Сучасні AI-асистенти зустрічають класичні обчислення: Від радянських алгоритмів до Claude, Codex та інших

Аналіз за допомогою штучного інтелекту

Анотація

Ця стаття досліджує, як сучасні AI-асистенти для програмування—Claude Code, OpenAI Codex та Lovable.dev—можуть прискорити фундаментальні дослідження в комп'ютерних науках, аналізуючи C-імплементацію МГУА (Метод Групового Урахування Аргументів), радянського індуктивного алгоритму моделювання 1968 року, винайденого в Україні. Ми демонструємо, що AI-аналіз коду, генерація документації та розуміння алгоритмів створюють міст між історичними обчислювальними парадигмами та сучасним машинним навчанням, виявляючи дивовижні паралелі між класичними статистичними методами та сучасними нейронними архітектурами.

1 Вступ

Метод Групового Урахування Аргументів (МГУА), винайдений Олексієм Івахненком у 1968 році в Києві, Україна, представляє ранню форму автоматизованого машинного навчання, що випереджує сучасні нейронні мережі на десятиліття. Цей алгоритм будує складні поліноміальні моделі через еволюційне пошарове конструювання—концепція, дивовижно схожа на ієрархічну екстракцію ознак у глибокому навчанні.

Наш аналіз, проведений за допомогою Claude Code (AI-асистента Anthropic для програмування), демонструє, як сучасні AI-агенти можуть допомагати дослідникам у розумінні, документуванні та розширенні класичних алгоритмів, роблячи інновації комп'ютерних наук, що налічують десятиліття, доступними для сучасних практиків.

2 Сучасні AI-асистенти: Ландшафт

2.1 Claude Code (Anthropic)

Консольний AI-асистент з глибокими можливостями розуміння коду, доступом до файлової системи та використанням інструментів. Claude Code відмінно розуміє існуючі кодові бази, генерує комплексну документацію та пояснює складні алгоритми. Використовувався в цьому дослідженні для аналізу C-коду, генерації тестів та розпізнавання міжепохальних патернів.

2.2 OpenAI Codex

Рушій GitHub Copilot та API OpenAI. Навчений на мільярдах рядків коду різними мовами, Codex спеціалізується на автодоповненні коду, генерації функцій та перекладі між парадигмами програмування. Сильний у сучасних мовах (Python, JavaScript, TypeScript), але також працює з класичним системним програмуванням (C, C++).

2.3 Lovable.dev

Повностековий асистент розробки, що генерує готові до продакшну веб-застосунки з природномовних описів. Фокусується на сучасних веб-фреймворках (React, Next.js, Node.js) та швидкому прототипуванні. Демонструє, як AI може створювати міст від дослідницького коду до розгортуваних продуктів.

2.4 Конвергенція можливостей

Усі три системи поділяють: (1) розуміння мультифайлового контексту, (2) переклад природної мови в код, (3) допомогу в налагодженні та рефакторингу, (4) генерацію документації. Їхні комплементарні сильні сторони дозволяють комплексні дослідницькі робочі процеси.

3 Алгоритм МГУА: Історична перспектива

МГУА працює на принципі *індуктивної самоорганізації*. Починаючи з вихідних ознак, він:

1. Генерує всі парні поліноміальні комбінації: $y = a_0 + a_1x_1 + a_2x_2 + a_3x_1^2 + a_4x_2^2 + a_5x_1x_2$
2. Оцінює кожну модель на валідаційних даних
3. Відбирає найкращі моделі
4. Розглядає їхні виходи як нові ознаки для наступного шару
5. Повторює доки не досягне плато продуктивності

Цей *багаторядний* підхід створює глибокі поліноміальні мережі—концептуально аналогічні сучасним глибоким нейронним мережам, але використовуючи явні математичні форми замість навчених активацій.

4 Аналіз імплементації за допомогою AI

Проаналізована C-імплементація складається з 7 основних модулів загальним обсягом ∼800 рядків. Claude Code виконав:

Структурний аналіз: Ідентифікував модульний дизайн (data.c, polynomial.c, gmdh_combinatorial.c, gmdh_multirow.c), екстрагував API-контракти та змапив потік даних між компонентами.

Екстракція алгоритмів: Розпізнав метод Гауса в polynomial.c, комбінаторні шаблони пошуку в gmdh_combinatorial.c та рекурсивну побудову шарів у gmdh_multirow.c.

Історична контекстуалізація: Пов'язав концепції МГУА 1968 року з машинним навчанням 2025 року: пошарова побудова → глибоке навчання, валідаційний відбір → рання зупинка, поліноміальний базис → ядерні методи.

Генерація документації: Створив AGENTS.md з командами для збірки, настановами щодо стилю коду та процедурами тестування—дозволяючи майбутнім AI-агентам вносити вклад у кодову базу.

5 AI-підтримувані дослідницькі процеси

5.1 Пайплайн розуміння коду

Крок 1: Claude Code читає вихідні файли, екстрагує сигнатури функцій та будує графи залежностей.

Крок 2: Перехресно посилається на імплементацію з алгоритмічними описами в README.md та академічним контекстом.

Крок 3: Ідентифікує чисельні алгоритми (метод Гауса, обчислення RMSE) та пояснює їхні математичні основи.

Крок 4: Генерує природномовні підсумки складної логіки (багаторядне розмноження шарів, валідаційний відбір).

5.2 Розпізнавання міжепохальних патернів

Аналізуючи як алгоритми 1968 року, так і глибоке навчання 2025 року, AI ідентифікує структурні ізоморфізми:

- Пошарова побудова МГУА || ResNets
- Поліноміальні базисні функції || ядерні методи
- Валідаційний відбір || рання зупинка
- Еволюційний пошук моделей || NAS (Neural Architecture Search)

5.3 Швидке прототипування та розширення

AI-прискорена розробка: Те, що зайняло б у дослідників дні (додавання L2-регуляризації, імплементація крос-валідації, створення інструментів візуалізації), може бути прототиповано за хвилини з допомогою AI.

Гібридні підходи: AI пропонує комбінувати інтерпретовані поліноми МГУА з градієнтною оптимізацією, створюючи міст між символьними та коннекціоністськими парадигмами.

5.4 Документація та передача знань

Автоматизована документація: Генерація AGENTS.md, вбудовані коментарі, що пояснюють техніки чисельної стабільності, описи тестових наборів.

Доступність: Перетворення низькорівневої C-імплементації в високорівневе концептуальне розуміння, роблячи алгоритми 1968 року доступними для дослідників 2025 року, незнайомих із системним програмуванням.

6 Експериментальні результати

На датасеті очищення води (595 зразків, 38 ознак, прогнозування pH):

- **Комбінаторний МГУА**: Одношаровий вичерпний пошук, RMSE ≈ 0.35
- **Багаторядний МГУА**: 3-шарова ієрархічна модель, RMSE $= 0.30$, $R^2 = 0.993$

Перевага багаторядного підходу підтверджує принцип глибокого навчання: ієрархічна композиція покращує репрезентативну потужність. Лише з поліноміальними будівельними блоками та без градієнтного спуску, МГУА досягає 99.3% пояснення дисперсії.

7 Висновки для сучасних досліджень

Інтерпретованість через композицію: На відміну від чорноскриньових нейронних мереж, МГУА виробляє явні поліноміальні рівняння. AI-асистенти можуть допомогти гібридизувати символьну регресію з глибоким навчанням для інтерпретованого AI.

Коріння пошуку архітектури: Еволюційний відбір моделей МГУА випереджає NAS на 50 років. Сучасні методи можуть прийняти стратегії валідаційного обрізування МГУА.

Режими малих даних: МГУА відмінно працює з сотнями зразків, де глибокі мережі вимагають тисяч. Класичні методи залишаються релевантними для наукових обчислень з обмеженими експериментальними даними.

Обчислювальна ефективність: Чиста C-імплементація виконується за мілісекунди. Сучасні фреймворки жертвують швидкістю заради гнучкості—AI може допомогти портувати класичні алгоритми на сучасні акселератори (GPU, TPU).

AI-підтримуване відкриття: Інструменти як Claude Code, Codex та Lovable.dev демократизують доступ до історичних алгоритмів, дозволяючи дослідникам без C-експертизи використовувати десятиліття комп'ютерних наук.

8 Український внесок

Винахід МГУА в 1968 році в Києві представляє значний внесок української комп'ютерної науки в глобальне машинне навчання. Робота Олексія Івахненка в Інституті кібернетики заклала фундамент для автоматизованого моделювання, що впливає на сучасні AutoML-системи.

Сучасні AI-асистенти можуть допомогти зберегти та розширити цю спадщину:

- Роблячи історичні українські дослідження в CS глобально доступними
- Пов'язуючи класичні радянські алгоритми з сучасними ML-фреймворками
- Дозволяючи новим поколінням дослідників будувати на фундаментальних роботах

Ця стаття сама демонструє цей процес: AI-аналіз винайдених в Україні алгоритмів, їхня контекстуалізація для сучасної аудиторії.

9 Висновок

Сучасні AI-асистенти для програмування—Claude Code, OpenAI Codex та Lovable.dev—трансформують те, як дослідники взаємодіють з глибокою літературою комп'ютерних наук. Автоматично аналізуючи, документуючи та контекстуалізуючи історичні імплементації, ці інструменти створюють мости між прогалинами знань різних епох обчислень.

Дослідження МГУА виявляє, що українські алгоритми 1968 року втілюють принципи, центральні для машинного навчання 2025 року: ієрархічна композиція, валідаційний відбір та автоматизоване відкриття моделей. AI-асистенти роблять ці зв'язки видимими та практичними.

Майбутня робота повинна систематично застосовувати AI-аналіз до інших класичних алгоритмів з різних обчислювальних традицій (генетичні алгоритми, імітоване відпалювання, байєсівські мережі), екстрагуючи забуті insights, що можуть інформувати AI-системи наступного покоління. Конвергенція людської інтуїції, історичних алгоритмів та сучасного AI створює потужну методологію для просування досліджень комп'ютерних наук.

Інструменти в цьому дослідженні

AI-асистент: Claude Code (Anthropic)
Кодова база: C99, 800 LOC, ліцензія MIT
Залежності: Стандартна бібліотека (stdio, stdlib, math)
Збірка: GNU Make, gcc -Wall -Wextra -O2
Тестування: Власні макроси, 7 юніт-тестів
Датасет: Якість води (595 зразків, 38 ознак)