WSP на практике: как мы создали непобедимую сеть для крестиков-ноликов, написав её веса вручную

Разбор исследования MIT CSAIL и отчёт о собственном эксперименте по методу Weight-Space Programming

Ноябрь 2026

Введение: когда нейросеть становится программой

Исследователи из MIT CSAIL в ноябре 2026 года представили работу, которая меняет представление о том, как можно взаимодействовать с нейросетями. Они доказали, что современные большие языковые модели (LLM) способны **напрямую читать и изменять веса нейросетей** — без градиентного спуска, без обучения, без подбора гиперпараметров.

Этот подход они назвали **Weight-Space Programming (WSP)** — программирование в пространстве весов.

Суть метода проста: вместо того чтобы обучать сеть на данных, мы даём модели описание задачи и формат весов, а она сама «пишет» нужные числа. Получается, что нейросеть становится не чёрным ящиком, который нужно тренировать, а программой, которую можно отредактировать как текст.

Что показали авторы исследования

В главном эксперименте модель Astra получила веса крошечной Qwen3.5-0.8B и задачу «улучшить математические способности». Результат: Astra нашла в весах нейроны, вызывавшие зацикливание в рассуждениях, и исправила их.

Результаты: - MATH-500: с 7,9% до 48,1% - GSM8K: с 13,8% до 67,1%

При этом было изменено всего **0,016% весов**.

Но это не всё. Astra также: - По весам (17 тысяч чисел) угадала, в какую игру обучена сеть (Breakout, Pong, Pac-Man, Freeway) — **100% точность** - Определила, какие фигуры рисуют диффузионные модели (круг, квадрат, треугольник) - Написала с нуля веса сетей

для игр MinAtar, которые **обогнали PPO-сети, обученные 100 млн шагов** - Создала сеть для игры в Го, которая выиграла у PPO-сетей, обученных 20 млн партий

Наш эксперимент: непобедимая сеть для крестиков-ноликов

Вдохновлённые исследованием, мы решили проверить метод на практике. Задача: написать веса нейросети для крестиков-ноликов **без обучения**, только через прямое редактирование чисел.

Почему крестики-нолики?

Не из легкомыслия. Эта задача идеальна для проверки, потому что: - У неё есть **полный перебор** — все $3^9 = 19\,683$ позиции известны - Любое утверждение о силе сети можно доказать или опровергнуть исчерпывающе - Нет места «в среднем» или «примерно» — либо сеть непобедима, либо нет

Версия 1: первый провал

Мы создали сеть 18→48→60→9 (вход — 18 признаков, три скрытых слоя, 9 выходов — клетки поля).

Архитектура: - **Слой 1**: детекторы линий («есть наш камень», «два в ряд») - **Слой 2**: тактика («угроза», «вилка», «центр», «углы») - **Выход**: приоритеты ходов (выигрыш > блокировка > вилка > центр > угол)

Результат: против случайного соперника — 98% побед. Против минимакса — «ничья».

Показалось, что всё готово.

Твой ход: живая игра вскрыла дыру

Ты сыграл несколько партий и заметил: *«Когда я ставлю угол, а он ставит центр — сетка всегда проигрывает»*.

Это была классическая ловушка «два противоположных угла». 2000 случайных партий и минимакс её не покрывали — только живой игрок нашёл уязвимость за пять минут.

Три рефери, которые врали

Дальше началось самое интересное. Я начал исправлять сеть — и трижды рапортовал «починил». Каждый раз проверка показывала, что я ошибся.

Рефери №1 утверждал: «сеть непобедима» — и параллельно выдавал линию, которой её можно разбить. Причина: функция считала строку ничьей `'D'` победителем.

Рефери №2 (переборный) нашёл настоящий проигрыш, но его «первый зевок» указывал на невозможную позицию. Баг был в инвертированной логике базового случая.

Рефери №3 (таблица всех позиций) показал истинную картину: **сеть не видит вилку как класс объектов**.

Корневая причина: два фундаментальных ограничения

1. Арифметика молчунов

Юнит «вилка в клетке» должен суммировать сигналы от линий, проходящих через эту клетку. Но линии, где ничего нет, тоже «голосуют» — каждая даёт −1.

- Края: 2 линии → −2
- Углы: 3 линии → −3
- Центр: 4 линии → −4

Единый порог для всех клеток физически не работает. Формула: `bias = G·(n−3)`, где n — число линий через клетку.

2. Глубина абстракции = глубина сети

«Клетка — это вилка» = «≥2 чистых линий через клетку». Посчитать пару может только **слой над детекторами одиночных линий**.

В трёхслойной версии вилочные юниты висели проводами не туда — они читали сигналы с нижнего слоя, где эти сигналы ещё не сформированы. Никакая настройка порогов не лечит детектор, который подключён не к тем проводам.

Решение: четвёртый слой

Финальная архитектура: **18→48→63→18→9**

- **Слой 1** (48): детекторы линий (наш/их/пусто, ≥1, ≥2)
- **Слой 2** (63): тактика (угрозы, вилки, центр, углы, **TRAP** — «враг держит оба конца диагонали»)
- **Слой 3** (18): **слой-парообразователь** — «через клетку j проходят ≥2 чистых линии» = вилка
- **Выход** (9): приоритеты (выигрыш 240 > блок 140 > своя вилка 80 > ломка чужой 60 > центр > угол; при TRAP: углы −84, края +12)

473 ненулевых веса из ~4400 (10,7%). Ни одного градиента.

Верификация: теперь честная

- **22 250 проверок** (все позиции × 2 стороны) против булевых спецификаций: **0 ошибок детекторов**
- **Полный перебор всех партий + DP-аудит**: сеть непобедима обеими сторонами
- Твоя ловушка: отражается — сеть сама видит TRAP и берёт край
- Против случайного: 98,5% (X) / 93,1% (O)
- JS-реализация в игре: побитово идентична Python

Что это значит

Веса можно писать руками как программу. Это работает. 473 числа — и сеть играет идеально.

Но ровно до того предела, какой задал **честный рефери**. Трижды я думал, что починил — и трижды ошибался, потому что проверка врала.

Иерархия слоёв — не гиперпараметр, а следствие задачи. Прежде чем править веса — проверь, на каком слое живёт признак и на каком его потребляют. «Нельзя настроить то, что подключено не туда».

Человек в контуре — генератор адских кейсов. Симуляции покрывают среднее. Живой игрок находит хвост.

Как это укладывается в WSP

Мы прошли весь цикл: 1. Задали архитектуру и формат весов 2. Написали первую версию «программы» в весах 3. Проверили в среде (игра) 4. Нашли дефект (живая игра → ловушка) 5. Исправили конкретные числа (добавили слой, настроили bias по формуле) 6. Проверили исчерпывающе (полный перебор)

Знание «при двух углах бери край» переехало в веса напрямую — **человеческое открытие стало нейроном**, без обучения. Это и есть WSP в чистом виде.

Вывод

Веса нейросети — это код. Его можно писать, отлаживать и верифицировать. Но ровно до тех пор, пока у тебя есть честный судья, который не врёт.

Сначала построй рефери. Потом хвастайся сеткой.

Идеи и эксперимент — автора (Telegram: @Tottop10, pro100.D); разбор и программная огранка — агент. *CC BY 4.0.*

Исследование MIT CSAIL: Sparks of Weight-Space Programming