Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning

Guo, X., Singh, S., Lee, H., Lewis, R. L., & Wang, X. (2014).
In*Advances in Neural Information Processing Systems* (pp. 3338-3346).

2015/5/21

D1 金子 貴輝

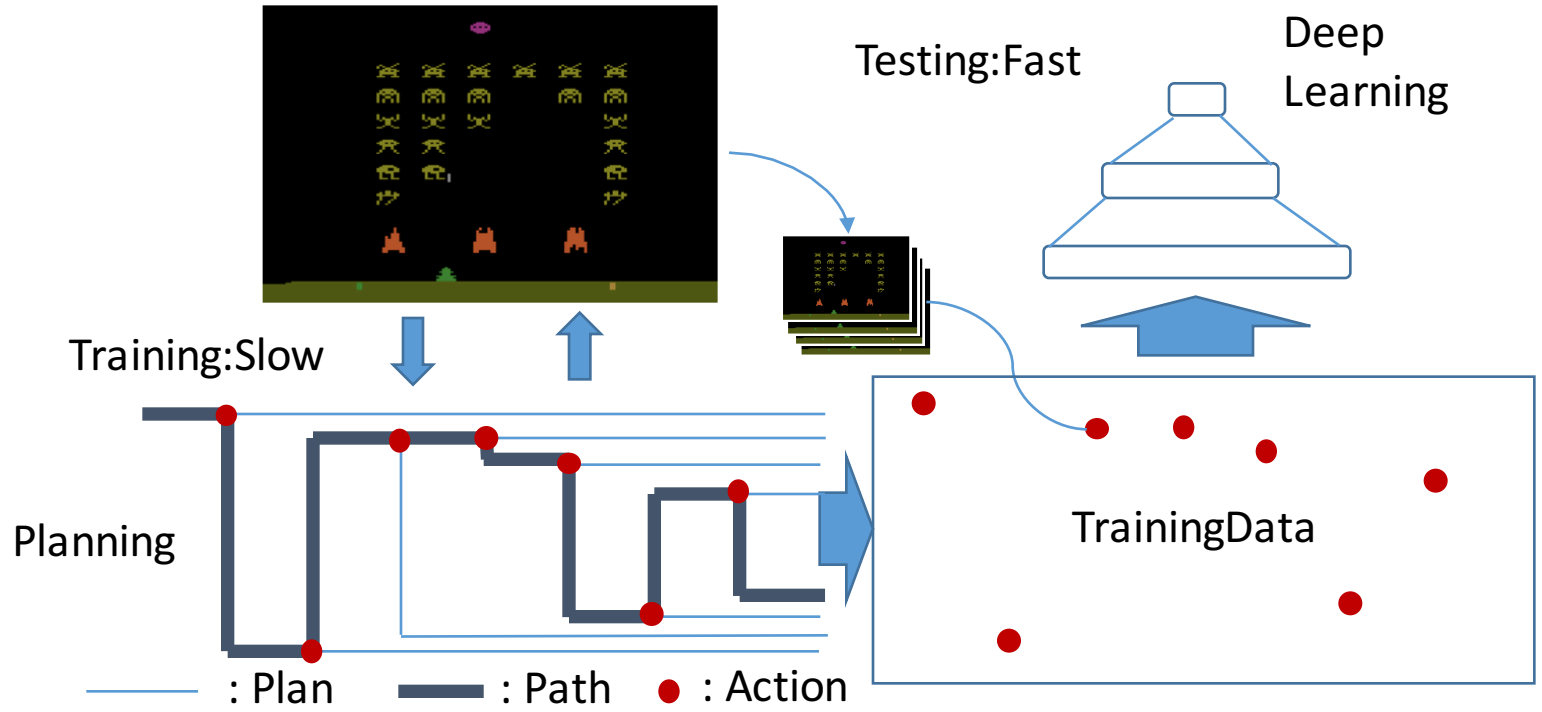
目次

- Abstract
- Introduction
- 従来手法
- 提案手法
- 実験結果
- 結論

目次

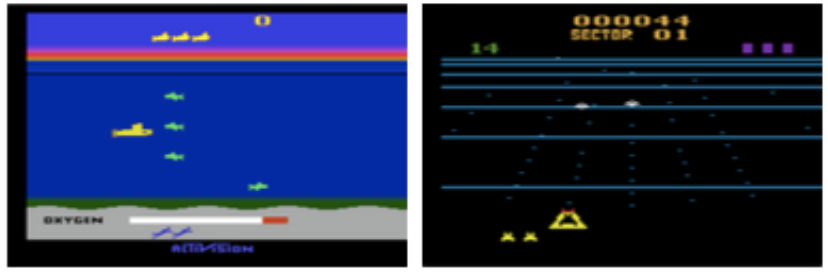
Abstract

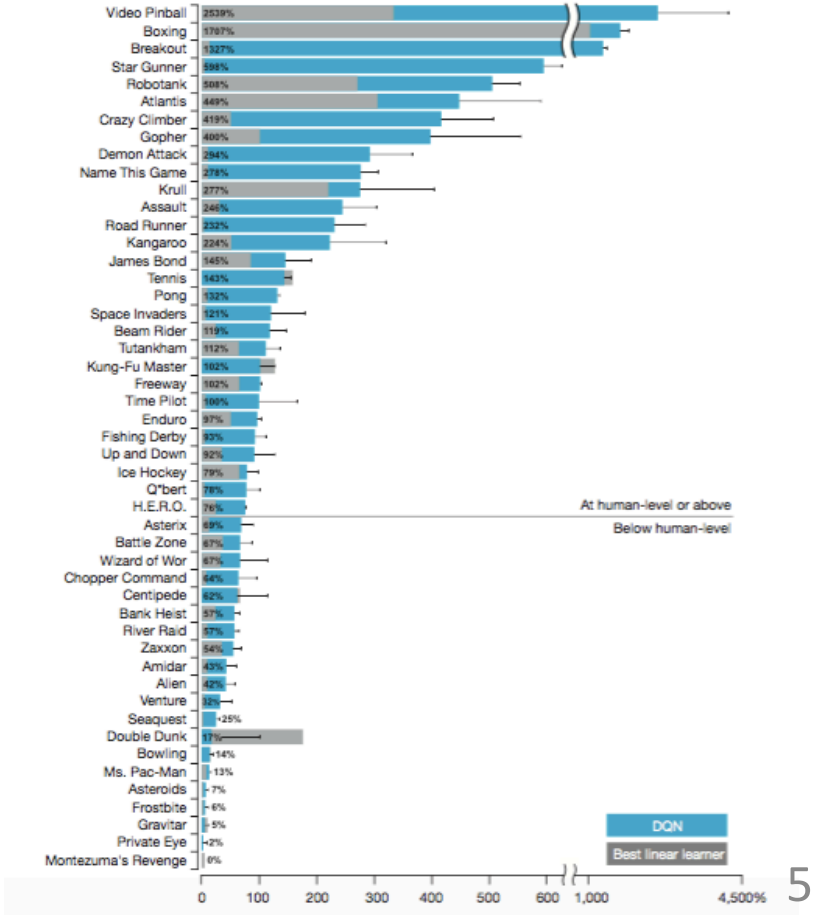
- NIPS2014 poster paper
- Planning RL + DL beats Deep Q-Network in 7 games.
- They uses MonteCarlo Tree Search for planning, and makes training data for DL.



Introduction

- DQNs which solve Atari2600 uses all the same hyper-parameters, no handcrafted features, and in real time, gets high score ,

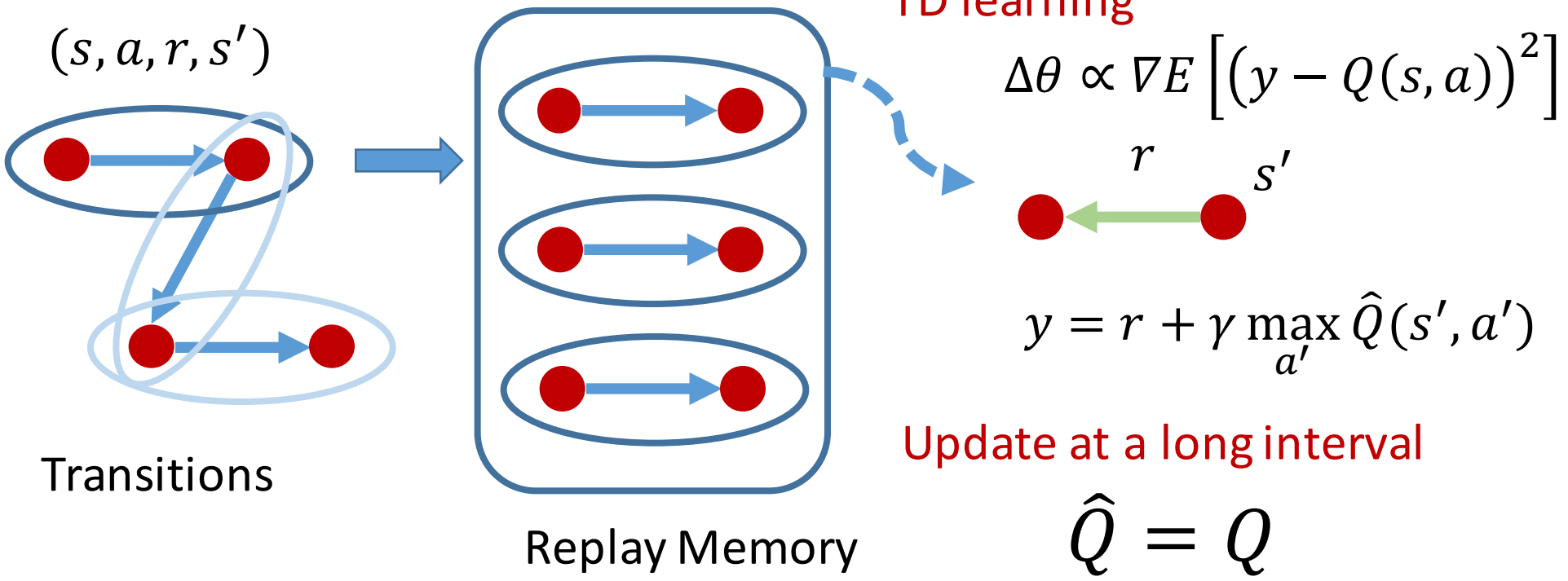




Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *Nature* 518.7540 (2015): 529-533.

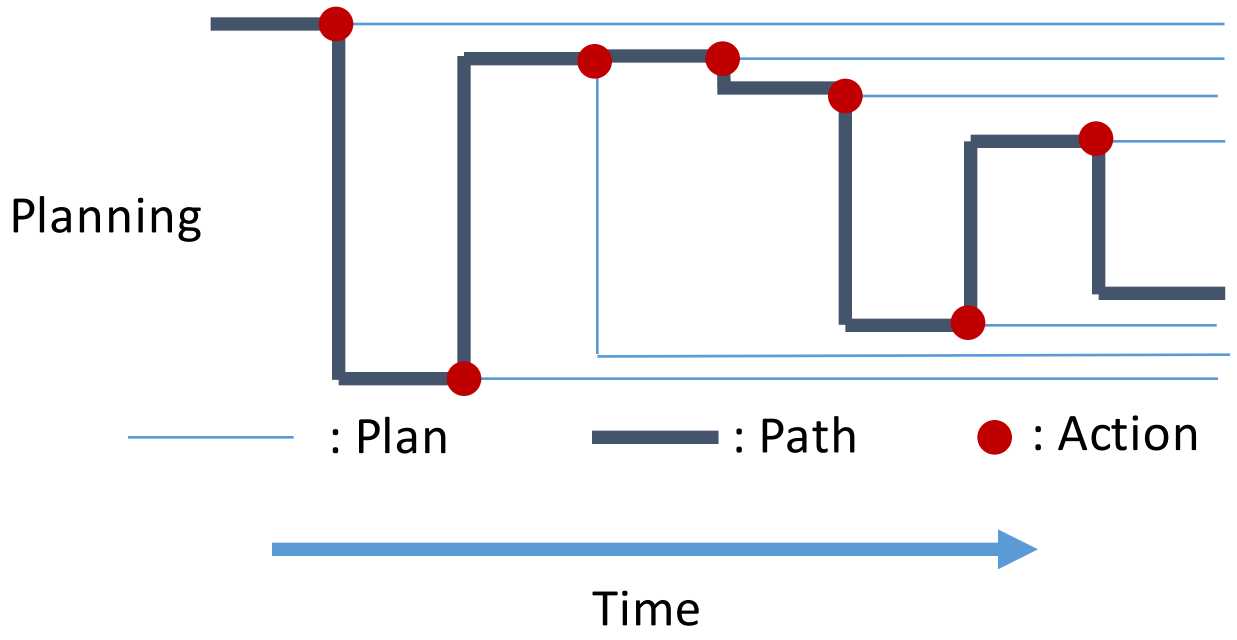
Introduction

- However, the part of RL in DQN is Q-Learning, which is weak in foreseeing.



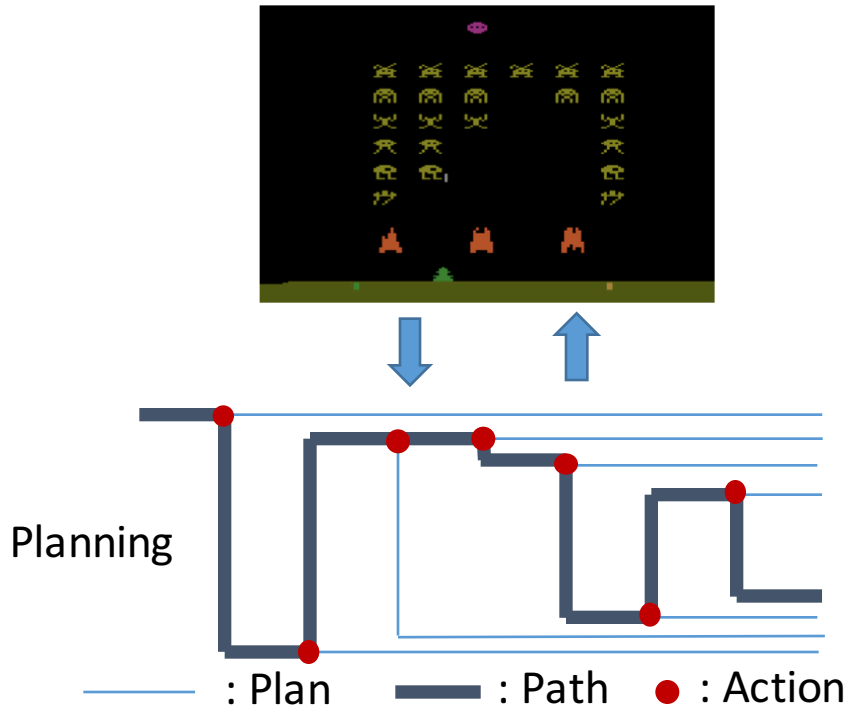
Introduction

- On the other hand, UCT(a kind of MonteCarlo Tree Search) directly foresee the future states and rewards.(planning)



Introduction

- But this method is too slow, and requires access to emulator.

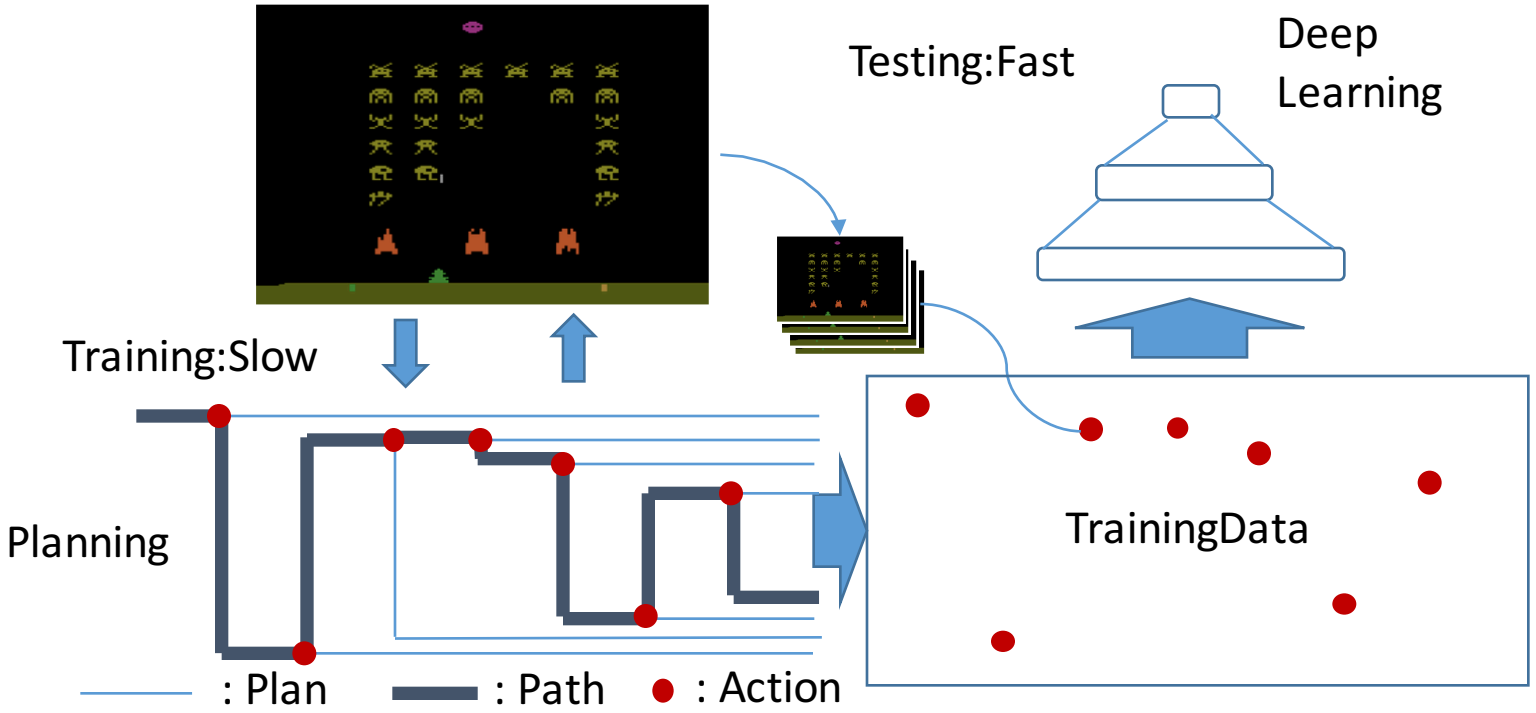


各状態で毎回，先読みする
先読みは深く回数も多い

計算がかなり重い

Introduction

- So, by using UCT at training, and using Deep Learning at testing, We obtain accurate and fast agent at testing.



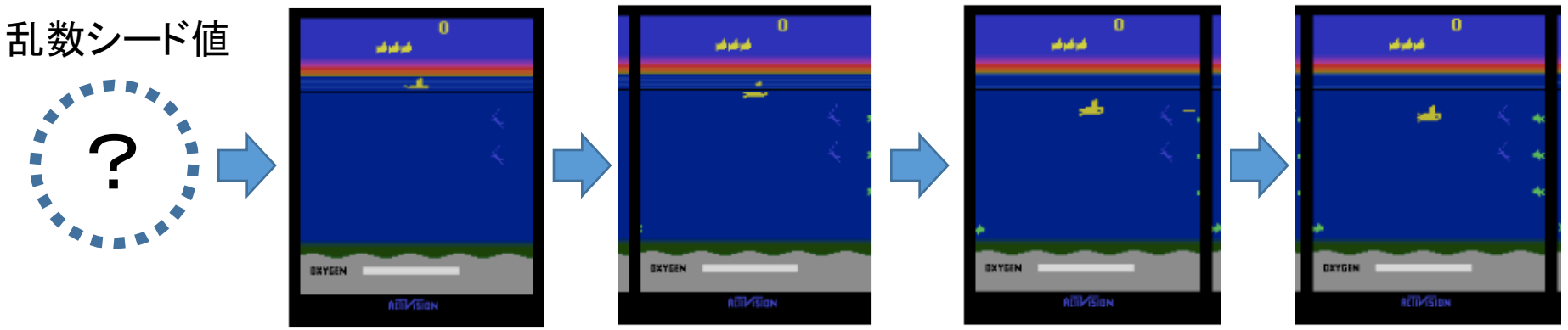
目次

- Abstract
- Introduction
- **従来手法**
- 提案手法
- 実験結果
- 結論

従来手法：Atari2600の問題設定

- 160x210 7bit 2D Frameを見て，最大18の離散的行動を選択する.
- 決定論的に状態遷移する. (乱数シードを指定すれば)

ディスプレイからはPOMDP



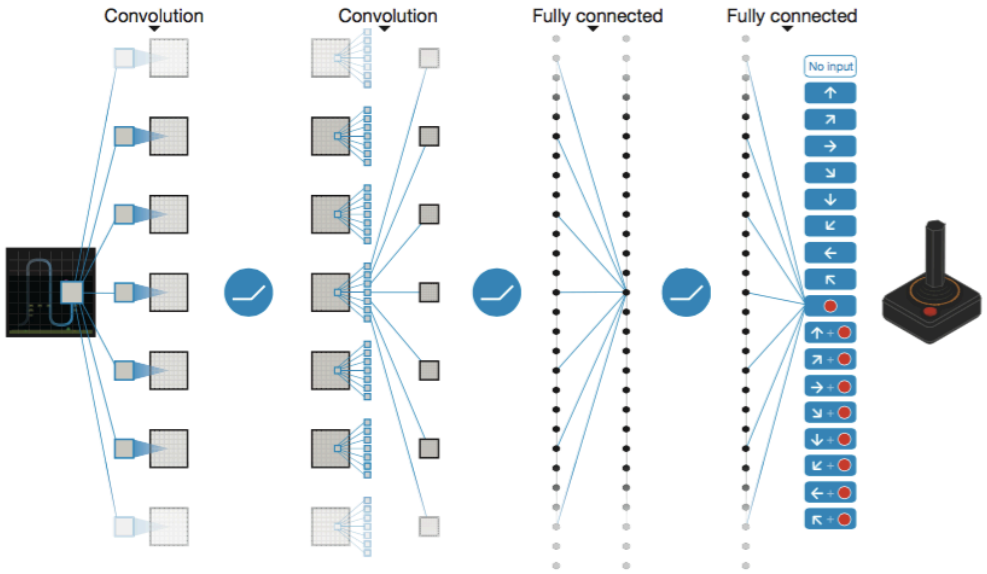
メモリからはMDP

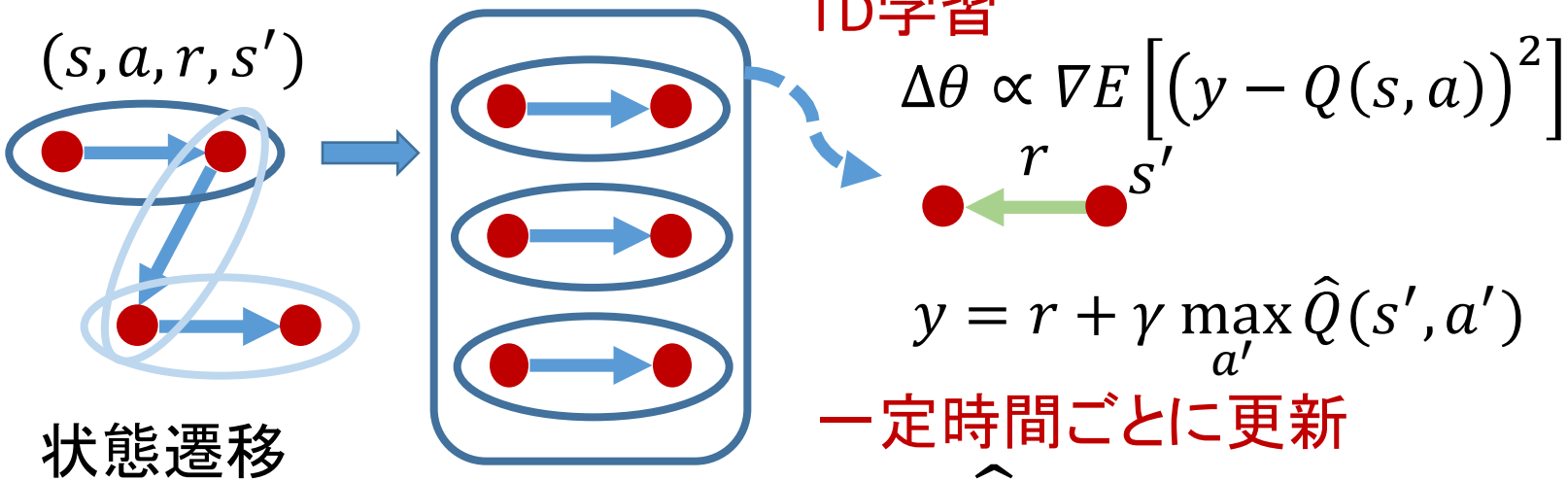
従来手法：Atariへのモデルフリーな手法

- モデルを使えないのでPOMDPとして問題を解く.
- 特徴量作り込みしてPOMDPの解決する手法が多かった.
- DQNは3層のCNNとQ-Learningで，このクラスでは最強.
  - 作り込み特徴の代わりに4フレームの入力を使用

従来手法：DQN

- 画面の画像からQ値(行動価値)を計算するCNN
- リプレイメモリに貯めた遷移をランダムに取り学習





TD学習

$$\Delta\theta \propto \nabla E\left[\left(y - Q(s,a)\right)^2\right]$$

r s'

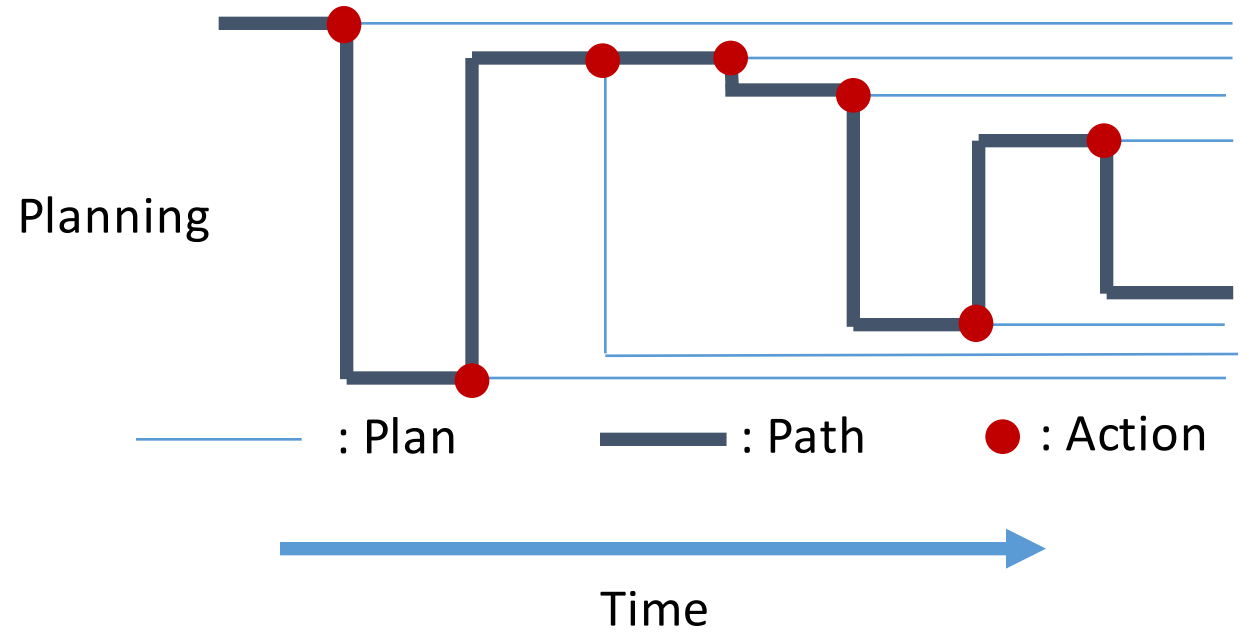
$$y = r + \gamma \max_{a'} \hat{Q}(s', a')$$

一定時間ごとに更新

$$\hat{Q} = Q$$

従来手法：UCT

- モンテカルロ木探索の一種(Upper Confidence bounds applied to Trees)
- モデルを使って，先読みを繰り返して行動価値を蓄えていく
- 追求項（状態深さ対の割引報酬のモンテカルロ平均）＋探索項（$=\sqrt{\log(n(s,d))/n(s,a,d)}$）のスコアを大きくするように軌跡を拡大する.
- 最大深度までスコア計算が終わると，根元まで戻り，行動を選択する.



目次

- Abstract
- Introduction
- 従来手法
- **提案手法**
- 実験結果
- 結論

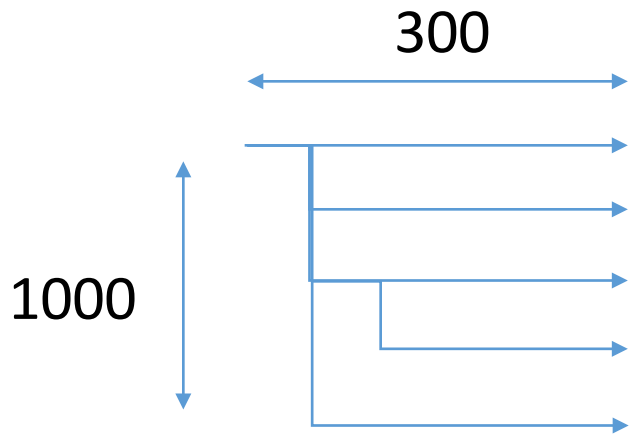
提案手法は3つ

- 回帰
- 分類
- 分類インターバル

提案手法：UCTの共通部分

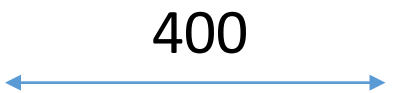
- UCTの２つのパラメータ設定．精度を上げるためなるべく大きめに．
- エピソードを800回反復するのに数日要した．

先読みの深さと本数



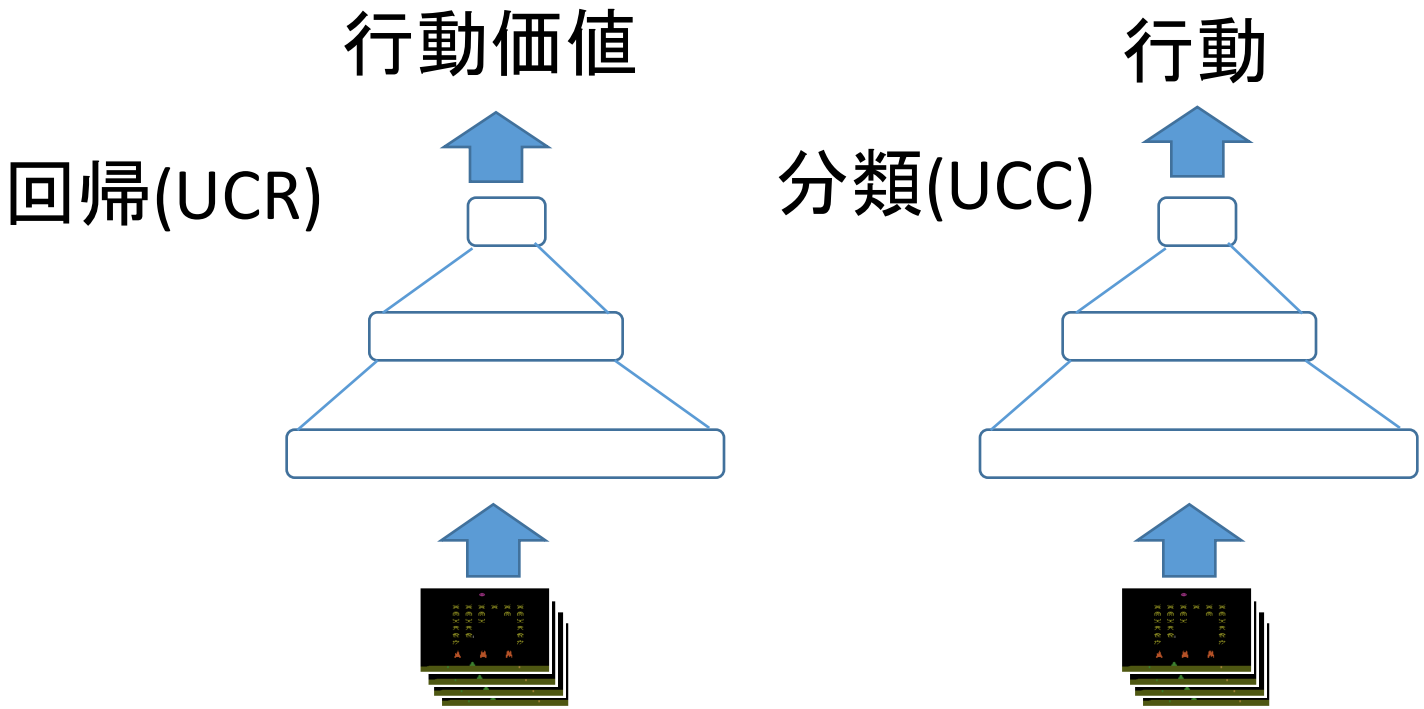
Pong

Enduro



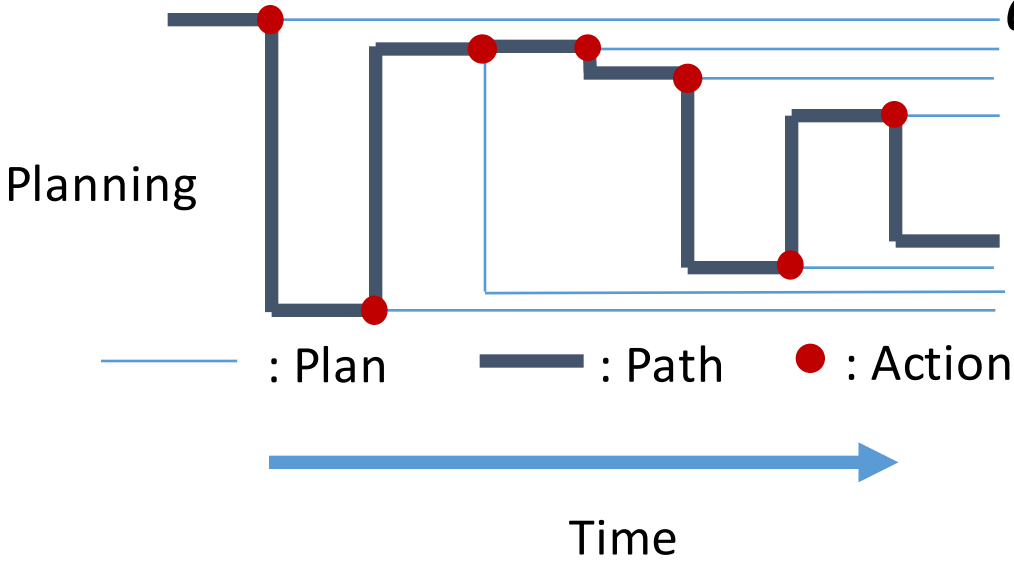
提案手法：データセットから回帰する，分類する

- UCTの800回の反復から，各状態の前の4フレームとその状態で取った行動を元に，CNNを学習させる．
- 行動価値を推定する回帰(UCTtoRegression)か，
  UCTと貪欲法で選ばれる行動を推定する分類(UCTtoClassification)か．

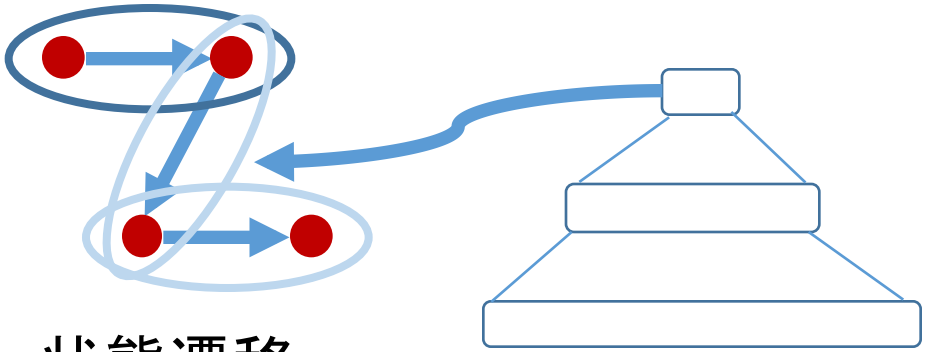


提案手法：手本どおりにできない問題

- UCCで学習されたCNNでも，　UCTの手本どおりに行動できるとは限らない
  → 手本が示した状態と，実際に必要になる状態は異なる
    → スコアが低くなる



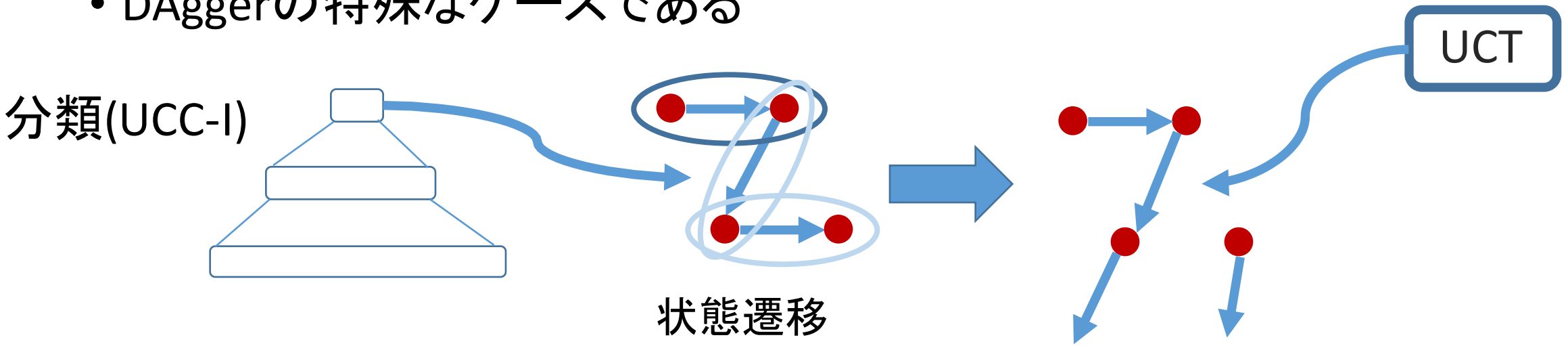
$$d^{\pi_{UCT}}(s) \neq d^{\pi_{CNN}}(s)$$



状態遷移

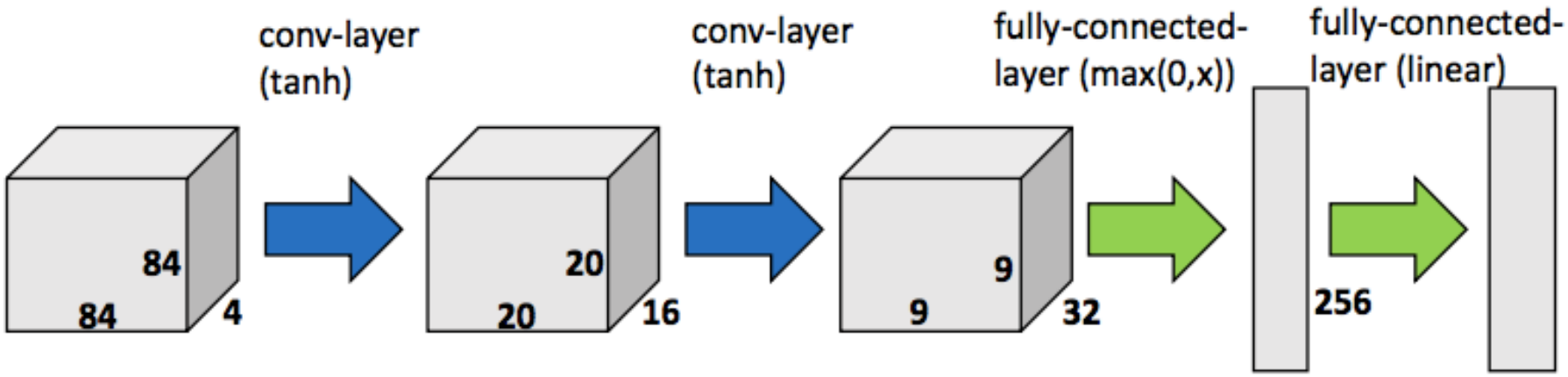
提案手法：UCTtoClassification-Interleaved

- 200回の手本でCNN教えた後，実際の状態遷移から，訓練データ(200)を再びUCTで作成する.
- 回帰接続は性能悪かったので不採用.
- CNNでの状態遷移は5%のε-greedy.
- DAggerの特殊なケースである



提案手法：CNN側の設定

- DQNと同じCNNを使用して比較可能に.
- DQNでの報酬のクリッピングは行わなかった.
- DQNと同様にSpace InvaderではK=3ほかではK=4のフレームスキップを行った. スキップ中は同じ出力が続く.



目次

- Abstract
- Introduction
- 従来手法
- 提案手法
- **実験結果**
- 結論

実験一覧

- 比較実験
- CNN特徴量の可視化
- 方策の可視化

比較実験ではスコアがほぼ向上

Agent	*B.Rider*	*Breakout*	*Enduro*	*Pong*	*Q*bert*	*Seaquest*	*S.Invaders*
DQN	4092	168	470	20	1952	1705	581
-best	5184	225	661	21	4500	1740	1075
UCC	5342 (20)	175(5.63)	558(14)	19(0.3)	11574(44)	2273(23)	672(5.3)
-best	10514	351	942	21	29725	5100	1200
-greedy	5676	269	692	21	19890	2760	680
UCC-I	5388(4.6)	215(6.69)	601(11)	19(0.14)	13189(35.3)	2701(6.09)	670(4.24)
-best	10732	413	1026	21	29900	6100	910
-greedy	5702	380	741	21	20025	2995	692
UCR	2405(12)	143(6.7)	566(10.2)	19(0.3)	12755(40.7)	1024 (13.8)	441(8.1)

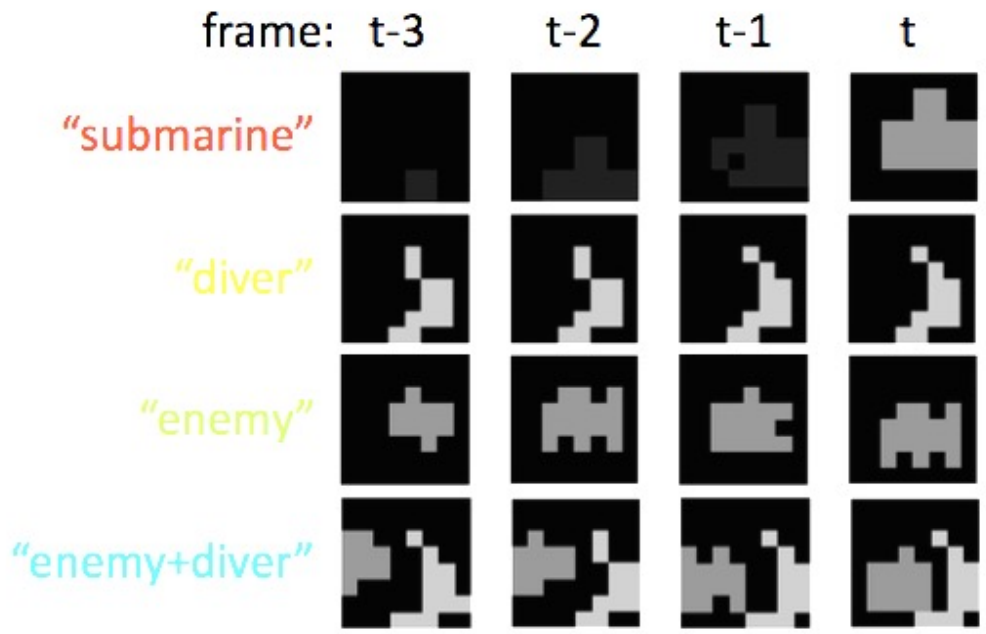
- DQNと提案3手法のスコア比較
- 括弧内が標準偏差，DQNは5%のε-greedy.
- テスト中に探索しないgreedyの行も用意した.

比較実験ではスコアがほぼ向上

Agent	*B.Rider*	*Breakout*	*Enduro*	*Pong*	*Q*bert*	*Seaquest*	*S.Invaders*
DQN	4092	168	470	20	1952	1705	581
-best	5184	225	661	21	4500	1740	1075
UCC	5342 (20)	175(5.63)	558(14)	19(0.3)	11574(44)	2273(23)	672(5.3)
-best	10514	351	942	21	29725	5100	1200
-greedy	5676	269	692	21	19890	2760	680
UCC-I	5388(4.6)	215(6.69)	601(11)	19(0.14)	13189(35.3)	2701(6.09)	670(4.24)
-best	10732	413	1026	21	29900	6100	910
-greedy	5702	380	741	21	20025	2995	692
UCR	2405(12)	143(6.7)	566(10.2)	19(0.3)	12755(40.7)	1024 (13.8)	441(8.1)

- 分類(UCC)は回帰(UCR)より強く，DQNよりもPong以外強い
  (Pongの最大スコアは21).
- おおよそ，UCC-IはUCCを改善できている.
- Enduroでは1600回反復でも試し，UCCで581，UCC-Iで670まで上昇.
  → より大きいデータセットだと，UCC-Iが優位になる.

特徴の可視化：1層目

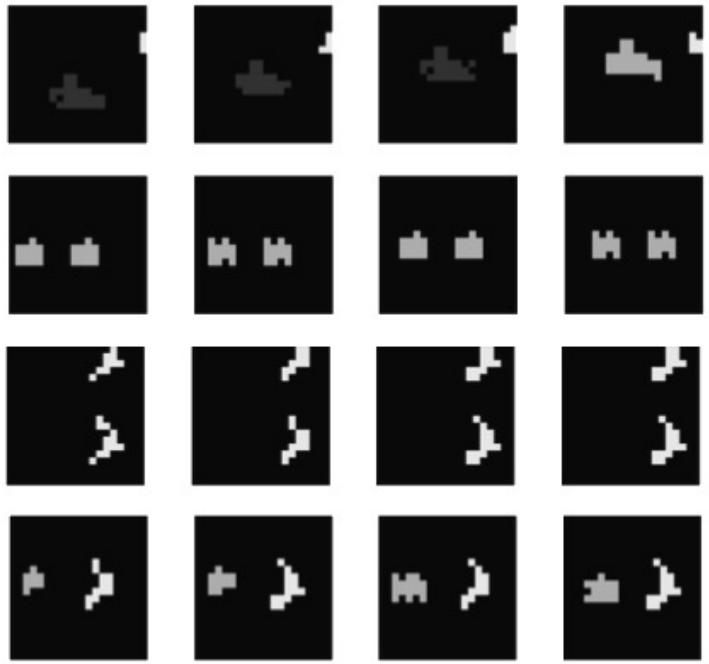


※明るさ，コントラストを見やすく変更した

- 「最適刺激法*」を使って可視化，たたみ込みの受容野に合わせて窓は決められる．1層目の受容野サイズは8x8x4.
- 4つのフィルタは，画面のオブジェクト部分とその変化をとらえている．

*D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical report, University of Montreal, 2009.

特徴の可視化：2層目



※明るさ，コントラストを見やすく変更した

- 同じく「最適刺激法」を使って可視化，受容野サイズは20x20x4.
- 4つのフィルタには，一緒に動く敵や方向に沿って動く潜水艦などをまとめてとらえている.

オブジェクトやその動きをまとめる特徴が確認された

方策の可視化

- 潜水艦を近づけて敵を連続撃破する方策（下図）.
  - 敵を倒す遅延報酬に対する行動が得られた.
  - 一方で大きな報酬が得られるのにダイバーを救わない.

  → ６人のダイバーを拾って水面に戻るのは遅延が長すぎる

※明るさ，コントラストを見やすく変更した

Step 69: FIRE　Step 70: DOWN · FIRE　Step 74:DOWN · FIRE　Step 75:RIGHT · FIRE

方策の可視化

- 潜水艦を近づけて敵を連続撃破する方策（下図）.
  - 敵を倒す遅延報酬に対する行動が得られた.
  - 一方で大きな報酬が得られるのにダイバーを救わない.

  → ６人のダイバーを拾って水面に戻るのは遅延が長すぎる

※明るさ，コントラストを見やすく変更した

Step 75:RIGHT-FIRE　Step 76:RIGHT-FIRE　Step 78: RIGHT-FIRE　Step 79:DOWN-FIRE

目次

結論

- DQNとUCTのギャップを埋めて7つのゲームで勝った.
- 今回の実験では，Q値の回帰より行動の分類の方がよかった.
- UCTの手本と実際の状態分布の違いが問題で，解決できた.

参考文献

- Guo, Xiaoxiao, et al. “Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning.” *Advances in Neural Information Processing Systems*. 2014.
- Mnih, Volodymyr, et al. “Human-level control through deep reinforcement learning.” *Nature* 518.7540 (2015): 529-533.
- D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. Technical report, University of Montreal, 2009.