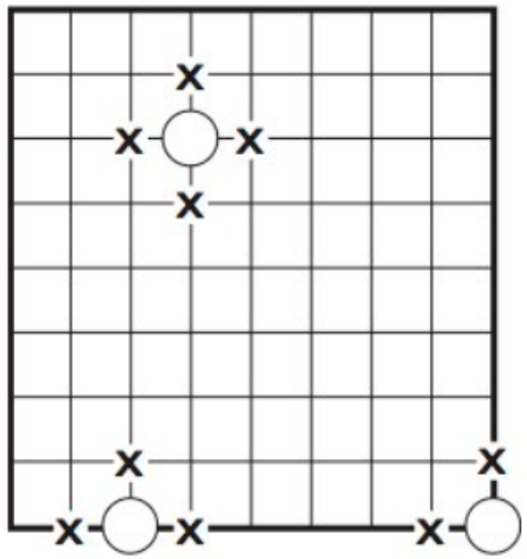
AlphaGo

Титизян Армине, 172

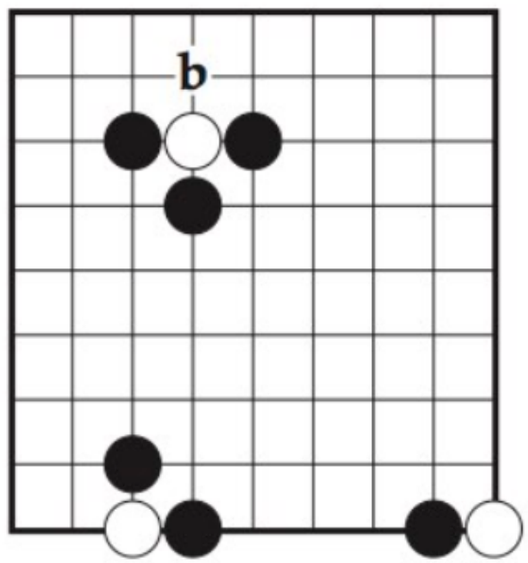
Го

- 19 x 19 доска с сеткой
- Черные и белые камни стоят на узлах
- Выиграл тот, кто больше занял территории
- Нельзя повторять позиции
- Коми

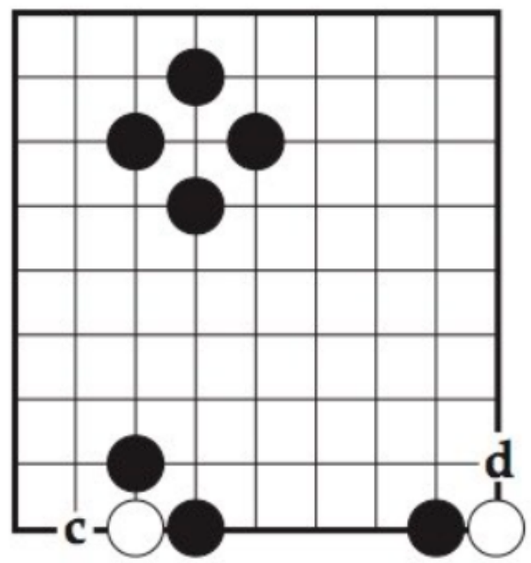
Го



Свободы камней

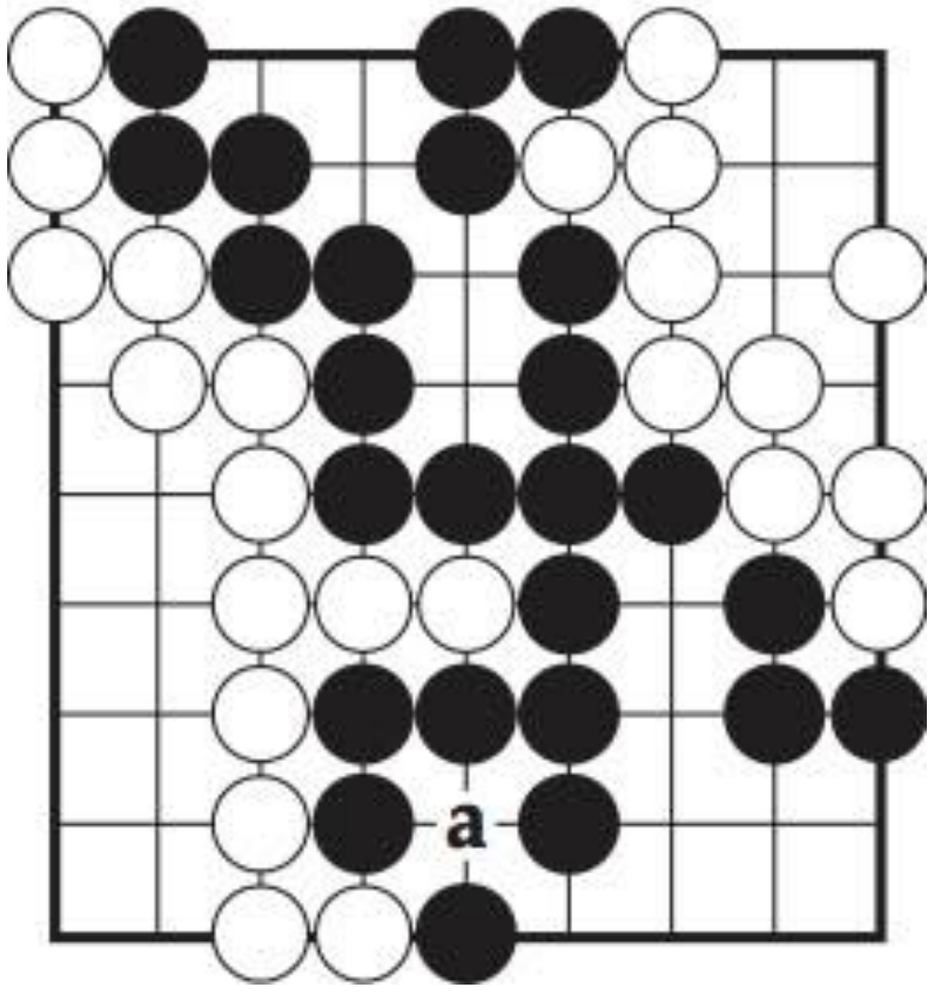


Захват камней



Захват камней

Го



Конец партии

В чем проблема?

В го примерно $b^d = 250^{150}$ возможных партий, b — количество разрешенных ходов, d — длина игры; в шахматах всего 35^{80}.

Отсюда проблема в обходе дерева позиций

До AlphaGo

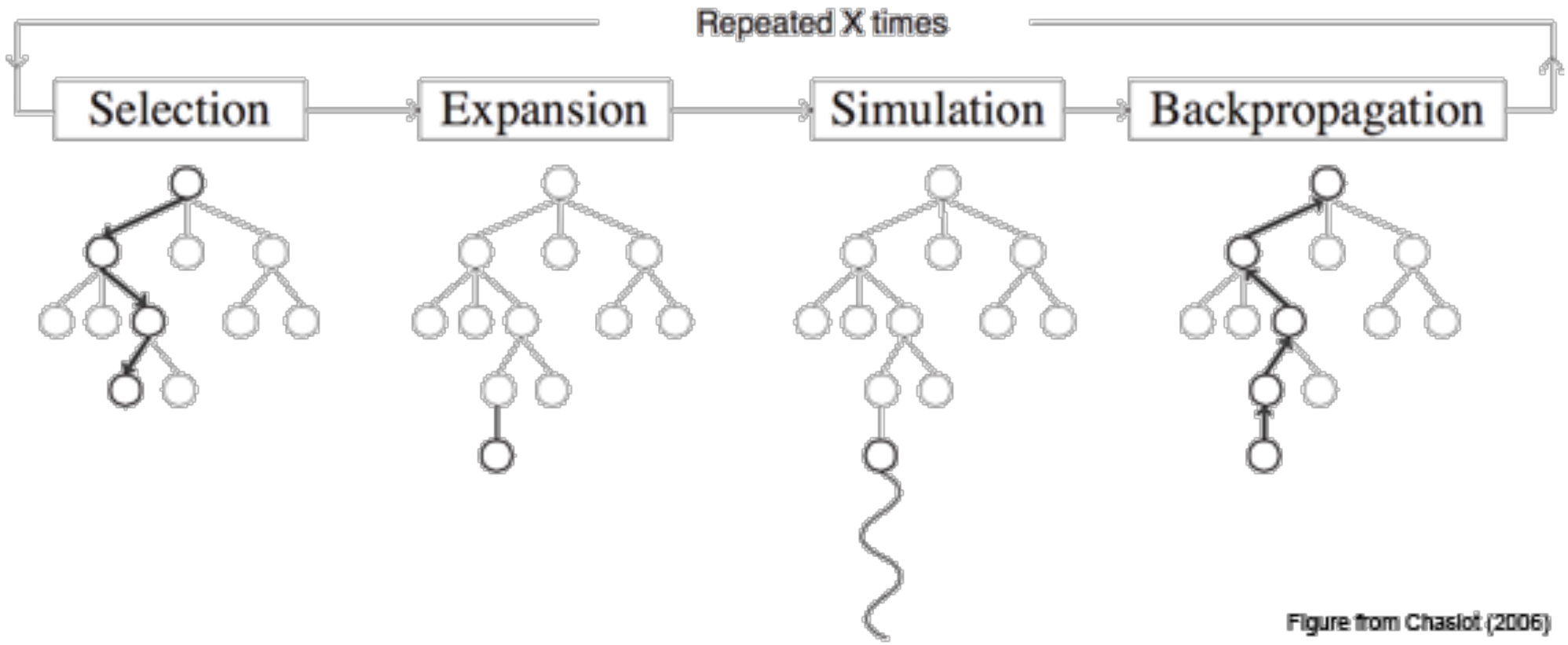


Figure from Chaslot (2006)

- Monte Carlo Tree Search (MCTS) — алгоритм принятия решений через поиск по дереву. Дерево представляет из себя структуру, в которой помимо хода и указателей есть количество сыгранных и количество выигранных партий. На основе этих двух параметров метод выбирает следующий шаг.

Шаг 1: Выбор — Selection. На этом шаге алгоритм выбирает ход своего противника. Если такой ход существует — мы его выберем, если нет — добавим.

Шаг 2: Расширение — Expansion. К выбранному узлу с ходом противника мы добавим узел со своим ходом и с нулевыми результатами.

Шаг 3: Симуляция — Simulation. Отыграем партию от текущего состояния игрового поля до чей-либо победы. Отсюда мы возьмём только первый ход (т.е. свой ход) и результаты.

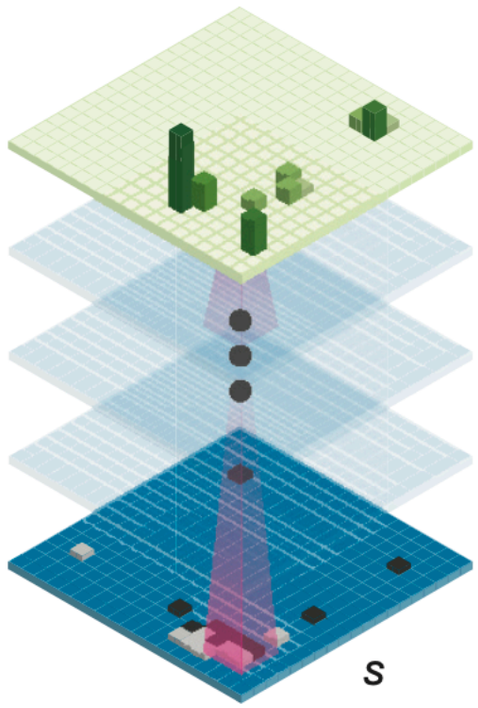
Шаг 4: Обратное распространение — Backpropagation. Результаты из симуляции мы будем распространять от текущего до корня. Ко всем родительским узлам мы добавим единицу в количество сыгранных

Общая архитектура

Policy network

$p_{\sigma/\rho}(a|s)$

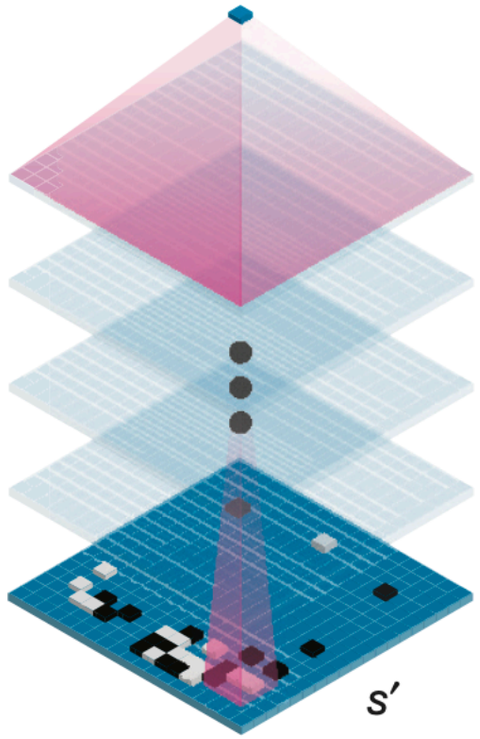
Показывает вероятные ходы



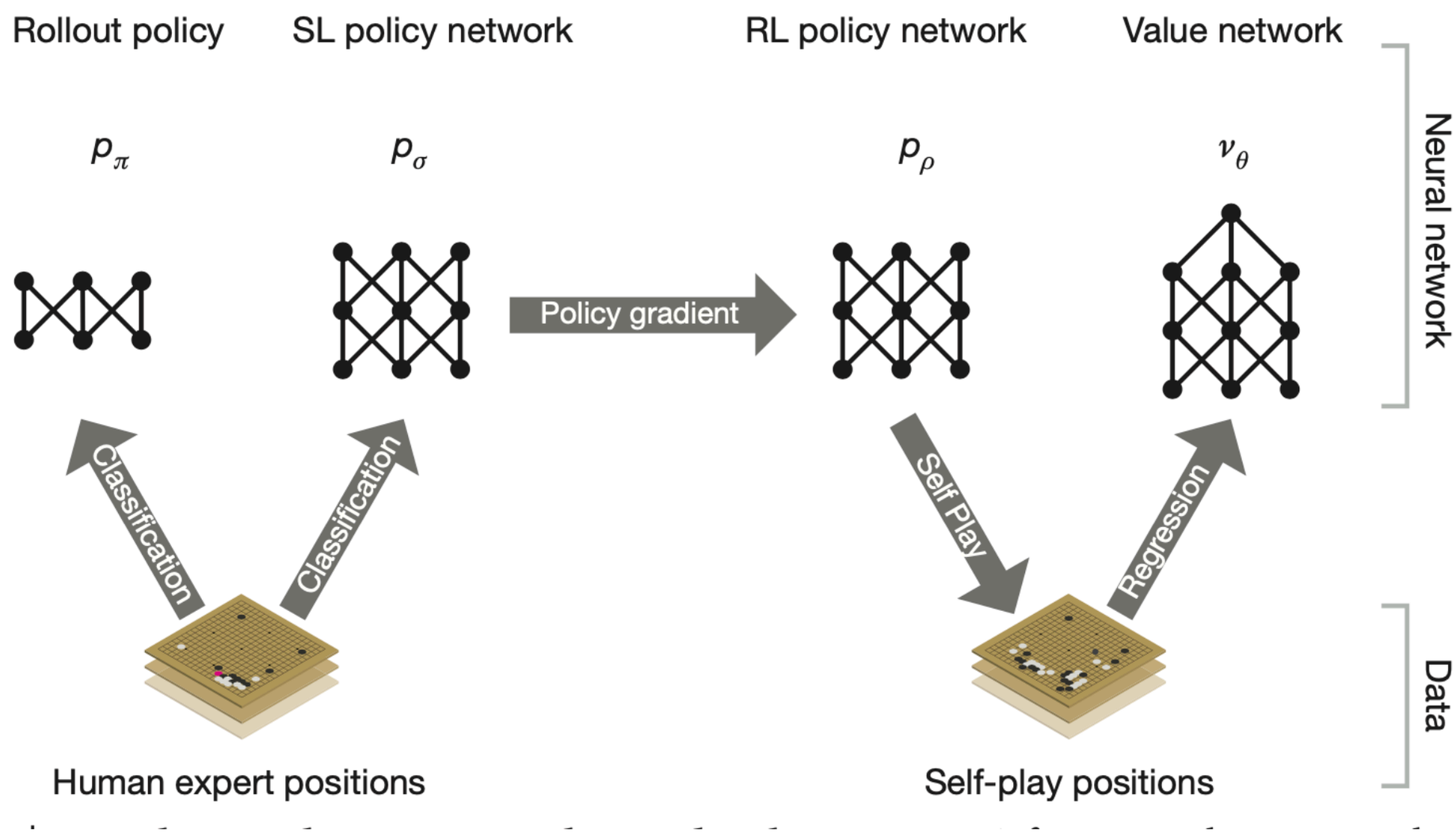
Value network

$v_\theta(s')$

Показывает вероятность выигрыша для клетки



Общая архитектура



Supervised learning Policy

- Учится предсказывать человеческие ходы
- 12 уровней convolution layers с нелинейностью и softmax на каждую клетку в конце.
- Использует человеческие партии на старте

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

Extended Data Table 2: **Input features for neural networks.** Feature planes used by the policy network (all but last feature) and value network (all features).

SL policy

σ — веса

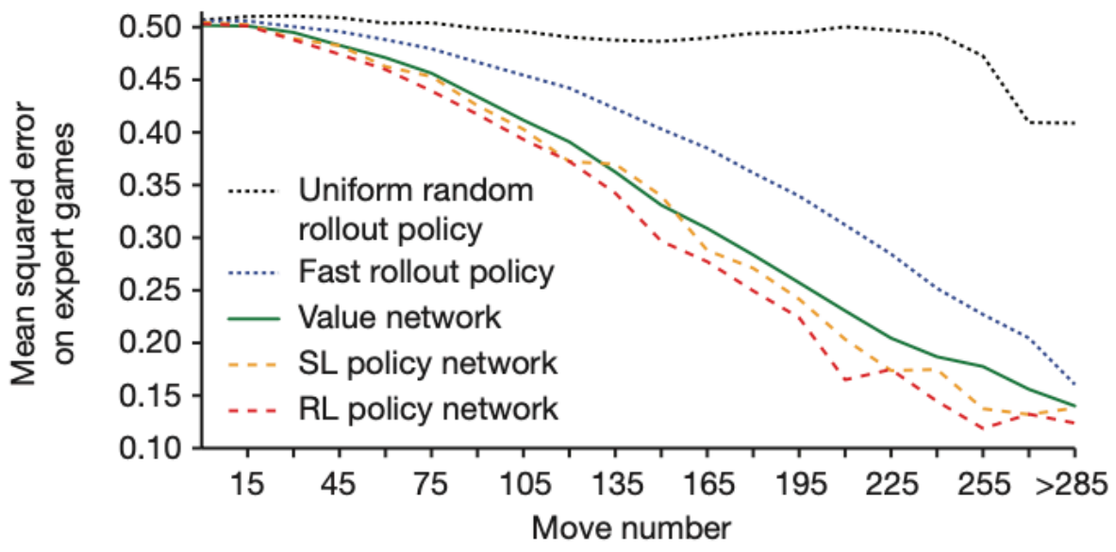
$$\Delta\sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial \sigma}$$

максимизируем
правдоподобие
стохастическим
градиентным
подъемом

Rollout

Feature	# of patterns	Description
Response	1	Whether move matches one or more response features
Save atari	1	Move saves stone(s) from capture
Neighbour	8	Move is 8-connected to previous move
Nakade	8192	Move matches a *nakade* pattern at captured stone
Response pattern	32207	Move matches 12-point diamond pattern near previous move
Non-response pattern	69338	Move matches 3×3 pattern around move

- Быстрая модель с плохими предсказаниями
- Линейный softmax
- Проверяем разные паттерны

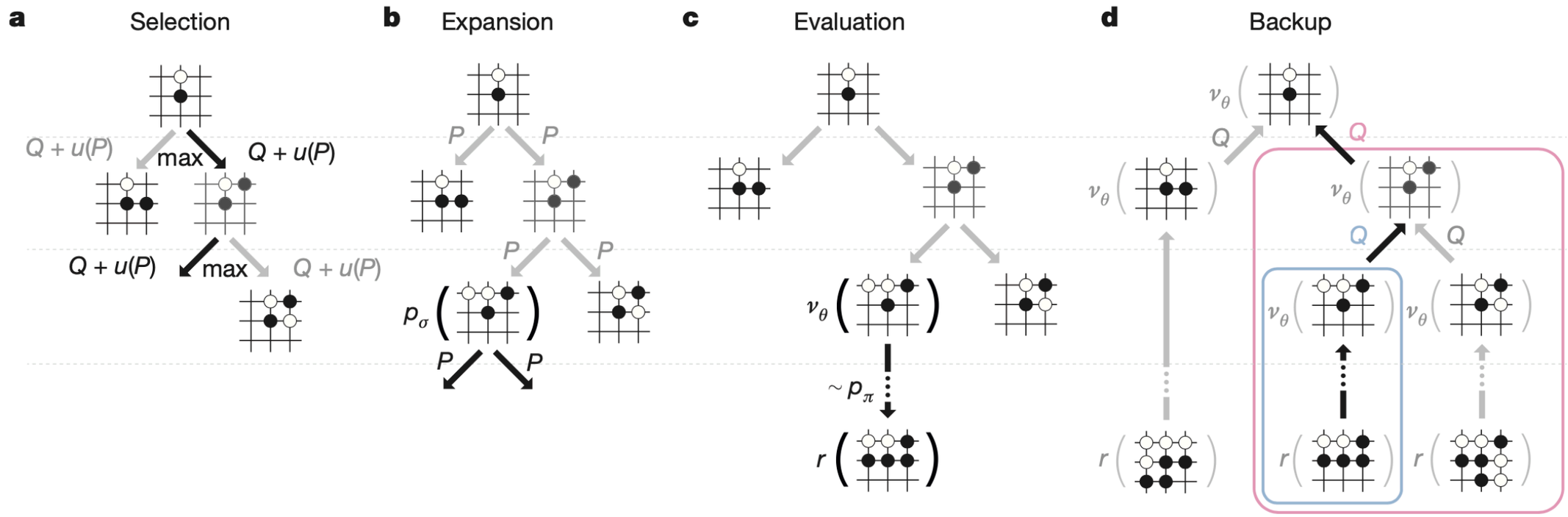


RL policy

- Архитектура как у SL + fully connected слой

$$\Delta\rho \propto \frac{\partial \log p_\rho(a_t|s_t)}{\partial \rho} z_t$$

MCTS



$$a_t = \underset{a}{\operatorname{argmax}}(Q(s_t, a) + u(s_t, a))$$

$$u(s, a) \propto \frac{P(s, a)}{1 + N(s, a)}$$

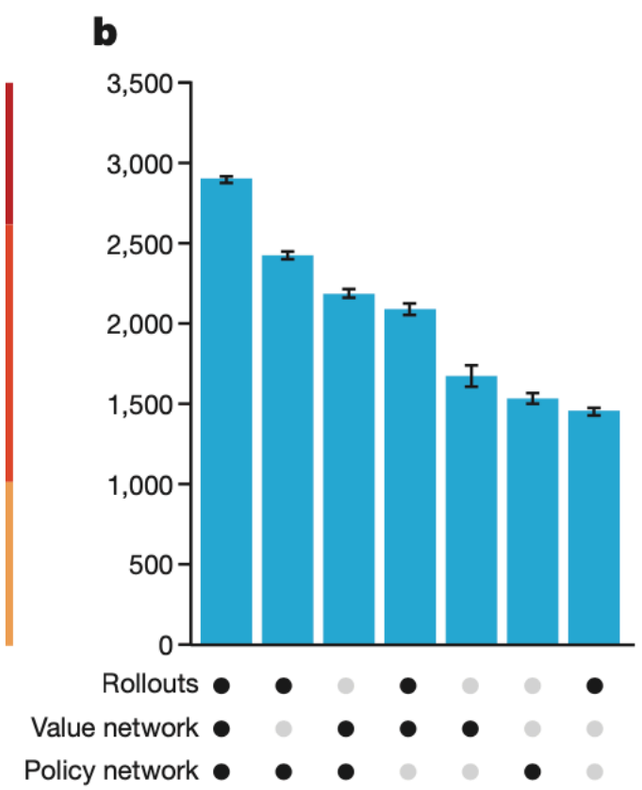
Инициализация новой вершины

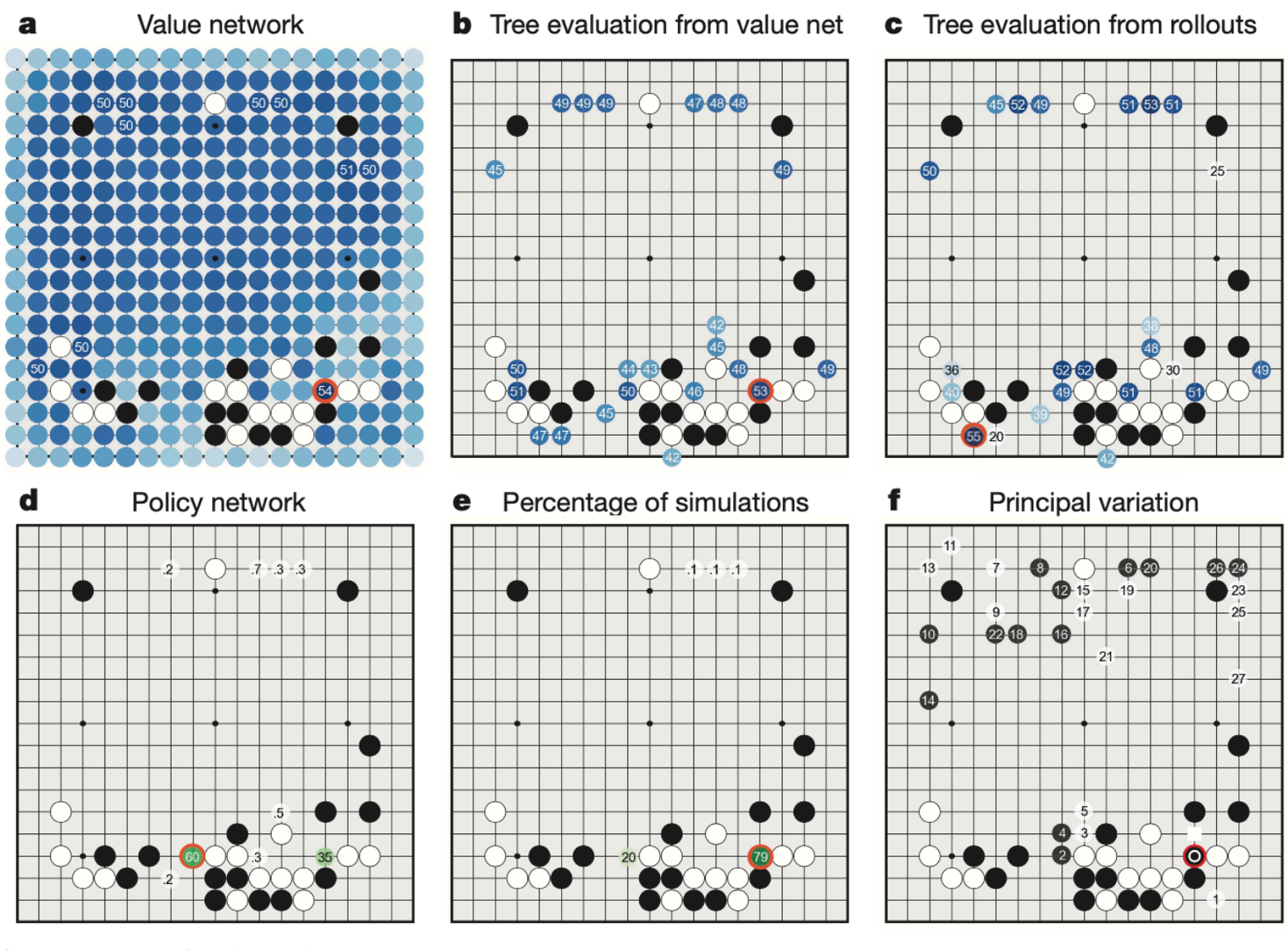
$$V(s_L) = (1 - \lambda)v_\theta(s_L) + \lambda z_L$$

$$N(s, a) = \sum_{i=1}^{n} 1(s, a, i)$$

$$Q(s, a) = \frac{1}{N(s, a)} \sum_{i=1}^{n} 1(s, a, i) V(s_L^i)$$

Итоговый выбор хода





Вопросы

- Назовите составные части AlphaGo и опишите их роль в модели.
- Опишите алгоритм MCTS. В чем отличия версии для AlphaGo от базового варианта?
- Зачем нужен rollout policy?
- Почему AlphaGo всегда выигрывает с маленьким отрывом (0.5-3 очка)?