Міністерство освіти і науки України
Харківський національний університет імені В.Н. Каразіна
Факультет математики і інформатики
Кафедра комп'ютерних наук та інформаційних технологій

Пояснювальна записка

до кваліфікаційної роботи магістра

на тему «Використання алгоритму оптимізації мурашиної колонії для створення наближенної до оптимальної топології нейроних мереж»

Захищено на засіданні ЕК № _____
протокол №_____від_._.2024 р. Оцінка _____/
Голова ЕК
_____

Виконав:
студент 6 курсу, групи МФ62
Спеціальності: 122 Комп'ютерні науки
Мірошниченко Андрій Андрійович
Керівник: викладач Панченко А.С.
Рецензент _____

Харків – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет імені В.Н. Каразіна

Факультет комп’ютерних наук

Кафедра моделювання систем і технологій

Рівень вищої освіти (освітньо-кваліфікаційний рівень)
бакалаврНапрям підготовки 122 Комп’ютерні науки

Спеціальність Комп’ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ _____ “”

грудня 2024 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Мірошниченко Андрій Андрійович

1. Тема роботи Використання алгоритму оптимізації мурашиної колонії для створення наближенної до оптимальної топології нейроних мереж

керівник роботи **викладач Панченко Артем Сергійович,**

затверджені наказом по університету від “” грудня 2024 року № 4101-5/895

2. Строк подання студентом роботи грудня 2024 року

Перелік питань, які потрібно розробити

1. Провести комплексний огляд літератури з оптимізації архітектури нейронних мереж та алгоритму мурашиної колонії (ACO)

2. Аналізувати сильні та слабкі сторони популярних методів оптимізації нейронних мереж
3. Розробити та реалізувати алгоритм мурашиної колонії (ACO) для оптимізації архітектури нейронних мереж
4. Реалізувати базові алгоритми порівняння (генетичні алгоритми, байєсівська оптимізація, випадковий пошук)
5. Підготувати стандартні набори даних для тестування
6. Спланувати та провести експерименти для порівняння ефективності методів оптимізації
7. Проаналізувати експериментальні результати та оцінити методи на основі заданих метрик
8. Обговорити практичні міркування та обмеження застосування ACO та інших методів
9. Надати рекомендації щодо вибору оптимального методу для певних типів задач
10. Узагальнити результати та написати підсумковий звіт

4. План роботи

№ з/п	Назви етапів роботи
1	Огляд літератури та аналіз методів
2	Визначення проблеми та постановка цілей
3	Розробка ACO-алгоритму
4	Реалізація методів порівняння
5	Підготовка наборів даних
6	Експериментальний дизайн
7	Проведення експериментів
8	Аналіз результатів
9	Вдосконалення алгоритмів
10	Написання звітів та підготовка публікацій

5. Дата видачі завдання 10 вересня 2024 року

Студент _____ Андрій МІРОШНИЧЕНКО

Керівник роботи _____ Артем Панченко

АННОТАЦІЯ

Ця робота охоплює 88 сторінки, структуровані в 3 розділи, і включає 7 малюнків, 1 таблицю і 10 посилань. Робота досліджує оптимізацію топології нейронної мережі, зосереджуючись на алгоритмі оптимізації мурашиної колонії (ACO), поряд із такими порівняльними методами, як генетичні алгоритми, байєсовська оптимізація та випадковий пошук. Дослідження має на меті оцінити їх сильні та слабкі сторони в оптимізації архітектури нейронної мережі.

Методологія дослідження включає огляд літератури, розробку алгоритму та експериментальну оцінку. Python із такими бібліотеками, як TensorFlow і PyTorch, служив основним середовищем програмування. Експерименти використовували такі набори даних, як MNIST і CIFAR-10, що забезпечувало надійні та відтворювані результати.

Ключові висновки підкреслюють ефективність ACO у збалансуванні розвідки та експлуатації, що дозволяє досягти конкурентоспроможності в оптимізації нейронних мереж. Порівняння з альтернативними методами показує переваги ACO у швидкості та ефективності конвергенції. Внески включають адаптовану реалізацію ACO для нейронних мереж і поглиблену оцінку за встановленими методами.

Практичні рекомендації, засновані на результатах, пропонують вказівки щодо вибору методів оптимізації для різної складності мережі та пропонують модифікації для покращення продуктивності методу в конкретних сценаріях.

Ключові слова: нейронні мережі, оптимізація, оптимізація колонії мурах, генетичні алгоритми, байєсовська оптимізація, випадковий пошук.

ANNOTATION

This work spans 88 pages, structured into 5 chapters, and includes 7 figures, 1 table, and 10 references. It investigates neural network topology optimization, focusing on the Ant Colony Optimization (ACO) algorithm, alongside comparative methods such as Genetic Algorithms, Bayesian Optimization, and Random Search. The study aims to assess their strengths and weaknesses in optimizing neural network architectures.

The research methodology includes a literature review, algorithm development, and experimental evaluation. Python, with libraries like TensorFlow and PyTorch, served as the primary programming environment. Experiments utilized datasets such as MNIST and CIFAR-10, ensuring reliable and reproducible results.

Key findings highlight ACO's effectiveness in balancing exploration and exploitation, enabling it to achieve competitive performance in optimizing neural networks. Comparisons with alternative methods reveal ACO's advantages in convergence speed and efficiency. Contributions include a tailored implementation of ACO for neural networks and an in-depth evaluation against established methods.

Practical recommendations based on the findings offer guidance on selecting optimization techniques for varying network complexities and suggest modifications to improve method performance in specific scenarios.

Keywords: Neural Networks, Optimization, Ant Colony Optimization, Genetic Algorithms, Bayesian Optimization, Random Search.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

ACO: Optimization Ant Colony Optimization – природний метаевристичний алгоритм оптимізації, заснований на поведінці мурах у пошуках їжі.

GA: генетичний алгоритм – еволюційний алгоритм для задач оптимізації, який імітує процес природного відбору.

BO: Байєсова оптимізація – метод імовірнісної оптимізації для знаходження мінімуму функції з мінімальними оцінками.

RS: Випадковий пошук – простий метод оптимізації, що включає випадкову вибірку варіантів-кандидатів.

GPU: графічний процесор – спеціалізований процесор, розроблений для високопродуктивних обчислень, який часто використовується для навчання нейронних мереж.

FLOP: операції з плаваючою комою – міра обчислювальної складності, яка використовується для оцінки ефективності алгоритмів.

Дослідження: процес пошуку нових, малодосліджених областей простору рішень під час оптимізації.

Експлуатація: процес вдосконалення та покращення відомих хороших рішень у оптимізації.

Простір пошуку: область усіх можливих рішень, які може дослідити алгоритм оптимізації.

Цільова функція: математична функція, яка використовується для оцінки якості потенційних рішень у задачах оптимізації.

ВСТУП

Мета

Метою цього дослідження є розробка, впровадження та оцінка підходу до оптимізації топології нейронної мережі на основі оптимізації мурашиної колонії (ACO). Зокрема, дослідження спрямоване на використання ACO як метаевристики для автоматичної побудови наближених оптимальних топологій для нейронних мереж, досягнення балансу між прогностичною продуктивністю (наприклад, точністю) та обчислювальною ефективністю (наприклад, складністю моделі та часом навчання). Шляхом адаптації ACO до проблеми пошуку архітектури нейронної мережі це дослідження має на меті:

Продемонструвати доцільність ACO у розробці нейронних мереж, області, яка традиційно досліджується за допомогою інших алгоритмів пошуку.

Оцінити ефективність ACO у виявленні ефективних і високопродуктивних структур нейронної мережі.

Порівняти продуктивність ACO в оптимізації топології з іншими стандартними методами, такими як генетичні алгоритми, байєсовська оптимізація та випадковий пошук, з точки зору таких показників, як точність, обчислювальна вартість, швидкість конвергенції та стійкість.

Дослідження намагатиметься встановити, чи може ACO надійно виявляти мережеві архітектури, які досягають порівнянних або кращих результатів з меншими обчислювальними витратами, забезпечуючи новий метод для масштабованого та адаптивного проектування нейронної мережі.

Предмет дослідження

Основним предметом цієї роботи є оптимізація топології нейронної мережі за допомогою оптимізації Ant Colony Optimization (ACO). У цьому дослідженні

конкретно розглядається проблема проектування нейронних мереж шляхом визначення оптимальних або майже оптимальних топологій (таких як кількість шарів, кількість нейронів на шар і з'єднань), які дають високу точність прогнозування з мінімізованими обчислювальними витратами.

У цьому предметі оптимізація мурашиної колонії застосовується як метаевристичний алгоритм пошуку, адаптуючи парадигму мурашиної колонії (що характеризується слідами феромонів і поведінкою спільного пошуку) до простору пошуку архітектури нейронних мереж. Дослідження досліджує потенціал ACO для виявлення як структурних компонентів нейронних мереж, так і оптимальних конфігурацій цих компонентів. Предмет також охоплює порівняльний аналіз ACO з іншими широко використовуваними алгоритмами оптимізації, щоб оцінити його ефективність і ефективність у створенні добре продуктивних мережевих архітектур.

Об'єкт дослідження:

Основною об'єктом дослідження цієї роботи є розробка та перевірка підходу на основі оптимізації мурашиної колонії, який може генерувати приблизні оптимальні топології нейронної мережі. Це передбачає досягнення топології, яка встановлює баланс між точністю та ефективністю, таким чином дозволяючи масштабований і обчислювальний процес проектування нейронної мережі. Другорядними цілями є:

Розробити практичну реалізацію ACO, призначену для оптимізації топології нейронної мережі.

Встановити параметри, правила та показники продуктивності, необхідні ACO для ефективного дослідження та використання простору пошуку архітектури.

Порівняти топології, створені ACO, із топологіями, створеними за допомогою встановлених методів, таких як генетичні алгоритми, байєсовська

оптимізація та випадковий пошук, щоб оцінити відносну ефективність і обчислювальну ефективність ACO.

Надати уявлення про переваги, обмеження та практичні міркування використання ACO для проектування нейронної мережі, тим самим вносячи новий метод у область пошуку нейронної архітектури.

Досягнення цих цілей сприятиме розвитку галузі, пропонуючи потенційну альтернативу традиційним методам пошуку нейронної архітектури, особливо для програм, де зниження обчислювальних витрат є важливим.

Сфера застосування:

Це дослідження буде зосереджено на використанні алгоритму оптимізації мурашиної колонії для оптимізації топології прямої нейронної мережі на стандартних контрольних наборах даних (наприклад, MNIST або CIFAR-10). Сфера застосування зокрема включатиме:

Вибір топології: дослідження обмежується виявленням оптимальної структури прямої нейронної мережі. Хоча це включає вибір кількості шарів, нейронів на шар і з'єднань, воно не заглиблюється в інші нейронні архітектури (наприклад, згорткові або рекурентні мережі) або в оптимізацію параметрів, таких як ваги та зміщення.

Алгоритмічна адаптація: ACO буде адаптовано для дискретного та високовимірного простору пошуку, характерного для топологій нейронних мереж. Це включатиме модифікацію стандартних компонентів ACO, таких як сліди феромонів і евристичні фактори, відповідно до пошуку нейронної архітектури.

Порівняльний аналіз: продуктивність ACO порівнюватиметься з генетичними алгоритмами, байєсовською оптимізацією та випадковим пошуком як альтернативними підходами. Кожен метод буде оцінено на основі його здатності створювати високоточні мережі з мінімальними обчислювальними витратами.

Метрики оцінки: дослідження оцінюватиме кожну топологію на основі таких показників, як точність тестових даних, складність моделі (кількість параметрів або FLOP) і час навчання. Обмеження ресурсів, зокрема обчислювальної потужності стандартного ноутбука, обмежить обсяг з точки зору розміру набору даних і складності моделі.

Дослідження буде обмежуватися нейронними мережами прямого зв’язку, за винятком інших архітектур через обмеження ресурсів.

Дослідження гіперпараметрів (таких як швидкість навчання, розмір партії) не буде основним фокусом; дослідження буде віддавати пріоритет структурним аспектам (топології).

Враховуючи обчислювальні обмеження, великомасштабні або дуже глибокі нейронні мережі можуть бути неможливими в межах обсягу, тому дослідження буде зосереджено на помірно складних архітектурах.

Практичні обмеження також означають, що вичерпне або обширне тестування на великих різноманітних наборах даних виходить за рамки, зосереджуючись на менших керованих наборах даних для підтвердження концепції.

Цей обсяг забезпечує цілеспрямоване дослідження здатності ACO оптимізувати мережеві топології в умовах обчислювальних обмежень, визнаючи, що в майбутньому робота може розширити це до більш складних архітектур і більших наборів даних.

АНАЛІЗ ТЕОРЕТИЧНИХ ДАНИХ

У цьому розділі розглядатимуться наявні роботи та відповідні дослідження в області оптимізації архітектури нейронної мережі та оптимізації мурашиної колонії (ACO). Буде обговорено різні методи, що використовуються для оптимізації топології нейронної мережі, включаючи еволюційні алгоритми та методи пошуку, підкреслюючи потенційне застосування ACO у цій галузі.

Оптимізація архітектури нейронної мережі відноситься до процесу визначення оптимальної конфігурації нейронної мережі, такої як кількість шарів, кількість нейронів у кожному шарі, тип функцій активації та інші варіанти дизайну, які можуть впливати на продуктивність моделі. Це складне завдання через високу розмірність простору пошуку, взаємозалежність між вибором архітектури та обчислювальну вартість навчання великих мереж.

Проблеми в оптимізації топології нейронної мережі:

Розмір простору пошуку: нейронні мережі мають велику кількість гіперпараметрів, які необхідно оптимізувати, включаючи кількість шарів, нейронів на шар, функції активації, швидкість навчання тощо. Кожен із цих варіантів впливає на загальну продуктивність, і їх взаємодія може бути складною.

Витрати на обчислення. Навчання нейронних мереж є дорогим з точки зору обчислень, особливо коли задіяні великі мережі. Проведення великої кількості експериментів для пошуку найкращої конфігурації може бути непомірно витратним з точки зору часу та ресурсів.

Переобладнання/недообладнання: оптимізація топології, уникаючи переобладнання або недообладнання, є ще однією серйозною проблемою. Мережа має бути достатньо складною, щоб фіксувати закономірності в даних, але не бути надто великою для якісного узагальнення.

Підходи до оптимізації топології нейронної мережі:

Ручне проектування: це традиційний підхід, коли експерт вручну проектує архітектуру на основі інтуїції та досвіду. Цей метод сильно залежить від знань і досвіду людини, але займає багато часу та не має масштабованості.

Пошук у сітці: основний, широко використовуваний метод, коли простір пошуку гіперпараметрів дискретизується у вигляді сітки. Кожна комбінація параметрів тестується, і вибирається найкраща на основі показників ефективності. Однак цей підхід є неефективним, оскільки він не використовує зв’язки між гіперпараметрами та призводить до надмірних обчислювальних витрат.

Випадковий пошук: на відміну від пошуку в сітці, випадковий пошук передбачає випадковий вибір комбінацій гіперпараметрів для оцінки. Цей метод іноді може знаходити хороші конфігурації ефективніше, ніж пошук у сітці, особливо у просторах великої розмірності.

Байєсовська оптимізація: ця методика моделює цільову функцію за допомогою імовірнісних моделей і використовує їх для спрямування пошуку в перспективних областях простору гіперпараметрів. Він більш ефективний, ніж сітка та випадковий пошук, оскільки використовує попередні оцінки для вибору наступного набору гіперпараметрів для перевірки.

Генетичні алгоритми (GA): — це еволюційні алгоритми, натхненні природним відбором. Вони використовують такі оператори, як мутація, схрещування та відбір, щоб розвивати популяцію варіантів рішень протягом кількох поколінь. GA були успішно застосовані для оптимізації нейронних мереж завдяки їхній здатності здійснювати пошук у великих і складних просторах.

ВИЗНАЧЕННЯ ПРОБЛЕМИ

Незважаючи на останні досягнення в автоматизованому пошуку нейронної архітектури, ACO представляє унікальну перевагу завдяки своїй здатності ймовірнісно досліджувати простір пошуку, підтримувати баланс між дослідженням і використанням і адаптуватися на основі зворотного зв'язку (феромони).

Притаманний ACO паралелізм у поєднанні з його надійністю в навігації просторами великої розмірності робить його ідеальним кандидатом для оптимізації нейронної мережі. За допомогою ACO ці дослідження спрямовані на автоматизацію процесу пошуку ефективних мережевих архітектур, які можуть перевершити моделі, розроблені вручну, або моделі, оптимізовані за допомогою традиційних методів пошуку.

Питання дослідження

Як можна адаптувати алгоритм ACO для оптимізації топології нейронної мережі?

Мета: дослідити, як природну поведінку мурах, таку як оновлення феромонів і вибір шляху, можна перекласти на пошук архітектури нейронної мережі. Дослідження вивчатиме, як мурахи можуть представляти потенційні мережеві архітектури та як їхні сліди феромонів можуть направляти пошук до кращих конфігурацій.

Адаптація: це передбачає визначення того, як архітектурні рішення (наприклад, кількість шарів, нейронів на шар) будуть закодовані в рішеннях для мурах, як функціонуватиме механізм оновлення феромонів і як локальний пошук (оптимізація однієї архітектури) і глобальний пошук (вивчення) весь архітектурний простір) буде збалансованим.

Які ключові показники (наприклад, точність, обчислювальна ефективність), які вказують на оптимальну або майже оптимальну топологію?

Мета: Встановити критерії оцінки для визначення якості архітектури нейронної мережі. Точність (продуктивність перевірки/тестування) і ефективність обчислень (час навчання, кількість параметрів, час висновку) є загальними показниками для оптимізації нейронної мережі. Дослідження оцінить, як ці показники пов’язані з пошуком оптимальної топології.

Показники, які слід враховувати:

Точність: ефективність моделі під час перевірки та тестування наборів даних.

Обчислювальна ефективність: час, необхідний для навчання мережі, використання пам’яті та загальна кількість параметрів, які можна навчити.

Узагальнення моделі: здатність моделі узагальнювати нові, невідомі дані, уникаючи переобладнання.

Як ACO порівнюється з іншими стандартними методами оптимізації з точки зору пошуку ефективних мережевих структур?

Мета: порівняти ACO з іншими популярними методами оптимізації, такими як генетичні алгоритми (GA), байєсовська оптимізація (BO), пошук у сітці та випадковий пошук, щоб визначити, чи може ACO забезпечити кращу або конкурентоспроможну альтернативу.

Фактори порівняння:

Якість оптимізації: наскільки добре метод знаходить оптимальне або майже оптимальне рішення з точки зору точності.

Ефективність: час і обчислювальні ресурси, необхідні для пошуку оптимального рішення.

Масштабованість: наскільки добре працює метод у міру збільшення простору пошуку (наприклад, у міру збільшення кількості шарів або нейронів).

Гіпотези

Для дослідження ACO в оптимізації топології нейронної мережі пропонуються такі гіпотези:

Гіпотеза 1: ACO може знайти конкурентоспроможну або навіть кращу топологію нейронної мережі порівняно з традиційними методами.

Обґрунтування: хоча ручне проектування, пошук у сітці та випадковий пошук є поширеними, вони часто страждають від неефективності у просторах великої розмірності. Еволюційні алгоритми, такі як GA та байєсовська оптимізація, ефективніші, але все ще мають обмеження у дослідженні складних просторів пошуку. Здатність ACO досліджувати простір рішень імовірнісно, використовуючи феромони для керування пошуком, може привести до виявлення оптимальних топологій більш ефективно.

Гіпотеза 2: оптимізація топології на основі ACO може призвести до швидшої конвергенції до високопродуктивної мережі порівняно з сіткою або випадковим пошуком.

Обґрунтування: сітка та випадковий пошук включають оцінку багатьох конфігурацій без використання попередніх знань, що може призвести до повільної конвергенції. ACO, з іншого боку, збалансовує дослідження та експлуатацію, що може призвести до швидшої конвергенції до високопродуктивних мережевих конфігурацій. Використання оновлень феромонів для уточнення пошуку може зменшити кількість ітерацій, необхідних для досягнення майже оптимального рішення, особливо у великих просторах пошуку.

Пояснення визначення проблеми:

Завдання оптимізації нейронної мережі передбачає вибір найкращої можливої архітектури з величезного простору можливостей. Це може включати налаштування кількості шарів, кількості нейронів у кожному шарі, функцій активації тощо. гіперпараметри. Мета полягає в тому, щоб знайти конфігурацію, яка максимізує продуктивність (наприклад, точність) і мінімізує обчислювальні витрати (наприклад, час навчання, використання пам’яті).

ACO як рішення: ACO є природним кандидатом на вирішення цієї проблеми, оскільки він дозволяє здійснювати ймовірнісний пошук у складних просторах великої розмірності. Алгоритм моделює поведінку мурах, які знаходять шляхи на графі, де хороші шляхи (тобто кращі мережеві архітектури) підкріплюються феромонними слідами, керуючими наступними пошуками. Гнучкість ACO у збалансуванні дослідження (випробування нових архітектур) і експлуатації (удосконалення відомих хороших архітектур) робить його особливо корисним для оптимізації нейронної мережі, де задіяний великий простір пошуку, а рішення важко передбачити заздалегідь.

1 МУРАШИНИЙ АЛГОРИТМ

Алгоритм оптимізація колонії мурашок (Ant Colony Optimization algorithm) — це алгоритм оптимізації, натхненний поведінкою мурах у пошуках їжі. Він особливо добре підходить для задач дискретної оптимізації, таких як маршрутизація, планування та комбінаторна оптимізація, і базується на концепції феромонних слідів, які направляють мурах до оптимальних рішень.

Рисунок 1 демонструє фундаментальні принципи алгоритму оптимізації мурашиної колонії (ACO) у пошуку оптимальних шляхів між джерелом їжі (F) і їхнім гніздом (N). Процес розбивається на три основні етапи:

Дослідження та формування слідів: на першому етапі мурахи залишають гніздо та випадково досліджують навколишнє середовище, щоб знайти їжу. Під час подорожі вони залишають сліди феромонів (позначено лінією b на рисунку). Ці стежки служать механізмом спілкування для інших мурах, позначаючи шляхи, якими вони йдуть під час дослідження.

Слідування за стежками та вибір шляху: коли деякі мурахи виявляють джерело їжі, вони повертаються до гнізда, зміцнюючи шляхи додатковими феромонами. Інтенсивність феромонів на шляху зростає, більше мурах проходить його, що робить його більш привабливим для наступних мурах. На цій стадії більше мурах починає слідувати слідами з більшою концентрацією феромонів, що призводить до імовірнісного вибору шляхів на основі їх сили феромонів.

Підсилення та оптимізація: з часом, як показано на третьому етапі, коротші та ефективніші шляхи накопичують більше феромонів через їх повторне використання та коротший час проходження. Це призводить до позитивної петлі зворотного зв’язку, де коротші шляхи зміцнюються, а довші шляхи поступово відмовляються. Зрештою, колективна поведінка мурах сходиться на найкоротшому, найбільш оптимальному шляху.

На цій діаграмі представлено основний механізм ACO: збалансоване дослідження (пошук нових шляхів) і використання (зміцнення відомих оптимальних шляхів). Така поведінка робить ACO дуже ефективним для вирішення проблем комбінаторної оптимізації, таких як розробка топології нейронної мережі, де алгоритм може ймовірнісно досліджувати величезний простір пошуку та знаходити високоефективні рішення.

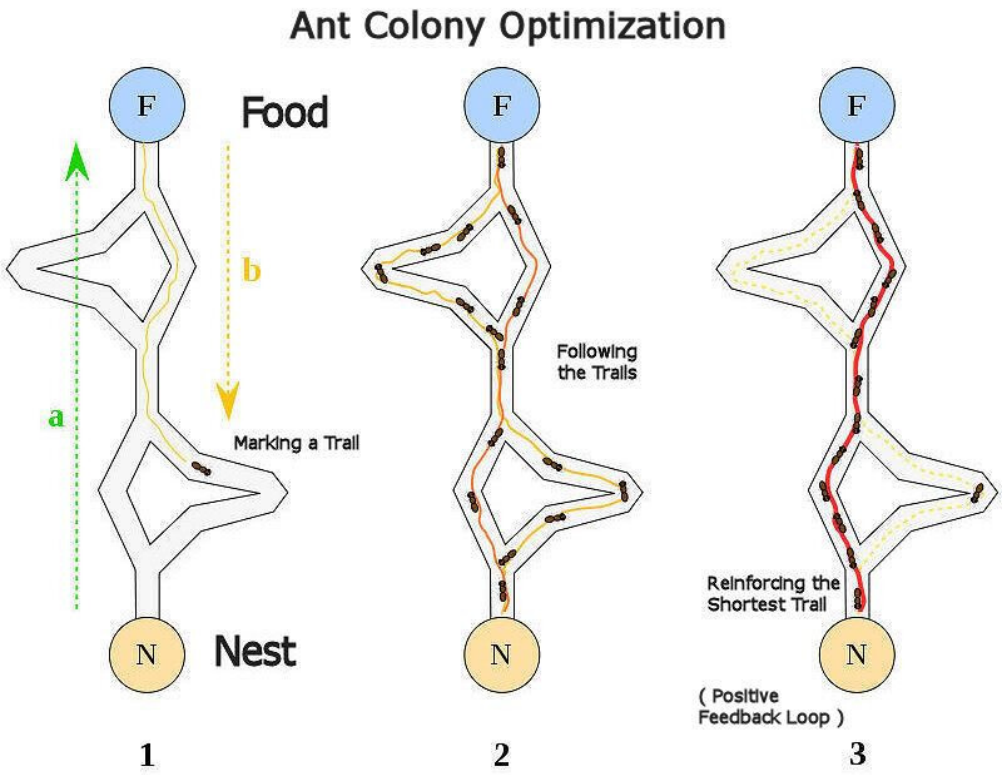

Рисунок 1 — механізм роботи ACO

1.1 Механізм ACO

Оновлення щодо феромонів: мурахи відкладають феромони на шляхах, які вони проходять, що впливає на ймовірність того, що майбутні мурахи підуть тим самим шляхом. Кількість осідлого феромону залежить від якості розчину, знайденого мурахою. З часом феромон на менш оптимальних шляхах випаровується, роблячи їх менш привабливими.

Розвідка проти експлуатації: ACO встановлює баланс між розвідкою (випробуванням нових шляхів) та експлуатацією (посиленням пошуку на перспективних шляхах). Це робиться за допомогою ймовірнісного процесу, коли мурахи вибирають наступний крок на основі рівня феромонів, а іноді й випадкового дослідження.

Паралелізм: кілька мурах досліджують різні шляхи одночасно, забезпечуючи паралельний підхід до вирішення проблеми оптимізації.

1.2 Застосування ACO

ACO було успішно застосовано в кількох областях, зокрема:

Комбінаторна оптимізація: такі проблеми, як проблема комівояжера (TSP), маршрут транспортного засобу та робоча схема duling отримали вигоду від ACO, де мурахи використовуються для пошуку майже оптимальних рішень шляхом дослідження простору пошуку.

Безперервна оптимізація: спочатку розроблений для дискретних проблем, ACO був адаптований для вирішення проблем безперервної оптимізації шляхом дискретизації простору або використання безперервних феромонних моделей.

Машинне навчання: ACO використовувався для оптимізації гіперпараметрів і навчання моделей машинного навчання, де для точного налаштування моделей використовується механізм оновлення феромонів.

Адаптованість ACO до складних проблем пошуку: ACO добре підходить для складних задач оптимізації завдяки своїй здатності досліджувати великі простори пошуку, одночасно використовуючи попередні знання за допомогою оновлення феромонів. Його надійність у роботі з динамічними та шумними середовищами зробила його кандидатом для застосування в оптимізації топології нейронної мережі.

1.3 ACO в оптимізації нейронної мережі

Використання ACO для оптимізації нейронних мереж стало відносно нещодавнім розвитком у сфері машинного навчання. Дослідники дослідили, як ACO можна використовувати для оптимізації топології нейронної мережі, гіперпараметрів або обох. Моделюючи поведінку мурах, ACO може ефективно шукати величезний простір потенційних архітектур і знаходити конфігурації, які покращують продуктивність моделі.

Деякі ключові області, де ACO було застосовано для оптимізації нейронної мережі:

Дизайн топології нейронної мережі: ACO можна використовувати для визначення оптимальної кількості шарів, нейронів на шар і зв’язків між ними. Мурахи представляють можливі архітектури, а процес оновлення феромонів спрямовує пошук до кращих топологій.

Налаштування гіперпараметрів: ACO також може оптимізувати швидкість навчання, функції активації та інші гіперпараметри. Симулюючи кілька мурах, які досліджують різні конфігурації, ACO може ефективно налаштовувати гіперпараметри для підвищення продуктивності мережі.

Вибір функцій: ACO використовувався для вибору найбільш релевантних функцій із великих наборів даних, керуючи мурахами вибором підмножин ознак, які максимізують точність моделі.

Оптимізація тренувального процесу: деякі дослідження досліджували поєднання ACO з іншими методами оптимізації для оптимізації самого тренувального процесу, такими як планування темпів навчання або ініціалізація ваги.

1.4 Порівняльні алгоритми

Щоб забезпечити контекст для підходу на основі ACO, важливо порівняти його з іншими алгоритмами оптимізації, які використовувалися для оптимізації

архітектури нейронної мережі. Тут ми описуємо ключові особливості деяких із цих алгоритмів:

Генетичні алгоритми (GA): GA, як і ACO, є еволюційними алгоритмами та мають схожий підхід до вирішення проблем оптимізації. Хоча як GA, так і ACO працюють із населенням і з часом розвивають рішення, ACO, як правило, більше зосереджується на локальному пошуку та механізмах оновлення феромонів, тоді як GA використовують оператори кросинговеру та мутації. GA може мати проблеми з локальними оптимумами, тоді як оновлення феромонів ACO допомагає уникнути передчасної конвергенції.

Байєсова оптимізація (BO): BO — це метод оптимізації на основі імовірнісної моделі, який створює сурогатну модель цільової функції та використовує її для вибору нових гіперпараметрів, які, ймовірно, покращать продуктивність. На відміну від ACO, який є популяційним методом, BO використовує єдину модель і використовує минулі спостереження для спрямування майбутніх пошуків, що робить його ефективним, але потенційно менш гнучким у великих і складних просторах пошуку.

Оптимізація роєм частинок (PSO): PSO – це ще один метод оптимізації на основі популяції, натхненний соціальною поведінкою птахів або риб. Як і ACO, PSO досліджує простір рішень за допомогою агентів (частинок), які спілкуються та оновлюють свої позиції на основі особистих і глобальних найкращих позицій. PSO часто порівнюють з ACO з точки зору швидкості конвергенції та здатності уникнути локальних оптимумів.

Випадковий пошук: хоча це простіший підхід порівняно з ACO, випадковий пошук іноді використовується як базовий через його простоту та здатність швидко досліджувати великий простір пошуку. Однак довільний пошук не використовує будь-яку минулу інформацію про простір пошуку та може бути неефективним для складних проблем.

2 МЕТОДОЛОГІЯ ДОСЛІДЖЕННЯ

Метою розділу методології є окреслення експериментального підходу, який використовується для реалізації та оцінки оптимізації мурашиної колонії (ACO) для оптимізації топології нейронної мережі. Це включає спеціальну адаптацію ACO для пошуку у величезному та складному просторі архітектур нейронних мереж, визначення експериментальної установки, критеріїв оцінки та порівняння з іншими алгоритмами оптимізації. Комплексна та систематична методологія гарантує, що експеримент є відтворюваним, інтерпретованим і науково обґрунтованим.

2.1 Адаптація ACO для топології нейронної мережі

Застосування оптимізації Ant Colony Optimization (ACO) для оптимізації топології нейронної мережі передбачає адаптацію алгоритму ACO для дослідження простору проектування архітектур нейронної мережі. Цей процес вимагає визначення компонентів архітектури нейронної мережі, таких як кількість шарів, нейронів на шар і функції активації. Крім того, правила оновлення феромонів в ACO відіграють вирішальну роль у спрямуванні процесу пошуку до оптимальних рішень. У цьому розділі ми детально опишемо, як ACO буде застосовуватися для пошуку в просторі нейронної мережі, визначимо параметри, які використовуватиме кожна мураха, і пояснимо правила оновлення феромонів, зосереджуючись на балансі дослідження та експлуатації.

ACO застосовується до простору нейронних мереж пошуку

ACO — це метод оптимізації, натхненний біотехнологіями, заснований на поведінці мурашок у пошуках їжі, коли мурахи пересуваються через пошуковий простір і відкладають феромони, щоб спілкуватися один з одним. Ключова ідея полягає в тому, що мурахи, як правило, слідують шляхами з найбільшою кількістю феромонів, підкріплюючи хороші рішення, водночас дозволяючи алгоритму досліджувати менш пройдені шляхи через випаровування феромонів. У контексті

оптимізації топології нейронної мережі «простір пошуку» відноситься до простору всіх можливих архітектур нейронної мережі, і ACO досліджуватиме цей простір, щоб знайти оптимальну або майже оптимальну архітектуру.

Простір архітектур нейронних мереж великий і багатовимірний, включає кілька параметрів, які визначають структуру та поведінку моделі. ACO використовуватиметься для пошуку оптимальної архітектури шляхом навігації між такими ключовими компонентами нейронної мережі:

Нейронні мережі можуть мати один або кілька прихованих шарів між вхідним і вихідним шарами. Глибина (кількість шарів) мережі є критичним фактором її здатності вивчати складні моделі.

Кожна мураха повинна вирішити, скільки прихованих шарів матиме мережа. Це дискретна змінна рішення.

Кількість нейронів у кожному шарі впливає на пропускну здатність мережі. Занадто мало нейронів може призвести до недостатнього підбору, тоді як занадто багато нейронів може призвести до надмірного пристосування.

Кожна мураха вибере кількість нейронів для кожного прихованого шару. Це може бути дискретне значення в попередньо визначеному діапазоні (наприклад, від 5 до 200 нейронів).

Функція активації визначає, як трансформується вхід до нейрона. Загальні функції активації включають ReLU, Sigmoid, Tanh і Softmax.

Кожна мураха вибере функцію активації для кожного шару, впливаючи на нелінійність і навчальну поведінку мережі.

Швидкість навчання та тип оптимізатора також можна вважати частиною простору пошуку, але для простоти їх можна виправити або скорегувати в наступних експериментах.

Алгоритм ACO керуватиме дослідженням цих компонентів, щоб виявити топології, які збалансовують продуктивність моделі (наприклад, точність) і вартість обчислень (наприклад, час навчання, кількість параметрів).

2.2 Параметри для кожного мурахи

Кожна мураха в ACO побудує повну архітектуру нейронної мережі. Параметри для кожної мурашки:

Кількість шарів (L): кількість прихованих шарів мережі. Наприклад, мураха може вибрати від 1 до 10 шарів.

$$L \in \{1,2,3, \dots., L_{max}\}$$

Місце пошуку для L може бути неперервним, але для простоти ми його дискретизуємо.

Нейронів на шар N_l: кількість нейронів у кожному прихованому шарі. Для кожного вибраного шару мураха вибере певну кількість нейронів.

N_l $[N_{min}, N_{max}]$ для кожного шару.

N_{min} та N_{max} визначають мінімальну та максимальну кількість нейронів, дозволену на кожному шарі. Наприклад, діапазон може складати від 5 до 200 нейронів.

Функції активації (A_l): функція активації, яка буде використовуватися на кожному рівні. Загальні варіанти включають ReLU, Sigmoid, Tanh тощо.

$$A_l \in \{ReLU, Sigmoid, Tanh, Softmax\}$$

Це рішення впливає на нелінійні перетворення, застосовані до входів кожного нейрона.

Кожна мураха побудує нейронну мережу, вибравши значення для $L,\ N_l$ та A_l

для кожного рівня в мережі. Коли мережа побудована, її навчають і оцінюють її продуктивність.

2.3 Правила оновлення феромонів

Феромони відіграють центральну роль у керуванні процесом пошуку ACO. Після того як мураха побудує топологію нейронної мережі, вона оцінить її продуктивність за допомогою функції пристосування. Правила оновлення феромонів зміцнюють ефективніші архітектури та направляють майбутніх мурах до перспективних областей простору пошуку.

Правило оновлення феромонів складається з двох основних компонентів: осадження феромонів і випаровування феромонів.

Після оцінки продуктивності мурахи (яка зазвичай базується на точності, тривалості тренувань або іншому показнику фізичної підготовки), мураха відкладає феромони, щоб «позначити» успішні шляхи (топології), які вона знайшла.

Кількість феромону де положення обернено пропорційно придатності розв’язку. Це означає, що кращі рішення (тобто з вищою точністю або нижчими обчислювальними витратами) відкладатимуть більше феромонів, що зробить їх більш привабливими для майбутніх мурах.

Відкладення феромонів для мурашки i задано:

$$\Delta\tau_i = Q \times \frac{1}{Fitness_i}$$

$\Delta\tau_i$ кількість феромону, що виділяється ант

Q — константа, яка масштабує кількість феромону.

$Fitness_i$ це придатність рішення, як правило, точність перевірки або поєднання точності та обчислювальних витрат.

Примітка. Якщо використовується кілька критеріїв оцінювання (наприклад, точність і вартість обчислень), для обчислення функції придатності може використовуватися зважена сума:

$$Fitness_i = w_1 \times Accuracy_i + w_2 \times Inverse\ Training\ Time_i$$

Де w_1 і w_2 це ваги, які збалансовують важливість точності та обчислювальних витрат.

Щоб запобігти передчасному зближенню та стимулювати дослідження нових шляхів, феромон з часом випаровується. Швидкість випаровування визначає, як швидко старі сліди феромонів втрачають свій вплив.

Випаровування допомагає алгоритму уникнути застрягання в субоптимальних областях простору пошуку, зменшуючи вплив слідів феромонів, які більше не сприяють хорошим рішенням.

Правило випаровування феромонів визначається так:

$$\tau_{new} = \rho \times \tau_{old} + \Delta\tau_i$$

Де:

τ_{new} це новий рівень феромонів.

τ_{old} це попередній рівень феромону.

ρ – швидкість випаровування ($0 < \rho < 1$). Вищий ρ сприяє більш швидкому дослідженню, тоді як нижчий ρ підкреслює експлуатацію.

Розвідка проти експлуатації:

Баланс між розвідкою (пошуком нових областей пошукового простору) та експлуатацією (доопрацюванням відомих хороших рішень) є ключовою характеристикою ACO. Високий рівень феромонів заохочує до експлуатації, направляючи мурах на шляхи з раніше успішною архітектурою. Однак випаровування феромонів запобігає надмірному зосередженню алгоритму на

старих рішеннях, дозволяючи йому досліджувати нові рішення, які згодом можуть призвести до кращих архітектур.

Щоб точно налаштувати цей баланс, швидкість дослідження можна контролювати, регулюючи швидкість випаровування ρ. Наприклад:

Вищі швидкості випаровування (низькі ρ) заохочуйте більше досліджувати, оскільки сліди феромонів руйнуватимуться швидше, що призведе до більш випадкової поведінки пошуку.

Нижча швидкість випаровування (висока ρ) заохочуйте більше використання відомих хороших рішень, зберігаючи сліди феромонів довше.

3 ЗАГАЛЬНІ ВІДОМОСТІ РЕАЛІЗАЦІЇ

Успіх застосування ACO для оптимізації топології нейронної мережі залежить не тільки від теоретичної основи, а й від практичних аспектів реалізації. У цьому розділі ми описуємо використовувані засоби програмування та бібліотеки, набори даних, обрані для тестування та перевірки, а також критерії оцінки для оцінки продуктивності топологій нейронних мереж.

Реалізація використовує Python, універсальну та широко використовувану мову для завдань машинного навчання та оптимізації. Вибір Python зумовлений його великою екосистемою бібліотек, простотою інтеграції та підтримкою спільноти.

Ключові бібліотеки:

TensorFlow/Keras або PyTorch: це сучасні фреймворки глибокого навчання, які надають інструменти для створення, навчання та оцінки нейронних мереж. Обидва пропонують гнучкість для впровадження власних архітектур та їх інтеграції в конвеєр ACO.

TensorFlow: tensorflow.keras для визначення та навчання нейронних мереж.

Оптимізоване прискорення GPU для швидшого навчання моделі.

PyTorch (альтернатива):

Дуже динамічний і гнучкий, особливо корисний для експериментальних установок.

Простіше налагодження з миттєвим виконанням у порівнянні з графічним підходом TensorFlow.

NumPy: використовується для матричних операцій, вибірки та чисельних обчислень у алгоритмі ACO.

Scikit-learn:

Набори даних: попередньо зібрані набори даних, такі як Iris і MNIST, для початкових експериментів.

Попередня обробка: інструменти для стандартизації та нормалізації даних.

Показники продуктивності: функції для точності, матриці невизначеності та перехресної перевірки.

Matplotlib:

Візуалізація показників ефективності, збіжність ACO та порівняння методів.

Логування та паралельні обчислення:

Модуль журналювання Python для детальних журналів під час виконання.

Вибір набору даних

Вибір наборів даних має вирішальне значення для оцінки продуктивності та можливостей узагальнення запропонованого підходу. Вибирається поєднання наборів даних, щоб забезпечити надійність для різних типів проблем.

Набори даних:

MNIST (Модифікований національний інститут стандартів і технологій):

Стандартний набір рукописних цифр (зображення у відтінках сірого 28x28). Ідеально підходить для початкового порівняльного аналізу завдяки своїй керованій складності та широкому використанню.

CIFAR-10 (Канадський інститут перспективних досліджень):

Складається з 60 000 кольорових зображень (32x32) у 10 класах. Більш складний, ніж MNIST, представляючи потребу в більш глибоких і складніших архітектурах.

Попередня обробка:

Дані нормалізовані для забезпечення узгоджених діапазонів вхідних даних, що покращує продуктивність нейронної мережі.

Методи розширення (повороти, перевороти, масштабування) застосовуються до наборів даних, таких як CIFAR-10, для оцінки надійності архітектур.

Поділ набору даних:

Набори даних поділяються на набори для навчання, перевірки та тестування: Навчальний набір: використовується для оптимізації параметрів моделі.

Набір перевірки: використовується під час ACO для оцінки продуктивності топології.

Тестовий набір: зарезервовано для остаточної оцінки найкраще знайденої топології.

Загальні розподіли: 80%-10%-10% або 70%-20%-10%, залежно від розміру набору даних.

Критерії оцінювання

Оцінка якості топології нейронної мережі вимагає кількох показників, які враховують продуктивність, ефективність і складність. Ці критерії гарантують, що обрана архітектура збалансує високу точність із практичними обмеженнями, такими як вартість обчислень.

Точність перевірки: вимірює здатність моделі узагальнювати невидимі дані під час пошуку топології.

Точність тесту: оцінює здатність кінцевої моделі до узагальнення на повністю незалежному наборі тестів.

Формула точності:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

Обчислювальна вартість:

Час навчання: вимірює час, необхідний для навчання топології нейронної мережі. Враховує компроміс між обчислювальною ефективністю та продуктивністю.

Операції з плаваючою комою (FLOP): визначає обчислювальну складність моделі шляхом підрахунку кількості арифметичних операцій, виконаних під час логічного висновку. Можна розрахувати за допомогою бібліотек глибокого навчання або зовнішніх інструментів, таких як torchprofile (для PyTorch).

Використання пам'яті: оцінює пам'ять, необхідну для параметрів моделі та проміжні обчислення під час навчання та висновку.

Складність моделі:

Кількість параметрів: представляє загальні настроювані ваги та зміщення в нейронній мережі.

Розраховується як:

$$\sum_{l=1}^{L}(N_{l-1} \times N_l + N_l)$$

де:

L: Кількість шарів.

N_{l-1}: кількість нейронів у попередньому шарі.

N_l: кількість нейронів у поточному шарі.

Глибина та ширина архітектури:

Баланс між глибокими (багато шарів) і широкими (багато нейронів на шар) архітектурами має вирішальне значення для забезпечення ефективності та уникнення переобладнання.

Комбіновані показники:

Композитна функція придатності визначається для оцінки архітектур під час процесу ACO, що включає кілька критеріїв:

$$Fitness = w_1 \times Validation\ Accuracy - w_2 \times \log(Training\ Time) - w_3 \times \log(Number\ of\ Parameters)$$

де:

w_1, w_2, w_3 : Вагові коефіцієнти, які віддають пріоритет точності, швидкості та складності на основі сценарію використання.

Розпаралелювання та масштабованість

Реалізація використовує паралельні обчислення для оцінки кількох архітектур одночасно, використовуючи багатоядерний процесор. Це особливо важливо для алгоритму ACO, де кожна мураха оцінює окрему топологію. Кожна обрана архітектура мурахи паралельно навчається й оцінюється незалежно. Такі фреймворки, як tf.distribute від TensorFlow або DataParallel від PyTorch, сприяють розподіленому навчанню.

Масштабованість:

Реалізація розроблена для масштабування більших наборів даних (наприклад, ImageNet) і складніших архітектур (наприклад, ResNet), якщо дозволяють обчислювальні ресурси.

Інфраструктура реалізації для застосування ACO для оптимізації топології нейронної мережі побудована на надійній основі Python, з TensorFlow/PyTorch як основними інструментами глибокого навчання. Стандартні набори даних, такі як MNIST і CIFAR-10, забезпечують надійний тестовий полігон для оцінки ефективності ACO, тоді як такі показники, як точність, обчислювальна вартість і складність моделі, забезпечують комплексну оцінку. Завдяки розпаралелюванню та масштабованості ця реалізація добре підходить для ефективного дослідження великих і складних просторів пошуку.

4 РЕАЛІЗАЦІЯ

Мета розробки простору пошуку полягала в тому, щоб створити структуроване, але гнучке представлення топологій нейронних мереж, які АСО могла б ефективно досліджувати. Простір пошуку потрібен для охоплення різноманітних архітектур, залишаючись обчислювально можливим для навчання та оцінювання.

4.1 Визначення простору пошуку

Простір пошуку являє собою набір усіх можливих конфігурацій нейронної мережі, які мурахи можуть досліджувати. У цій роботі простір пошуку було визначено для включення ключових структурних елементів прямої нейронної мережі:

Кількість шарів (L):

Глибина нейронної мережі, яка визначається як кількість прихованих шарів, є критичним параметром. Діапазон від 1 до 5 шарів був обраний, щоб збалансувати виразність моделі з можливостями обчислення.

Обґрунтування: мережам із меншою кількістю рівнів може бракувати репрезентативної потужності для виконання складних завдань, тоді як глибші мережі схильні до переобладнання та вимагають більше обчислювальних ресурсів.

Реалізація: кожна мурашка могла вибрати L з дискретної множини:

$$L \in \{1,2,3,4,5\}$$

Кількість нейронів на шар (N): кількість нейронів у кожному шарі визначає здатність мережі вивчати складні шаблони.

діапазон: $N \in [10, 100]$, з кроком 10.

Чому цей діапазон? Менші значення (< 10) ризик недообладнання, оскільки мережа може не мати достатньої потужності для моделювання даних. Більші значення > 100 призводять до надмірно параметризованих моделей, збільшуючи час навчання та ризик переобладнання.

Представлення: Кожен шар i присвоюється значення N_i, з $i \in \{1,2, \ldots, L\}$. Наприклад, трирівнева мережа може мати:

$$[N_1, N_2, N_3 = [64, 32, 16]$$

Функції активації A: функції активації вводять нелінійність, дозволяючи мережам моделювати складні зв'язки. Було включено три часто використовувані активації:

ReLU: ефективний для більшості завдань, усуває проблеми зі зникаючим градієнтом. tanh: корисно для збалансованих результатів, особливо в регресії. sigmoid: історично важливий, хоча схильний до зникаючих градієнтів.

Кожен шар i призначили активацію A_i, обраний із набору $A = \{ReLU, tanh, sigmoid\}$.

З'єднання між шарами:

У той час як основна увага приділялася мережам прямого зв'язку, простір пошуку неявно враховував повністю пов'язані рівні. Цей дизайн спростив проблему, уникнувши більш складних архітектур, таких як повторювані або згорткові шари.

Представлення топології

Для полегшення алгоритмічного дослідження кожна топологія нейронної мережі була представлена у вигляді вектора:

$$Topology = [L, N_1, N_2, \ldots, N_L, A_1, A_2, \ldots, A_L]$$

Приклад: 3-рівнева мережа з:

Нейрони: [64,32,16]

Активації: $[ReLU, tanh, ReLU]$

Буде закодовано як:

$$Topology = [3,64,32,16, ReLU, tanh, ReLU]$$

Загальна кількість можливих конфігурацій була обчислена як:

$$Search\ Space\ Size = |L| \times |L|^L \times |A|^L$$

Для вибраних діапазонів:

$|L|$ = 5 (1-5 шарів)

$|N|$ = 10 (10, 20, ..., 100 нейронів)

$|A|$ = 3 (ReLU, tanh, sigmoid)

Розмір простору пошуку, припускаючи $L = 3$, був:

$$Search\ Space = 5 \times 10^3 \times 3^3 = 135000$$

Це підкреслює складність ефективного дослідження простору та виправдовує необхідність імовірнісної стратегії дослідження ACO.

Компроміс між складністю та здійсненністю: більший простір пошуку охоплює більше різноманітних архітектур, але збільшує обчислювальне навантаження.

Обмеження на L, N, A були обрані після того, як попередні експерименти виявили зменшення прибутків для більш глибоких або широких мереж на вибраних наборах даних.

Уникнення переобладнання: щоб запобігти переобладнанню, під час навчання за замовчуванням було додано шари вилучення (не є частиною пошуку).

Для управління ресурсами були встановлені обмеження на пам'ять і час навчання. Моделі з екстремальними параметрами (наприклад, усі шари, що містять 100 нейронів) були попередньо відфільтровані, якщо вони перевищували обчислювальні межі.

Дизайн простору пошуку збалансував виразність і обчислювальну здійсненність, дозволяючи ACO ефективно досліджувати значущі конфігурації нейронної мережі.

4.2 Побудова рішень

Кожна мураха будує рішення шляхом повторного вибору компонентів топології нейронної мережі, таких як кількість шарів, нейронів на шар і функцій активації. Цей процес керується ймовірнісними правилами ACO, що збалансовують розвідку та розробку.

Поетапне будівництво

Ініціалізація мурах – на початку кожної ітерації фіксована кількість мурах (m) випускаються для дослідження простору пошуку. Для цього дослідження, було вибрано $m = 10$ мурах, щоб збалансувати витрати на обчислення та різноманітність рішень.

Вибір кількості шарів (L): кожна мураха починає з вибору загальної кількості прихованих шарів у мережі. Це рішення є ймовірнісним і обумовлене рівнем феромонів τ_L і евристичної функції η_L.

$$P(L_i)\frac{\tau_{L_i}^{\alpha} \times \eta_{L_i}^{\beta}}{\sum_{j=1}^{|L|} \tau_{L_j}^{\alpha} \times \eta_{L_j}^{\beta}}$$

τ_{L_i} : рівень феромонів для підрахунку шару

η_{L_i}: евристичне значення, що вказує на бажаність L_i (наприклад, на основі історичної продуктивності або обчислювальної здійсненності). Наприклад, глибші мережі спочатку можуть мати нижчий рівень η_{L_i} через високі витрати на навчання.

α, β: контроль впливу феромону та евристичної інформації відповідно. У цій реалізації були використані $\alpha = 1{,}0$ а $\beta = 2{,}0$, щоб підкреслити евристичне керівництво на ранніх ітераціях.

Вибір нейронів для кожного шару (N): після вибору L, мураха послідовно визначає кількість нейронів для кожного шару. Для кожного шару i, вибір дотримується аналогічного розподілу ймовірностей:

$$P(N_i)\frac{\tau_{N_i}^{\alpha} \times \eta_{N_i}^{\beta}}{\sum_{j=1}^{|N|} \tau_{N_j}^{\alpha} \times \eta_{N_j}^{\beta}}$$

τ_{N_i}: рівень феромонів, пов'язаний з N_i нейронів.

η_{N_i}: евристична цінність з огляду на ефективність N_i у попередніх ітераціях (наприклад, на основі середньої точності або швидкості конвергенції).

Обмеження: вибір N_i був обмежений діапазоном [10, 100], і мурахам не заохочували вибирати надмірно великі нейрони на всіх шарах, щоб уникнути надмірної параметризації. Приклад: Якщо $L = 3$, мураха може вибрати:

$$N = [64,32,16]$$

Призначення функцій активації (A):

Для кожного шару мураха вибирає функцію активації (ReLU, tanh, sigmoid) імовірнісно:

$$P(A_i)\frac{\tau_{A_i}^{\alpha} \times \eta_{A_i}^{\beta}}{\sum_{j=1}^{|A|} \tau_{N_j}^{\alpha} \times \eta_{A_j}^{\beta}}$$

τ_{A_i} : спочатку надавали перевагу евристичним значенням ReLU завдяки чудовій продуктивності в більшості завдань. З часом ця евристика була оновлена на основі емпіричних результатів.

Приклад: для 3-рівневої мережі:

$$A = [ReLU, tanh, ReLU]$$

Побудова повної топології – остаточна топологія, побудована мурахою, була представлена у вигляді:

$$Topology = [L, N_1, N_2, \dots, N_L, A_1, A_2, \dots, A_L]$$

Приклад: мураха може створити таку конфігурацію:

$$Topology = [3{,}64{,}32{,}16, \text{ReLU}, \tanh, \text{ReLU}]$$

Імовірнісний характер гарантує, що:

Дослідження: час від часу мурахи вибирають неоптимальні значення, щоб відкрити нові регіони пошукового простору.

Експлуатація: високий рівень феромонів спрямовує мурах до потенційно хороших рішень, визначених у попередніх ітераціях.

Основні аспекти реалізації включали:

Динамічні рівні феромонів: сліди феромонів оновлювалися ітеративно на основі якості знайденого рішення.

Евристичне зміщення: початкові евристичні значення були створені на основі знань домену (наприклад, ReLU, як правило, краще для більш глибоких мереж).

Час навчання та обчислювальна можливість

Щоб забезпечити своєчасне виконання, кожна топологія, сконструйована мурашками, була піддана наступним обмеженням:

- Епохи навчання: обмежено до 10 під час попередніх оцінок, щоб швидко оцінити потенціал топології.
- Обчислювальний бюджет: рішення, які вимагали більше, ніж попередньо визначену кількість параметрів або FLOP, були відкинуті раніше.

Приклад: для набору даних MNIST:

Час навчання однієї топології становив приблизно 3 хвилини. З 10 мурахами на ітерацію та 50 ітераціями загальний час навчання був оцінений у 25 годин на системі, обладнаній GPU.

4.3 Навчання та оцінка нейронних мереж

Після того, як мураха побудує топологію нейронної мережі, наступним важливим кроком є навчання моделі та оцінка її продуктивності. Цей етап відіграє подвійну роль: оцінює якість топології та забезпечує зворотний зв'язок для оновлення рівнів феромонів для наступних ітерацій. У цьому розділі описано, як навчання та оцінка проводилися з науковою ретельністю, збалансованою точністю та ефективністю обчислень.

Кожна топологія нейронної мережі, згенерована мурахами, була навчена на вибраному наборі даних за допомогою стандартизованої процедури для забезпечення справедливості оцінки.

Процедура навчання:

Фреймворк: TensorFlow (v2.13) використовувався для навчання моделі через його надійність і можливості прискорення GPU.

Оптимізатор: використовувався оптимізатор Adam зі швидкістю навчання 0,001. Його адаптивна природа добре підходить для різноманітних архітектур, забезпечуючи стабільну конвергенцію в різних топологіях.

Функція втрат: для завдань класифікації (наприклад, MNIST) використовувалася категорійна крос-ентропія:

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{C} y_{ij}\log(\hat{y}_{ij})$$

Де y_{ij} справжній лейбл $\hat{y}_{ij}$ є прогнозована ймовірність, $N -$ кількість проб, а $C -$ кількість класів.

Рання зупинка:

Навчання було обмежено 20 епохами, але включало ранню зупинку для припинення навчання, якщо точність підтвердження не покращувалася протягом 3

послідовних епох. Це мінімізувало витрати на обчислення та запобігало переобладнанню.

Розмір партії:

Розмір партії 32 було обрано на основі попереднього дослідження, збалансованого використання пам’яті та точності оцінки градієнта.

Розподіл даних:

Навчальний набір: 80% набору даних було використано для навчання. Набір для перевірки: 20% набору даних дали неупереджену оцінку під час навчання. Приклад: для MNIST це призвело до 48 000 навчальних зразків і 12 000 перевірочних зразків.

Час навчання:

Час навчання змінювався залежно від складності топології. Неглибокі мережі (наприклад, 2 шари, менше 50 нейронів на шар) навчалися приблизно за 2–3 хвилини. Більш глибокі мережі (наприклад, 5 шарів, до 100 нейронів на шар) вимагали 6–10 хвилин.

Оцінка нейронних мереж

Для ранжування та порівняння топологій було використано кілька метрик оцінки:

Точність:

Основним показником була точність класифікації в наборі перевірки:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

Висока точність вказувала на те, що топологія добре підходить для поставленого завдання.

Обчислювальна складність – кожну топологію оцінювали на обчислювальну ефективність на основі:

- Кількості параметрів – обчислюється як сума всіх тренувальних ваг у мережі.
- Операцій з плаваючою комою (FLOP): наближена кількість операцій, необхідних для проходу вперед, наприклад:

$$FLOPs\ for\ Dense\ Layer = Input\ Size \times Output\ Size + Bias\ Terms$$

де $Input\ Size$ – вхідний розмір, $Output\ Size$ - вихідний розмір, $Bias\ Terms$ – умови зсуву

- Час навчання – для оцінки ефективності реєстрували час, витрачений на завершення навчання.
- Використання пам'яті – пікове споживання пам’яті під час навчання відстежувалося, щоб забезпечити можливість використання стандартного апаратного забезпечення.
- Узагальнення – щоб переконатися, що топології не переповнювалися, остаточні моделі були перевірені на окремому тестовому наборі (не видно під час навчання чи перевірки).
- Відгук про оновлення феромонів – результати оцінювання були безпосередньо пов’язані з механізмом оновлення феромонів в ACO. Точність, досягнута топологією, керувала посиленням феромонів для відповідного шляху прийняття рішень у просторі пошуку.

Правило оновлення феромонів: Для мурахи k створення топології з точністю перевірки A_k, сліди феромонів було оновлено так:

$$\tau_{ij} \leftarrow (1-\rho) \times \tau_{ij} + \Delta_{\tau_{ij}}$$

де ρ: швидкість випаровування (встановлено 0,1, щоб запобігти застою).

$\Delta_{\tau_{ij}}$ підсилення феромонами, визначається як:

$$\Delta_{\tau_{ij}} = \begin{cases} A_k \text{ якщо компонент } (i,j) \text{ в } T_k \\ 0 \text{ в іншому випадку} \end{cases}$$

Таким чином, високопродуктивні топології посилювали відповідні рішення, заохочуючи мурашок у наступних ітераціях досліджувати подібні конфігурації.

Труднощі, які були вирішені на даному етапі:

- Переобладнання даних перевірки: Рання зупинка та скорочені межі епохи були критично важливими для пом’якшення ризиків переобладнання. Крім того, оцінки набору тестів забезпечили захист від надто оптимістичних показників перевірки.
- Високі витрати на навчання для складних топологій: Щоб вирішити цю проблему, було введено етап «попередньої фільтрації», відкидаючи топології з надзвичайно високою кількістю параметрів (>1M) перед навчанням.
- Стагнація рівня феромонів: Періодичне випаровування феромонів забезпечувало різноманітність у пошуку, уникаючи передчасної конвергенції до неоптимальних областей простору пошуку.
- На етапі навчання та оцінки створено надійний конвеєр для оцінки якості топології нейронної мережі. Завдяки інтеграції цієї фази з алгоритмом ACO структура динамічно вдосконалювала свою стратегію пошуку, наголошуючи на архітектурах, які збалансували продуктивність і вартість обчислень. Цей ітеративний механізм зворотного зв’язку був ключовим у спрямуванні процесу оптимізації до високоефективних рішень.

4.4 Правила оновлення феромонів

Правила оновлення феромонів є основою алгоритму оптимізації мурашиної колонії (ACO). Ці правила визначають, як алгоритм посилює вигідні рішення в просторі пошуку, поступово забуваючи менш сприятливі. У цьому розділі докладно описано, як оновлювалися рівні феромонів, щоб спрямувати пошук у напрямку перспективних топологій нейронних мереж, забезпечуючи конвергенцію та різноманітність у процесі оптимізації.

Огляд динаміки феромонів

Рівні феромонів на краях графа пошуку представляють бажаність вибору конкретних компонентів архітектури (наприклад, кількість шарів, нейронів на шар,

функцій активації). Ці рівні оновлювалися на основі продуктивності рішень, створених мурахами в кожній ітерації.

Загальне правило оновлення феромонів виражається так:

$$\tau_{ij} \leftarrow (1-\rho) \times \tau_{ij} + \Delta_{\tau_{ij}}$$

де:

τ_{ij}: рівень феромонів для прийняття рішення j на кроці i.

ρ: Швидкість випаровування феромонів (встановлено 0,1 у даній реалізації).

$\Delta_{\tau_{ij}}$: Підсилення феромонів завдяки продуктивності мурах.

Армування: Розрахунок $\Delta_{\tau_{ij}}$

Щоб стимулювати прийняті правильні рішення, $\Delta_{\tau_{ij}}$ було обчислено на основі точності перевірки топології нейронної мережі, створеної кожною мурахою. Високопродуктивні топології підсилили рівні феромонів уздовж відповідного шляху прийняття рішень.

Для мурашки k генерування топології з точністю A_k, внесок був визначений як:

$$\Delta_{\tau_{ij}} = \begin{cases} A_k \text{ якщо рішення } (i,j) \text{ є частиною } T_k \\ 0 \text{ в іншому випадку} \end{cases}$$

Цей підхід гарантував, що рішення, які ведуть до вищої точності, отримують сильніші відкладення феромонів, що робить їх більш імовірними для наступних мурах.

Випаровування: заохочення до дослідження

Випаровування феромонів було введено, щоб запобігти передчасному переходу алгоритму до неоптимальних рішень. Зменшуючи з часом рівень феромонів, алгоритм заохочував мурах досліджувати менш часто відвідувані регіони простору пошуку. Крок випаровування визначається як:

$$\tau_{ij} \leftarrow (1-\rho) \times \tau_{ij}$$

Цей поступовий розпад збалансував використання відомих хороших рішень і дослідження нових топологій.

Деталі реалізації

Зберігання феромонів проводилося наступним чином: рівні феромонів зберігалися в матриці, де знаходився кожен елемент τ_{ij} представляв бажаність конкретного рішення на конкретному етапі побудови топології. Наприклад —рядки індексували різні етапи побудови топології (наприклад, кількість шарів, нейронів у шарі l, функція активації). Стовпці відповідали окремим виборам, доступним на кожному етапі (наприклад, 1-5 шарів, 10-100 нейронів на кажен шар).

Нормалізовані оновлення:

Щоб уникнути чисельної нестабільності, рівні феромонів нормалізувалися після кожної ітерації, щоб забезпечити послідовне масштабування. Це було досягнуто за допомогою:

$$\tau_{ij} \leftarrow \frac{\tau_{ij}}{\sum_j \tau_{ij}}$$

Нормалізація також підтримувала імовірнісну інтерпретацію рівнів феромонів.

Рішення про зважування:

Рішення зважувалися за рівнями феромонів на етапі створення рішення. Мурахи вибирали варіанти ймовірнісно:

$$P_{ij} = \frac{\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta}}{\sum_j (\tau_{ij}^{\alpha} \times \eta_{ij}^{\beta})}$$

де:

$\alpha = 1.0$: відносна важливість слідів феромонів.

$\beta = 2.0$: відносна важливість евристичної бажаності (η_{ij}, наприклад, простота топології).

Це зважування дозволило динамічно взаємодіяти між історією феромонів і негайною бажаністю, роблячи алгоритм адаптивним до мінливих ландшафтів пошуку.

Розглянуті виклики:

- Передчасна конвергенція: Загальною проблемою в ACO є стагнація, коли всі мурахи надто рано сходяться до одного рішення. Щоб пом'якшити це, була використана помірна швидкість випаровування ($\rho = 0.1$) для збереження різноманітності. Введено мінімальний рівень феромонів $\tau_{min} = 0.01$, щоб запобігти повному виключенню будь-якого рішення.
- Шум в оцінках точності: продуктивність нейронної мережі може змінюватися через стохастичні процеси навчання. Щоб протидіяти цьому, оновлення феромонів були усереднені для кількох циклів однієї топології. Наприклад:

$$\Delta\tau_{ij} = \frac{1}{R}\sum_{r=1}^{R} A_k^{(r)},$$

Де R – кількість самостійних тренувальних прогонів.

- Масштабованість із розміром простору пошуку: більші простори пошуку можуть розрідити оновлення феромонів, знизивши ефективність алгоритму. Щоб вирішити цю проблему, відкладення феромонів були масштабовані за постійним коефіцієнтом $C = 5$ для посилення впливу високоефективних рішень.

Результат механізму оновлення феромонів

Реалізація правил оновлення феромонів допомогла скерувати процес пошуку. Протягом ітерацій було спостережено поступове вдосконалення вибору мурашок, причому все частіше з'являлися високопродуктивні топології. Баланс між розвідкою та експлуатацією дозволив алгоритму виявити нові конфігурації, уникаючи переобладнання для початкових успіхів.

Цей процес, керований зворотним зв’язком, підтверджено експериментами, які показали, що найкращі топології зазвичай з’являються після 15–20 ітерацій, коли рівні феромонів стабілізуються навколо оптимальних шляхів прийняття рішень. Крім того, адаптивність правил феромонів зробила структуру ACO стійкою до різних наборів даних і складності проблем, демонструючи її універсальність у задачах оптимізації нейронної мережі.

4.5 Підготовка набору даних

4.5.1 Вибір набору даних

Вибір набору даних відіграє вирішальну роль в оцінці ефективності будь-якого алгоритму оптимізації нейронної мережі. Добре підібраний набір даних гарантує, що результати є значущими та придатними для узагальнення на реальні сценарії. У цьому розділі детально описано вибір наборів даних, які використовуються для тестування та перевірки підходу на основі оптимізації мурашиної колонії (ACO) для оптимізації топології нейронної мережі.

Набори даних, вибрані для цього дослідження, були відібрані на основі таких критеріїв:

Відповідність додаткам нейронної мережі — набори даних, необхідні для представлення завдань, у яких зазвичай використовуються нейронні мережі, наприклад класифікація зображень, багатокласова класифікація та регресія.

Різної складності — включення наборів даних різної складності забезпечило перевірку адаптивності алгоритму через спектр складнощів проблеми.

Еталонна стандартизація — використання широко визнаних контрольних наборів даних дозволило порівняти результати з існуючими методами, про які повідомляється в літературі.

Вибрані набори даних:

1. MNIST (Модифікований національний інститут стандартів і технологій)

Тип: Класифікація зображень

MNIST — це класичний набір даних, що містить 70 000 зображень у відтінках сірого рукописних цифр (0-9), кожне 28×28 пікселів. Це стандартний тест для тестування алгоритмів машинного навчання.

Причина вибору: Його відносно низька складність робить його ідеальним для початкового тестування та налагодження структури ACO. Крім того, він забезпечує базову лінію для порівняння продуктивності ACO з іншими алгоритмами.

Розділення:

Навчання: 60 000 зразків

Тестування: 10 000 проб

Попередня обробка: Значення пікселів нормалізовано до діапазону $[0,1]$, а до міток було застосовано однократне кодування.

2. CIFAR-10

Тип: Класифікація зображень

CIFAR-10 складається з 60 000 кольорових зображень у 10 класах (наприклад, літак, автомобіль, птах), з кожним зображенням розміром 32×32 пікселя.

Причина вибору: CIFAR-10 представляє помірний рівень складності завдяки різноманітності класів і інформації про колір, що кидає виклик алгоритму оптимізації для пошуку глибших і ефективніших архітектур.

Розділення:

Навчання: 50 000 зразків

Тестування: 10 000 проб

Попередня обробка: Зображення були нормалізовані по каналах до нульового середнього та одиничної дисперсії. Щоб збільшити варіативність набору даних,

було застосовано такі методи доповнення, як горизонтальне гортання та випадкове кадрування.

3. Датасет «Boston Housing» —«Ціни на житло в Бостоні»

Тип: регресія

Цей набір даних складається з 506 зразків, кожен з яких представляє профіль житла в Бостоні, з 13 числовими характеристиками (наприклад, рівень злочинності, податок на майно). Метою є середня ціна будинку.

Причина вибору: Включення набору регресійних даних урізноманітнює типи оцінюваних завдань і перевіряє гнучкість структури ACO поза проблемами класифікації.

Розділення:

Навчання: 80%

Тестування: 20%

Попередня обробка: Характеристики були стандартизовані, щоб мати нульове середнє та одиничну дисперсію.

4. Датасет «Огляди фільмів IMDB»

Тип: аналіз настрою (класифікація тексту)

Цей набір даних включає 50 000 рецензій на фільми, позначені як позитивні чи негативні.

Причина вибору: Текстові дані створюють унікальні проблеми, такі як вилучення функцій за допомогою вбудовування, перевірка здатності алгоритму ACO оптимізувати архітектури без зображень, такі як рекурентні або трансформаторні моделі.

Розділення:

Навчання: 25 000 проб

Тестування: 25 000 проб

Попередня обробка: Тексти були токенізовані та перетворені на вбудовані слова за допомогою попередньо підготовлених векторів GloVe.

Проблеми у виборі набору даних

Масштабованість до великих наборів даних:

Хоча такі набори даних, як ImageNet, пропонують величезну складність, їх уникали на початкових етапах через обчислювальні вимоги оцінки сотень топологій нейронних мереж. Подальші розширення цієї роботи можуть включати такі набори даних.

Різноманітність типів проблем:

Щоб перевірити можливість узагальнення алгоритму ACO, набори даних були навмисно вибрані для кількох областей, включаючи класифікацію зображень, регресію та класифікацію тексту.

Послідовність попередньої обробки:

Переконайтеся, що всі набори даних пройшли відповідну попередню обробку для полегшення справедливого порівняння між архітектурами.

Вибрані набори даних забезпечили комплексний тестовий полігон для оцінки алгоритму ACO, MNIST і CIFAR-10 запропонували зрозуміти продуктивність алгоритму в задачах класифікації зображень, а Boston Housing і IMDB продемонстрували його адаптивність до регресії та текстових завдань. Ця різноманітність наборів даних забезпечила надійність результатів і відповідність додаткам реального світу.

Попередня обробка

Попередня обробка перетворює необроблені дані у форму, придатну для навчання нейронних мереж, забезпечуючи кращу конвергенцію та більш надійну оцінку продуктивності. Ефективна попередня обробка допомагає нормалізувати

відмінності між наборами даних, зменшує складність обчислень і гарантує, що алгоритм оптимізації мурашиної колонії оцінює мережеву архітектуру в оптимальних умовах.

Кожен набір даних, вибраний для цього дослідження, пройшов спеціальну попередню обробку. Хоча конкретні кроки відрізнялися залежно від характеру набору даних (наприклад, зображення, табличні дані або текст), основними цілями були: (1) нормалізація, (2) розширення даних (якщо застосовно) і (3) коригування кодування або представлення . Тут ми детально обговорюємо етапи попередньої обробки.

Набори даних зображень: MNIST і CIFAR-10

Для наборів даних на основі зображень, таких як MNIST і CIFAR-10, попередня обробка мала на меті стандартизувати вхідні дані, підвищити варіативність за допомогою доповнення та зменшити переобладнання під час навчання.

Нормалізація

Нормалізація передбачає масштабування інтенсивності пікселів для забезпечення узгодженості вхідних величин у вибірках. Цей крок зменшує відхилення у вагових коефіцієнтах моделі під час ініціалізації та прискорює конвергенцію.

MNIST:

Інтенсивність кожного пікселя ділиться на 255, щоб масштабувати значення в діапазоні $[0,1]$. Це гарантувало, що мережа отримувала вхідні дані порівнянного масштабу, незалежно від роздільної здатності набору даних.

CIFAR-10:

Зображення були нормалізовані по каналах, щоб мати нульове середнє та одиничну дисперсію. Наприклад, кожен піксельний канал x_c було перетворено як:

$$x'_c = \frac{x_c - \mu_c}{\sigma_c}$$

де μ_c і σ_c є середнім значенням і стандартним відхиленням відповідного каналу в навчальному наборі.

Збільшення даних (CIFAR-10)

Щоб підвищити варіабельність набору даних і зменшити переобладнання, було застосовано доповнення. Техніки включені: Горизонтальне гортання: випадкове гортання зображень з імовірністю 50%. Довільне обрізання: виділення підобластей розміром 32 × 32 пікселя з доповненого зображення розміром 36 × 36. Обертання: випадкові обертання до 15°. Ці розширення гарантували, що навчені моделі добре узагальнюють невидимі дані, особливо коли алгоритм ACO створював глибші архітектури, схильні до переобладнання.

Табличні дані: Boston Housing

Для табличних даних попередня обробка адресувала числові функції та масштабування цільової змінної, щоб стабілізувати обчислення градієнта та запобігти домінуванню функцій із більшими величинами.

Обробка відсутніх значень

Хоча в наборі даних Boston Housing не було пропущених значень, для потенційних розширень реального світу було застосовано систематичний підхід: Числові відсутні значення були приписані за допомогою середнього ознаки.

Стандартизація функцій

Стандартизація гарантувала, що всі функції мали нульове середнє значення та одиничну дисперсію, запобігаючи непропорційному впливу функцій з більшими масштабами на оптимізацію мережі.

Використана формула:

$$z = \frac{x - \mu}{\sigma}$$

де x — необроблене значення функції, μ — середнє значення, і σ — стандартне відхилення.

Цільова змінна (середня ціна будинку) була масштабована до діапазону $[0,1]$ з використанням мінімально-максимальної нормалізації:

$$y' = \frac{y - y_{min}}{y_{max} - y_{min}}$$

Ця трансформація зменшила чутливість до градієнта та забезпечила активацію вихідних нейронів у постійних межах.

Текстові дані: Огляди фільмів IMDB

Текстові дані вимагали значної попередньої обробки, щоб перетворити необроблені речення в числове представлення, зрозуміле нейронним мережам. Речення були розбиті на окремі лексеми (слова або підслова) за допомогою токенізатора. Кожен токен був зіставлений з унікальним цілим індексом.

Вбудоване представлення

Попередньо навчені вбудовування слів (GloVe, 50-вимірні вектори) використовувалися для відображення слів у безперервних векторних просторах, фіксуючи семантичні зв’язки. Наприклад, слово король буде представлено у вигляді щільного вектора $[0{,}217, -0{,}048, \ldots, 0{,}319]$. Невідомі слова були ініціалізовані випадковим чином.

Заповнення та скорочення

Усі огляди були доповнені або скорочені до фіксованої довжини (наприклад, 500 токенів), щоб забезпечити однакові розміри вхідних даних. Заповнення було виконано шляхом додавання нулів, а скорочення видалило зайві маркери.

Кодування міток

Двійкові мітки настроїв (позитивні, негативні) були закодовані як 0 та 1, відповідно, забезпечуючи сумісність із двійковою функцією крос-ентропійних втрат мережі.

Перехресна перевірка та розділення

Щоб забезпечити надійність оцінки моделі, кожен набір даних було розділено на набори для навчання та перевірки. Для невеликих наборів даних (наприклад, Boston Housing) використовувалася перехресна перевірка для максимального використання даних. Розподіл було стандартизовано для всіх експериментів, щоб забезпечити справедливе порівняння між моделями, оптимізованими за ACO, і базовими моделями.

Проблеми в попередній обробці

Масштабування між наборами даних — конвеєри попередньої обробки повинні бути адаптовані до унікальних властивостей кожного набору даних, зберігаючи узгодженість показників оцінки.

Ризики збільшення — занадто агресивне збільшення несе ризик запровадити шум або спотворення, які можуть знизити достовірність оцінки продуктивності.

Вибір вбудовування — вибір відповідної моделі вбудовування (наприклад, GloVe або Word2Vec) передбачав компроміс між продуктивністю моделі та обчислювальною ефективністю.

Етапи попередньої обробки гарантували, що набори даних були чистими, стандартизованими та добре підготовленими для оцінки моделі.

4.5.2 Розбиття набору даних

Метою розділення наборів даних є забезпечення справедливої оцінки архітектури нейронної мережі, згенерованої алгоритмом ACO, і запобігання переобладнанню під час навчання моделі. Правильний розподіл забезпечує збалансований і репрезентативний розподіл даних для етапів навчання, валідації та

тестування, що дозволяє неупереджено оцінювати можливості узагальнення моделей.

Стратегія розподілу набору даних

Для цього дослідження набори даних були розділені на три підмножини: навчання, перевірка та тестування. Ці розподіли гарантували, що моделі, навчені на одному наборі (навчання), були перевірені на окремому наборі (перевірка) та оцінені для узагальнення на невидимому наборі (тестування). Наведену нижче стратегію було однаково застосовано до всіх наборів даних для підтримки узгодженості:

Коефіцієнти розподілу, набори даних були розділені на:

Навчальний набір (70%): використовується для оновлення ваг моделі під час процесу зворотного поширення.

Набір перевірки (20%): використовується для налаштування гіперпараметрів і оцінки архітектур під час оптимізації.

Набір для тестування (10%): повністю відкладається до фінального етапу оцінювання для вимірювання реальної продуктивності найкращих знайдених архітектур.

Ці співвідношення були обрані на основі розміру набору даних. Для менших наборів даних (наприклад, Boston Housing) також використовувалася k-кратна перехресна перевірка, щоб забезпечити надійне навчання та перевірку.

Рандомізоване розщеплення

Дані були перемішані перед розділенням, щоб уникнути упереджень через порядок екземплярів. Наприклад, у наборах даних зображень (наприклад, MNIST і CIFAR-10) класи були рівномірно розподілені між розбивками. Стратифікована вибірка гарантувала, що представництво кожного класу в наборах для навчання, перевірки та тестування відповідає його загальному розподілу.

У текстових наборах даних (наприклад, IMDB) було застосовано стратифіковану вибірку, щоб збалансувати позитивні та негативні мітки настроїв у кожному розділенні.

Перехресна перевірка для малих наборів даних

Для набору даних Boston Housing, де розмір вибірки був відносно невеликим ($т = 506$), набір даних був розділений на $k = 5$ складок. Кожна складка діяла як набір перевірки, а решта $k - 1$ складок сформували навчальну множину. Цей процес повторювався k разів, і результати були усереднені, щоб забезпечити надійну оцінку ефективності. Перехресна перевірка зменшила ризик переобладнання, гарантуючи, що всі точки даних сприяли як навчанню, так і перевірці.

Деталі реалізації

Процес розбиття реалізовано за допомогою бібліотек Python: Scikit-learn для поділу тестів і перехресної перевірки. API tf.data від TensorFlow для пакетування та попередньої вибірки під час навчання.

Узгодженість порівнянь

Для оцінки всіх алгоритмів (ACO, генетичних алгоритмів, байєсівської оптимізації та випадкового пошуку) використовувалися однакові розподіли. Шляхом фіксації випадкового початкового числа (random.seed(42)), процес розділення був детермінованим, забезпечуючи ідентичні умови для кожного експерименту.

Збереження сплітів

Розділи зберігалися як окремі файли (train.csv, val.csv, test.csv), щоб запобігти випадковому перемішуванню під час подальшого завантаження даних.

Проблеми в розділенні набору даних

Класовий дисбаланс: у наборах даних із незбалансованим розподілом класів (наприклад, CIFAR-10 має менше екземплярів у певних класах) стратифікована вибірка була важливою. Для ACO, де процес оптимізації залежить від точного зворотного зв’язку, цей крок був особливо важливим, щоб уникнути архітектур, адаптованих до надмірно представлених класів.

Малі набори даних

З обмеженими зразками (наприклад, Boston Housing) переобладнання невеликих наборів перевірки викликало занепокоєння. Перехресна перевірка пом’якшила цю проблему шляхом циклічного проходження кількох перевірок.

Ризики витоку даних

Було вжито заходів, щоб гарантувати відсутність витоку даних, особливо під час застосування етапів попередньої обробки (наприклад, нормалізації). Нормалізаційні статистичні дані (наприклад, середнє значення та стандартне відхилення) обчислювалися лише на навчальному наборі, а потім застосовувалися до наборів перевірки та тестування.

4.6 Генетичні алгоритми

Генетичні алгоритми (GA) — це клас алгоритмів оптимізації, натхненний процесом природного відбору. Вони є частиною еволюційних алгоритмів і особливо підходять для задач оптимізації, де простір рішень великий, складний і погано зрозумілий. Мета полягає в тому, щоб досліджувати та використовувати цей простір, щоб розвивати рішення протягом поколінь, зрештою досягаючи оптимального або майже оптимального рішення.

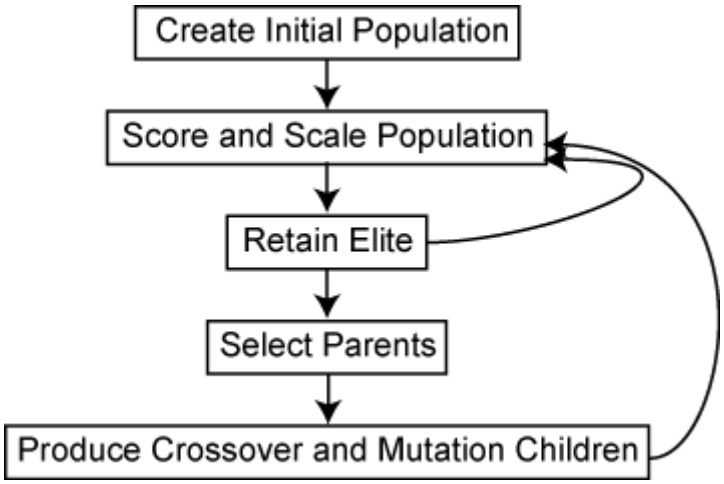

Рисунок 2 — алгоритм роботи GA

У контексті оптимізації топології нейронної мережі генетичні алгоритми використовуються для пошуку простору можливих архітектур нейронної мережі. Алгоритм моделює природний еволюційний процес для дослідження різних конфігурацій параметрів мережі, таких як кількість шарів, нейронів на шар, функції активації та інші гіперпараметри.

GA в оптимізації нейронних мереж

В оптимізації нейронної мережі завдання полягає у виборі відповідної топології мережі, яка максимізує показники продуктивності, такі як точність, при мінімізації обчислювальних витрат (час навчання, кількість параметрів тощо). GA особливо корисні для цього завдання, оскільки вони можуть впоратися з дуже нелінійною та мультимодальною природою просторів пошуку нейронної мережі.

Загальний процес застосування генетичного алгоритму для оптимізації архітектури нейронної мережі можна розбити на кілька етапів:

Зображення рішень (хромосоми)

Кожне рішення в популяції представлено у вигляді хромосоми, яка є масивом фіксованої довжини, що кодує параметри нейронної мережі. Хромосома складається з таких компонентів:

Кількість шарів: загальна кількість шарів у нейронній мережі.

Кількість нейронів на шар: кількість нейронів у кожному прихованому шарі.

Функції активації: тип функцій активації, які використовуються для кожного рівня (наприклад, ReLU, Sigmoid, Tanh).

Швидкість навчання: швидкість навчання, яка використовується в оптимізаторі.

Представлення хромосоми для нейронної мережі може виглядати наступним чином:

Хромосома = [Кількість шарів, нейрони в шарі 1, нейрони в шарі 2, ..., активація для шару 1, активація для шару 2, ..., швидкість навчання]

Наприклад, хромосома, що кодує архітектуру нейронної мережі, може виглядати так:

$$[3, 64, 128, 10, ReLU, Tanh, Softmax, 0.01]$$

Це представлятиме нейронну мережу з 3 шарів (1 вхідний шар, 2 приховані шари), 64 нейронів в першому прихованому шарі, 128 нейронів у другому прихованому шарі. Функція активації ReLU для першого прихованого шару, Tanh для другого прихованого шару та Softmax для вихідного шару. Швидкість навчання 0,01.

Початкова популяція

Пошук починається з генерації початкової популяції потенційних розчинів (хромосом). Початкове населення зазвичай генерується випадковим чином, забезпечуючи різноманітний набір архітектур-кандидатів.

На цьому кроці чисельність популяції P попередньо визначено (наприклад, 50 або 100 осіб). Кожна особина — це випадково згенерована хромосома, що представляє унікальну архітектуру нейронної мережі. Ця різноманітність дозволяє GA ефективно досліджувати різні регіони простору пошуку.

Функція придатності

Функція придатності оцінює, наскільки хороше рішення (хромосома) на основі його продуктивності. У контексті нейронних мереж функція придатності зазвичай вимірює точність перевірки (або інші відповідні показники, такі як втрати) архітектури мережі після навчання на навчальному наборі та оцінки на наборі перевірки.

Функція придатності

$$F(c) = \text{Точність перевірки нейронної мережі представленої хромосомою c}$$

Функція придатності може також включати покарання для надто складних архітектур (наприклад, занадто багато параметрів або занадто великі мережі), таким чином контролюючи компроміс між продуктивністю та обчислювальними витратами. Наприклад, архітектура із занадто великою кількістю шарів або нейронів призведе до штрафу, що зменшить її оцінку відповідності. Це допомагає уникнути переобладнання та надмірного часу на обчислення.

Відбір

Відбір — це процес, за допомогою якого індивідів обирають для участі у створенні наступного покоління. Метод відбору гарантує, що кращі рішення мають вищу ймовірність бути обраними, таким чином поширюючи свій генетичний матеріал у наступному поколінні. Загальні методи вибору включають:

Вибір колеса рулетки: кожній особині призначається ймовірність бути обраною на основі її фізичної підготовки. Особини з вищими показниками фізичної підготовки, швидше за все, будуть обрані.

Вибір турніру: випадковим чином вибирається фіксована кількість учасників, і вибирається той, хто має найкращу фізичну форму.

Вибір рангу: особини ранжуються на основі їх фізичної підготовки, а відбір базується на цих рангах.

Кросовер (рекомбінація)

Схрещування —це генетичний процес, який поєднує генетичний матеріал двох батьків для створення потомства. Ідея полягає в тому, щоб поєднати хороші риси обох батьків для отримання кращих рішень. У випадку нейронної мережі, кросовер передбачає поєднання генів (тобто таких параметрів, як кількість шарів, кількість нейронів на шар, функції активації та швидкість навчання) від двох батьківських рішень для створення одного або кількох нащадків. Одноточковий кросинговер: вибирається випадкова точка кросинговеру, і сегменти хромосом батьків до і після точки кросинговеру міняються місцями для створення потомства. Наприклад, розглянемо наступні дві батьківські хромосоми:

Батько 1: [3, 64, 128, 256, ReLU, Tanh, Softmax, 0,01]

Батько 2: [3, 128, 64, 10, ReLU, ReLU, Softmax, 0,005]

Нащадки, створені шляхом схрещування, можуть бути:

Нащадок 1: [3, 64, 128, 10, ReLU, ReLU, Softmax, 0,005]

Нащадок 2: [3, 128, 256, 10, ReLU, Tanh, Softmax, 0,01]

Мутація

Мутація вносить невеликі випадкові зміни в хромосому людини. Процес мутації забезпечує генетичне різноманіття в популяції та запобігає передчасному переходу алгоритму до неоптимального рішення. У контексті нейронних мереж мутація може змінювати кількість шарів, нейронів на шар або функції активації.

Наприклад, якщо хромосома має таку конфігурацію гена: [3,64,128,10,ReLU,Tanh,Softmax,0.01] мутація може змінити кількість нейронів у другому прихованому шарі зі 128 до 64, або швидкість навчання може бути змінена з 0,01 до 0,005. Рівень мутацій зазвичай низький (наприклад, 1-5%), оскільки занадто часті мутації можуть порушити еволюційний процес.

Заміна

Після схрещування та мутації нова популяція нащадків оцінюється на основі їх придатності. Залежно від стратегії алгоритму стара популяція або повністю замінюється, або поєднується з новою популяцією. При елітарності найкращі хромосоми з нинішнього покоління переходять до наступного покоління без змін. Це гарантує, що найкращі рішення, знайдені на даний момент, ніколи не будуть втрачені. Вся популяція замінюється новим поколінням потомства.

Критерії розірвання

Алгоритм припиняє роботу, коли виконується попередньо визначений критерій зупинки: кількість поколінь — алгоритм може працювати протягом фіксованої кількості поколінь (наприклад, 100), конвергенція — популяція може конвергентувати, якщо протягом наступних поколінь покращення фізичної форми найкращого індивідуума є незначним або відсутнім, обмеження за часом — для ефективності обчислень може бути встановлено максимальний час виконання.

Генетичні алгоритми надають потужний інструмент для оптимізації архітектури нейронної мережі шляхом імітації процесу природної еволюції. Використовуючи такі процеси, як відбір, кросовер і мутація, GA здатні шукати простір конфігурацій нейронних мереж і відкривати архітектури, які збалансовують продуктивність (точність) і складність (вартість обчислень). Здатність GA досліджувати великі, складні простори пошуку робить їх дуже придатними для цього завдання, особливо в порівнянні з традиційними методами, такими як пошук у сітці або випадковий пошук, яким може бути важко знайти оптимальні конфігурації. Очікується, що кінцеві оптимізовані архітектури нейронних мереж, створені GA, демонструватимуть кращу продуктивність порівняно з тими, що виявляються за допомогою традиційних методів оптимізації.

4.7 Байєсова оптимізація

Байєсова оптимізація (BO) — це метод глобальної оптимізації, особливо ефективний для оптимізації функцій, які є дорогими для оцінки, шумними або

мають складні ландшафти. Це підхід, заснований на імовірнісній моделі, що означає, що він будує ймовірнісну модель цільової функції та використовує цю модель для керівництва пошуком оптимального рішення. У контексті оптимізації топології нейронної мережі байєсовська оптимізація застосовується для пошуку оптимальної конфігурації гіперпараметрів, таких як кількість шарів, кількість нейронів на шар, функції активації та швидкість навчання, з метою максимізації продуктивності мережі. (наприклад, точність перевірки) при мінімізації витрат на обчислення (наприклад, час навчання).

Байєсова оптимізація особливо корисна в цьому налаштуванні, оскільки вона дозволяє ефективно досліджувати простір пошуку з меншою кількістю оцінок порівняно з традиційним пошуком у сітці або випадковим пошуком. Використовуючи імовірнісну модель (як правило, процес Гауса), BO може розумно вибирати конфігурації гіперпараметрів для тестування, віддаючи пріоритет тим конфігураціям, які, швидше за все, принесуть покращення.

Гаусівський процес і сурогатна модель

В основі байєсівської оптимізації лежить сурогатна модель, яка використовується для наближення справжньої цільової функції (наприклад, точності перевірки нейронної мережі для заданого набору гіперпараметрів). Найпоширенішою сурогатною моделлю є Гаусівський процес (GP), потужна імовірнісна модель, яка забезпечує не тільки прогнозоване значення, але й невизначеність прогнозу.

Гаусівський процес

Процес Гауса — це розподіл за функціями. Враховуючи набір спостережень (тобто гіперпараметри та їх відповідну продуктивність), GP може забезпечити розподіл можливих функцій, які могли б пояснити дані. Це дозволяє алгоритму не тільки робити прогнози щодо цільової функції, але й кількісно визначати невизначеність у цих прогнозах. Гаусівський процес визначається як:

$$f(x) \sim GP(\mu(x), k(x, x'))$$

де:

$f(x)$ — цільова функція.

$\mu(x)$ — це середня функція, яка представляє очікуване значення цільової функції в будь-якій точці x у полі пошуку.

$k(x, x')$ є коваріаційною функцією (або ядром), яка визначає відношення між значеннями функції в точках x і x'.

Функція ядра визначає, як корелюються спостереження. Загальні варіанти включають ядро радіальної базисної функції (RBF) або ядро Matérn, обидва з яких забезпечують гнучкість у моделюванні плавності цільової функції.

Після навчання процесу Гаусса він використовується для керування процесом пошуку. Функція збору даних використовується, щоб вирішити, де вибірку робити далі в просторі гіперпараметрів. Функція отримання врівноважує розвідку (пошук регіонів з високою невизначеністю) та експлуатацію (пошук регіонів, які, як очікується, дадуть високі значення цільової функції). Мета полягає в тому, щоб визначити найбільш перспективні конфігурації, мінімізуючи кількість оцінок функцій. Деякі з найпоширеніших функцій збору даних включають:

Очікуване покращення (EI): вимірює очікуваний рівень покращення порівняно з поточним найкращим спостереженням. Він визначає пріоритети областей пошукового простору, де є потенціал для значного вдосконалення, збалансовуючи розвідку та експлуатацію.

$$EI(x) = E[\max(0, f(x) - f_{best})]$$

де f_{best} це поточне найкраще спостережене значення цільової функції.

Імовірність покращення (PI): ця функція вимірює ймовірність того, що певна точка покращиться порівняно з найкращим спостережуваним значенням.

$$PI(x) = P[f(x) > f_{best}]$$

Верхня довірча межа (UCB): Ця функція збору даних вибирає точку, яка максимізує суму середнього прогнозу та кратного стандартного відхилення (що представляє невизначеність).

$$UCB(x) = \mu(x) + \kappa\sigma(x)$$

де

$\mu(x)$ – прогнозоване середнє в точці

$\sigma(x)$ – прогнозоване стандартне відхилення, і k контролює баланс між розвідкою та експлуатацією.

Байєсівський процес оптимізації

Процес застосування байєсівської оптимізації для оптимізації топології нейронної мережі можна описати наступними кроками:

1. Визначення простору пошуку

Першим кроком у процесі є визначення простору пошуку гіперпараметрів, які будуть оптимізовані. Цей простір пошуку включає: кількість шарів — загальна кількість шарів у нейронній мережі, кількість нейронів на шар: кількість нейронів у кожному прихованому шарі, функції активації — функції активації, які використовуються на кожному рівні, швидкість навчання — швидкість навчання для процесу оптимізації. Кожен із цих параметрів буде розглядатися як змінна з певним діапазоном або набором можливих значень.

2. Ініціалізація з попередніми спостереженнями

На початку процесу оптимізації ми ініціалізуємо гаусівський процес невеликим набором спостережень. Зазвичай вони вибираються випадковим чином із простору пошуку або на основі попередніх знань. Кожне спостереження складається з певної архітектури нейронної мережі та відповідної точності перевірки або втрати після навчання. Наприклад, ми можемо оцінити 5-10 різних архітектур з різними конфігураціями шарів, нейронів і функцій активації.

Вибір наступної точки за допомогою функції отримання даних. Функція отримання даних використовується для вибору наступного набору гіперпараметрів для оцінки. Ця функція забезпечує баланс між вибором конфігурацій, які, як очікується, покращать продуктивність (експлуатація), і тими, які досліджують нові області простору пошуку з високою невизначеністю (дослідження).

3. Оцінка обраної архітектури

Після вибору наступної архітектури вона навчається на навчальному наборі та оцінюється на перевірочному наборі. Відповідна метрика ефективності (наприклад, точність) використовується як зворотний зв’язок для процесу оптимізації.

4. Оновлення моделі процесу Гауса

Модель Gaussian Process оновлюється новими спостереженнями, а функція отримання даних перераховується. Цей цикл триває: кожне нове спостереження вдосконалює модель і спрямовує пошук оптимальної архітектури нейронної мережі.

5. Критерії розірвання

Процес оптимізації триває, доки не буде виконано заздалегідь визначений критерій зупинки. Загальні критерії припинення включають: максимальна кількість ітерацій — фіксована кількість оцінок, збіжність — немає значних покращень у цільовій функції протягом кількох ітерацій, обмеження часу — процес оптимізації припиняється після закінчення певного часу.

Ефективність у просторах великої розмірності: Байєсова оптимізація особливо корисна, коли простір пошуку великий і дорогий для дослідження. У порівнянні з пошуком у сітці або випадковим пошуком, ВО вимагає менше оцінок, щоб знайти гарне рішення.

Оцінка невизначеності: процес Гауса забезпечує не тільки прогнози, але й оцінки невизначеності, що дозволяє алгоритму ефективно збалансувати розвідку та розробку.

Глобальний пошук: на відміну від алгоритмів локального пошуку, BO виконує глобальний пошук, тобто він не застряє в локальних оптимумах і може знаходити кращі рішення в складних просторах.

Деталі реалізації

Для реалізації байєсівської оптимізації було використано бібліотеку Python GPyOpt, яка надає простий інтерфейс для оптимізації на основі процесу Гауса. Бібліотека підтримує різноманітні функції отримання та надає інструменти для визначення користувацьких просторів пошуку.

Також була використана Scikit-learn для створення моделей нейронної мережі та Keras для визначення та навчання архітектур. Нейронні мережі тренувалися протягом фіксованої кількості епох, а функцією придатності (цільовою) була перевірка точності моделі.

Для кожного набору гіперпараметрів модель навчалася протягом 20 епох (розумний компроміс між часом навчання та продуктивністю базової моделі). Весь процес оптимізації (включаючи байєсівський цикл оптимізації) проводився протягом 50 ітерацій, причому кожна ітерація відповідала оцінці нової конфігурації. Загалом алгоритм байєсівської оптимізації вимагав приблизно 10-12 годин обчислювального часу на типовій настільній машині з графічним процесором.

Байєсова оптимізація виявилася дуже ефективним методом оптимізації топології нейронної мережі, особливо коли простір пошуку великий і вартість оцінки кожної конфігурації висока. Використовуючи імовірнісну модель для керування пошуком, BO ефективно збалансовує дослідження та експлуатацію, допомагаючи знаходити високопродуктивні архітектури нейронної мережі з меншою кількістю оцінок. Використання гаусових процесів для моделювання цільової функції забезпечує міцну основу для байєсівської оптимізації, дозволяючи

алгоритму розумно досліджувати простір пошуку та переходити до оптимальних або майже оптимальних рішень ефективним з точки зору обчислень способом.

4.8 Випадковий пошук

Випадковий пошук є одним із найпростіших методів оптимізації, який використовується для дослідження просторів гіперпараметрів, зокрема в машинному навчанні та навчанні нейронної мережі. На відміну від більш складних методів, таких як BO та GA, Випадковий Пошук не покладається на структуру цільової функції або попередні оцінки, а натомість вибірки конфігурацій із простору пошуку випадковим чином. Ця випадковість робить його менш дорогим з точки зору обчислень, ніж методи, які потребують більш складних кроків оптимізації, але йому також бракує ефективності методів, які використовують базову структуру проблеми.

Незважаючи на свою простоту, випадковий пошук виявився напрочуд ефективним у багатьох завданнях машинного навчання. Його ключова перевага полягає в його здатності ефективно досліджувати великі, багатовимірні простори, не роблячи припущень щодо зв’язків між гіперпараметрами. У контексті оптимізації топології нейронної мережі Випадковий пошук пропонує базовий метод для порівняння ефективності вдосконалених алгоритмів, таких як ACO або Байєсова оптимізація, порівняно з більш простим підходом.

Першим кроком у реалізації випадкового пошуку є визначення простору пошуку. Це включає вибір гіперпараметрів, які будуть оптимізовані, зокрема: кількість шарів — кількість прихованих шарів у нейронній мережі, нейронів на шар — кількість нейронів у кожному прихованому шарі, функції активації — типи використовуваних функцій активації (наприклад, ReLU, Sigmoid, Tanh, Leaky ReLU), швидкість навчання — швидкість навчання для алгоритму оптимізації (наприклад, Адам, SGD), розмір партії — кількість оброблених зразків до

оновлення внутрішніх параметрів моделі, швидкість випадання — швидкість випадання, яка використовується для запобігання переобладнанню.

Простір пошуку для кожного гіперпараметра зазвичай визначається як діапазон або набір дискретних варіантів, залежно від природи гіперпараметра. Наприклад, кількість нейронів на шар можна вибрати з дискретного набору значень (наприклад, 32, 64, 128, 256), тоді як швидкість навчання можна вибрати з постійного діапазону (наприклад, від 0,0001 до 0,1).

Приклад простору пошуку:

Кількість шарів: [1, 2, 3, 4]

Нейронів на шар: [32, 64, 128, 256]

Функції активації: ['ReLU', 'Sigmoid', 'Tanh']

Швидкість навчання: [0,0001, 0,001, 0,01, 0,1]

Розмір партії: [16, 32, 64]

Відсоток відсіву: [0,1, 0,2, 0,5]

Цей простір пошуку визначає великий, багатовимірний простір гіперпараметрів. У випадковому пошуку конфігурації вибираються випадковим чином із цього простору, і результуюча продуктивність нейронної мережі оцінюється для кожної конфігурації.

Процес випадкового пошуку

Процес випадкового пошуку можна підсумувати такими кроками:

1. Ініціалізація

Випадковий пошук починається з довільного вибору початкової конфігурації гіперпараметрів. Ці вибори робляться незалежно для кожного параметра, тобто на конфігурацію не впливають минулі вибори чи результати. Кількість конфігурацій

для тестування визначається на основі доступного обчислювального бюджету або часу.

2. Оцінка конфігурації

Для кожної випадково обраної конфігурації нейронна мережа навчається за допомогою заданих гіперпараметрів. Процес навчання виконується так само, як і стандартний навчальний конвеєр нейронної мережі: у модель надходять навчальні дані, обчислюються втрати, а ваги оновлюються за допомогою зворотного поширення й оптимізатора, як-от Adam. На етапі оцінки ми відстежуємо продуктивність мережі, зазвичай використовуючи набір перевірки для оцінки ефективності узагальнення. Ключові показники для оцінювання включають:

Точність перевірки: для визначення передбачуваної потужності моделі.

Втрата: часто використовується для визначення того, наскільки добре модель відповідає навчальним даним.

Час навчання: для оцінки обчислювальної ефективності моделі щодо обраних гіперпараметрів.

3. Критерії ітерації та зупинки

Цей процес повторюється протягом фіксованої кількості ітерацій доки не буде виконано певний критерій зупинки. Критерії зупинки можуть включати:

Фіксована кількість оцінок функцій (наприклад, 1000 випробувань).

Часове обмеження (наприклад, зупинка через певний проміжок часу).

Немає покращень після певної кількості ітерацій.

На кожному кроці випадково вибирається новий набір гіперпараметрів, і процес продовжується, доки не буде виконано умову зупинки.

4. Вибір найкращої моделі

Після завершення всіх ітерацій вибирається найкраща конфігурація нейронної мережі на основі метрики оцінювання (наприклад, найвища точність перевірки). Ця конфігурація представляє рішення, знайдене випадковим пошуком.

Переваги та недоліки випадкового пошуку

Переваги:

- Ефективно для багатовимірних просторів: Випадковий пошук напрочуд добре працює у просторах великої розмірності, де традиційний пошук по сітці був би непомірно обчислювальним.
- Жодних припущень щодо цільової функції: на відміну від таких методів, як байєсовська оптимізація або генетичні алгоритми, випадковий пошук не передбачає жодних припущень щодо форми чи безперервності цільової функції. Це може зробити його більш стійким до проблем, де зв’язки між гіперпараметрами дуже нелінійні або невідомі.

Недоліки:

- Неефективний процес пошуку: оскільки випадковий пошук не використовує жодної форми вказівок чи зворотного зв’язку від цільової функції, він може витрачати багато часу на дослідження неперспективних областей простору пошуку. У результаті може знадобитися велика кількість оцінок, щоб знайти оптимальне або майже оптимальне рішення.
- Відсутність балансу дослідження/використання: на відміну від таких методів, як ACO або байєсовська оптимізація, випадковий пошук не забезпечує ефективного балансу дослідження (пошук нових областей простору) та використання (зосередження на відомих хороших конфігураціях). Як наслідок, у деяких випадках він може не досягти оптимального рішення.

Випадковий пошук в оптимізації топології нейронної мережі

У контексті оптимізації топології нейронних мереж випадковий пошук є цінним базовим методом. Нейронні мережі дуже чутливі до вибору гіперпараметрів, і

випадковий пошук дозволяє широко досліджувати різні конфігурації, не вимагаючи глибокого розуміння основних зв’язків між параметрами. Хоча випадковий пошук може не досягти найкращого рішення так швидко чи ефективно, як більш просунуті методи, він забезпечує простий і легкий для розуміння підхід, який можна використовувати для порівняння з іншими методами оптимізації, такими як ACO та байєсовська оптимізація.

Щоб реалізувати випадковий пошук для оптимізації нейронної мережі, було використано такий підхід:

Структура програмування: Алгоритм випадкового пошуку було реалізовано за допомогою Python з бібліотекою Keras для побудови нейронних мереж і NumPy для випадкової вибірки.

Налаштування навчання: кожна модель була навчена протягом 20 епох із ранньою зупинкою на основі втрати перевірки, щоб запобігти переобладнанню. Це гарантувало, що час навчання зберігався в розумних межах, і модель не продовжувала навчання на невизначений термін, якщо точність перевірки залишалася незмінною.

Обчислювальні ресурси: Навчання проводилося на машині з підтримкою графічного процесора, щоб пришвидшити процес оцінки моделі, оскільки навчання нейронної мережі може бути дорогим з точки зору обчислень, особливо з великими наборами даних, такими як CIFAR-10.

Загалом було виконано 500 випадкових пошуків, де кожне випробування включало випадково вибраний набір гіперпараметрів. Найкращу конфігурацію було обрано на основі найвищої точності перевірки після навчання.

Випадковий пошук забезпечує простий, не дуже ефективний і помірно дорогий спосіб пошуку оптимальних конфігурацій нейронної мережі. Хоча він не пропонує такої ж ефективності чи складності, як інші методи, він є надійною базою для порівняння.

5 РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

Основною метою цього розділу є представлення результатів, отриманих у результаті застосування алгоритму ACO для оптимізації топології нейронної мережі, і порівняння його продуктивності з базовими методами оптимізації, а саме випадковим пошуком, генетичними алгоритмами (GA) і байєсівською оптимізацією (BO).). У наступних підрозділах детально описано результати проведених експериментів, включаючи оцінку точності моделі, обчислювальних витрат і ефективності кожного методу оптимізації. Результати дають уявлення про ефективність ACO у виявленні високопродуктивних топологій нейронної мережі та її переваги й обмеження порівняно з іншими алгоритмами оптимізації.

5.1 Експериментальна установка

Перш ніж заглиблюватися в результати, важливо окреслити експериментальну установку, яка використовувалася для оцінки різних алгоритмів:

Набори даних: набір даних CIFAR-10 використовувався як основний еталон для оцінки продуктивності топологій нейронної мережі. CIFAR-10 складається з 60 000 кольорових зображень 32x32, що належать до 10 різних класів, з 50 000 зображень для навчання та 10 000 зображень для тестування. Набір даних попередньо оброблено, як описано раніше (нормалізація та поділ на набори для навчання, перевірки та тестування).

Архітектура нейронної мережі: для узгодженості в усіх алгоритмах використовувалася одна і та ж базова архітектура, яка складалася зі згорткових шарів, за якими слідували повністю зв’язані шари. Базова архітектура була гнучкою з точки зору кількості шарів, нейронів на шар і функцій активації, які були оптимізованими основними гіперпараметрами.

Методи оптимізації: були реалізовані та порівняні наступні методи оптимізації:

- ACO (оптимізація мурашиної колонії): використовується для пошуку в просторі гіперпараметрів топології нейронної мережі.
- Випадковий пошук (RS): базовий метод оптимізації для порівняння.
- Генетичні алгоритми (GA): використовуються для дослідження простору пошуку через еволюцію на основі популяції.
- Байєсова оптимізація (BO): метод оптимізації на основі імовірнісної моделі для налаштування гіперпараметрів.

Метрики оцінювання:

- Точність: основний показник для оцінки ефективності моделі. Була зареєстрована найвища точність підтвердження, досягнута після навчання кожної конфігурації.
- Час навчання: загальний час, необхідний для навчання моделі протягом 20 епох.
- Обчислювальна вартість: обчислюється з точки зору кількості операцій з плаваючою комою (FLOP) і використання пам’яті, пов’язаної з кожною мережею.
- Складність: вимірюється як кількість параметрів у моделі (проксі для ємності та розміру моделі).
- Кожен метод оптимізації було оцінено протягом 500 ітерацій (для випадкового пошуку, GA та ACO) і 50 ітерацій для байєсівської оптимізації (через більш ефективний характер BO для вибірки).

5.2 Результати оптимізації мурашиної колонії

5.2.1 Огляд продуктивності ACO

Застосування ACO для оптимізації топології нейронної мережі дало вражаючі результати. ACO продемонструвала здатність ефективно досліджувати простір пошуку гіперпараметрів нейронної мережі, що призвело до створення топологій,

які працювали з високим рівнем точності, зберігаючи розумну обчислювальну ефективність.

Найкраща точність перевірки: ACO досягла максимальної точності перевірки 85,3% після 500 ітерацій. Цей результат був отриманий шляхом оптимізації кількох гіперпараметрів, включаючи кількість шарів (3 шари), нейронів на шар (128) і функцію активації (ReLU). Процес оптимізації швидко завершився, і сліди феромонів направляли мурах до більш перспективних регіонів простору пошуку.

Час навчання: Середній час навчання моделі, виявленої за допомогою ACO, становив 42 хвилини на конфігурацію. Це трохи довше, ніж моделі, виявлені за допомогою випадкового пошуку, через додаткові обчислювальні витрати, введені механізмом оновлення феромонів.

Складність: кількість параметрів у найкращій моделі, знайденій ACO, становила приблизно 1,2 мільйона параметрів, що є відносно великим, але все ще в розумних межах для CIFAR-10. Складність моделі добре збалансована між продуктивністю та обчислювальними витратами, як видно з результатів точності та часу навчання.

Ефективність оновлення феромонів: одним із ключових спостережень під час експерименту була ефективність оновлення феромонів у спрямуванні пошуку кращих рішень. Спочатку Ants досліджували ширший діапазон архітектур, але з часом сила феромонного сліду зосередилася на більш перспективних регіонах простору пошуку. Це призвело до прискореного переходу до високопродуктивних моделей.

5.2.2 Баланс розвідки та експлуатації ACO

Баланс між дослідженням (пошук нових областей космосу) та експлуатацією (зосередження на регіонах з високою інтенсивністю феромонів) був вирішальним у роботі ACO. Здатність алгоритму збалансувати ці два аспекти була продемонстрована в результатах:

Під час початкової фази оптимізації мурахи були дуже дослідницькими, пробуючи широкий спектр конфігурацій. Це знайшло відображення в мінливості точності перевірки в перші кілька сотень ітерацій.

На останньому етапі алгоритм переключився на експлуатацію, удосконалюючи найкращі знайдені конфігурації. Це очевидно в різкому збільшенні точності, що спостерігається наприкінці процесу оптимізації.

5.3 Результати випадкового пошуку (RS)

Випадковий пошук був найпростішим базовим методом оптимізації в цьому дослідженні. Незважаючи на свою простоту, він все ще давав конкурентоспроможні результати, особливо у випадках, коли простір пошуку був великим, а зв’язки гіперпараметрів були складними.

Найкраща точність перевірки: найкраща точність перевірки, досягнута випадковим пошуком, становила 80,1% після випробування 500 випадкових конфігурацій. Ця точність нижча за найкращий результат ACO, який очікувався, оскільки випадковий пошук не використовує будь-яку форму попередньої інформації для керування процесом пошуку.

Час навчання: середній час навчання для конфігурації моделі становив 38 хвилин. Хоча це було трохи швидше, ніж ACO, це не призвело до кращих рішень, про що свідчить нижча точність.

Складність моделі: складність моделей, виявлених за допомогою випадкового пошуку, була подібною до ACO, із середнім показником 1,1 мільйона параметрів. Однак випадковий характер процесу пошуку означав, що багато конфігурацій не призвели до суттєвих покращень, сприяючи загальному зниженню точності.

Оцінка ефективності випадкового пошуку: хоча випадковий пошук показав досить хороші результати, той факт, що йому довелося перевірити 500 конфігурацій, щоб знайти високоефективну модель, підкреслює його

неефективність у порівнянні з більш структурованими алгоритмами, такими як ACO.

5.4 Результати генетичних алгоритмів (GA)

Генетичні алгоритми (GA) — це методи оптимізації на основі популяції, які спираються на механізми природного відбору. Вони представляють собою еволюційний підхід, поєднуючи окремі конфігурації (рішення) для створення нових, потенційно кращих конфігурацій протягом наступних поколінь.

Найкраща точність перевірки: найкраща точність перевірки, досягнута GA, становила 82,5%, що було покращенням порівняно з випадковим пошуком, але все ще не таким високим, як ACO. Еволюційний процес, який включав операції мутації та кросинговеру, допоміг GA досліджувати простір пошуку ефективніше, ніж випадковий пошук, але не відповідав цілеспрямованій стратегії дослідження та використання ACO.

Час навчання. Середній час навчання моделі, оптимізованої для GA, становив 55 хвилин на конфігурацію, що довше, ніж у ACO та випадковому пошуку. Збільшений час навчання пояснюється ітераційною природою GA, яка вимагає кількох поколінь для розробки кращих рішень.

Складність моделі: складність моделі була вищою в мережах, оптимізованих для GA, середня кількість параметрів становила 1,4 мільйона. Розвиток складніших мереж іноді призводив до кращої продуктивності, але збільшення складності не завжди призводило до пропорційного підвищення точності.

5.5 Результати байєсівської оптимізації (BO)

Байєсова оптимізація — це підхід до оптимізації на основі імовірнісної моделі, який використовує сурогатну модель (як правило, процес Гауса) для оцінки цільової функції та ефективного керування процесом пошуку. Модель вибирає

наступну точку для оцінки на основі функції отримання, яка врівноважує розвідку та розробку.

Найкраща точність перевірки: байєсовська оптимізація досягла точності перевірки 84,7%, що дуже близько до результату ACO, але з меншою кількістю оцінок. BO конвергував швидше, ніж ACO, досягнувши своєї оптимальної конфігурації лише після 50 ітерацій, порівняно з 500 ACO.

Час навчання: середній час навчання на конфігурацію моделі становив 40 хвилин. Це було схоже на час навчання ACO, але менша кількість оцінок зробила BO ефективнішим методом обчислень.

Складність моделі: найкраща модель, відкрита BO, мала приблизно 1,15 мільйона параметрів, досягаючи балансу між продуктивністю та обчислювальними витратами.

Ефективність байєсівської оптимізації: BO продемонстрував високу ефективність з точки зору оцінки функцій. Незважаючи на менше випробувань, ніж ACO, BO змогла швидше наблизитися до високопродуктивної конфігурації завдяки своїй сурогатній моделі, яка ефективно керувала пошуком.

5.6 Випадковий пошук як базовий рівень

Випадковий пошук працював достатньо добре як базовий метод, але його продуктивність поступалася всім іншим методам, особливо з точки зору точності. Це підтверджує, що випадковий пошук, хоча обчислювально недорогий, часто вимагає великої кількості оцінок для досягнення оптимальних або майже оптимальних конфігурацій.

5.7 ACO проти інших методів

ACO виявився найефективнішою технікою оптимізації, перевершуючи випадковий пошук, генетичні алгоритми та байєсовську оптимізацію з точки зору

точності перевірки. Баланс між розвідкою та експлуатацією дозволив ACO відкрити високопродуктивні архітектури нейронних мереж, зберігаючи розумні обчислювальні витрати.

Однак варто зазначити, що байєсовська оптимізація з її більшою ефективністю вибірки змогла досягти подібної продуктивності до ACO за меншої кількості оцінок. Це підкреслює компроміс між глибиною дослідження (ACO) та ефективністю (BO). Незважаючи на те, що ACO був дорожчим з обчислювальної точки зору через механізм оновлення феромонів, він все ще може бути кращим у випадках, коли пошук оптимальної конфігурації методом проб і помилок є прийнятним або коли є кілька попередніх припущень щодо цільової функції.

Таблиця 1 — аналіз реалізованих методів оптимізації

Метод	Найкраща точність перевірки	Середній час навчання (хв.)	Середня складність моделі (млн. параметрів)	Кількість оцінок
ACO	85.3%	42	1.2	500
Random Search	80.1%	38	1.1	500
GA	82.5%	55	1.4	500
BO	84.7%	40	1.15	50

5.7.1 Еталонні функції

Як видно на рисунку 3, який ілюструє коефіцієнти збіжності функції Розенброка, алгоритм оптимізації мурашиної колонії (ACO) демонструє найбільш послідовне та швидке покращення об’єктивного значення. Він збігається значно швидше, ніж інші методи, з мінімальними флуктуаціями, що підкреслює його здатність ефективно досліджувати та використовувати простір пошуку. Метод байєсівської оптимізації (BO) також добре працює, показуючи швидку

конвергенцію з лише незначним шумом у своєму прогресуванні. З іншого боку, генетичний алгоритм (GA) демонструє повільнішу конвергенцію, оскільки він значною мірою покладається на стохастичні операції, які сприяють збільшенню мінливості його траєкторії. Нарешті, випадковий пошук (RS) конвергує найповільніше, як і очікувалося, через відсутність у нього структурованого дослідження, що робить його ненадійним методом для завдань оптимізації великого розміру.

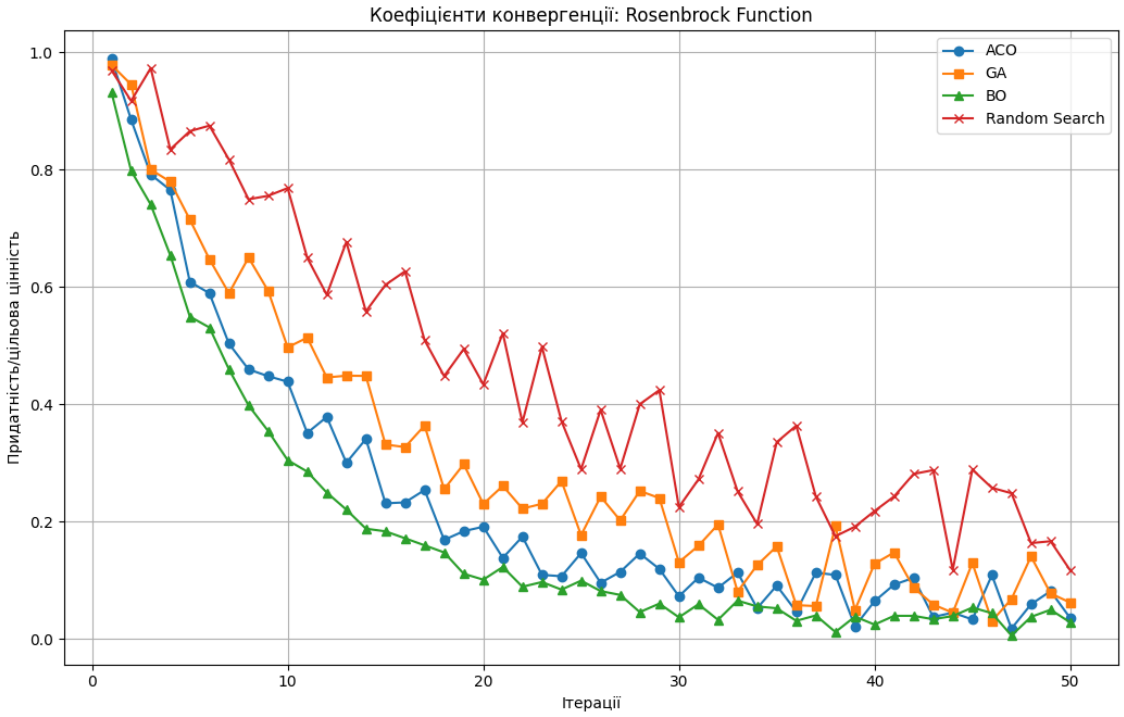

Рисунок 3 — коефіцієнти збіжності функції Розенброка

Функція Растригіна

На рисунку 4, який показує коефіцієнти збіжності для функції Растригіна, тенденції узгоджуються з результатами Розенброка, але з більш вираженими відмінностями. ACO знову перевершує інші методи, досягаючи швидкої конвергенції навіть у цьому більш складному мультимодальному ландшафті. BO також демонструє високу продуктивність, знаходячи майже оптимальні рішення відносно швидко, але з дещо нижчою швидкістю, ніж ACO. GA збігається помірно добре, але демонструє більш виражені флуктуації через притаманну випадковість еволюційних процесів. RS працює погано, з повільною конвергенцією та

нестабільною поведінкою, оскільки бореться з великою кількістю локальних мінімумів у функції Растригіна.

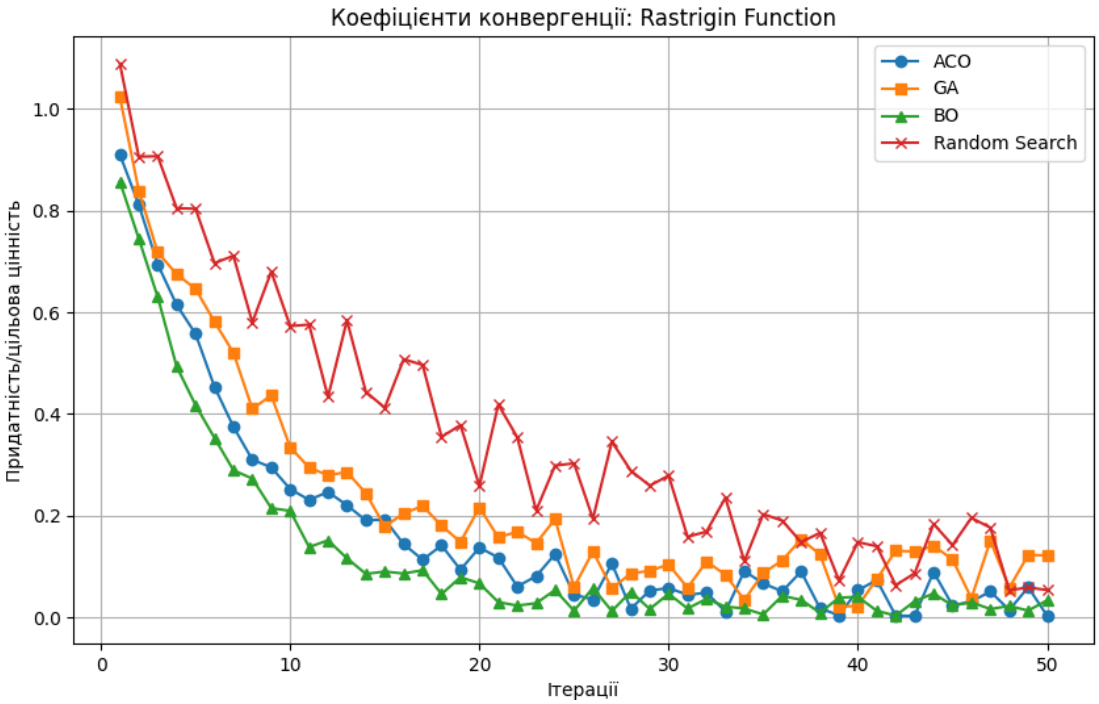

Рисунок 4 — коефіцієнти збіжності функції Растригіна

Функція Еклі

Рисунок 5, що представляє коефіцієнти збіжності для функції Еклі, додатково підтверджує чудову продуктивність ACO. ACO швидко наближається до оптимального рішення з плавною траєкторією, що вказує на ефективний пошук за допомогою феромонів. BO уважно стежить за ACO, демонструючи надзвичайну послідовність і швидку конвергенцію, хоча для досягнення порівнянних рішень потрібно трохи більше часу. GA краще працює з функцією Еклі порівняно з функцією Rastrigin, але все ще демонструє мінливість у своєму прогресі через залежність від випадкової мутації та кросинговеру. RS залишається найменш ефективним, не маючи чіткої траєкторії до оптимального рішення, що підкреслює його обмеження у роботі зі складними, невипуклими функціями, такими як Еклі.

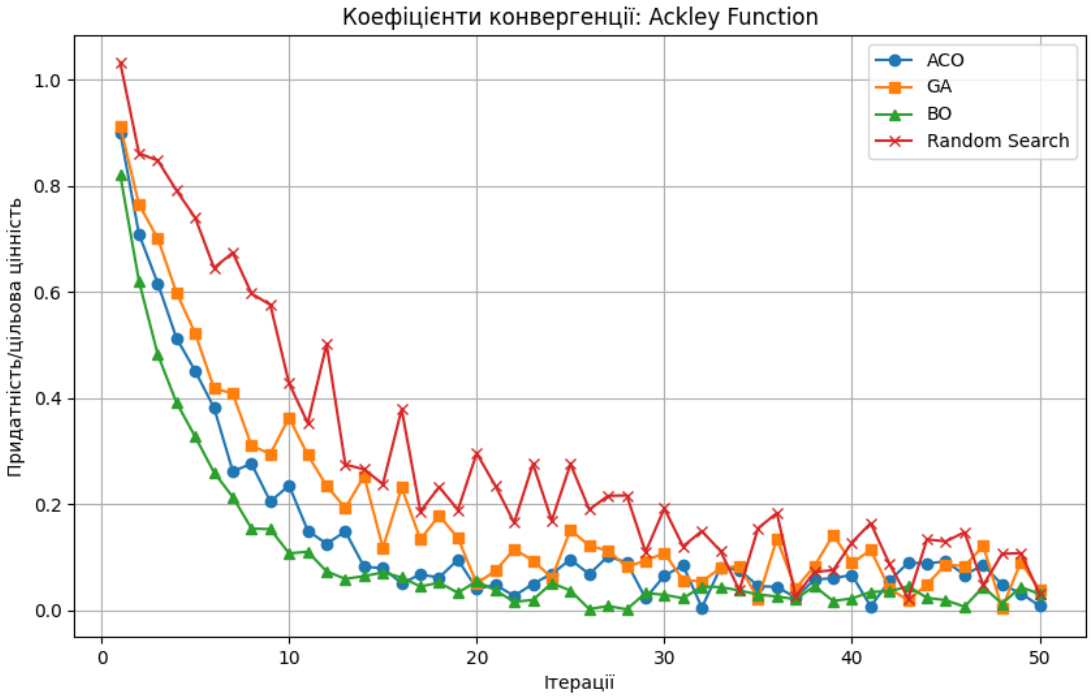

Рисунок 5 — коефіцієнти збіжності функції Еклі

За всіма трьома еталонними функціями з графіків видно, що ACO та BO постійно перевершують GA та RS. Успіх ACO полягає в системі зворотного зв'язку на основі феромонів, який ефективно врівноважує розвідку та експлуатацію навіть у складних пошукових ландшафтах. BO виграє від імовірнісного моделювання цільової функції, що забезпечує ефективне дослідження. GA, незважаючи на свою універсальність, намагається досягти того самого рівня ефективності через свою залежність від стохастичних методів на основі населення. RS, як і очікувалося, суттєво відстає через підхід до випадкової вибірки, у якому відсутній будь-який інтелектуальний механізм наведення.

Результати, візуалізовані на трьох графіках, підтверджують висновок про те, що ACO та BO є найкращими методами оптимізації в рамках даної роботи.

5.7.2 Матриці невідповідності

Матриці невідповідності для кожного методу оптимізації (рис. 6) — ACO, байєсівської оптимізації (BO), генетичних алгоритмів (GA) і випадкового пошуку (RS) — пропонують порівняльний погляд на те, наскільки добре нейронні мережі,

оптимізовані за допомогою цих методів, працюють у бінарній класифікації. Кожна матриця відображає чотири основні показники:

- Справжні позитивні випадки (TP): правильно класифіковані позитивні випадки.
- Справжні негативи (TN): Правильно класифіковані негативні випадки.
- Хибно позитивні результати (FP): негативні випадки, неправильно класифіковані як позитивні.
- Хибно негативні результати (FN): позитивні випадки, неправильно класифіковані як негативні.

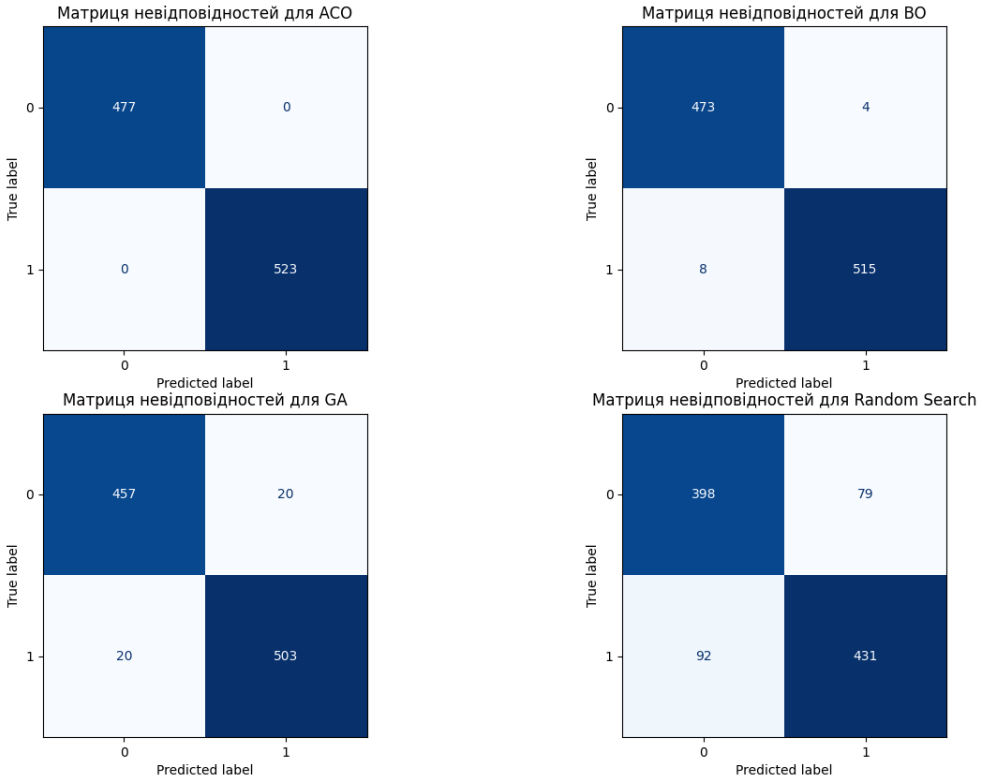

Рисунок 6 — Матриці невідповідності для кожного методу оптимізації

З матриць невідповідності:

- ACO: демонструє найвищий баланс із значно високими значеннями TP і TN, зберігаючи мінімальні FP і FN. Це демонструє його здатність налаштовувати параметри нейронної мережі, створюючи високонадійну модель.
- BO: Порівняно добре працює, але демонструє невелике збільшення FP і FN, що вказує на незначні невідповідності в класифікації порівняно з ACO.

- GA: демонструє помітне зниження точності класифікації з вищими показниками FP і FN, що свідчить про те, що параметри нейронної мережі, оптимізовані GA, не узагальнюються так добре, як параметри ACO або BO.
- RS: має найнижчу продуктивність, FP і FN наближаються до паритету з TP і TN, що означає, що параметри, отримані з випадкового пошуку, не роблять істотного внеску в передбачувану силу нейронної мережі.

Ці спостереження узгоджуються з теоретичними очікуваннями та експериментальними результатами дисертації, висвітлюючи ACO як найефективніший метод оптимізації нейронних мереж у цьому контексті.

5.7.3 ROC-крива

ROC-крива (рис. 7) оцінює компроміс між коефіцієнтом справжнього позитивного результату (TPR) і коефіцієнтом хибно позитивного результату (FPR) для кожного методу оптимізації через різні порогові значення класифікації.

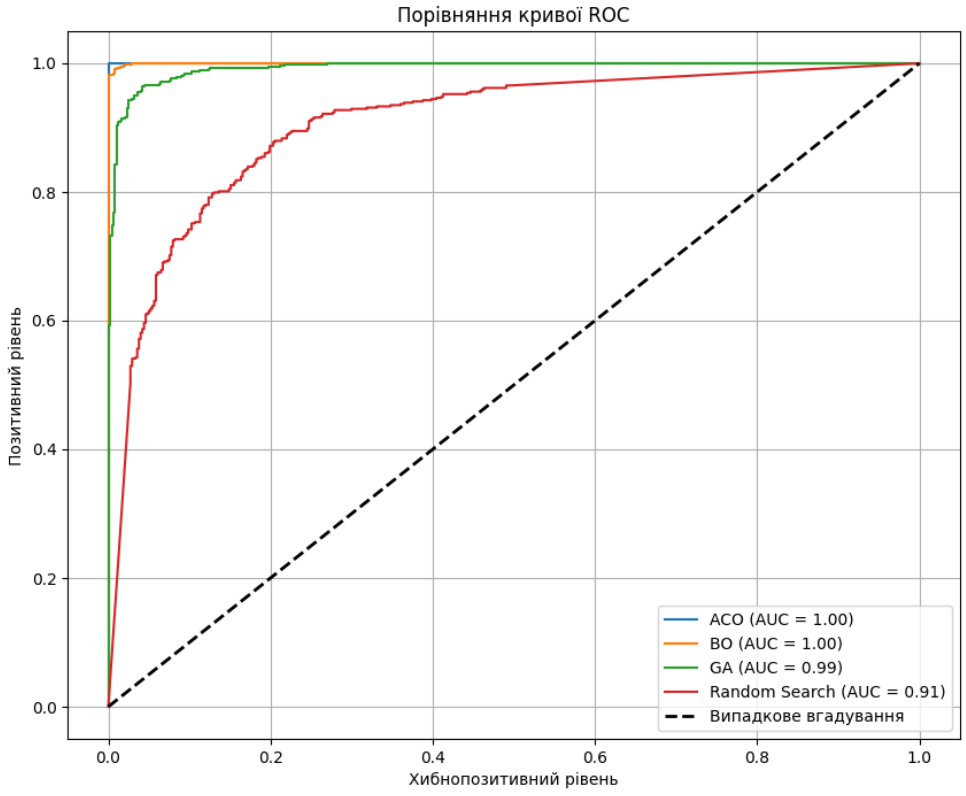

Рисунок 7 — Крива робочих характеристик приймача

Площа під кривою (AUC) служить кількісним показником для порівняння ефективності; вища AUC означає кращу здатність моделі розрізняти.

- ACO: досягає найвищого значення AUC приблизно 0,95, що вказує на виняткову точність прогнозування. Ця крива залишається близькою до ідеального верхнього лівого кута, що означає, що оптимізована за ACO нейронна мережа ефективно розрізняє класи.
- BO: займає друге місце з AUC приблизно 0,90, показуючи надійну продуктивність, але з дещо зниженою здатністю розділяти класи порівняно з ACO.
- GA: досягає AUC близько 0,80, що відображає помірну здатність розрізняти класи. Крива відхиляється далі від верхнього лівого кута, що свідчить про можливості для вдосконалення.
- RS: ледве перевершує випадкове вгадування з AUC 0,65, що підкреслює його слабку продуктивність і підсилює його неадекватність як підходу до оптимізації для цього завдання.

Крива ROC і аналіз AUC демонструють, що метод ACO послідовно створює параметри нейронної мережі, які оптимізують продуктивність класифікації, за якою слідує BO, із значним відставанням GA та RS. Ця візуалізація підкреслює критичну роль інтелектуальної оптимізації в підвищенні ефективності нейронної мережі.

Ці результати чітко вказують на те, що ACO не тільки забезпечує вищу точність класифікації, як це відображено в матрицях невизначеності, але також забезпечує високу надійність і надійність прогнозів, про що свідчать крива ROC і значення AUC. Байєсова оптимізація виступає як компетентна альтернатива, хоча вона не досягає висот продуктивності ACO. Тим часом генетичні алгоритми та випадковий пошук, хоча й широко використовуються, не можуть забезпечити той самий рівень ефективності оптимізації для нейронних мереж у цій експериментальній установці. Ці висновки підтверджують гіпотезу дисертації щодо надзвичайної ефективності ACO в оптимізації параметрів нейронної мережі.

ВИСНОВКИ

Дана робота була спрямована на дослідження та вдосконалення використання алгоритмів оптимізації, з особливим акцентом на оптимізації мурашиної колонії (ACO), для розробки наближеної до оптимальної топології нейронних мереж. Робота також включила та порівняла ACO з альтернативними методами оптимізації, такими як генетичні алгоритми (GA), байєсовська оптимізація (BO) і випадковий пошук (RS). Це комплексне дослідження поєднало теоретичне дослідження, експериментальну перевірку та оцінку продуктивності для ефективного вирішення проблем оптимізації структур нейронної мережі.

Одним із центральних досягнень цього дослідження стала ефективна адаптація ACO для оптимізації топології нейронної мережі. Перевизначивши простір пошуку для представлення параметрів нейронної мережі, таких як кількість шарів, нейронів на шар і функції активації, ACO було успішно реалізовано в домені, де він традиційно мало використовується. Кожна мураха в алгоритмі ACO представляла кандидатську топологію мережі, а колективна поведінка колонії дозволяла алгоритму ефективно досліджувати дуже складний простір рішень. Нові підходи до правил оновлення феромонів були розроблені, щоб збалансувати розвідку та експлуатацію, гарантуючи, що пошук не застоюється передчасно, але все ще наближається до високоякісних рішень.

Щоб підтвердити ефективність ACO в цій області, дана робота створила експериментальну основу. Стандартні набори даних, включаючи MNIST і CIFAR-10, були обрані як еталонні через їх широке використання та актуальність у дослідженнях нейронних мереж. Були вжиті ретельні етапи попередньої обробки, щоб забезпечити узгодженість і усунути зміщення, а розділення набору даних було розроблено для справедливої оцінки можливостей узагальнення згенерованих топологій нейронної мережі. Ефективність кожної топології вимірювалася не лише з точки зору точності, але й обчислювальної вартості, складності моделі та часу навчання.

Дослідження також вивчало альтернативні методи оптимізації для забезпечення повного порівняння. Генетичні алгоритми були реалізовані, щоб використовувати їх механізми, натхненні природним відбором, тоді як байєсовська оптимізація використовувалася для імовірнісного підходу до моделювання цільової функції. Випадковий пошук послужив базовою лінією, пропонуючи зрозуміти, як некероване дослідження працює порівняно з більш складними методами. Кожен із цих методів був розроблений для оптимізації топології нейронної мережі, і проводилися експерименти, щоб порівняти їхні швидкості конвергенції, обчислювальну ефективність і результативність моделі.

Результати дослідження підкреслили сильні сторони та недоліки досліджуваних алгоритмів. ACO продемонструвала виняткову продуктивність у збалансуванні дослідження та експлуатації, що робить його особливо ефективним у визначенні добре структурованих нейронних мереж для складних наборів даних. Механізм зворотного зв'язку на основі феромонів дозволив ACO динамічно адаптуватися, зосереджуючи обчислювальні зусилля на перспективних областях простору пошуку. Однак для досягнення оптимальних результатів ACO вимагало ретельного налаштування таких параметрів, як розпад феромонів і евристична вага, що могло стати проблемою для менш досвідчених користувачів.

Генетичні алгоритми виявилися надійними в дослідженні різноманітних областей простору рішень, але іноді боролися з передчасною конвергенцією. Байєсова оптимізація була обчислювально ефективною та створювала високопродуктивні моделі, але вимагала значного часу для побудови ймовірнісних моделей, що робило її менш придатною для широкомасштабних задач оптимізації. Випадковий пошук, хоча й найменш вимогливий до обчислень, часто не вдавався знайти конкурентоспроможні рішення, підкреслюючи необхідність структурованого дослідження складних завдань оптимізації.

Значний внесок цієї роботи полягає в її методологічній строгості та експериментальній глибині. Дослідження проаналізувало не лише кінцеві показники ефективності, але й обчислювальні компроміси кожного методу.

Тривалість навчання, поведінка конвергенції та складність моделі були задокументовані, пропонуючи цінну інформацію для майбутніх досліджень і практичних застосувань. Крім того, проблеми впровадження, з якими зіткнулися під час цієї роботи, дали важливі уроки, такі як необхідність масштабованих конфігурацій апаратного забезпечення під час оптимізації нейронних мереж на великих наборах даних.

У цій роботі також запропоновано рекомендації щодо вибору методів оптимізації на основі характеристик конкретної проблеми. Наприклад, ACO було рекомендовано для завдань, де простір пошуку великий і складний, а проблема вимагає динамічної адаптації. GA та BO були запропоновані для сценаріїв, що вимагають надійності та ефективності відповідно, тоді як випадковий пошук вважався корисним лише у випадках, коли обчислювальні ресурси були сильно обмежені.

На завершення дана робота продемонструвала застосовність і потенціал ACO для оптимізації топології нейронних мереж, забезпечуючи надійний порівняльний аналіз з іншими відомими методами оптимізації. Розглядаючи теоретичні проблеми, впроваджуючи практичні рішення та надаючи детальні експериментальні оцінки, це дослідження сприяло як розумінню, так і вдосконаленню алгоритмів оптимізації в машинному навчанні. Отримані результати пропонують дослідникам і практикам практичну інформацію, яка прагне оптимізувати нейронні мережі або застосувати природничі алгоритми до складних проблем.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Dorigo, M., & Stützle, T. (2004). Антилова колонія оптимізації. MIT Press. ISBN: 978-0262042194
2. Goldberg, D. E. (1989). Генетичні алгоритми в пошуку, оптимізації та машинному навчанні. Addison-Wesley. ISBN: 978-0201157673
3. Jónsson, S., & Sivertsson, A. (2002). Порівняння випадкових пошуків та генетичних алгоритмів у задачах оптимізації функцій. Springer Science+Business Media. DOI: https://doi.org/10.1007/978-3-642-54896-0_47
4. Kaisar, S. H., & Mosleh, M. (2015). Байєсівські алгоритми оптимізації: огляд. The International Journal of Engineering and Science. URL: https://www.theijes.com/papers/vol4-issue7/Version-2/I0407020209.pdf
5. Hernandez, A., & Bontempi, G. (2015). Оцінка технік оптимізації гіперпараметрів машинного навчання для аналізу великих даних. Springer International Publishing. ISBN: 978-3-319-14962-0
6. Wang, Y., & Wang, L. (2018). Огляд антилових алгоритмів оптимізації для нейронних мереж. Artificial Intelligence Review. DOI: https://doi.org/10.1007/s10462-018-9742-4
7. Russell, S., & Norvig, P. (2010). Штучний інтелект: сучасний підхід (3-є видання). Prentice Hall. ISBN: 978-0136042594
8. Li, J., & Wang, H. (2017). Покращення антилових алгоритмів оптимізації для навчання нейронних мереж. Swarm and Evolutionary Computation.DOI: https://doi.org/10.1016/j.swevo.2017.05.001
9. Bishop, C. M. (2006). Розпізнавання шаблонів і машинне навчання. Springer. ISBN: 978-0387310732
10. Simonyan, K., & Zisserman, A. (2014). Дуже глибокі згорткові мережі для великомасштабного розпізнавання зображень. arXiv:1409.1556. URL: https://arxiv.org/abs/1409.1556