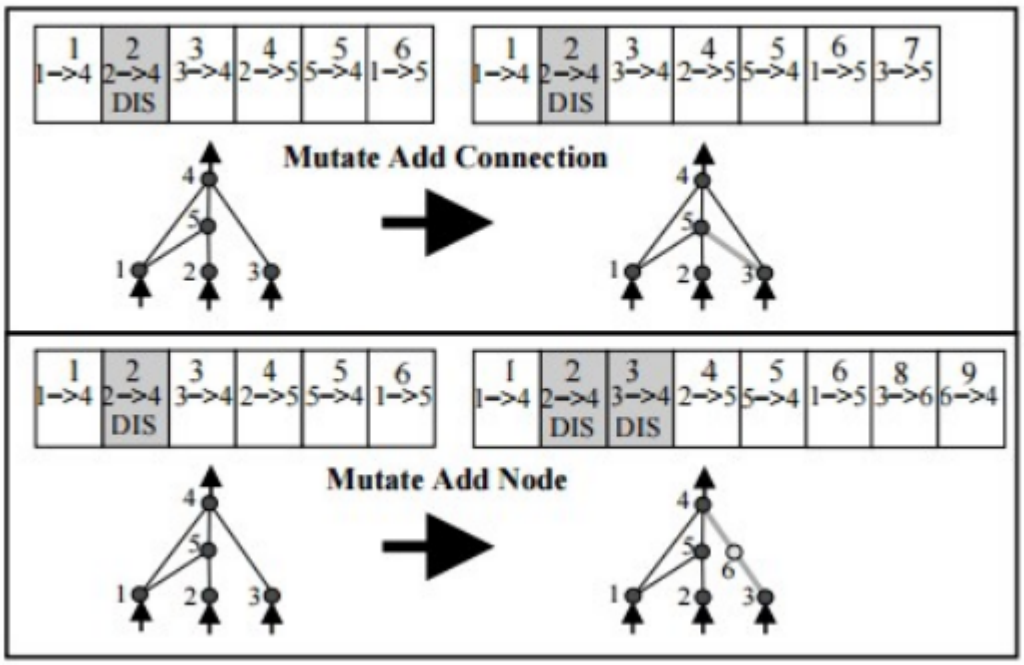

Рисунок 1.9 – Схема мутації

Ще одним вирішенням великої проблеми розвитку топологій нейронних мереж є те, що в алгоритмі NEAT називається «конкуруючими

умовами». Ідея полягає в тому, що просто сліпе перетинання геномів двох нейронних мереж може призвести до жахливо мутованих та нефункціональних мереж. Якщо дві мережі залежать від центральних вузлів, які обидва виходять з мережі у результаті кросінговеру, це буде проблемою.

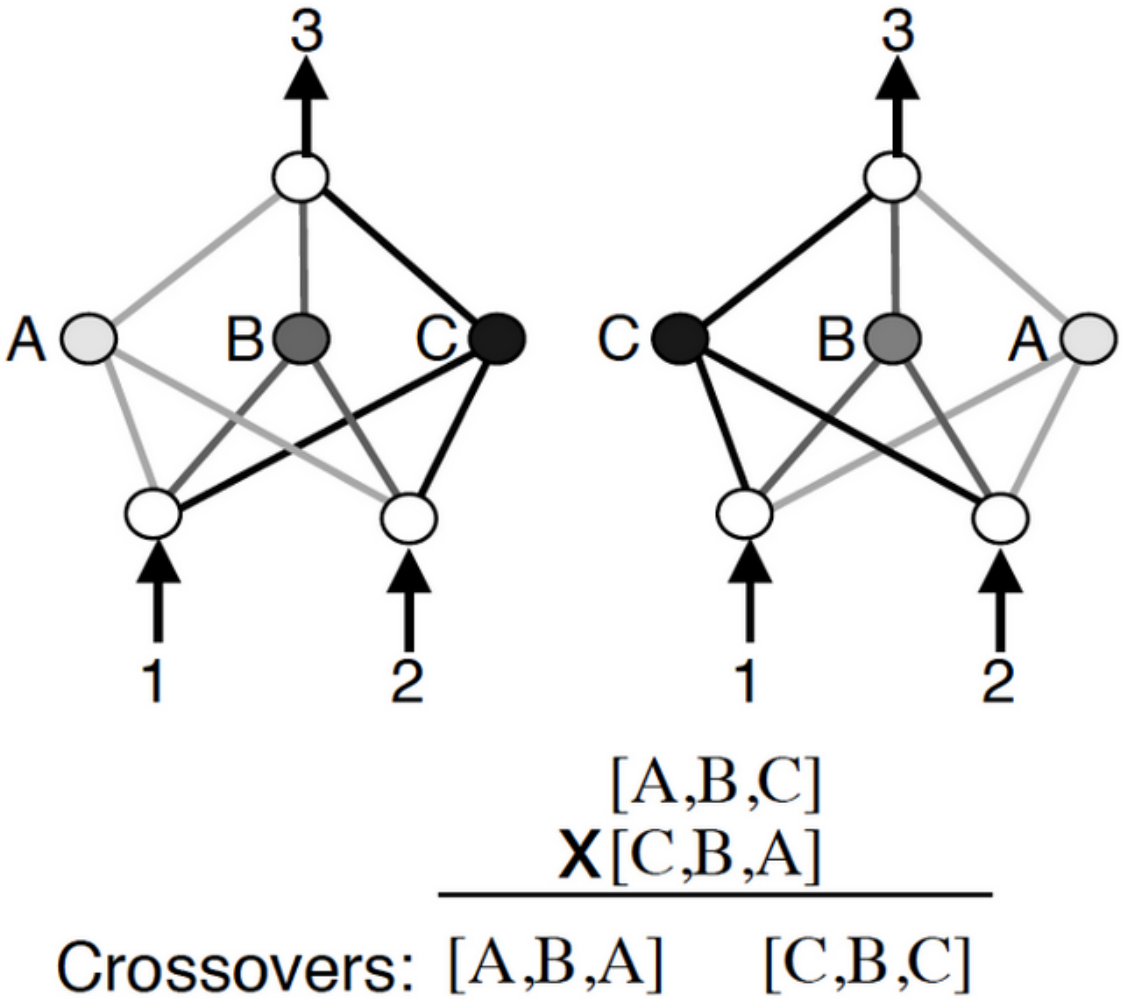

Рисунок 1.10 – Схема кросоверу

Більше того, геноми можуть бути різних розмірів. Як ми вирівняємо геноми, які не є очевидно сумісними? У біології це вирішується за допомогою ідеї, яка називається гомологією. Гомологія – це вирівнювання хромосом на основі відповідності генів за певною ознакою. Як тільки це станеться, перехрещення може статися з набагато меншим шансом помилки, ніж якби хромосоми були сліпо змішані між собою. NEAT вирішує цю проблему шляхом використання історичних маркувань (як показано вище). Позначаючи нові еволюції історичним числом, коли настає час схрещування двох індивідів, це можна зробити з набагато меншими шансами створити особи, які не функціонують. Кожен ген може бути вирівняний та (потенційно) схрещений. Кожного разу, коли виникає новий вузол або новий тип підключення, призначається історичне маркування, що дозволяє легко вирівнювати, коли мова заходить про породження двох наших особин.

Дуже цікава ідея, висунута в NEAT, полягала в тому, що більшість нових еволюцій не є хорошими. Насправді, додавання нового з'єднання або вузла до будь-якої оптимізації ваг часто призводить до того, що створюється особа з меншими показниками. Це ставить нові більш складні структури у невигідне становище. Як ми можемо захистити нові структури і дозволити їм оптимізуватись, перш ніж повністю виключити їх із населення? Алгоритм NEAT пропонує видоутворення. Видоутворення просто розділяє популяцію на кілька видів на основі подібності топології та зв’язків. Якби конкурентна конвенційна проблема все ще існувала, це було б дуже важко виміряти. Однак, оскільки NEAT використовує історичне маркування у своєму кодуванні, це стає набагато легше виміряти. Функція вирішення способу розжілу на види наведена в роботі, але важливо відзначити, що особини в популяції повинні лише конкурувати з іншими особинами в межах цього виду. Це дозволяє створювати та оптимізувати нову структуру, не боячись, що вона буде ліквідована до того,

як її можна буде по-справжньому дослідити. Більше того, NEAT робить ще один крок вперед завдяки ідеї, яка називається явним обміном фітнесом всередені видів. Це означає, що особини повідомляють одна одній, наскільки добре вони працюють всередні виду, збільшуючи високоефективні види, хоча все-таки дозволяються іншим видам досліджувати свою оптимізацію структури, перш ніж вони будуть усунені з популяції.

Однією з головних цілей алгоритма NEAT було створення фреймворку для утвореня мінімальних за розміром мереж. Автори не хотіли створювати алгоритм, який спочатку знаходив хороші мережі, а потім фактично повинен був зменшити кількість вузлів та з'єднань. Натомість ідея полягала в тому, щоб побудувати алгоритм, який починався з мінімальної кількості вузлів та з'єднань, що ускладнюється з часом, якщо і лише тоді, коли це виявиться корисним та необхідним. NEAT встановлює свій алгоритм розвитку мінімальних мереж, запускаючи всі мережі без прихованих вузлів. Кожна особа в початковій сукупності – це просто вхідні вузли, вихідні вузли та ряд генів зв’язку між ними. Само по собі це не обов'язково повинно спрацювати, але в поєднанні з ідеєю видоутворення це виявляється потужною ідеєю розвитку мінімальних, але високопродуктивних мереж. Розгляньмо ще декілька ідей, які можуть розвинути ідею побудови мінімальних і водночас продуктивних нейронних мереж.

1.3.3 Байесівські нейронні мережі

Суть BNN досить проста – кожна вага пов'язана з розподілом ймовірностей, включаючи ваги та вектор вільних членів. У байєсівському світі існують значення, що називаються «випадковими величинами», яке надає різне значення кожного разу, коли до нього звертаються. Нехай X є

випадковою величиною, яка представляє нормальний розподіл; кожен раз, коли ви отримуєте доступ до X, воно матиме інше значення. Цей процес отримання різного значення для кожного доступу називається вибіркою. Яке значення вийде з кожної вибірки, залежить від розподілу ймовірностей. Чим ширший розподіл ймовірностей, тим більша невизначеність. У традиційній нейронній мережі кожен шар має фіксовані ваги та вільні члени, які визначають вихід. Але нейромережа Байєса матиме розподіл ймовірностей, прикріплений до кожного шару. Для проблеми класифікації ви виконуєте кілька проходів вперед кожного разу з новими зразками ваг. Для кожного прямого проходу передбачений один вихід. Невизначеність буде високою, якщо вхідне зображення – це те, що мережа ніколи не бачила для всіх вихідних класів.

Баєсові методи забезпечують природний підхід до кількісної оцінки невизначеності в процесі глибокого навчання. Байєсові нейронні мережі часто мають кращу міру калібрування, ніж класичні нейронні мережі, тобто їх прогнозована невизначеність більше відповідає спостережуваним помилкам. Іншими словами, вони не є ані надмірно самовпевненими, ані недостатньо самовпевненими в порівнянні зі своїм не-байєсівським методом. Вагові параметри BNN не є фіксованими значеннями, а семплюються з розподілу. Хоча параметри цього розподілу можна вивчити [30], кількість параметрів часто перевищує кількість ваг. Нещодавно були запропоновані [34] дисперсійні мережі, які відбирають кожну вагу з розподілу з нульовим середнім значенням та вивченим параметром дисперсії, і показують, що оцінка ансамблю може покращити ефективність завдань розпізнавання зображень. Я застосовую подібний підхід, відбираючи ваги нейрона з фіксованого рівномірного розподілу з нульовим середнім значенням, а також оцінюючи ефективність мережевих ансамблів.

1.3.4 Елементи алгоритмічної інформаційної теорії

Ця ідея розвиває ключову ідею необхідності мінімальної складності вихідної мережі. В AIT (алгоритмічній інформаційній теорії) складність Колмогорова [23] обчислюваного об'єкта є мінімальною довжиною програми, яка може його обчислити. Мінімальна довжина опису (МДО) – це формалізація бритви Оккама, в якій хороша модель - це та, яка найкраще стискає її дані, включаючи вартість опису самої моделі. Ідеї, пов'язані з МДО, з метою зробити нейронні мережі «простими» були запропоновані в 1990-х роках, такі як спрощення мереж шляхом спільного використання ваг, зменшення обсягу інформації у вагах, додаючи до ваг шум, та спрощення простору їх пошуку. Останні роботи пропонують сучасну обробку та застосування [27] цих принципів у контексті більших, глибоких архітектур нейронних мереж. Хоча вищезазначені роботи зосереджуються на інформаційній спроможності, необхідній для представлення ваг заздалегідь визначеної архітектури мережі, в цій роботі AIT застосовується до пошуку мінімальних архітектур, які можуть представляти рішення різних завдань. Оскільки наші мережі все ще вимагають ваг, у цьому підході розглядається вага як випадкова величина, відібрана з фіксованого розподілу

1.3.5 Зкорочення великих нейронних мереж (network pruning)

Видаляючи з навченої нейронної мережі з'єднання з малими значеннями ваги, такі підходи до обрізки можуть створити розріджені мережі, які утримують лише невелику частку з'єднань, зберігаючи схожі ефективності для завдань класифікації зображень порівняно з повною мережею. Зберігаючи вивченні значення ваг, ці роздільні мережі можна навчити навіть з нуля, щоб досягти вищої точності на валідації [23], ніж

оригінальні мережі. Подібно до цього методу, у цій роботі виявилені зкорочені мережі, які можуть досягти точності класифікації зображень, яка є набагато кращою, ніж випадкова точність, навіть із випадково ініціалізованими вагами. Зкорочення мереж є додатковим підходом до цього; він починається з повноцінної, навченої мережі і забирає зв’язки, тоді як у цьому підході ми починаємо з одного зв’язку і додаємо складність по мірі необхідності. Порівняно з цим підходом, обрізка вимагає попереднього навчання всієї мережі, щоб заздалегідь отримати корисну інформацію про кожну вагу. Крім того, зкорочені архітектури, обмежені повною мережею, тоді як у досліджуваному методі немає верхньої межі складності мережі.

1.3.6 Зв’язок з нейронаукою

Штучні нейронні мережі (ШНМ) пережили революцію, каталізатором якої стали контрольовані алгоритми навчання. Однак, на відміну від молодих тварин (включаючи людей), навчання таких мереж вимагає величезної кількості позначених прикладів, що призводить до переконання, що тварини повинні покладатися переважно на навчання без нагляду. Стверджується, що поведінка більшості тварин не є результатом розумних алгоритмів навчання – під наглядом чи без нагляду – а закодована в геномі. Зокрема, тварини народжуються з високоструктурованими мозковими зв’язками, що дозволяє їм дуже швидко вчитися. Оскільки схема підключення є занадто складною, щоб бути чітко визначеною в геномі, її необхідно стиснути через «геномне вузьке місце». Геномне вузьке місце передбачає шлях до ШНМ, здатних до швидкого навчання. Після винаходу комп’ютерів у 1940-х роках багато хто вірив, що комп'ютери незабаром досягнуть або перевершать інтелект людського рівня. Герберт Саймон, піонер штучного інтелекту (ШІ), у 1965 році

передбачив, що «машини будуть здатні протягом двадцяти років виконувати будь-яку роботу, яку може виконати людина» – для досягнення загального ШІ. Звісно, ці прогнози виявилися дуже неправдивими. Сьогодні у світі технологій знову високий оптимізм. Значна частина цього оновленого оптимізму випливає з вражаючих останніх досягнень у галузі штучних нейронних мереж (ШНМ) і машинного навчання, зокрема «глибокого навчання». Застосування цих методів – для машинного зору, розпізнавання мовлення, автономних транспортних засобів, машинного перекладу та багатьох інших областей – набуває настільки швидкого темпу, що багато хто передбачає, що ми наближаємося до «технологічної сингулярності», моменту, коли штучний суперінтелект запускає швидке зростання і трансформує людську цивілізацію. У цьому сценарії, коли комп’ютери збільшуються в потужності, стане можливим побудувати машину, яка буде розумнішою, ніж розробники. Ця надінтелектуальна машина побудує ще більш розумну машину, і врешті-решт цей рекурсивний процес буде прискорюватися, поки інтелект не досягне меж, встановлених фізикою чи інформатикою. Але, незважаючи на цей прогрес, ШНМ все ще далекі від наближення до людського інтелекту. ШНМ може обіграти опонентів-людей в таких іграх, як шахи та го, але за більшістю активностей – мова, міркування, здоровий глузд – вони не можуть наблизитися до когнітивних здібностей чотирирічної дитини. Можливо, більш вражаючим є те, що ШНМ ще більш невдало копіюють здібності простих тварин. Багато з найпростіших видів поведінки – поведінки, які здається невимушеною навіть для простих тварин – виявляється оманливо складною та недоступною для ШІ.

Ми не можемо побудувати машину, здатну будувати гніздо, переслідувати здобич чи завантажувати посудомийну машину. Багато в чому ШІ далекий від досягнення інтелекту собаки, миші чи навіть павука, і не здається, що просте розширення сучасних підходів дозволить досягти

цих цілей. Гарна новина полягає в тому, що, якщо нам коли-небудь вдасться досягти навіть інтелекту на рівні миші, людський інтелект може бути зімітований дуже швидко. Наші хребетні предки, які з’явилися приблизно 500 мільйонів років тому, могли мати приблизно інтелектуальні здібності акули. Великим стрибком в еволюції нашого інтелекту стала поява неокортексу, основна організація якого була вже встановлена, коли перші плацентарні ссавці виникли близько 100 мільйонів років тому [12]; велика частина людського інтелекту, здається, походить від появи неокортексу. Сучасні люди (Homo sapiens) еволюціонували лише кілька сотень тисяч років тому – мить в еволюційному часі – що свідчить про те, що такі якості, як мова та розум, які ми вважаємо унікально людськими, насправді можуть бути відносно легко досягнутими за умови міцної нейронної основи. Хоча існують гени і, можливо, типи клітин, унікальні для людини – так само, як і для будь-якого виду, – немає жодних доказів того, що людський мозок використовує якісь принципово нові нейробіологічні принципи, яких ще немає у миші (або будь-якого іншого ссавця), тому розрив між мишею та людським інтелектом може бути набагато меншим, ніж інтелект між нинішнім ШІ та мишею. Це свідчить про те, що навіть якщо наша кінцева мета полягає у досягненні (або навіть перевищенні) людського інтелекту, розумною найближчою метою для ШІ буде відповідати інтелекту миші. Як випливає з назви, ШНМ були винайдені у спробі побудувати штучні системи на основі обчислювальних принципів, які використовуються нервовою системою [13]. Далі ми припускаємо, що додаткові принципи нейронауки можуть прискорити досягнення мети створення штучного мишачого і, зрештою, людського інтелекту. Стверджують, що на відміну від ШНМ, тварини значною мірою покладаються на комбінацію як засвоєних, так і вроджених механізмів. Ці вроджені процеси виникають в результаті еволюції, кодуються в геномі і набувають форми правил для підключення мозку6. Зокрема, є поняття

«геномного вузького місця» – стиснення в геном будь-яких вроджених процесів, які фіксуються еволюцією – як регулююче обмеження на правила підключення мозку. По суті, підкреслення важливості вроджених механізмів є лише нещодавнім проявом давньої дискусії «природа проти виховання», яка точилася принаймні з пір стародавніх греків. Історично, велика частина цієї дискусії зосереджувалась на набутті характеристик, таких як інтелект і моральність, які традиційно вважалися специфічно людськими. Термін «чиста дошка», введений Локком, став скороченням важливості виховання. Через століття Іммануїл Кант виступав на користь природи у своїй впливовій «Критиці чистого розуму». Зовсім недавно дебати розгорнулися в таких дисциплінах, як когнітивна психологія та лінгвістика.

Термін «навчання» використовується по-різному в ШНМ та нейронауці. На найбільш абстрактному рівні навчання можна визначити як процес кодування статистичних закономірностей із зовнішнього світу в параметри (переважно вагові показники) мережі. Але в контексті навчання тварин джерело вхідних даних для навчання обмежується лише «досвідом» тварини, тобто тими подіями, які відбуваються протягом життя тварини. Правила підключення, закодовані в геномі, які не залежать від досвіду, як-от ті, що використовуються для підключення сітківки, зазвичай не називаються «навчанням». Оскільки терміни «життя» та «досвід» не є чітко визначеними, коли застосовуються до ШНМ, узгодження два визначення навчання в ІНС проти нейронауки становлять виклик. Якщо, як ми стверджували вище, велика частина поведінки тварини є вродженою, то життєвий досвід тварин представляє лише невелику частину даних, які сприяють її придатності; інший потенційно набагато більший пул даних сприяє його вродженій поведінці та уявленням. Ці вроджені поведінки та уявлення виникають внаслідок еволюції шляхом природного відбору. Таким чином, еволюцію, як і навчання, можна також розглядати як

механізм вилучення статистичних закономірностей, хоча й у набагато більшому часовому масштабі, ніж навчання.

Еволюцію можна розглядати як свого роду підкріплювальний алгоритм, що діє на часовій шкалі поколінь, де сигнал підкріплення складається з кількості потомків, які породжує індивід. ШНМ беруть участь у процесі оптимізації, який повинен імітувати як те, що було засвоєно в ході еволюції, так і процес навчання протягом життя, тоді як для тварин навчання стосується лише змін протягом життя. З цієї точки зору, контрольоване навчання в ШНМ не слід розглядати як аналог навчання в тварин. Натомість, оскільки більшість даних, які сприяють придатності тварин, закодовані еволюцією в геномі, було б точним перейменувати його в «контрольовану еволюцію». Таке перейменування підкреслить, що «навчання під наглядом» ШНМ дійсно узагальнює вилучення статистичних закономірностей, які відбуваються у тварин як при еволюції, так і завдяки навчанню. У тварин є два вкладених процеси оптимізації: зовнішній цикл «еволюції», що діє на часовому масштабі поколінь, і внутрішній цикл «навчання», який діє протягом життя окремої особини. Контрольована (штучна) еволюція може бути набагато швидшою за природну, яка досягає успіху лише тому, що вона може отримати користь від величезної кількості даних, представлених життєвим досвідом квадрильйонів тварин протягом сотень мільйонів років.

Хоча є паралелі між навчанням та еволюцією, є також важливі відмінності. Примітно, що в той час як навчання може діяти безпосередньо на синаптичні ваги за допомогою Гебіанського [17] та інших механізмів, еволюція діє на мозок лише опосередковано, через геном. Геном не кодує репрезентації чи поведінку безпосередньо; він кодує правила підключення та мотиви (motifs/патерни) підключення. Обмежена ємність геному ссавців – на порядок менша, ніж була б потрібна для явного визначення всіх зв’язків – може діяти як регуляризатор або інформаційне вузьке місце,

зміщуючи баланс від дисперсії до зміщення (упередження). У зв’язку з цим цікаво відзначити, що сам розмір геному не є фіксованим обмеженням, а сам по собі сильно варіюється для різних видів. Розмір людського геному приблизно середній для ссавців, але менший за розміром у багатьох риб і земноводних; мармуровий геном у легені риби більш ніж у 40 разів більший, ніж у людини [17]. Той факт, що людський геном потенційно міг бути набагато більшим, свідчить про те, що ефект регуляризації, викликаний обмеженими можливостями генома, може представляти собою особливість, а не помилку.

Отож, підкреслимо, що тварини здатні добре функціонувати швидко після народження, оскільки вони народжуються з високоструктурованими мозковими зв’язками. Ця зв’язність створює риштування, на якому може відбуватися швидке навчання. Таким чином, вроджені механізми працюють синергетично з навчанням. Ми припускаємо, що аналогічні підходи можуть надихнути на нові підходи для прискорення прогресу в ШНМ. Перше спостереження з нейронауки полягає в тому, що велика частина поведінки тварин є вродженою, а не виникає внаслідок навчання. Тваринний мозок – це не чиста дошка, оснащена загальним алгоритмом навчання, готовим навчитися чому завгодно, як припускають деякі дослідники ШІ; існує сильний тиск відбору на тварин, щоб обмежити їхнє навчання лише тим, що необхідно для їхнього виживання. Ідея про те, що тварини схильні швидко навчатися певним речам, пов’язана з «метанавчанням» та «індуктивними упередженнями» в дослідженнях ШІ та когнітивних науках [13]. У цьому формулюванні є зовнішній цикл (наприклад, еволюція), який оптимізує механізми навчання, щоб мати індуктивні упередження, які дозволяють нам дуже швидко вивчати дуже конкретні речі маючи рішення попередніх пов’язаних проблем. Справді, ця ідея пов’язана з активною сферою досліджень в ШНМ, «навчанням із перенесенням», у якому

зв’язки, попередньо навчені у вирішенні однієї задачі, передаються для прискорення навчання відповідного завдання.

Наприклад, мережа, навчена класифікувати такі об’єкти, як слони та жирафи, може бути використана як відправна точка для мережі, яка розрізняє дерева чи автомобілі. Однак навчання з перенесенням значно відрізняється від вроджених механізмів, які використовуються в мозку. У той час як при трансферному навчанні вся матриця зв’язків ШНМ (або її значна частина) зазвичай використовується як відправна точка, у мозку тварин кількість інформації, що «передається» від покоління до покоління, менша, оскільки вона повинна проходити через вузьке місце – геном. Передача інформації через вузьке місце генома може вибрати для зв’язку та пластичності правила, які є більш загальними, і, отже, з більшою ймовірністю будуть добре узагальнені. Наприклад, зв’язок зорової кори дуже схожий на зв’язок слухової кори (хоча кожна область має свої особливості). Це говорить про те, що гіпотетичний канонічний кортикальний ланцюг забезпечує, можливо, лише незначні варіації, основу для широкого спектру завдань, які виконують ссавці. Нейронаука припускає що можуть існувати потужніші механізми – свого роду узагальнення навчання з перенесенням, – які діють не тільки в рамках однієї сенсорної модальності, як-от зір, але й між сенсорними модальностями і навіть за її межами. Друге спостереження нейронауки випливає з того факту, що геном не кодує уявлення чи поведінку безпосередньо або принципи оптимізації безпосередньо. Геном кодує правила та шаблони підключення, які потім повинні ініціювати поведінку та уявлення. Саме ці правила підключення є метою еволюції. Це передбачає топологію підключення та архітектуру мережі як ціль для оптимізації в штучних системах. Класична ШНМ в основному ігнорувала деталі мережевої архітектури, керуючись, мабуть, теоретичними результатами щодо універсальності повноз'єднаних трирівневих мереж.

Але, звичайно, одним із головних досягнень сучасної ери ШНМ були згорткові нейронні мережі (CNN), які використовують дуже обмежені проводки для використання факту трансляційної симетрії. Натхненням для цієї революційної технології була частково структура зорових рецептивних полів. Це тип вроджених обмежень, які, у тварин виникнли в ході еволюції; може бути ще багато інших, які ще не відкриті. Інші обмеження на підключення та правила навчання інколи накладаються в ШНМ через гіперпараметри, і існує багато досліджень про оптимізацію гіперпараметрів. Наразі, однак, ШНМ використовують лише крихітну частину можливих мережевих архітектур, що підвищує ймовірність того, що ще потрібно відкрити більш потужні архітектури.

Одним із предметів дослідження в нейронауці є конектом. Конектом [19] – це граф відображення всіх нейронних зв’язків мозку. Незважаючи на те, що складно скласти карту людського коннектома, з нашими 90 мільярдами нейронів та 150 трильйонів синапсів, побудовано коннектом простих організмів, а в останніх роботах нанесено на карту весь мозок маленької плодової мухи. Мотивацією для вивчення конектома є те, що це допоможе скерувати майбутні дослідження про те, як мозок вчиться та представляє спогади у своїх зв’язках. Для людей очевидно, особливо в ранньому дитинстві що ми вчимося навичкам і формуємо спогади, формуючи новісинаптичні зв'язки, і наш мозок переробляє себе на основі нашого нового досвіду. Конектом розглядається як граф і аналізується за допомогою інструментів з графової теорії, мережевої науки та комп'ютерного моделювання. Ця робота також спрямована на вивчення мережевих графів, що дозволяють кодувати навички та знання для штучного агента в середовищі. Послаблючи значення параметрів ваг, агент заохочується натомість розвивати постійно зростаючі мережі, які можуть кодувати набуті навички на основі його взаємодії з навколишнім

середовищем. Як і коннектом простих організмів, мережі, виявлені цим підходом, досить малі, тобто їх можна проаналізувати.

1.4 Навчання з підкріпленням

Для перевірки якості нейромереж цілком природньо в нашому випадку вибрати задачі зі сфери навчання з підкріпленням, тому що вони наглядно зможуть показати процес еволюції мережі на основі генетичного алгоритму. Тож розглянемо основні моменти цієї сфери машиного навчання.

Є багато невирішених проблем, які комп’ютери могли б вирішити, якби існувало відповідне програмне забезпечення. Системи управління польотом літаків, автоматизовані виробничі системи та складні системи авіоніки – це все приклади складних, нелінійні проблем управління. Багато з цих проблем в даний час нерозв'язні не тому, що поточні комп'ютери занадто повільні або мають занадто мало пам'яті, а просто тому, що занадто складно визначити, що повинна робити програма. Якби комп’ютер міг навчитися вирішувати проблеми методом спроб і помилок, це мало б велике практичне значення. Навчання з підкріпленням - це підхід до машинного інтелекту, який поєднує дві дисципліни для успішного вирішення проблем, які жодна з дисциплін не може вирішити окремо. Динамічне програмування – це область математики, яка традиційно застосовується для вирішення проблем оптимізації та управління, однак традиційне динамічне програмування обмежується за розміром та складністю проблем. Однак для навчання з вчителем потрібні зразки пар вхід-вихід з функції, яку слід вивчити. На жаль, є багато ситуацій, коли ми не знаємо правильних відповідей, що вимагає контрольоване навчання. Наприклад, у системі управління польотом питанням буде набір усіх показань датчиків протягом певного часу, а відповідь полягатиме в тому, як

повинні рухатися поверхні управління польотом протягом наступної мілісекунди. Прості нейронні мережі не можуть навчитися літати на літаку, якщо не існує набору відомих відповідей, тому якщо ми не знаємо, як спочатку створити контролер, просте контрольоване навчання не допоможе. З цих причин останнім часом великий інтерес викликає інший підхід, відомий як RL. Навчання з підкріпленням не є різновидом нейронних мереж і не є альтернативою нейронним мережам. Швидше, це ортогональний підхід, який розглядає інше, більш складне питання. RL приваблює багатьох дослідників через його загальність. У RL комп'ютеру просто дається мета яку треба досягти. Потім комп'ютер дізнається, як досягти цієї мети шляхом взаємодії методом проб і помилок із навколишнім середовищем. Таким чином, багато дослідників вивчають цю форму машинного інтелекту і схвильовані можливістю вирішення проблем, які раніше не були вирішуваними. Розгляньмо задачу їзди на велосипеді. Під час першого випробування RL system починає їздити на велосипеді і виконує ряд дій, в результаті яких велосипед нахиляється на 45 градусів праворуч. На даний момент можливі дві дії: повернути ручки вліво або повернути вправо. Система RL повертає ручки вліво і негайно падає на землю, отримуючи таким чином негативне підкріплення. Система RL щойно навчилася не повертати ручки вліво при нахилі на 45 градусів вправо. У наступному випробуванні система RL виконує ряд дій, в результаті яких велосипед знову нахиляється на 45 градусів вправо. Система RL знає, що не слід повертати ручки вліво, тому вона виконує єдину можливу дію: поверніть праворуч. Він відразу ж падає на землю, знову отримуючи сильне негативне підкріплення. На даний момент система RL не тільки дізналася, що поворот ручки вправо або вліво при нахилі на 45 градусів вправо погано, але і те, що «стан» того, що титулується на 45 градусів вправо, є поганим. Знову ж таки, система RL починає чергову пробу і виконує ряд дій, в результаті яких велосипед

нахиляється на 40 градусів вправо. Можливі дві дії: поверніть праворуч або поверніть ліворуч. Система RL повертає ручки ліворуч, в результаті чого велосипед нахиляється на 45 градусів вправо, і в кінцевому підсумку призводить до сильного негативного посилення. Тож система RL щойно навчилася не повертати ручки ручки вліво, якщо руль на 40 градусів направо. Виконуючи достатню кількість цих взаємодій методом спроб і помилок із навколишнім середовищем, система RL зрештою навчиться, як запобігти коли-небудь падіння велосипеда.

1.4.1 Складові частини проблеми навчання з підкріпленням

У стандартній моделі навчання з підкріплення агент взаємодіє із своїм оточенням. Ця взаємодія набуває форми агента, що відчуває навколишнє середовище, і на основі цього сенсорного входу вибирає дію для здійснення в навколишньому середовищі. Дія якимось чином змінює середовище, і ця зміна повідомляється агенту через скалярний підсилювальний сигнал. Існує три фундаментальні частини RL проблеми: навколишнє середовище, функція підкріплення та ціннісна функція. Кожна RL система вивчає відображення ситуацій у дії шляхом взаємодії методом спроб і помилок з динамічним середовищем. Це середовище повинно бути принаймні частково спостережуваним системою RL, а спостереження можуть мати форму показань датчиків, символічних описів. Якщо система RL може повністю відстежувати всю інформацію в середовищі, яка може вплинути на вибір дії для виконання, тоді система RL вибирає дії на основі справжніх «станів» середовища. Цей ідеальний випадок є найкращою можливою основою для RL і, насправді, є необхідною умовою для більшої частини пов'язаної з цією проблемою теорії.

На рисунку 1.11 зображен типовий сценарій RL проблеми з агентом, який вибирає деяку дію, змінює своє положення та отримує оцінку своєї дії

від середовища. Як зазначалося раніше, системи RL вивчають відображення ситуацій у дії шляхом взаємодії методом проб і помилок з динамічним середовищем.

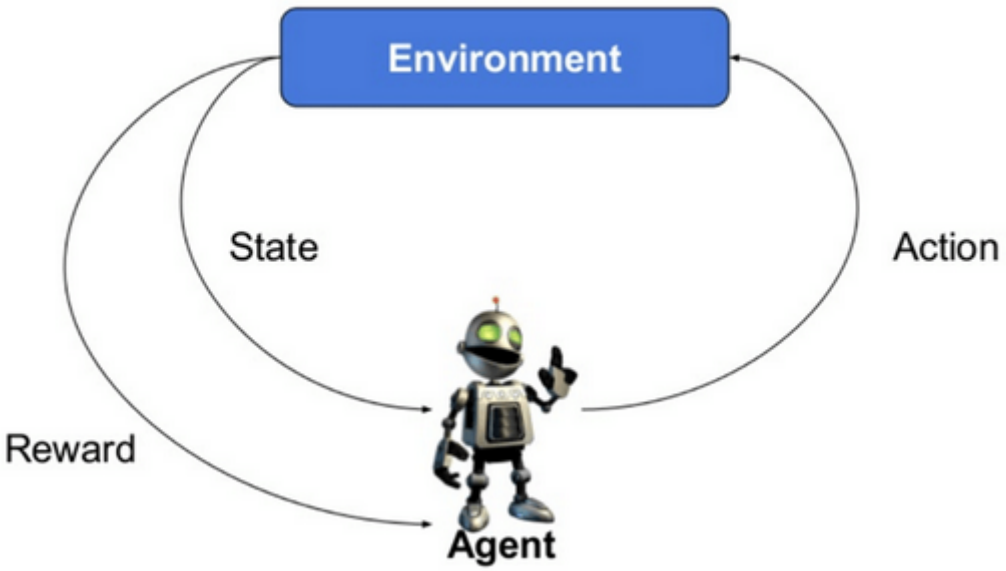

Рисунок 1.11 – Типовий сценарій RL проблеми

Мета системи RL визначається за допомогою концепції функції підсилення, яка є точною функцією майбутніх підкріплень, які агент прагне максимізувати. Іншими словами, існує перетворення від пар стану / дії до підкріплення; після виконання дії в даному стані агент RL отримає деяке підкріплення (винагороду) у вигляді скалярного значення. Агент RL вчиться виконувати дії, які дозволять максимізувати суму підкріплень, отриманих при запуску з деякого початкового стану та переходу до кінцевого стану. Завданням конструктора системи RL є визначення функції підсилення, яка належним чином визначає цілі агента RL.

1.5 Постановка задачі

Ґрунтуючись на вищевикладеному, формально завдання дослідження зводиться до створення штучної нейронної мережі, реалізації еволюційного алгоритму для рішення задачі з підкріпленням та задачі з вучителем. Прикладом застосування для цієї системи є зажача безперервного контролю теліжки, ведення автомобіля у грі та задача класифікації цифр. Задачу з вучителем буде зведено до задачі з підкріпленням. Потрібно дослідити обрану штучну нейрону мережу для знаходження кількості вхідних нейронів, створити системи для вишевказаних задач, реалізувати генетичний алгоритм у парі зі штучною нейронною мережею, «навчити» цю мережу на основі навчання з підкріпленням та проходу багатьох поколінь, протестувати отриману модель в різних умовах, проаналізувати отримані результати та порівняти з існуючими реалізаціями.

Крім того, важливим елементом алгоритму є можливість бачити ускладнення нейронної мережі, буде дана спроба інтепретування обраних мережею комбінацій ознак для задачі безперевного контролю.

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Алгоритм WANN

Створення мережевих архітектур, які кодують рішення завданнь машиного навчання, є принципово іншою проблемою, ніж проблема, яку вирішує пошук нейронної архітектури (NAS). Метою NAS-методів є створення архітектурних конструкцій, які, навчившись, перевершують ті, що розроблені людьми. Ніколи не стверджується, що рішення є природженим для структури мережі. Мережі, створені NAS, надзвичайно «піддаються навчанню», але ніхто не припускає, що ці мережі вирішать завдання, без навчання ваг. Ваги – це рішення; знайдені архітектури просто краща підкладка для розміщення ваг. Для створення архітектур, які самі кодують рішення, важливість ваг повинна бути зведена до мінімуму. Замість того, щоб оцінювати мережі за їхніми показниками оптимальних значеннень ваг, ми можемо замість цього виміряти їх ефективність, коли їхні значення ваги взяті з випадкового розподілу. Заміна тренування ваг на вибірку ваг з рівномірного розподілу гарантує, що ефективність є продуктом лише мережевої топології. На жаль, через високу розмірність, надійний відбір зразків вагового простору є неможливим для всіх, крім найпростіших мереж. Незважаючи на те, що прокляття розмірності заважає нам ефективно проводити семплювання з рівномырного розподілу окремо для кожного нейрона, завдяки примусовому розподілу значеннь на всіх вагах кількість унікальних значень ваг зменшується до одного. Систематична вибірка одного значення ваги є прямою та ефективною, що дозволяє нам наблизити продуктивність мережі лише за декілька випробувань. Потім це наближення може бути використано для стимулювання пошуку все кращих архітектур. Пошук цих вагових агностичних нейронних мереж можна узагальнити наступним чином:

– спочатку маємо сукупність мінімальних топологій нейронних мереж;

– кожна мережа оцінюється за допомогою кількох евалюацій, з різним значенням загальної ваги;

– мережі класифікуються відповідно до їх продуктивності та складності;

– нова сукупність створюється шляхом варіювання найвищих рейтингових топологій мережі.

Наочно цей алгоритм узагальнен на рисунку 2.1.

Потім алгоритм повторюється, отримуючи топології, поступово зростаючої складності, які ефективніші протягом наступних поколінь

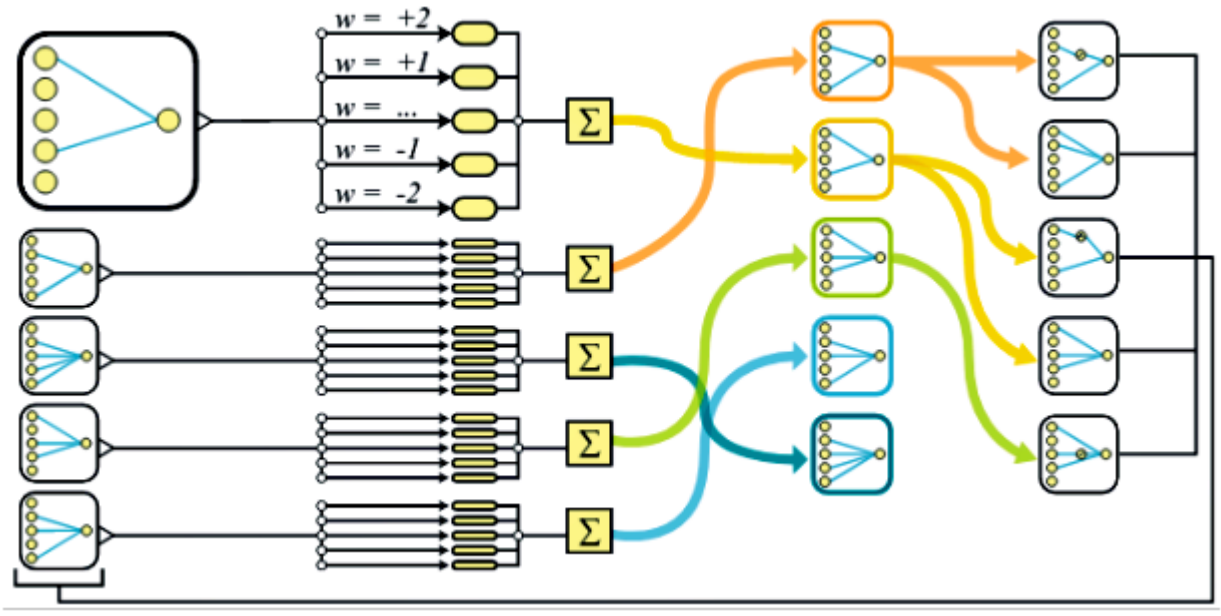

Рисунок 2.1 – Огляд пошуку нейронних мереж з випадковою ініціалізацією

Пошук нейронних мереж з однаковими ваговими коефіцієнтами дозволяє уникнути тренуваня нейронної мережі, досліджуючи простір топологій нейронних мереж, відбираючи одну загальну вагу при кожній оцінці ефективності. Мережі оцінюються декілька раз. При кожному разі призначається значення єдиної загальної ваги та реєструється сукупна винагорода за пробний період. Потім сукупність мереж класифікується відповідно до їх продуктивності та складності. Потім найвищі рейтингові мережі вибирають імовірнісно і варіюють випадковим чином, щоб сформувати нову сукупність, і процес повторюється.

2.1.1 Пошук топології

Оператори, що використовуються для пошуку топологій нейронних мереж, натхненні добре усталеним алгоритмом нейроеволюції NEAT. Хоча в NEAT одночасно оптимізуються значення топології та ваги, ми ігноруємо ваги і застосовуємо лише оператори топологічного пошуку. Початкова сукупність складається з малозв'язаних мереж, мереж без прихованих вузлів і лише частки можливих зв'язків між входом і виходом. Нові мережі створюються шляхом модифікації існуючих мереж за допомогою одного з трьох операторів: вставки вузла, додавання з'єднання або зміни активації (Рисунок 2.2). Щоб вставити вузол, розділиляємо існуюче з'єднання на два з'єднання, які проходять через цей новий прихований вузол. Функція активації цього нового вузла призначається випадковим чином. Нові з'єднання додаються між раніше не підключеними вузлами, з урахуванням властивості прямої передачі мережі. Коли функції активації прихованих вузлів змінюються, вони призначаються випадково. Сукпність функцій активації включають як загальні (наприклад, лінійні, сигмоподібні, ReLU), так і більш екзотичні (гауссові, синусоїдні, ступінчасті), кодуючи різноманітні взаємозв'язки між входами та виходами.

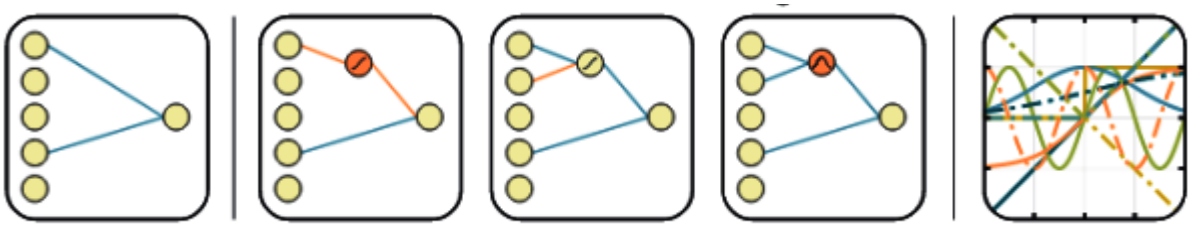

Рисунок 2.2 – Оператори для пошуку простору мережевих топологій

Зліва зображено мінімальну топологію мережі, вхід та вихід якої підключені лише частково. Посередені зображені способи змінення мережі. Вставити вузол означає, що новий вузол вставляється шляхом розділення існуючого з'єднання. При додаванні підключення нове підключення додається шляхом підключення двох раніше не підключених вузлів. Зміна активації означає, що функція активації прихованого вузла перепризначається. Зправа зображені можливі функції активації (лінійна, крокова, косинусова, гауссова, тангова, сигмоїдна, абсолютна величина, інвертна (тобто від’ємна лінійна), ReLU тощо).

2.1.2 Продуктивність та складність

Топології мережі оцінюються за допомогою кількох спільних значень ваги. При кожному разі нове значення ваги призначається всім підключенням, а мережа тестується за завданням. У цих експериментах використовувалася фіксована серія значень ваги ([-2, -1, -0,5, + 0,5, + 1, + 2]), щоб зменшити дисперсію між оцінками. Далі розраховується середня продуктивність мережевої топології шляхом усереднення її сукупної винагороди за всі евалюації з використанням цих різних значень ваги. Пам’ятаючи про алгоритмічну теорію інформації, нас цікавить не просто неймовірно великі нейронні мережі, а мережі, які можна описати якомога

простіше. Враховуючи дві різні мережі з однаковою продуктивністю, віддається перевага більш простій мережі. Сформулювавши пошук як багатоцільову задачу оптимізації, беремо до уваги розмір мережі, а також її ефективність при її ранжируванні серед сукупності. Застосовується методика вартості підключення з [15], показану для створення мереж. які є більш простими, модульними та зручними для розвитку. Топології мереж оцінюються на основі трьох критеріїв: середньої продуктивності за всіма значеннями ваги, максимальної продуктивності одного найкращого значення ваги та кількості підключень у мережі. Замість того, щоб намагатися збалансувати ці критерії з евристичною функцією винагороди за кожне нове завдання, класифікуймо рішення на основі відносин домінування. Для ранжування мереж таким чином потрібно, щоб будь-яке збільшення складності супроводжувалось підвищенням продуктивності. Ця інтуітивна ідея була запозичена з алгоритмічної теорії інформації, але є ще більш просте пояснення. Якщо згадати інформаційний критерій Акаіке [19], то ми побачимо, що найкраща модель із набору моделей визначається простою евристичною формулою, яка пов’язує кількість параметрів та продуктивність моделі. Тобто тут змінюється кількість параметрів на мінімальну довжину нейронної мережі. Заохочуючи мінімальні та модульні мережі, це обмеження може зробити важкі структурні зміни, які можуть вимагати кількох доповнень перед тим як вони підвищать свою ефективність порівняно за більш простим екземпляром. Щоб послабити це обмеження класифікуємо за складністю лише з імовірністю: у 80% випадків мережі класифікуються за середньою продуктивністю та кількістю з'єднань, в інших 20% рейтинг визначається за середньою продуктивністю та максимальною продуктивністю.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Огляд бібліотек

Мовою програмування алгоритму обрано Python 3.5. Python – це інтерпретована, об’єктно-орієнтована мова програмування високого рівня з динамічною семантикою, розроблена Гвідо ван Россумом. Спочатку він був випущений у 1991 році. Розроблений так, щоб бути легким і веселим, назва «Python» є нагадуванням британської комедійної групи Monty Python. Python має репутацію мови, зручної для початківців, замінюючи Java як найбільш широко використовувану вступну мову, оскільки вона справляється зі складністю для користувача, дозволяючи новачкам зосередитися на повному сприйнятті концепцій програмування, а не на дрібних деталях. Python використовується для розробки програмного забезпечення, математики та створення системних сценаріїв, а також популярний для швидкої розробки додатків, а також як мова скриптів для зв'язування існуючих компонентів завдяки своїм високорівневим вбудованим структурам даних. Витрати на обслуговування програм зменшуються з Python завдяки легко засвоюваному синтаксису та акценту на читабельності. Крім того, підтримка Python модулів і пакетів полегшує створення модульних програм та повторного використання коду. Python – мова спільноти з відкритим вихідним кодом, тому численні незалежні програмісти постійно створюють для нього бібліотеки та функціональні можливості.

Для побудови нейромережі ми будемо використовувати нативні структури даних Python: такі як списки та словарі та вектори і матриці з бібліотеки NumPy, адже нам не потрібно навчати нейромережі. NumPy – пакет для обробки масивів загального призначення. Він надає інтерфейс для створення високопродуктивних багатовимірних масивів та інструменти

для роботи з цими масивами. Це основний пакет для наукових обчислень з Python. Він містить різні функції, включаючи такі важливі як: інтерфейс для створення N-вимірного масиву, корисні функції лінійної алгебри та генератори випадкових чисел. Окрім очевидного наукового використання, NumPy також можна використовувати як ефективний багатовимірний контейнер загальних даних.

Для того, щоб ефективно розпаралелити пошук та валідацію нейромереж будемо використовувати реалізацію інтерфейсу MPI (message passing interface) для Python. MPI (message passing interface) – стандартизована та переносна система передачі повідомлень (бібліотека функцій). Стандарт визначає синтаксис і семантику бібліотечних функцій, що використовуються при написанні програм, незалежно віж мови програмування. Іншими словами, MPI – це програмний інструментарій для забезпечення зв'язку між гілками паралельного додатку. Стандарт MPI-1 був прийнятий групою близько 80 дослідників із 40 організацій (як академічних, так і промислових) під час серії зустрічей у 1992-93 роках. В обговоренні стандарту могли взяти участь усі охочі. У червні 1994 р. була реалізована версія 1.0 Існує кілька протестованих ефективних реалізацій MPI, у тому числі для вільного використання (наприклад, mpi4python). Парадигма передачі повідомлень широко використовується на паралельних комп'ютерах, в основному на масштабованих з розподіленою пам'яттю та мережі робочих станцій. Основна мета MPI – досягти переносимість програмного забезпечення для різних паралельних комп'ютерів. У тому числі і для комп'ютерів з пам'яттю, що розділяється, якщо для них реалізований MPI. Розроблений код також може виконуватися на мережі робочих станцій, або як безліч процесів на одній станції. Це, зокрема, дає змогу налагоджувати програми на персональному комп'ютері. Інший тип сумісності, пропонований MPI – це можливість прозорої роботи на різнорідних системах (безліч процесорів з різною архітектурою). У цьому

випадку MPI автоматично проводитиме конвертацію даних і використовуватиме коректний протокол взаємодії (проте треба використовувати реалізацію MPI для різнорідних архітектур). Інша важлива перевага MPI – висока ефективність, навіть у порівнянні з "рідними" системами. Також, від користувача приховано, як саме виконується та чи інша операція, важливим є лише те, що вона робить. Ефективність MPI досягається також за рахунок невиконання непотрібної роботи з надсилання великої кількості зайвої інформації з кожним повідомленням або кодування-декодування заголовків повідомлень. MPI був розроблений так, щоб підтримувати одночасне виконання обчислень та комунікацій, щоб використовувати наявність співпроцесорів. Це досягається використанням неблокованих комунікаційних викликів, які поділяють ініціацію комунікації та її завершення. Для підвищення швидкості в MPI використовуються прийоми, про які прикладні програмісти часто просто не замислюються. Наприклад, вбудована буферизація дозволяє уникнути затримок під час відправлення даних – управління в передаючу гілка повертається негайно, навіть якщо гілка-одержувач ще не підготувалася до прийому. MPI використовує багатопоточність (multi-threading), виносячи більшу частину своєї роботи в потоки (threads) з низьким пріоритетом. Буферизація та багатопоточність зводять до мінімуму негативний вплив неминучих простоїв при пересилання на продуктивність прикладної програми. На передачу даних типу «один-усім» витрачається час, пропорційний не числу гілок, що беруть участь, а логарифму цього числа.

Для моделювання агентів (теліжки чи двоного ходака) використовуватимо бібліотеку OpenAI Gym. OpenAI Gym – це набір інструментів для розробки та порівняння алгоритмів навчання з підкріпленням. Він підтримує навчання будб-яких агентів , від ходьби до ігор, таких як понг або пінбол. Gym – це інтерфейс із відкритим кодом для

RL завдань. Він забезпечує середовище, і розробник може реалізувати будь-які алгоритми навчання з підкріпленням. Розробники можуть написати агента, використовуючи наявну бібліотеку чисельних обчислень, таку як NumPy.

Для навчання агента, який керує машиною, будемо використовувати ознаки високого порядку, які будуть отримані із заздалегідь натревнованого варіаційного автокодеру. Будемо виконувати цей код на фреймворці Tensorflow. TensorFlow дозволяє створювати графи потоків даних і структури, щоб визначити, як дані переміщуються по графу, беручи вхідні дані як багатовимірний масив, який називається Tensor. Це дозволяє побудувати блок-схему операцій, які можна виконати на цих входах, яка йде на одному кінці і надходить на іншому кінці як вихід. Архітектура Tensorflow складається з трьох частин: попередньої обробки даних, будування моделі,тренування і оцінка моделі. Він називається Tensorflow, оскільки приймає вхідні дані як багатовимірний масив, також відомий як тензори. Ви можете побудувати своєрідну блок-схему операцій, яку потрібно виконати на цьому вході. Вхідний кінець входить з одного кінця, а потім проходить через цю систему множинних операцій і виходить на інший кінець як вихід. Ось чому він називається TensorFlow, оскільки тензор проходить через список операцій, а потім виходить з іншого боку.

3.2 Експериментальні результати

3.2.1 Задача безперервного контролю

WANN оцінюються за трьома завданнями безперервного контролю. Перший, CartPoleSwingUp, – це класична проблема управління, коли, враховуючи систему полюсів візка, стовп потрібно перевести з положення спокою у вертикальне положення, а потім збалансувати, не виходячи за

межі колії. Це більш складне завдання, ніж CartPole [11], де стовп одразу становиться у вертикальне положення і його потрібно збалансувати. На відміну від більш простого завдання, воно не може бути вирішене за допомогою лінійного контролера [11]. Винагорода на кожному кроці залежить від відстані візка до полу та кута стовпа. Це середовище засноване на середовищі, описаному в роботах. Друге завдання, BipedalWalker-v2 [12], полягає у проведенні двоногих агентів через випадково сформований терен. Нагороди присуджуються за пройдену відстань. Кожна нога контролюється тазостегновим та колінним суглобами у відповідь на 24 входи, включаючи датчики LIDAR, які виявляють місцевість та пропріоцептивну інформацію, таку як швидкість суглоба агента. Третє завдання, CarRacing-v0 [13], – це автомобіль, що їде зверху вниз із середовища пікселів. Автомобіль, керований трьома безперервними командами (газ, кермо, гальмо), має завдання відвідати якомога більше плиток випадково сформованої доріжки протягом обмеженого часу. Дотримуючись підходу, описаного в [14], можемо передати завдання інтерпретації пікселів попередньо навченому варіаційному автокодеру (VAE), який стискає подання пікселів до 16 прихованих розмірів. Ці розміри даються як вхідні дані до мережі. Використання вивчених функцій перевіряє здатність WANN засвоювати абстракційні асоціації, а не кодувати явні геометричні взаємозв'язки між входами. Мережі, знайдені в літературі [15], порівнюються з найкращими мережами, які знайдені за допомогою WANN для кожного завдання. Порівнюється середня ефективність за 100 випробувань за 4 умов:

– випадкові ваги: індивідуальні ваги, взяті з U (-2,2);

– випадкові спільні ваги: одне загальне число, взяте з U (-2,2);

– налаштована загальна вага: найбільш ефективне значення загальної ваги в діапазоні U(-2,2);

– налаштовані ваги: індивідуальні ваги, налаштовані за допомогою REINFORCE на основі популяції.

Ми порівнюємо кумулятивну винагороду (в середньому 100 випадкових випробувань) найкращих мережевих архітектур зі стандартними політиками повнозв’язних мереж. Власне упередження топології мережі можна спостерігати, вимірюючи її ефективність, використовуючи загальну вагу, відібрану з рівномірного розподілу. Настроюючи цей загальний параметр ваги, ми можемо виміряти його максимальну продуктивність. Для полегшення порівняння з базовими архітектурами також проводимо експерименти, де мережі мають унікальні параметри ваги та налаштовані. На відміну від звичайних мереж фіксованої топології, що використовуються як базові моделі, які виробляють корисну поведінку лише після екстенсивного налаштування ваг, WANN ефективно виконують ці завдання навіть із випадковими спільними вагами. Незважаючи на те, що їхні архітектури кодують рішення із сильним упередженням, WANN не є повністю незалежними від значень ваги вони справді зазнають невдач, коли окремі значення ваги присвоюються випадковим чином. WANN функціонують, кодуючи взаємозв'язки між входами та виходами, і тому, хоча значення величини ваг не є критичним, їх узгодженість, особливо узгодженість знака має місто. Додатковою перевагою однієї спільної ваги є те, що налаштовувати цей єдиний параметр стає тривіальною задачею, не вимагає використання методів, що базуються на градієнті. Найефективніше значення спільної ваги дає задовільну, якщо не оптимальну поведінку. Основна поведінка кодується повністю в архітектурі мережі. І хоча WANN можна виконувати без тренувань, ця схильність не заважає їм досягти подібних найсучасніших показників, коли ваги тренуються.

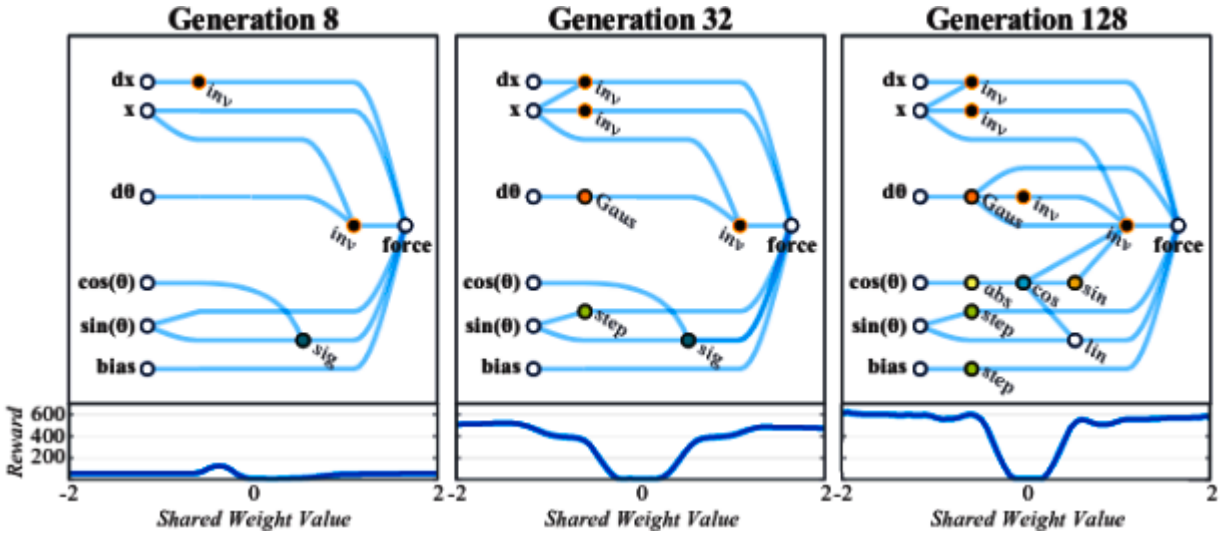

Рисунок 3.1 – Розвиток нейромережевих топологій з плином часу

Покоління 8: Рання мережа, яка погано працює майже з усіма вагами. Покоління 32: Встановлено залежності між положенням візка та швидкістю полюса. Покоління 128: Додано складності, щоб уточнити балансуючу поведінку піднесеного полюса. Оскільки виявлені мережі є досить малими, їх можна намагатись інтерпретувати, ми можемо отримати уявлення про їх функціонування, дивлячись на мережеві схеми.

Вивчаючи розробку WANN, який вирішує CartPoleSwingUpis, також можна продемонструвати, як кодуються відносини в архітектурі. У перших поколіннях простір мереж досліджується по суті випадковим чином. До покоління 32 виникають попередні конструкції, які забезпечують стабільну роботу: три інвертори, застосовані до експозиції, утримують візок від виїзду з колії. Центр доріжки в 0, зліва негативний, правий – позитивний. Застосовується позитивна сила, коли візок знаходиться в негативному положенні, і навпаки, кодується сильний атрактор до центру колії. Взаємодія між регулюванням положення та активацією Гауса відповідає за поведінку повороту, також розроблену поколінням 32. На початку випробування полюс нерухомий: застосовується гауссова активація. Коли

полюс рухається до краю, вузли, з'єднані з входом, які утримують візок у центрі, починають надсилати сигнал протилежної сили. Просування до візка сповільнюється, і зміна прискорення призводить до повороту полюса, і, таким чином, зменшуючи сигнал, який штовхає візок до краю. Це уповільнення спричиняє подальше прискорення полюса, приводячи в рух петлю зворотного зв'язку, що призводить до швидкого розсіювання сигналу. Отримане закріплення візка назад до центру призводить до повороту жердини вгору. Коли полюс падає і осідає, повторюється однакова поведінка повороту, і контролер отримує винагороду, коли полюс стоїть вертикально. По мірі того, як процес пошуку триває, деякі з цих контролерів затримуються у вертикальному положенні довше, ніж інші, а з покоління 128 – тривалість в усталеному виді. Хоча цей більш складний механізм балансування менш надійний при змінних вагах, ніж поведінка коливання та центрування, більш надійні способи гарантують, що система відновлюється і намагається знову, поки не буде знайдено збалансований стан. Примітно, що, оскільки ці мережі кодують взаємозв'язки і лише на основі напруги між системами, налаштованими одна на одну, їхня поведінка все ще залишається незмінною навіть із широким діапазоном загальних значень ваг. На Рисунку 3.2 зображен типовий приклад змінення різних параметрів фітнес-функції при проходженні еволюційного алгоритму.

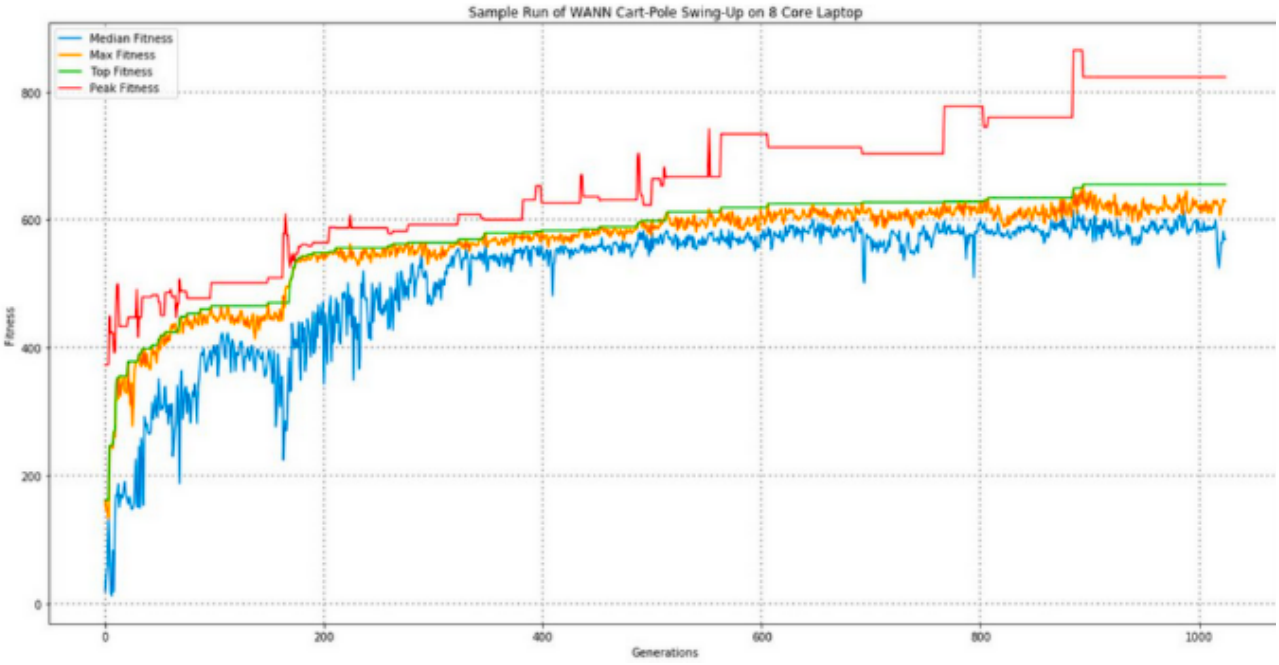

Рисунок 3.2 – Покажчики ефективності агентів

Контролери WANN для BipedalWalker-v2 та CarRacing-v0 також відрізняються своєю простотою та модульністю. Двоногий контролер використовує лише 17 з 25 можливих входів, ігноруючи багато датчиків LIDAR та швидкості руху колін. Архітектура WANN не тільки вирішує завдання без підготовки індивідуальних ваг, але використовує лише 210 з'єднань, на порядок менше ніж загальновживані топології (2804 з'єднання, що використовуються в базовій SOTA [15]). Архітектура, яка кодує стабільну поведінку водіння в автомобілі racer також вражає своєю простотою (Рисунок 3.3). Для кодування грамотної поведінки водіння потрібна лише малопов'язана двошарова мережа та одне значення ваги. Базовий рівень SOTA [15] також давав приховані стани попередньо навченої моделі RNN, на додаток до представлення VAE своєму контролеру; наш контролер працює лише на прихованому просторі VAE. Тим не менше, йому вдалося розробити контролер, який досягає порівнянних результатів. Мережі, показані на рисунку були обрані згідно

критеріїв продуктивності і читабельності. У багатьох випадках додається велика складність лише за мінімальний виграш у продуктивності, у цих випадках вважаємо за краще демонструвати більш елегантні мережі. Мережі фінальних чемпіонів показані на Рисунку 3.3.

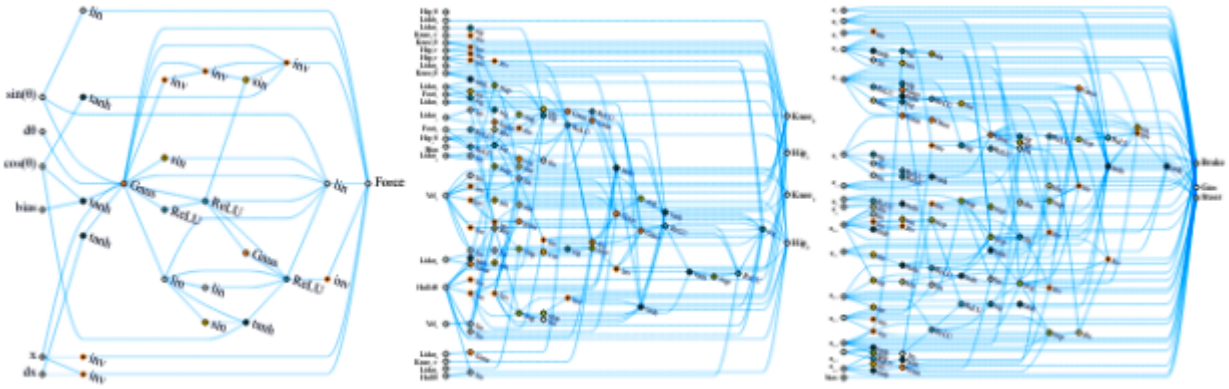

Рисунок 3.3 – Мережі чемпіонів для безперервних завдань управління

Зліва направо (Кількість з'єднань): Swing up (52), Biped (210), Car Racing (245) - це високопродуктивні, але простіші мережі, вибрані для наочності.

3.2.2 Результати для задачі навчання з вучителем

Перспективні результати WANN для завдань з підкріпленням змушують розглянути, де ще можна застосовувати підхід WANN. WANN кодують взаємозв'язки між входами та добре підходять для завдань RL: входи низького розміру в поєднанні з внутрішніми станами та взаємодія середовища з низьким рівнем відкриття реактивних та адаптивних контролерів. Однак класифікація це проблема, де, на відміну від RL, проектування архітектур вже давно в центрі уваги. Як доказ концепції, досліджується, як WANN працюють на наборі даних MNIST [19], завданні

класифікації зображень, де вчені займалися пошуком архітектур десятиліттями. Навіть у цій багатовимірній задачі класифікації WANN працюють надзвичайно добре. Обмежені до одного значення ваги, WANN здатні класифікувати цифри MNIST, а також отримати співставні результати з одношаровою нейроною мережею з тисячами ваг, які треновані градієнтним спуском. Створені архітектури все ще підтримують гнучкість, дозволяючи тренуватися на прикладах градієнтим спуском, дозволяючи подальше підвищення точності.

Набір MNIST – це велика колекція рукописних цифр. Це дуже популярний набір даних у галузі обробки зображень. Він часто використовують для тестування алгоритмів машинного навчання. MNIST – це скорочення від модифікованої бази даних Національного інституту стандартів та технологій. MNIST містить колекцію з 70000 зображень 28 x 28 рукописних цифр від 0 до 9.

Версія MNIST, використана в цьому експерименті, має дескриптовану версію, цифри зменшені з 28x28 до 16x16 пікселей, і виправлеі за допомогою бібліотеки OpenCV [20]. Найкраща вага мережі MNIST була обрана для мережі з найвищою точністю на навчальному наборі. Щоб укластися в наш існуючий підхід, класифікація MNIST перероблена як проблема навчання з підкріпленням. Кожна вибірка в MNIST зменшується до зображення 16x16 та інтенсивність пікселів нормалізується від 0 до 1. WANN створюються із входом для кожного з 256 пікселів та одним виходом для кожної з 10 цифр. На кожну оцінювальну мережу подається 1000 зразків, довільно відібраних з навчального набору, і отримують винагороду на основі кросс-ентропії softmax. Мережі перевіряються різноманітністю спільних значень ваги, максимізуючи продуктивність за всіма вагами, мінімізуючи кількість з'єднань. На Рисунку 3.4 зображені результати WANN, створені з різними значеннями ваг.

WANN, які діють як ансамбль, працюють набагато краще, ніж коли ваги відбираються навмання.

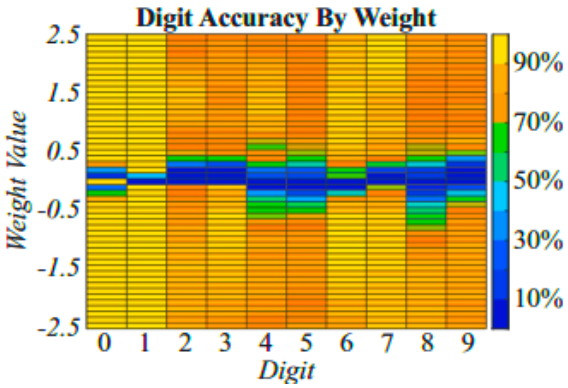

Рисунок 3.4 – Класифікаційна точність на MNIST

У таблиці 3.1 приведені точності WANN алгоритму з чотрима модифікаціями. Також можна порівняти ці результати з результатами класичних усталених алгоритмів в таблиці 3.2

Таблиця 3.1 – Таблиця результатів WANN для MNIST

WANN	**Test Accuracy**
Random Weight	82.0% ± 18.7%
Ensemble Weights	91.6%
Tuned Weight	91.9%
Trained Weights	94.2%

Таблиця 3.2 – Таблиця результатів деяких традиційних алгоритмів для MNIST

ANN	**Test Accuracy**
Linear Regression	91.6%
Two-layer CNN	99.3%

Те, що WANN можна створити за допомогою різних різних мереж, має інтригуючі можливості для створення ансамблів. Прямо вперед пронестися по діапазону ваг, щоб знайти значення, яке найкраще відповідає навчальному набору, але структура WANN пропонує ще одну інтригуючу можливість. Для кожного значення ваги прогноз WANN різний. На MNIST це можна побачити в різній точності на кожній цифрі на Рисуноку 3.4. Кожне значення ваги мережі можна розглядати як окремий класифікатор, створюючи можливість використання одного WANN з кількома значеннями ваги як самостійного ансамблю. У найпростішому підході ансамблю, колекція мереж створюється шляхом створення WANN з діапазон значень ваги. Кожна з цих мереж отримує один голос, а ансамбль класифікує зразки відповідно до категорії, яка набрала найбільше голосів. Цей підхід дає прогнози набагато точніші, ніж випадково вибрані значення ваги, і лише трохи гірші, ніж найкраща можлива вага. Те, що результат цього наївного ансамблю є успішним, надихає для експериментів із більш досконалими техніками ансамблювання під час пошуку архітектур. На Рисунку 3.5 ми бачимо, що не всі нейрони та з'єднання використовуються для прогнозування кожної цифри.

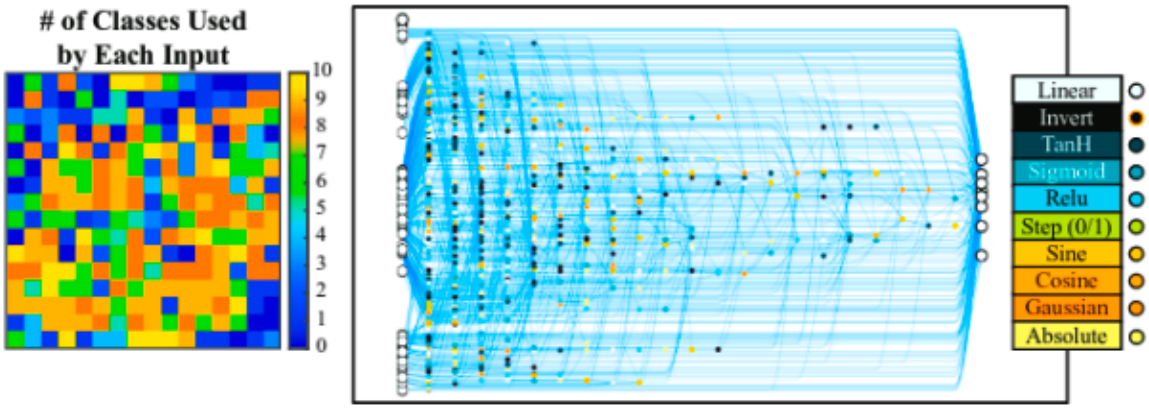

Рисунок 3.5 – Мережа для класифікації MNIST (1849 з'єднань)

Починаючи з вихідного з'єднання для певної цифри, ми можемо простежити підмережу, а також визначити, яка частина вхідного зображення використовується для класифікації кожної цифри. Індивідуальні ваги можна додатково налаштувати на найкращу спільну вагу. Однак пошук архітектури мережі з використанням випадкових ваг для кожного з’єднання не презвів до хороших результатів в цих екперементах. Замість того, щоб встановлювати всі ваги різними випадковими значеннями, ми мали встановити всі ваги тим самим випадковим значенням. Тож це скоріше фіча, а не баг, що цей результат і метод в науковій спільноті розглядають через MDL та AIT підхід.

Також можна експериментувати, додаючи гауссовий шум до значень ваги, щоб кожна вага буде різною, але варіювалася, відхиляючись не більш ніж на деяке значення від середньо встановленого середнього значення при кожному випуску. Це розвинення ідеї полягає в тому, що додавання шуму зробить кінцеві топології ще більш стійкими до змін у ваговій величині, але для цього доведяться робити більш тестів, щоб зібрати реперезнтативну статистику, як ця мережа виконує певне завдання. Ця ідея немає чітких переваг, тож було вирішено дотримуватися підходу якомога простішим у концептуальному плані.

Іншою ідеєю для розвинення алгоритму є застосування методу зворотного розповсюдження для точної настройки вагів WANN. Існують автоматичні пакети, такі як JAX [37], які можуть брати градієнт для будь-яких архітектур для точної настройки індивідуальних ваг WANN. В доповнення можна припустити, що різні активації, запропоновані алгоритмом пошуку WANN, могли створити поверхні оптимізаційної функції, які складніші для проходження методів на основі градієнта порівняно зі стандартними архітектурами глибоких мереж на основі ReLU, тож ця ідея залишена неімплементованою.

Ще один простір для подальших експериментів полягає у виборі активаційних функцій. Без конкретних значень ваги, на які можна спиратися, натомість метод покладається на кодування взаємозв’язків між входами в мережу. Це могло бути зроблено за допомогою ReLU або сигмоїдів, але включаючи відносини, такі як симетрія та повторення, це дозволяє отримати більш компактні мережі. Експерементів щодо зкорочення простору вибору активацій не проводилось, але інтуїція полягає в тому, що різноманітність активацій є ключовою. Що стосується біологічних аналогій, не стверджується, що косинус є точною моделлю того, як нейрони активуються у мозку, але також не є вірогідним, що мережа прямого передавання, яка складається з сигмоїдних нейронів буде більш біологічно вірогідною.

ВИСНОВКИ

У цій роботі введен метод пошуку простих архітектур нейромереж із сильними індуктивними упередженнями. Оскільки мережі оптимізовані для ефективної роботи, використовуючи загальну вагу для ряду значень, цей єдиний параметр може бути легко налаштований для підвищення продуктивності.

Здатність швидко точно налаштовувати ваги корисна для нечисленних завдань і може знайти застосування у постійному навчанні, коли агенти постійно набувають, доопрацьовують та передають навички протягом усього свого життя, як у тварин. Для розробки єдиного WANN, здатного кодувати безліч різних корисних завдань у своєму середовищі, можна розглянути можливість створення WANN із сильним внутрішнім упередженням, і оптимізувати отриману архітектуру, щоб добре працювати у новому середовищі. Подібний WANN може кодувати, через сигнал винагороди за цікавість, безліч навичок, які згодом легко можна відрегулювати під конкретне подальше завдання в його середовищі. Хоча цей підхід вивчає мережеві архітектури, що збільшуються, додаючи з'єднання, підходи до обрізки мережі знаходять нові архітектури шляхом їх вилучення. Крім того, можна навчити поглиблену мережу, здатну виконувати додаткові завдання без вивчення ваг. Є дослідження, які намагаються у великій нейромережі знайти підмережу – маску, де підмережа, обрізана за допомогою цієї маски, виконує розпізнавання зображень, навіть із випадково ініціалізованими вагами – цікаво, що їх підходи досягають подібного діапазону продуктивності на MNIST порівняно з цим. Поки цей метод пошуку базується на еволюції, але цей підхід можна розширити, включивши останні ідеї, які формують пошук архітектури за допомогою оптимізації ваг градієнтним спуском, щоб зробити пошук більш ефективним.

Успіх глибокого навчання пояснюється нашою здатністю тренувати ваги великих нейронних мереж, які складаються з добре продуманих будівельних блоків на великих масивах даних, використовуючи градієнтний спуск. Хоча в цій області було досягнуто значного прогресу, є й обмеження, оскільки ми обмежуємося простором архітектур, який градієнтний спуск здатний тренувати. Наприклад, ефективні навчальні моделі, які покладаються на дискретні компоненти або використовують адаптивні обчислювальні механізми з градієнтними методами, залишаються складним напрямом досліджень. Цей підхід сприятиме відкриттю нових архітектур, які не лише мають індуктивні упередження для практичних доменів, але також можуть бути навчені алгоритмами, які можуть не вимагати градієнтних обчислень. Не дивно, що мережі, знайдені в цій роботі, не відповідають продуктивності згорткової нейронної мережі. Протягом десятиліть архітектури CNN вдосконалювались вченими-інженерами. Колись згорткові шари були новими будівельними блоками, що мають сильну упередженість до завдань комп’ютерного зору, відкриття та застосування у цій області сприяли неймовірному прогресу в глибокому навчанні. Обчислювальні ресурси, доступні науковому співтовариству, значно зросли з часу відкриття згорткових нейронних мереж. Якщо ми присвячуємо такі ресурси автоматизованому відкриттю і сподіваємось досягти більшого, ніж поступового вдосконалення мережевих архітектур, ми також вважаємо, що варто спробувати відкрити нові будівельні блоки, а не лише їх композиції. Нарешті, можна наголосити, що подібні ідеї циркулюють у спільноті нейронаук, є критика вивчення досвіду, яке в даний час впроваджене в штучні нейронні мережі. Тож ця робота мотивована цією критикою для досягнення цілей поєднання вродженої поведінки та навчання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Kenneth O. Stanley,Risto Miikkulainen. Evolving Neural Networks Through Augmenting Topologies. Evolutionary Computation, 2002. 99–127 p.

2. Kenneth O. Stanley,Risto Miikkulainen. Efficient Reinforcement Learning Through Evolving Neural Network Topologies. Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002), 2002.

3. Kenneth O. Stanley; Bobby D. Bryant,Risto Miikkulainen. Evolving Adaptive Neural Networks with and without Adaptive Synapses. Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC-2003).2003.

4. Colin Green.Phased Searching with NEAT: Alternating Between Complexification And Simplification, 2004.

5. T. Aaltonen et al. Measurement of the top quark mass with dilepton events selected using neuroevolution at CDF. Physical Review Letters, 2009.

6. Risi, Sebastian; Stanley, Kenneth O. An Enhanced Hypercube-Based Encoding for Evolving the Placement, Density and Connectivity of Neurons. Artificial Life. 2012. 331–363 p.

7. P. J. AngelineMorphogenic evolutionary computations: Introduction, issues and examples. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel, editors, Evolutionary Programming IV: The Fourth Annual Conference on Evolutionary Programming, pages 387–401. MIT Press, 1995.

8. P. J. Angeline, G. M. Saunders, and J. B. Pollack. An evolutionary algorithm that constructs recurrent neural networks. Neural Networks, IEEE Transactions on, 5(1):54–65, 1994.

9. Holland, John. Adaptation in Natural and Artificial Systems. Cambridge, MA: MITPress. 1992. ISBN 978-0262581110.

10. J. S. Astor and C. Adami. A developmental model for the evolution of artificial neural networks. Artificial Life, 6(3):189–218, 2000.

11. P. J. Bentley and S. Kumar. The ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999), pages 35–43, San Francisco, 1999. Kaufmann.

12. Бодянский Е. В. Искусственные нейронные сети: архитектуры, обучение, применения.Харьков ТЕЛЕТЕХ, 2004. 372 с.

13. A. M. Zador. A critique of pure learning and what artificial neural networks can learn from an-imal brains.Nature communications, 10(1):1–7, 2019.

14. R. Tedrake. Underactuated robotics: Learning, planning, and control for efficient and agile machines:Course notes for mit 6.832.Working draft edition, 3, 2009.

15. A. L. Tierney and C. A. Nelson III. Brain development and the role of experience in the early years.Zeroto three, 30(2):9, 2009.

16. A. Trask, F. Hill, S. E. Reed, J. Rae, C. Dyer, and P. Blunsom. Neural arithmetic logic units. InAdvancesin Neural Information Processing Systems, pages 8035–8044, 2018.

17. A. M. Turing.Intelligent machinery. NPL. Mathematics Division, 1948.

18. D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. In Proceedings of the IEEE Conference onComputer Vision and Pattern Recognition, pages 9446–9454, 2018.

19. M. P. Van Den Heuvel and O. Sporns. Rich-club organization of the human connectome. Journal ofNeuroscience, 31(44):15775–15786, 2011.

20. S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numericalcomputation.Computing in Science and Engineering, 13(2):22, 2011.

21. L. R. Varshney, B. L. Chen, E. Paniagua, D. H. Hall, and D. B. Chklovskii. Structural propertiesof the caenorhabditis elegans neuronal network.PLoS computational biology, 7(2):e1001066, 2011.

22. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin.Attention is all you need. InAdvances in neural information processing systems, pages 5998–6008, 2017.

23. J. G. White, E. Southgate, J. N. Thomson, and S. Brenner. The structure of the nervous system ofthe nematode caenorhabditis elegans.Philos Trans R Soc Lond B Biol Sci, 314(1165):1–340, 1986.

24. R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning.Machine learning, 8(3-4):229–256, 1992.

25. D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervisedprediction. InProceedings of the IEEE Conference on Computer Vision and Pattern RecognitionWorkshops, pages 16–17, 2017.

26. H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean. Efficient neural architecture search via parametersharing. InInternational Conference on Machine Learning, pages 4092–4101, 2018.

27. J. C. F. Pujol and R. Poli. Evolving the topology and the weights of neural networks using a dualrepresentation.Applied Intelligence, 8(1):73–84, 1998.

28. T. Raiko and M. Tornio. Variational bayesian learning of nonlinear hidden state-space models formodel predictive control.Neurocomputing, 2009.

29. E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized evolution for image classifier architecturesearch. InAssociation for the Advancement of Artificial Intelligence (AAAI), 2019.

30. E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin. Large-scale evolution of image classifiers. InProceedings of the 34th International Conference on MachineLearning-Volume 70, pages 2902–2911. JMLR. Org, 2017.

31. D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inferencein deep generative models. InInternational Conference on Machine Learning, pages 1278–1286, 2014.

32. J. Rissanen. Modeling by shortest data description.Automatica, 14(5):465–471, 1978.

33. J. Rissanen.Information and complexity in statistical modeling. Springer Science and Business Media, 2007.

34. A. Roli and L. Melandri. Introduction to reservoir computing methods, 2014.