

# RL: Partially Observable Markov Decision Process

# План

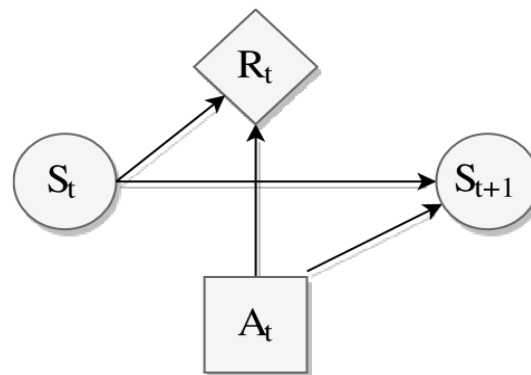
- ❑ Что не так с MDP
  - MDP процесс
- ❑ Детали POMDP
  - Байесовская фильтрация
- ❑ Приближенное обучение
  - Глубокое рекуррентное Q-обучение
  - Глубокий рекуррентный градиент политики
- ❑ Явная память
  - Нейронная карта

# MDP

## Definition of Markov Decision Process

MDP is a tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ , where

- 1  $\mathcal{S}$  – set of states of the world
- 2  $\mathcal{A}$  – set of actions
- 3  $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \Delta(\mathcal{S})$  – state-transition function, giving us  $p(s_{t+1} | s_t, a_t)$
- 4  $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$  – reward function, giving us  $\mathbb{E}_R [R(s_t, a_t) | s_t, a_t]$ .



## Markov property

$$p(r_t, s_{t+1} | s_0, a_0, r_0, \dots, s_t, a_t) = p(r_t, s_{t+1} | s_t, a_t)$$

(next state, expected reward) depend on (previous state, action)

## Беспилотный автомобиль

- Имеет точные датчики зрения
- Имеет точную карту местности
- Имеет совершенную механику
- Должен безаварийно и оптимально перемещаться



## Беспилотный автомобиль

- Имеет точные датчики зрения
- Имеет точную карту местности
- Имеет совершенную механику
- Должен безаварийно и оптимально перемещаться

Как можно смоделировать автомобиль с помощью mdp?



# Причина неопределенности

Обычно состояние автономного агента состоит из

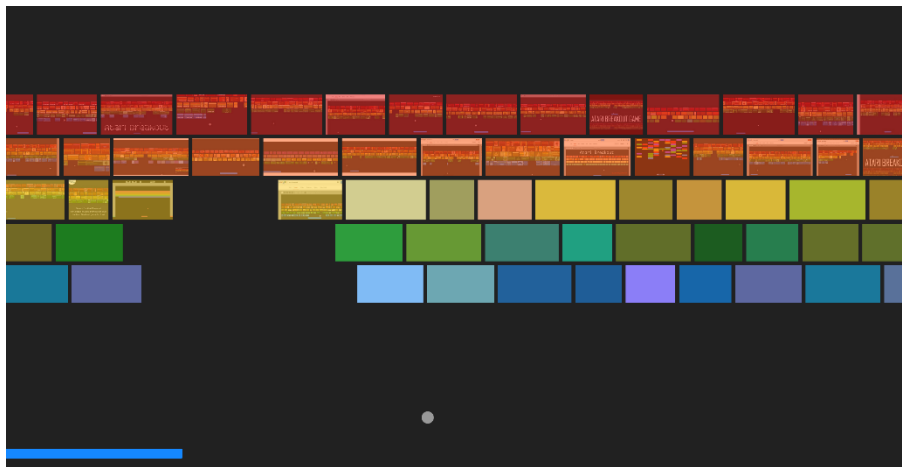
- ☐ измерений окружающей среды
- ☐ и самого агента

☐ В реальной системе существует еще больше неопределенности:

- 1 несовершенное самоощущение (положение, крутящий момент, скорость и т.д.)
- 2 несовершенное восприятие окружающей среды
- 3 неполное наблюдение за (нестационарной?) средой

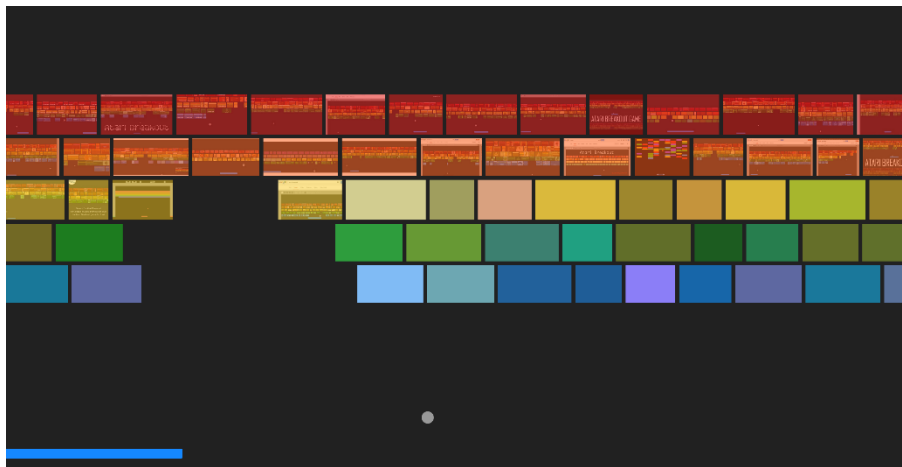
Как включить неопределенность в процесс принятия решений?

# Проблемы MDP



В каком состоянии игры мы находимся?

# Проблемы MDP



В каком состоянии игры мы находимся?

128 байт ненаблюдаемой оперативной памяти симулятора Atari



# POMDP – мощная математическая абстракция

## ☐ Промышленные приложения

- Техническое обслуживание машин (Shani et al., 2009)
- Беспроводные сети (Pajarinen et al., 2013)
- Управление ветряными электростанциями (Memarzadeh et al., 2014)
- Избежание столкновения самолетов (Bai et al., 2012)
- Выбор продавцов на электронных торговых площадках (Irissappane et al., 2016)

## ☐ Вспомогательное лечение

## ☐ Робототехника

## ☐ Системы разговорного диалога

## Детали POMDP: Байесовская фильтрация

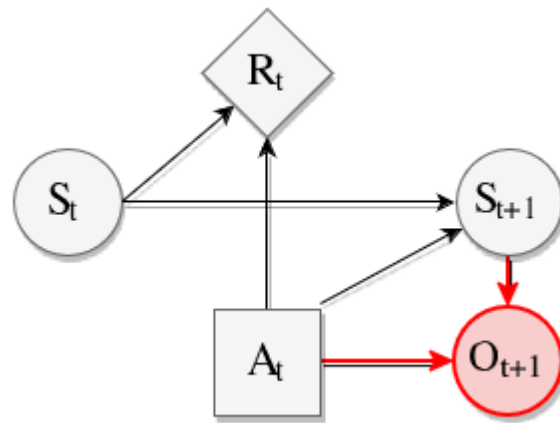
## Место POMDP в модели мира

Markov Models		Do we have control over the state transitions?	
		NO	YES
Are the states completely observable?	YES	<b>Markov Chain</b>	<b>MDP</b> Markov Decision Process
	NO	<b>HMM</b> Hidden Markov Model	<b>POMDP</b> Partially Observable Markov Decision Process

# Модель POMDP

## ❑ Определение

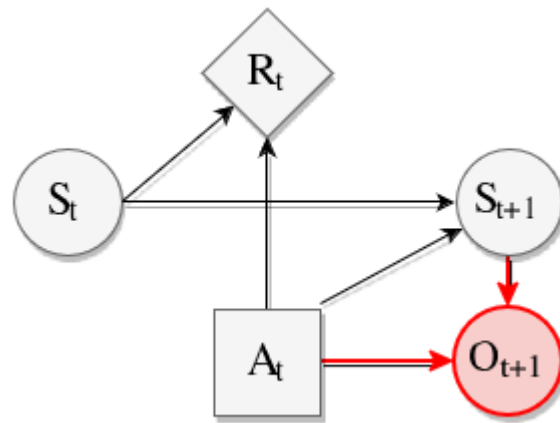
- POMDP – это кортеж  $\langle S, A, P, R, \Omega, O \rangle$ 
  1.  $S, A, P, R$  – такие же как в MDP
  2.  $\Omega$  – конечное множество наблюдений
  3.  $O: S \times A \rightarrow \Delta(\Omega)$  функция наблюдения, которая дает для каждого состояния и действия распределение вероятности по  $\Omega$ , т.е.  $p(o | s_{t+1}, a_t) \quad \forall o \in \Omega$



# Модель POMDP

## ❑ Определение

- POMDP – это кортеж  $\langle S, A, P, R, \Omega, O \rangle$ 
  1.  $S, A, P, R$  – такие же как в MDP
  2.  $\Omega$  – конечное множество наблюдений
  3.  $O: S \times A \rightarrow \Delta(\Omega)$  функция наблюдения, которая дает для каждого состояния и действия распределение вероятности по  $\Omega$ , т.е.  $p(o | s_{t+1}, a_t) \quad \forall o \in \Omega$









НО КАК ПОНЯТЬ, В КАКОМ СОСТОЯНИИ МЫ СЕЙЧАС НАХОДИМСЯ?

# Рассуждения о неопределенности состояния

Состояние убежденности (Belief state)

Распределение по пространству состояний,  $P_{s \in S} b(s) = 1$ ,  $0 \leq b(s) \leq 1$

$$A = \{left, right\} \quad p(\bar{A} | do(A)) = 0.1$$

	$b(s_1)$	$b(s_2)$	$b(s_3)$	$b(s_4)$	
	0.333	0.333		0.333	
	0.1	0.450		0.450	
	0.1	0.164		0.736	

# Рассуждения о неопределенности состояния

1. *Состояние убеждений является достаточной статистикой:  
содержит всю информация, необходимая для принятия решения  
(Striebel, 1965)*
2. POMDP - это MDP над правильно обновленными состояниями убеждений  
(Astrom, 1965)

## Обновление убеждений (фильтр Байеса)

**Хорошие новости:** обновление убеждений довольно простое (правило Байеса)

$$b'(s') = p(s'|o', a, b) = \frac{p(o'|s', a)p(s'|a, b)}{\sum_o p(o|s', a)p(s'|a, b)}$$

$$\propto p(o'|s', a)p(s'|a, b) \propto p(o'|s', a) \sum_s p(s'|a, b, s)p(s|a, b)$$

$$\propto p(o'|s', a) \sum_s p(s'|a, s)b(s)$$



# Обновление убеждений (фильтр Байеса)

**Плохая новость:** обновление убеждений может быть вычислено точно только для

1. дискретных низкоразмерных пространства состояний
2. линейно-гауссовская динамика (приводящая к фильтру Калмана),  
т.е.

$$s' \sim N(s' | T_s s + T_a a, \Sigma_s), \quad o' \sim N(o' | O_s s', \Sigma_o)$$

$$R(s, a) = s^T R_s s + a^T R_a a$$

# Таксономия задач POMDP

1. Задача обучения / планирования
2. Конечный / бесконечный горизонт
3. Онлайн / офлайн подход
4. Приблизительный / точный алгоритм
5. Дискретные / непрерывные состояния
6. Дискретное / непрерывное действие
7. Дискретное / непрерывное время
8. Стационарная / нестационарная среда
9. Один / много агентов

# План

- Приближенное обучение
  - Глубокое рекуррентное Q-обучение
  - Глубокий рекуррентный градиент политики

# Глубокое рекуррентное Q-обучение

**Проблема:** мы не можем оценить  $Q(s_t, a_t)$ , т.к. мы не знаем  $s_t$

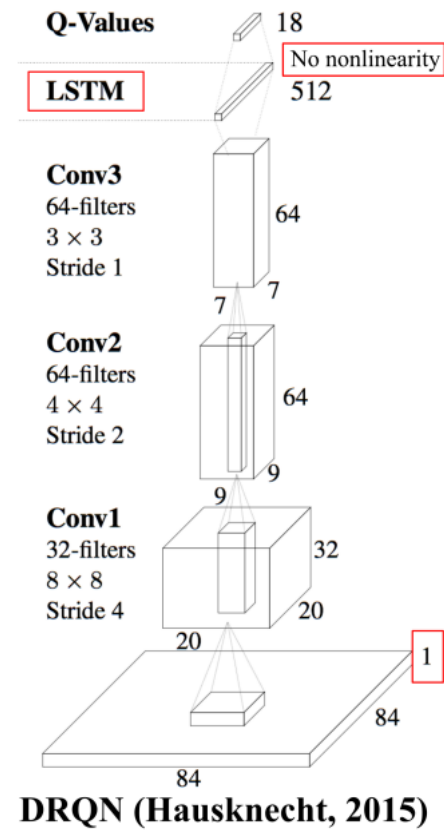
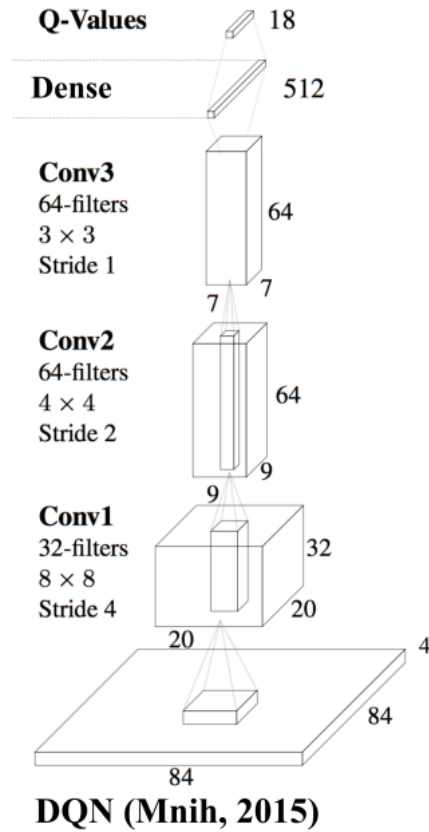
DRQN решение: (Hausknecht et al., 2015)

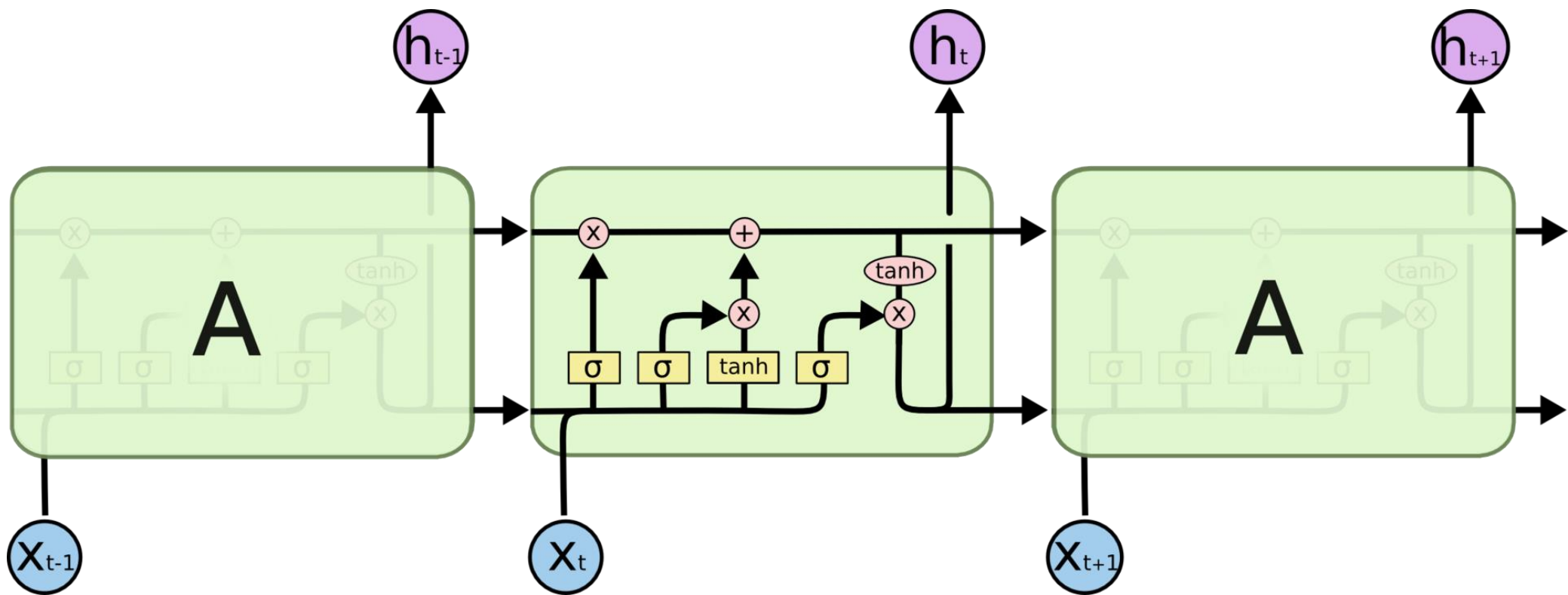
1. Снабдить агента памятью  $h_t$
2. Аппроксимировать  $Q(s_t, a_t)$  на основе  $Q(o_t, h_{t-1}, a_t)$
3. Устранить зависимость от  $o_t$  путем моделирования  $h_t = LSTM(o_t, h_{t-1})$

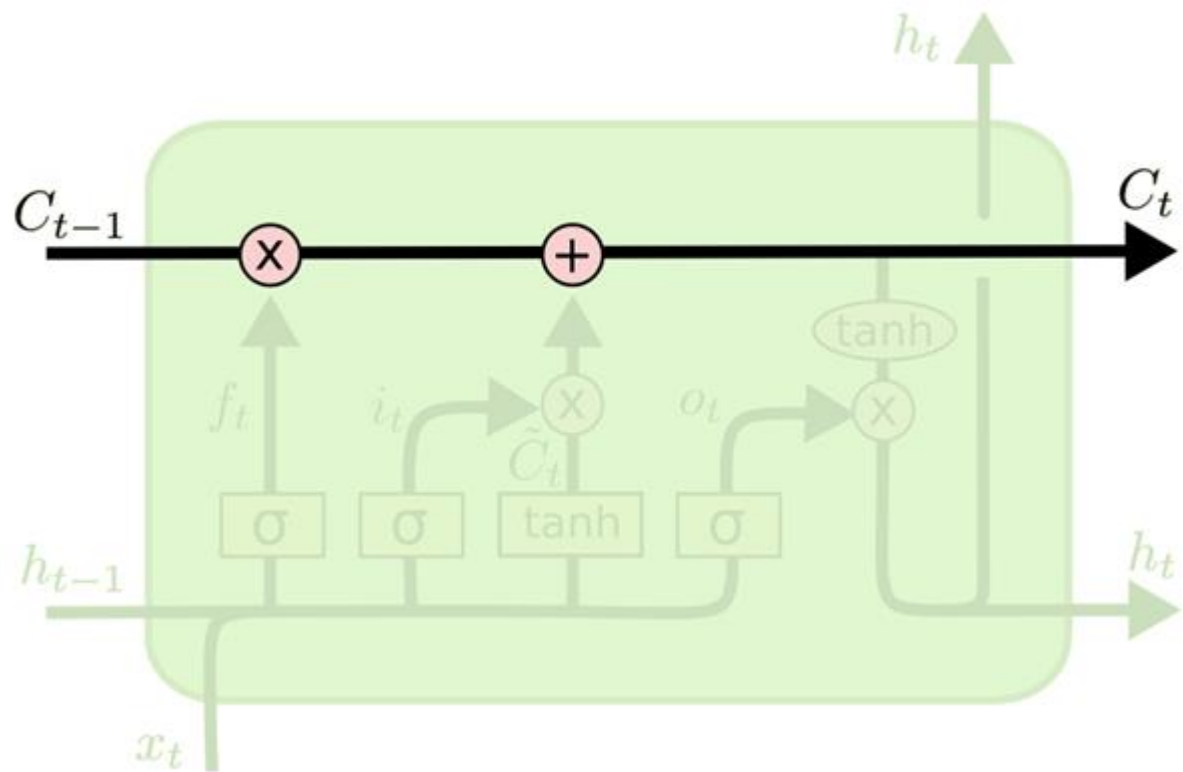
**Преимущества:**

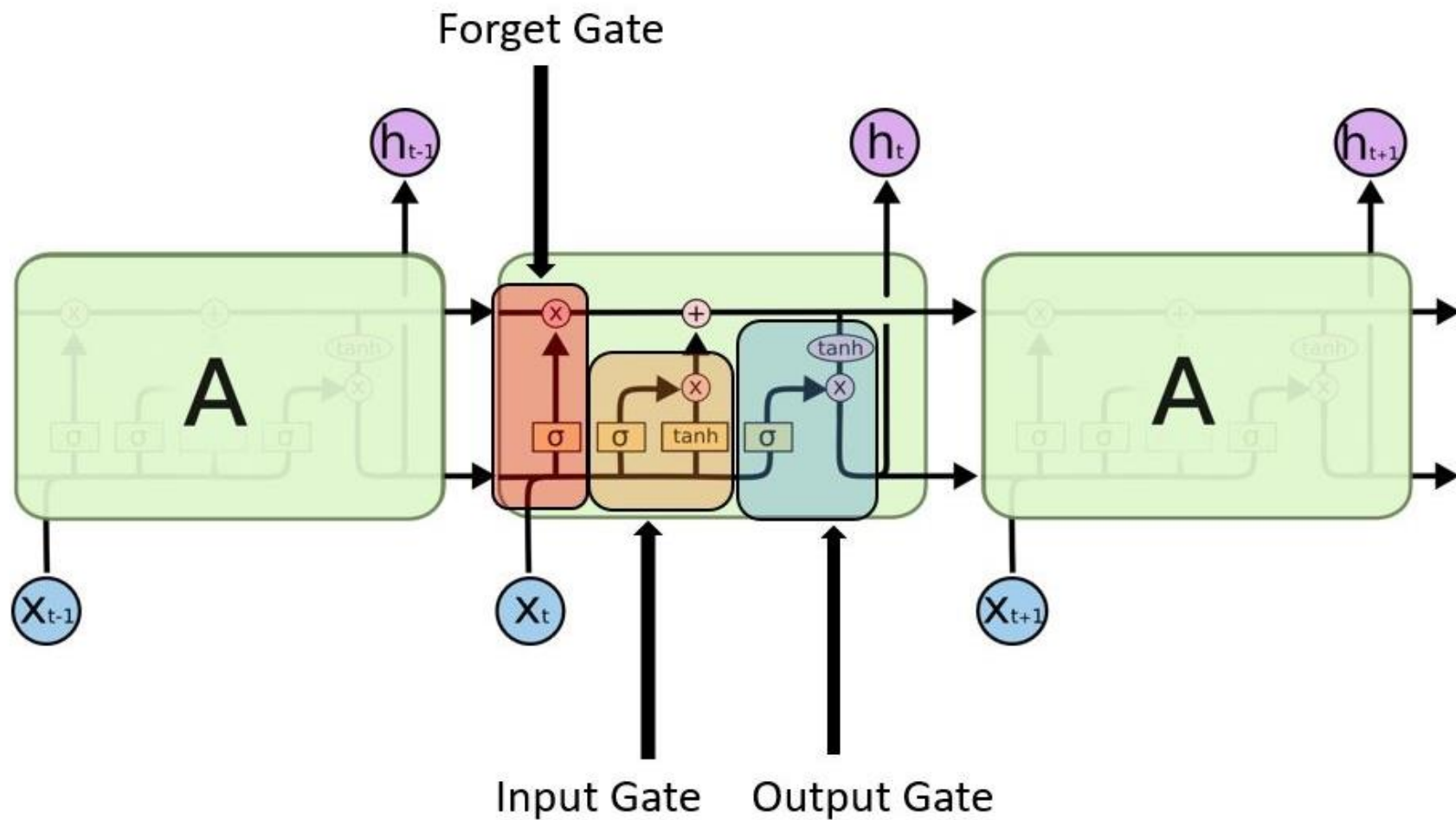
1. простой приближенный решатель POMDP с одним кадром на вход
2. необходимо только смоделировать  $Q(s_t, a_t)$
3. незначительные изменения в архитектуре ванильной DQN

# DRQN архитектура

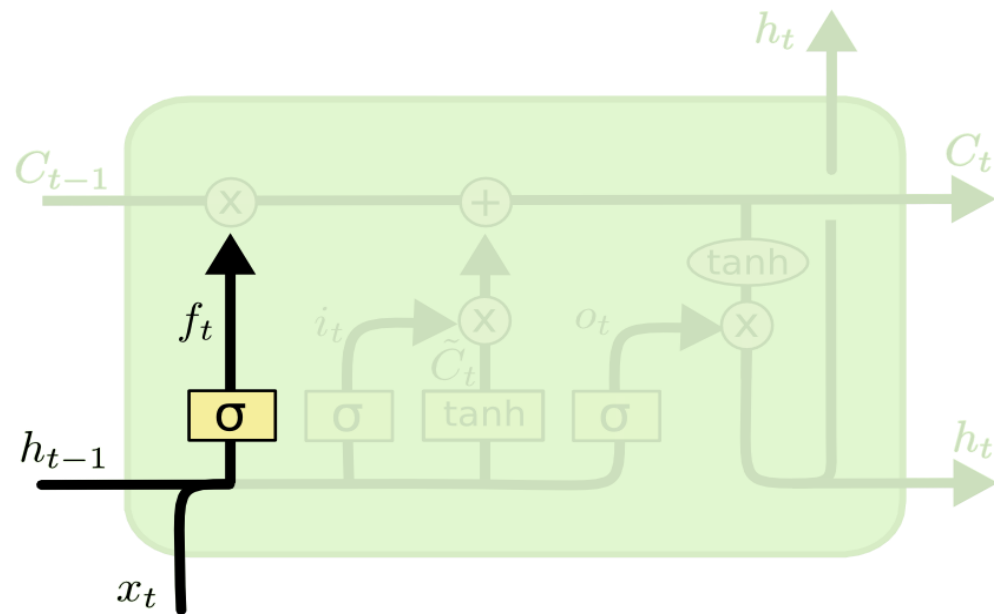




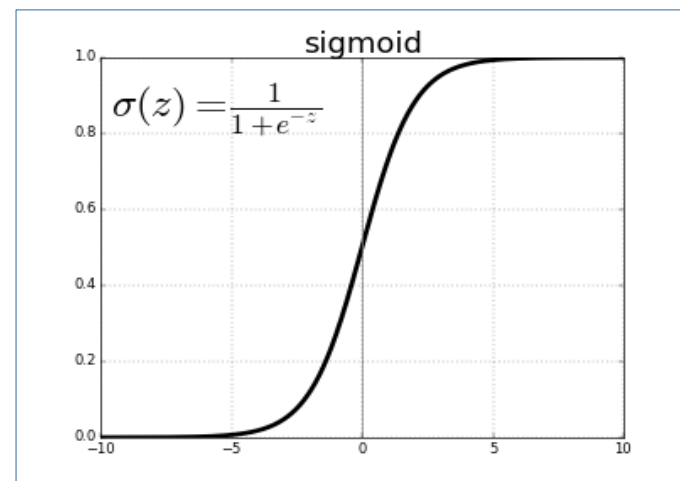


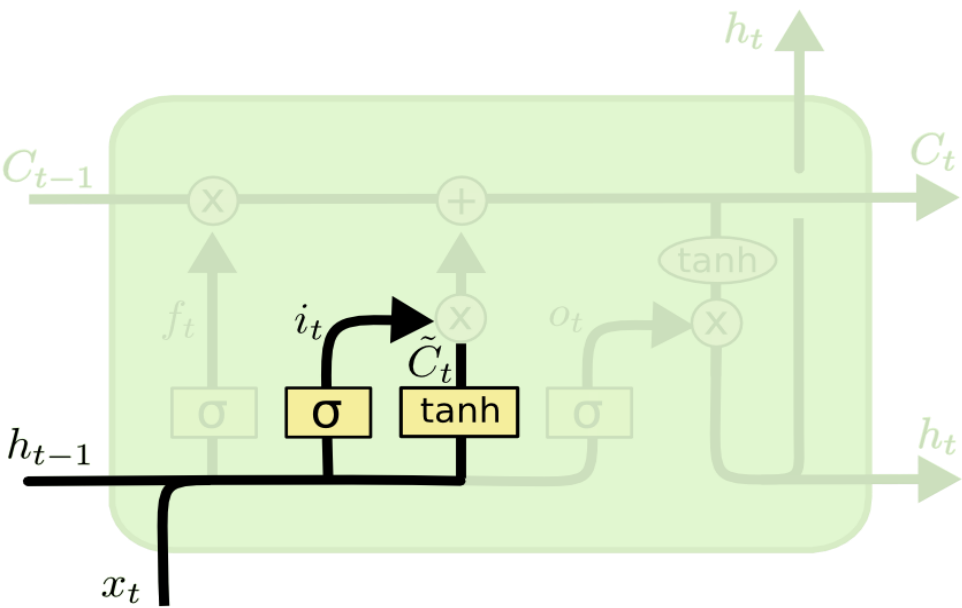






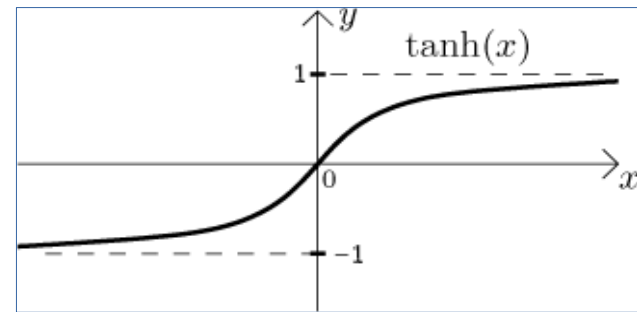
$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

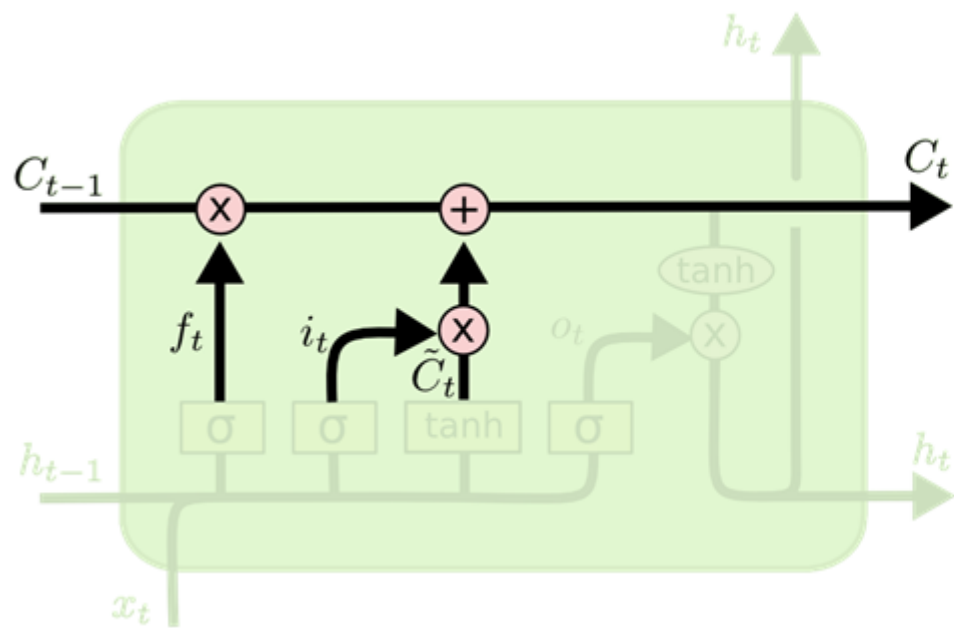




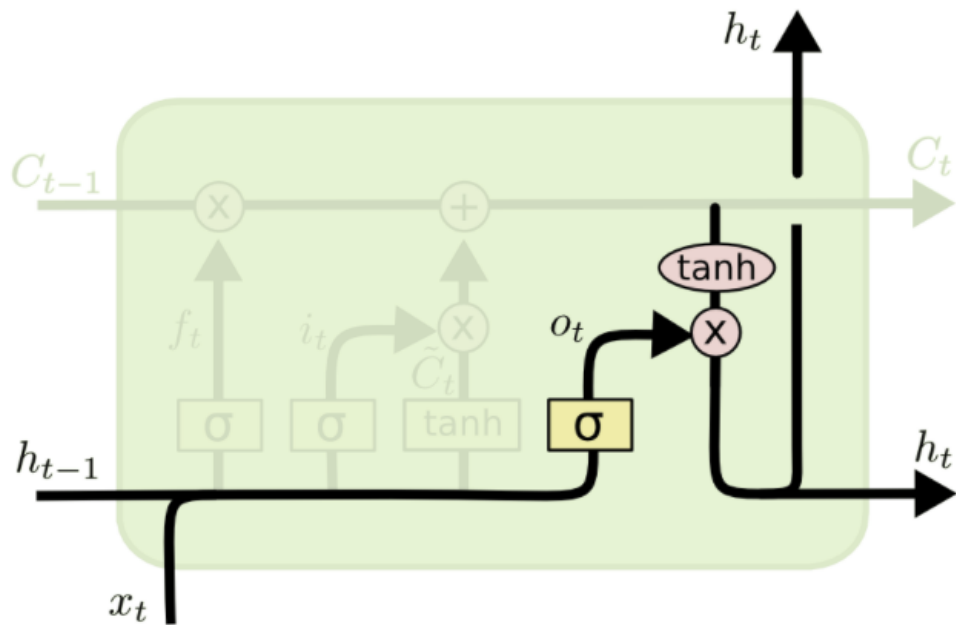
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$





$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



$$o_t = \sigma (W_o [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh (C_t)$$

# ADRQN (Zhu et al., 2017)

Оптимальное обновление убеждений предполагает совершение **действия**

$$b'(s') = p(s'|o', a, b) = \frac{p(o'|s', a)p(s'|a, b)}{\sum_{o'} p(o'|s', a)p(s'|a, b)}$$

ADRQN алгоритм:

- Вход LSTM -  $\{a_{t-1}, o_t\}$ 
  - Встраивание one-hot действия в вектор размерности 512
- Сохранить  $\langle \{a_{t-1}, o_t\}, a_t, r_t, o_{t+1} \rangle$  в Experience Replay
- Обрезать вознаграждения на  $[-1, 1]$ , как это делает DQN
- Схема обновления как в DRQN

# Рекуррентный Experience Replay (ER) - I

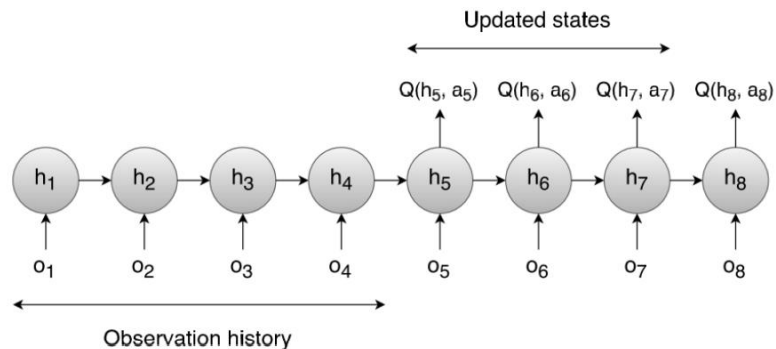
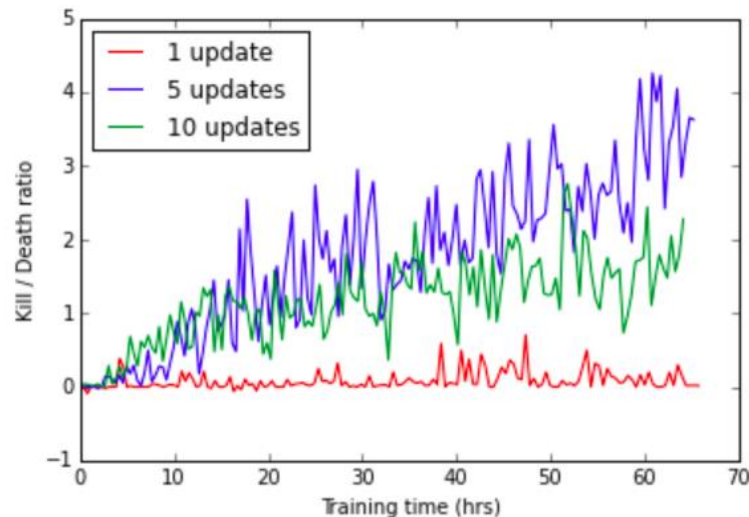
- При неограниченном взаимодействии с окружающей средой не требуется ER
  - асинхронно обучать несколько агентов
  - совместно использовать все параметры (включая параметры LSTM)
  - не делятся состояниями ячеек LSTM
  - приводит к быстрому, почти идентичному обучению
- ❖ *Два оригинальных пути с ER (Хаускнехт и др., 2015)*
  1. Последовательные обновления: случайная выборка полного эпизода из ER и выполнить последовательное обновление
    - Нарушается случайная выборка из ER
    - Обновления коррелируют
  2. Случайные обновления: выборка случайного момента времени в случайном эпизод из ER и обучение на k последующих кадрах
    - Скрытое состояние LSTM должно быть обнулено в начале сессии
    - Первые несколько обновлений потенциально ошибочны

# Рекуррентный Experience Replay (ER) - II

- Основано на истории:  
(Lample et al., 2016) то же самое, что и случайные обновления, но обновляются только последние кадры, т.е. кадры с признаками

$$t + k, \dots, t + \tau - 1$$

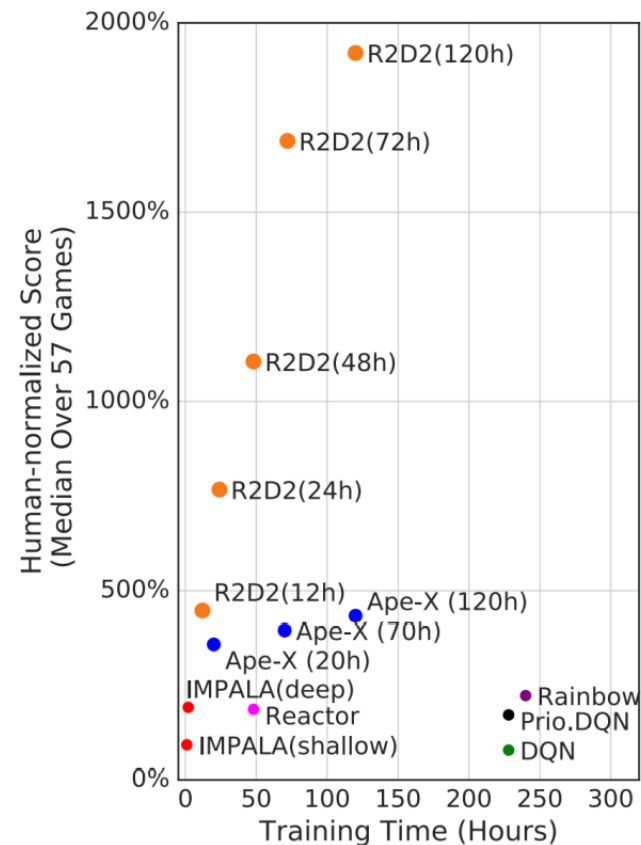
- Эффективность выборки / корреляция



# Рекуррентный ER в распределенной RL (Kapturowski et al., 2019)

## □ Два основных отличия от предыдущих работ:

1. Инициализация РНС с помощью хидденов, собранных при генерации опыта
2. Ввод в РНС: действие, вознаграждение, наблюдения от каждый шаг
  - LSTM
  - N-шаговая оценка TD
  - Изменение масштаба вознаграждения
  - Двойной DQN
  - Дуэльный DQN





# Рекуррентный Advantage Actor Critic

$$\nabla J(\pi_{\theta}) \approx (r + \gamma V(s') - V(s)) \cdot \nabla_{\theta} \log \pi_{\theta}(a|s)$$

Каждая функция состояния  $s$  заменяется скрытым состоянием RNN (LSTM/GRU)

- Рекуррентная  $\pi_{\theta}(a|s)$  (Mnih et al., 2016)
  - Дополнительные 256 ячеек LSTM поверх базовой архитектуры DQN
- Рекуррентная Deterministic Policy Gradient (Song et al., 2017)
  - $Q(h, a)$  как в DRQN, но без BPTT в actor
- Глубокое вариативное обучение с подкреплением (Igl et al., 2018)

# Явная память: нейронная карта

## □ Память существенна

### ▪ *Память в RL имеет разные вкусы*

1. *Темпоральная сверточная память (к последних кадров в DQN)*
  - *простая, но очень ограничивающая*
2. *RNN-подобная память (слой LSTM в DRQN, DARQN)*
  - *емкая, но подходит только для простых задач*
3. *Банкоподобная память вкраплений с преднамеренным доступом на чтение (Oh et al., 2016)*
  - *более интеллектуальная, но может быть избыточной, требует эксперта*
- *4 Человекоподобная пространственная память с преднамеренным доступом для чтения*
- *(Parisotto et al., 2017)*
  - *имеет большой потенциал, но требует структурных предположений*

# Нейронная карта (НМ): (Parisotto et al., 2017)

- *Основные характеристики:*

- *Структурированная память, разработанная специально для RL-агентов в 3D*
- *Размер и вычислительные затраты не растут с временным горизонтом среды*
- *Обучающие алгоритмы: A2C с синхронными обновлениями*

## A2C с синхронными обновлениями

- $r_t = read(M_t)$
- $c_t = contex(c_t, M_t, r_t)$
- $w_{t+1}^{(x_t, y_t)} = write(s_t, r_t, c_t, M_t^{(x_t, y_t)})$
- $M_{t+1} = update(M_t, w_{t+1}^{(x_t, y_t)})$
- $o_t = [r_t, c_t, w_{t+1}^{(x_t, y_t)}]$
- $\pi(a|s_t) = Softmax(f(o_t))$
- $M_t$  – tensor  $C \times H \times W$
- $r_t$  - глобальная информация о NM
- $s_t$  - состояние эмбединга
- $x_t \in \{1, \dots, W\}, y_t \in \{1, \dots, H\}$  положение агента на карте
- $w_{t+1}^{(x_t, y_t)}$  - характеристики для записи
- $o_t$  - выход NM в момент времени  $t$
- $f()$  – сеть политики

# Нейронная карта (NM): подробности работы

1. **Глобальное чтение**,  $r_t = CNN(M_t)$  – 3-х слойная конволюционная сеть (3x3n8), 256fc, 32fc
2. **Контекст**,  $c_t$  - целевое изменение памяти
  1.  $q_t = W[s_t, r_t]$  – запрос (query), относящийся к текущему состоянию
  2.  $\alpha_t^{(x,y)} \propto \exp\left(q_t^T M_t^{(x,y)}\right)$  нормализованное (по оси (x, y)) сходство между признаками NM и запроса
  3.  $c_t = \sum_{(x,y)} \alpha_t^{(x,y)} M_t^{(x,y)}$  - средневзвешенное значение характеристик NM
3. **Локальная запись** вычисляет новые характеристики  $M_t^{(x,y)}$ 
  1.  $w_{t+1}^{(x_t,y_t)} = g\left([s_t, r_t, c_t, M_t^{(x,y)}]\right)$ , где  $g(\cdot)$  другая нейронная сеть (например GRU) с внутренним состоянием равным  $M_t^{(x,y)}$
4. **Обновление** простое - переписываются только характеристики, соответствующие текущему местоположению  $x_t, y_t$   $M_t^{(x,y)} = w_{t+1}^{(x_t,y_t)}$

# Нейронная карта: эмпирические результаты



(a) Maze



(b) Observation

1. Тест на 1000 невидимых лабиринтах
2. Эпизод заканчивается на train/test – 100/500 тиков
3. Положительная награда за нахождение фонаря с правильным светом
4. Отрицательная награда за нахождение фонаря с неправильным светом

Agent	Goal-Search					
	Train			Test		
	7-11	13-15	Total	7-11	13-15	Total
Random	41.9%	25.7%	38.1%	46.0%	29.6%	38.8%
LSTM	60.6%	41.8%	59.3%	65.5%	47.5%	57.4%
MemNN-32	85.1%	58.2%	77.8%	92.6%	69.7%	83.4%
Neural Map	92.4%	80.5%	89.2%	93.5%	87.9%	91.7%
Neural Map (GRU)	<b>97.0%</b>	<b>89.2%</b>	<b>94.9%</b>	<b>97.7%</b>	<b>94.0%</b>	<b>96.4%</b>