

ОБЗОР СОВРЕМЕННЫХ ИДЕЙ В RL

Сергей Николенко

Академия MADE – Mail.Ru

30 октября 2021 г.

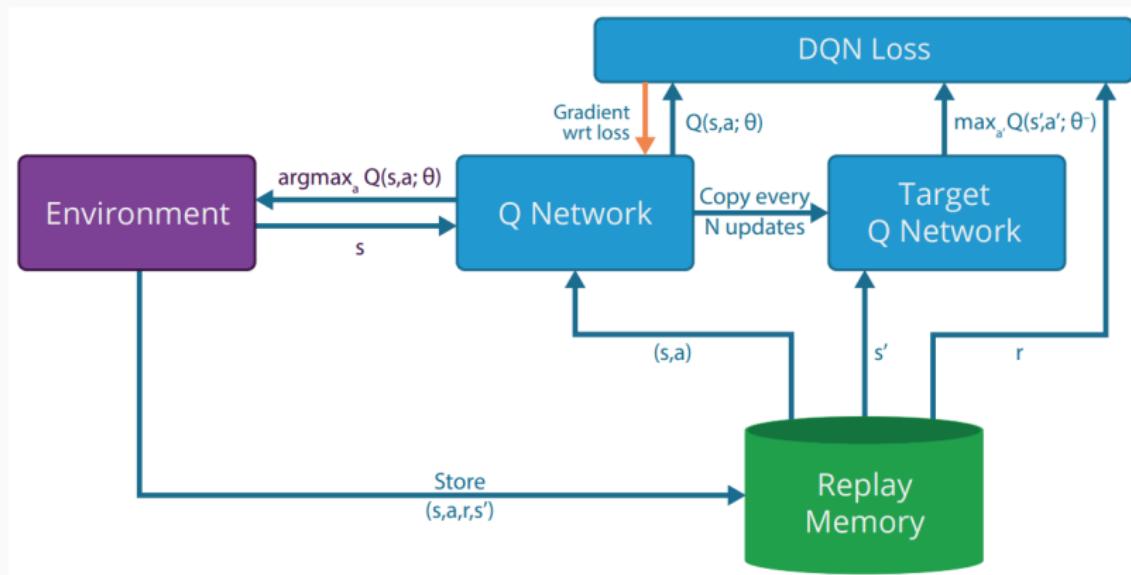
Random facts:

- 30 октября в России – День памяти жертв политических репрессий; 30 октября 1990 г. на Лубянской площади был установлен Соловецкий камень в память о них
- 30 октября 1905 г. Николай II ввёл Высочайший Манифест об усовершенствовании государственного порядка («Манифест 17 октября»)
- 30 октября 1907 г. русский физик Борис Розинг получил патент №18076 на «Способ электрической передачи изображений на расстояние» (то есть телевидение)
- 30 октября 1938 г. Орсон Уэллс поставил на радиостанции CBS «Войну миров», стилизованную под прямой репортаж с места событий; многие американцы поверили, и постановка вызвала настоящую панику; в ходе последовавших разбирательств выяснилось, что около 300 тысяч американцев лично видели марсиан
- 30 октября 1941 г. Франклин Рузвельт одобрил выделение 1 млрд долларов в качестве помощи Советскому Союзу

ПАРАЛЛЕЛЬНЫЕ РЕАЛИЗАЦИИ RL И
СОВРЕМЕННЫЕ ACTOR-CRITIC
АЛГОРИТМЫ

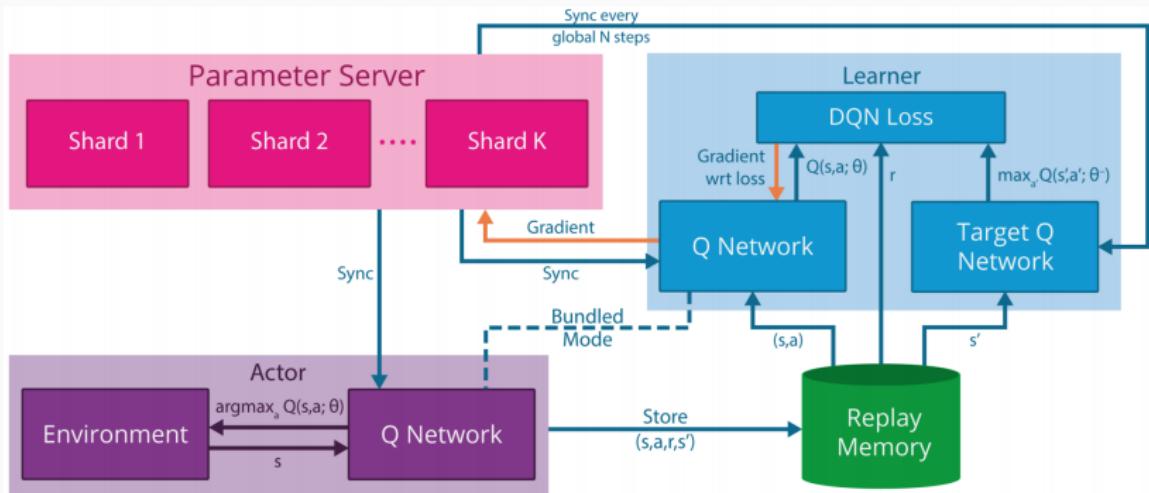
СОВРЕМЕННЫЕ ACTOR-CRITIC АЛГОРИТМЫ

- А что с policy gradient? Это тесно связано с тем, как распараллеливать RL.
- Вот что нам нужно сделать, например, в DQN:



СОВРЕМЕННЫЕ ACTOR-CRITIC АЛГОРИТМЫ

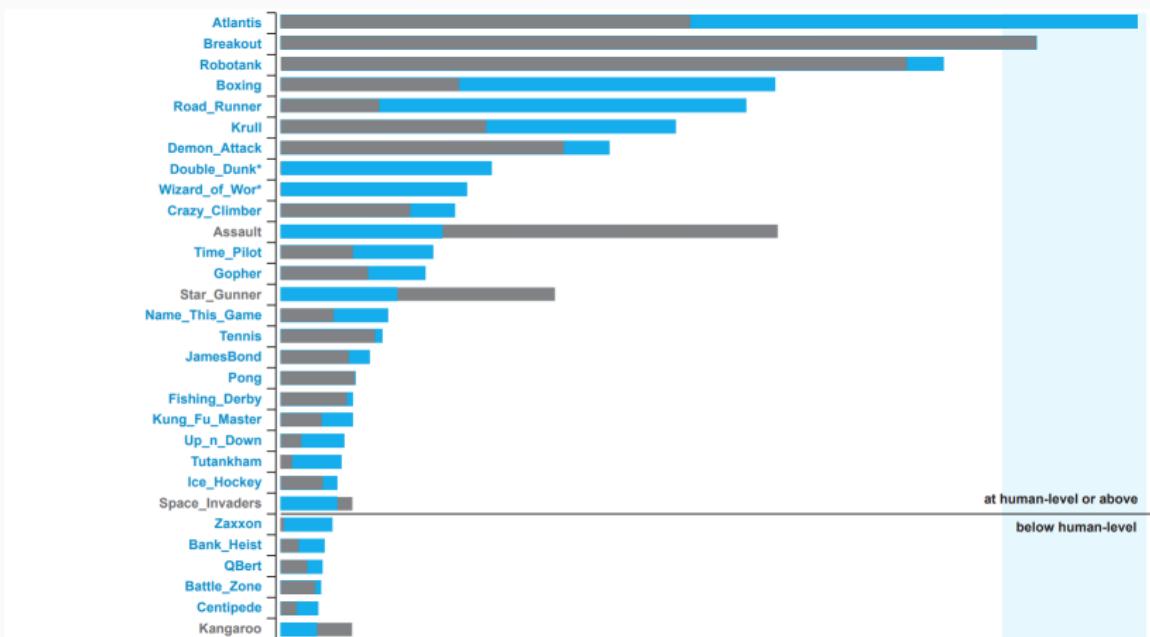
- (Nair, 2015), DeepMind: Gorila (General Reinforcement Learning Architecture)



- Параллелизум на кучу агентов, храним всё в единой памяти, достаём оттуда для обучения, которое тоже параллелизовано и время от времени синхронизируется.

СОВРЕМЕННЫЕ ACTOR-CRITIC АЛГОРИТМЫ

- Достаточно типичная параллельная архитектура, улучшала базовый DQN от (Mnih et al., 2013) в большинстве случаев:



СОВРЕМЕННЫЕ ACTOR-CRITIC АЛГОРИТМЫ

- (Mnih et al., 2016), DeepMind: важная работа для actor-critic методов; во-первых, асинхронное DQN обучение, где просто накапливаются градиенты и иногда сбрасываются:

Algorithm 1 Asynchronous one-step Q-learning - pseudocode for each actor-learner thread.

```
// Assume global shared  $\theta$ ,  $\theta^-$ , and counter  $T = 0$ .
Initialize thread step counter  $t \leftarrow 0$ 
Initialize target network weights  $\theta^- \leftarrow \theta$ 
Initialize network gradients  $d\theta \leftarrow 0$ 
Get initial state  $s$ 
repeat
    Take action  $a$  with  $\epsilon$ -greedy policy based on  $Q(s, a; \theta)$ 
    Receive new state  $s'$  and reward  $r$ 
     $y = \begin{cases} r & \text{for terminal } s' \\ r + \gamma \max_{a'} Q(s', a'; \theta^-) & \text{for non-terminal } s' \end{cases}$ 
    Accumulate gradients wrt  $\theta$ :  $d\theta \leftarrow d\theta + \frac{\partial(y - Q(s, a; \theta))^2}{\partial\theta}$ 
     $s = s'$ 
     $T \leftarrow T + 1$  and  $t \leftarrow t + 1$ 
    if  $T \bmod I_{target} == 0$  then
        Update the target network  $\theta^- \leftarrow \theta$ 
    end if
    if  $t \bmod I_{AsyncUpdate} == 0$  or  $s$  is terminal then
        Perform asynchronous update of  $\theta$  using  $d\theta$ .
        Clear gradients  $d\theta \leftarrow 0$ .
    end if
until  $T > T_{max}$ 
```

СОВРЕМЕННЫЕ ACTOR-CRITIC АЛГОРИТМЫ

- Интереснее для нас – A3C (asynchronous advantage actor-critic):

- поддерживаем $\pi(A_t|S_t, \theta)$ и $V(S_t, \mathbf{w})$;
- обновляем

$$\nabla_{\theta'} \log \pi(a_t|s_t, \theta') A(s_t, a_t, \theta, \theta_v),$$

где $A(s_t, a_t, \theta, \theta_v)$ – это оценка advantage function, т.е.

$$\sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}, \theta_v) - V(s_t, \theta_v);$$

- обычно у сети softmax для $\pi(a_t|s_t, \theta)$ и ещё отдельно выход для $V(s_t, \theta_v)$;
- ещё хорошо бы добавить exploration, для этого мы добавляем в целевую функцию регуляризатор в виде энтропии π :

$$\nabla_{\theta'} \log \pi(a_t|s_t, \theta') (R_t - V(s_t, \theta_v)) + \beta \nabla_{\theta'} H(\pi(a_t|s_t, \theta')).$$

СОВРЕМЕННЫЕ ACTOR-CRITIC АЛГОРИТМЫ

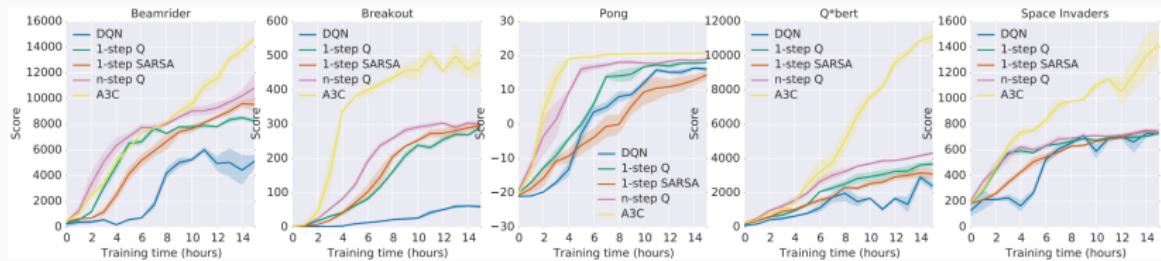
- Итого получается такой алгоритм:

Algorithm S3 Asynchronous advantage actor-critic - pseudocode for each actor-learner thread.

```
// Assume global shared parameter vectors  $\theta$  and  $\theta_v$  and global shared counter  $T = 0$ 
// Assume thread-specific parameter vectors  $\theta'$  and  $\theta'_v$ 
Initialize thread step counter  $t \leftarrow 1$ 
repeat
    Reset gradients:  $d\theta \leftarrow 0$  and  $d\theta_v \leftarrow 0$ .
    Synchronize thread-specific parameters  $\theta' = \theta$  and  $\theta'_v = \theta_v$ 
     $t_{start} = t$ 
    Get state  $s_t$ 
    repeat
        Perform  $a_t$  according to policy  $\pi(a_t|s_t; \theta')$ 
        Receive reward  $r_t$  and new state  $s_{t+1}$ 
         $t \leftarrow t + 1$ 
         $T \leftarrow T + 1$ 
    until terminal  $s_t$  or  $t - t_{start} == t_{max}$ 
     $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t // \text{Bootstrap from last state} \end{cases}$ 
    for  $i \in \{t - 1, \dots, t_{start}\}$  do
         $R \leftarrow r_i + \gamma R$ 
        Accumulate gradients wrt  $\theta'$ :  $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i; \theta')(R - V(s_i; \theta'_v))$ 
        Accumulate gradients wrt  $\theta'_v$ :  $d\theta_v \leftarrow d\theta_v + \partial (R - V(s_i; \theta'_v))^2 / \partial \theta'_v$ 
    end for
    Perform asynchronous update of  $\theta$  using  $d\theta$  and of  $\theta_v$  using  $d\theta_v$ .
until  $T > T_{max}$ 
```

СОВРЕМЕННЫЕ ACTOR-CRITIC АЛГОРИТМЫ

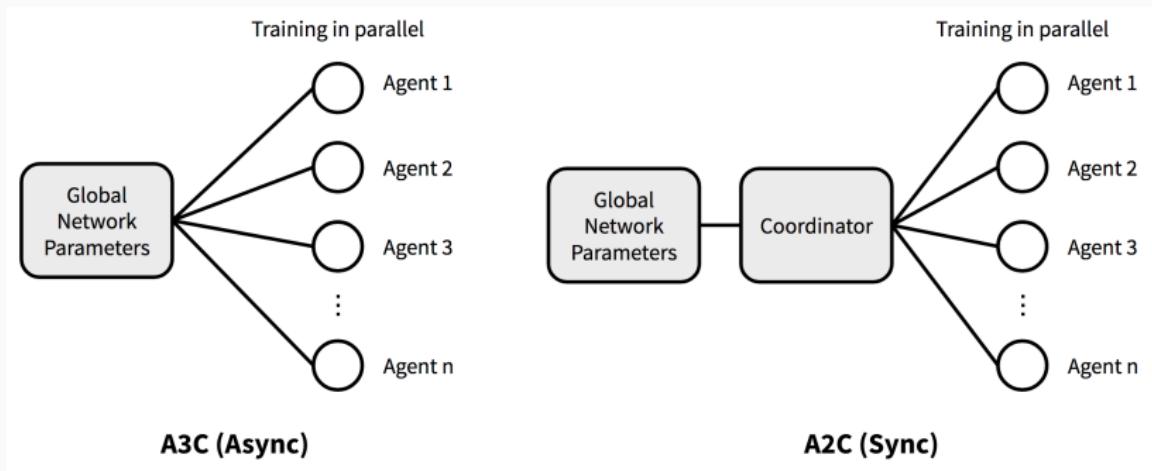
- И оказывается, что АЗС побеждает всех подряд, да ещё и обучается быстрее:



Method	Training Time	Mean	Median
DQN	8 days on GPU	121.9%	47.5%
Gorila	4 days, 100 machines	215.2%	71.3%
D-DQN	8 days on GPU	332.9%	110.9%
Dueling D-DQN	8 days on GPU	343.8%	117.1%
Prioritized DQN	8 days on GPU	463.6%	127.6%
A3C, FF	1 day on CPU	344.1%	68.2%
A3C, FF	4 days on CPU	496.8%	116.6%
A3C, LSTM	4 days on CPU	623.0%	112.6%

СОВРЕМЕННЫЕ ACTOR-CRITIC АЛГОРИТМЫ

- Потом, правда, выяснилось, что A2C (то же самое, но без асинхронности) может работать даже лучше



DETERMINISTIC POLICY GRADIENT

- Deterministic policy gradient (DPG): можно ли сделать то же самое с детерминированной стратегией $a = f(s; \theta)$ на непрерывном множестве действий?
- Кажется, что вряд ли, но вспомним, что целевая функция – это всё равно ожидание по состояниям, по времени $\mu_f(s)$, которое стратегия там проводит:

$$\theta_{k+1} = \theta_k + \alpha \mathbb{E}_{s \sim \mu_f} [\nabla_\theta Q_f(s, f(s; \theta))] .$$

- Возьмём градиент:

$$\theta_{k+1} = \theta_k + \alpha \mathbb{E}_{s \sim \mu_f} [\nabla_\theta f(s; \theta) \nabla_a Q_f(s, a) |_{a=f(s; \theta)}] .$$

DETERMINISTIC POLICY GRADIENT

- И тогда можно сделать всё то же самое, доказать policy gradient теорему, off-policy actor-critic и т.д. (Silver et al., 2014)
- Например, такой алгоритм получится для on-policy actor-critic (Sarsa):

$$\begin{aligned}\delta_t &= R_t + \gamma Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t, a_t), \\ \mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha_w \delta_t \nabla_{\mathbf{w}} Q_w(s_t, a_t), \\ \theta_{t+1} &= \theta_t + \alpha_\theta \nabla_\theta f(s_t, \theta) \nabla_a Q_w(s_t, a) \mid_{a=f_\theta(s)}\end{aligned}$$

- А в off-policy даже и менять теперь ничего не надо, просто ожидание по β , а цель – это Q для стратегии f :

$$\begin{aligned}\delta_t &= R_t + \gamma Q_w(s_{t+1}, f(s_{t+1}, \theta)) - Q_w(s_t, a_t), \\ \mathbf{w}_{t+1} &= \mathbf{w}_t + \alpha_w \delta_t \nabla_{\mathbf{w}} Q_w(s_t, a_t), \\ \theta_{t+1} &= \theta_t + \alpha_\theta \nabla_\theta f(s_t, \theta) \nabla_a Q_w(s_t, a) \mid_{a=f_\theta(s)}\end{aligned}$$

DETERMINISTIC POLICY GRADIENT

- Нейросети сюда применились в (Lillicrap et al., 2016): DDPG (Deep Deterministic Policy Gradient)

Algorithm 1 DDPG algorithm

Randomly initialize critic network $Q(s, a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ with weights θ^Q and θ^μ .

Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^Q, \theta^{\mu'} \leftarrow \theta^\mu$

Initialize replay buffer R

for episode = 1, M **do**

 Initialize a random process \mathcal{N} for action exploration

 Receive initial observation state s_1

for t = 1, T **do**

 Select action $a_t = \mu(s_t|\theta^\mu) + \mathcal{N}_t$ according to the current policy and exploration noise

 Execute action a_t and observe reward r_t and observe new state s_{t+1}

 Store transition (s_t, a_t, r_t, s_{t+1}) in R

 Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from R

 Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$

 Update critic by minimizing the loss: $L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i|\theta^Q))^2$

 Update the actor policy using the sampled policy gradient:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s|\theta^\mu)|_{s_i}$$

 Update the target networks:

$$\theta^{Q'} \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau \theta^\mu + (1 - \tau) \theta^{\mu'}$$

end for

end for

DETERMINISTIC POLICY GRADIENT

- (Barth-Maron et al., 2018): можно то же самое с распределениями сделать, D4PG (distributed distributional deep deterministic)

Algorithm 1 D4PG

Input: batch size M , trajectory length N , number of actors K , replay size R , exploration constant ϵ , initial learning rates α_0 and β_0

- 1: Initialize network weights (θ, w) at random
- 2: Initialize target weights $(\theta', w') \leftarrow (\theta, w)$
- 3: Launch K actors and replicate network weights (θ, w) to each actor
- 4: **for** $t = 1, \dots, T$ **do**
- 5: Sample M transitions $(\mathbf{x}_{i:i+N}, \mathbf{a}_{i:i+N-1}, r_{i:i+N-1})$ of length N from replay with priority p_i
- 6: Construct the target distributions $Y_i = \sum_{n=0}^{N-1} \gamma^n r_{i+n} + \gamma^N Z_{w'}(\mathbf{x}_{i+N}, \pi_{\theta'}(\mathbf{x}_{i+N}))$
- 7: Compute the actor and critic updates

$$\delta_w = \frac{1}{M} \sum_i \nabla_w (Rp_i)^{-1} d(Y_i, Z_w(\mathbf{x}_i, \mathbf{a}_i))$$
$$\delta_\theta = \frac{1}{M} \sum_i \nabla_\theta \pi_\theta(\mathbf{x}_i) \mathbb{E}[\nabla_\mathbf{a} Z_w(\mathbf{x}_i, \mathbf{a})] \Big|_{\mathbf{a}=\pi_\theta(\mathbf{x}_i)}$$

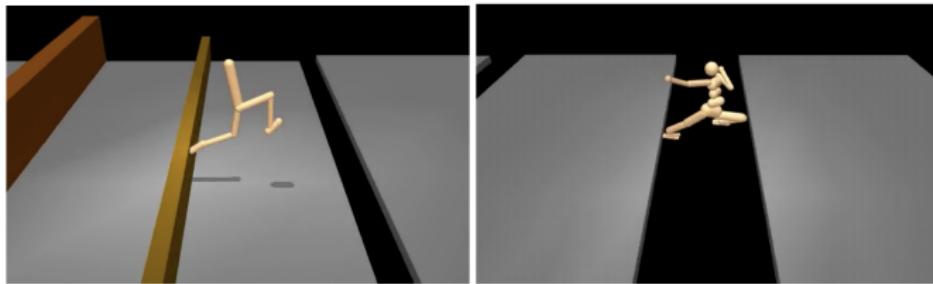
- 8: Update network parameters $\theta \leftarrow \theta + \alpha_t \delta_\theta$, $w \leftarrow w + \beta_t \delta_w$
- 9: If $t = 0 \bmod t_{\text{target}}$, update the target networks $(\theta', w') \leftarrow (\theta, w)$
- 10: If $t = 0 \bmod t_{\text{actors}}$, replicate network weights to the actors
- 11: **end for**
- 12: **return** policy parameters θ

Actor

- 1: **repeat**
- 2: Sample action $\mathbf{a} = \pi_\theta(\mathbf{x}) + \epsilon \mathcal{N}(0, 1)$
- 3: Execute action \mathbf{a} , observe reward r and state \mathbf{x}'
- 4: Store $(\mathbf{x}, \mathbf{a}, r, \mathbf{x}')$ in replay
- 5: **until** learner finishes

DETERMINISTIC POLICY GRADIENT

- Это всё теперь, кстати, можно применять к роботике, т.е. к ситуациям с непрерывными действиями:



DETERMINISTIC POLICY GRADIENT

- Получается хорошо:

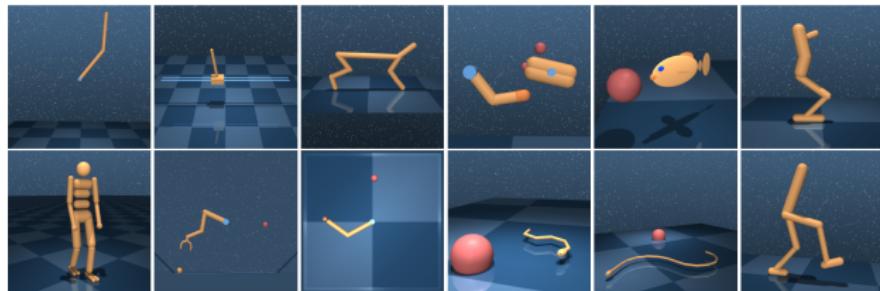
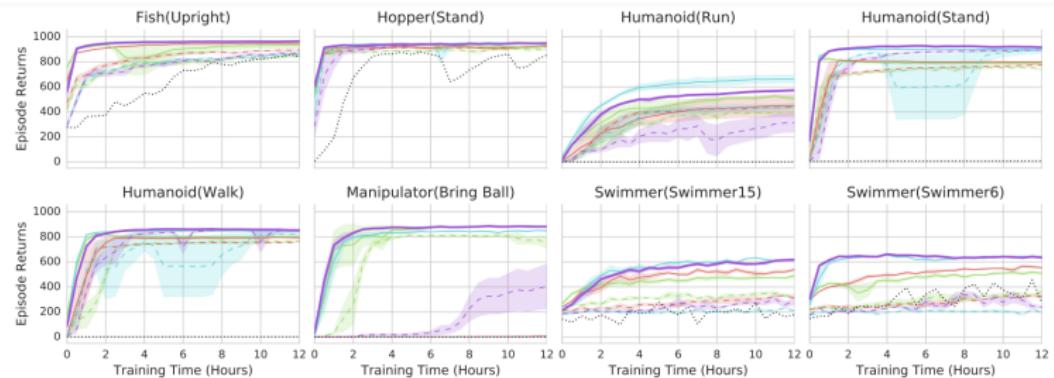


Figure 9: Control Suite domains used for benchmarking. *Top*: acrobot, cartpole, cheetah, finger, fish, hopper. *Bottom*: humanoid, manipulator, pendulum, reacher, swimmer6, swimmer15, walker.



- И последняя модификация: Proximal Policy Optimization (PPO)
- От OpenAI (Schulman et al., 2018):
 - обычно мы оптимизируем через автоматическое дифференцирование функции ошибки

$$L(\theta) = \mathbb{E}_t [\log \pi(a_t | s_t, \theta) A_t],$$

где A_t – оценка advantage function в момент времени t ;

- но если мы будем так вот напрямую оптимизировать, делая несколько шагов по данным с одной и той же траектории, ничего хорошего не получится;
- и вообще непонятно, как выбирать скорость обучения, как далеко идти и т.д.

PROXIMAL POLICY OPTIMIZATION

- (Schulman et al., 2017): TRPO, Trust Region Policy Optimization
- Давайте оптимизировать

$$L(\theta) = \mathbb{E}_t \left[\frac{\pi(a_t|s_t, \theta)}{\pi(a_t|s_t, \theta_{\text{old}})} \log \pi(a_t|s_t, \theta) A_t \right] = \mathbb{E}_t [r_t(\theta) \log \pi(a_t|s_t, \theta) A_t]$$

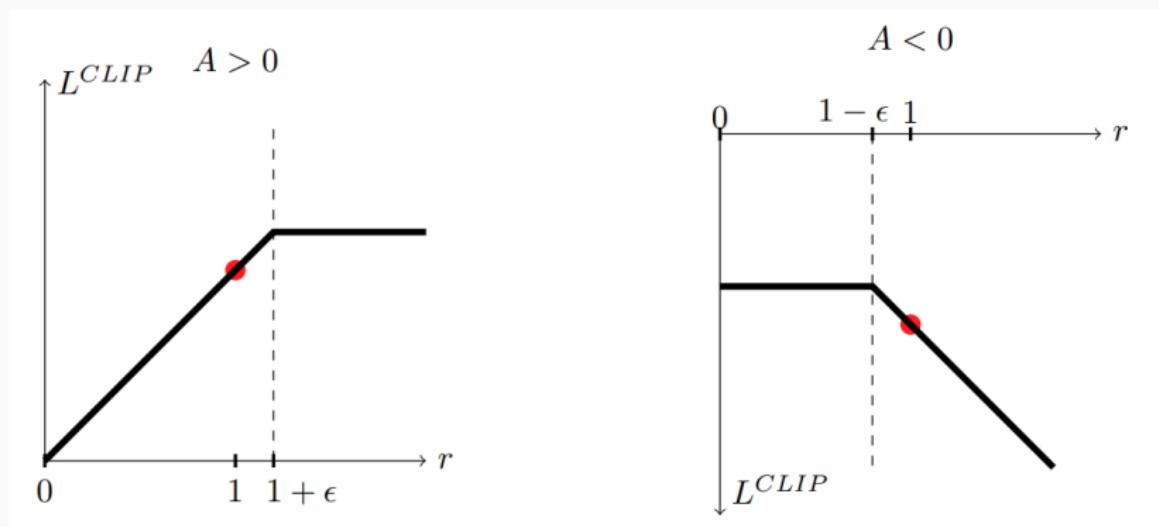
пока $\mathbb{E}_t [\text{KL}(\pi(\cdot|s_t, \theta_{\text{old}}) \| \pi(\cdot|s_t, \theta))] \leq \delta$.

- Условие можно заменить на регуляризатор, но трудно!
Значение регуляризатора будет меняться даже в процессе обучения одной задачи.
- А с условием хорошо работает, но нелегко реализовать вычислительно.

PROXIMAL POLICY OPTIMIZATION

- PPO – давайте обрежем вместо регуляризатора:

$$L(\theta) = \mathbb{E}_t [\min (r_t(\theta) A_t, \text{clip} (r_t(\theta), 1 - \epsilon, 1 + \epsilon) \log \pi (a_t | s_t, \theta) A_t)].$$



PROXIMAL POLICY OPTIMIZATION

- Сам алгоритм теперь простой, считаем

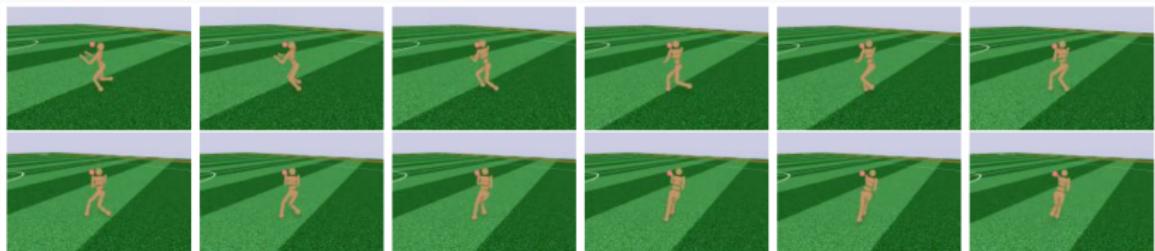
$A_t = -V(s_t) + r_t + \gamma r_{t+1} + \dots + \gamma^{T-t} V(s_T)$ и всё обучаем распределённо:

Algorithm 1 PPO, Actor-Critic Style

```
for iteration=1, 2, ... do
    for actor=1, 2, ..., N do
        Run policy  $\pi_{\theta_{\text{old}}}$  in environment for  $T$  timesteps
        Compute advantage estimates  $\hat{A}_1, \dots, \hat{A}_T$ 
    end for
    Optimize surrogate  $L$  wrt  $\theta$ , with  $K$  epochs and minibatch size  $M \leq NT$ 
     $\theta_{\text{old}} \leftarrow \theta$ 
end for
```

PROXIMAL POLICY OPTIMIZATION

- Получается хорошо:



- <https://openai.com/blog/openai-baselines-ppo/>

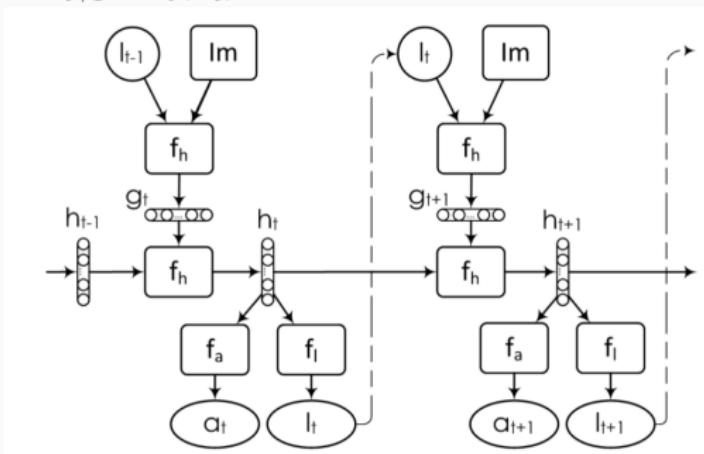
POLICY GRADIENT для других ПРИМЕНЕНИЙ

ДРУГИЕ ПРИЛОЖЕНИЯ POLICY GRADIENT

- RL – это про то, как агенты действуют в окружающей среде.
- То есть игры или роботы – это естественные задачи для RL.
- Но оказывается, что есть и другие задачи, которые тоже логично формулировать как «награды, которые выдают в конце эпизода, нечасто»
- И там тоже появляется RL, иногда в совершенно неожиданных местах.

RECURRENT VISUAL ATTENTION

- Первый пример – механизмы внимания.
- По-настоящему attention в современном виде появилось в DL в (Mnih et al., 2014), «Recurrent Models of Visual Attention»:
 - из предыдущего \mathbf{h}_{t-1} и положения l_t для нового «взгляда» f_g делает \mathbf{g}_t , вход для шага t ;
 - из \mathbf{h}_{t-1} и \mathbf{g}_t функцией f_h получается \mathbf{h}_t ;
 - из него – «действие» $a_t = f_a(\mathbf{h}_t)$ и положение следующего «взгляда» $l_{t+1} = f_l(\mathbf{h}_t)$.



- Давайте разберёмся в модели формально:

$$\begin{aligned}\mathbf{g}_t &= f_g(\mathbf{x}_t, \mathbf{l}_{t-1}; \theta_g), \\ \mathbf{h}_t &= f_h(\mathbf{h}_{t-1}, \mathbf{g}_t; \theta_h), \\ \mathbf{l}_t &\sim p(\cdot \mid f_l(\mathbf{h}_t; \theta_l)), \\ a_t &\sim p(\cdot \mid f_a(\mathbf{h}_t; \theta_a)).\end{aligned}$$

- После очередного действия получается новое наблюдение \mathbf{x}_{t+1} и награда r_t , которая будет скорее всего в конце, после всех шагов, за правильную классификацию.
- Что это напоминает?..

RECURRENT VISUAL ATTENTION

- ...о да, это reinforcement learning!
- Выучить надо стохастическую стратегию $\pi((\mathbf{l}_t, a_t) | \mathbf{s}_{1:t}; \theta)$, которая по истории будет выдавать следующее действие.
- У нас π задаётся через RNN, а оптимизировать надо

$$J(\theta) = \mathbb{E}_{p(\mathbf{s}_{1:T}; \theta)} [R] = \mathbb{E}_{p(\mathbf{s}_{1:T}; \theta)} \left[\sum_{t=1}^T r_t \right].$$

- Выглядит очень сложно – ожидание по последовательностям действий, т.е. по пространству большой размерности.

RECURRENT VISUAL ATTENTION

- Но можно воспользоваться всё тем же REINFORCE, в котором доказывается и используется выборочная оценка этого ожидания
- Нам надо оптимизировать

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=1}^T r_t(s_t, a_t) \right] \approx \frac{1}{M} \sum_{i=1}^M \sum_{t=1}^T r(s_t^{(i)}, a_t^{(i)}),$$

где мы взяли M примеров траекторий τ .

- Определим $r(\tau) = \sum_t r(s_t, a_t)$. Тогда

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} [r(\tau)] = \int \pi_\theta(\tau) r(\tau) d\tau,$$

т.е. тот самый страшный интеграл по траекториям.

- Но оказывается, что можно продифференцировать по θ ...

- Продифференцируем по θ :

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \int \nabla_{\theta} \pi_{\theta}(\tau) r(\tau) d\tau \\&= \int \pi_{\theta}(\tau) \frac{\nabla_{\theta} \pi_{\theta}(\tau)}{\pi_{\theta}(\tau)} r(\tau) d\tau \\&= \int \pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau) d\tau \\&= \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} [\nabla_{\theta} \log \pi_{\theta}(\tau) r(\tau)] \\&\approx \frac{1}{M} \sum_{i=1}^M \nabla_{\theta} \log \pi_{\theta}(\tau^{(i)}) r^{(i)}(\tau),\end{aligned}$$

если приблизить выборкой; но сначала давайте ещё посмотрим на $\pi_{\theta}(\tau)$...

RECURRENT VISUAL ATTENTION

- Вероятность определяется как

$$\pi_\theta(\tau) = p(s_1) \prod_{t=1}^T \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t).$$

- Берём логарифм, а потом заметим, что от θ зависят только действия:

$$\begin{aligned}\nabla_\theta \log \pi_\theta(\tau) &= \\ &= \nabla_\theta \left(\log p(s_1) + \sum_{t=1}^T \log \pi_\theta(a_t | s_t) + \sum_{t=1}^T \log p(s_{t+1} | s_t, a_t) \right) = \\ &= \sum_{t=1}^T \nabla_\theta \log \pi_\theta(a_t | s_t).\end{aligned}$$

RECURRENT VISUAL ATTENTION

- Итого получается вполне tractable градиент:

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{\tau \sim \pi_{\theta}(\tau)} \left[r(\tau) \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \right] \\ &\approx \frac{1}{M} \sum_{i=1}^M r(\tau^{(i)}) \sum_{t=1}^T \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}).\end{aligned}$$

- У нас тоже можно считать, что награда R даётся целиком:

$$\nabla_{\theta} J(\theta) \approx \frac{1}{M} \sum_{i=1}^M \sum_{t=1}^T \nabla_{\theta} \log \pi \left(l_t^{(i)}, a_t^{(i)} | s_{1:t}^{(i)}; \theta \right) R^{(i)}.$$

- Т.е. надо уметь считать $\log \pi \left(l_t^{(i)}, a_t^{(i)} | s_{1:t}^{(i)}; \theta \right)$, но в случае RNN это просто градиент сети, который можно посчитать через backpropagation.
- Ещё можно сделать частично supervised loss на последнем шаге, где мы знаем классификацию.

RECURRENT VISUAL ATTENTION

- Результаты:



(a) Translated MNIST inputs.



(b) Cluttered Translated MNIST inputs.

(a) 28x28 MNIST

Model	Error
FC, 2 layers (256 hiddens each)	1.69%
Convolutional, 2 layers	1.21%
RAM, 2 glimpses, 8×8 , 1 scale	3.79%
RAM, 3 glimpses, 8×8 , 1 scale	1.51%
RAM, 4 glimpses, 8×8 , 1 scale	1.54%
RAM, 5 glimpses, 8×8 , 1 scale	1.34%
RAM, 6 glimpses, 8×8 , 1 scale	1.12%
RAM, 7 glimpses, 8×8 , 1 scale	1.07%

(b) 60x60 Translated MNIST

Model	Error
FC, 2 layers (64 hiddens each)	6.42%
FC, 2 layers (256 hiddens each)	2.63%
Convolutional, 2 layers	1.62%
RAM, 4 glimpses, 12×12 , 3 scales	1.54%
RAM, 6 glimpses, 12×12 , 3 scales	1.22%
RAM, 8 glimpses, 12×12 , 3 scales	1.2%

RECURRENT VISUAL ATTENTION

- Результаты:

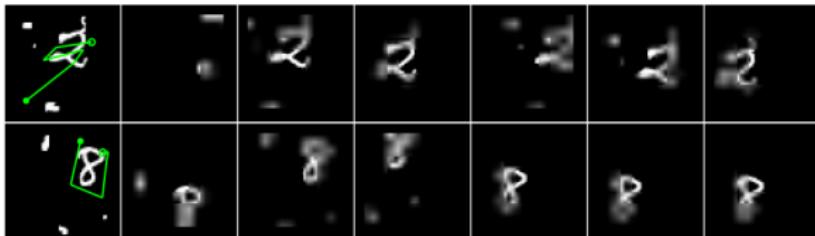
(a) 60x60 Cluttered Translated MNIST

Model	Error
FC, 2 layers (64 hiddens each)	28.58%
FC, 2 layers (256 hiddens each)	11.96%
Convolutional, 2 layers	8.09%
RAM, 4 glimpses, 12×12 , 3 scales	4.96%
RAM, 6 glimpses, 12×12 , 3 scales	4.08%
RAM, 8 glimpses, 12×12 , 3 scales	4.04%
RAM, 8 random glimpses	14.4%

(b) 100x100 Cluttered Translated MNIST

Model	Error
Convolutional, 2 layers	14.35%
RAM, 4 glimpses, 12×12 , 4 scales	9.41%
RAM, 6 glimpses, 12×12 , 4 scales	8.31%
RAM, 8 glimpses, 12×12 , 4 scales	8.11%
RAM, 8 random glimpses	28.4%

- А вот как внимание гуляет по картинке:



ПОРОЖДЕНИЕ ДИАЛОГОВ

- Второй пример задачи, в которой получается что-то похожее – диалоговые агенты (Li et al., 2016)
- Диалоговые агенты, обучающиеся максимизировать правдоподобие в seq2seq моделях, часто застревают в циклах, отвечают на всё "I don't know" и т.п.

Baseline mutual information model (Li et al. 2015)	Proposed reinforcement learning model
A: Where are you going? (1) B: I'm going to the restroom. (2) A: See you later. (3) B: See you later. (4) A: See you later. (5) B: See you later. (6) A: how old are you? (1) B: I'm 16. (2) A: 16? (3) B: I don't know what you are talking about. (4) A: You don't know what you are saying. (5) B: I don't know what you are talking about . (6) A: You don't know what you are saying. (7) ...	A: Where are you going? (1) B: I'm going to the police station. (2) A: I'll come with you. (3) B: No, no, no, no, you're not going anywhere. (4) A: Why? (5) B: I need you to stay here. (6) A: I don't know what you are talking about. (7) ... A: How old are you? (1) B: I'm 16. Why are you asking? (2) A I thought you were 12. (3) B: What made you think so? (4) A: I don't know what you are talking about. (5) B: You don't know what you are saying. (6) ...

- Справа здесь агенты, обученные через RL. Но как?

ПОРОЖДЕНИЕ ДИАЛОГОВ

- Получается такая модель:
 - два агента разговаривают друг с другом – это эпизоды обучения
 - действие a – это реплика в диалоге, т.е. по сути действий бесконечно много
 - состояние $[p_i, q_i]$ – это две предыдущие реплики (наша и собеседника)
 - стратегия – это $p(p_{i+1} | p_i, q_i)$, параметризованная, скажем, через LSTM
 - награда r – это сложная штука, состоит из нескольких слагаемых; давайте подробнее...

ПОРОЖДЕНИЕ ДИАЛОГОВ

- Слагаемые в r :

- ease of answering: штраф за вероятность ответов типа «не знаю»

$$r_1 = -\frac{1}{N_{\S}} \sum_{s \in \S} \frac{1}{N_s} \log p_{\text{seq2seq}}(s \mid a);$$

- information flow: штраф за семантическую похожесть p_{i+1} на p_i

$$r_2 = -\log \cos(h_{p_i}, h_{p_{i+1}});$$

- semantic coherence: награда за правдоподобие этого ответа и реплики собеседника при обратном порождении:

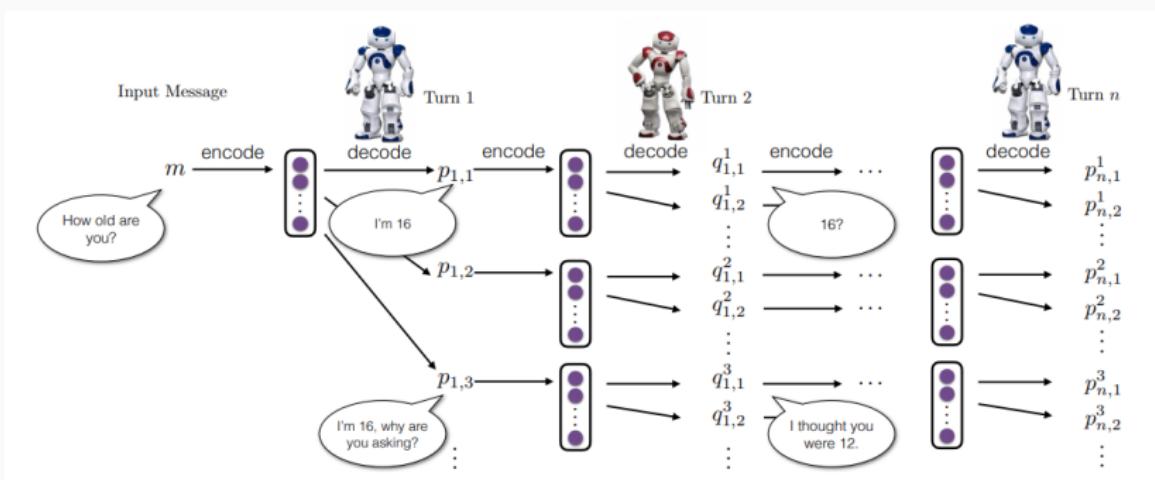
$$r_3 = \frac{1}{N_a} \log p_{\text{seq2seq}}(a \mid p_i, q_i) + \frac{1}{N_{q_i}} \log p_{\text{seq2seq}}^{\text{backward}}(q_i \mid a);$$

- Итого берём

$$r(a, [p_i, q_i]) = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3$$

ПОРОЖДЕНИЕ ДИАЛОГОВ

- Получается такая схема:



ПОРОЖДЕНИЕ ДИАЛОГОВ

- А обучение, как обычно, через REINFORCE:

$$J(\theta) = \mathbb{E}_{p(a_{1:T})} \left[\sum_{i=1}^T r(a_i, [p_i, q_i]) \right]$$

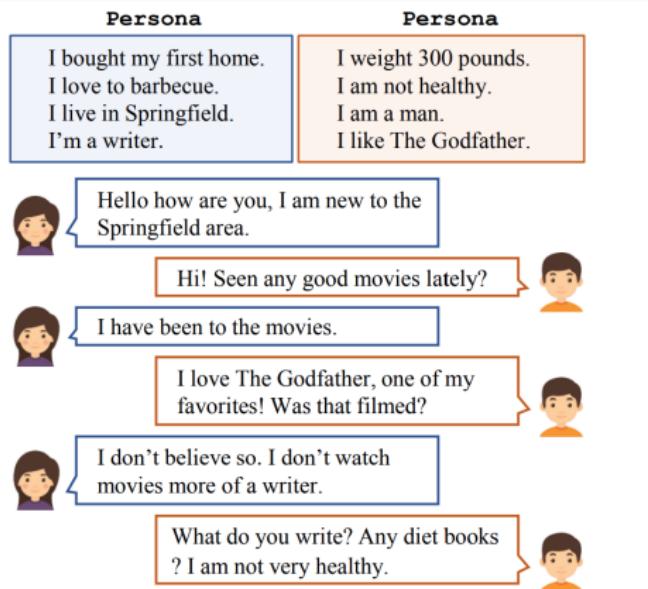
$$\nabla_{\theta} J \approx \sum_i \nabla \log p(a_i | p_i, q_i) \sum_{i=1}^T r(a_i, [p_i, q_i])$$

- И действительно получается лучше:

Input Message	Mutual Information Model	RL
How old are you?	I'm 16.	I'm 16, why are you asking?
What's your full name?	I have no idea.	What's yours?
I don't want to go home tonight.	Really?	Why?
Do you have any feelings for me?	I don't know what you are talking about.	I don't want to hurt your feelings.
How much time do you have here?	Not long enough. Sorry, sir.	Ten seconds.
Shall we get started?	Of course!	Yes. We've got a lot of work to do here.
Do you play football?	No, i don't	Yes. I love football!
We'd have to talk to him.	I mean, he's a good guy	About what ?
How come you never say it?	Because I don't want to hurt you.	I don't think it's a good idea to say it.

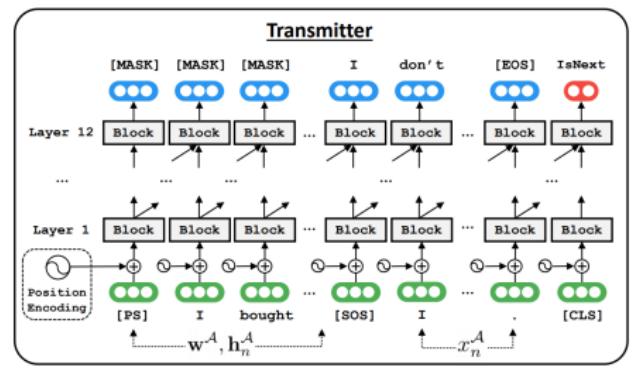
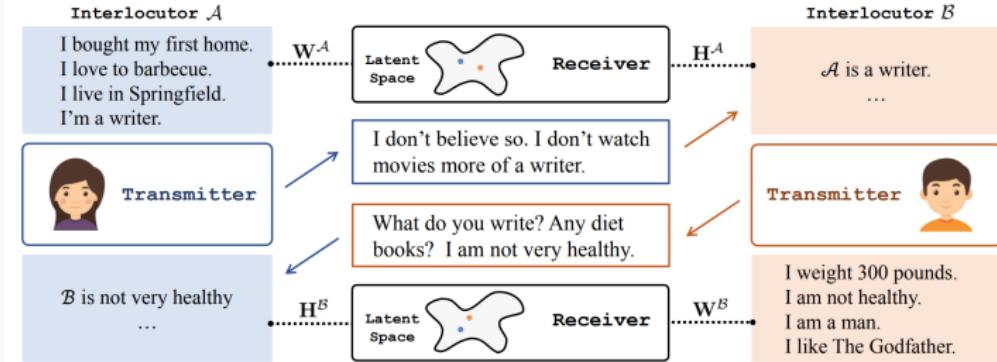
ПОРОЖДЕНИЕ ДИАЛОГОВ

- Пример: You Impress Me (Liu et al., 2020)
- Chit-chat dialogue на датасете PersonaChat с персоналиями



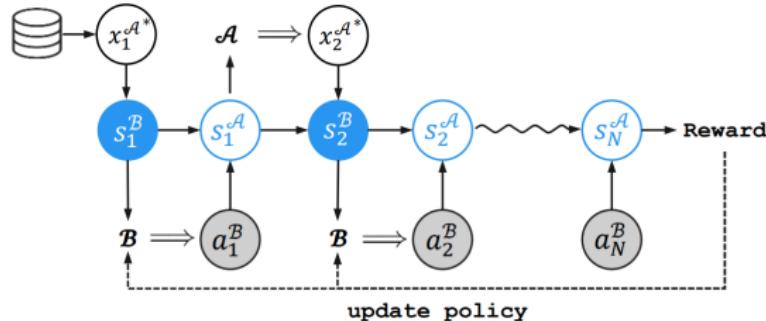
ПОРОЖДЕНИЕ ДИАЛОГОВ

- Схема агента:

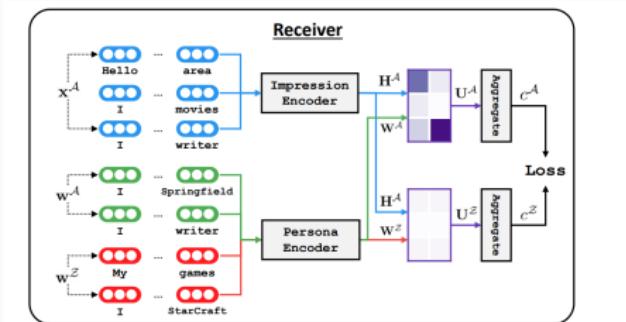


ПОРОЖДЕНИЕ ДИАЛОГОВ

- Обучаются сначала supervised, потом fine-tuning через RL:



- Есть ещё Receiver, который оценивает, насколько персона похожа:



ПОРОЖДЕНИЕ ДИАЛОГОВ

- Получается, увы, неплохо:

Model	1 (%)	2 (%)	3 (%)	4 (%)	Avg
Lost In Conversation	26.3	48.7	22.0	3.0	2.017
Transfertransfo	41.7	25.3	28.7	4.3	1.956
\mathcal{P}^2 BOT (Our)	18.9	26.3	28.6	26.2	2.621

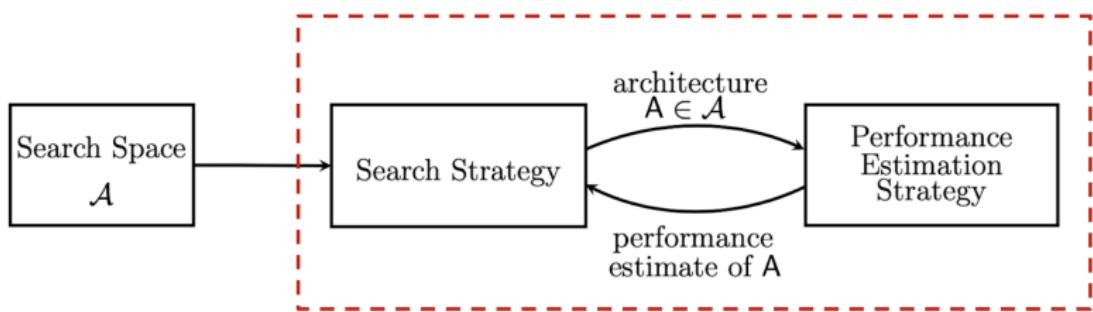
Table 2: Human evaluation results.

PERSONA	i. I love new kids on the block. ii. I was born in the early 80's. iii. I also like old school hip hop. iv. My favorite toy as a child as my lite brite.	i. I am a blonde girl with really short hair. ii. I love wearing skinny jeans and leggings. iii. I'm rather skinny as I like to stay in shape. iv. My favorite hobbies are listening to music and playing video games.	i. I've an allergy to water. ii. I'm a famous twitch streamer. iii. I helped design the game StarCraft. iv. I am working on building a robot to be my body double.
CONTEXT	Do you like clowns? They horrify me!	What video game? I relax by knitting. Love it!	Cool. I did hardcore exercise four hours everyday.
* HUMAN	No! My favorite toy as a kid was lite brite because I was so scared!	I was playing Fallout. What do you look like? I'm blonde thin.	That's a lot of exercising. I hate exercising. I just work on my robot most days .
* \mathcal{P}^2 BOT	I do not like them at all. I am a little old for them .	I play RPGs. I like to wear skinny jeans and tees .	I do not exercise at all. I'm busy with my robot project .
* TRANSFERTRANSFO	I do not have time for clowns. Do you like to listen to music ?	My fav color is blue, I have a lot of friends in my group.	I work at a computer company. I could make you an Android!
* LOST IN CONVERSATION	I love clowns. they are my favorite.	I love HALO 3, what do you knit?	That sounds like a lot of fun !

NEURAL ARCHITECTURE SEARCH

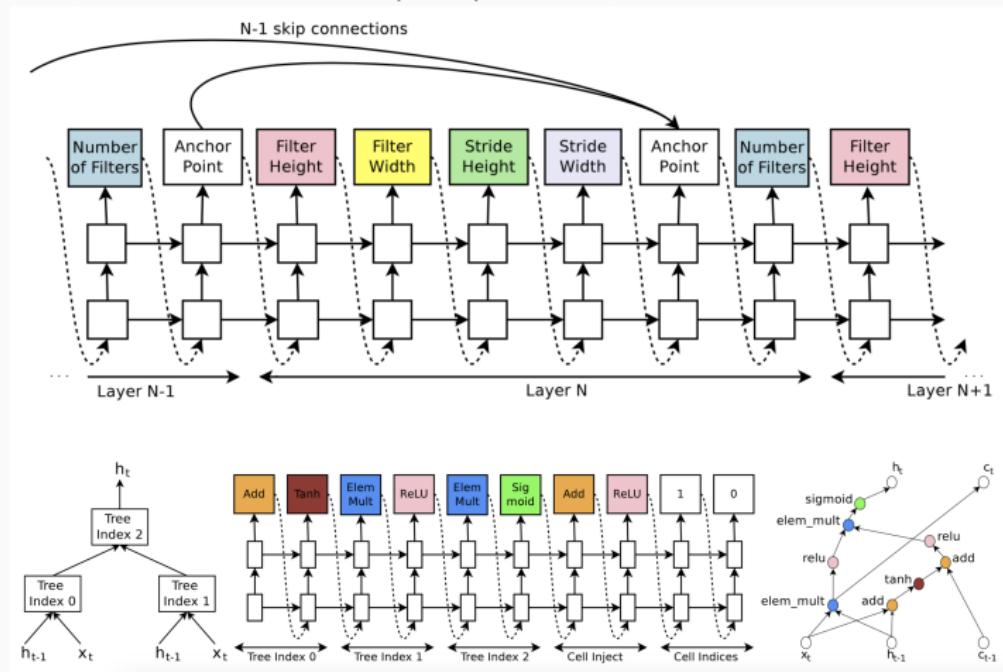
- Neural architecture search: способ автоматически найти наилучшую архитектуру для данной задачи или класса задач:
 - определим пространство поиска (как базовые операции объединяются в архитектуру сети)
 - алгоритм поиска сэмплирует популяцию архитектур, используя метрики как вознаграждение
 - алгоритм оценки тоже непростой, надо очень много оценивать, поэтому желательно быстро это делать

One-shot approach:
learning model architecture parameters and weights together



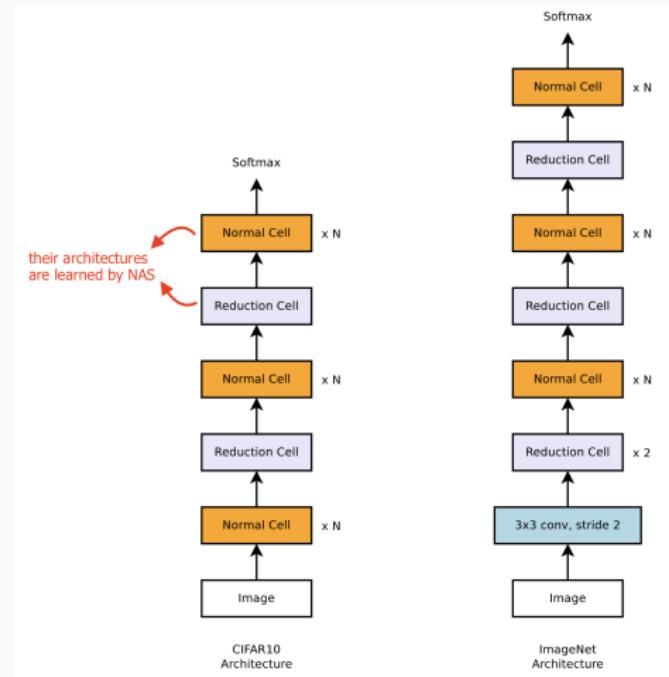
NEURAL ARCHITECTURE SEARCH

- В ранних работах (Zoph, Le, 2017; Baker et al., 2017) была последовательная реализация, где перебирались слой за слоем; очень большое пространство, всё долго и сложно:



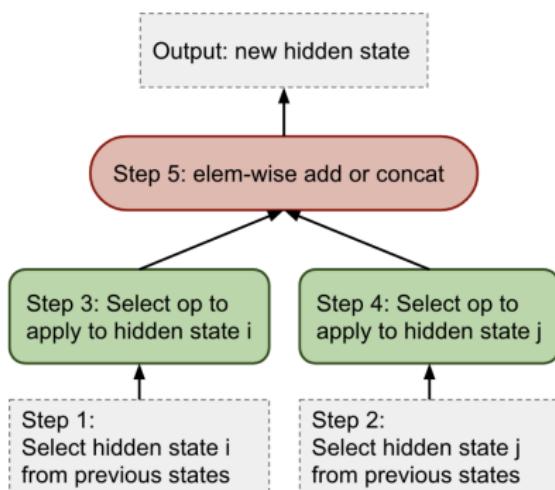
NEURAL ARCHITECTURE SEARCH

- Потом (Zoph et al., 2018) перешли на структуру из повторяющихся модулей (motif-based architecture), как в Inception — NASNet:

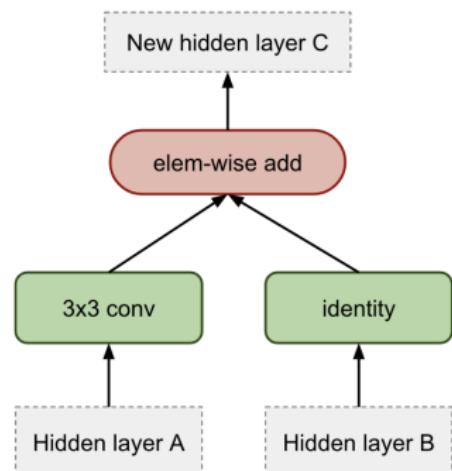


NEURAL ARCHITECTURE SEARCH

- Каждая ячейка состоит из нескольких блоков, каждый блок порождается некоторыми классификаторами, которые выбирают, что делает эта часть



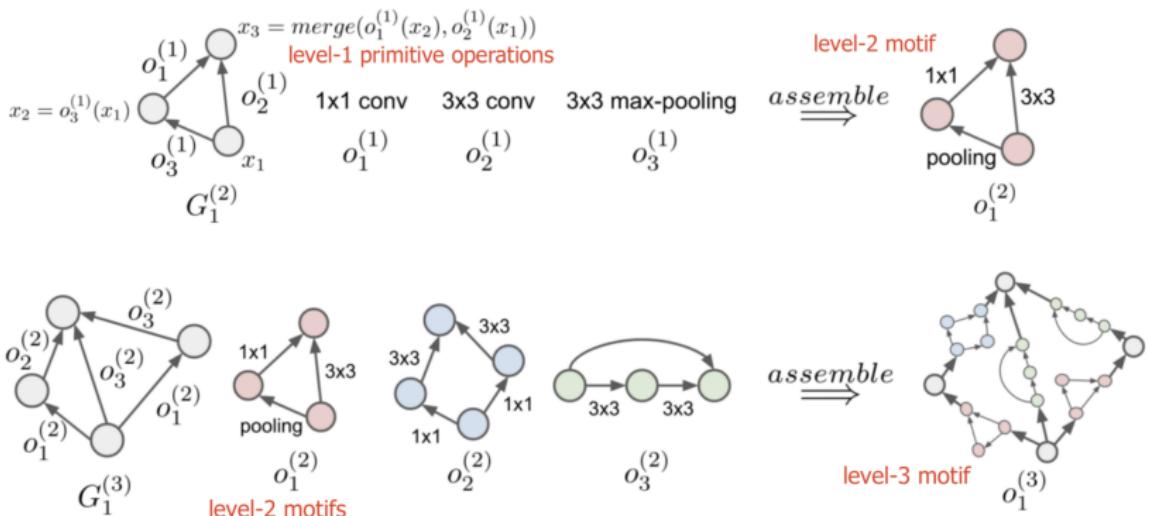
(a) 5 discrete choices in each block



(b) A concrete example

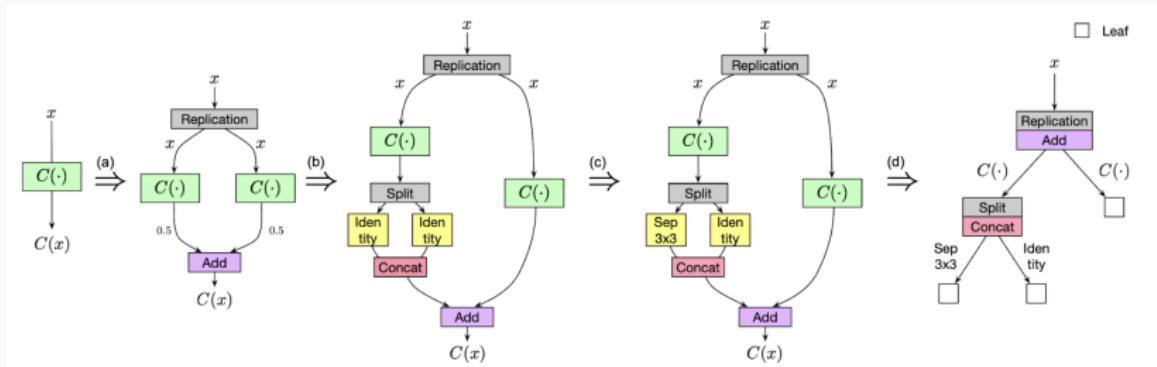
NEURAL ARCHITECTURE SEARCH

- Можно сделать и явным образом Hierarchical NAS (Liu et al., 2017), где более примитивные мотивы объединяются в мотивы следующего уровня:



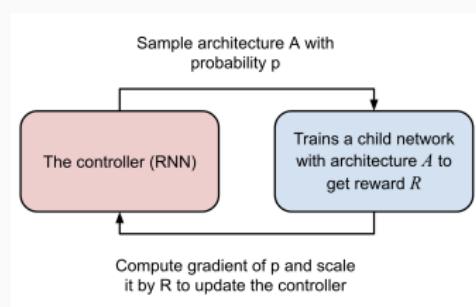
NEURAL ARCHITECTURE SEARCH

- Или задать правила иерархической трансформации (Cai et al., 2018):



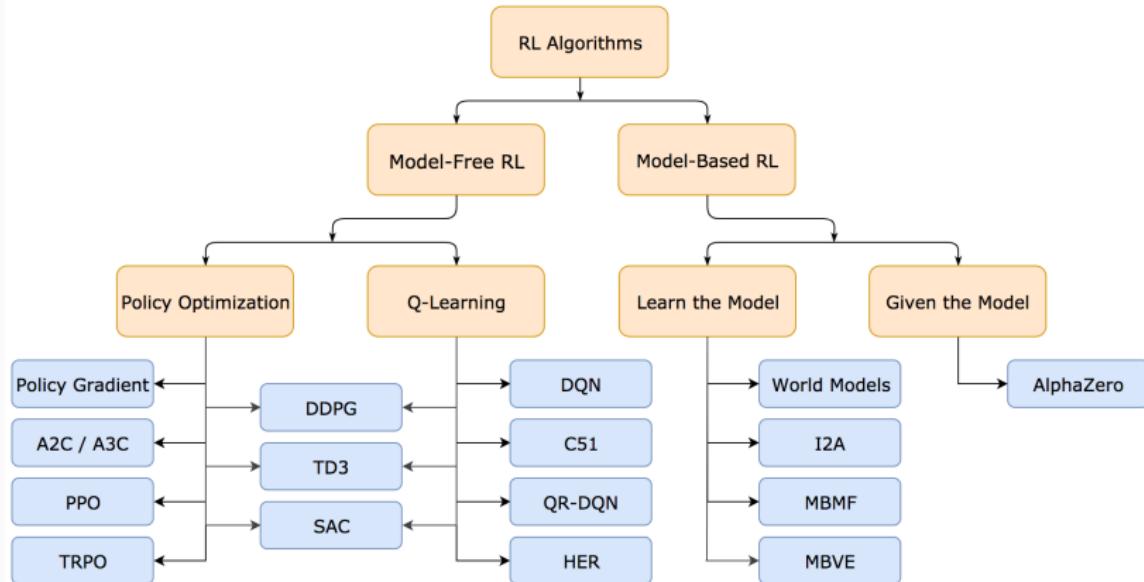
NEURAL ARCHITECTURE SEARCH

- А алгоритм поиска — это как раз может быть RL:
 - пространство действий задаёт набор сетей, которые можно породить;
 - награда — это точность сети или другие метрики;
 - и параметры выбора действия (сети) обучаются через policy gradient.



БОЛЕЕ СЛОЖНЫЕ И
МУЛЬТИАГЕНТНЫЕ ИГРЫ

КРАТКОЕ РЕЗЮМЕ



OPENAI FIVE

- Есть на свете и не только настольные игры. Например, DotA 2 (Berner et al., 2019):



- OpenAI Five победила OG, чемпионов The International, в 2019

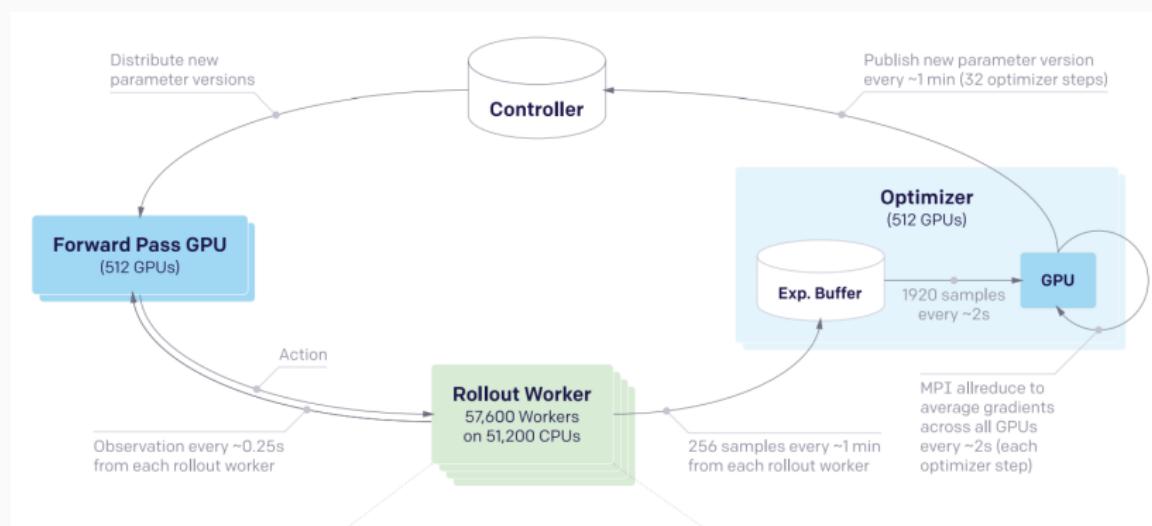
- DotA 2 отличается от го и шахмат:
 - длинные горизонты планирования (20000 шагов на эпизод вместо 50-200)
 - частично наблюдаемые состояния
 - очень много действий, пространство высокой размерности (16000 значений на входе, 8000-80000 действий)
- Как же они этого добились?

- Базовая структура не очень сложная:

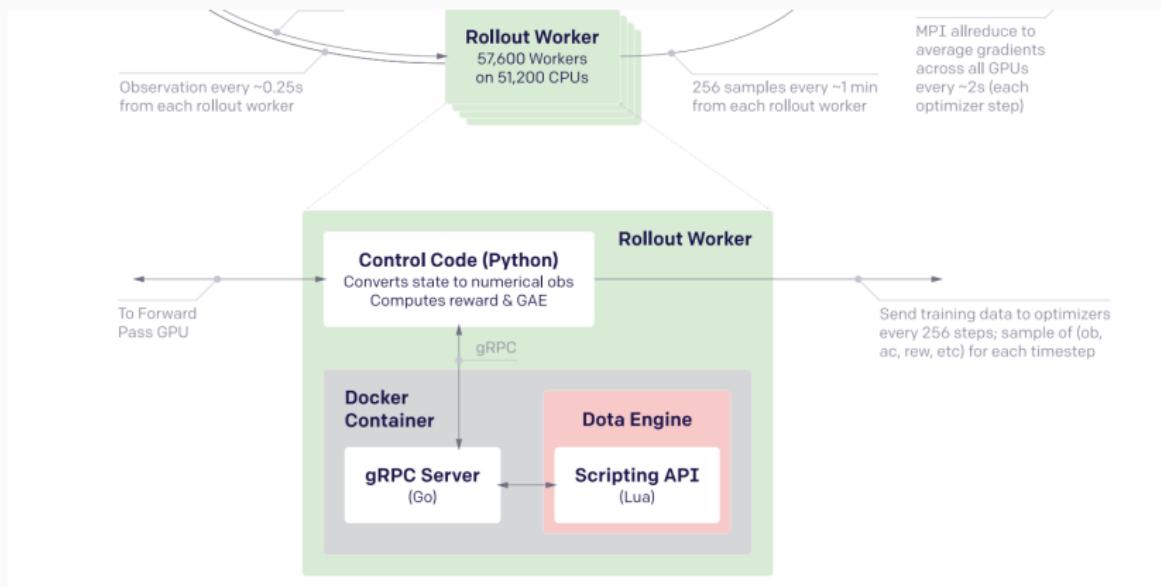


- Каждый из пяти героев управляет через LSTM, в котором 84% всех параметров модели.

- Обучение идёт через PPO (proximal policy optimization)
- Но интересно, как структура распределённого обучения организована:

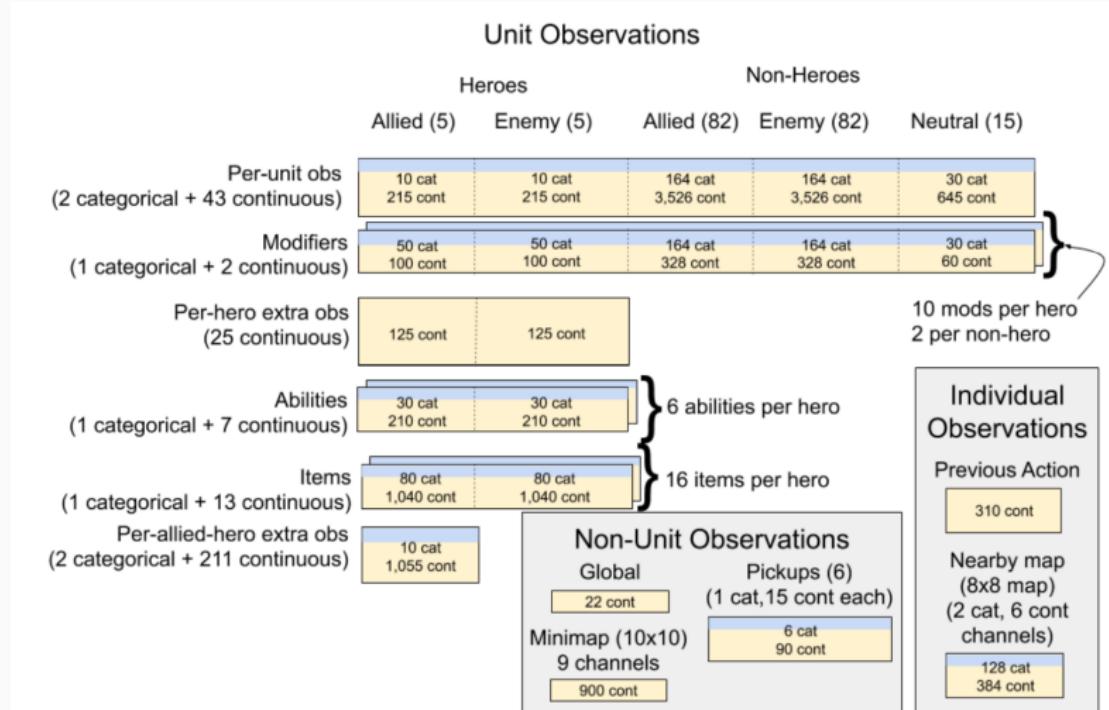


- ...продолжение картинки:

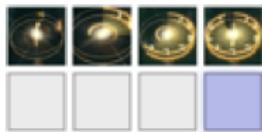


- И самое интересное – исследователи меняют архитектуру и структуру модели, но как не переобучать месяцами каждый раз?
- Surgery: как изменить старую обученную стратегию π так, чтобы она подходила к новому окружению и структуре.
- Net2Net transformations
- Это позволило обучать чуть ли не целый год, не выкидывая предыдущие результаты.

- А дальше просто масштабные представления
входов-выходов и много-много compute.



- А дальше просто масштабные представления входов-выходов и много-много compute.



(a) **Delay:** An integer from 0 to 3 indicating which frame during the next frameskip to take the action on (see Appendix L). If 0, the action will be taken immediately when the game engine processes this time step; if 3, the action will be taken on the last game frame before the next policy observation. This parameter is never ignored.



(b) **Unit Selection:** One of the 189 visible units in the observation. For actions and abilities which target units, either enemy units or friendly units. For many actions, some of the possible unit targets will be invalid; attempting an action with an invalid target results in a noop.



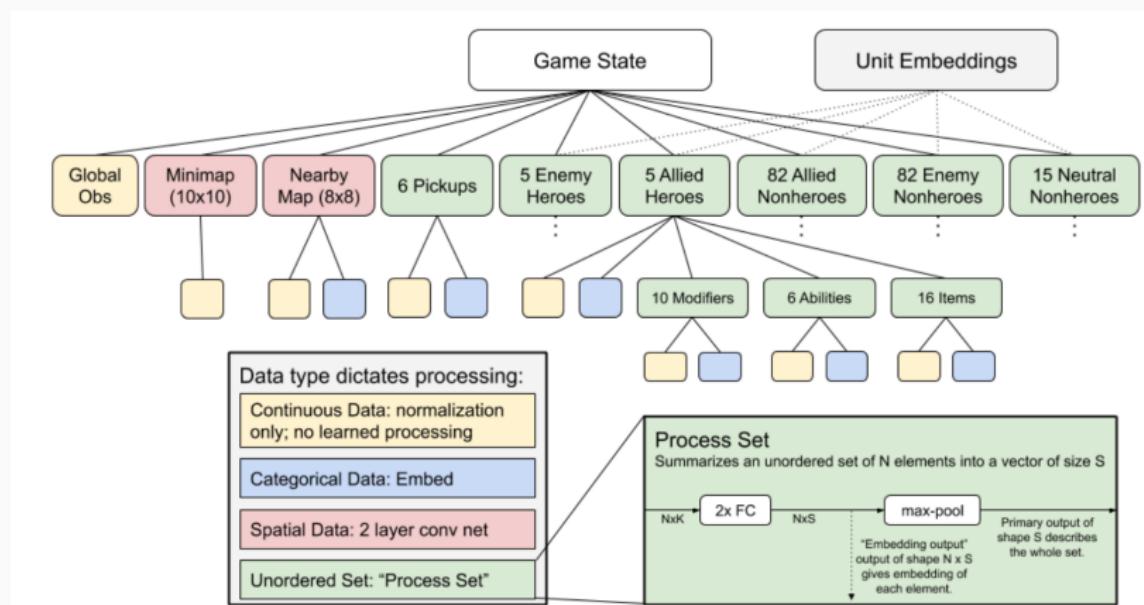
(c) **Offset:** A 2D (X, Y) coordinate indicating a spatial offset, used for abilities which target a location on the map. The offset is interpreted relative to the caster or the unit selected by the Unit Selection parameter, depending on the ability. Both X and Y are discrete integer outputs ranging from -4 to +4 inclusive, producing a grid of 81 possible coordinate pairs.

Figure 15: Action Parameters

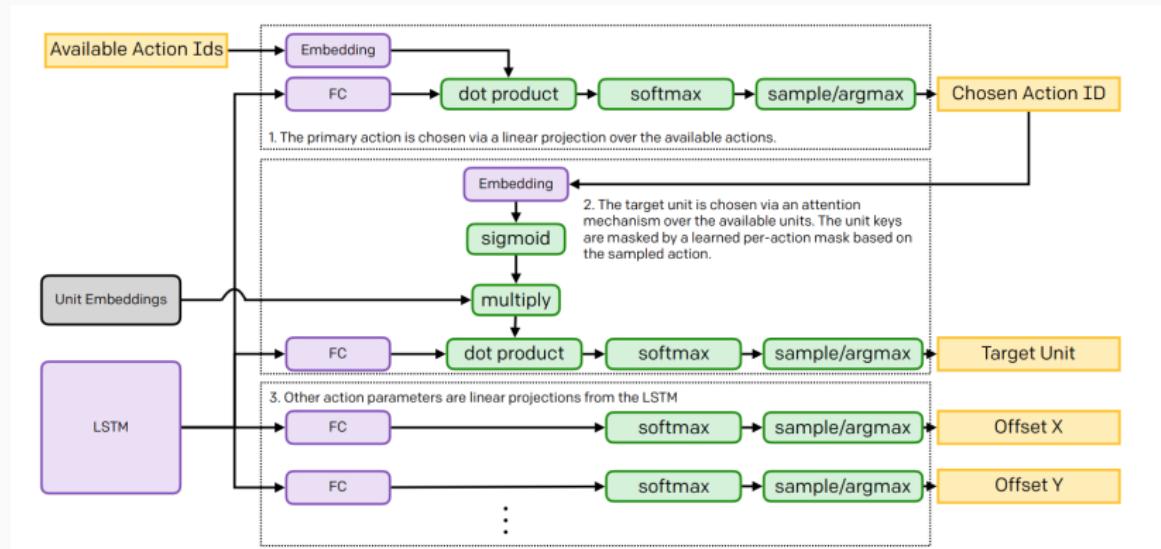
- Награды, кстати, дают и за промежуточные достижения, иначе обучение стопорится.

Name	Reward	Heroes	Description
Win	5	Team	
Hero Death	-1	Solo	
Courier Death	-2	Team	
XP Gained	0.002	Solo	
Gold Gained	0.006	Solo	For each unit of gold gained. Reward is not lost when the gold is spent or lost.
Gold Spent	0.0006	Solo	Per unit of gold spent on items without using courier.
Health Changed	2	Solo	Measured as a fraction of hero's max health. [‡]
Mana Changed	0.75	Solo	Measured as a fraction of hero's max mana.
Killed Hero	-0.6	Solo	For killing an enemy hero. The gold and experience reward is very high, so this reduces the total reward for killing enemies.
Last Hit	-0.16	Solo	The gold and experience reward is very high, so this reduces the total reward for last hit to ~ 0.4.
Deny	0.15	Solo	
Gained Aegis	5	Team	
Ancient HP Change	5	Team	Measured as a fraction of ancient's max health.
Megas Unlocked	4	Team	
T1 Tower*	2.25	Team	
T2 Tower*	3	Team	
T3 Tower*	4.5	Team	
T4 Tower*	2.25	Team	
Shrine*	2.25	Team	

- Наблюдения превращаются в плоский вектор и подаются в LSTM

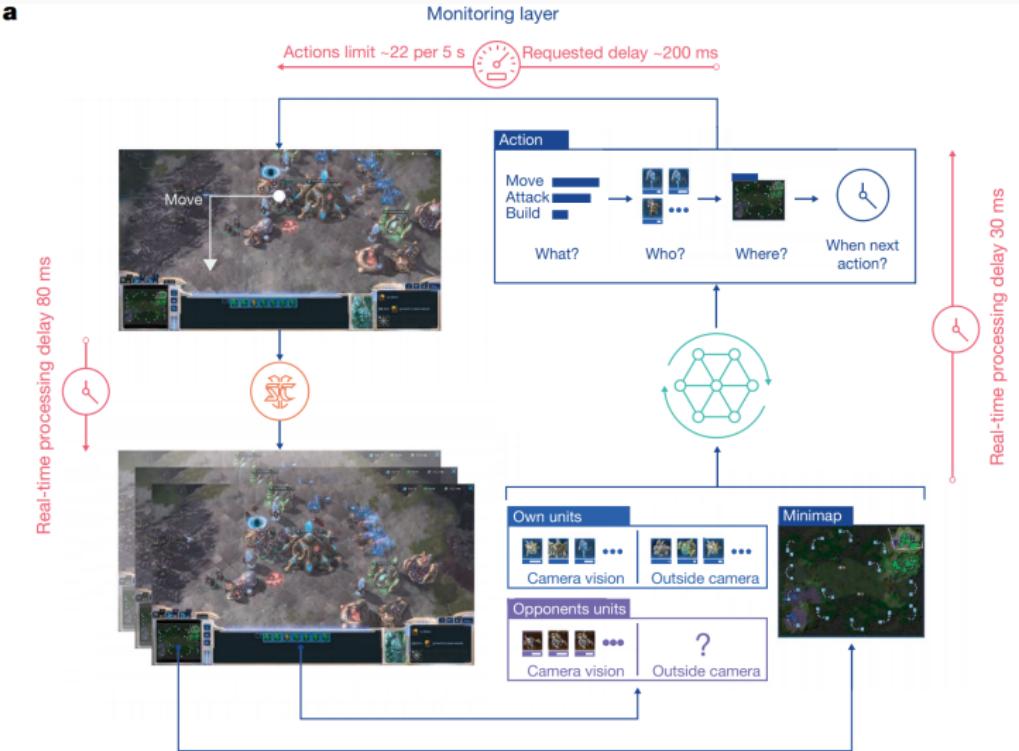


- Там архитектура нетривиальная, но вполне логичная:

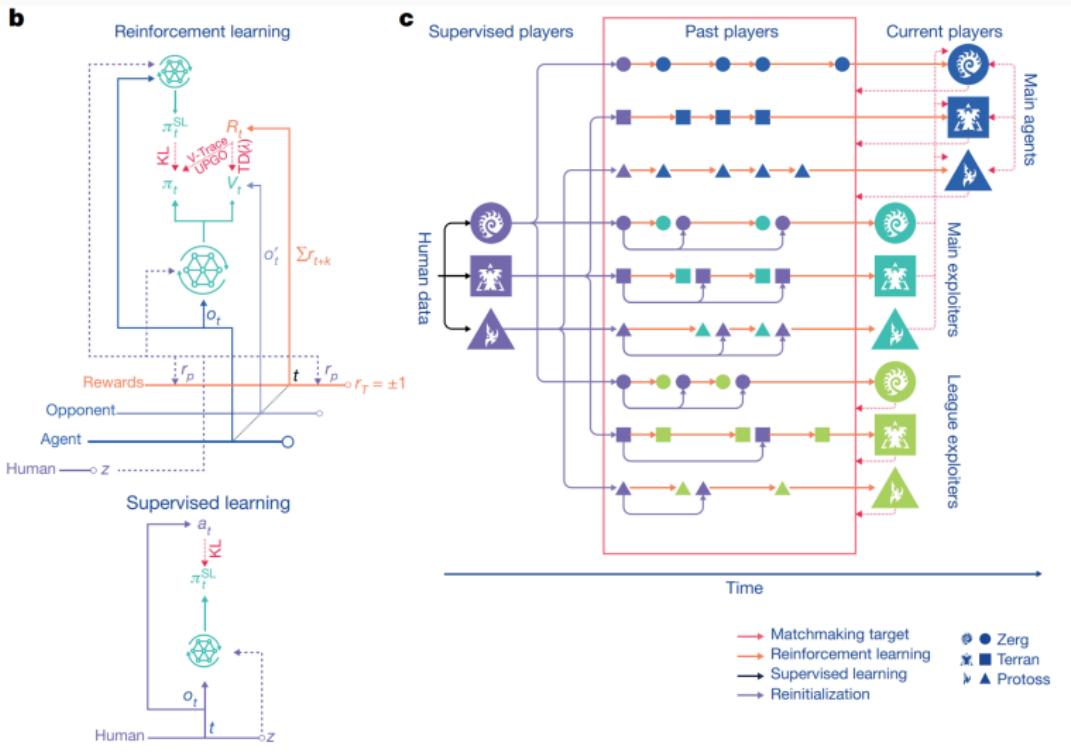


- И в целом всё, всё работает и побеждает!

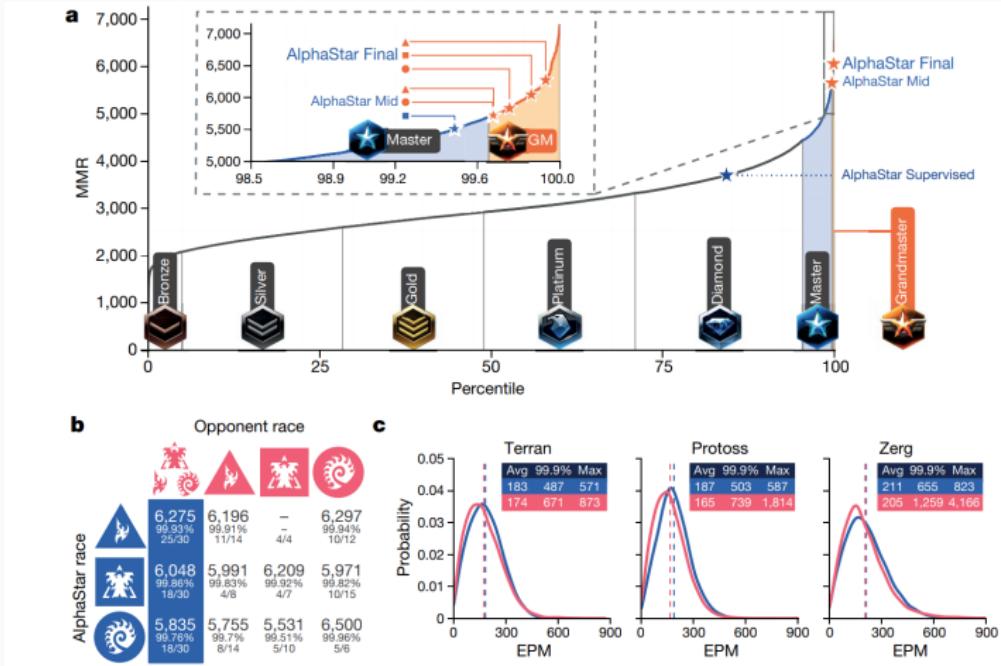
- А DeepMind в это время решал StarCraft:



- A DeepMind в это время решал StarCraft:

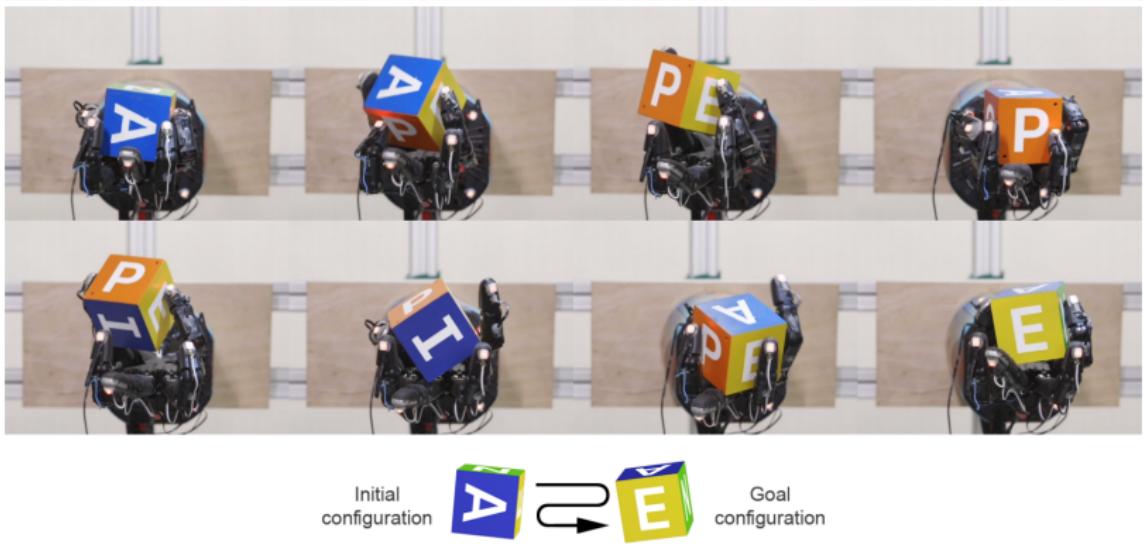


- A DeepMind в это время решал StarCraft:



Роботы: DACTYL

- Как заставить робота сложить кубик Рубика?
- OpenAI попробовал. Сначала добились более простых вещей (OpenAI et al., 2019):



Роботы: DACTYL

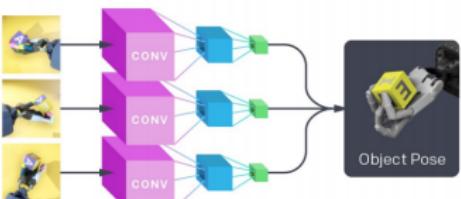
- Обучение на синтетических данных – важная тема в современном компьютерном зрении



B We train a control policy using reinforcement learning. It chooses the next action based on fingertip positions and the object pose.

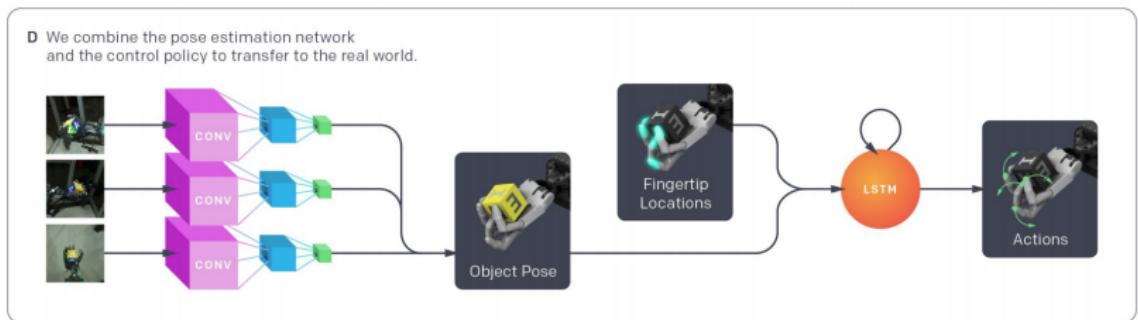


C We train a convolutional neural network to predict the object pose given three simulated camera images.



Роботы: DACTYL

- Обучение на синтетических данных – важная тема в современном компьютерном зрении



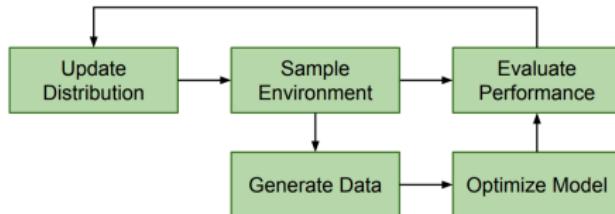
- А по сути опять PPO...

Роботы: DACTYL

- Domain randomization:



- Но хотелось кубик Рубика! Автоматический domain randomization (OpenAI et al., 2019)



- <https://openai.com/blog/solving-rubiks-cube>
- <https://www.youtube.com/watch?v=QyJGXc9WeNo>

- (Baker et al., 2019), OpenAI: очень интересная работа про командные цели и окружение
- Как создать «умного» агента, который сам придумывает, что лучше сделать, использует своё окружение и всё такое прочее?
- Явно задавать целевую функцию не всегда возможно, собирать данные для imitation learning тем более.
- Один подход – добавлять «любопытство» (вроде энтропии), но это пока не получается обобщить на сложные среды.

EMERGENT TOOL USE

- Идея: агенты соревнуются друг с другом в группах
- Hiders прячутся, seekers их пытаются поймать в поле зрения; если нашли, награждаем seekers, иначе hiders
- Правила игры:



The agents can **move** by setting a force on themselves in the x and y directions as well as rotate along the z-axis.



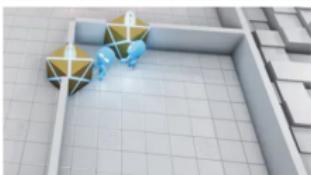
The agents can **see** objects in their line of sight and within a frontal cone.



The agents can **sense** distance to objects, walls, and other agents around them using a lidar-like sensor.



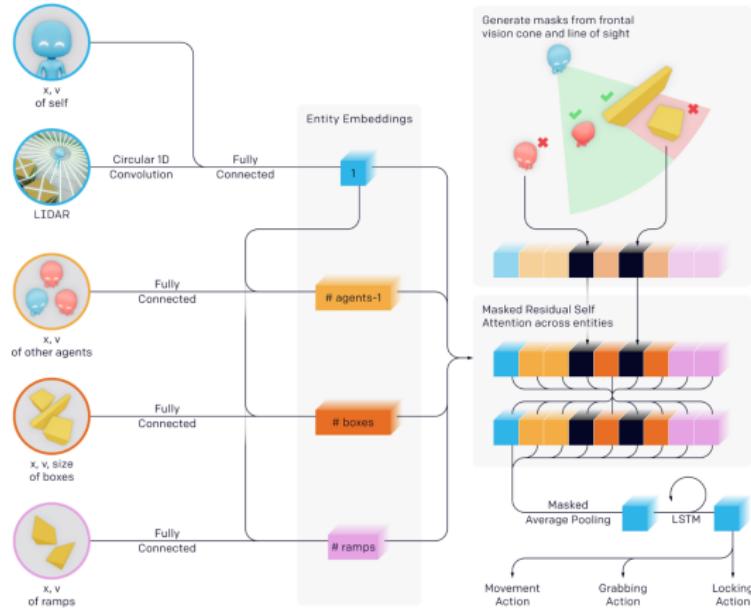
The agents can **grab and move** objects in front of them.



The agents can **lock** objects in place. Only the team that locked an object can unlock it.

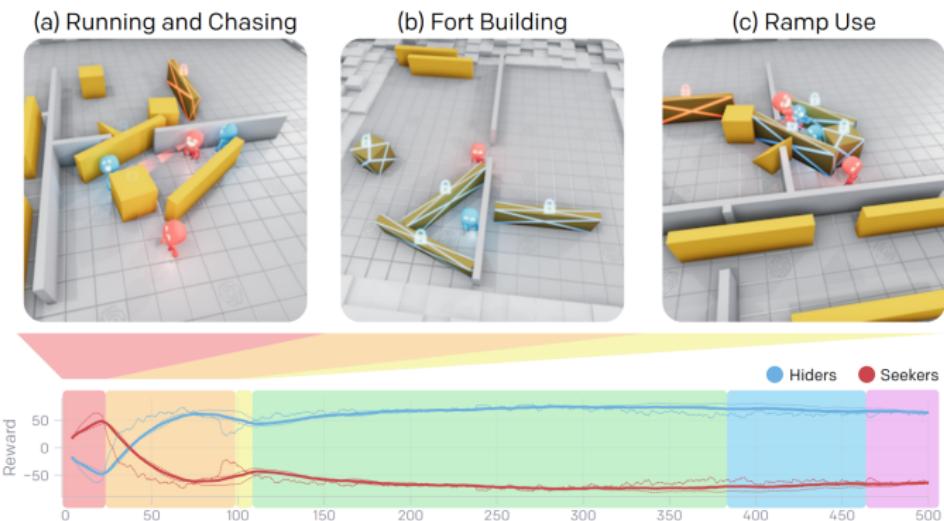
EMERGENT TOOL USE

- Сами агенты обычные: policy network (справа) обучается действиям, critic network предсказывает будущие награды
- Proximal policy optimization, generalized advantage estimation (по сути TD-обучение), self-attention на объекты и т.д.



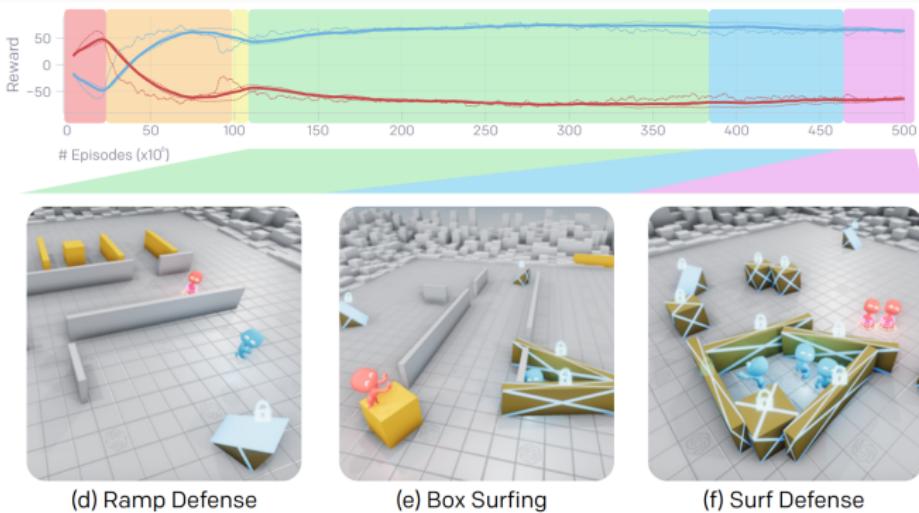
EMERGENT TOOL USE

- И оказывается, что агенты сами собой обучаются сразу нескольким фазам поведения:
 - Сначала просто учатся бегать друг за другом
 - Потом *hiders* начинают строить «форты», где их не достать
 - Потом *seekers* обучаются туда залезать через рампы



EMERGENT TOOL USE

- И оказывается, что агенты сами собой обучаются сразу нескольким фазам поведения:
 - От рамп можно защититься, заблокировав их
 - Но seekers затем обучаются «ездить» на кубиках (это баг окружения, ставший фичей)
 - И, наконец, hiders учатся блокировать все объекты (если успеют)



EMERGENT TOOL USE

- Почему важно, что команды соперничают? Они поочерёдно создают новые задачи друг для друга. Примерно как игра с самим собой в шахматы у AlphaZero, только гораздо сложнее.
- Это называется autocurriculum (Leibo et al., 2019); очень интересная статья с «манифестом» развития multi-agent intelligence

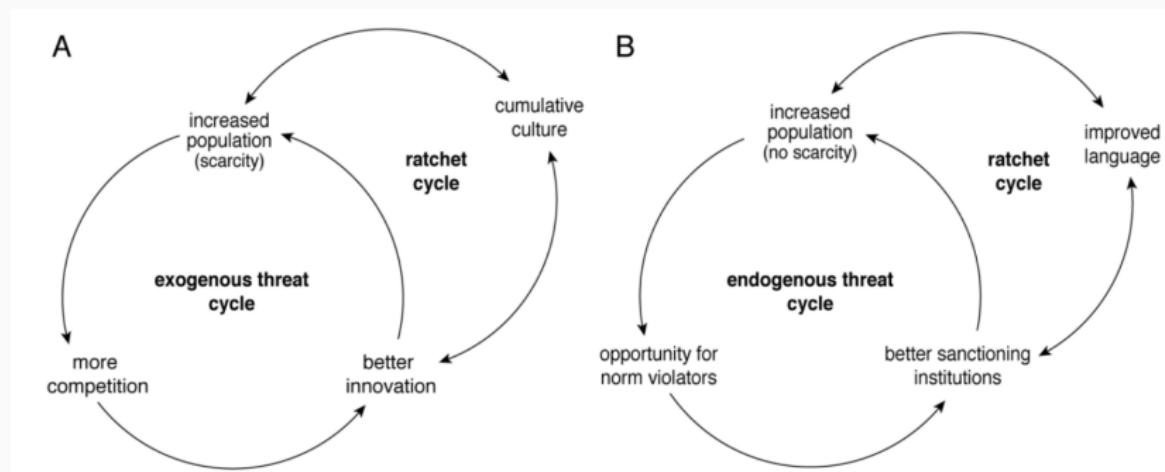


Figure 2 | (A) The cumulative culture loop. (B) The self-domestication loop.

WORLD MODELS

WORLD MODELS

- World models (Ha, Schmidhuber): обучаем сжатое представление окружающей среды, чтобы потом планировать там

At each time step, our agent receives an **observation** from the environment.

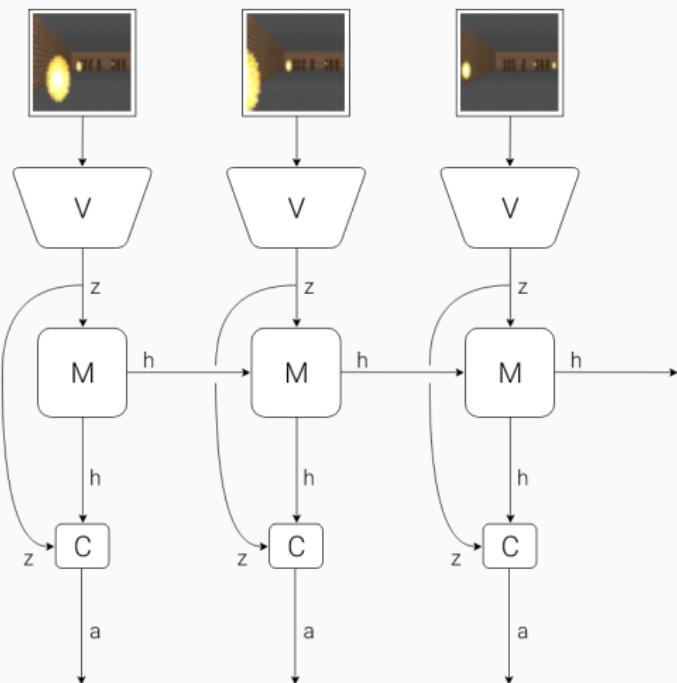
World Model

The **Vision Model (V)** encodes the high-dimensional observation into a low-dimensional latent vector.

The **Memory RNN (M)** integrates the historical codes to create a representation that can predict future states.

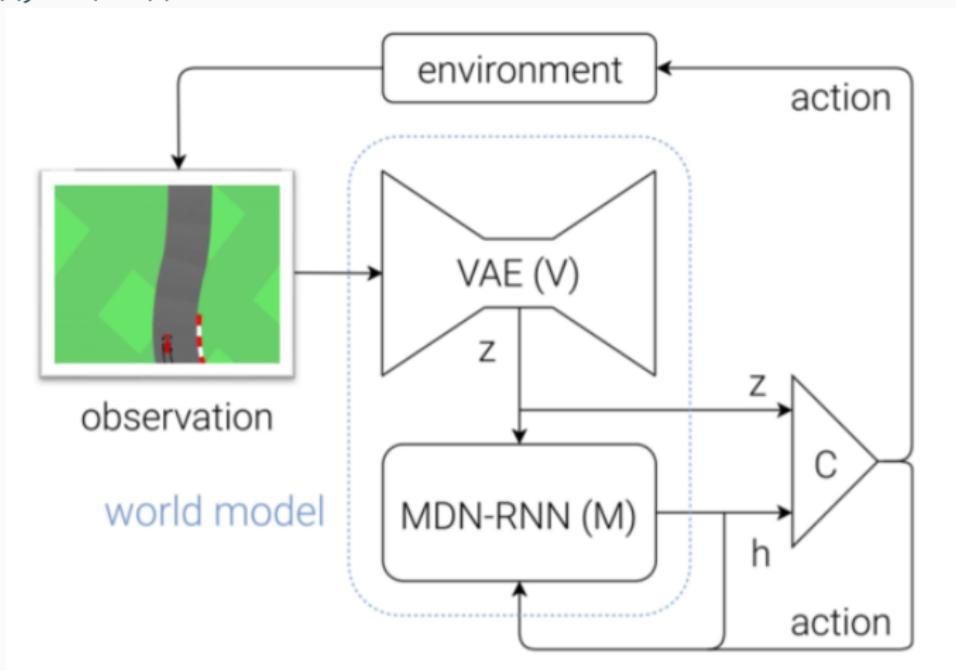
A small **Controller (C)** uses the representations from both **V** and **M** to select good actions.

The agent performs **actions** that go back and affect the environment.



WORLD MODELS

- Наблюдение кодируется через VAE (Vision), результат идёт в MDN-RNN (Memory), которая предсказывает следующее состояние, а контроллер (C) просто отображает это в следующее действие:



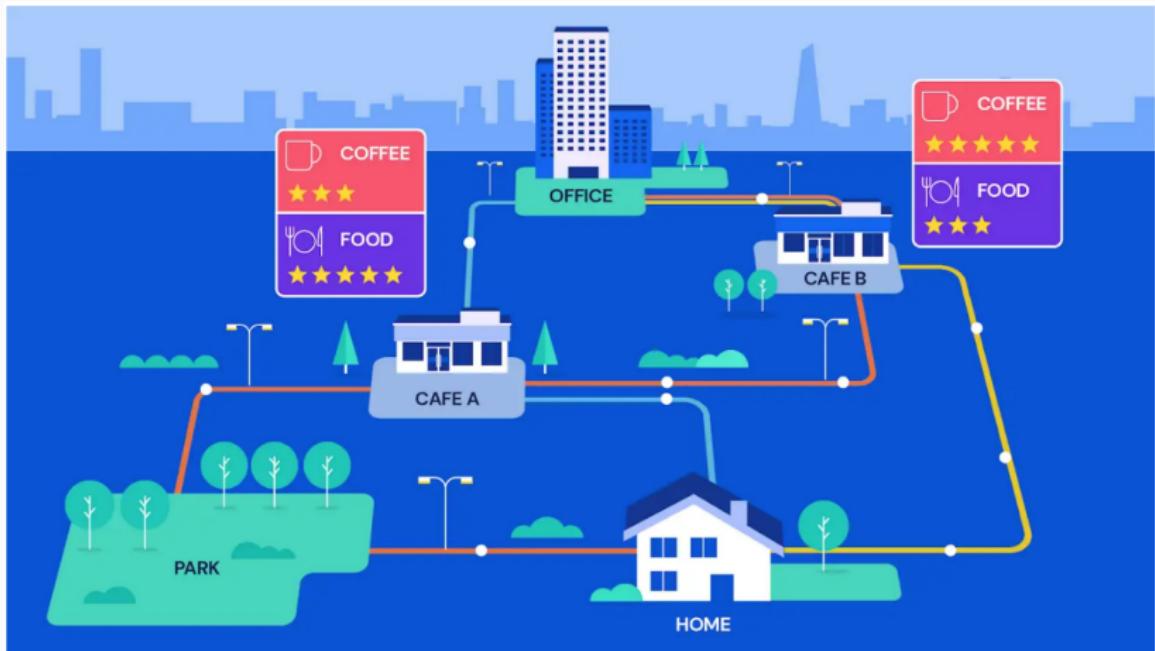
SUCCESSOR FEATURES И КОМПОЗИЦИОНАЛЬНОСТЬ В RL

Композиционность в RL

- Когда мы чему-то обучаемся, мы приносим в это обучение все те умения, которыми уже обладали раньше
- Это часто позволяет быстро получить новое умение, рассмотрев его как композицию предыдущих с некоторыми новыми компонентами
- Представьте, что для этого курса вам было бы нужно заново учиться складывать и брать производные, а для каждой компьютерной игры нужно было бы заново учиться управлять геймпадом.
- Но RL-агенты именно в такой ситуации и находятся! Может быть, неудивительно, что они так медленно обучаются?..

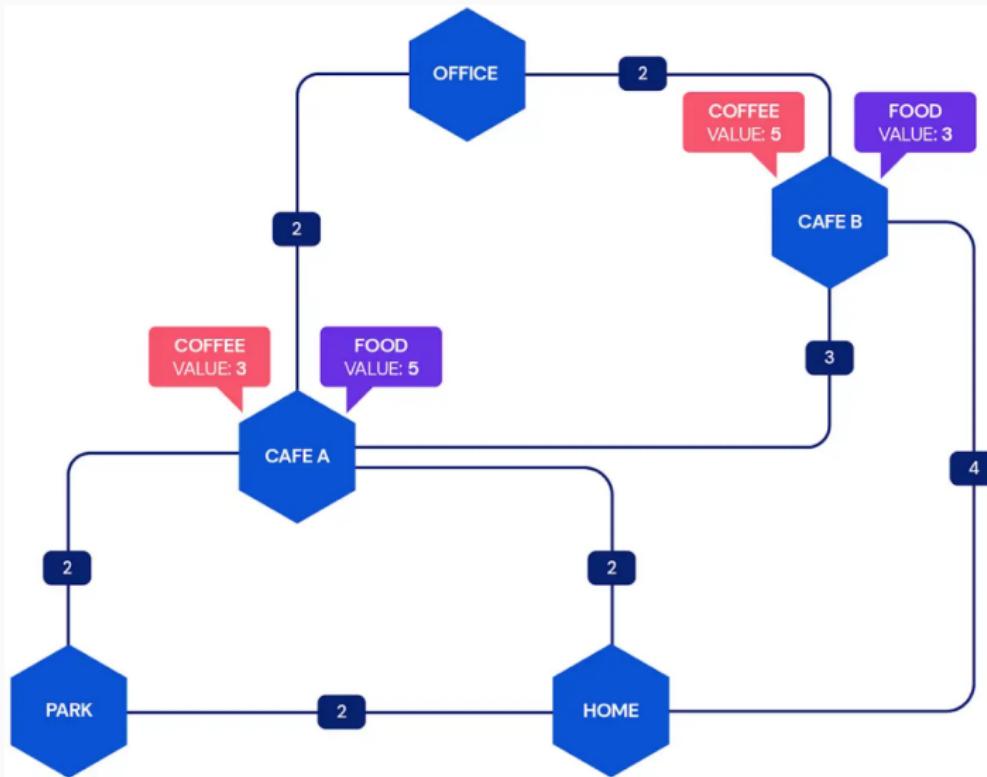
Композиционность в RL

- Мы пока что разделяли RL на model-based и model-free
- Пример из поста DeepMind (октябрь 2020):



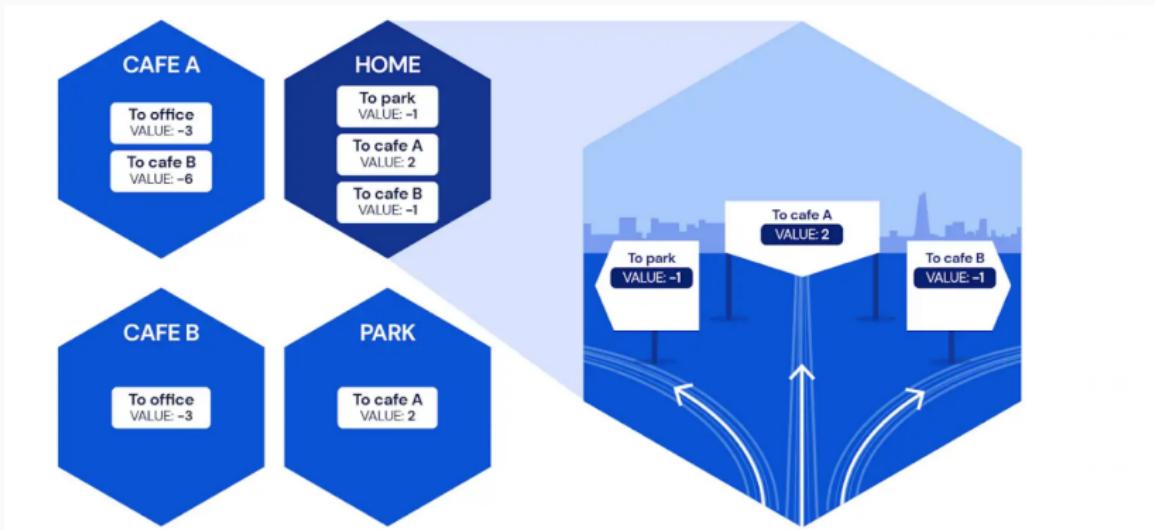
Композиционность в RL

- Model-based агент построит модель окружения:



Композиционность в RL

- А model-free агент просто будет знать, куда ехать:

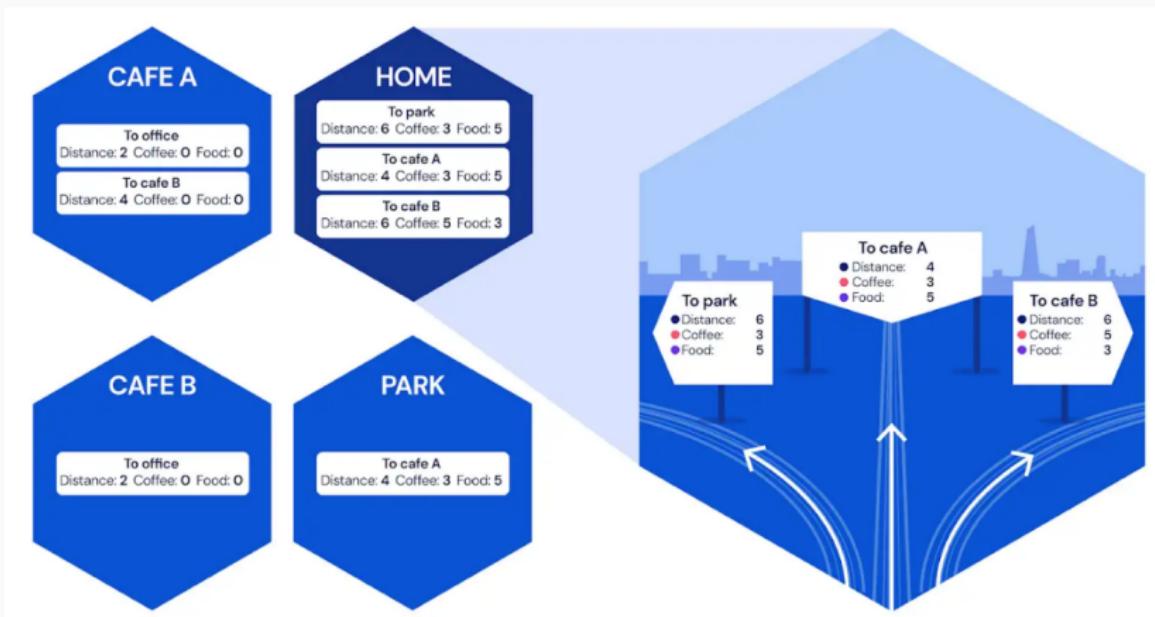


Композициональность в RL

- Получается, что:
 - model-free агент не обобщается на другой набор предпочтений;
 - a model-based агент легко обобщается, но долго учится, и, может быть, делает много лишнего.
- Successor features – это попытка найти золотую середину (Barreto et al., 2017; Borsa et al., 2018; Barreto et al., 2019; Hansen et al., 2020).

Композициональность в RL

- Базовая идея – давайте запишем в признаки «качество кофе» и «качество еды»:



Композициональность в RL

- Формально у нас теперь reward представляется в виде модели с признаками:

$$r(s, a, s') = \phi(s, a, a')^\top \mathbf{w},$$

и это протягивается в уравнения Беллмана:

$$\begin{aligned} Q_\pi(s, a) &= \mathbb{E}_\pi [\phi_{t+1}^\top \mathbf{w} + \gamma \phi_{t+2}^\top \mathbf{w} + \dots \mid S_t = s, A_t = a] = \\ &= \mathbb{E}_\pi \left[\mathbf{w}^\top \sum_{i=t}^{\infty} \gamma^{i-t} \phi_{i+1} \mid S_t = s, A_t = a \right] \end{aligned}$$

- Successor features – это

$$\psi_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{i=t}^{\infty} \gamma^{i-t} \phi_{i+1} \mid S_t = s, A_t = a \right],$$

т.е. то, что нужно умножить на \mathbf{w} , чтобы получить $Q_\pi(s, a)$.

Композиционность в RL

- Теперь мы можем получать новые процессы принятия решений, меняя rewards, т.е. меняя веса \mathbf{w} .
- Generalized policy evaluation (GPE): можно оценить качество стратегий в зависимости от \mathbf{w} , для разных предпочтений.
- Generalized policy improvement (GPI): можно обобщить PI на несколько стратегий, т.е. если мы возьмём

$$\pi(s) = \arg \max_a \max_i Q_{\pi_i}(s, a),$$

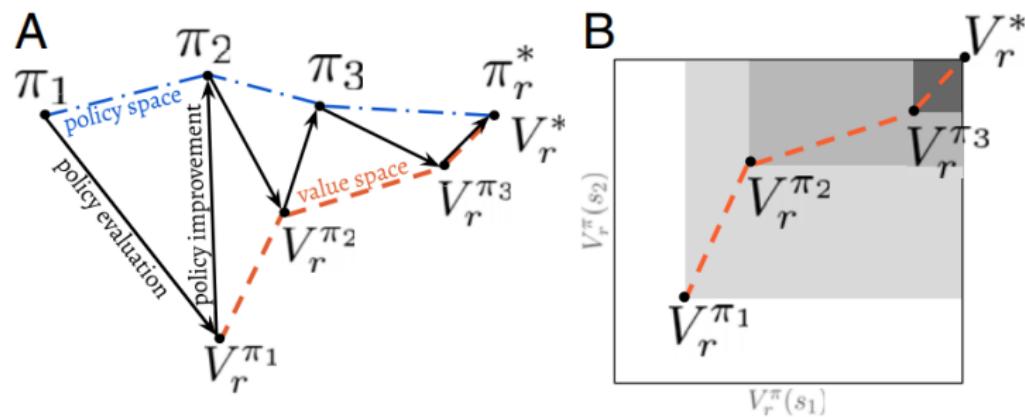
то результат будет лучше любой из π_i :

$$Q_\pi(s, a) \geq \max_i Q_{\pi_i}(s, a),$$

и это можно обобщить на приближённые оценки функций Q и на successor features. $Q'_*(s, a) = \psi^*(s, a)^\top \mathbf{w}'$.

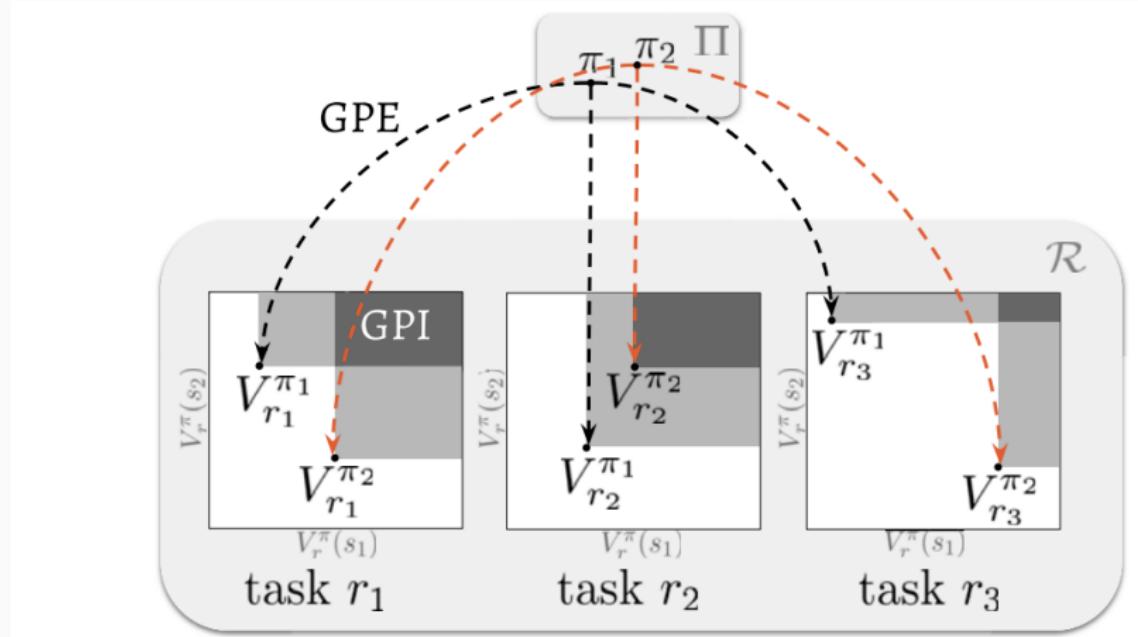
Композиционность в RL

- Главная мысль в том, что если мы знаем $\psi^*(s, a)$ для оптимальной стратегии, а потом предпочтения поменялись на w' , можно получить новую функцию Q практически бесплатно
- Т.е. вот так выглядит обычный policy improvement:



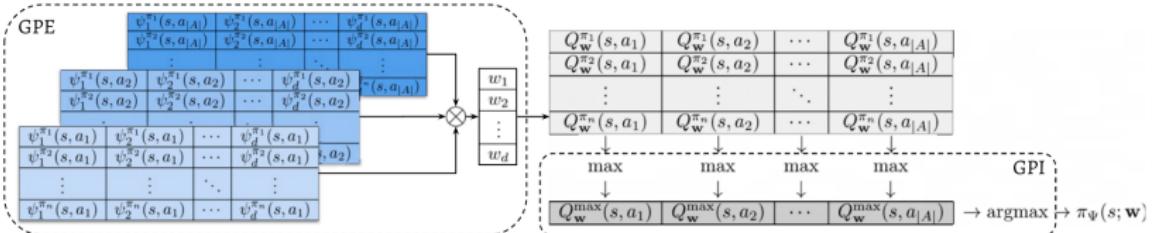
Композиционность в RL

- Так выглядит обобщённый; тут две стратегии и три задачи, и GPI переводит нас в тёмные прямоугольники:



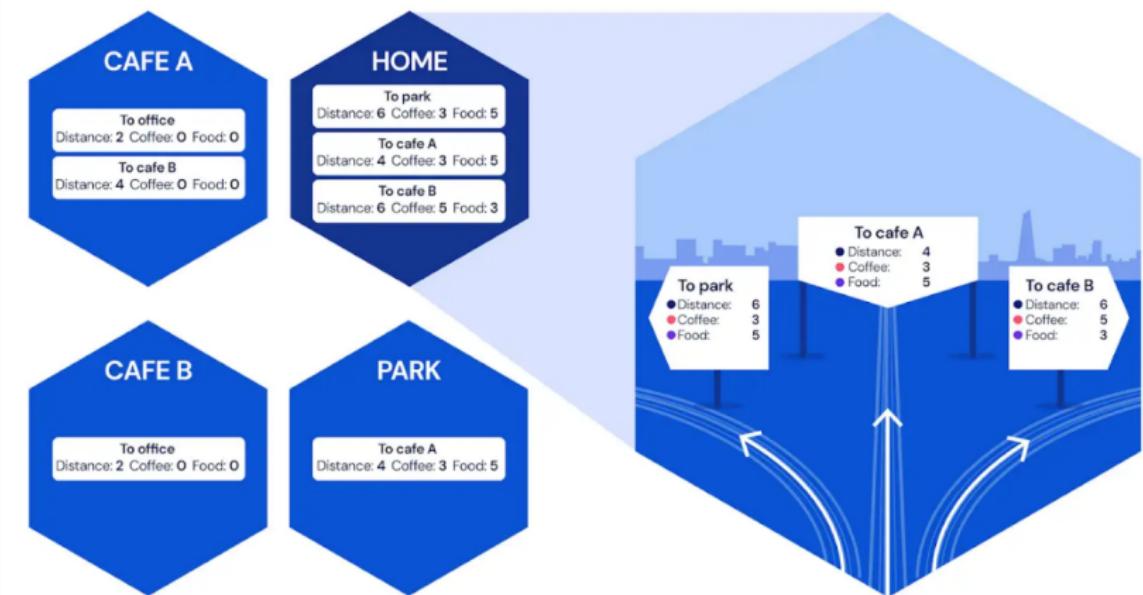
Композициональность в RL

- И теперь можно реализовать обобщённую стратегию π_ψ :
 - вычисляем $\psi_{\pi_i}(s, a)$ для каждого a ,
 - умножаем на \mathbf{w} , получая тензор из $Q_{\pi_i}(s, a | \mathbf{w})$,
 - а потом выбираем из всего этого максимум:



Композициональность в RL

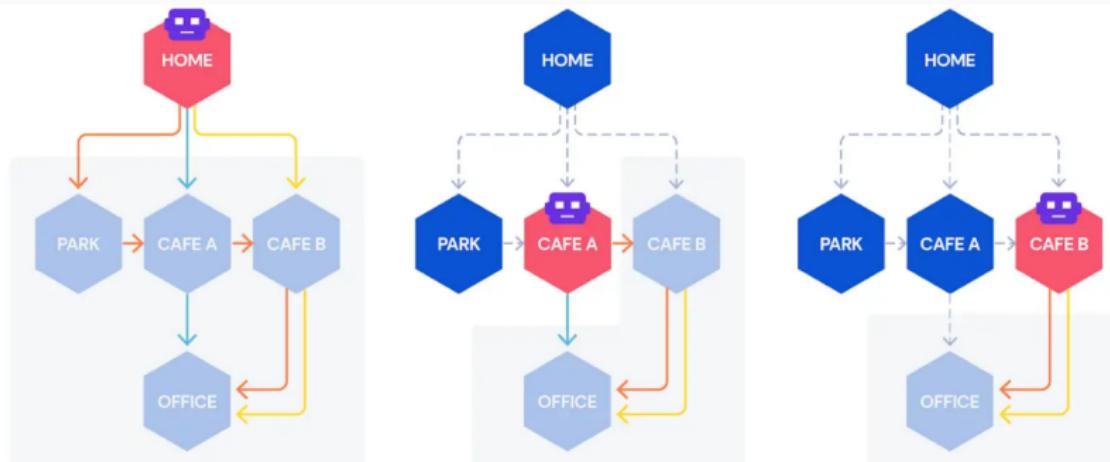
- В примере это значит, что агент будет понимать, где какие признаки, и оценивать, нужнее ли ему кофе или еда:



- Но это не всё...

Композиционность в RL

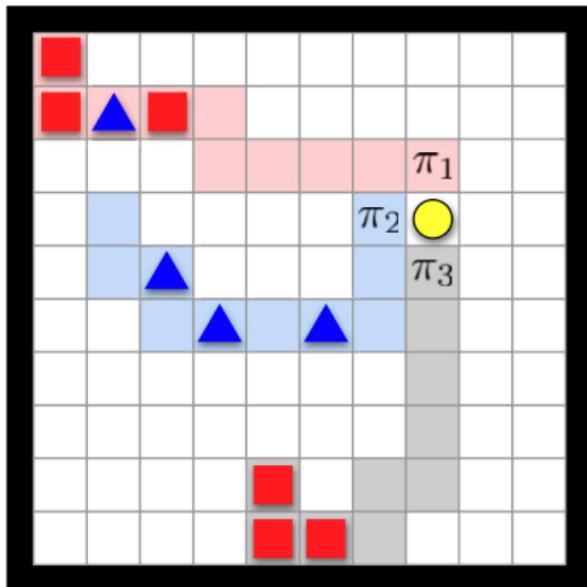
- Более того, агент сможет совсем новые стратегии порождать! Просто жадно следуем Q_* для новых предпочтений.
- Пусть в том же примере агенту пройденное расстояние на 50% важнее, чем качество кофе и еды; такого раньше не было, и наши три стратегии так не умеют. Однако:



Agent

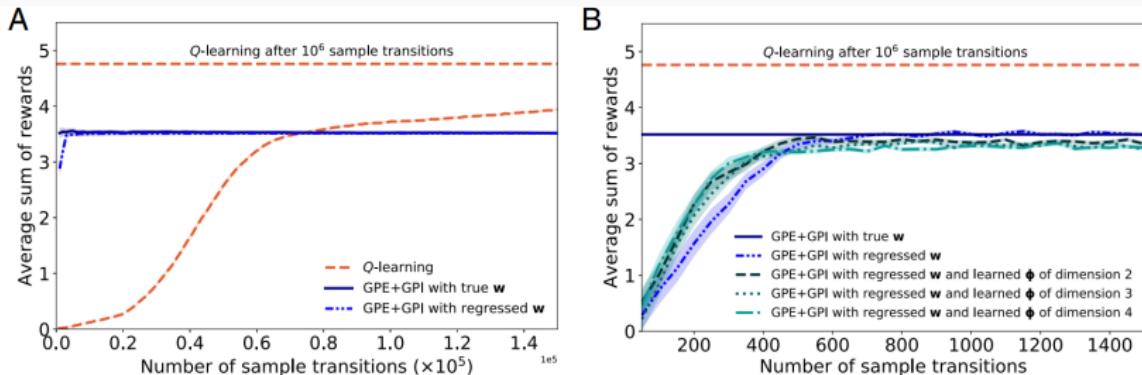
Композициональность в RL

- Пример: пусть агент обучается ходить по окружению, причём разные типы объектов могут стоить по-разному: π_1 для $w_1 = (1 \ 0)$, π_2 для $w_2 = (0 \ 1)$, π_3 для $w_3 = (1 \ -1)$:



Композиционность в RL

- Эксперимент: обучим ψ_{π_1} и ψ_{π_2} , а потом попробуем перенести на \mathbf{w}_3 :



- Как видите, мы почти мгновенно получаем неплохой результат! Там ещё можно и нужно \mathbf{w} обучать, это отдельный вопрос, но тоже можно.
- Это, кстати, вполне соответствует человеческому обучению тоже, но это уже совсем другая история...

Спасибо!

Спасибо за внимание!

