

What rebuilding AlphaGo teaches us about self-play, RL, and future of LLMs - Eric Jang

@DwarkeshPatel · 2026-05-15 · https://www.youtube.com/watch?v=X_ZVSPcZhtw



Эрик Джанг (бывший вице-президент по ИИ в 1X Technologies и старший научный сотрудник Google DeepMind Robotics) обсуждает архитектуру AlphaGo, которую он воссоздал с нуля во время саббатикала. Выпуск представляет собой глубокое техническое погружение в алгоритмы поиска, обучение с подкреплением (RL) и экстраполяцию этих методов на современные языковые модели.

Главная мысль выпуска заключается в том, что алгоритм Monte Carlo Tree Search (MCTS) в связке с нейросетями решает фундаментальную проблему неэффективности классического RL. Способность нейросети «сжимать» вычислительно нерешаемое дерево поиска в один forward pass объясняет не только успех AlphaGo, но и указывает на ограничения современных методов обучения LLM (таких как REINFORCE), предлагая пути для создания самообучающихся ИИ-исследователей.

Вычислительная сложность Го и правила игры

Снижение стоимости обучения 01:38: В 2020 году open-source проект KataGo сократил объем вычислений, необходимых для обучения сильного ИИ для Го с нуля (tabula rasa), в 40 раз. То, на что у команды DeepMind уходили миллионы долларов и годы исследований, сегодня можно воспроизвести за несколько тысяч долларов на арендованных GPU с помощью написания кода через LLM.

Фактчек: KataGo сократил вычисления в 40 раз (2020) — ⚠: Порядок верен (в статье Ву заявлено ~50× относительно ELF OpenGo/Leela Zero), но первый релиз —

2019, а не 2020. Gemini считает утверждение полностью подтверждённым; Claude уточняет, что «несколько тысяч долларов» дают уровень сильного дана, а сверхчеловеческий уровень у KataGo требовал месяцев распределённого обучения.

Правила Тромпа-Тейлора (Tromp-Taylor) 02:45: В отличие от человеческой игры, где статус мертвых камней и финальный счет определяются консенсусом игроков (value function людей), ИИ обучается на правилах Тромпа-Тейлора. Они абсолютно однозначны и алгоритмически разрешимы: игра идет до тех пор, пока оба игрока не спасуют, а очки начисляются за все контролируемые камни и пустые пересечения, не касающиеся камней противника.

Фактчек: Правила Тромпа-Тейлора, формула PUCT, усреднение value net с rollout в AlphaGo Lee ($\lambda=0.5$) и отказ от него в Zero — : Все рецензенты сходятся — технические детали изложены корректно.

Проблема комбинаторного взрыва 09:56: На доске 19x19 в начале игры доступно 361 действие. Средняя длина партии составляет 250-300 ходов. Наивное дерево поиска без слияния узлов имеет размер порядка 361^{300} вариантов, что значительно превышает число атомов во Вселенной. Из-за этого десятилетиями считалось, что Го не поддается решению методами перебора в этом веке.

Фактчек: Дерево поиска 361^{300} , «Го не решить в этом веке» — : $361^{300} \approx 10^{767}$ — это грубая наивная оценка; стандартная оценка дерева игры $\sim 10^{360}$, легальных позиций $\sim 10^{170}$, средняя профессиональная партия ~ 210 ходов. Качественный вывод (перебор невозможен) не меняется, но цифры завышены, а прогнозы до 2016 давали «10+ лет», а не «век». Gemini принял цифру без оговорок, Claude указал на преувеличение.

Monte Carlo Tree Search (MCTS) и нейросети

Алгоритмы выбора действий (UCB1 и PUCT) 13:16: Для навигации по дереву поиска используется формула, балансирующая эксплуатацию (выбор лучшего известного хода) и исследование (проверка новых веток). Алгоритм PUCT выбирает действие, максимизируя сумму $Q(s,a)$ (средняя ценность действия) и бонуса за исследование: $c \cdot P_a \cdot \sqrt{N} / (1 + N_a)$, где P_a — вероятность хода от нейросети, N — общее число визитов в родительский узел, а N_a — число визитов в конкретный дочерний узел.

Роль вероятностей в детерминированной игре 20:32: Го — игра с полной информацией и без элемента случайности. Понятие вероятности возникает из метода Монте-Карло: Q-значение интерпретируется как ожидаемая ценность действия при случайном распределении, индуцированном процессом поиска.

Две нейросети для отсеечения дерева 24:56: Чтобы сделать поиск решаемым, AlphaGo использует две нейросети. Value network (функция ценности) оценивает вероятность победы из текущего состояния, отсекая необходимость искать в

глубину до конца игры. Policy network (сеть политики) предсказывает распределение вероятностей хороших ходов, отсекая ширину дерева (ненужные ветки).

[31:03-32:09] Рекламная вставка Cursor.

Архитектура нейросетей 32:09: Сеть принимает на вход закодированную доску (каналы для черных, белых и пустых клеток). ResNet превосходит трансформеры в условиях малого бюджета вычислений благодаря индуктивному смещению локальных сверток. Трансформеры становятся лучше только при необходимости глобального контекста (агрегации признаков с разных концов доски), что в KataGo решалось добавлением слоев глобального пулинга.

Фактчек: ResNet превосходит трансформеры при малом бюджете вычислений — □: Индуктивное смещение локальных свёрток даёт преимущество, трансформерам нужно больше данных и FLOP для тех же пространственных зависимостей.

Инициализация экспертными данными 39:13: В глубоком обучении критически важна инициализация. Оригинальная AlphaGo Lee стартовала с supervised learning на базе партий людей-экспертов. Сеть училась предсказывать ходы победителей. Даже без MCTS такая «сырая» сеть, стреляющая навскидку (argmax по логитам), уже играет на уровне сильного человека.

Фактчек: AlphaGo Lee стартовала с supervised learning на партиях людей; обучение — предсказывать распределение визитов MCTS — △: Первая часть верна. Но схема «сеть учится предсказывать распределение MCTS» — это AlphaGo Zero/AlphaZero; в оригинальной AlphaGo Lee политика после supervised-этапа улучшалась через policy gradient в self-play. «Только ходы победителей» и старт с досок 4×4/9×9 — тоже не стандартный протокол оригинальных систем, а практика самого Джанга при воссоздании (расхождение: Gemini □, GPT △).

Четыре шага симуляции MCTS 43:24: На каждом ходу дерево строится заново через итерации (от 200 до 2048 симуляций). Шаг 1 (Selection): спуск по дереву на основе PUCT. Шаг 2 (Expansion): добавление новых дочерних узлов. Шаг 3 (Evaluation): оценка нового узла через Value network (в AlphaGo Lee это усреднялось с реальным случайным плеяутом до конца игры, но позже от этого отказались ради скорости). Шаг 4 (Backup): обновление Q-значений вверх по дереву до корня.

Self-Play и обучение с подкреплением

[58:51-1:00:30] Рекламная вставка Jane Street.

MCTS как оператор улучшения политики 1:00:30: После завершения MCTS получается распределение визитов, которое гораздо точнее и увереннее, чем изначальное предсказание Policy network. Суть обучения AlphaGo (Self-Play) —

заставить сырую нейросеть предсказывать финальное распределение MCTS. Это дистиллирует тысячи шагов поиска в один forward pass.

Аналогия с алгоритмом DAgger 1:06:00: MCTS работает как эксперт в робототехнике. Даже если агент проиграл партию (отклонился от оптимальной траектории), MCTS для каждого сделанного шага вычисляет строго лучшее действие. Обучение на этих парах «состояние — улучшенное действие» возвращает агента на оптимальную траекторию.

Фактчек: MCTS для каждого состояния вычисляет строго лучшее действие и даёт независимый лейбл — □: Гарантии улучшения нет — при ограниченном бюджете поиска и ошибочной value network результат хуже исходной политики (пересказ сам это признаёт в разделе «когда MCTS работает хуже»). Лейблы зависят от той же сети, так что «независимый эксперт» — натяжка.

Когда MCTS работает хуже политики 1:08:09: Улучшение через MCTS не гарантировано математически, это эвристика. Если Value network плохо обучена на эндшпилях (например, если боты всегда сдаются рано и в буфере обмена нет терминальных состояний), ошибки оценки поднимутся вверх по дереву, и MCTS выдаст распределение хуже, чем изначальная интуиция сети.

Бутстраппинг с малых досок 1:11:22: Для tabula rasa обучения (AlphaZero) выгодно начинать со случайной игры на досках 4x4 или 9x9. На малых досках случайная игра быстро генерирует реалистичные эндшпили, позволяя Value network быстро выучить концепцию победы/поражения. Эти веса затем отлично переносятся (transfer learning) на доску 19x19.

Философия вычислений: сжатие сложности и хаос

Амортизация NP-трудных задач 1:17:12: Главный прорыв AlphaGo — демонстрация того, как 10 слоев нейросети (10 последовательных шагов распределенных вычислений) могут аппроксимировать вычислительно нерешаемую задачу поиска. Это ставит под сомнение нашу фиксацию на худших случаях (worst-case complexity) в проблеме $P=NP$. Тот же принцип позже позволил AlphaFold решить задачу фолдинга белка.

Фактчек: 10 слоёв сети амортизируют NP-трудную задачу, ставя под сомнение фиксацию на $P=NP$; тот же принцип решил фолдинг в AlphaFold — □/□: Хорошая игра на типичных позициях ничего не говорит о худшем случае — P vs NP по определению про worst-case. AlphaFold — supervised learning на PDB без MCTS и амортизации поиска; она предсказывает структуру, но не решает физику процесса сворачивания. Критика фиксации на worst-case как исследовательской привычки — обоснованное мнение, не более.

Макроструктура хаоса 1:20:27: Го похожа на хаотическую систему (погоду), где малейшее изменение (один камень) полностью меняет точное будущее состояние доски. Однако нейросеть учится предсказывать макроскопическую

величину (кто победит), подобно аттрактору Лоренца. В отличие от криптографических хеш-функций, где структура намеренно уничтожается, в Го и физике есть макроструктура, которую ИИ способен уловить на границе хаоса.

MCTS против RL для LLM (Policy Gradient)

Проблема дисперсии в наивном RL 1:24:17: Современные LLM обучаются методами вроде REINFORCE, где вся сгенерированная последовательность токенов получает единый скалярный реворд в конце. Если модель случайно сгенерировала один гениальный токен среди тысяч обычных и получила награду, алгоритм усиливает вероятности *всех* токенов в эпизоде. Это создает колоссальную дисперсию градиента, растущую квадратично от длины эпизода.

Фактчек: Дисперсия REINFORCE растёт квадратично от длины эпизода — $\triangle$: Рост дисперсии с горизонтом реален, но степень зависит от корреляций, структуры награды и baseline (в теории встречаются $O(T^2)$ – $O(T^3)$); PPO/GRPO с advantage-оценками частично гасят проблему. Gemini ссылается на errata самого Джанга, где он признал формулировку про квадратичный рост некорректной — другие рецензенты этого не подтверждают, проверить не удаётся.

Преимущество MCTS в назначении кредита (Credit Assignment) 1:37:34: В отличие от LLM, где награда «всасывается через соломинку» (по выражению А. Карпати), MCTS в Го даёт плотный, независимый сигнал (label) для *каждого* отдельного действия. Сеть просто учится предсказывать этот улучшенный лейбл через supervised learning (минимизация KL-дивергенции), что делает обучение невероятно стабильным.

Neural Fictitious Self-Play 1:38:44: В играх с неполной информацией (StarCraft), где MCTS невозможен, используется обучение лучшего ответа (best response). Агент обучается максимизировать победы против фиксированного набора прошлых версий противников (с помощью PPO или Q-learning). Затем эти политики дистиллируются в единую смешанную стратегию, которая становится устойчивой к любым действиям оппонента.

Фактчек: Neural Fictitious Self-Play для StarCraft; при неполной информации MCTS невозможен — $\triangle$: NFSP (Heinrich & Silver, 2016) — про покер; AlphaStar использовал league training с prioritized fictitious self-play — родственная, но другая схема. Для неполной информации существуют information-set MCTS; «невозможен» — преувеличение.

Сложность применения MCTS к LLM 1:46:05: В Го ширина дерева ограничена (361), а глубина отсекается объективной функцией ценности. В языке пространство действий (словарь) огромно, и агент почти никогда не сэмплирует один и тот же дочерний узел дважды. Эвристика PUCT (с корнем из N) ломается, так как нет возможности локально оценить «ценность» промежуточного токена независимо от решения всей задачи.

Фактчек: PUCT ломается для LLM из-за размера словаря — $\triangle$: Трудности реальны (огромное пространство действий, нет локальной ценности токена), но поиск для LLM применяют — через отбор кандидатов, ветвление по шагам рассуждения, verifier-модели.

Эффективность обучения: Bits per FLOP и Off-Policy

Информационная плотность RL против Supervised Learning 2:11:34: В начале обучения LLM с нулем знаний (словарь 100 000 токенов) вероятность угадать правильное слово («blue») равна 1/100 000. В RL модель пробует случайные слова, получает отказ и узнает почти ноль бит информации (энтропия бинарной переменной при вероятности успеха близкой к нулю). Модель тратит 99.99% времени впустую.

Сила Soft Targets (дистилляции) 2:13:56: В Supervised Learning модель сразу получает правильный ответ и через cross-entropy loss корректирует все вероятности, получая $-\log(p)$ бит информации. AlphaGo идет еще дальше: она обучается не на жесткий лейбл (лучший ход), а на *распределение* MCTS (soft targets). Энтропия такого распределения огромна, что дает экспоненциально больше информации на каждый сэмпл (Bits per FLOP).

Фактчек: Soft targets дают экспоненциально больше бит на сэмпл — $\triangle/\square$: Бинарный реворд ≤ 1 бит, supervised-токен $\leq \log_2(100k) \approx 17$ бит — разница существенная, но «экспоненциально» — риторика. GPT жёстче: высокая энтропия сама по себе $\neq$ информативность (равномерное распределение максимально энтропийно и бесполезно), правильная формула — $-\log_2 p$, а bits/FLOP нельзя оценить без учёта стоимости получения учительских распределений.

Опасность Off-Policy обучения 2:01:05: В RL часто используют replay buffers со старыми траекториями. Если запустить MCTS-перемаркировку на абсолютно случайных состояниях из буфера, которые текущая политика *никогда* не посетит, емкость нейросети тратится впустую. Буфер должен содержать состояния на оптимальной траектории + состояния небольшого дрейфа вокруг нее, чтобы сеть училась корректировать ошибки.

Автоматизация ИИ-исследований и масштабирование

Упрощение архитектуры со временем 1:51:34: При воссоздании AlphaGo выяснилось, что многие сложные трюки из статьи KataGo (асинхронный RL, сложные аугментации) больше не нужны. Благодаря росту скорости GPU (Blackwell/Ada против V100), сырая вычислительная мощность позволяет использовать простую синхронную архитектуру. Быть первым стоит миллионы, догонять — копейки.

LLM как ИИ-исследователи 2:22:21: Модели уровня Claude Opus (4.6/4.7) уже отлично справляются с оптимизацией гиперпараметров и написанием кода для

экспериментов. Они могут самостоятельно переписать data loader или изменить архитектуру для устранения затухания градиентов, выжимая максимум метрик из фиксированного датасета.

Фактчек: Claude Opus 4.6/4.7 как ИИ-исследователи — [1]: Версии за пределами знаний рецензентов; оценка способностей — мнение из личной практики Джанга.

Проблема латерального мышления 2:24:30: Текущие LLM проваливаются в выборе *следующего* эксперимента. Если гипотеза в корне неверна, ИИ продолжает копать вглубь (rabbit-hole), вместо того чтобы сделать шаг назад, переосмыслить первые принципы и сменить вектор исследования.

Го как Outer Loop для самосовершенствования ИИ 2:26:08: Главная проблема автоматизации науки — верифицируемость идей (Inner Loop). В реальной науке сложно понять, не работает идея из-за бага в коде или из-за фундаментальной ошибки. Го предоставляет идеальный «внешний цикл» (Outer Loop): ИИ-исследователь может предлагать новые архитектуры или законы масштабирования, а проверка занимает минуты (win rate против KataGo). Навыки, полученные ИИ при оптимизации таких проверяемых сред, в будущем перенесутся на сложные задачи вроде поиска лекарств или создания AGI.

ЛИЧНО ДЛЯ ТЕБЯ

Догонять дёшево, быть первым дорого — используй это в решениях: фронтир 2016 года воспроизводится соло за тысячи долларов, причём сложные трюки (асинхронный RL, аугментации) отпали за счёт роста сырого compute. Не плати «налог первопроходца» там, где нет выигрыша от скорости; выбирай задачи с быстрым проверяемым фидбек-лупом — там и ты, и агент учись на порядки быстрее.

Твоя роль в связке с AI — латеральный шаг назад, а не валидация строк: модели уже хорошо крутят гиперпараметры и чинят код, но при неверной гипотезе роют вглубь (rabbit-hole). Перед запуском эксперимента фиксируй гипотезу, метрику, бюджет и условие остановки; после 2–3 неудач требуй альтернативное объяснение, а не очередную настройку.

Плотный сигнал на каждый шаг вместо одной оценки в конце: бинарное «получилось/нет» несёт почти ноль информации, разбор «что было лучше на каждом шаге» — максимум. Работает для гитары и барабанов (разбор по такту, а не «сыграл пьесу целиком») и для работы с AI-кодом (промежуточные проверки, а не финальный тест).

Не переноси доверие к игровым бенчмаркам на здоровье: в Го победа — объективный проверяемый исход, в longevity улучшение биомаркера ещё не доказывает прибавку лет жизни. Успех оптимизации внутри

«симулятора» (модель, surrogate-метрика) не заменяет клинической проверки — держи это в голове при оценке любого протокола.

КОНСЕНСУС РЕЦЕНЗЕНТОВ

Все трое подтвердили механику MCTS/PUCT, правила Тромпа-Тейлора и тезис «догонять дёшево»; сошлись, что квадратичный рост дисперсии REINFORCE и «экспоненциальный» выигрыш soft targets — преувеличения, а связь с $P=NP$ и AlphaFold — риторика.

Разошлись в строгости: Gemini принял почти всё (включая 361^{300} и $40\times$), Claude поправил цифры и хронологию (KataGo 2019, $\sim 50\times$, размер дерева, $NFSP \neq \text{AlphaStar}$), GPT самый жёсткий — поставил $\square$ на «строгое улучшение через MCTS», bits/FLOP и $P=NP$ и указал, что схема обучения на распределение MCTS относится к AlphaGo Zero, а не к AlphaGo Lee.