Обучение с подкреплением Reinforcement Learning

Александр Панов

План лекции

1. Последовательное принятие решений
2. Особый раздел машинного обучения
3. Основные алгоритмы: от Q-обучения до RLHF
4. Какие задачи помогает решать?
5. Примеры

О лекторе

Панов Александр

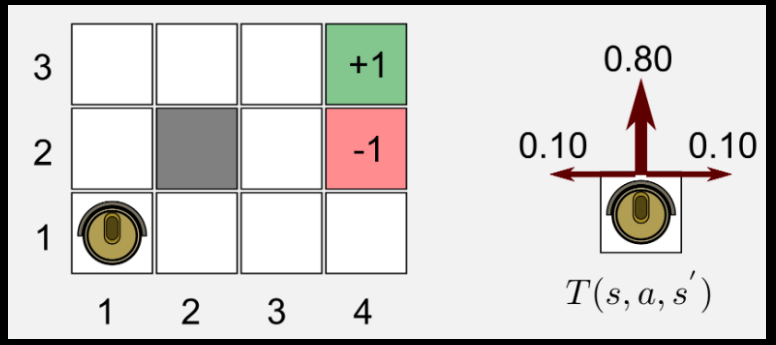
Директор лаборатории CAIS AIRI, директор Центра когнитивного моделирования МФТИ, д.ф.-м.н., доцент

- Обучение с подкреплением на основе модели мира
- Когнитивная робототехника
- Планирования поведения (языковые модели, многоагентные системы)

Контакты: Telegram (@grafft, @ai_panov), panov@airi.net

Последовательное принятие решений

Принятие решения



Входная информация

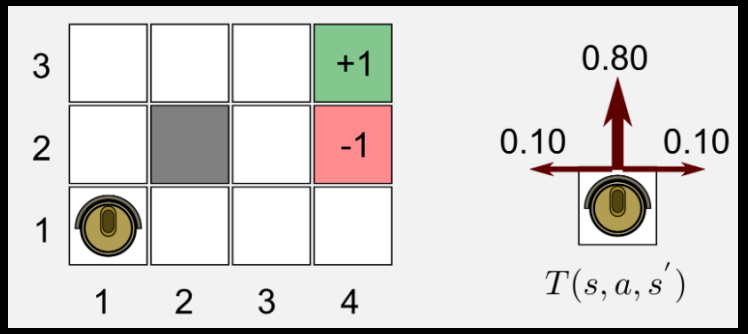
Принятие решений

Реализация решения

Оценка результата

- Входная информация: координаты GPS, запись о клиенте в БД
- Принятие решения: обработка, анализ в соответствии с целью (итоговым результатом)
- Реализация решения: движение вперед, звонок клиенту
- Оценка результата: расстояние до цели, выплата задолженности

Последовательное принятие решений

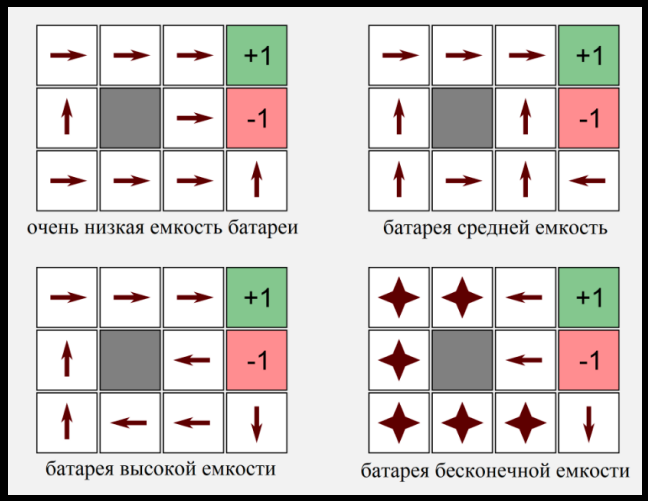


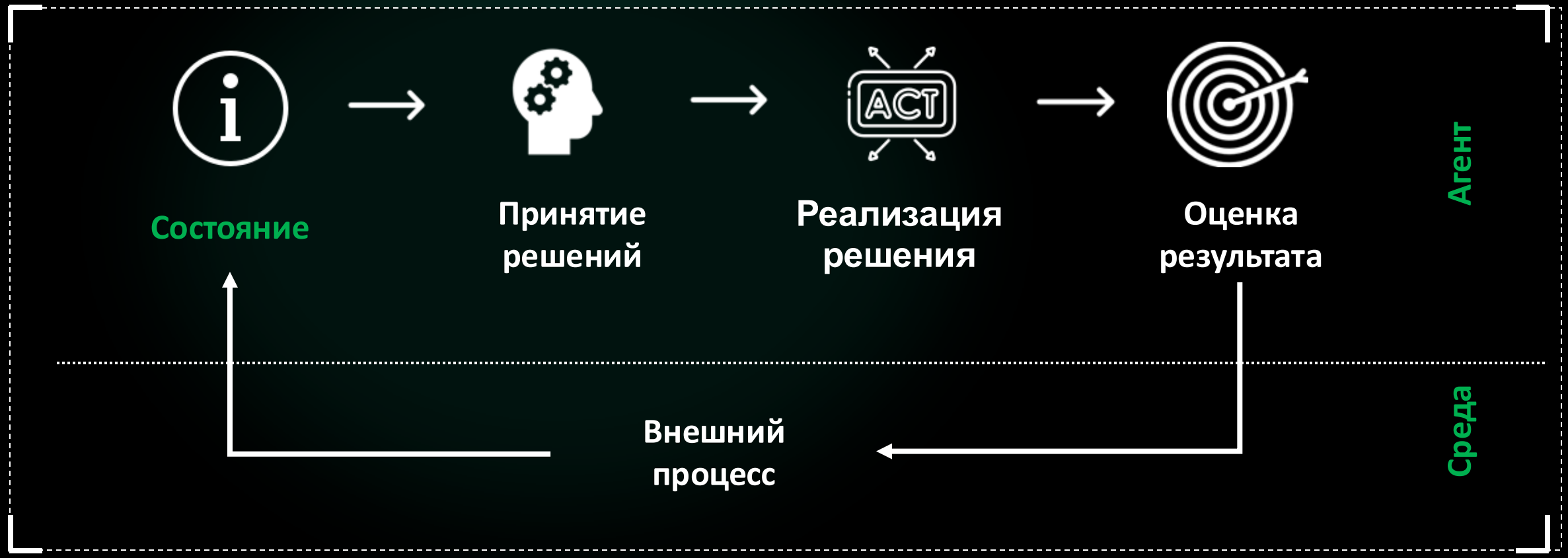
Входная информация → **Принятие решений** → **Реализация решения** → **Оценка результата**

Внешний процесс
(зарядка батареи, размышления клиента)

Последовательное принятие решений

- Дискретные моменты времени
- Внутренний (агент) и внешний (среда) процессы разделяются условно

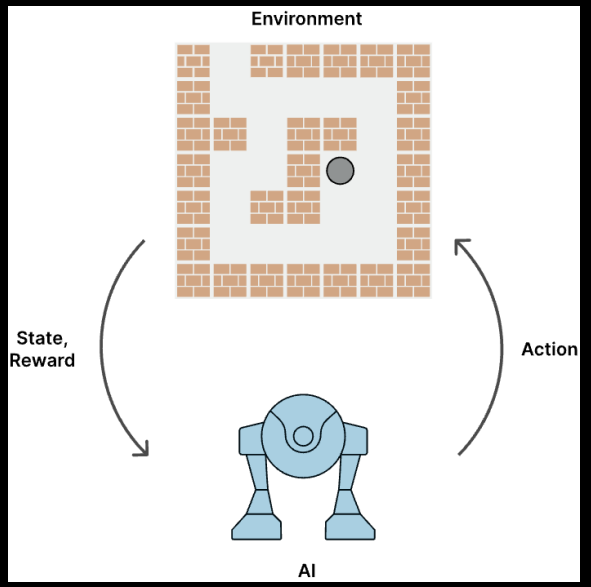




Обучение с подкреплением

Обучение интеллектуального агента **хорошей последовательности принятия решения** в среде с неполной информацией

Марковский процесс принятия решений



Пусть $< S, A, T, R, \ \gamma >$ - марковский процесс принятия решений, где:

- S – пространство состояний (информационных),
- A – множество действий (дискретных, непрерывных),
- $T: S \times A \to S$ – функция переходов (не известна агенту),
- $R: S \times A \to \mathbb{R}$ - функция вознаграждений (не известна агенту),
- γ – дисконтирующий множитель

Агент выполняет действия в среде, используя функцию стратегии (стохастическую или детерминированную)

$$\pi: S \to A$$

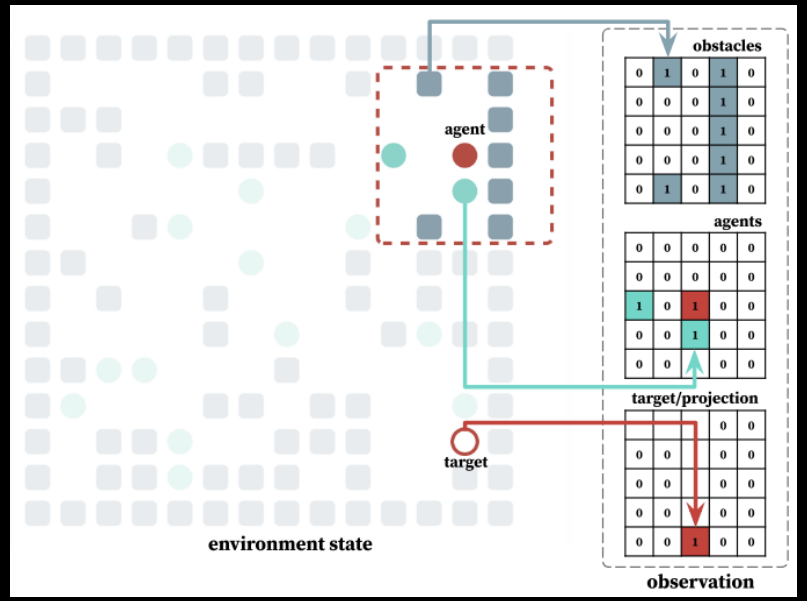
Цель агента – максимизировать ожидаемую отдачу по стратегии π:

$$\mathbb{E}_\pi \sum_{t=0}^{\tau} \gamma^t R(s_t, a_t)$$

Наблюдение и полезность

Состояния и наблюдения агента на примере клеточной среды:

- a_t - перемещения из одной клетки в соседнюю не занятую,
- $o_t \in \mathbb{R}^{(2d)^2}$ – наблюдения агента,
- $\hat{s}_t = f(o_1, \dots, o_t)$ – функция аппроксимации состояния (рекуррентная нейронная сеть)

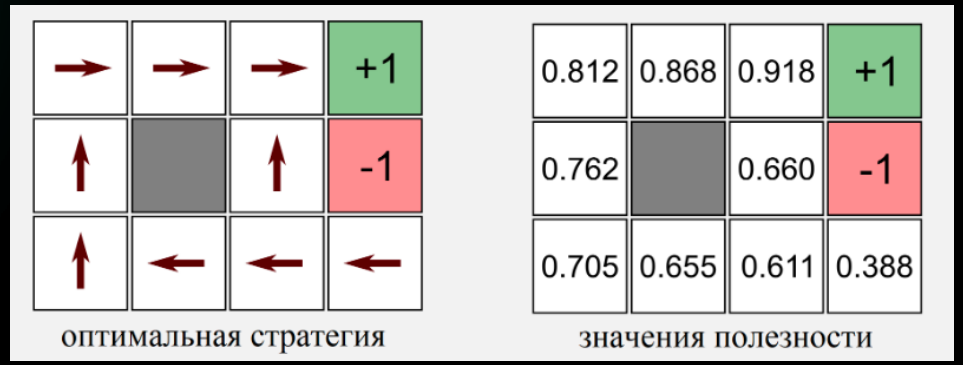


Функция полезности Q – ожидаемая отдача для текущего состояния и действия:

$$Q(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{i=t}^{\tau} \gamma^i R(s_i, a_i) \right]$$

Уравнение Беллмана для оптимальной функции полезности (оптимальная стратегия жадная по полезности):

$$Q^*(s, a) = \mathbb{E}_{s_t \sim T} \left[r_t + \gamma \max_{a_t} Q^*(s_t, a_t) \,|s, a \right]$$

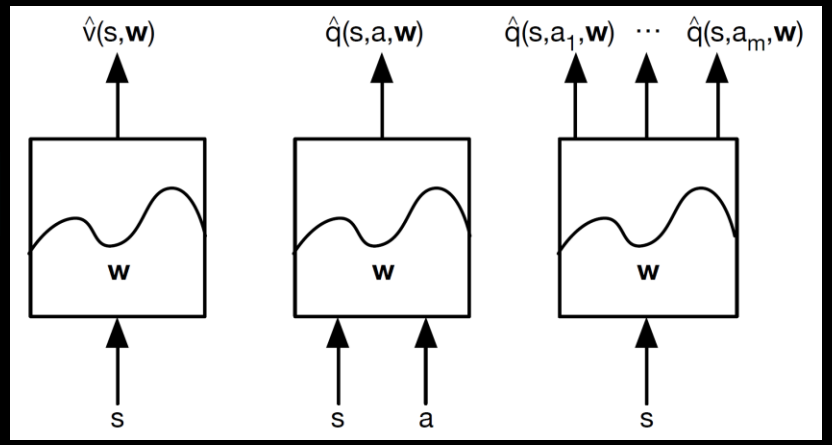


Аппроксимация полезности

- Уравнение Беллмана обычно решается итеративными методами и введением аппроксимации функции полезности: $\hat{Q}(s,a;\theta) \approx Q(s,a)$
- Для поиска оптимальных значений параметров θ вводится функция потерь:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a\sim\mathcal{D}}[(y - Q(s,a;\theta))^2]$$

$$y = \mathbb{E}_{s_t\sim T}\left[r_t + \gamma \max_{a_t} Q\,(s_t,a_t;\theta)\,|s,a\right]$$



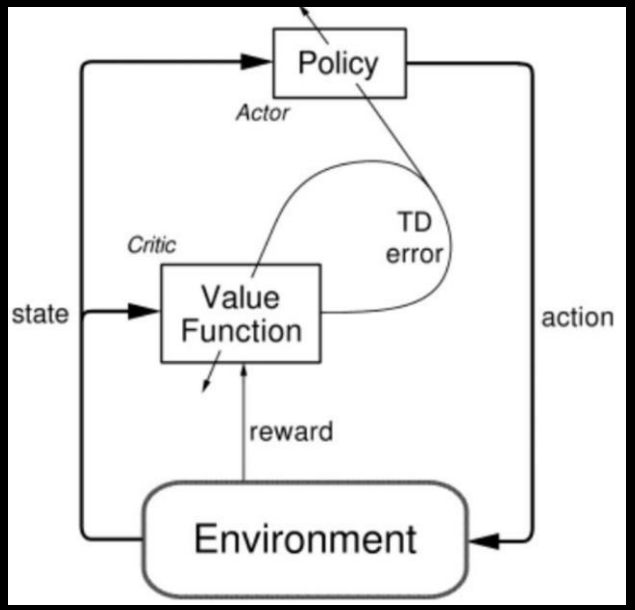
- Поиск минимума такой функции потерь можно проводить градиентными методами:

$$\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{s,a\sim\mathcal{D};s_t\sim T}\left[\left(r_t + \gamma \max_{a_t} Q\,(s_t,a_t;\theta) - Q(s,a;\theta)\right)\nabla_\theta \mathrm{Q}(\mathrm{s},\mathrm{a};\theta)\right]$$

Алгоритмы актор-критик

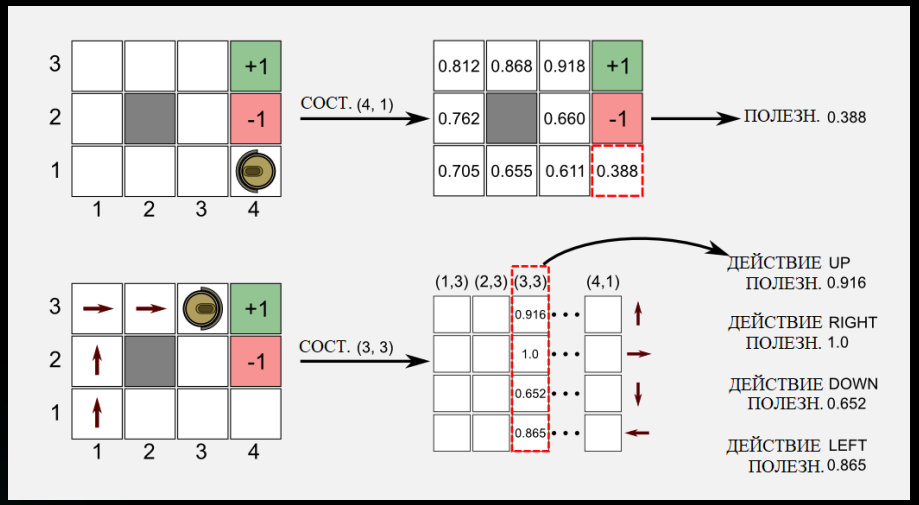
- Введем напрямую параметризацию стратегии: $\hat{\pi}(s; w) = \mathbb{P}[a|s, w]$
- Пусть задана дифференцируемая функция полезности стратегии J, тогда справедливо выражения для градиента:

$$\nabla_w J(w) = \mathbb{E}_{\hat{\pi}(w)}\left[\nabla_w \log \hat{\pi}(s; w)\, Q^{\hat{\pi}(w)}(s, a)\right]$$



- Для оценки значения функции полезности Q используется критик – получается архитектура актора-критика:
  - веса критика обновляются с помощью функции потерь $\mathcal{L}(\theta)$, обычно чаще, чем веса стратегии,
  - веса актора обновляются в соответствии с максимизацией функции полезности $J(w)$

Обучение с подкреплением



1. Предсказание **полезности** (value) – будущей суммы вознаграждений по опыту и **уравнению Беллмана**
2. Состояние (state) – **полная информация** о текущем моменте для оптимального принятия решения
3. Постепенное накопление опыта – **исследование среды** (exploration)
4. Стратегия (policy) – (почти) **жадная** по полезности

Использование опыта

1. Когда (?) опыт достаточно разнообразен – его можно **использовать** (exploitation) для обновления полезности/стратегии
2. Накопление опыта после нескольких обновлений – обучение с **отложенным опытом** (off-policy)
3. Использование только самого свежего опыта – обучение с **актуальным опытом** (on-policy)

Автономный полет вертолета

Игры Atari

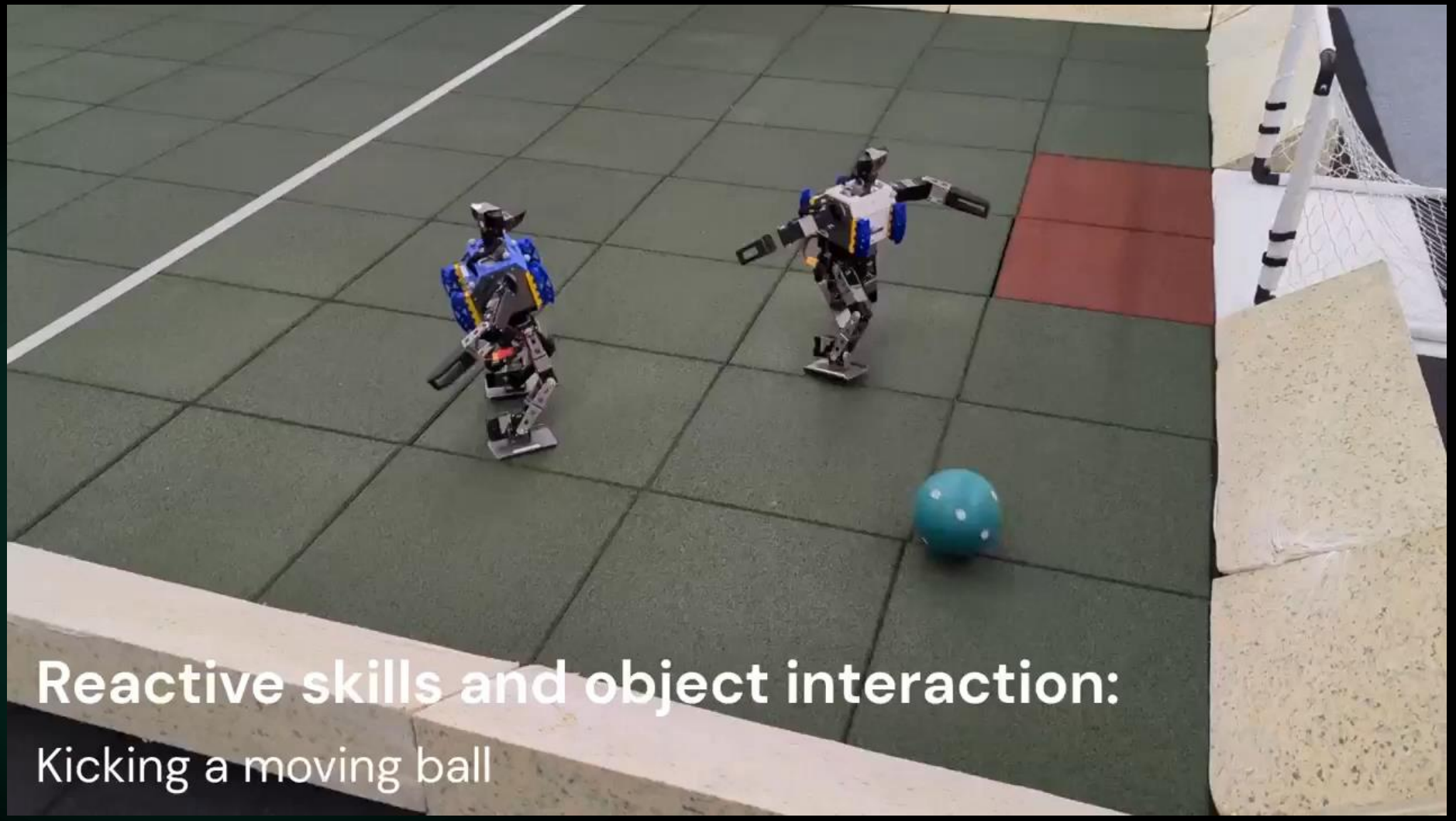
Google Deepmind DQN playing
Atari Breakout

Setup:
NVIDIA GTX 690
i7-3770K - 16 GB RAM
Ubuntu 16.04 LTS
Google Deepmind DQN

Манипуляция объектами

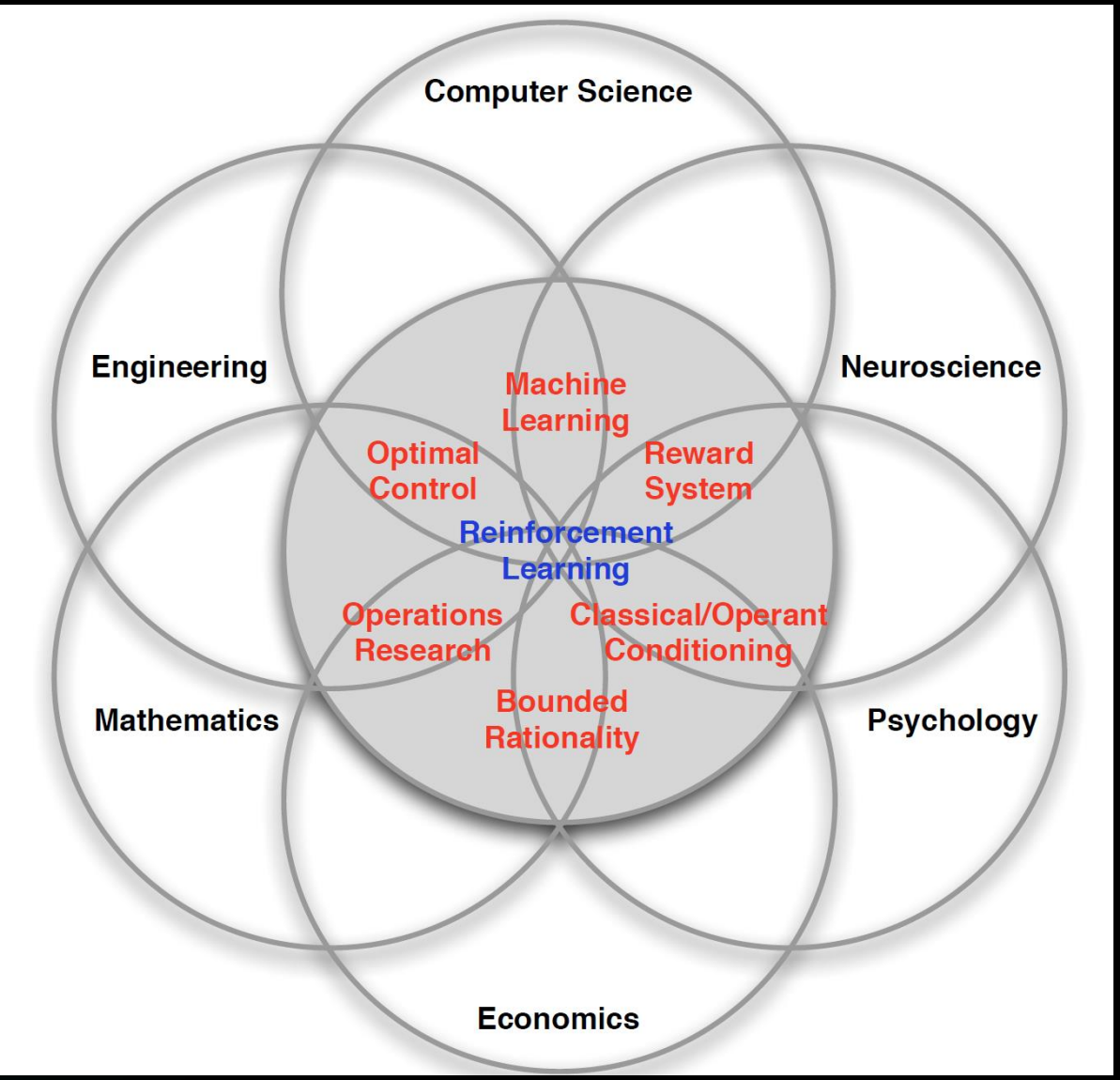
Многоагентная задача

Робофутбол


Reactive skills and object interaction:
Kicking a moving ball

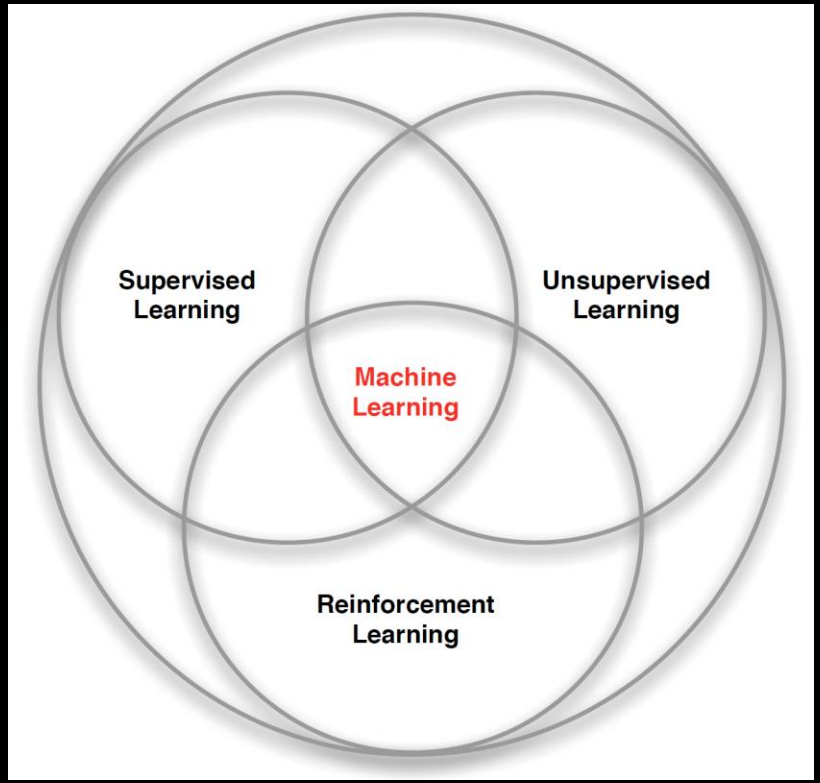
Особый раздел машинного обучения

RL и другие науки



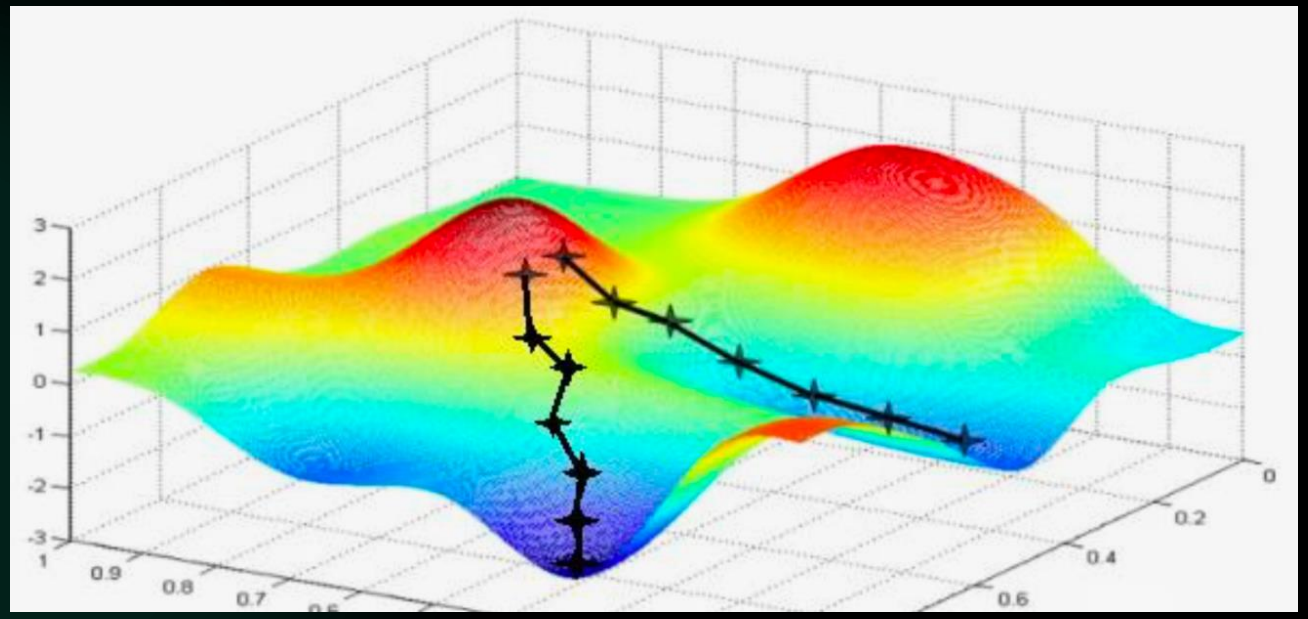
RL и классический ML

- Нет учителя, т. е. ошибка не задается явно, а косвенно передается через вознаграждение
- Действия агента влияют на поступающие в дальнейшем данные
- Целевая переменная (полезность) - нестационарна
- Обратная связь от среды может поступать с задержкой
- Поступающие данные сильно скоррелированы (не работает гипотеза iid)



RL – это мягкая оптимизация

- Полезность – параметризованное семейство функций (нейронная сеть)
- Стратегия – параметризованное семейство функций (нейронная сеть)
- Задача оптимизации:



Найти такие **параметры** функции, которые бы давали **максимальную** полезность или **оптимальную** стратегию

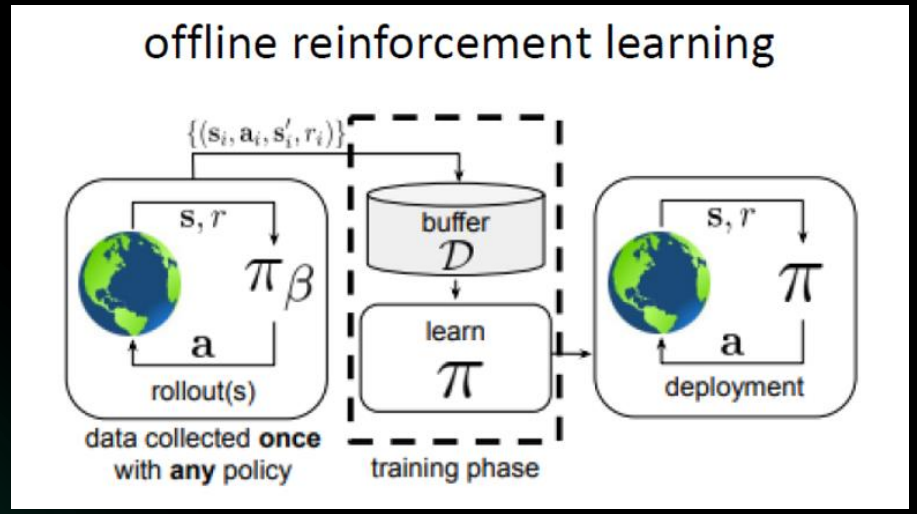
Особенности оптимизации

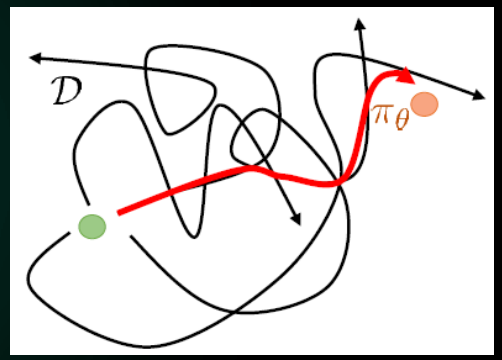
1. Оптимизируемый функционал – сумма дискретных величин → нужно строить суррогатные функционалы
2. Нестационарность задачи → необходимо оптимизировать градиентный шаг
3. Скоррелированность данных → нужно рандомизировать/приоритизировать пакет данных

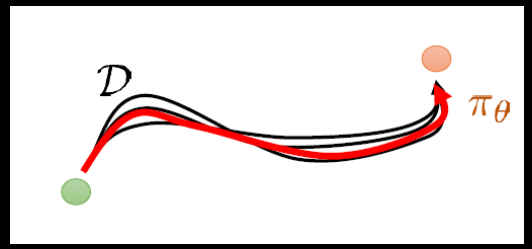
Основные алгоритмы

База 1: Клонирование поведения

- Пусть есть экспертные траектории (близкие к оптимальным)
- Обучение с учителем: предсказание того же действия, что и у эксперта
- Оптимизация полезности





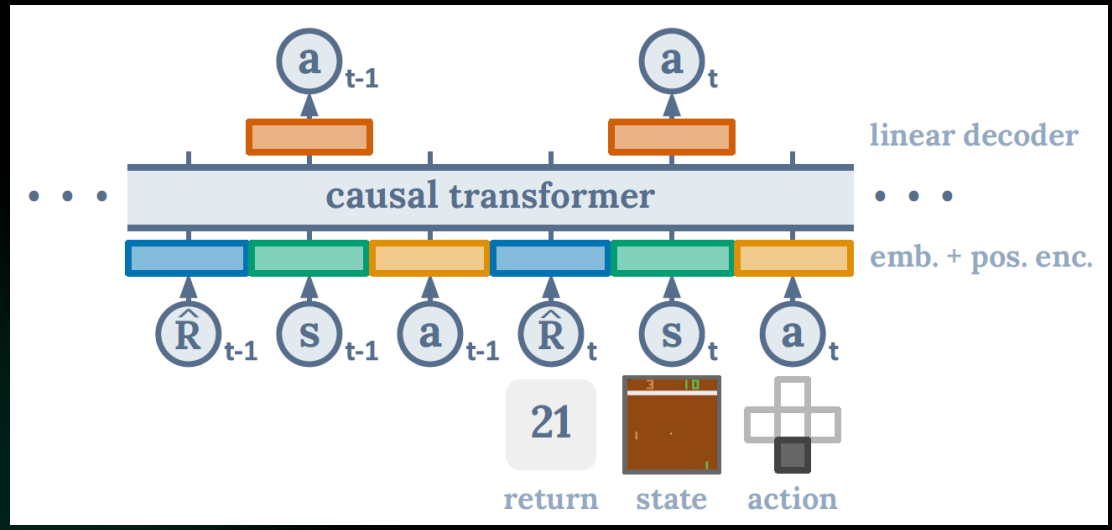


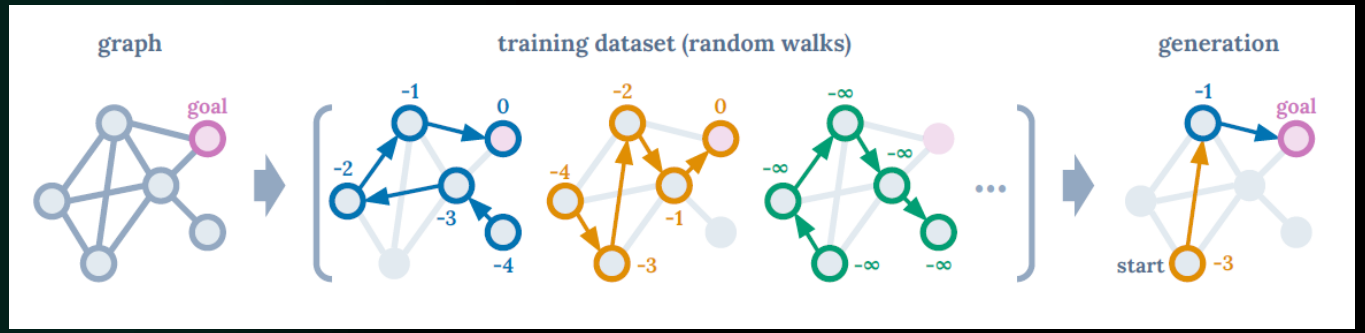
Плюсы: эффективный метод при отсутствии симулятора

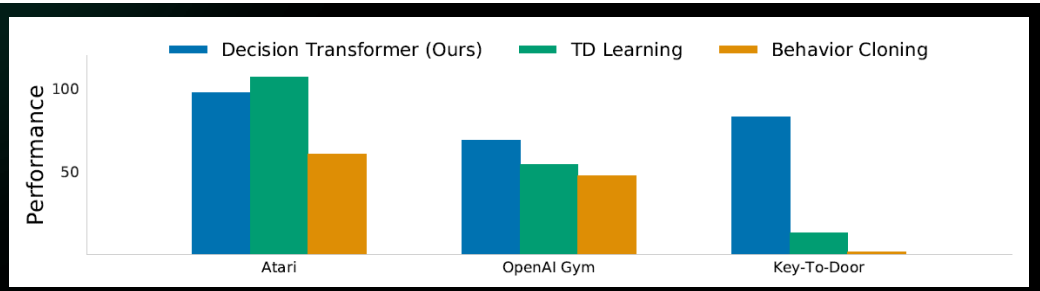
Минусы: сильное переобучение и слабая переносимость

Decision Transformer (DT)

- Использование трансформера на токенизированных экспертных траекториях
- Использование будущих вознаграждений для маркера качества
- Генеративная модель в качестве стратегии

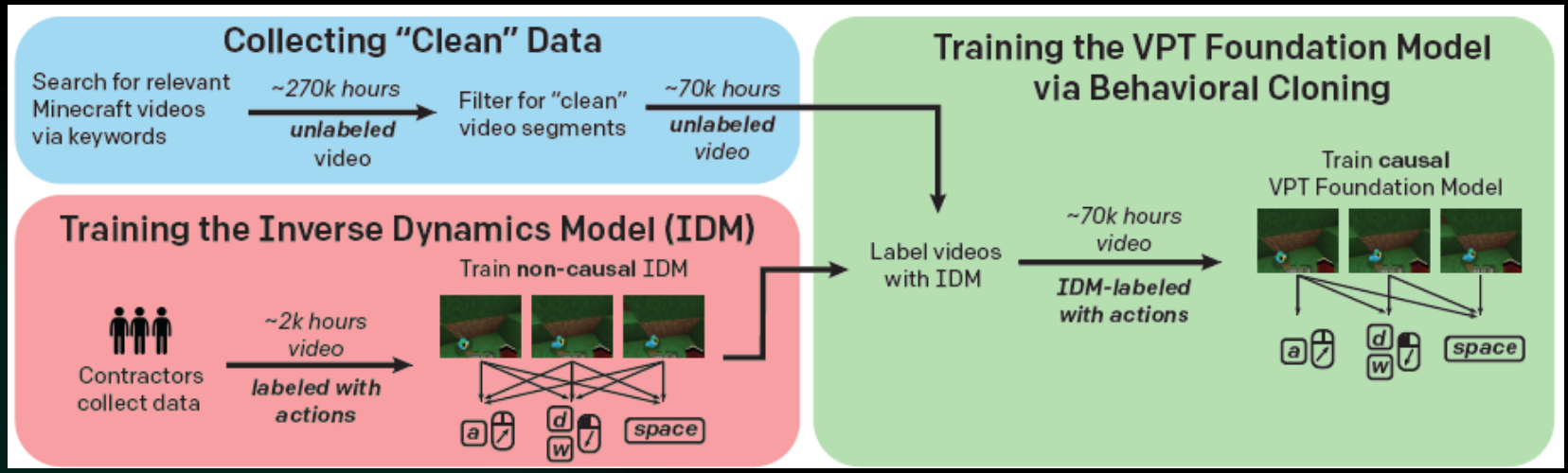


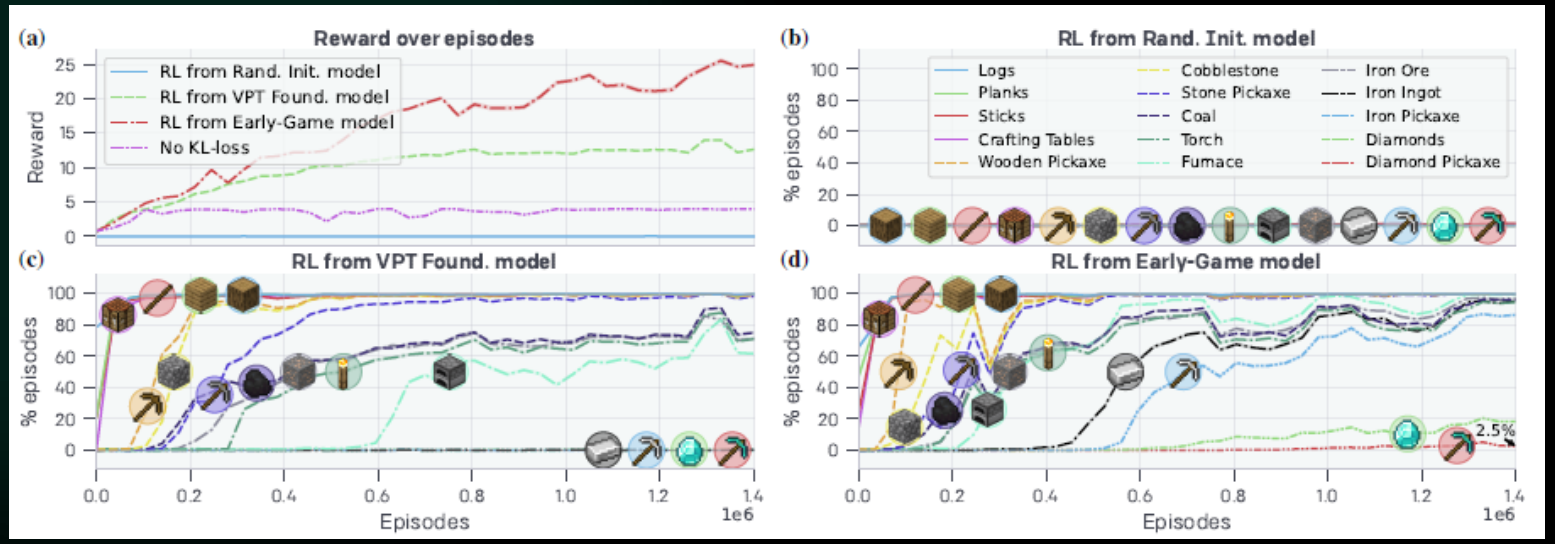




VPT

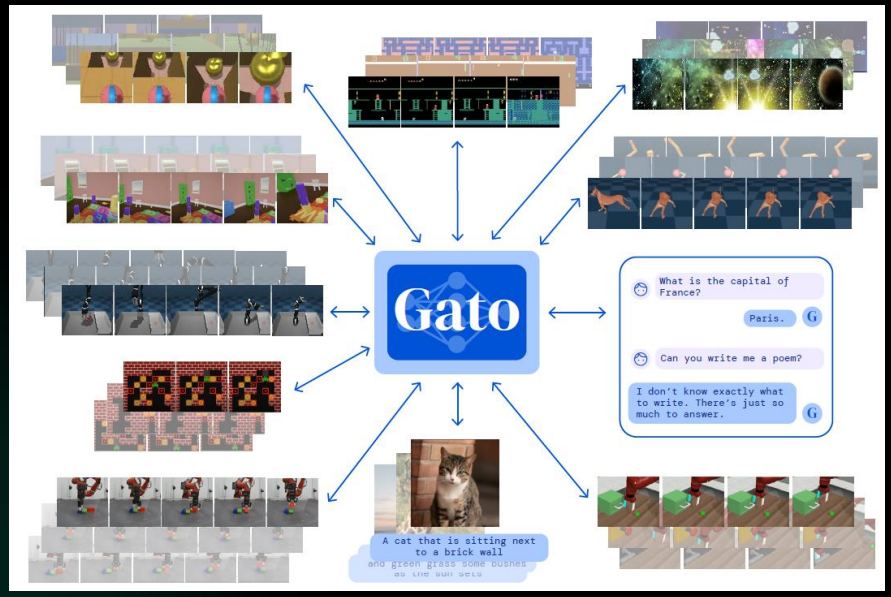
- Youtube – неисчерпаемый источник демонстраций
- Необходимо обучение обратной динамики по размеченным данным
- Эффективное применение трансформеров

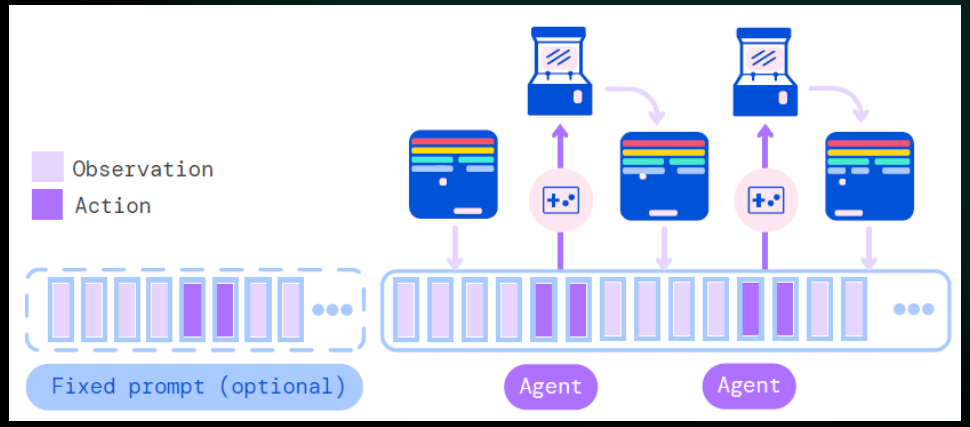


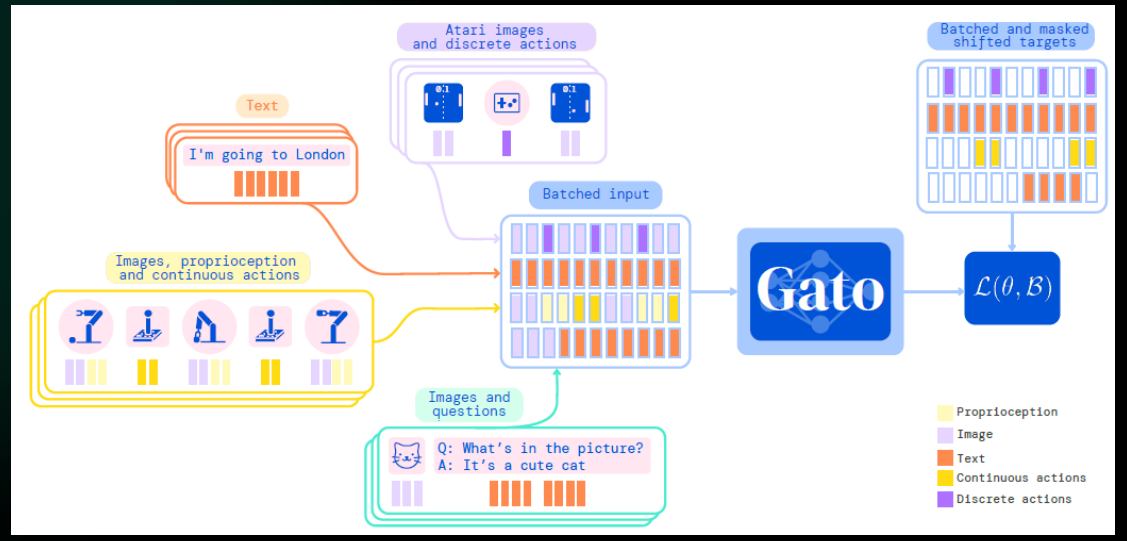


Gato

- Один трансформер на несколько задач, в том числе и без действий
- Универсальная затравка (prompt)
- Универсальная токенизация данных

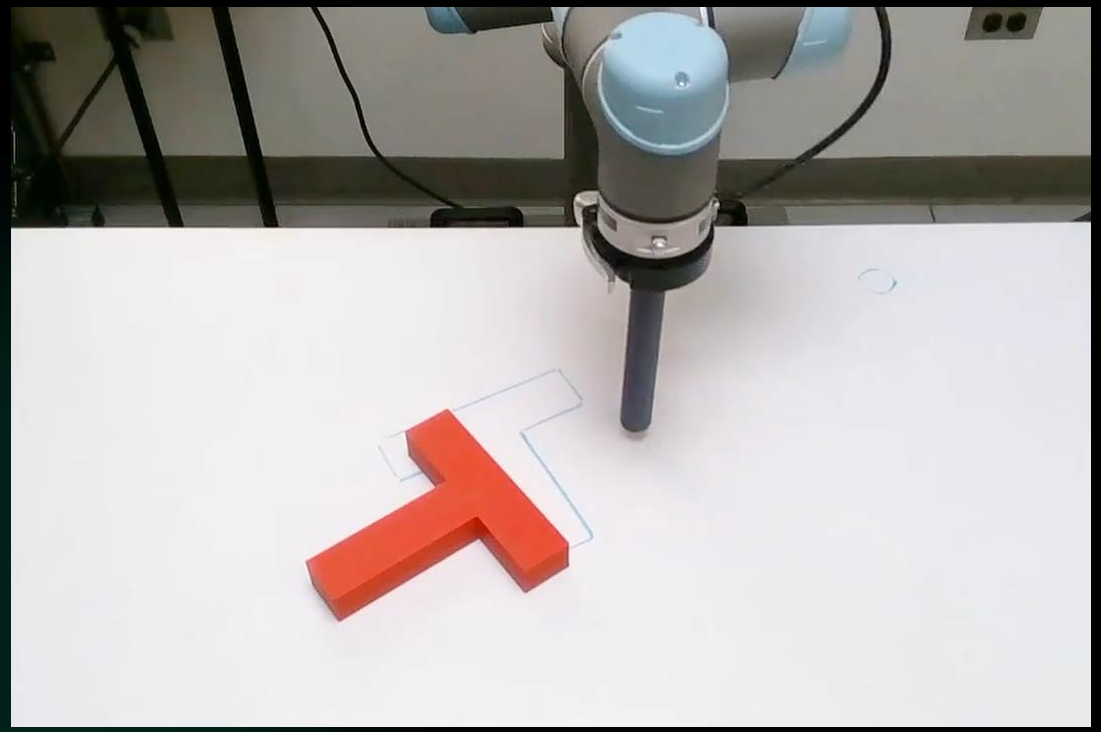


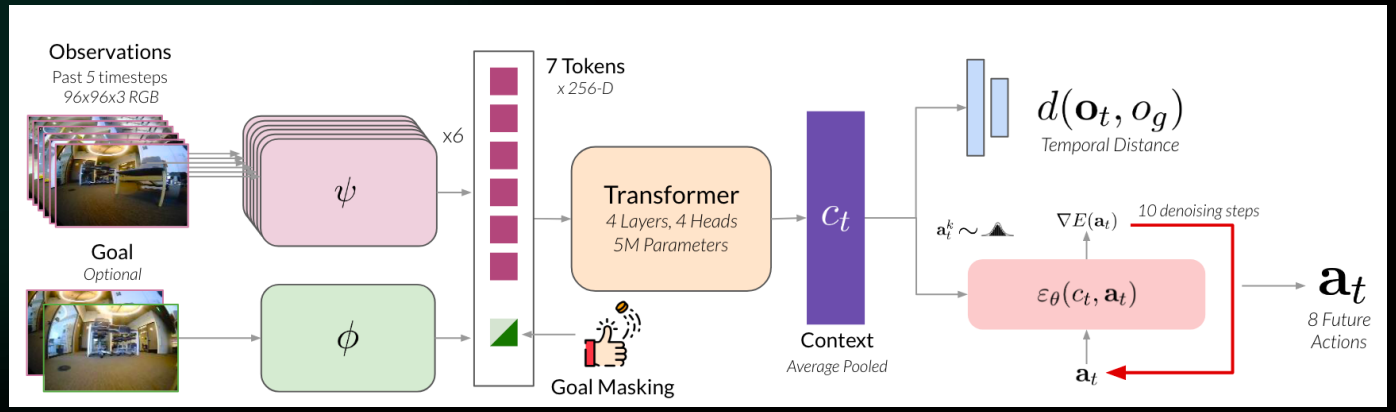




Диффузионная стратегия

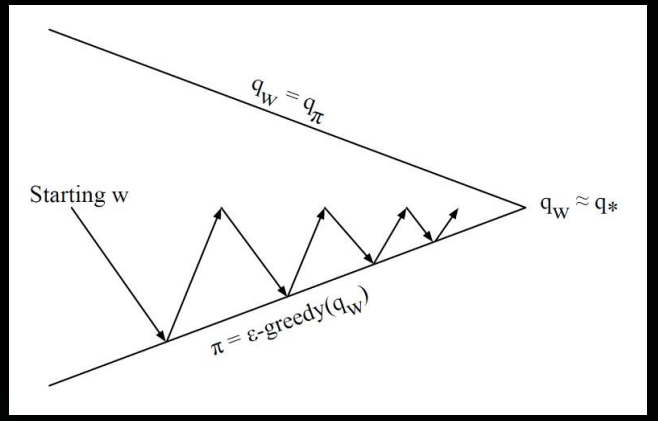
- Генеративная стратегия в виде диффузионного процесса
- Обуславливание на градиент полезности
- Необходимо использовать эффективные энкодеры

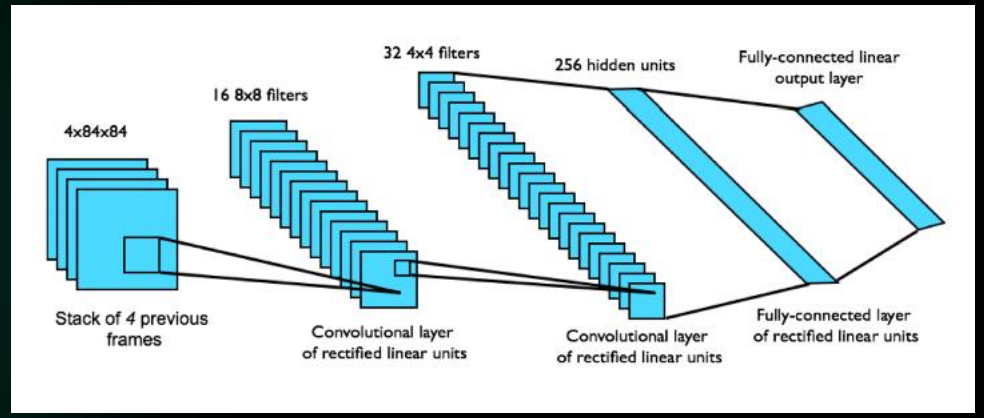




База 2: Q-обучение и DQN

- Обобщенная итерация по стратегиям – Q-обучение
- Аппроксимация состояния нейронной сетью – DQN
- Трюк #1 – память прецедентов (replay buffer)
- Трюк #2 – замороженная целевая полезность



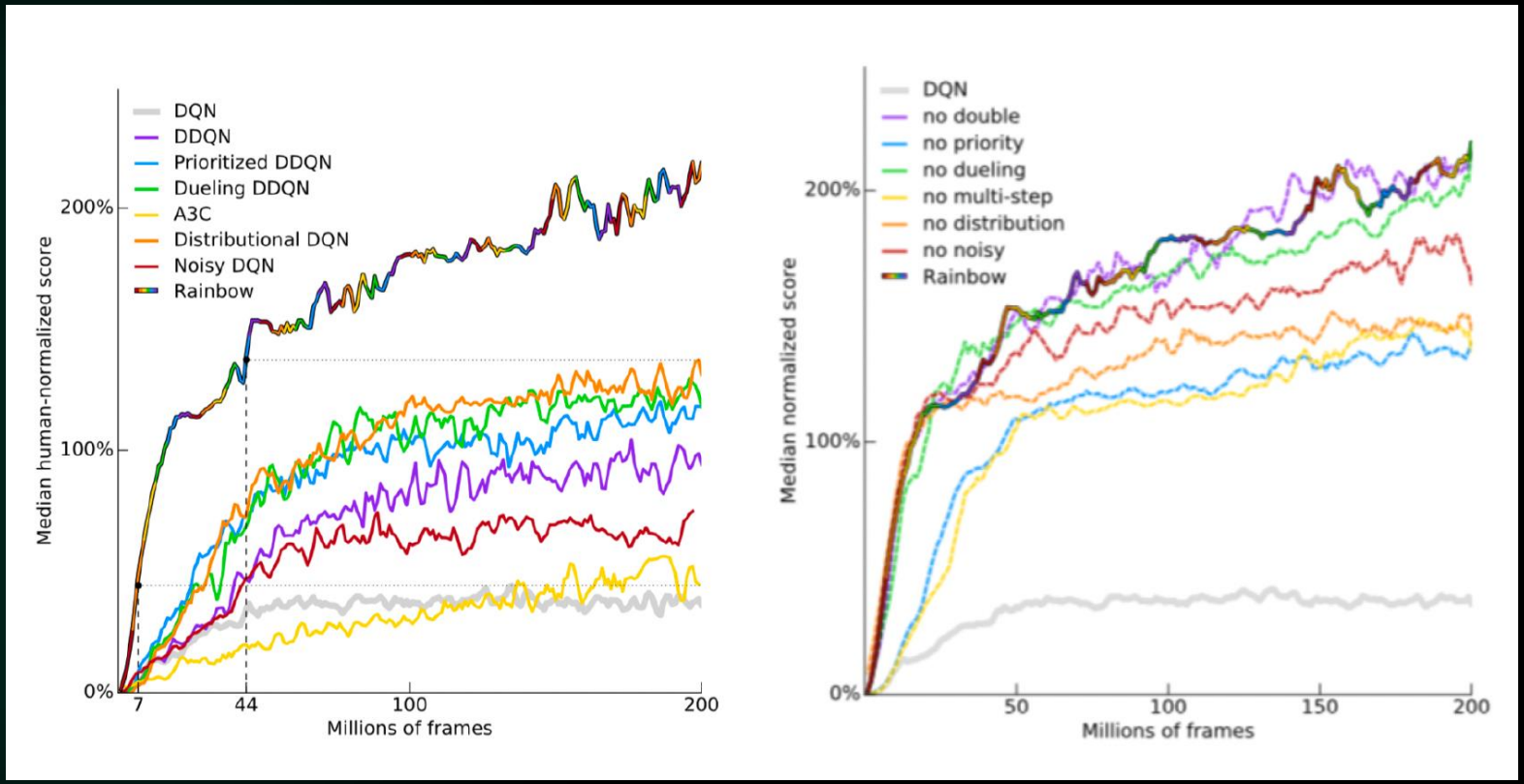


Плюсы: простота реализации и быстрая сходимость

Минусы: детерминированная стратегия, марковость

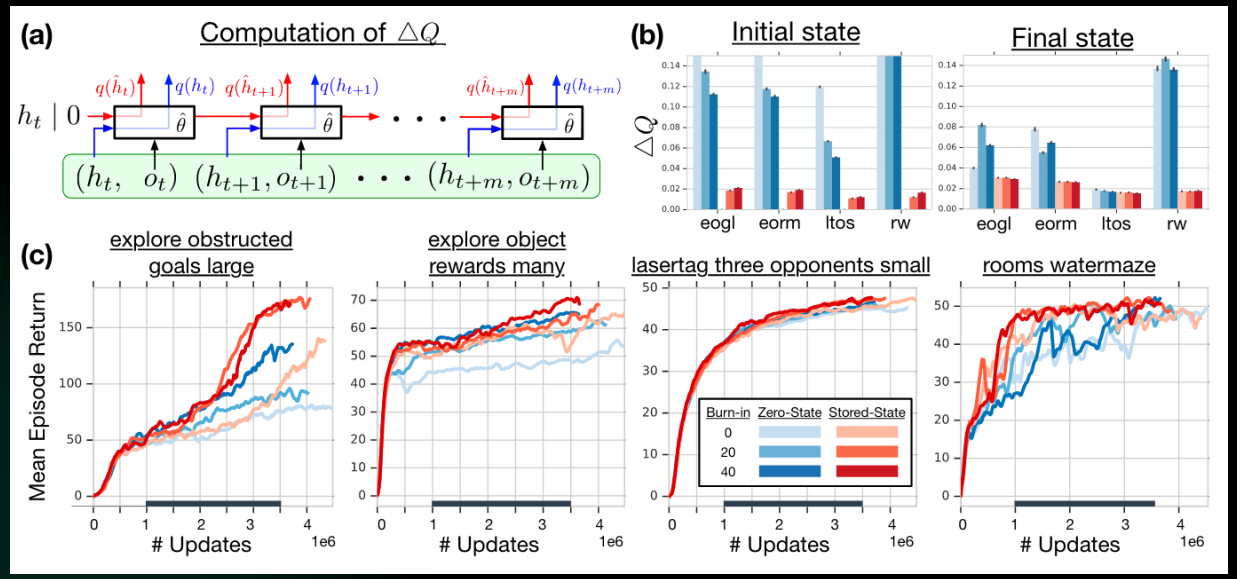
Rainbow

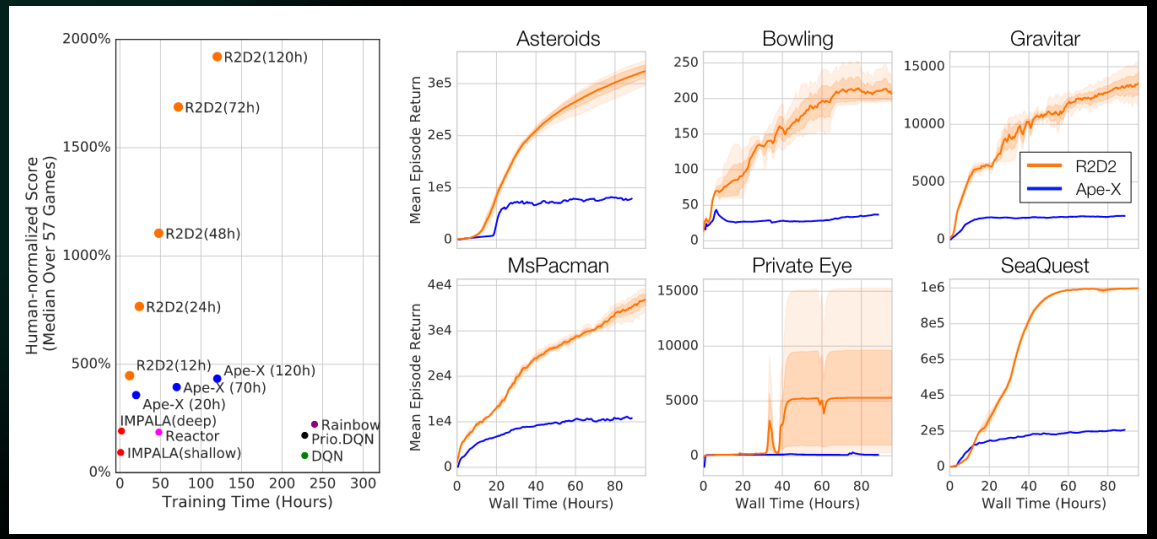
- Приоритезация памяти прецедентов
- Дуэльный DQN – разделение полезности действия и состояния
- Многошаговая оценка полезности
- Шумовые слои для исследования
- Распределение на функции полезности



R2D2

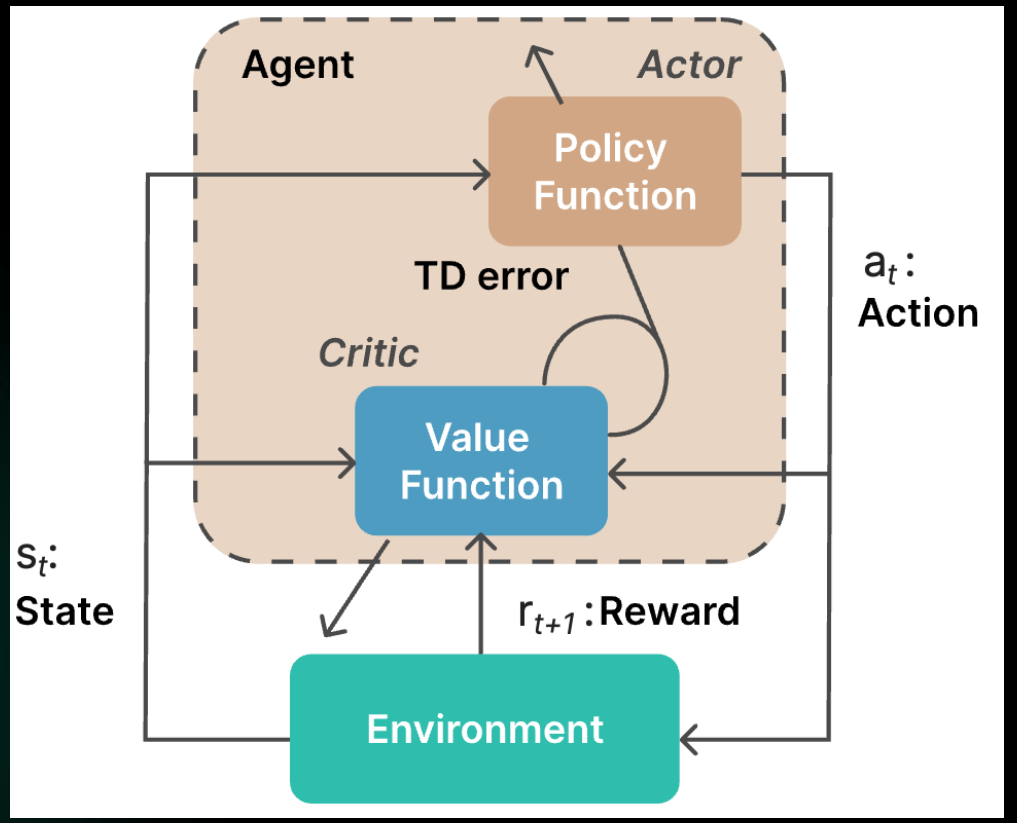
- Использования рекуррентной нейронной сети для аппроксимации состояния
- Распределенная память прецедентов: сбор данных многими акторами параллельно





База 3: Актор-критик

- Одновременная параметризация полезности и стратегии
- Функция полезности для стратегии
- Полезность от критика – как поправка к обновлению стратегии

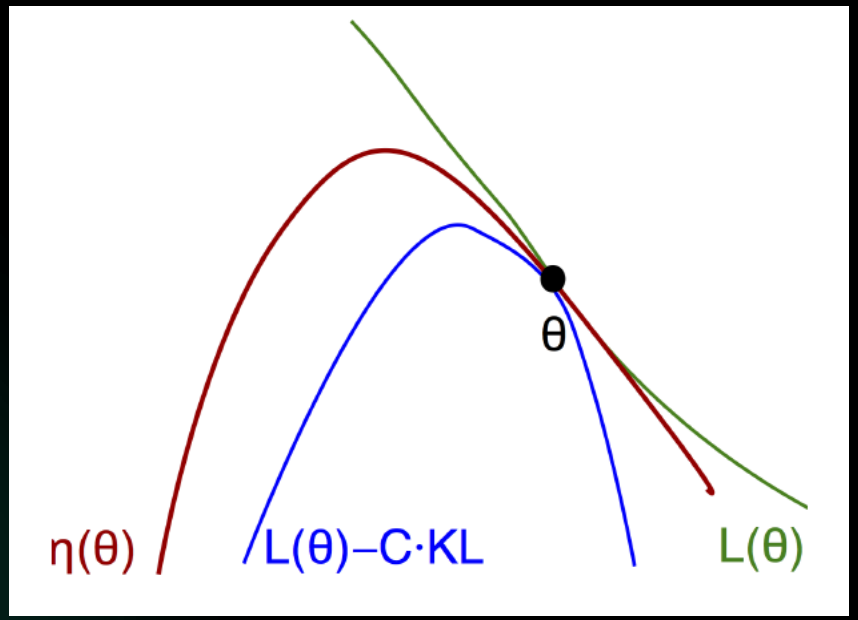


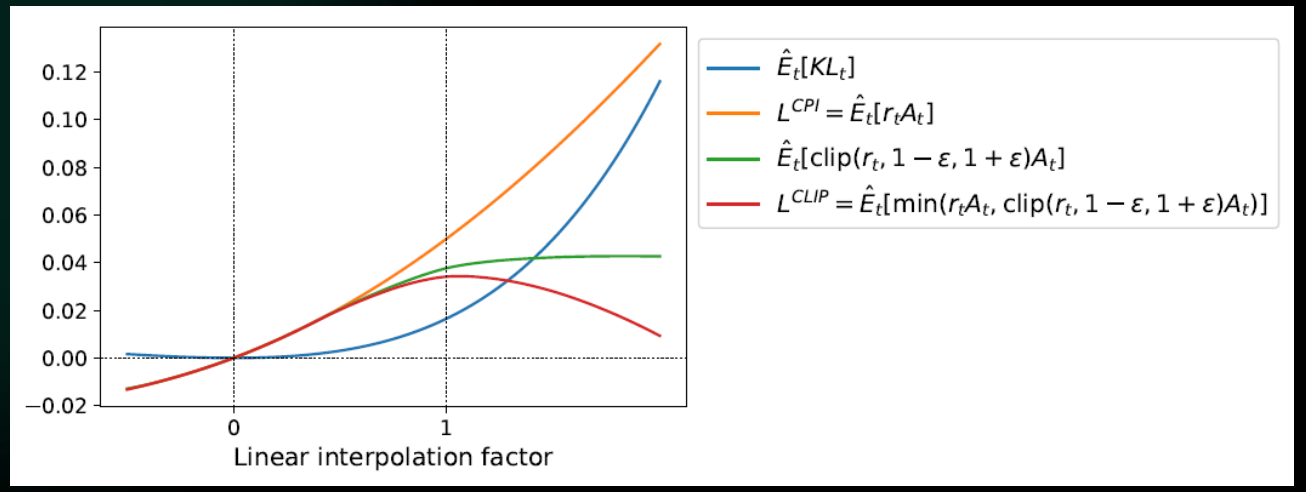
Плюсы: стохастическая стратегия

Минусы: нестабильное долгое обучение, совместимость критика

PPO

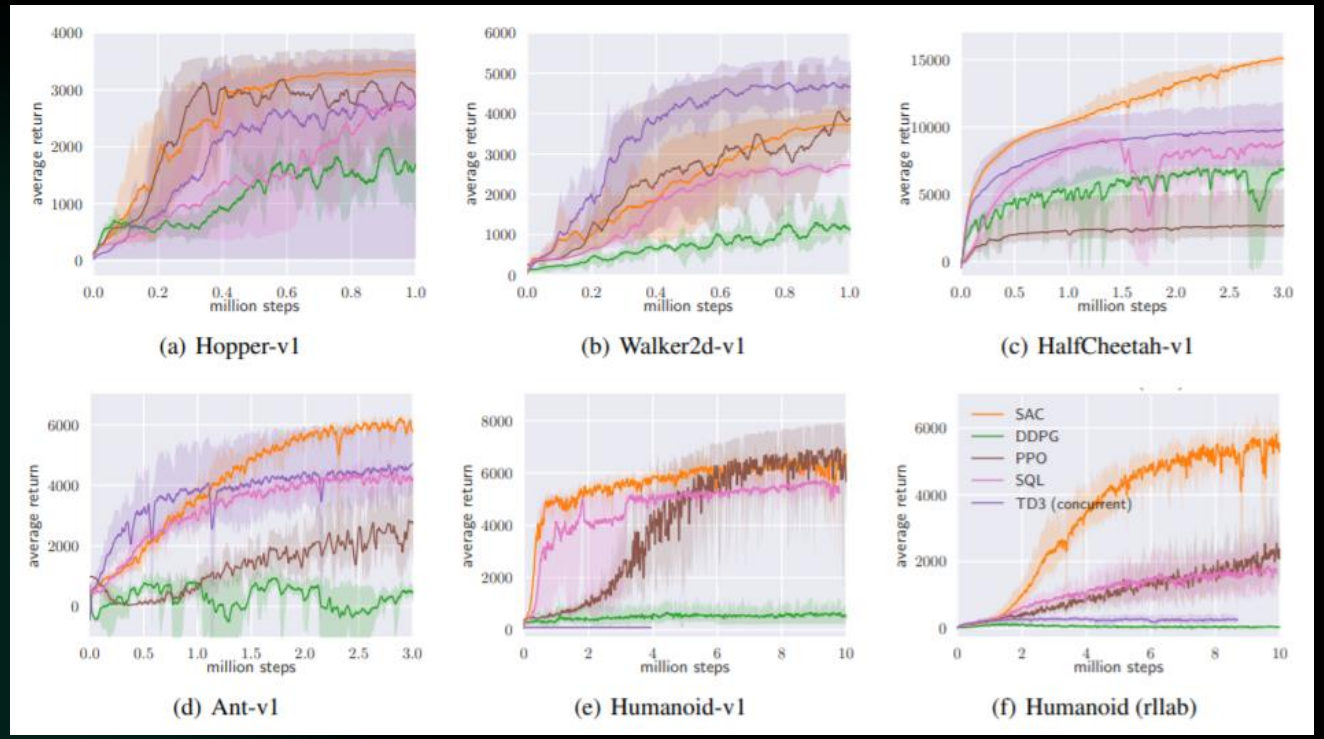
- Локальное приближение стратегии в малой окрестности
- Честное решение задачи оптимизации с ограничением (TRPO) – сложно и долго
- Эвристическая поправка функции потерь – усечение (clipping)

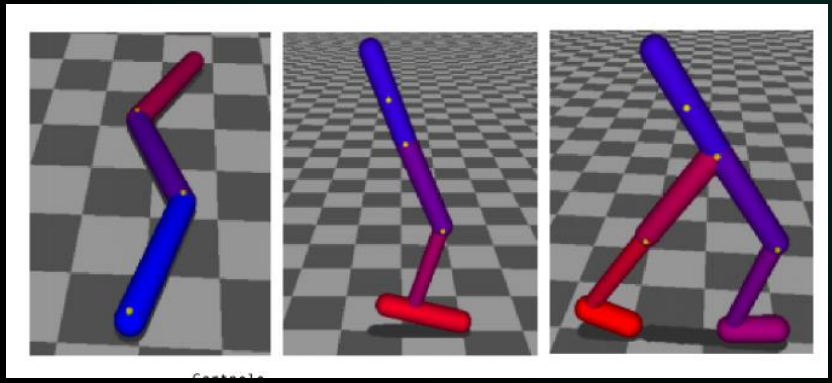




SAC

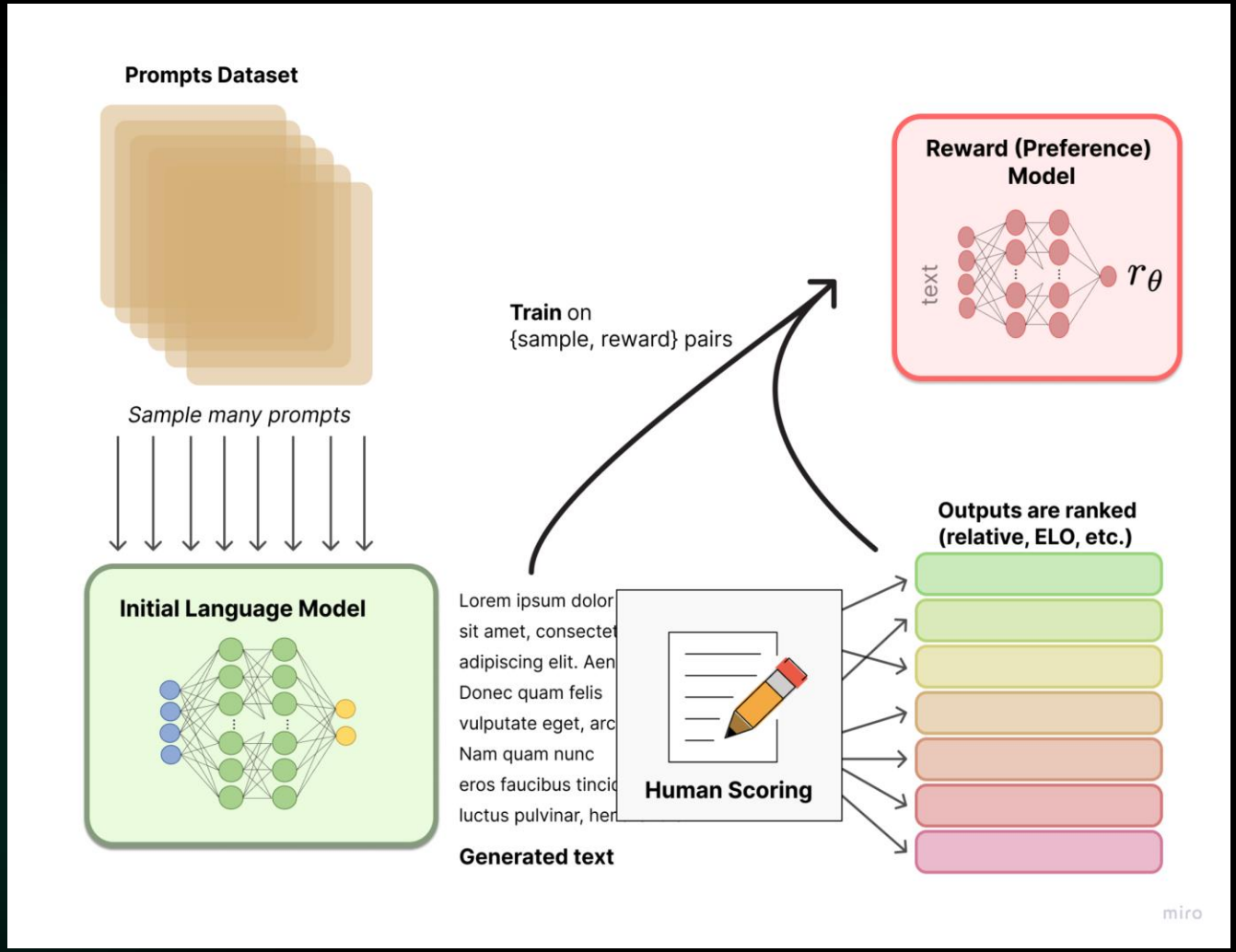
- Использование памяти прецедентов в схеме актор-критик
- Мягкое Q-обучение – регуляризация на энтропию стратегии
- Стратегия – стохастическая с максимизацией полезности





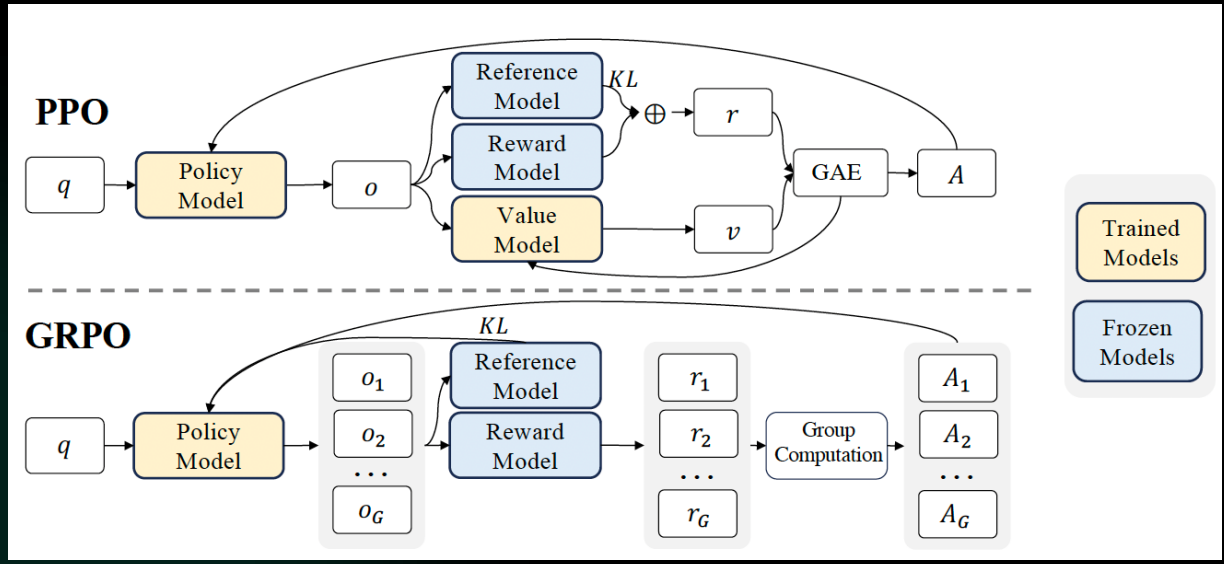
RLHF – дообучение языковых моделей

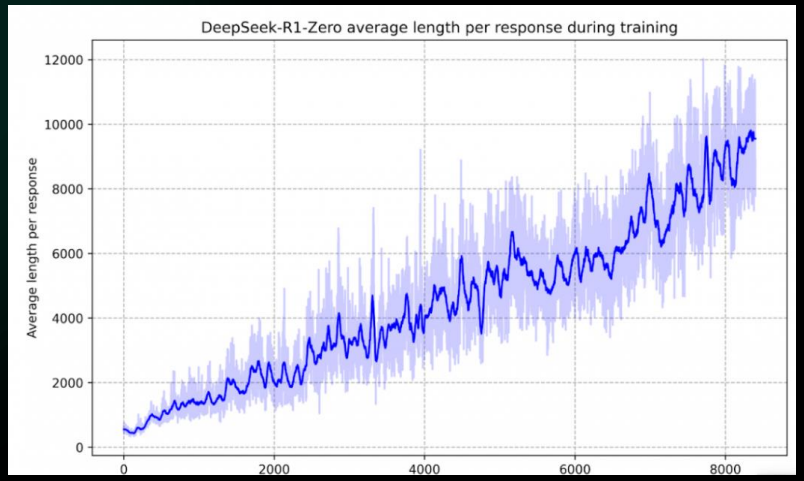
- Перенос предпочтений пользователей в модель
- Решение проблемы безопасности ответов языковой модели



GRPO – дообучение в DeepSeek без критика

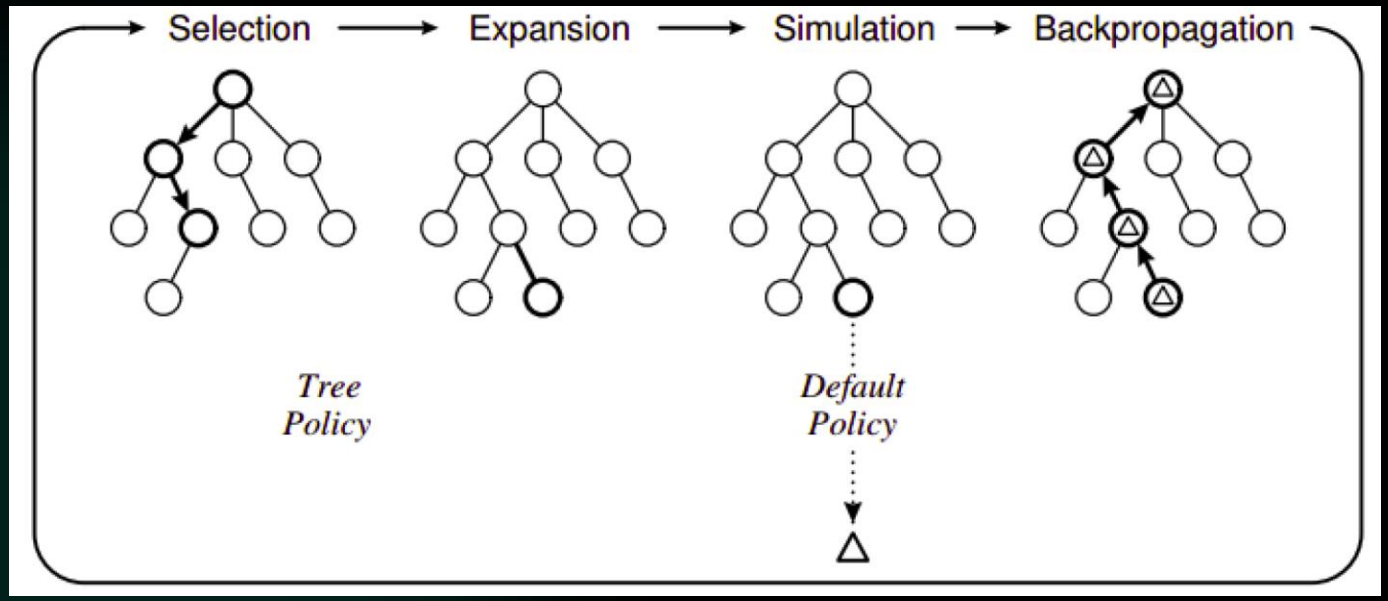
- Усреднение функции потерь по нескольким выходам модели
- Вместо критика используется усредненное вознаграждение





База 4: MCTS

- Прогнозирование состояний и вознаграждений в среде
- Планирование по предсказанному дереву вариантов
- В каждом узле обновляется статистика посещений

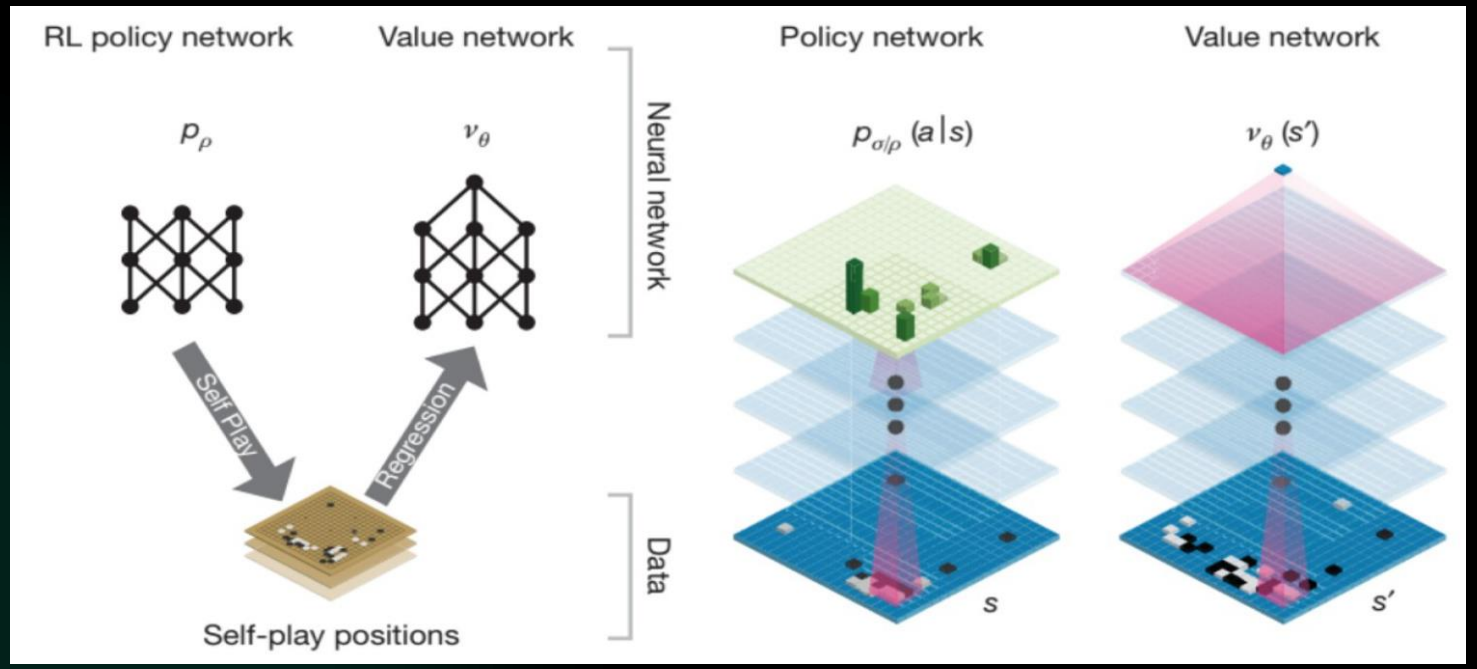


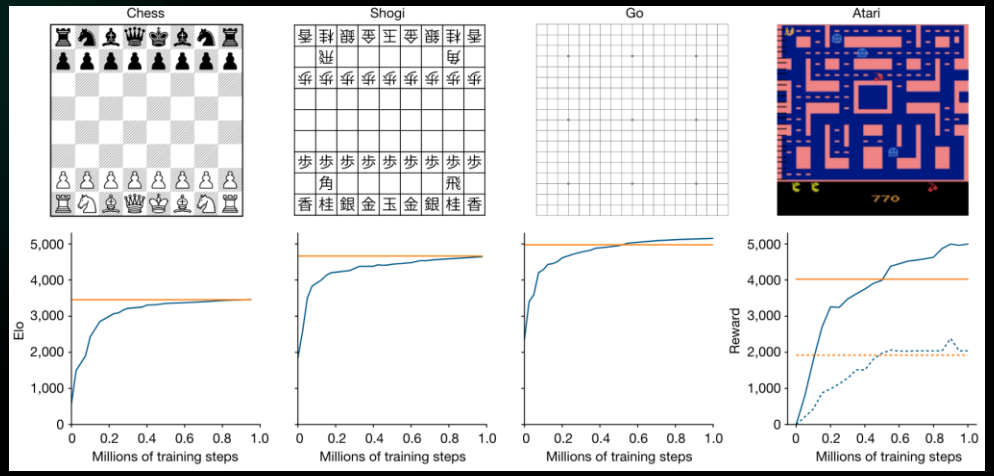
Плюсы: наиболее эффективный способ планирования для RL

Минусы: **работает только с дискретными состояниями, медленный**

AlphaZero

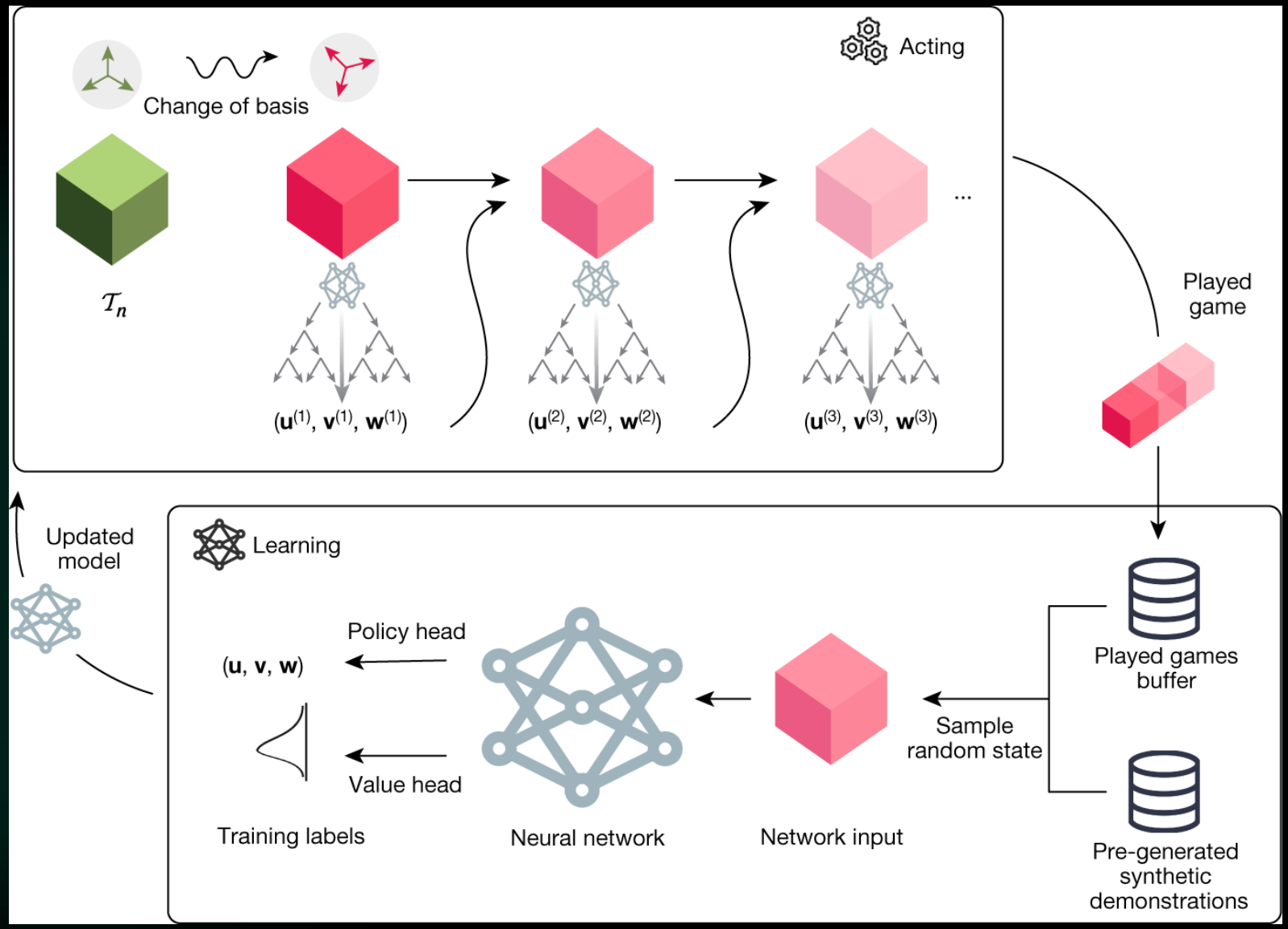
- Аппроксимация состояния нейронной сетью
- Предсказания действий и полезности с помощью нейросетевых актора и критика
- Развитие без известной модели - MuZero





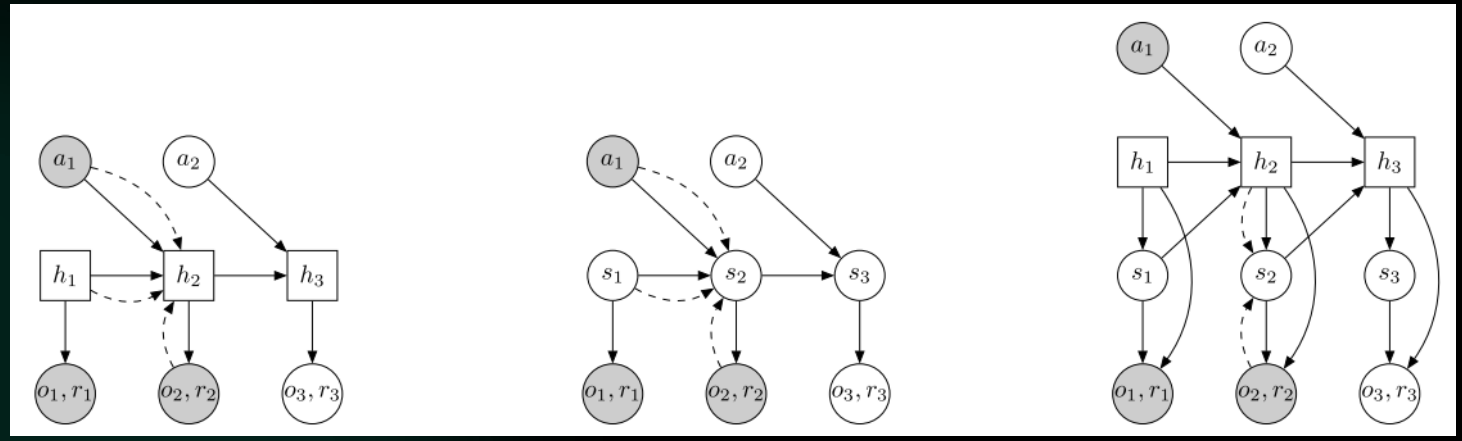
AlphaTensor

- Использование AlphaZero для поиска алгоритма умножения матриц
- Для случая 4x4 найден алгоритм, превосходящий по эффективности имеющийся 50-летней давности



База 5: Модель мира

- Частичная наблюдаемость и исходное пространство признаков изображения
- Модель динамики в латентном пространстве
- Непрерывное обучение за небольшое количество итераций

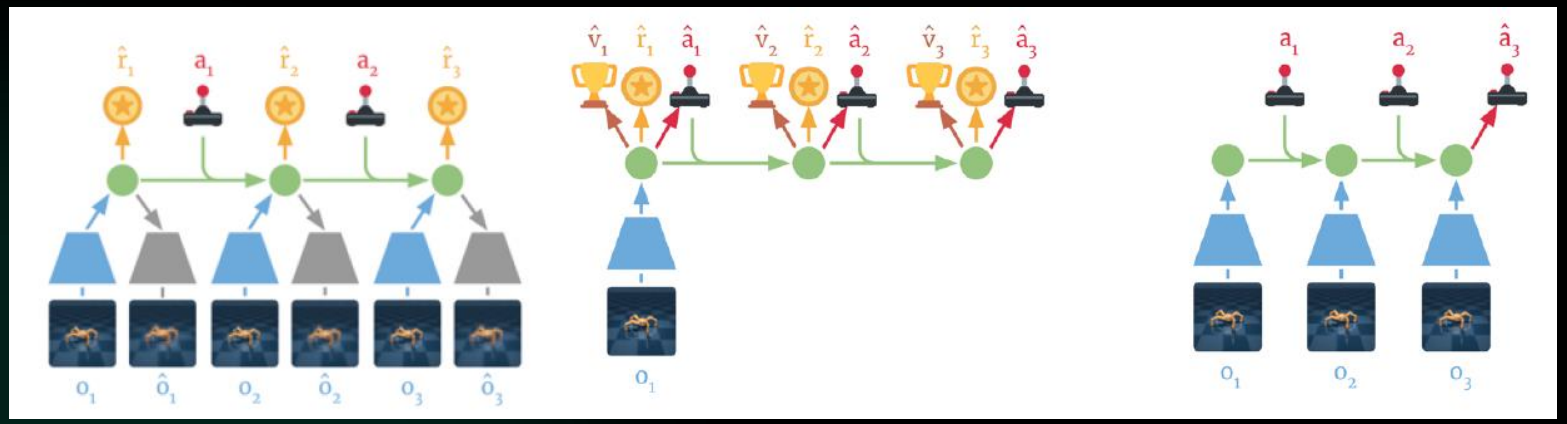


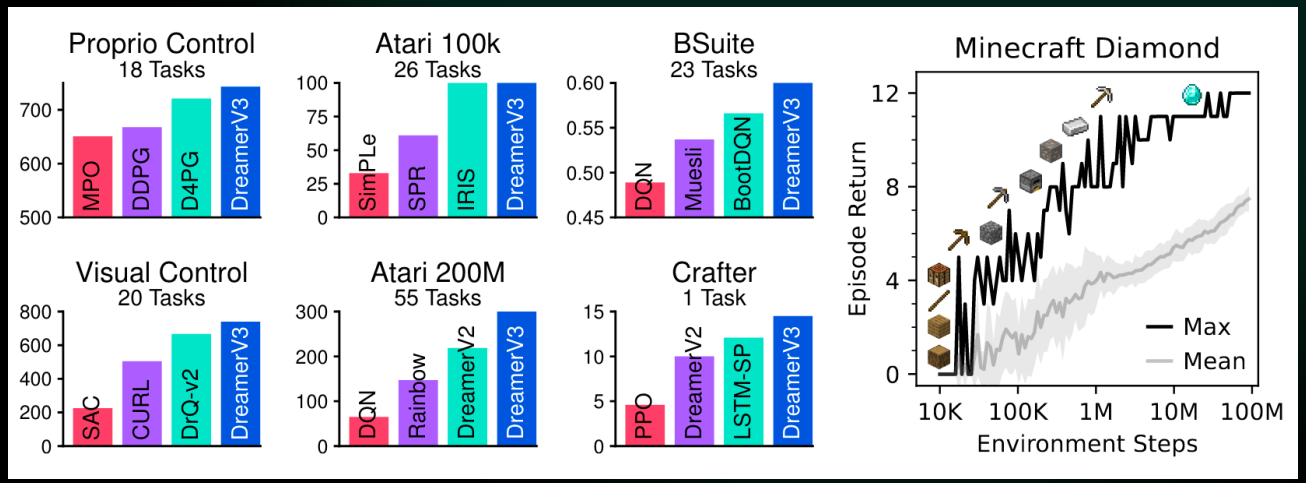
Плюсы: работа с наблюдениями высокой размерности

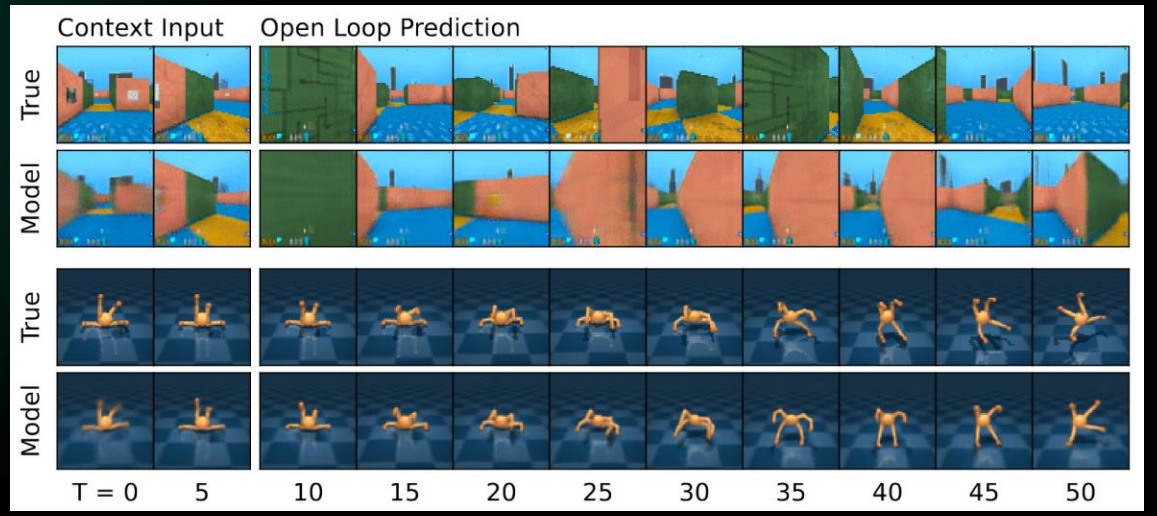
Минусы: **неустойчивость обучения, много гиперпараметров**

Dreamer

- Латентная модель мира на векторных представлениях
- Работает и с дискретными и с непрерывными действиями

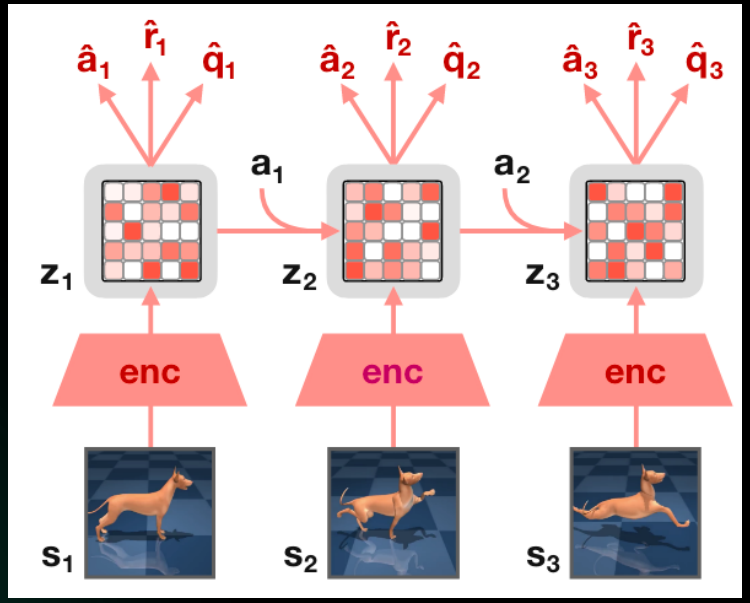


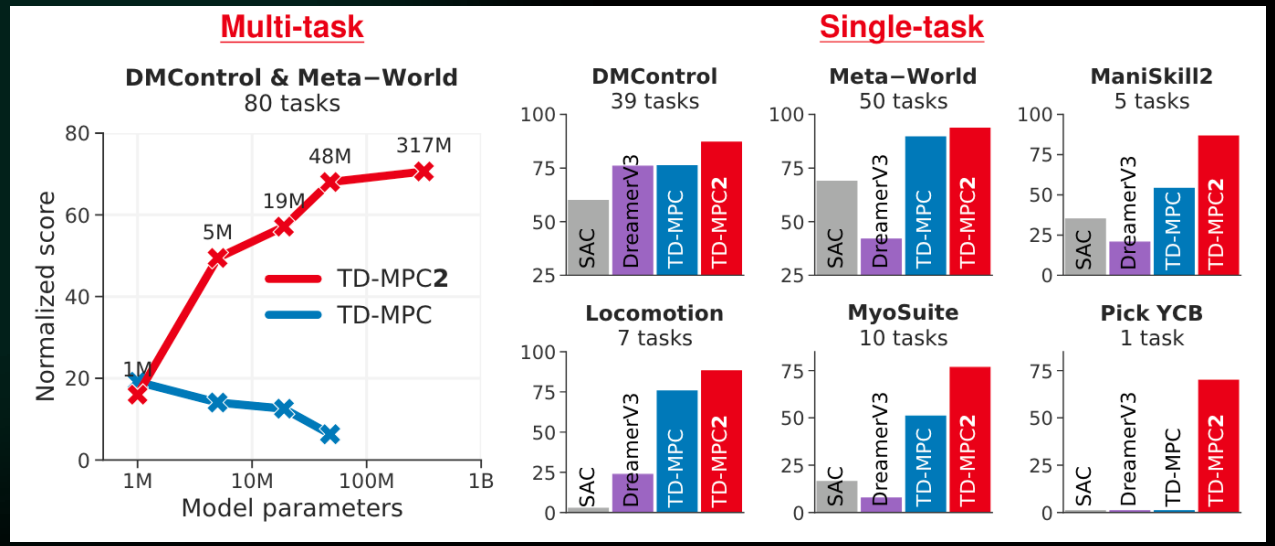




TD-MPC

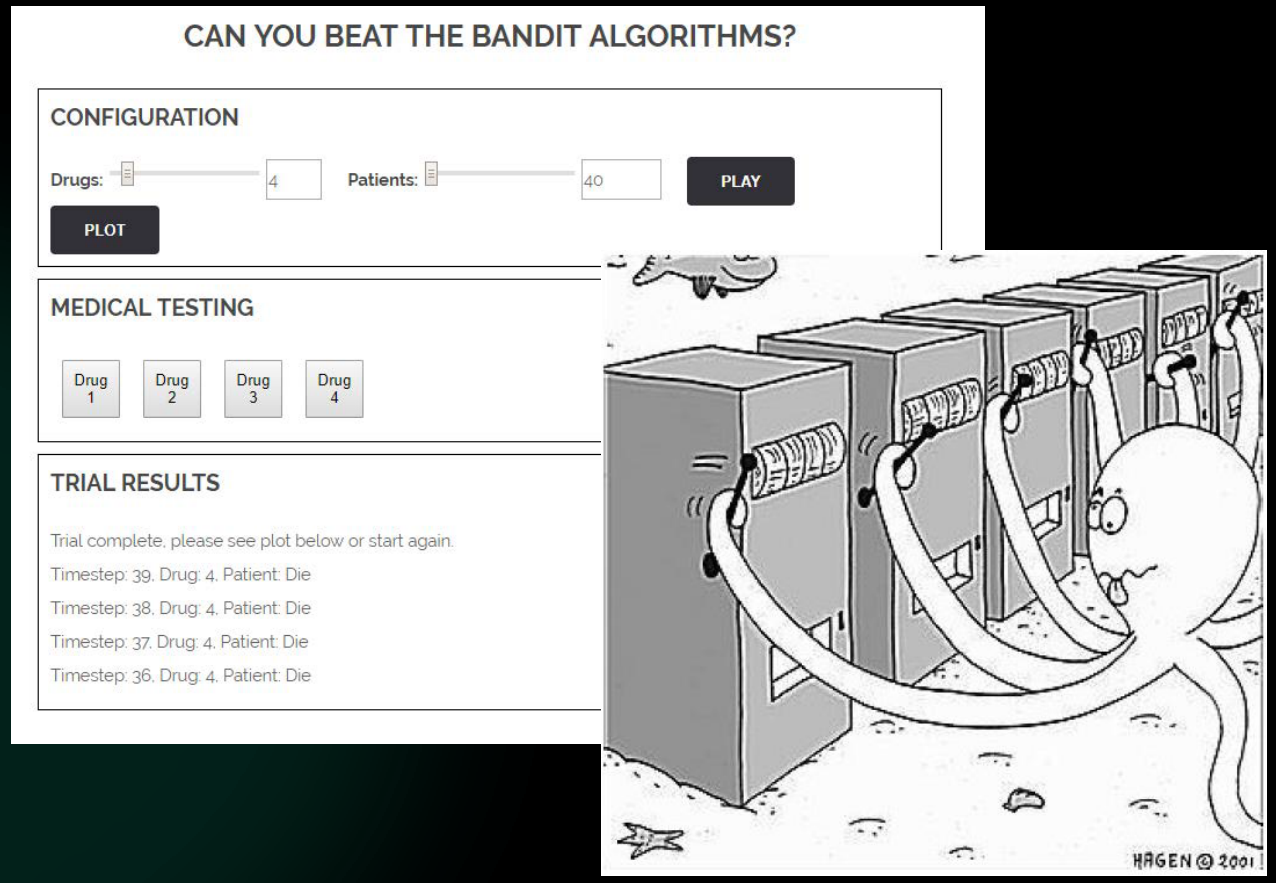
- Простая модель мира без рекуррентной сети
- Предсказание в режиме MPC (MPPI реализация) – оптимизация без производных
- Эффективная работа с векторными наблюдениями





База 6: многорукие бандиты

- Пример с подбором лекарства
- Одношаговый эпизод – действие и вознаграждение
- Решение задачи исследования/использования на множестве эпизодов
- Оптимизм в условиях неопределенности

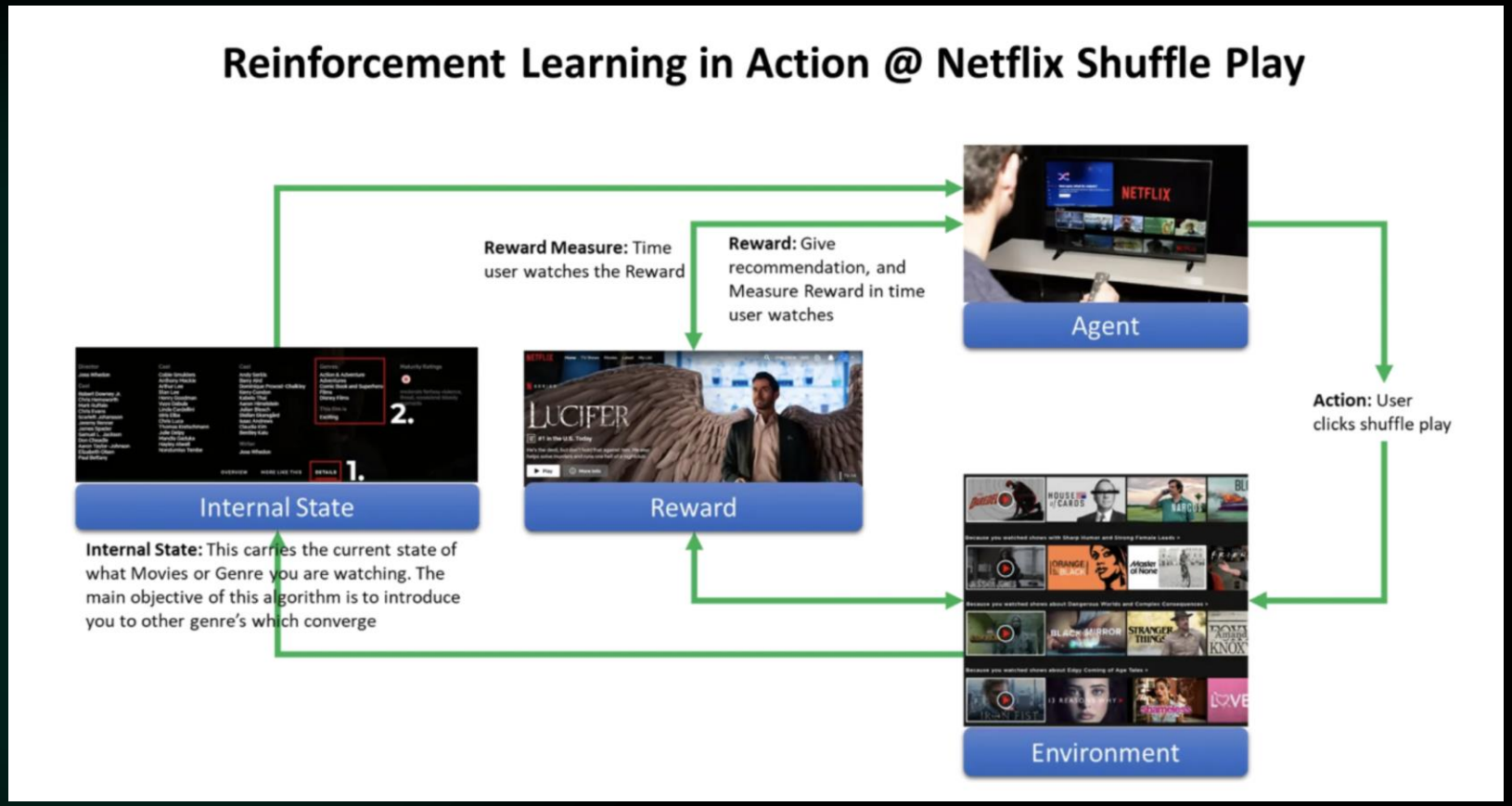


Плюсы: возможность строить теоретические оценки

Минусы: узкая применимость

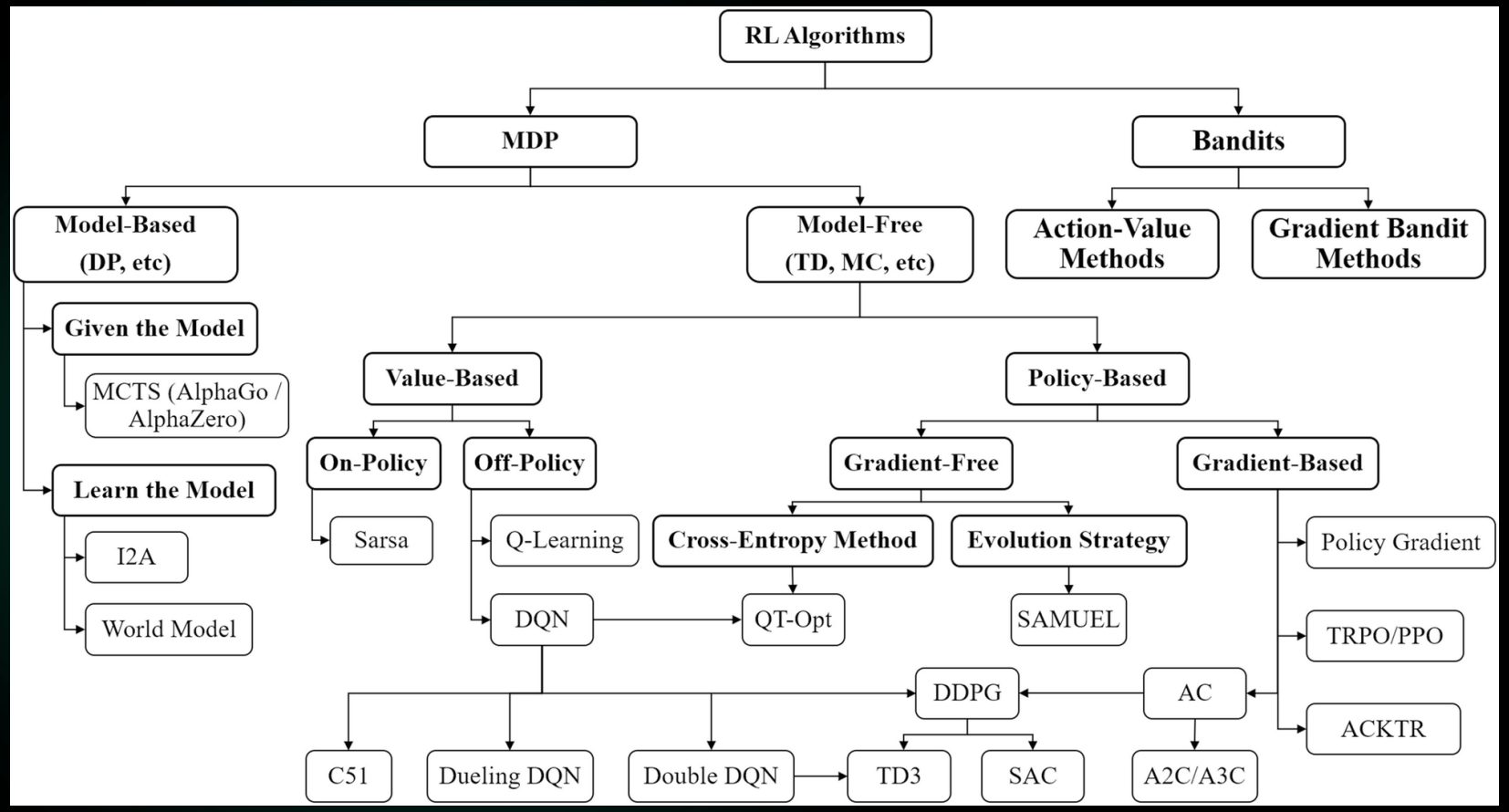
Контекстные бандиты

- Добавление начального состояния в одношаговый эпизод
- Удачное применение в рекламе и рекомендациях
- Возможность использовать с аппроксиматорами



Какие задачи помогает решать?

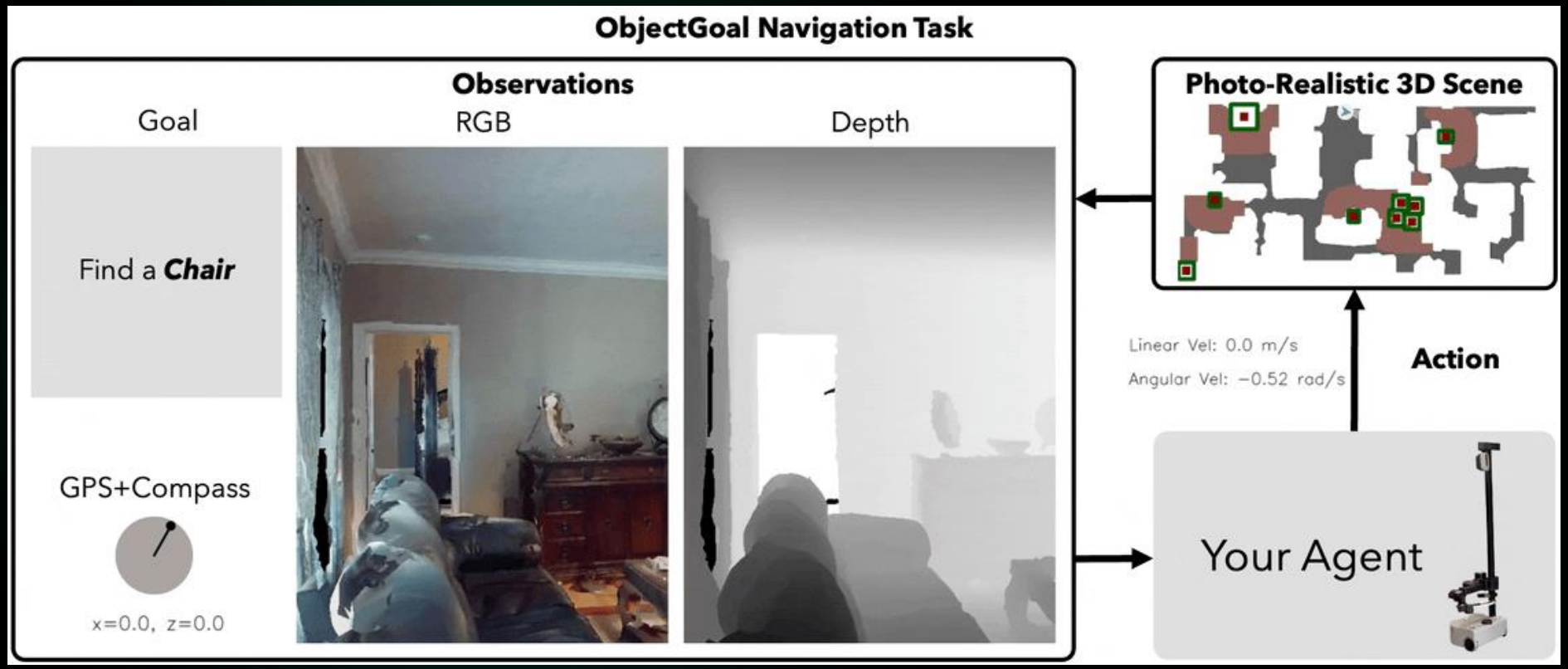
RL алгоритмы – общий взгляд



Нужен ли RL – анализ задачи

- Есть ли возможность отделить агента от среды?
- Есть ли действия и пространство состояний?
- Может ли задача быть решена методом проб и ошибок?
- Есть ли вознаграждения, в т.ч. отложенные?
- Формулируется ли эта задача как задача управления?

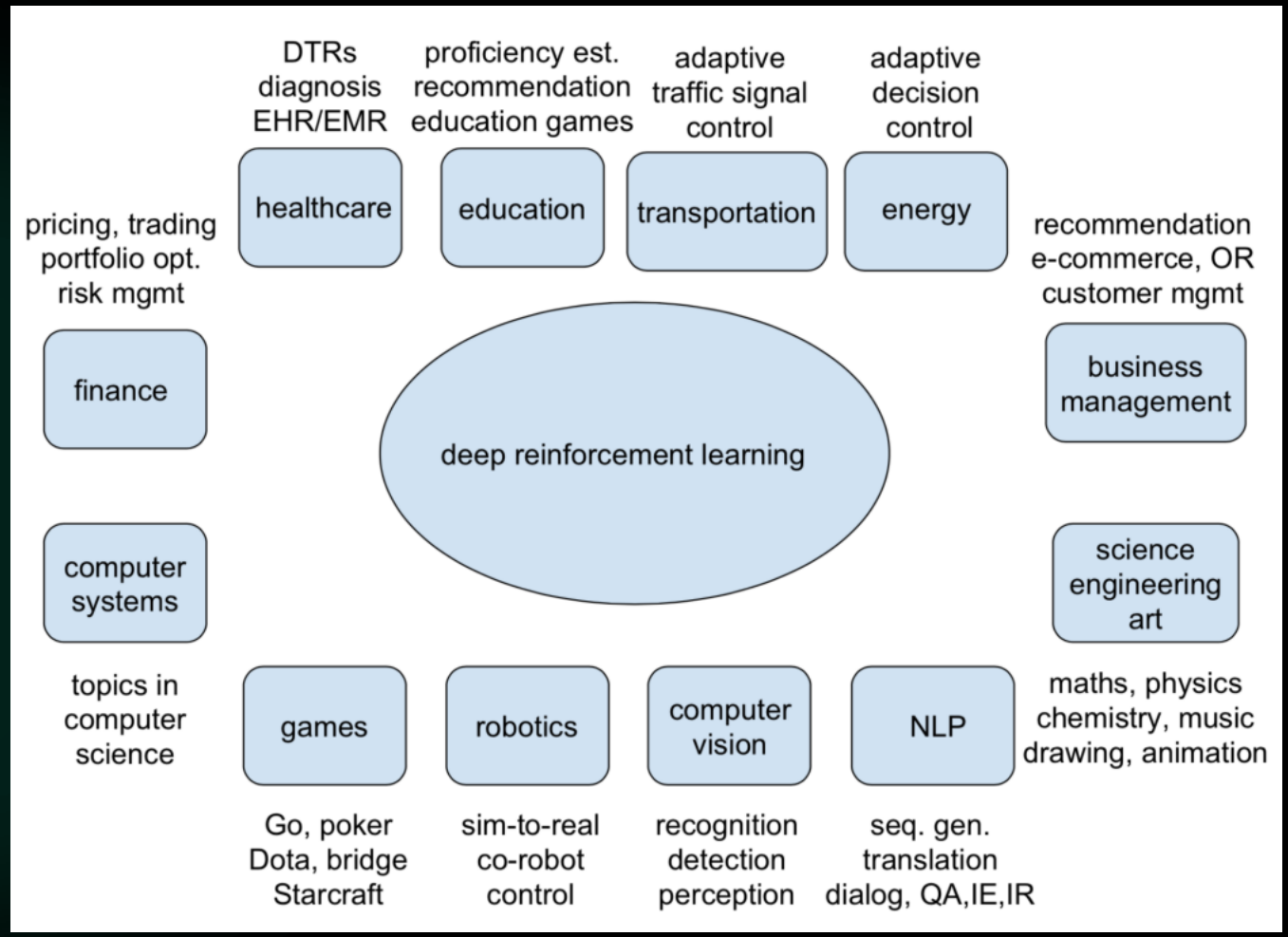
Есть ли симулятор?



Критерии выбора алгоритмов

- Короткие одношаговые эпизоды → (контекстные) бандиты
- Дискретные действия →
  - простые (векторные) состояния → Q-обучение
  - сложные (картиночные) состояния →DQN, ApeX
  - простые состояния и сложная динамика → PPO
- Непрерывные действия
  - любые состояния → PPO
- Сложное для исследования векторное пространство состояний→SAC
- Сложное для исследования картиночное пространство состояний→ICM, Hindsight, RIG
- Необходимость в мультизадачности→Dreamer
- Наличие быстрого симулятора →MCTS
- Дорогой симулятор, наличие демонстраций →BC,DT

Потенциальные области применения

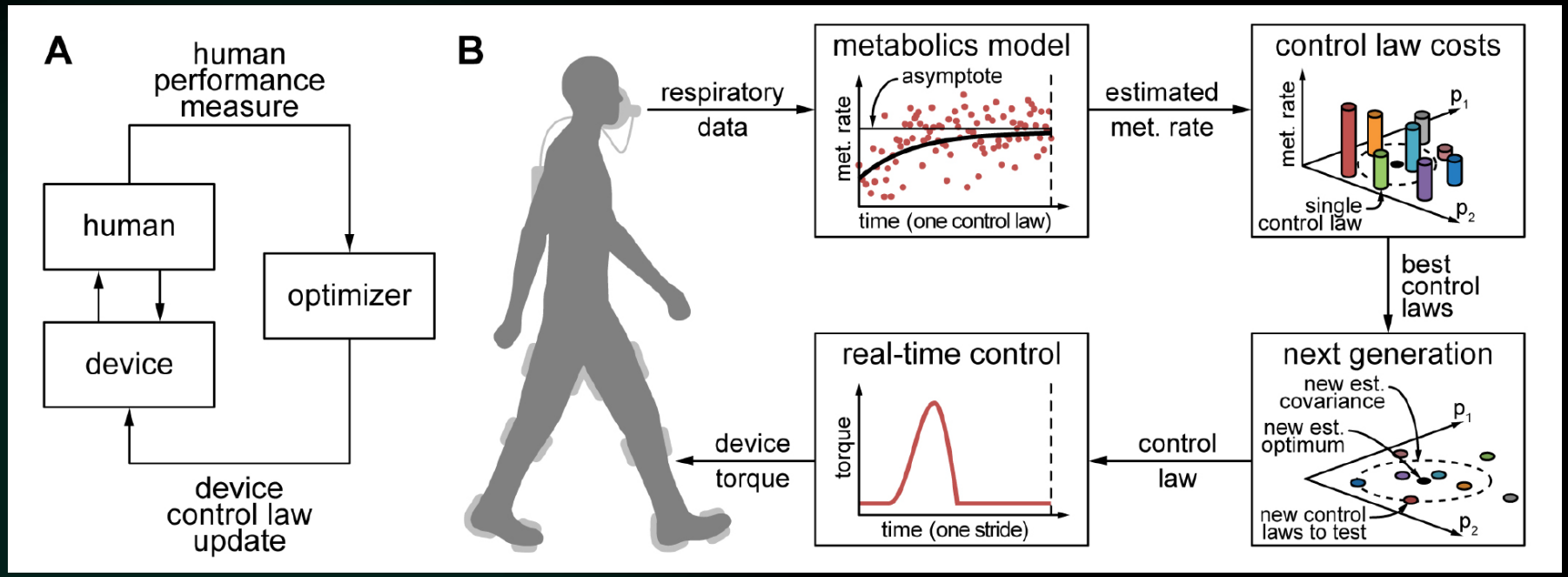


Примеры

Экзоскелет

Оптимизация параметров экзоскелета по метаболизму пользователя

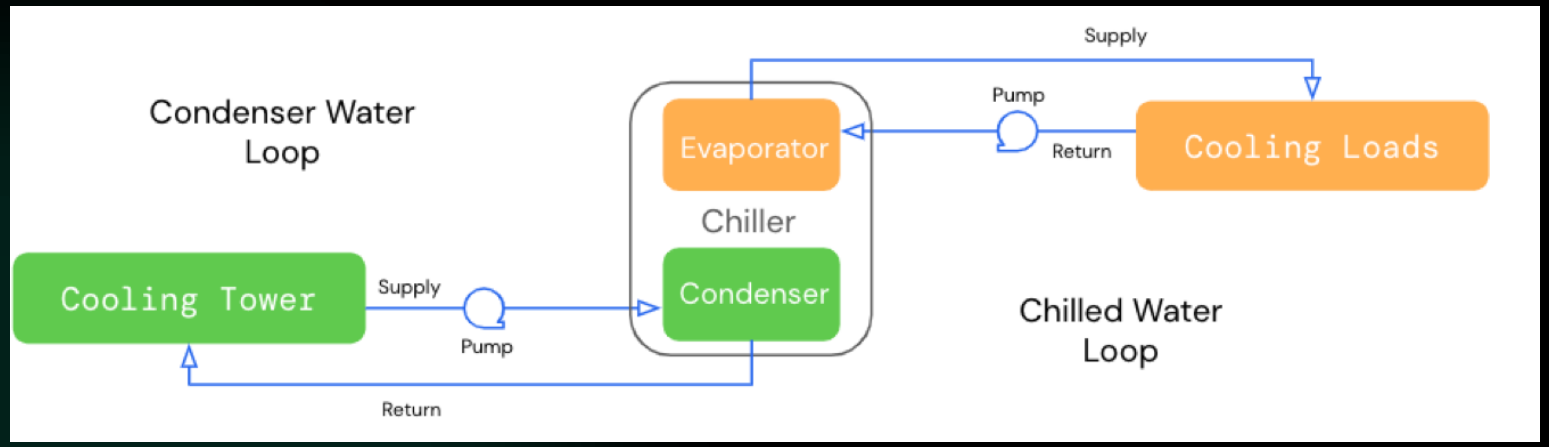
Science 2017

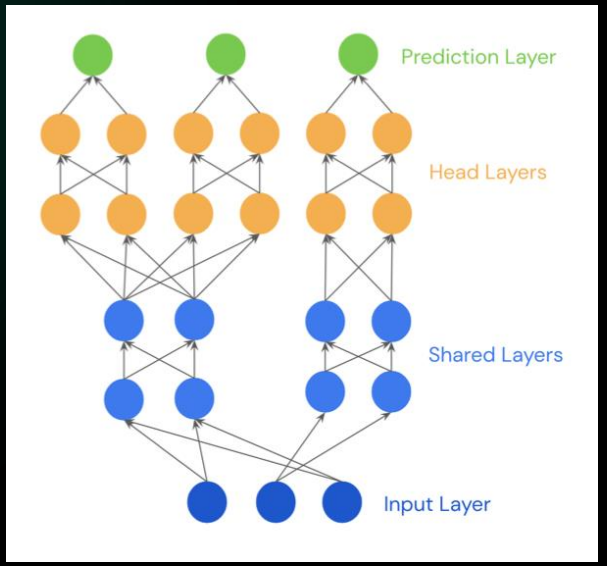


Управление системой кондиционирования

Управление системой для большого здания – экономия в энергозатрата в 9% и 13% для двух локаций

DeepMind 2022

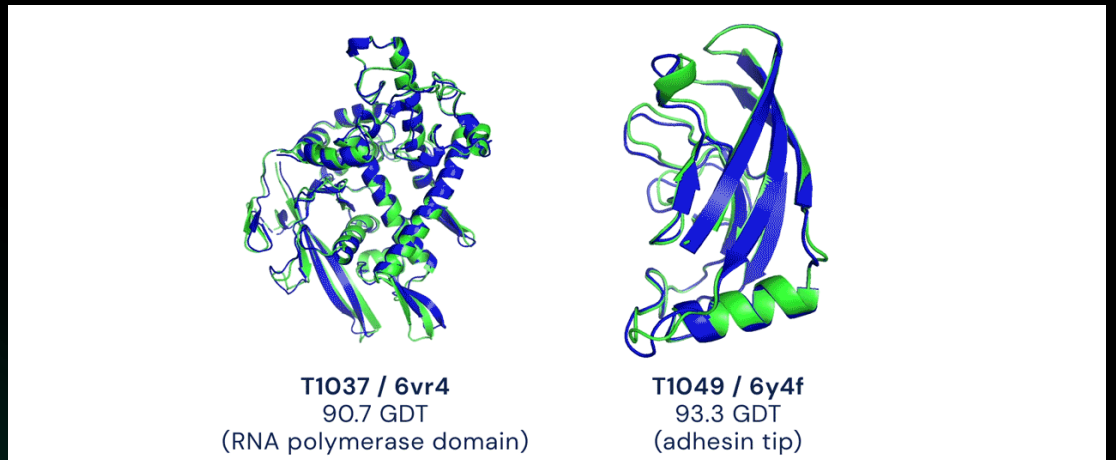


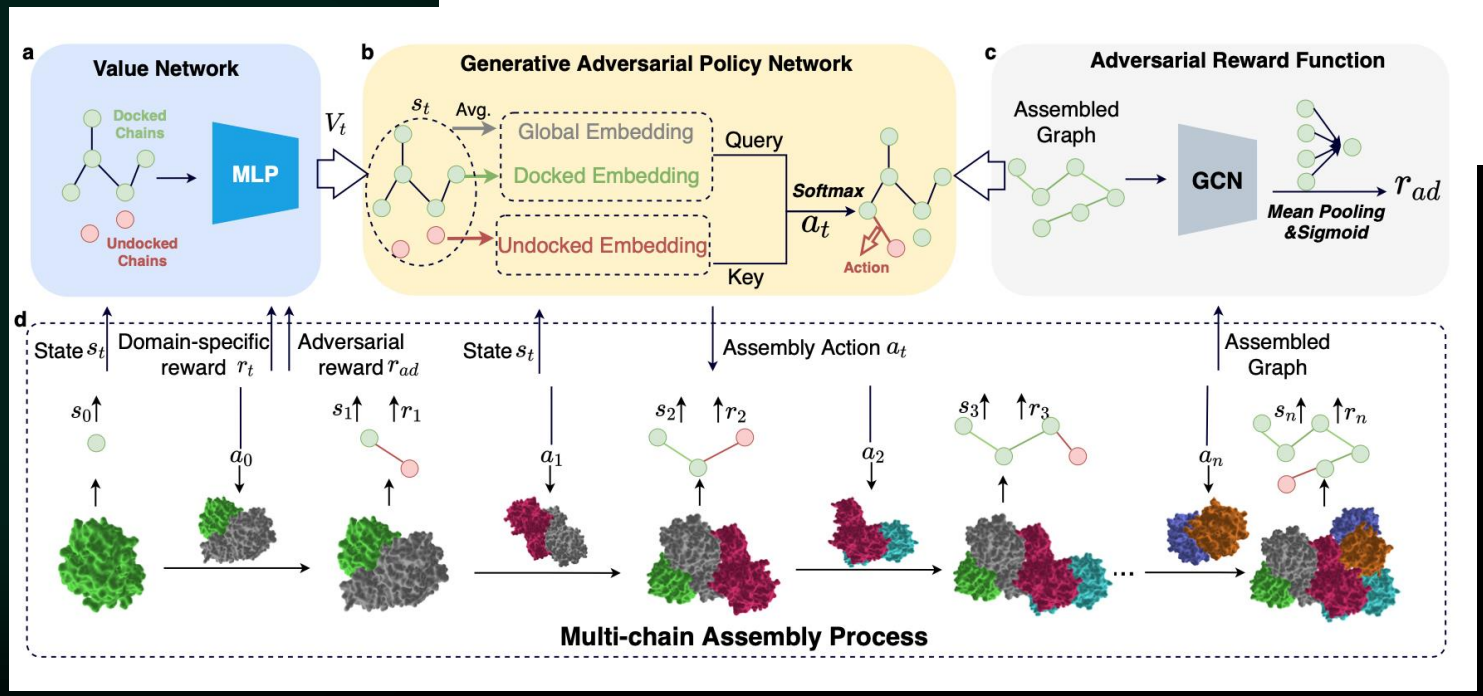


Определение структуры белка

AlphaFold - определение трехмерной структуры белка по аминокислотной последовательности

HKUST 2024

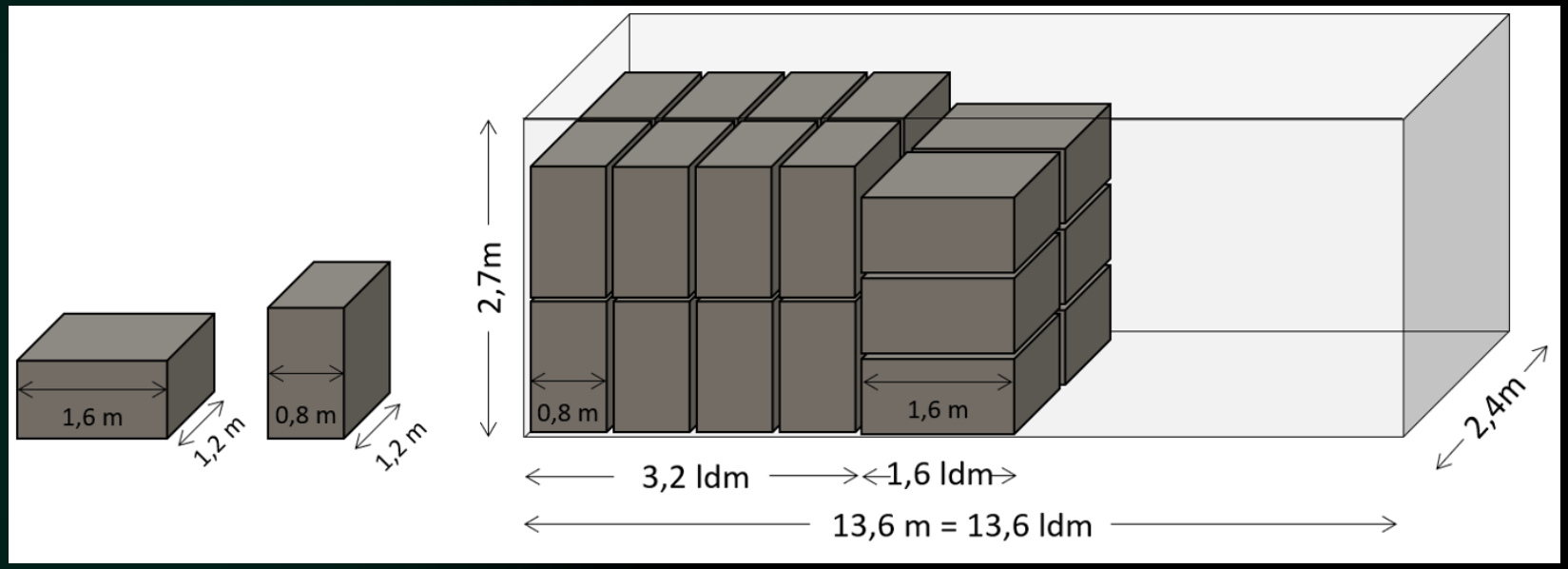




Задача упаковки

Комбинаторная задача
– подбор объектов для
полной загрузки палеты

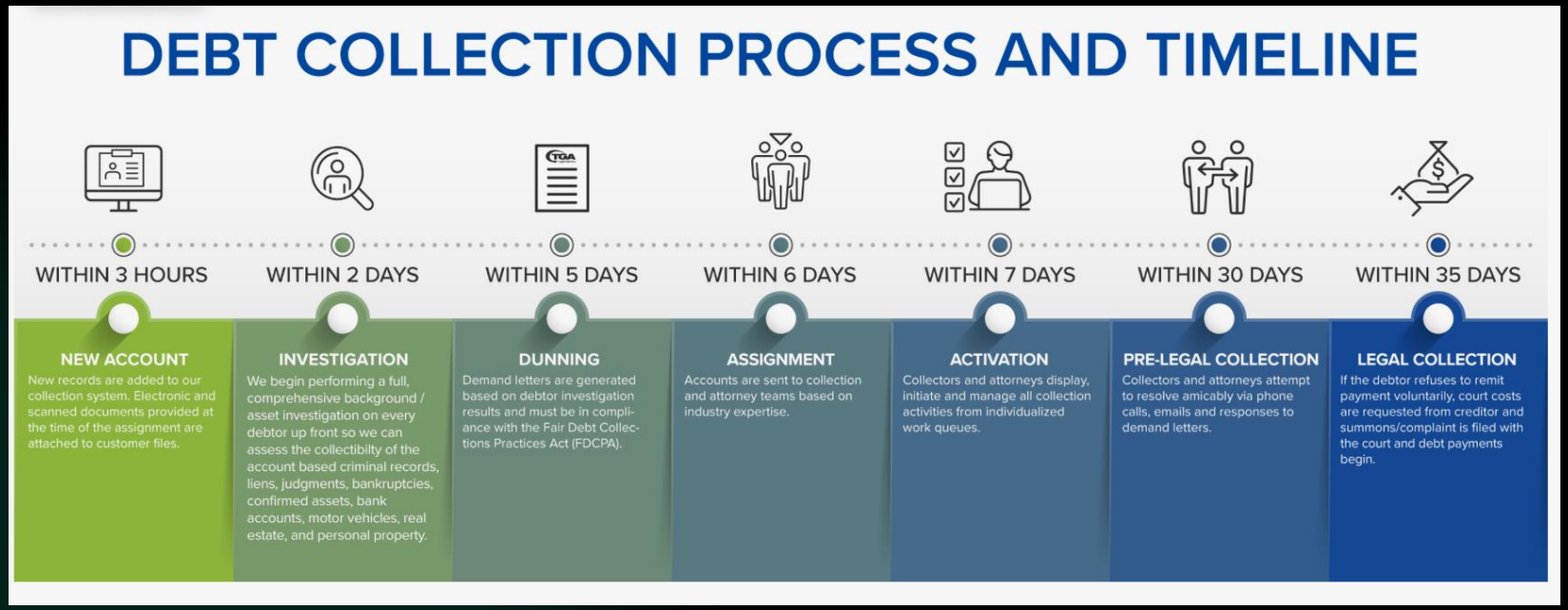
KEDGE 2023

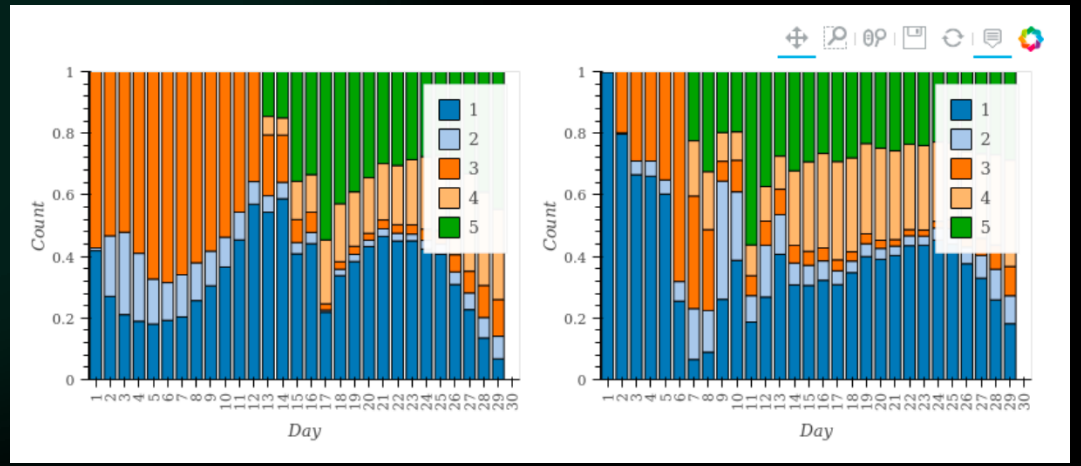


Задолженность клиента

Выбор оптимальной последовательности действия для улучшения возврата задолженности

Tinkoff 2022

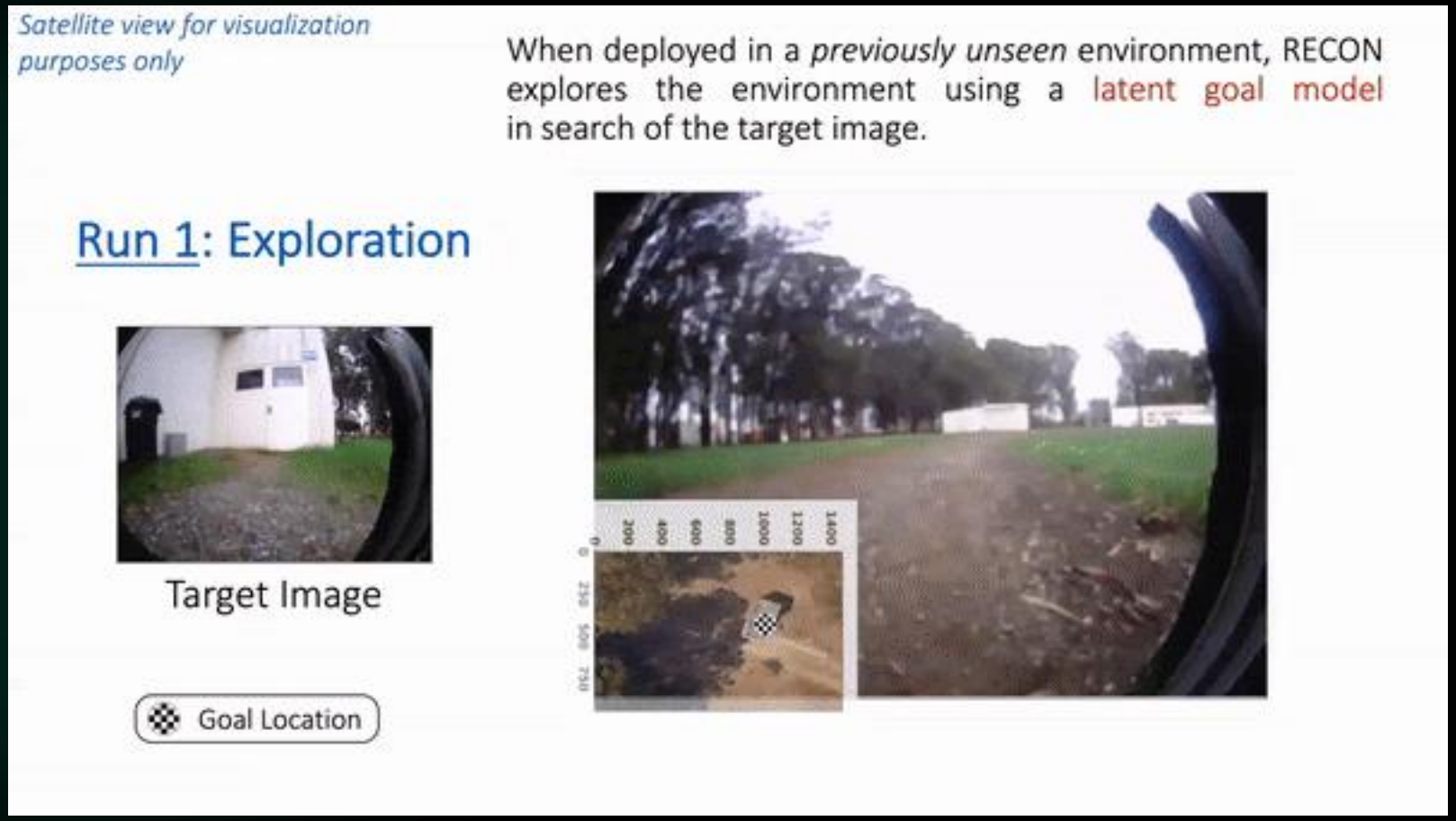




Навигация робота

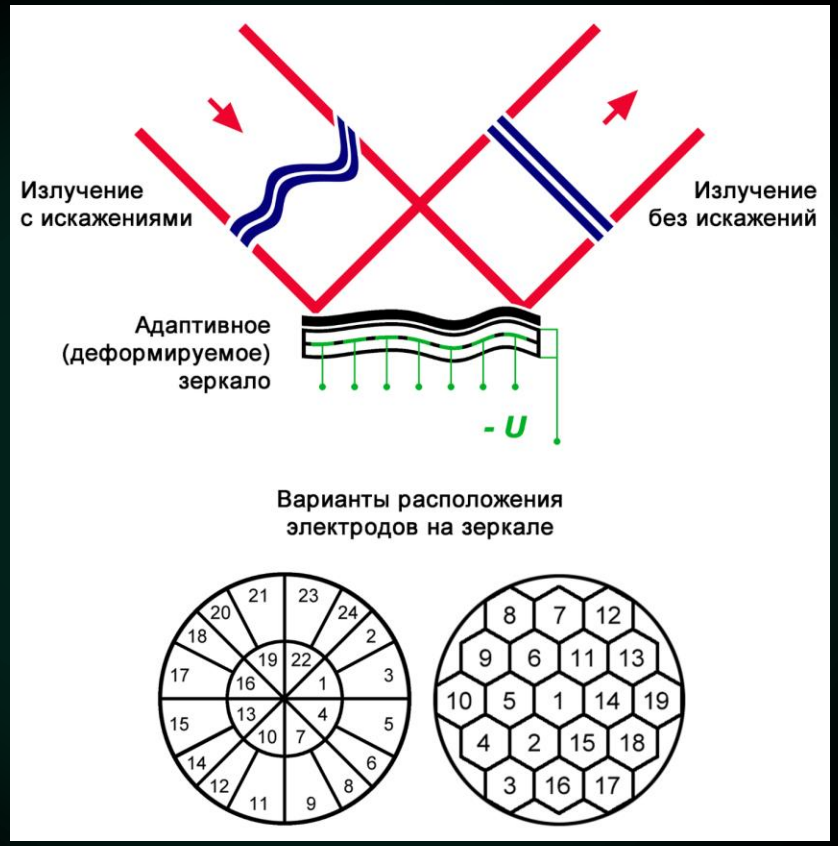
Быстрая робастная навигация наземного робота по камерам

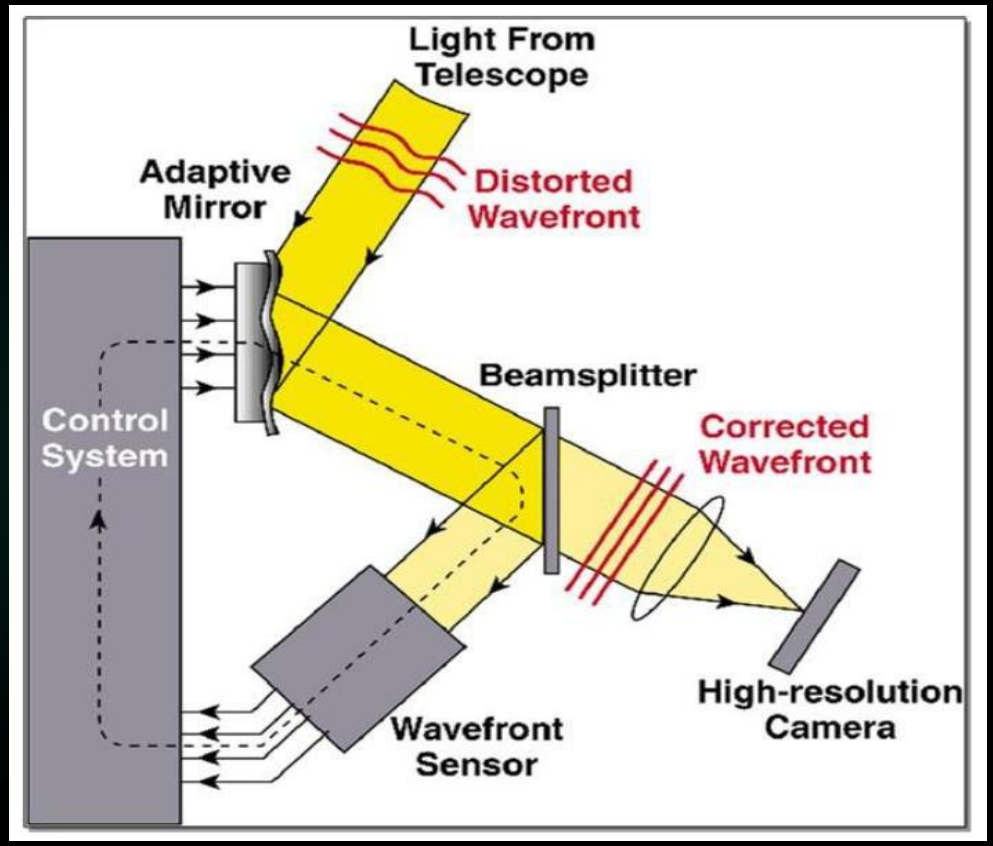
BAIR 2021



Калибровка датчиков

Адаптивно подбирается форма деформируемого зеркала, чтобы исправить искажения получаемые с камеры





Резюме

Обучение с подкреплением – обучение принятию решений, когда нет модели

1. **Обучение с подкреплением – обучаемый подход к решению задачи последовательного принятия решений**
2. **Обучение с подкреплением– особый вид машинного обучения**
3. **Обучение по имеющимся данным – автономный RL**
4. **Интерактивный RL – требовательность к симулятору**
5. **Проблемы: слабая устойчивость, переносимость**