ФРЕЙМВОРК MUZERO: ИСТОРИЯ РАЗВИТИЯ, ТЕХНИЧЕСКИЕ ВОЗМОЖНОСТИ

ИВАН ГРЕЧХИН (НИУ ВШЭ-НН/ЕРАМ)

Agenda

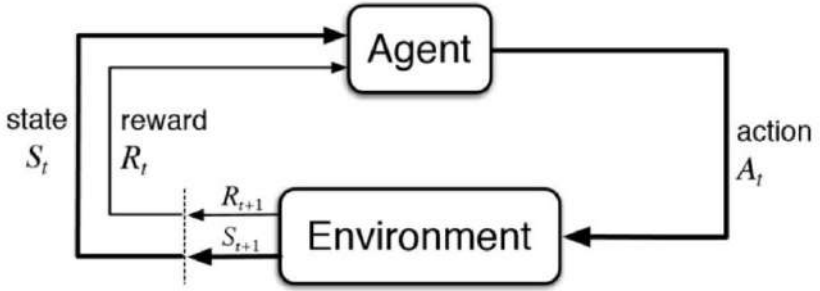
- Введение
- AlphaGo
- AlphaGo Zero
- Alpha Zero
- MuZero

Как познакомился с фреймворком

- Заказчик захотел решать задачу, где нужно было идти от исходного состояния к целевому через последовательность шагов
- Уже существующее решение использует эвристики/детерминистические алгоритмы
- Было предложено решение – системы с агентом, получающим вознаграждение – по сути reinforcement learning
- Рассматривались разные варианты, моя часть заключалась в исследовании MuZero для использования в задаче

Кратко про RL:

- RL = Reinforcement Learning, обучение с подкреплением
  - Некоторый марковский процесс: модель выбирает некоторое действие на основе текущего **состояния (state)** и предыдущего **действия (action)**. (Иногда не одного действия)
  - Каждое действие приводит к изменению состояния, которое для модели выражается в **награде (reward)** – по сути некоторый loss-наоборот для модели, по которому происходит обучение на входных данных: текущем состоянии и выбираемому действию.
  - Предполагается, что в процессе обучения мы подбираем оптимальную **стратегию (policy)** - вероятности выбора действий, приводящих к максимальной награде.



С чего всё началось?

- Deep Blue
  - Компьютер, играющий в шахматы
  - С начала разработки (1985) до победы над чемпионом мира по шахматам (1996) – 11 лет
  - Оценивал 200 миллионов позиций в секунду
  - До сих пор является сильнейшим по вычислительной мощности компьютером, игравшим с гроссмейстером
  - Использовал алгоритм alpha-beta pruning – умный брутфорс с минимаксом

Сравнение между Шахматами и Го

ШАХМАТЫ

- 6 типов фигур (считая слонов за 1 тип)
- Поле 8x8
- 20 возможных первых ходов
- Задача – побить короля без возможности того избежать атаки
- Возможна ничья
- Количество игр $35^{80} \sim 10^{123}$
- Гроссмейстерский уровень игры заключается в знании дебютов, которые много раз сыграны и опробованы, изучении игр соперника. По сути, единственное поле для «творчества» – миттельшпиль (и то не всегда). Существуют качественные оценки силы позиции и каждой фигуры; стратегии игры, которые значимо улучшают качество игры

ГО

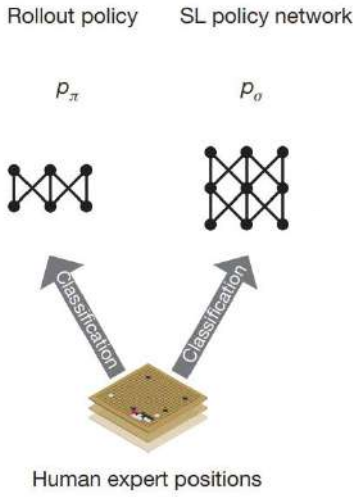
- 1 тип фигур
- Поле 19x19
- 361 возможный первый ход
- Задача – захватить больше территории
- Нет ничьих
- Количество игр $250^{150} \sim 10^{360}$
- Гроссмейстеры даже высокого дана часто объясняют свои ходы «интуицией» выработонной годами игры в го; несмотря на наличие некоторых рекомендаций, они в большей части полезны для начинающих игроков и игроков первых данов.

Alpha Go (2014-2016)

- Нейросетевой подход к RL:
    - Входные данные – изображение игровой доски на вход свёрточной нейронной сети + human-made features
    - Monte Carlo tree search – алгоритм построения дерева действий
    - 4 сверточных сети:
    - Результаты RL и Value используются в MCTS для построения дерева действий. Оценки этих нейронок используются для оценки возможных будущих позиций. SL используется для расширения и углубления дерева (expansion), Rollout используется для быстрого вычисления вознаграждения (симулируется партия до конца). Все значения обновляются на каждой симуляции. Исходя из дерева выбирается ход.

Первый этап обучения

- SL policy network – обучается на 30 миллионах известных позиций. Архитектура – 13x(conv + ReLU)+softmax слой, предсказывающий распределение вероятностей среди легальных ходов. Нейронка старалась предсказать человеческий ход в каждой позиции. Точность – 57% на всех фичах, 55.7% используя только позиции на доске в качестве входа, время предсказания - 3мс.
- Rollout policy – сеть поменьше и менее точная, чем SL, с той же функцией, точность 24.2% (удобно для обучения, поскольку в процессе обучения необходимо много раз оценивать следующий ход, для чего нужна быстрая вариация предсказывающей модели, время предсказания 2 мкс)
- Процесс обучения на первом этапе – предсказание следующего хода человека на основе текущей позиции и её качестве. «Тяжелая» нейросеть нужна для того, чтобы предсказывать policy – вектор вероятности легальных ходов человека в данной позиции с высокой точностью. Rollout нужна для того, чтобы производить много дешёвых симуляций, дающих менее точные ходы, но сокращающие временные затраты.



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

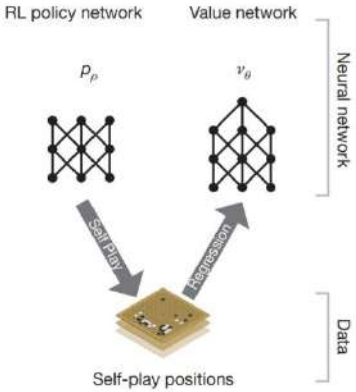
Второй этап обучения

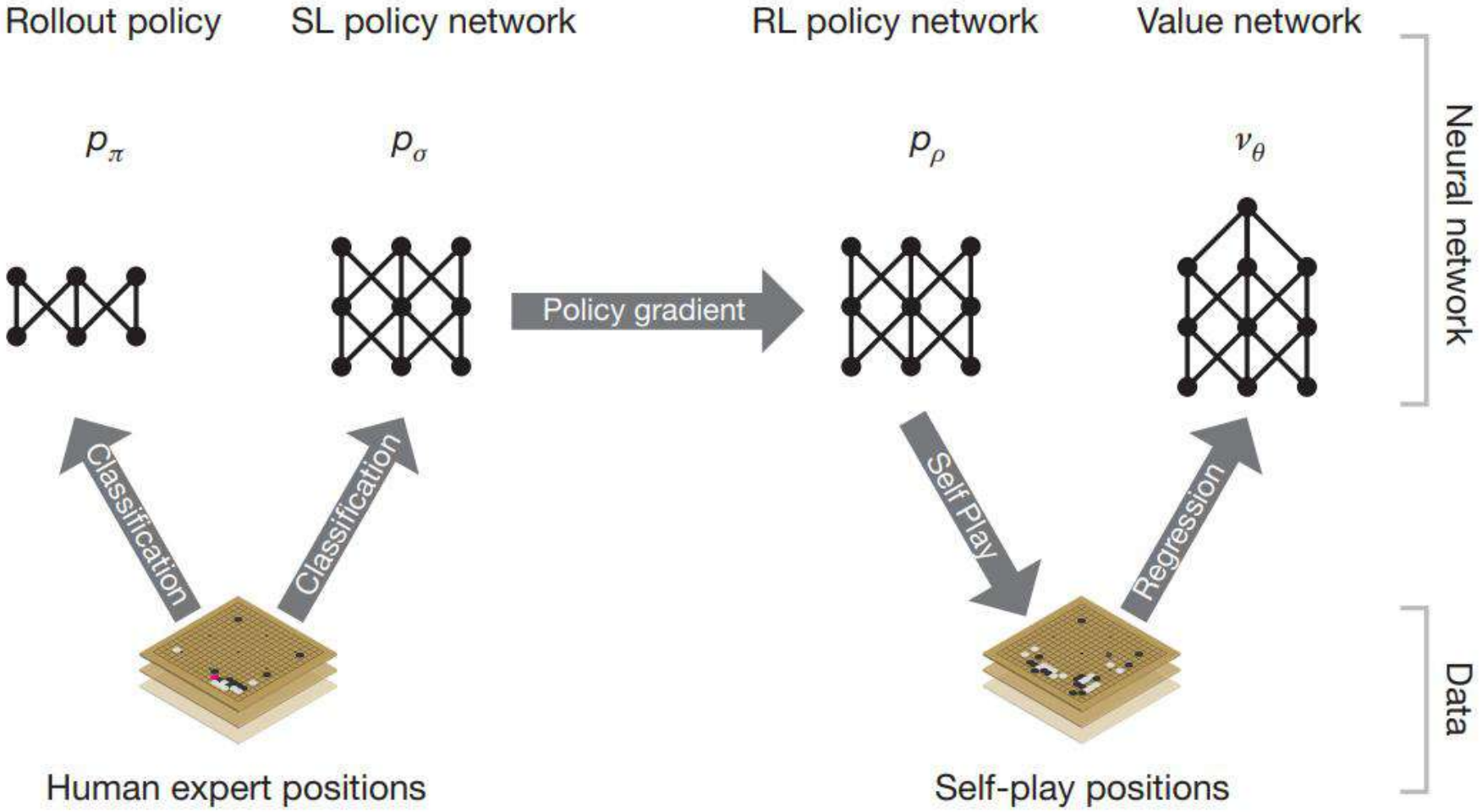
- RL policy network – сеть идентичная SL policy, инициализируемая весами от SL policy, обучающаяся на играх с одной из своей предыдущих версий, которая выбирается случайно. Функция вознаграждения равна 0 если ход не последний в партии и +1, -1 если ход последний и партия завершена от лица игрока совершившего ход.
    - Ходы выбираются рандомно согласно полученным policy
- Value network – предсказание качества позиции на основе policy подсчитанной для обоих игроков в текущей позиции. Предсказания пытаются минимизировать MSE предсказанного value согласено RL policy. Архитектура совпадает с RL с добавлением последнего слоя с единственным выходом.
    - Обучаясь на 30M партий произошло переобучение, MSE 0.19 на train и 0.37 на test (сеть переобучилась на результаты игр)
    - Из каждой партии был взят один snapshot, от которого производилась RL policy игра. 0.226 на train и 0.234 на test

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

$$v^p(s) = \mathbb{E}[z_t|s_t = s,\ a_{t\ldots T} \sim p]$$

$$\Delta\theta \propto \frac{\partial v_\theta(s)}{\partial \theta}(z - v_\theta(s))$$




Rollout policy
SL policy network
RL policy network
Value network
p_π
p_σ
p_ρ
v_θ
Policy gradient
Classification
Classification
Self Play
Regression
Neural network
Data
Human expert positions
Self-play positions

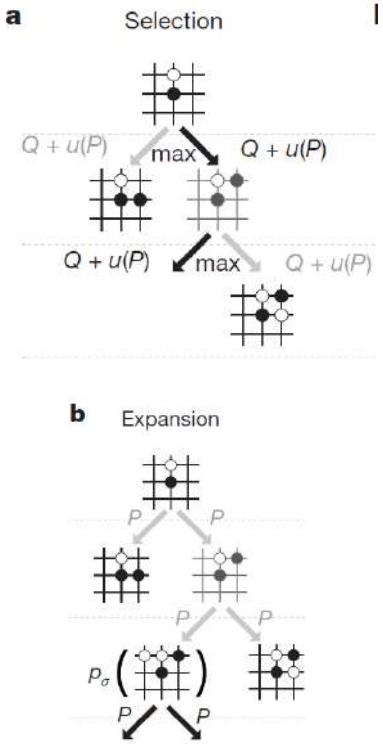
Monte Carlo Tree Search

- Обученные сетки уже играли в Го на уровне SOTA того времени.
- RL policy и Value используется в MCTS алгоритме:

1. Selection: Начиная с корня дерева (текущее игровое состояние), алгоритм двигается по дереву поиска до тех пор пока не достигнет листа. Проход по дереву использует специальную формулу выбора следующего действия, включающую в себя
    - Q – оценка действия
    - P – вероятность по SL policy
    - N – количество посещений данной ноды

$$a_t = \underset{a}{\operatorname{argmax}}(Q(s_t, a) + u(s_t, a)) \qquad u(s, a) \propto \frac{P(s, a)}{1 + N(s, a)}$$

2. Expansion: В листе происходит выбор нового действия, один раз вызывается SL policy и сохраняется навсегда как P.



Monte Carlo Tree Search

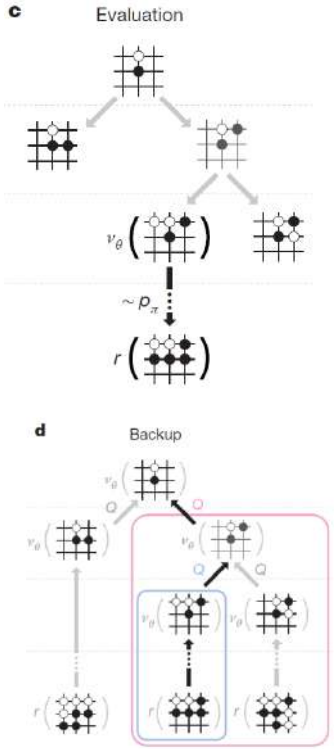
3. Evaluation или Simulation. На этом этапе происходит рандомный розыгрыш действий до получения результата. В Alpha Go комбинируют результаты прогона с помощью Rollout Policy и Value сети.

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

4. Backup. Сохранение результатов, обновление Q, N

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

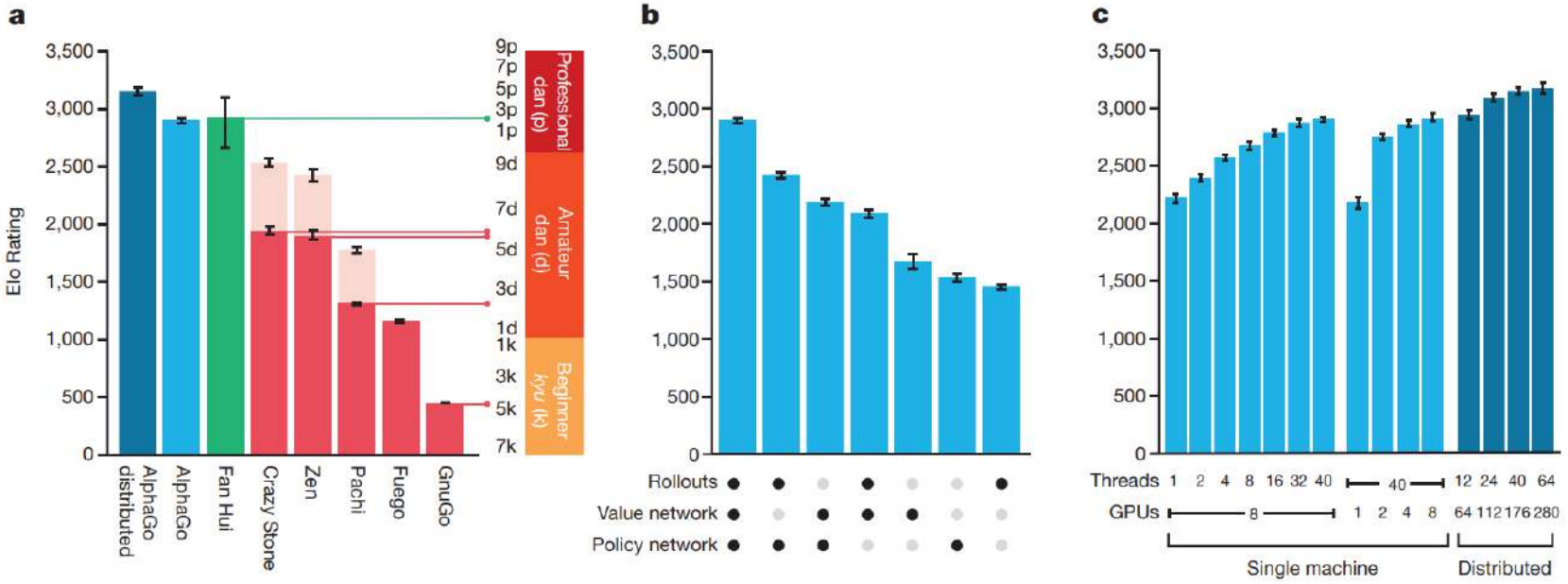
$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i)$$



- Интересная особенность – SL показывала себя лучше, чем RL в данном алгоритме и поэтому использовалась в expansion шаге. Однако, Value network обученная на RL работала лучше.
- Вычислительные ресурсы: 48 CPU, 8 GPU, 40 потоков поиска в MCTS в обычной версии, 1202 CPU, 176 GPU в распределённой

Alpha Go

- Для сравнения, был проведён турнир последней версии Alpha Go сама с собой и с известными коммерческими и открытыми программами (5 секунд на ход). Разница в 230 Elo означает 79% вероятность победы.
- Был проведён матч с европейским чемпионом по Го, где была одержана победа 5-0 (8-2 учитывая неформальные матчи)

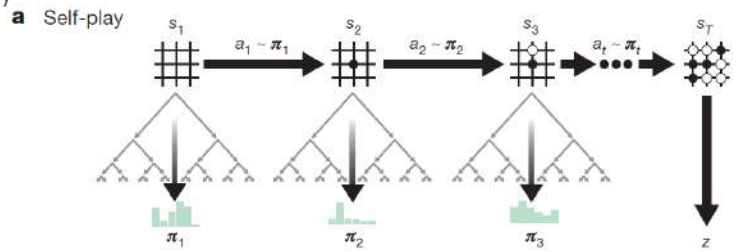


AlphaGo Zero (2016-2017)

- Нейросетевой подход к RL, избавляемся от сложности предыдущей модели:
  - Входные данные – изображение игровой доски на вход свёрточной нейронной сети, только положения, без фич
  - 1 сверточная сеть $f(\theta)$ RL policy+Value (ResNet-like)
  - MCTS теперь используется при обучении нейронной сети, выдавая policy π. MCTS воспринимается как метод значительного улучшения policy исходной нейросети
  - На каждой итерации, нейронная сеть генерирует ходы двух игроков с помощью MCTS, пока игра не закончится. Игре даётся оценка +1 или -1. Параметры нейронной сети обучаются таким образом, чтобы Value предсказывал оценку партии, а Policy предсказывал вектор вероятностей соответствующий тому, что выдаёт MCTS:

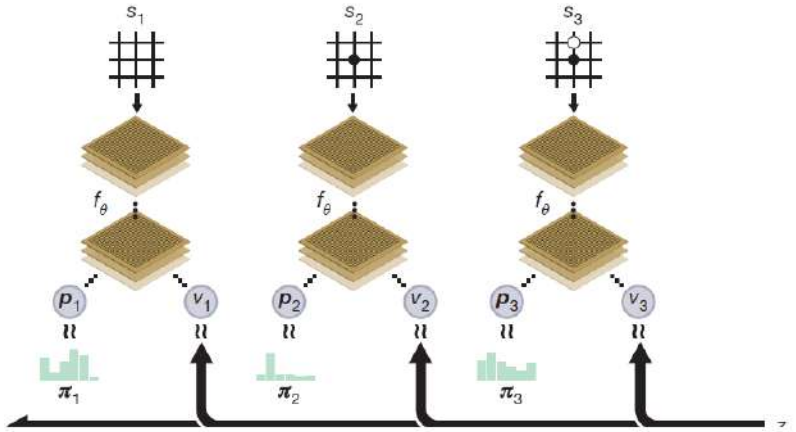
- Показатели P, Q модифицируются и используют $f(\theta)$
- Policy π вычисляется как вектор, обратно пропорциональный количеству посещений соответстующей ноды, $\pi \sim N^{\frac{1}{\tau}}$ (первые 30 ходов $\tau = 1$)
- MCTS используется на каждом ходу, в результате получается тройка (s, π, z). На этих тройках обучается нейронная сеть согласно указанной функции потерь
- Нейронная сеть состоит из 20 или 40 residual convolution блоков с batchnorm и relu

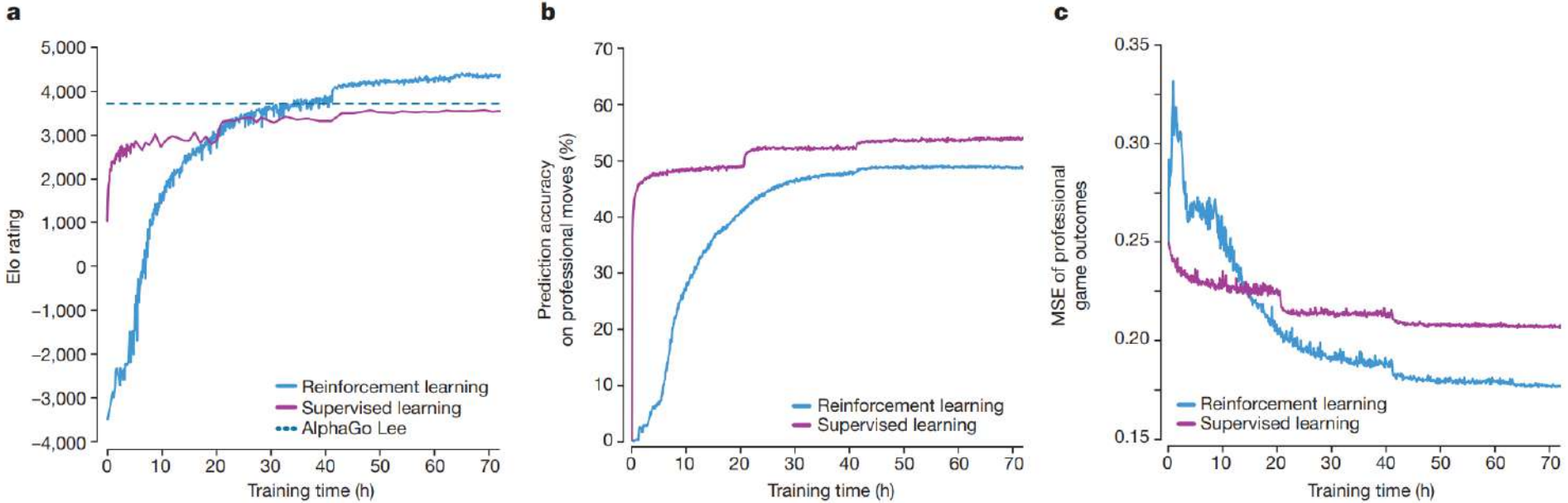
$$(\boldsymbol{p}, v) = f_\theta(s) \text{ and } l = (z - v)^2 - \boldsymbol{\pi}^{\mathrm{T}} \log \boldsymbol{p} + c\|\theta\|^2$$

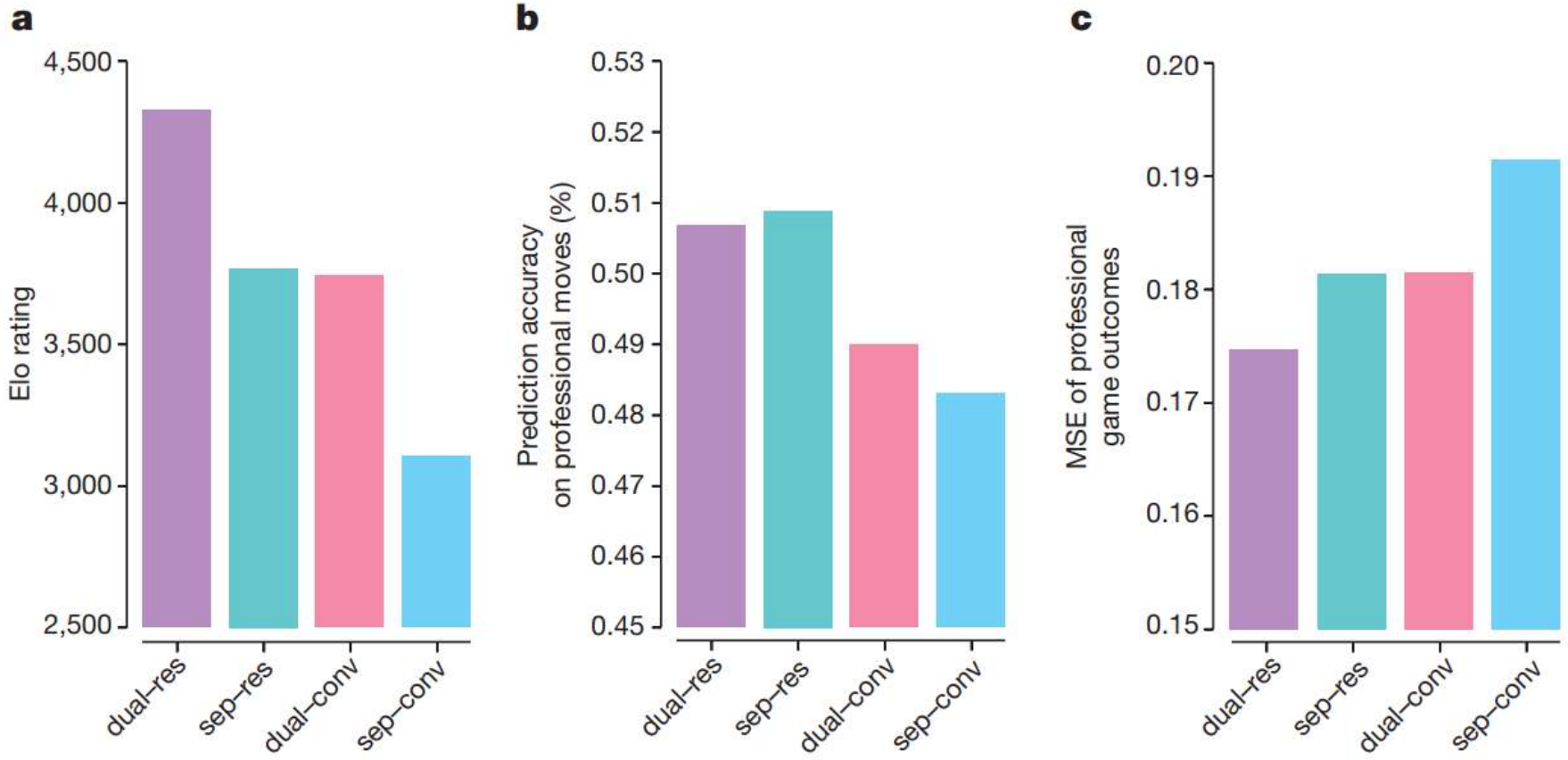


Процесс обучения

- Сначала абсолютно случайное поведение
- 4900000 партий с 1600 симуляций на MCTS (~0.4 секунды на ход). Обновление параметров 700000 раз, каждый батч состоит из 2048 позиций
    - 64 GPU (mini-batch 32), 19CPU
    - 2048 позиций сэмплированы случайно из 500к последних игр
- Обучение длилось 3 дня
- После 3х дней обучения AlphaGo Zero победила Alpha Go Lee со счётом 100-0, при этом:
    - AlphaGo Lee обучалась несколько месяцев
    - 4 TPU у Zero против 48 TPU (распределённо) у AlphaGo Lee




a
Elo rating
5,000
4,000
3,000
2,000
1,000
0
−1,000
−2,000
−3,000
−4,000
0
10
20
30
40
50
60
70
Training time (h)
Reinforcement learning
Supervised learning
AlphaGo Lee
b
Prediction accuracy on professional moves (%)
70
60
50
40
30
20
10
0
0
10
20
30
40
50
60
70
Training time (h)
Reinforcement learning
Supervised learning
c
MSE of professional game outcomes
0.35
0.30
0.25
0.20
0.15
0
10
20
30
40
50
60
70
Training time (h)
Reinforcement learning
Supervised learning


a
Elo rating
4,500
4,000
3,500
3,000
2,500
dual–res
sep–res
dual–conv
sep–conv
b
Prediction accuracy on professional moves (%)
0.53
0.52
0.51
0.50
0.49
0.48
0.47
0.46
0.45
dual–res
sep–res
dual–conv
sep–conv
c
MSE of professional game outcomes
0.20
0.19
0.18
0.17
0.16
0.15
dual–res
sep–res
dual–conv
sep–conv

AlphaGo Zero

- Архитектура, обучившаяся игре сама с собой превзошла такую же архитектуру, обученную на известных партиях более чем на 300 Elo, а также, в процессе обучения, нашла несколько известных угловых ситуаций (как гамбиты или дебюты в шахматах) , а также несколько новых, ранее неизвестных
- В процессе обучения:
    - После 3х часов архитектура концентрировалась на жадном захвате камней
    - После 19 часов архитектура пыталась контролировать территорию и избегать ловушек
    - После 70 часов архитектура вела множественные «битвы» в разных участках доски, а также *ko* битвы.
- Было отмечено, что некоторые известные позиции становились более популярны или менее популярны со временем

- Возможно ли применить для шахмат? Другие правила, зависимость ходов от позиции фигур, возможность ничьи…

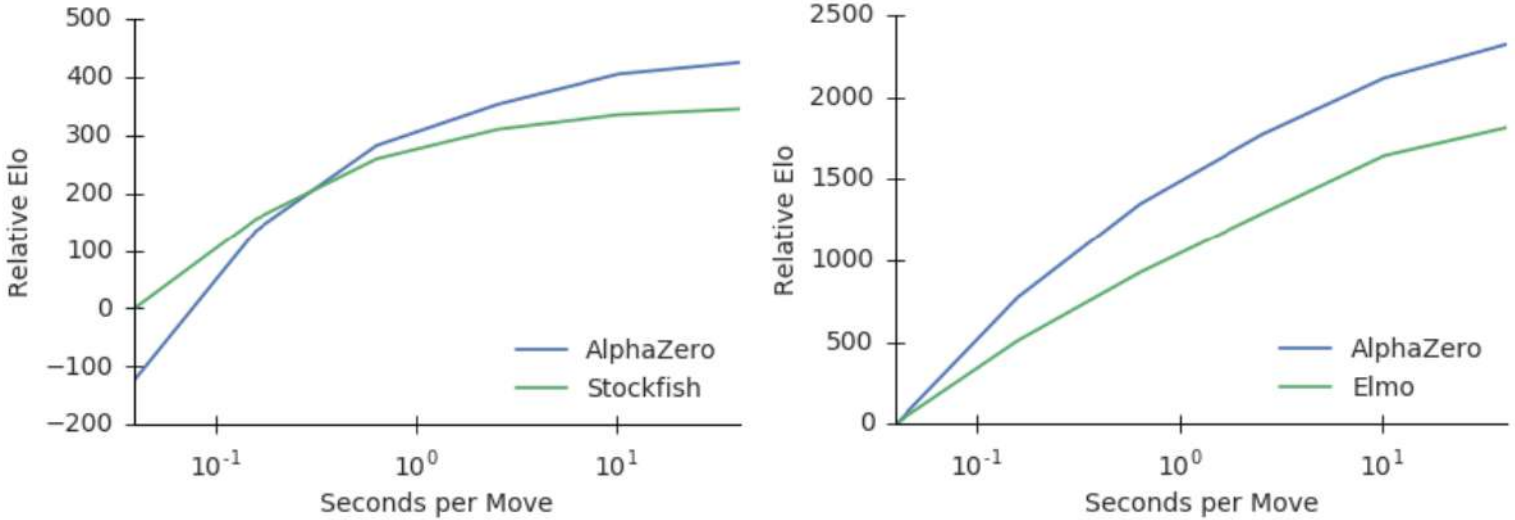
AlphaZero (2017)

- Вместо бинарного выхода используется набор выходов (например, победа, ничья, поражение)
- Нет аугментации обучающих данных за счёт возможных симметрий
- Нет «лучшего предыдущего» игрока, нейронная сеть одна и обучается постоянно на играх сама с собой.
- Результаты тренировки (700000 батчей по 4096 позиций, 5000 TPU for self-play, 64 TPU for NN):
  - За 2 часа были побиты результаты лучших программ в Шоги (Elmo)
  - За 4 часа – шахматы (Stockfish)
  - За 8 часов – AlphaGo Lee

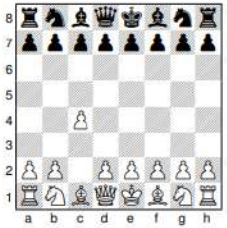
Game	White	Black	Win	Draw	Loss
Chess	*AlphaZero*	*Stockfish*	25	25	0
	Stockfish	*AlphaZero*	3	47	0
Shogi	*AlphaZero*	*Elmo*	43	2	5
	Elmo	*AlphaZero*	47	0	3
Go	*AlphaZero*	*AG0 3-day*	31	–	19
	AG0 3-day	*AlphaZero*	29	–	21

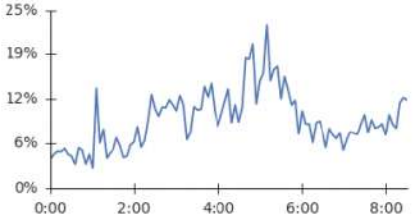
AlphaZero

- В шахматах в секунду изучается 80.000 позиций против 70.000.000 в Stockfish. В шоги 40.000 против 35.000.000 от Elmo
- В шахматах, AlphaZero самостоятельно открыла известные дебюты и меняла частоту их применения со временем (возможно, тем самым выясняя успешность каждого). В каждом из дебютов Stockfish был обыгран.
- Мощность в игре – 4 TPU



A10: English Opening

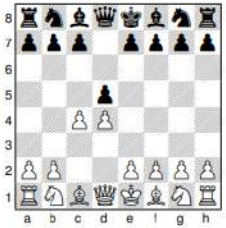


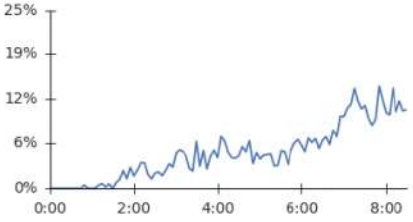


w 20/30/0, b 8/40/2

1...e5 g3 d5 cxd5 ♘f6 ♗g2 ♘xd5 ♘f3

D06: Queens Gambit

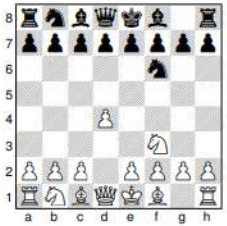


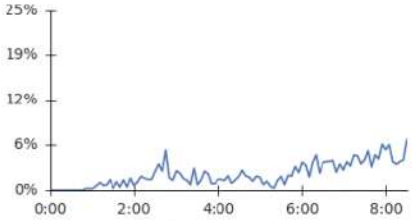


w 16/34/0, b 1/47/2

2...c6 ♘c3 ♘f6 ♘f3 a6 g3 c4 a4

A46: Queens Pawn Game

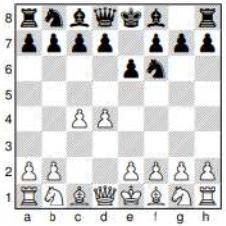


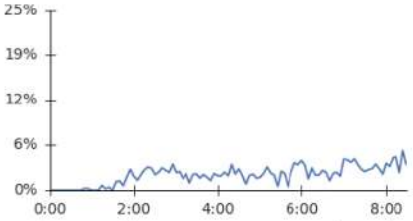


w 24/26/0, b 3/47/0

2...d5 c4 e6 ♘c3 ♗e7 ♗f4 O-O e3

E00: Queens Pawn Game

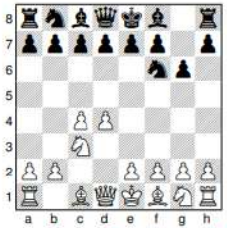


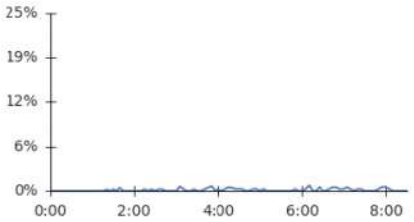


w 17/33/0, b 5/44/1

3.♘f3 d5 ♘c3 ♗b4 ♗g5 h6 ♕a4 ♘c6

E61: Kings Indian Defence

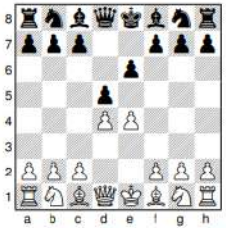


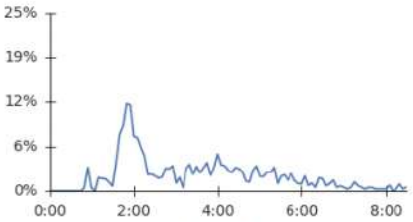


w 16/34/0, b 0/48/2

3...d5 cxd5 ♘xd5 e4 ♘xc3 bxc3 ♗g7 ♗e3

C00: French Defence





w 39/11/0, b 4/46/0

3.♘c3 ♘f6 e5 ♘d7 f4 c5 ♘f3 ♗e7

Go		Chess		Shogi	
Feature	Planes	Feature	Planes	Feature	Planes
P1 stone	1	P1 piece	6	P1 piece	14
P2 stone	1	P2 piece	6	P2 piece	14
		Repetitions	2	Repetitions	3
				P1 prisoner count	7
				P2 prisoner count	7
Colour	1	Colour	1	Colour	1
		Total move count	1	Total move count	1
		P1 castling	2		
		P2 castling	2		
		No-progress count	1		
Total	17	Total	119	Total	362

MuZero (2019)

- Возможность использовать на играх с одним игроком
- Ненулевое вознаграждение на промежуточных шагах
- Основы:
    - Три функции-нейронки:
        - Динамическая: функция, на каждом шаге вычисляющая вознаграждение на основе предыдущего состояния и совершённого действия. В AlphaZero мы знали динамическую функцию наперёд и задавали её правилами игры.
        - Предсказательная: RL policy + Value
        - Представительная: функция, кодирующая исходное состояние s0 – по сути кодировщик, для универсализации игрового поля.
- Мы можем создавать свои игровые среды, в которых задаем:
    - Игровое поле как исходное состояние на вход кодировщика
    - Список легальных ходов (либо полный список ходов с отрицательным вознаграждением, если ход нелегален). В AlphaZero мы могли наперёд сказать среде (MCTS) какие ходы нелегальны, MuZero же придётся выучить, какие ходы в каких позициях нелегальны.
    - Процесс изменения игрового поля в зависимости от выбранного шага с указанием вознаграждения. Также, необходимо прописывать смену игрока после хода (в случае 2х и более игроков).

Детали

- MCTS выбор действия:

$$a^k = \arg\max_a \left[Q(s,a) + P(s,a) \cdot \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)} \left(c_1 + \log\left(\frac{\sum_b N(s,b) + c_2 + 1}{c_2} \right) \right) \right]$$

- Функция потерь:

$$l_t(\theta) = \sum_{k=0}^{K} l^r(u_{t+k}, r_t^k) + l^v(z_{t+k}, v_t^k) + l^p(\pi_{t+k}, \mathbf{p}_t^k) + c||\theta||^2$$

- Все нейронки тренируются одновременно на одну функцию потерь
- В сравнении с AlphaGo, MuZero:
  - Самостоятельно выучивает переходы между состояниями
  - Выучивает «запрещённые» ходы
  - Понимает конечные состояния
- На вход представительной нейронки подаются «траектории» – наборы предыдущих состояний, полученных через MCTS
- Входом динамической нейронки является выход представительной/результат предыдущей динамической + action
- Все архитектуры – ResNet как в AlphaZero

Summary

Model

$$\left.\begin{array}{ll} s^0 & = h_\theta(o_1, ..., o_t) \\ r^k, s^k & = g_\theta(s^{k-1}, a^k) \\ \mathbf{p}^k, v^k & = f_\theta(s^k) \end{array}\right\} \quad \mathbf{p}^k, v^k, r^k = \mu_\theta(o_1, ..., o_t, a^1, ..., a^k)$$

Search

$$\nu_t, \pi_t = MCTS(s_t^0, \mu_\theta)$$
$$a_t \sim \pi_t$$

Learning Rule

$$\mathbf{p}_t^k, v_t^k, r_t^k = \mu_\theta(o_1, ..., o_t, a_{t+1}, ..., a_{t+k})$$

$$z_t = \begin{cases} u_T & \text{for games} \\ u_{t+1} + \gamma u_{t+2} + ... + \gamma^{n-1} u_{t+n} + \gamma^n \nu_{t+n} & \text{for general MDPs} \end{cases}$$

$$l_t(\theta) = \sum_{k=0}^{K} l^r(u_{t+k}, r_t^k) + l^v(z_{t+k}, v_t^k) + l^p(\pi_{t+k}, p_t^k) + c||\theta||^2$$

Losses

$$l^r(u, r) = \begin{cases} 0 & \text{for games} \\ \boldsymbol{\phi}(u)^T \log \mathbf{r} & \text{for general MDPs} \end{cases}$$

$$l^v(z, q) = \begin{cases} (z-q)^2 & \text{for games} \\ \boldsymbol{\phi}(z)^T \log \mathbf{q} & \text{for general MDPs} \end{cases}$$

$$l^p(\pi, p) = \boldsymbol{\pi}^T \log \mathbf{p}$$

Выводы

- MuZero работает, но есть много «но»:
  - Необходимо чётко понимать, как преобразовать свою задачу в игровую для решения
  - Требует полиномиально много ресурсов (чем больше параметров/состояний, тем хуже)
- Для относительно больших задач
  - Очень много степов (зацикливание)
  - Нужно подбирать гиперпараметры для обучения
  - Очень долго обучать
  - Необходимо качественно составить env и обучающую выборку

Литература

- Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., … Hassabis, D. (2017). *Mastering the game of Go without human knowledge. Nature, 550(7676), 354–359.* doi:10.1038/nature24270
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., … Hassabis, D. (2016). *Mastering the game of Go with deep neural networks and tree search. Nature, 529(7587), 484–489.* doi:10.1038/nature16961
- *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm*. David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, Demis Hassabis. https://arxiv.org/abs/1712.01815
- *Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model*. Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap, David Silver. https://arxiv.org/abs/1911.08265