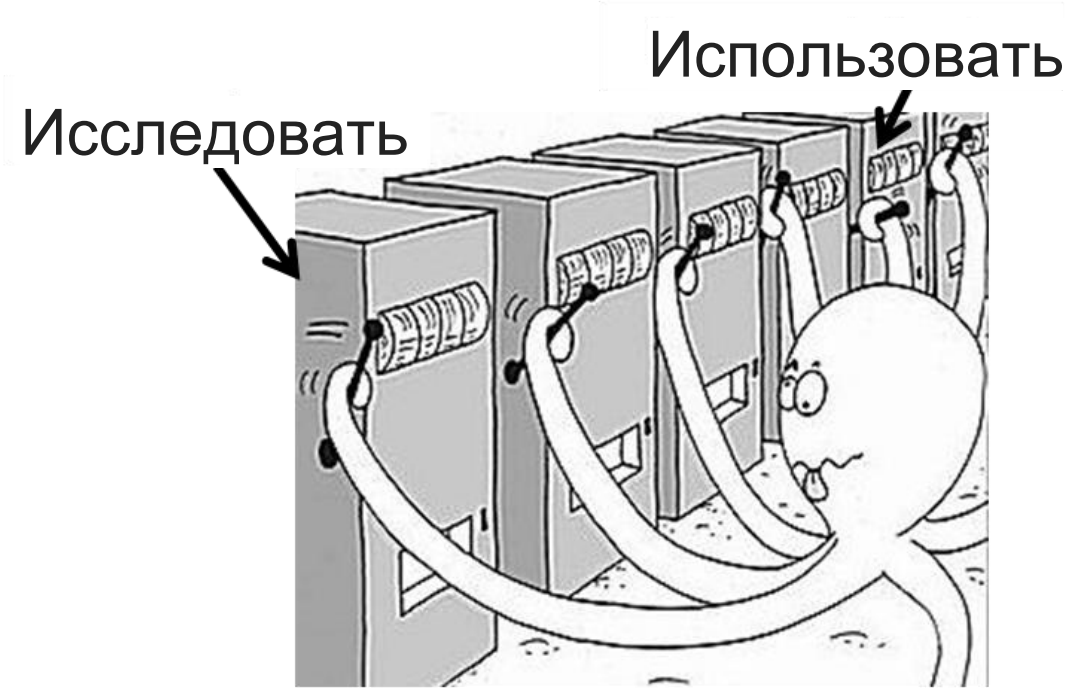
Обучение с подкреплением

Exploration

Задача исследования среды

Epsilon-жадный алгоритм



With probability (1 - epsilon):
$A \leftarrow max_a Q(S, a)$
With probability epsilon:
$A \leftarrow random\ A$
return A

Две основные идеи

Идея: хотим награждать агента за новизну полученной информации о среде

+ Добавить дополнительную награду не сложно
- Что такое новизна?

Идея: будем стараться посещать состояния / пары (state, action) одинаково часто

+ Это равносильно максимизации энтропии распределения
+ Гарантирует совершение оптимальных переходов
- Как поощрять агента за исследование среды?

Count-based алгоритм

1) Совершить действие, получить награду и новое состояние
2) Обновить счетчик посещения состояния

$$N(s) = N(s) + 1$$

1) Посчитать дополнительную награду

$$r^+ = \beta(N(s))^{-\frac{1}{2}}$$

1) Посчитать новую награду за переход

$$r = r + r^+$$

1) Продолжить обучение

Хорошее свойство: $\lim r^+ \underset{N(s)\to\infty}{\longrightarrow} 0$

Count-based алгоритм и дискретизация

Идея: просто дискретизируем состояние

Проблема: как правильно дискретизировать?

Чего хотим:

1. Похожие состояния дискретизируются в одно
2. Разные состояния дискретизируются в разные
3. Количество дискретных состояний не слишком маленькое
4. Количество дискретных состояний не слишком большое

1) Совершить действие, получить награду и новое состояние
$$s_d = \phi(s)$$
2) Обновить счетчик посещения состояния
$$N(s_d) = N(s_d) + 1$$
1) Посчитать дополнительную награду
$$r^+ = \beta N(s_d)^{\frac{-1}{2}}$$
1) Посчитать новую награду за переход
$$r = r + r^+$$
1) Продолжить обучение

Дискретизация: SimHash

Пусть $A \sim \mathcal{N}(0,1)^{k \times D}$ - случайная матрица и $g : S \to \mathbb{R}^D$ - функция предобработки.

Тогда определим хеш как $\phi(s) = \operatorname{sgn} Ag(s)$

Считаем количество посещений для каждого захешированного состояния

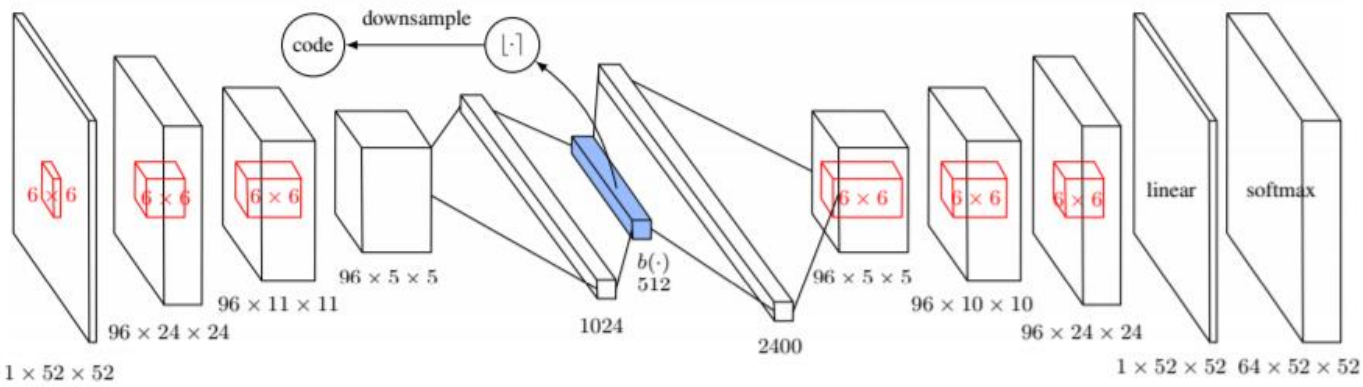
$$\hat{N}(s) = N(\phi(s))$$

Добавочная награда

$$r^+ = \beta(\hat{N}(s))^{-\frac{1}{2}}$$

Эффективность зависит от гранулярности - насколько много информации о состоянии несет в себе хеш. Чем больше k - тем выше гранулярность. Большая или маленькая гранулярность - плохо.

Дискретизация: AutoEncoder



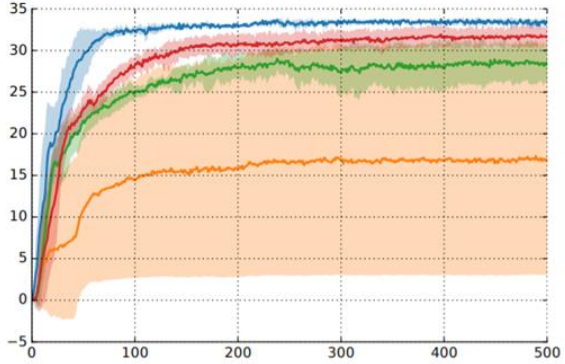
Вместо хеширования используем autoencoder.

Latent space $z \in [-1, 1]^k$

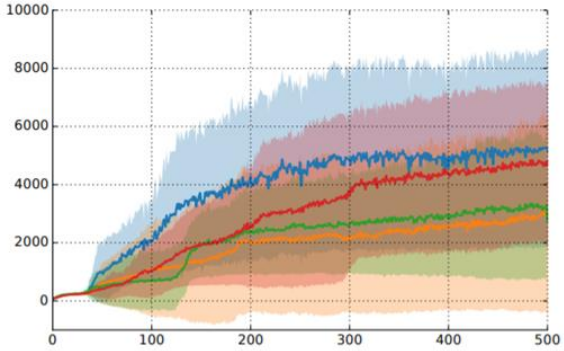
Проблема - слишком высокая информативность компонент z

Решение - добавление константного шума в z

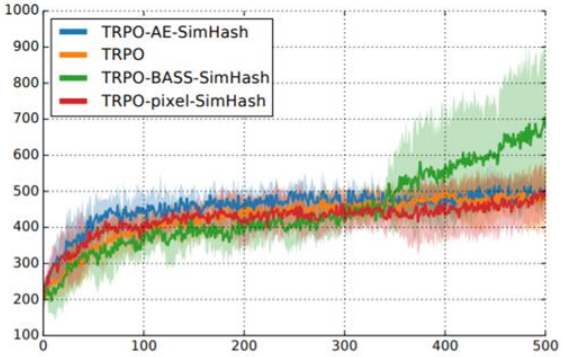
Дискретизация: AutoEncoder



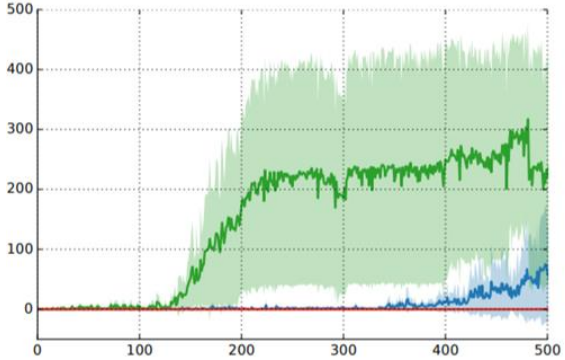
(a) Freeway



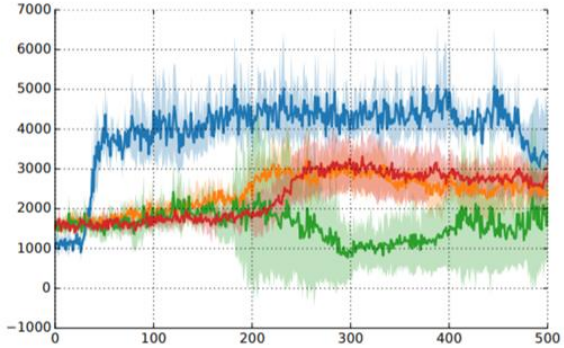
(b) Frostbite



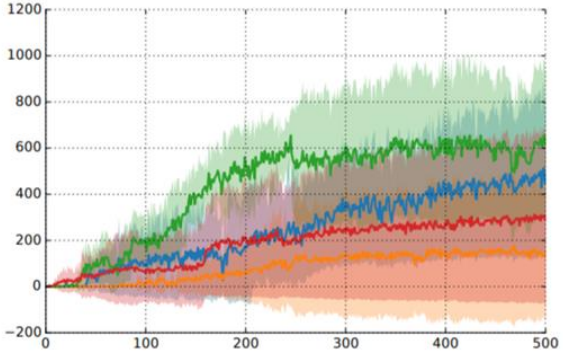
(c) Gravitar



(d) Montezuma's Revenge



(e) Solaris



(f) Venture

Count-based алгоритм и вероятности

Вместо прямого подсчета количества посещения состояний будем его оценивать с помощью вероятностей.

Пусть $\hat{N}(s)$ - оценка числа посещений состояния, а $\hat{n}$ - оценка суммарного количества просмотренных состояний

$p_n(s) = \frac{\hat{N}(s)}{\hat{n}}$ - оценка вероятности попасть в состояние после n обновлений оценки распределения

$p_{n+1}(s) = \frac{\hat{N}(s)+1}{\hat{n}+1}$ - после одного шага обновления

Count-based алгоритм и вероятности

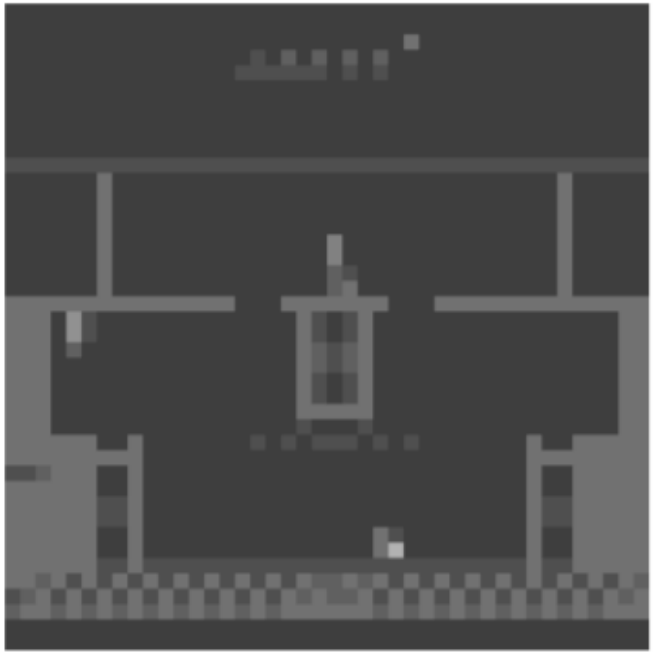
$p_n(s) = \frac{\hat{N}(s)}{\hat{n}}$ - оценка вероятности попасть в состояние после n обновлений оценки распределения

$p_{n+1}(s) = \frac{\hat{N}(s)+1}{\hat{n}+1}$ - после одного шага обновления

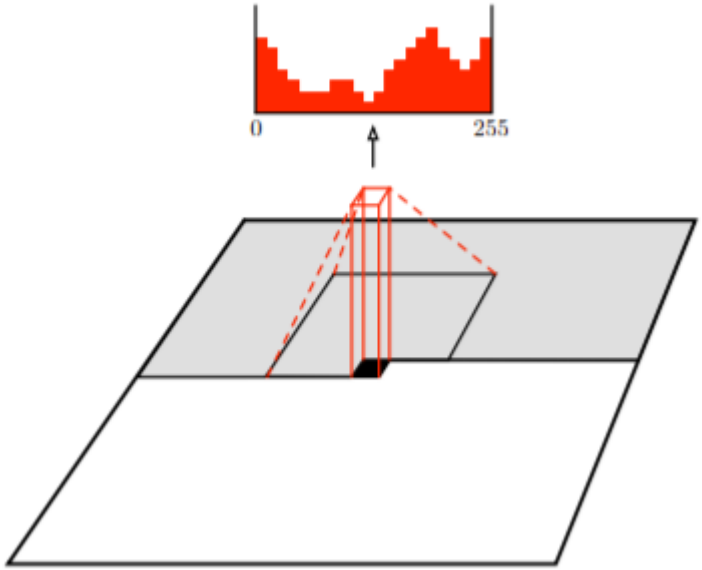
Как теперь посчитать $\hat{N}(s)$?

$$\hat{N}(s) = p(s)\hat{n} = \frac{p_n(s)(1-p_{n+1}(s))}{p_{n+1}(s)-p_n(s)}$$

Atari: Montezuma's Revenge



Оцениваем распределение: Gated PixelCNN



1	1	1	1	1
1	1	1	1	1
1	1	0	0	0
0	0	0	0	0
0	0	0	0	0

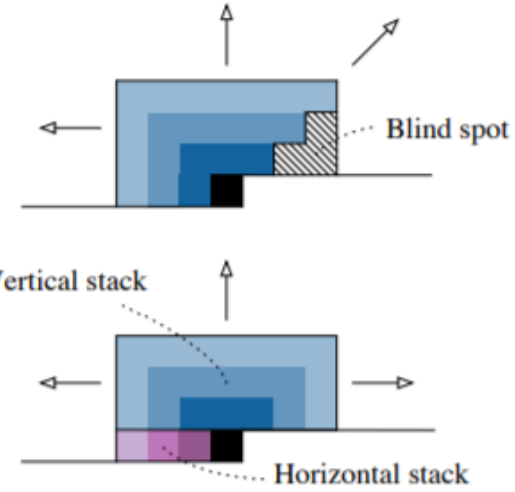
Предсказание вероятностей

Маска. Скрывает следующие пиксели

$$p(x) = \prod p(x_i | x_1, \ldots, x_{i-1})$$

Оцениваем распределение: Gated PixelCNN

Проблема: видим только ограниченную часть изображения. Если сеть глубокая - теряем много информации



Решение: сделать gate так же, как в LSTM

Вместо обычных сверток используем

$$y = \tanh(W_f * x) \odot \sigma(W_g * x)$$

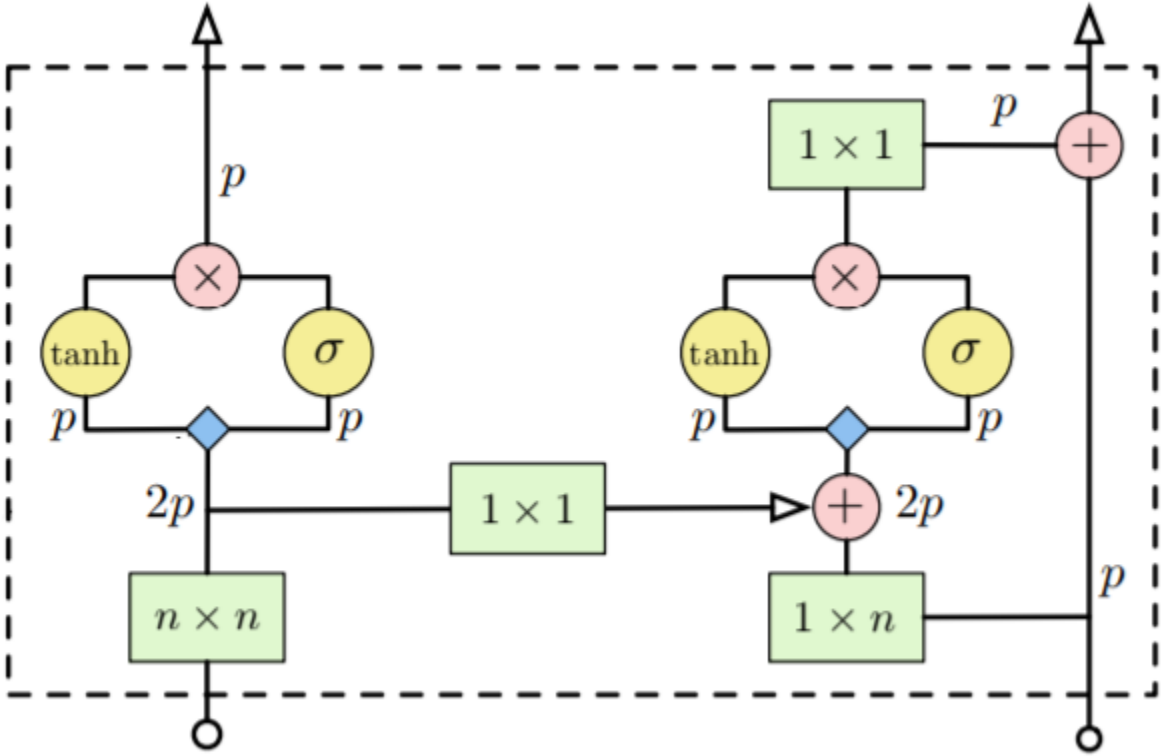
Здесь $\odot$ - поэлементное произведение

$*$ - операция свертки

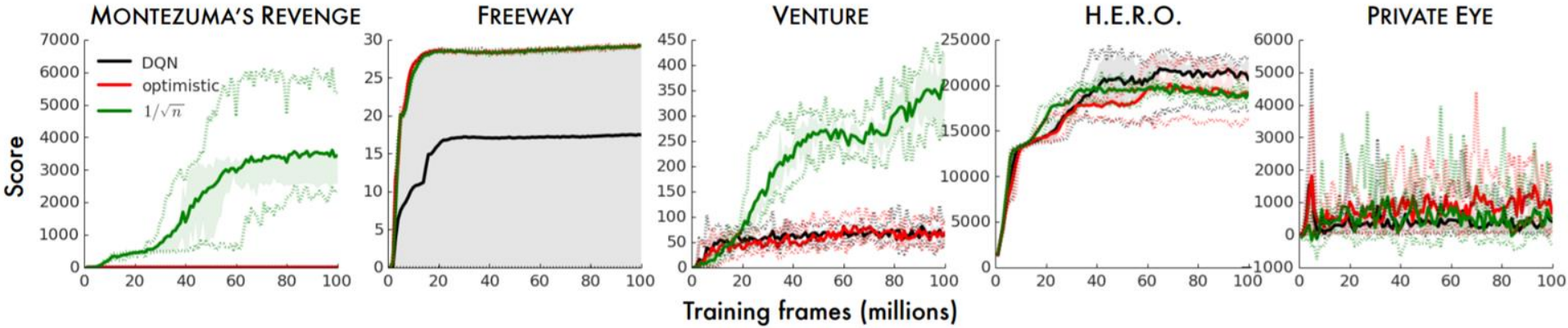
Проблема: не видим правую часть картинки

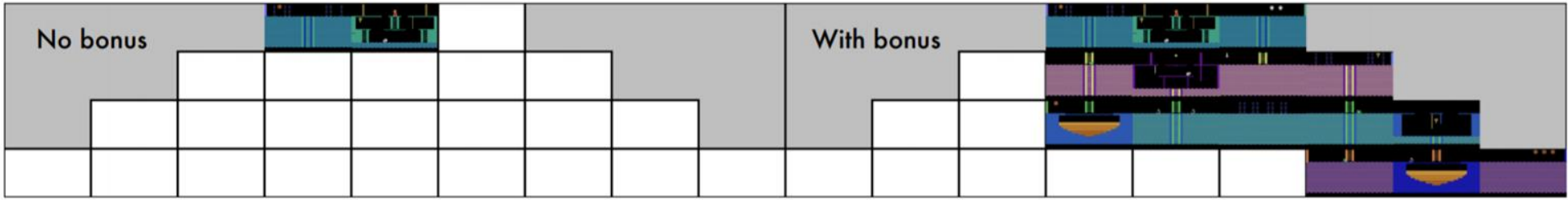
Решение: сделать разные фильтры для последней строки и для верхней части

Оцениваем распределение: Gated PixelCNN



Count-based exploration и Gated PixelCNN





Random Network Distillation

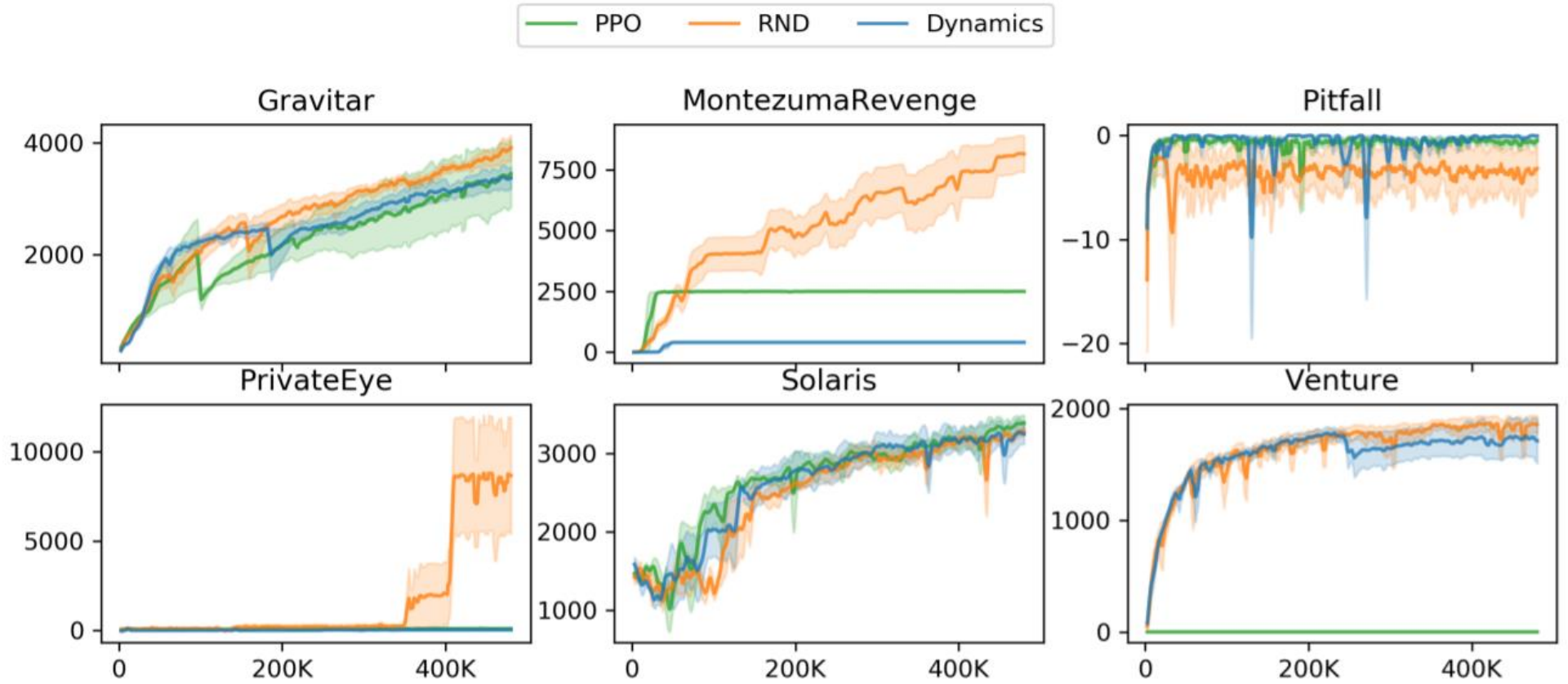
$f(s) : S \to \mathbb{R}^k$ - target network со случайной инициализацией

$f'(s, \theta) : S \to \mathbb{R}^k$ - predictor network с параметрами θ

Минимизируем $||f'(s, \theta) - f(s)||^2$

Интуиция: со временем ошибка меньше для тех состояний, на которых уже обучались

Random Network Distillation



Curiosity-driven exploration

Пусть $\phi(s)$ - представление состояния в некотором latent space

Будем предсказывать закодированное состояние по действию и предыдущему состоянию

$$\hat{\phi}(s_{t+1}) = f_{\theta_F}(\phi(s_t), a)$$

Для обучения предсказателя используем функцию потерь

$$L_F = ||\hat{\phi}(s_{t+1}) - \phi(s_{t+1})||_2^2$$

Дополнительная награда за любопытство

$$r^+ = \frac{\beta}{2}||\hat{\phi}(s_{t+1}) - \phi(s_{t+1})||_2^2$$

Интуиция: со временем переход станет “скучным”, а потому $r^+ \to 0$

Curiosity-driven exploration

Как выбрать ϕ?

Критерии выбора:

1. Компактность

   Latent space должен быть как можно более сжатым и отражать только то, что зависит от действий агента

1. Достаточность

   Latent space должен быть достаточным для описания зависящей от действий

   агента динамики среды

1. Стабильность

   Чем меньше функция меняется со временем - тем лучше, так как повышается сходимость

Curiosity-driven exploration

Как выбрать ϕ?

Существующие варианты:

1. $\phi(s) = s$ - само состояние.
   Нет компактности, зато стабилен и достаточен.
1. $\phi(s) = g(s)$, где g - случайная сеть.
   Компактность может быть, а может и не быть. Достаточность также может быть, а может - нет. Есть стабильность.
1. $\phi(s) = g(s)$, где g - VAE.
   Нет стабильности, может не быть компактности и достаточности.
1. $\phi(s) = g(s)$, где g - Inverse Dynamics Features.
   Нет стабильности, есть компактность и достаточность.

Inverse Dynamics Features

Идея: научиться предсказывать действия по двум состояниям

$$\hat{a}_t = g_{\theta_G}(\phi(s_t), \phi(s_{t+1}))$$

Одновременно обучаем и g, и ϕ

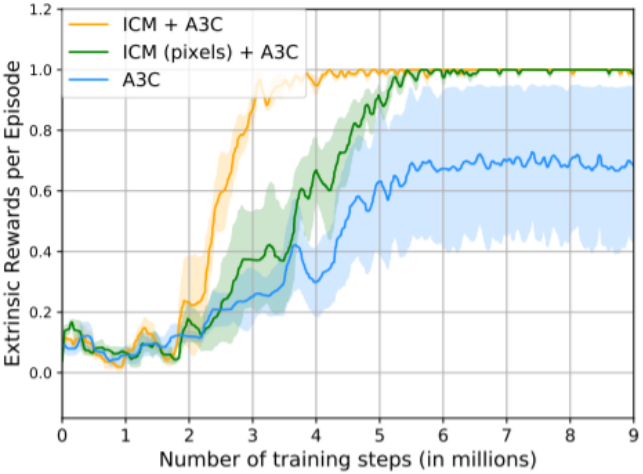
Функция потерь для непрерывных действий

$$L_G = ||a_t - \hat{a}_t||_2^2$$

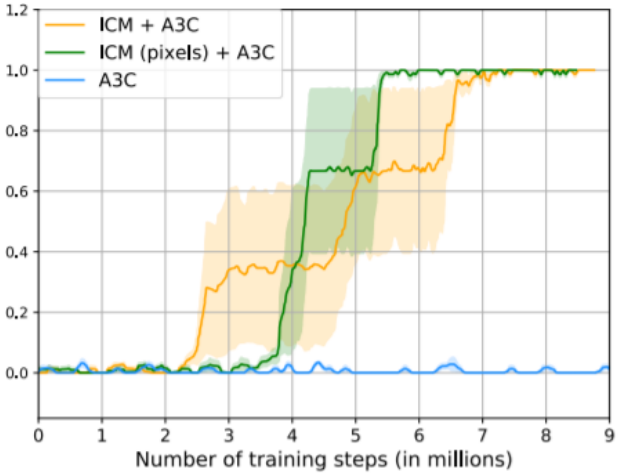
Для дискретных действий используем softmax и cross entropy loss

Интуиция: такой подход позволит выделить в состоянии те признаки, которые важны для предсказания динамики среды.

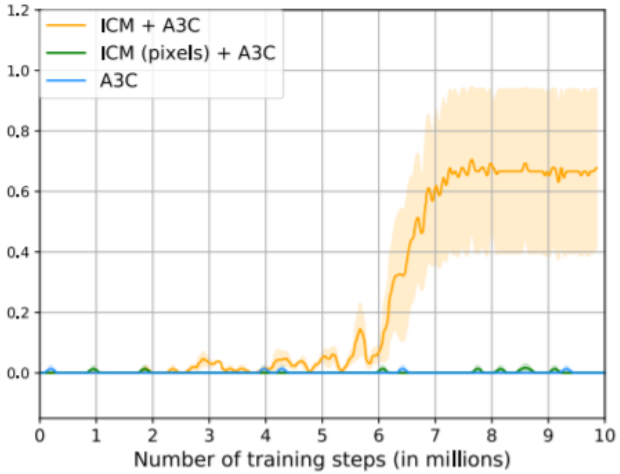
Curiosity-driven exploration



(a) “dense reward” setting

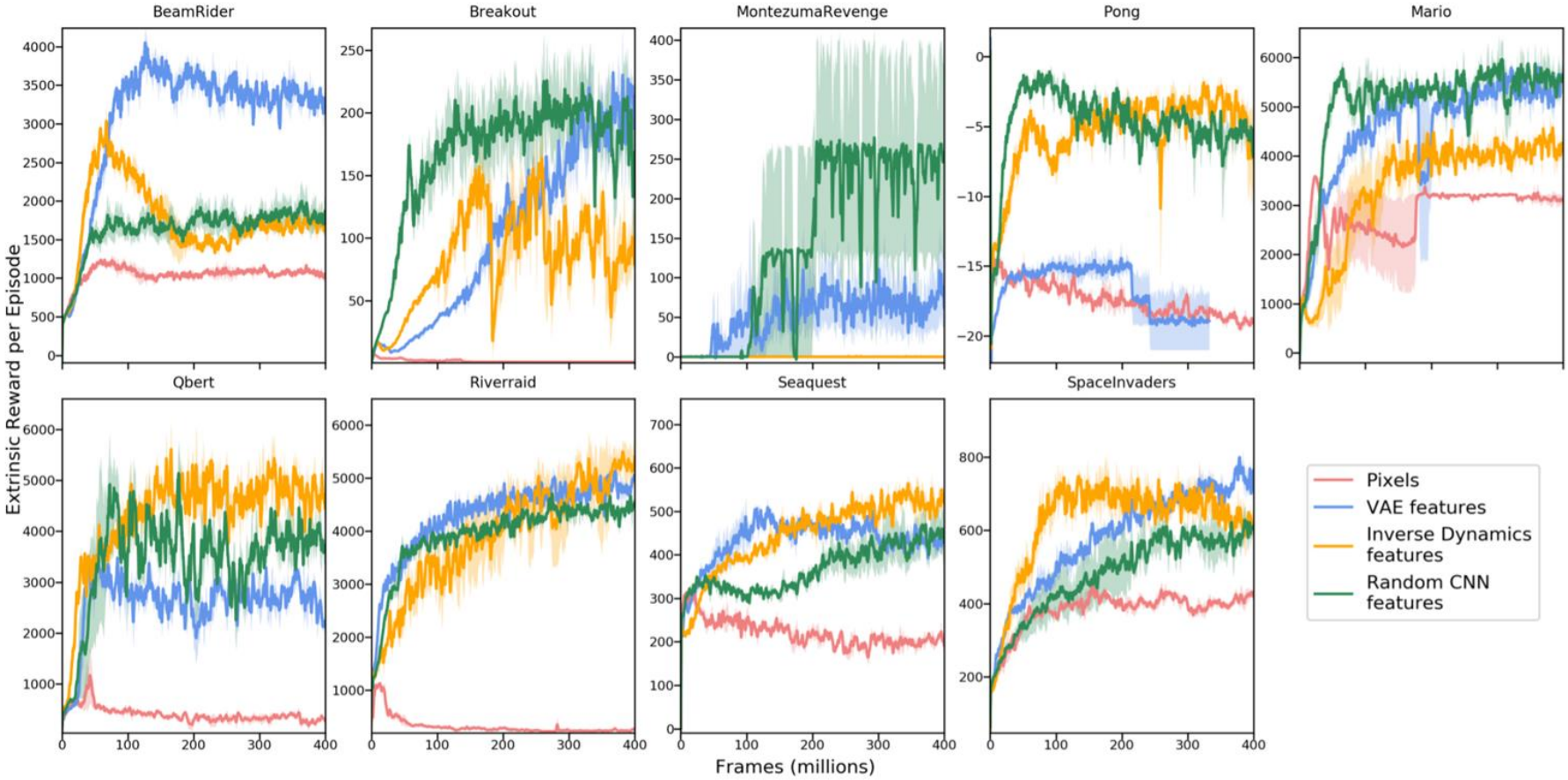


(b) “sparse reward” setting



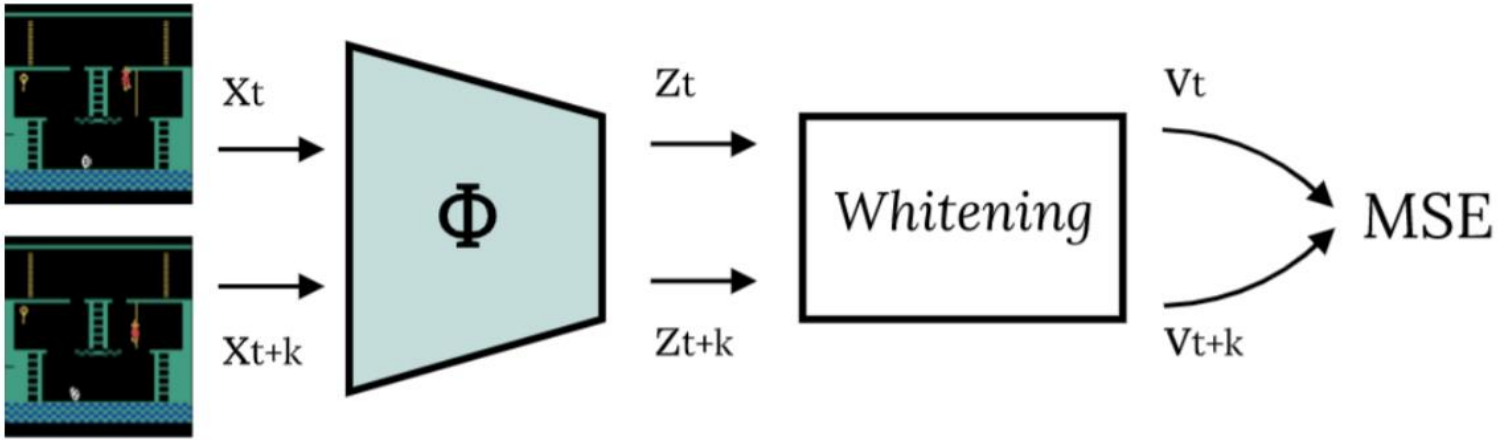
(c) “very sparse reward” setting

Curiosity-driven exploration



Latent WM For Intrinsically Motivated Exploration

Закодируем состояние среды в латентное пространство, а затем будем использовать ошибку предсказания латентной модели мира в этом пространстве для определения новизны полученного состояния.



Чтобы представление среды не выродилось в константу, репараметризуем латентное пространство по батчу.

Латентное пространство состояний

Хотим решить такую задачу оптимизации:

$$min_\theta \, \mathbb{E} \, ||\mathbf{v}_t - \mathbf{v}_{t+k}||_2^2$$

$$s.t. \; cov(\mathbf{v}_t, \mathbf{v}_t) = cov(\mathbf{v}_{t+k}, \mathbf{v}_{t+k}) = I$$

Ограничение на ковариацию означает, что все компоненты вектора независимы, и что для каждой компоненты среднеквадратичное отклонение равно 1

Whitening operation

Whitening - это функция, которая репараметризует множество векторов так, чтобы оно было несмещенным, а матрица ковариаций была равна identity

$$\text{Whitening} : Z \to Y$$

$$\text{Whitening}(z) = W_Z(z - \mu_Z)$$

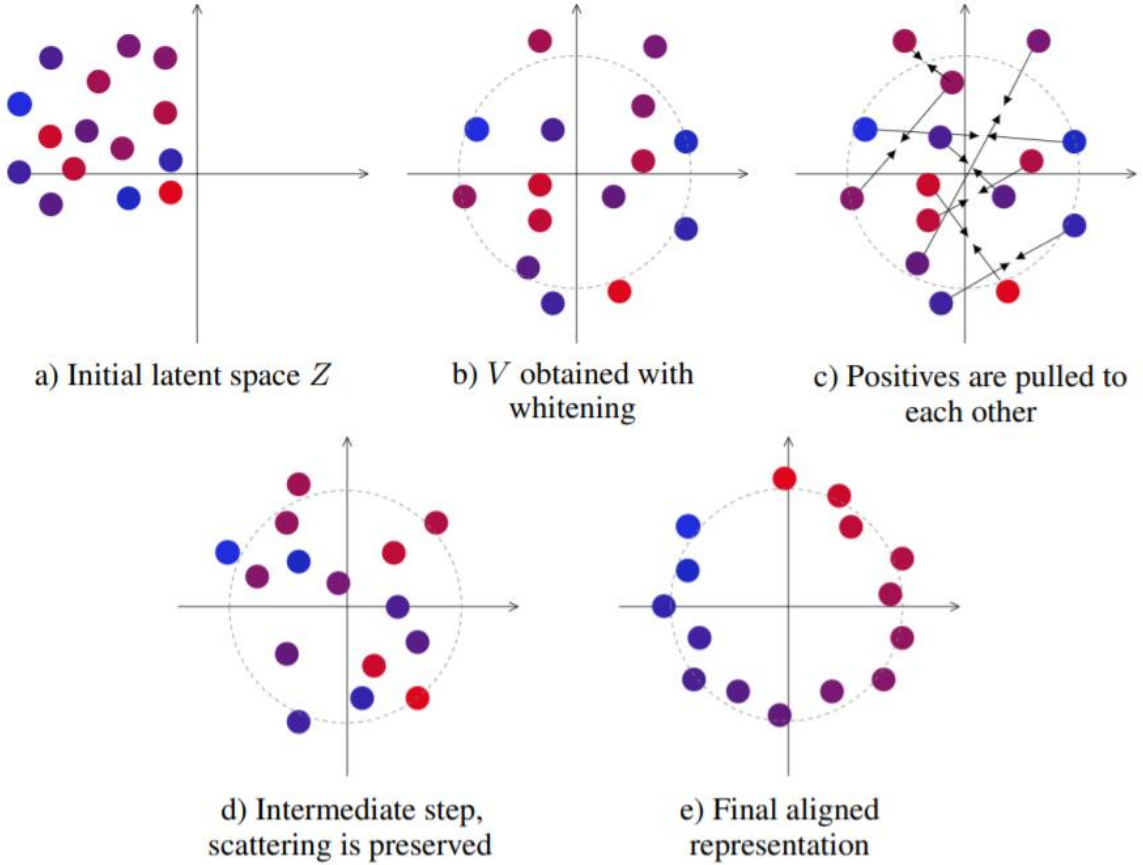
В данной формуле:

$$\mu_Z = \frac{1}{N}\sum z_i$$

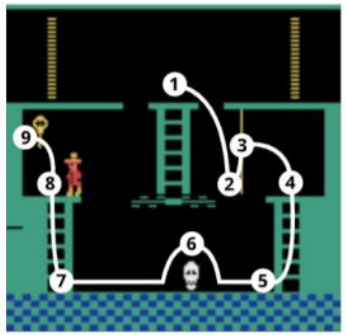
$$W_Z^T W_Z = \Sigma_Z^{-1}$$

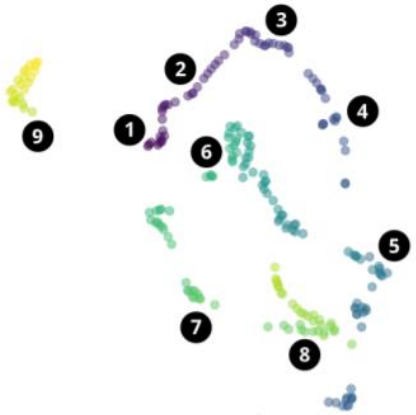
$$\Sigma_Z = \frac{1}{N-1}\sum (z_i - \mu_i)(z_i - \mu_i)^T$$

Whitening operation

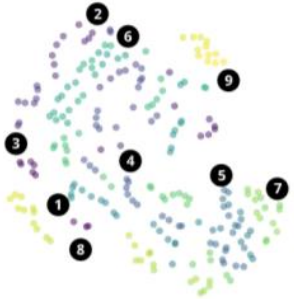


Whitening operation



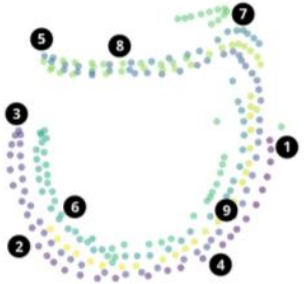


W-MSE



RF

IDF



CPC|Action

Latent World Model

В качестве модели мира используем рекуррентную нейронную сеть со скрытым состоянием b.

Обучаем с помощью MSE предсказания, и так же определяем внутреннюю награду:

$$r_{\text{intrinsic}} = L_{\text{LWM}} = MSE(\text{LWM}(\Phi(s), a, b), \Phi(s'))$$

Atari

	EMI [19]	EX2 [10]	ICM [25]	RND [6]	AE-SimHash [33]	LWM
Freeway	**33.8**	27.1	33.6	33.3	33.5	30.8
Frostbite	7002	3387	4465	2227	5214	**8409**
Venture	646	589	418	707	445	**998**
Gravitar	558	550	424	546	482	**1376**
Solaris	2688	2276	2453	2051	**4467**	1268
Montezuma	387	0	161	377	75	**2276**