HOKKAIDO UNIVERSITY

# 講義「人工知能」
# 第9回 Deep Q-Network
## 人よりもビデオゲームが上手なAI

北海道大学大学院情報科学研究院
情報理工学部門 複合情報工学分野
調和系工学研究室 准教授 山下倫央
http://harmo-lab.jp
tomohisa@ist.hokudai.ac.jp
2024年5月7日(木)

# Deep Q-Network(DQN)とは

Deep Mind

- 2010年創業
- 2014年　Googleに買収される
- AlphaGo等を開発

論文　強化学習(Q学習)とDeep Learningを使用して
Atari2600のビデオゲームをプレイする

- Playing Atari with Deep Reinforcement Learning
  Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et,al., arxXiv, 2013
- Human-level control through deep reinforcement learning
  Volodymyr Mnih, Koray Kavukcuoglu, David Silver, et,al., Nature, 2015


Atari2600
1977年に発売
スティックとボタン一つ
レトロゲーム

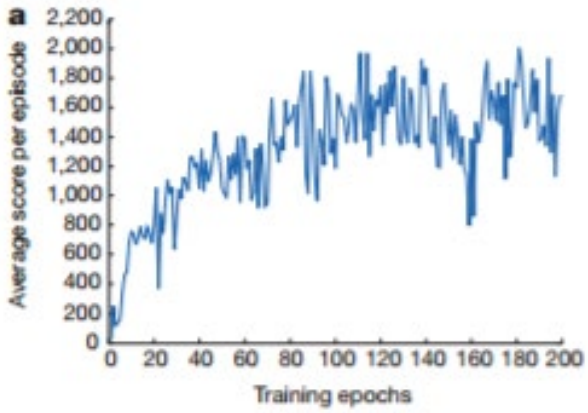
# 使用例(Breakout)


試行錯誤しながら行動を学習
学習が進むと人間レベルのプレイができる

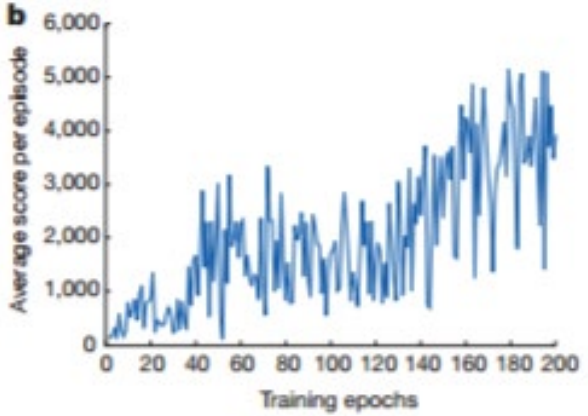
https://www.youtube.com/watch?time_continue=12&v=Q70ulPJW3Gk

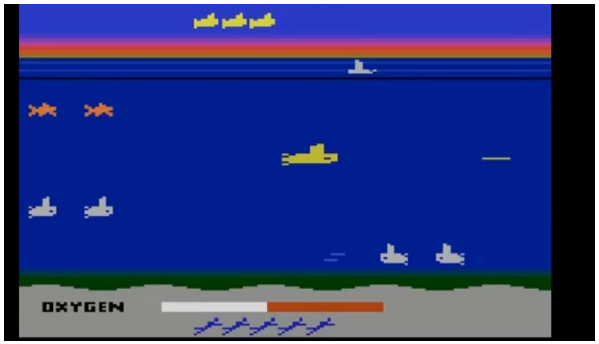
# 学習過程

Space Invaders




Seaquest




１ゲームの平均スコア

学習が進むほどにスコアが高くなっている

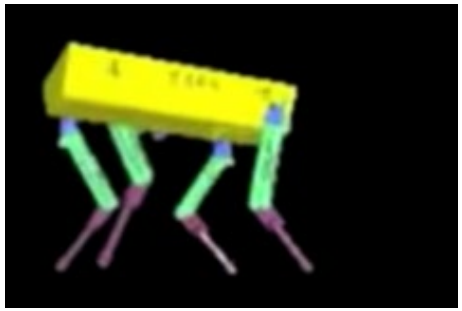
Space Invaders
宇宙人の弾を避けながら打ち落とす

Seaquest
息継ぎをしながらサメや潜水艦を避けて漂流者を救助する

# DQNを使用するとどんなことができそうなのか…?

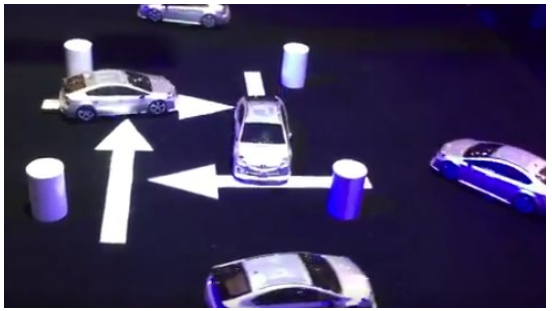
DQNはビデオゲームなど、対象の動作を学習させることができる

ロボット



Deep Mind

自動運転



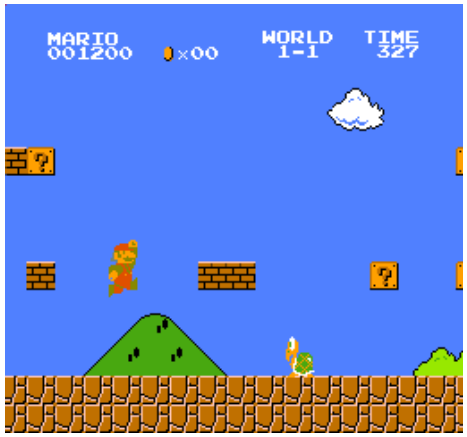
Preferred Networks

ロボットなどの制御対象が状態や環境が観測できるならば
DQNを使用して学習ができる

# 何が新しいのか

遺伝的アルゴリズム

2009~12にYouTubeなどに多く投稿




NNを用いた強化学習(ボードゲーム)

TD-Gammon [Tesauro 1994]

入力:コマの数
コマの位置
コマの移動数

ステージの状態を
入力にしている

- 画面を入力とする
- 同じアルゴリズムでさまざまなゲームに対応している
- 学習を成功させる工夫を導入している
- いくつかのゲームで人間の上級者より高いスコアを記録

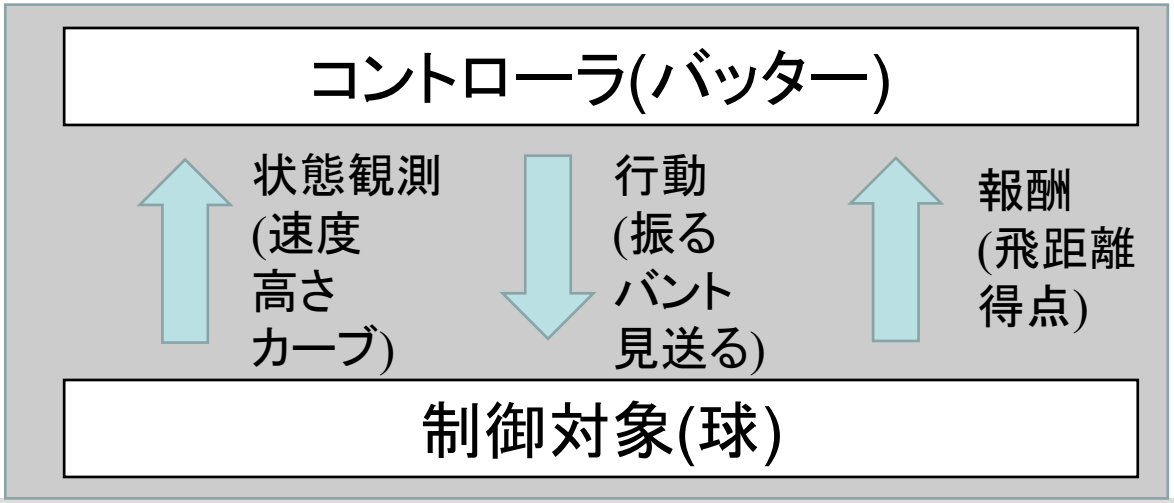
# 強化学習(Reinforcement Learning)

与えられる報酬を手がかりに行動ルールを決定する

(学習初期)空振り

当たるほど報酬を得る

(学習完了)ホームラン！



# Q学習

状態s：速度、位置、スピン

行動a：振る、バント、見送る

報酬r：飛距離、得点

方策π:S→A　状態sの時には行動aをとる：ストレートではバットを振る

行動価値関数：

$$Q^{\pi}_{(s,a)} = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s_t = s, a_t = a, \pi]$$

状態s　行動a

1ステップ後に期待される報酬

Q-table

| 状態 | 行動 | Q |
|---|---|---|
| 150km/h,高さ1m | 振る | 1 |
| 150km/h,高さ1m | 見逃す | 10 |
| 100km/h,高さ1.5m | 振る | 7 |
| 100km/h,高さ1.5m | 見逃す | 4 |

割引率γ:
未来の期待報酬に
対して重み付け(0~1)

# Q学習

行動価値関数:

$$Q^{\pi}_{(s,a)} = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots | s_t = s, a_t = a, \pi]$$

最適方策　期待累積報酬和を最大化する方策π*

最適行動価値観数Q*

Bellman方程式

$$Q^{*}_{(s,a)} = E[r + \gamma \max_{a'} Q^{*}_{(s',a')} | s, a]$$

- $Q^{*}_{(s,a)}$: (s,a)の時の累積期待報酬
- $r$: (s,a)の報酬
- $\max_{a'} Q^{*}_{(s',a')}$: 1ステップ未来の(s',a')の時の累積期待報酬

更新式

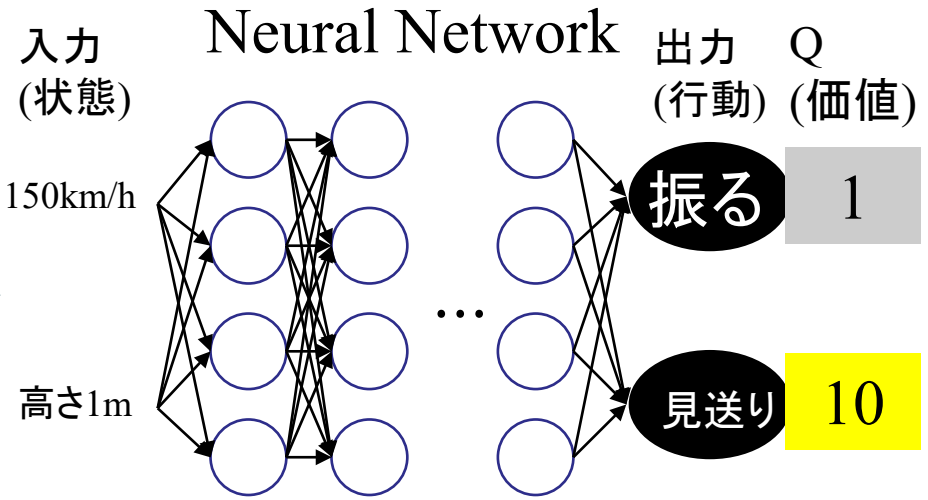
$$Q_{(s,a)} \leftarrow Q_{(s,a)} + \alpha(r + \gamma \max_{a'} Q_{(s',a')} - Q_{(s,a)})$$

# NNを適応したQ学習

ニューラルネットが行動価値Qを出力

Q table

| 状態 | 行動 | Q |
|---|---|---|
| 150km/h,高さ1m | 振る | 1 |
| 150km/h,高さ1m | 見逃す | 10 |
| 100km/h,高さ1.5m | 振る | 7 |
| 100km/h,高さ1.5m | 見逃す | 4 |

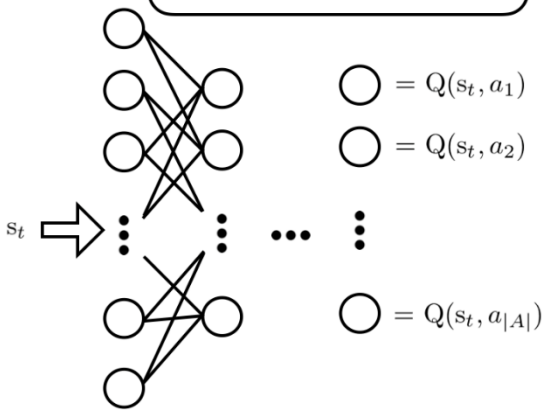



Look Up Table

$$Q = \begin{pmatrix} Q(s_1,a_1) & Q(s_1,a_2) & \dots & Q(s_1,a_{|A|}) \\ Q(s_2,a_1) & Q(s_2,a_2) & \dots & Q(s_2,a_{|A|}) \\ \vdots & \vdots & \ddots & \vdots \\ Q(s_n,a_1) & Q(s_n,a_2) & \dots & Q(s_n,a_{|A|}) \\ \vdots & \vdots & \ddots & \vdots \\ Q(s_{|S|},a_1) & Q(s_{|S|},a_2) & \dots & Q(s_{|S|},a_{|A|}) \end{pmatrix}$$

$s_t = s_n$

DQN



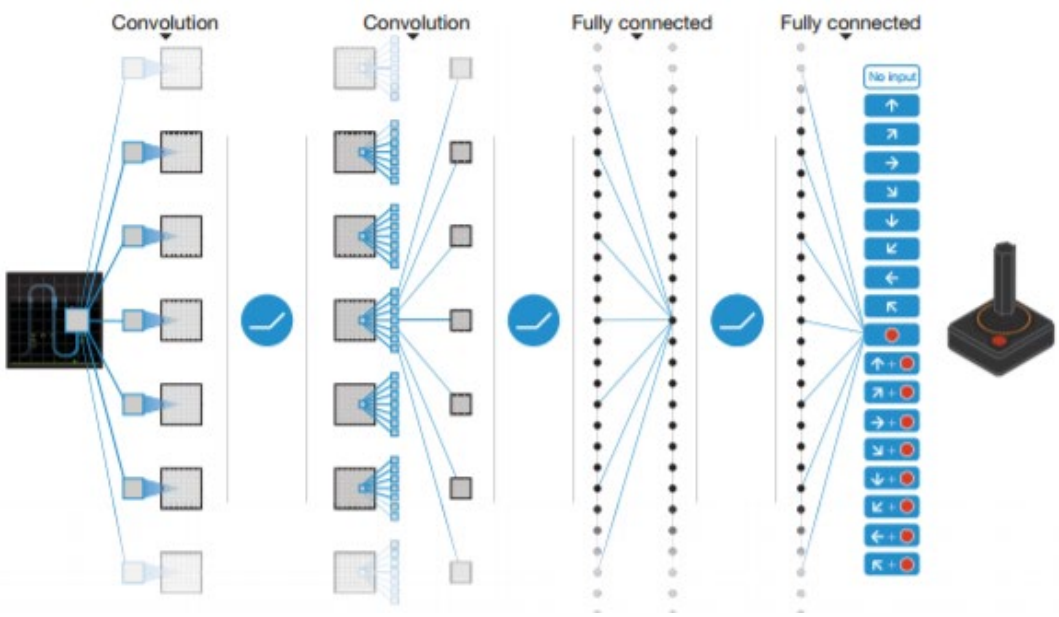

DQNの理論説明
https://www.renom.jp/ja/notebooks/tutorial/reinforcement_learning/DQN-theory/notebook.html

# 論文で実装されたDQN(1/2)

入力：Atariのゲーム画面(84px × 84px ×4frame)

210×160のRGBから84×84の白黒に変換

出力：行動(ゲームによって異なる　例：左右止＋ボタン)




| Layer | Filrer size | Num fikters | Stride | Activation |
|---|---|---|---|---|
| conv1 | 8×8 | 32 | 4 | ReLU |
| conv2 | 4×4 | 64 | 2 | ReLU |
| conv3 | 3×3 | 64 | 1 | ReLU |
| fc | | 512 | | ReLU |
| output | | num actions | | Linear |

バッチサイズ32

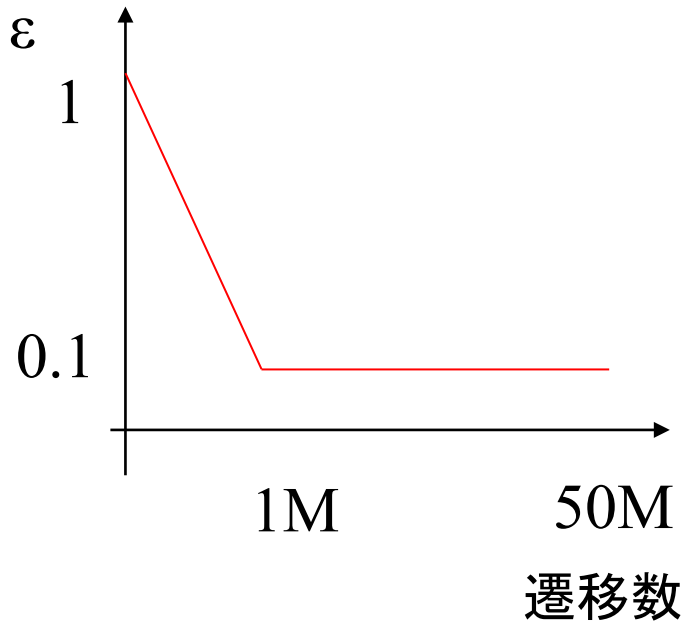
学習回数　100万回

# 論文で実装されたDQN(2/2)

- ε-greedy法
  確率εでランダムな行動を選ぶ

ずっと撃つだけ ⇨ 攻撃ばかり学習

たまに避けてみる ⇨ 避けることも学習

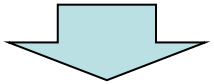



- フレームのスキップ
  行動は4フレームに一度変更する
  60fpsの入力のため計算コストを減らす

# テクニック①　experience replay

学習に使用するデータ

1ゲーム分の状態&行動のセット



そのまま学習

エピソードの内容を強く反映

一定の行動を取る傾向がある

1ステップの遷移ごと保存し、ランダムに抽出する

# 導入テクニック②　target network

$$Q_{(s,a)} \leftarrow Q_{(s,a)} + \alpha(r + \gamma \max_{a'} Q_{(s',a')} - Q_{(s,a)})$$

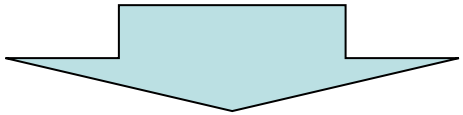
$\max_{a'} Q_{(s',a')}$ → 教師信号

教師信号を出力するNNが何度も更新
学習が安定化しない

教師信号のNN(target network)と
推論のNN(Q network)を分ける
更新に時間差ができることで発散や振動を防ぐ

# 導入テクニック③　error clipping

希少確率の状況を学習するとき、学習の誤差が大きい



誤差　$r + \gamma \max_{a'} Q_{(s',a')} - Q_{(s,a)}$

-1~1にクリップ

大きすぎる更新を抑える働きがある

# 導入テクニック④　reward clipping

さまざまなゲームに対応するため報酬を一定にする

| 負の報酬 | 報酬なし | 正の報酬 |
| --- | --- | --- |
| -1 | 0 | 1 |

例：Seaquest


サメを倒す：＋20点　→+1
人間を救助する：＋150点　→+1
攻撃を受ける：残機が減る　→-1

細かい設計をしなくても学習することができる

# DQN 学習手順



1. 初期化
   - リプレイ用のメモリ *D* の初期化
   - Q-network $Q_\theta$, Target Q-network $Q_{\theta^{target}}$ の初期化

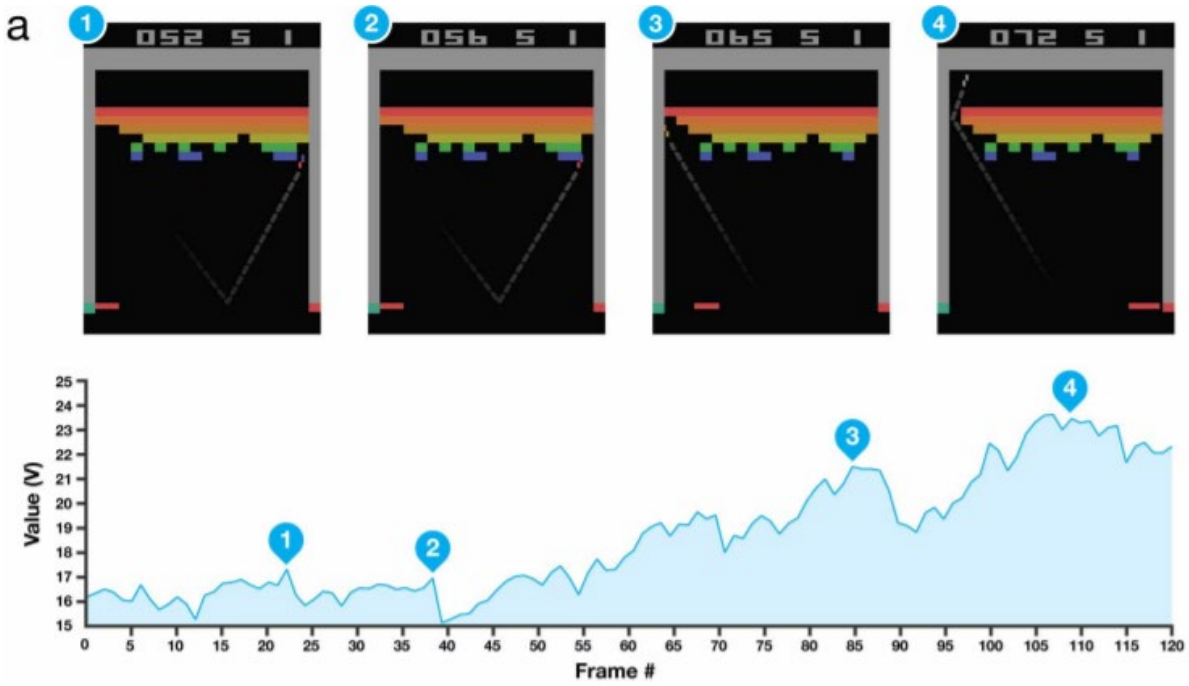
エピソード終了まで 2 , 3 , 4 を繰り返す

2. 行動選択
   - Q-network $Q_\theta$ に状態を入力して、$Q_\theta(s_t, a_t)$ を算出
   - ε-greedy法に従って行動 $a_t$ を選択
   - 行動 $a_t$ を実行
   - データセット $s_t$, $a_t$, $r_t$, $s_{t+1}$, $Q_\theta(s_t, a_t)$ をメモリ D に保存

3. 学習： Q-network $Q_\theta$ の更新
   - メモリDからランダムにデータセット $s_t$, $a_t$, $r_t$, $s_{t+1}$, $Q_\theta(s_t, a_t)$ を選択
   - Q-network $Q_\theta$ の重み $\theta$ を更新
     - 誤差： $r_{t+1} + \gamma \max_{a'} Q_{\theta^{target}}(s_{t+1}, a') - Q_\theta(s_t, a_t)$ による勾配降下法を実施

4. 学習：Target Q-network $Q_{\theta^{target}}$ の更新
   - 設定した試行ステップごとに Target-Network の重みを更新
     - Target Q-network の重みを Q-networkと同期

# 行動価値の変化（Breakout）



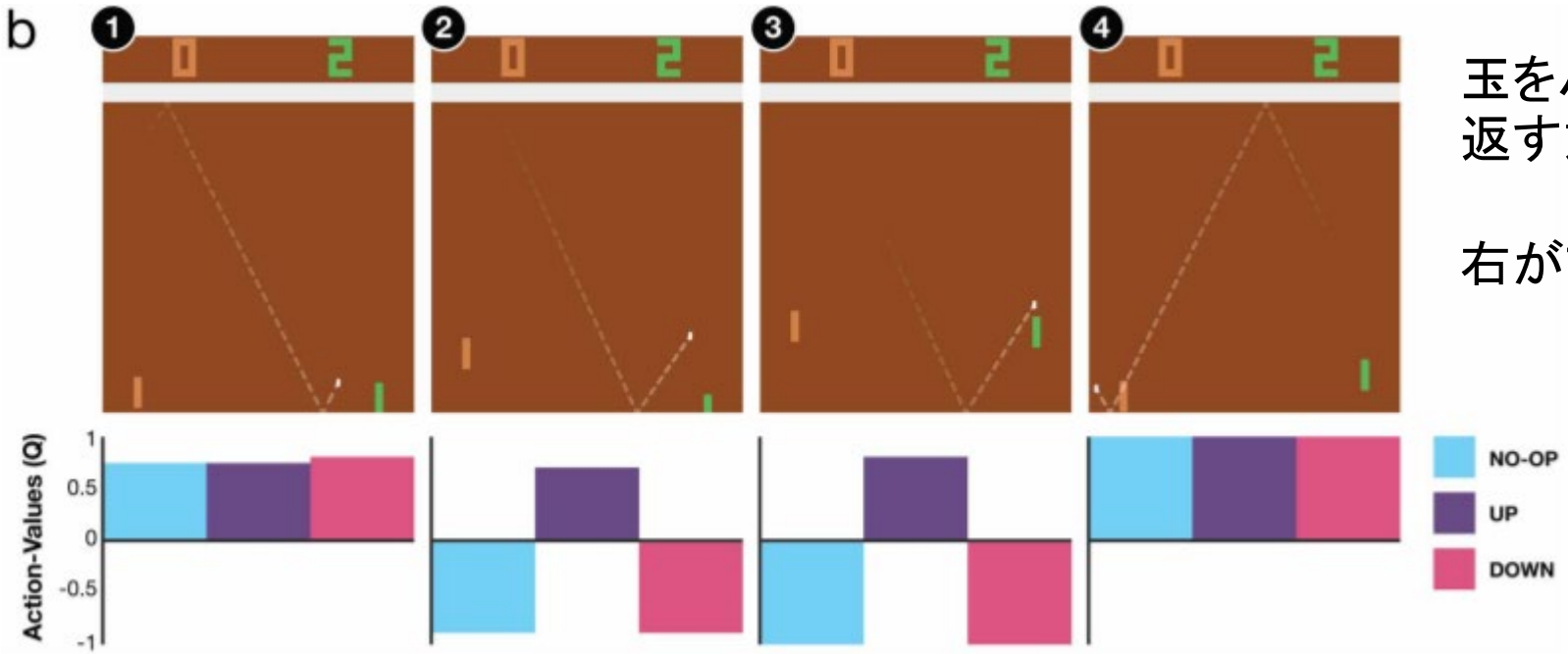

ブロックを崩すゲーム

玉をバーで弾いて画面上方のブロックを消していく

ブロックの塊よりも上に玉が入ると、操作をしなくとも多くのブロックを壊せる

①②得点を得る直前に行動価値が大きくなっている
③④連続して得点を得られそうな時は特に行動価値が大きい

# 行動ごとの行動価値の違い(Pong)




玉をバーで打ち返す対戦ゲーム

右がプレイヤー

①④どの行動をとってもほぼ変わらない
②③上に移動しなければ失点してしまう

# 学習実験

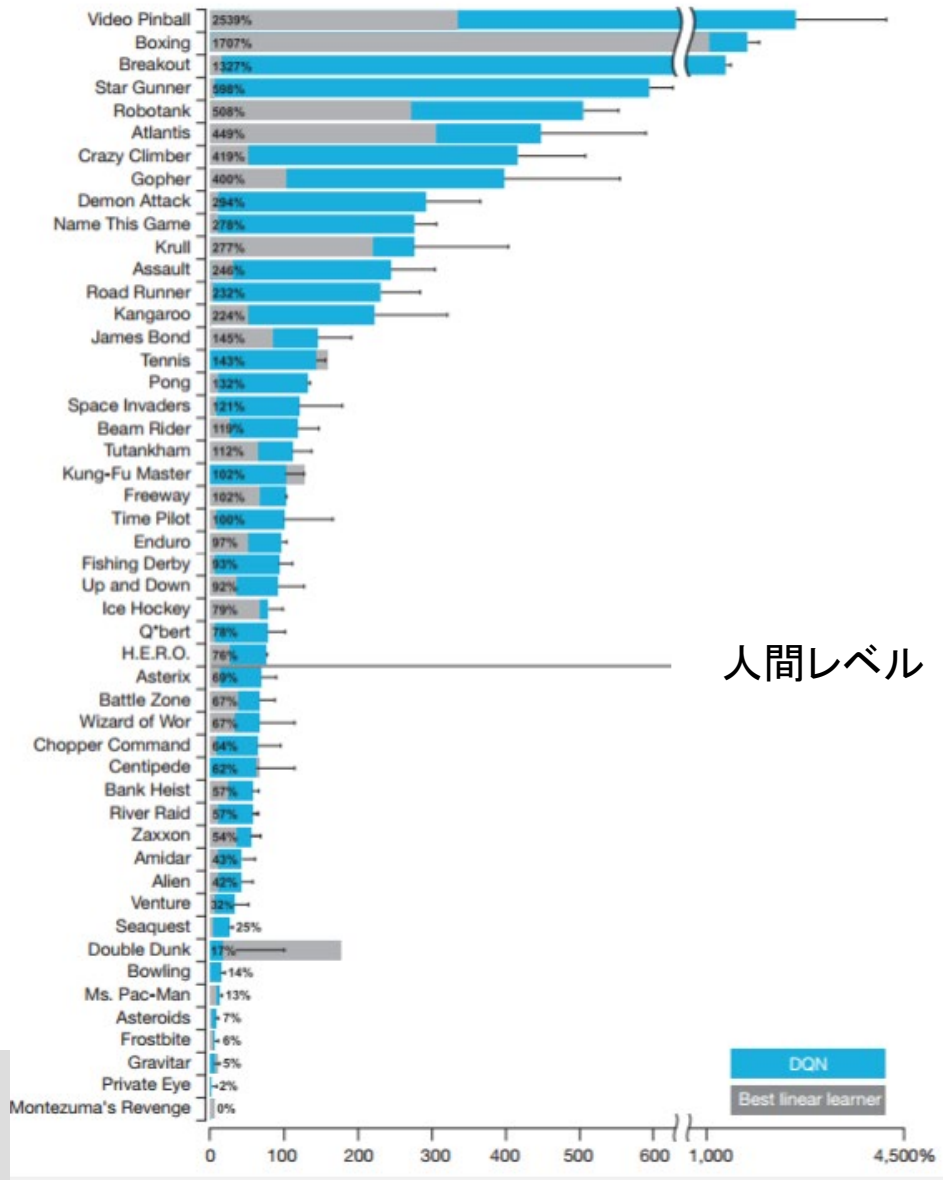
実験設定

- Atari2600のゲームのうち49のゲームで学習を行う
- 上級者のスコアの75%以上のスコアを人間レベルとする
- DQN,Linear Learner,上級者,ランダムでスコアを比較

学習

- 32行動　100万回学習
- すべての学習に38日間必要

# 最終結果




人間レベル

横軸：上級者のスコアと比べた率(%)

青：DQNのスコア

灰：人がコーディングしたプログラムのスコア

100%：上級者のスコア

0%：ランダムなプレイでのスコア

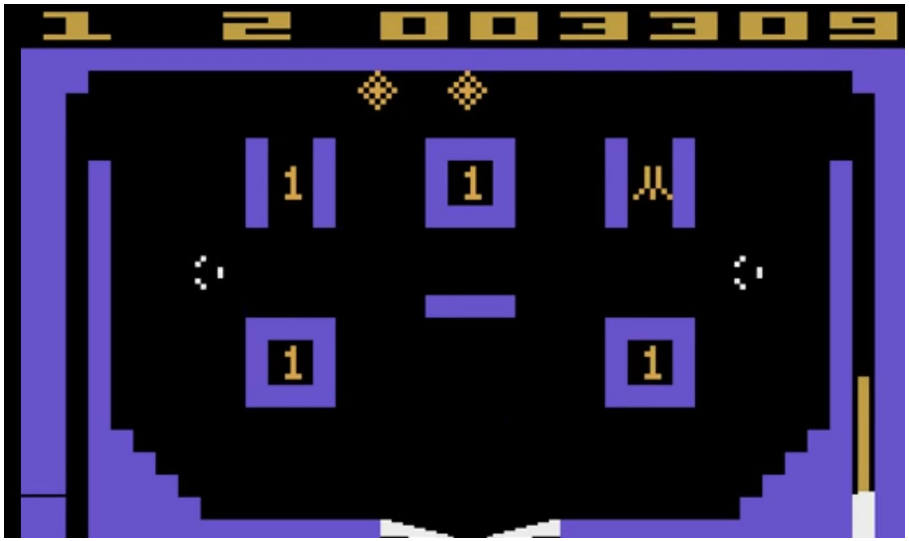
100回の平均を使用

49のゲームのうち29ゲームで
上級者の75%以上のスコアを獲得

# 人間の操作よりも大きくスコアを記録したもの

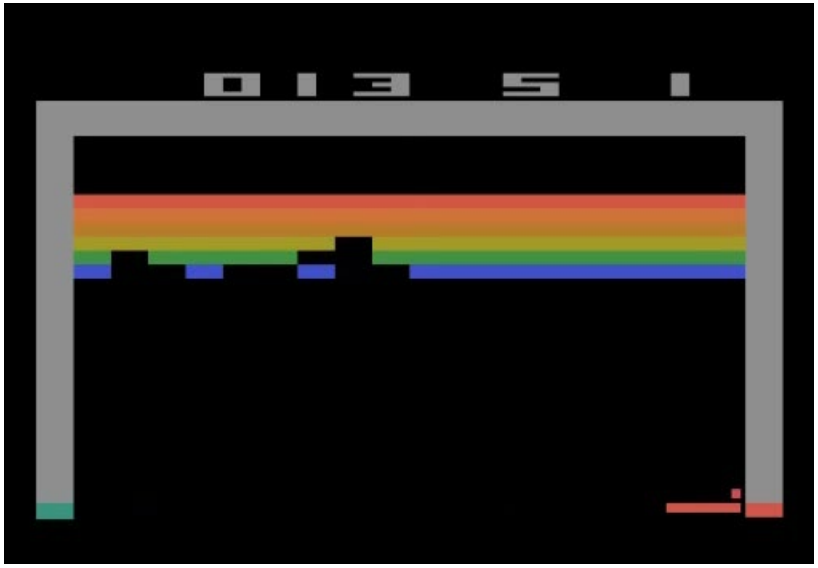
Video pinball

2539%




Breakout

1327%

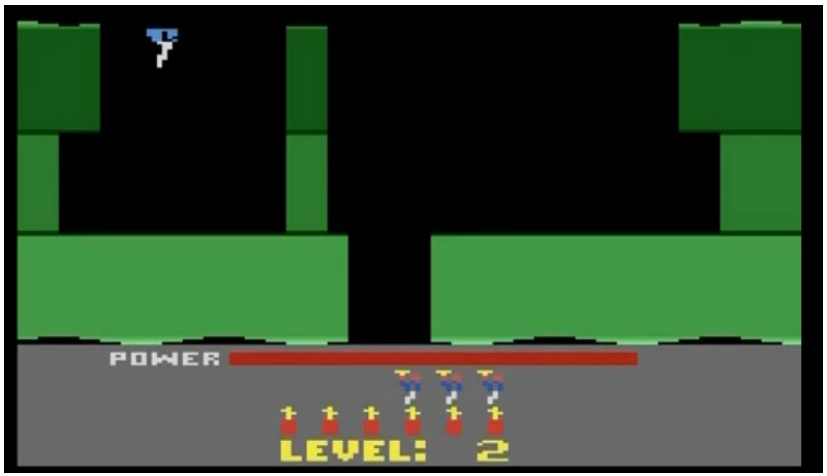



「玉を落とさない」のような単純なルール

# ほぼ人間レベルのもの

H.E.R.O.

76%




Battle Zone

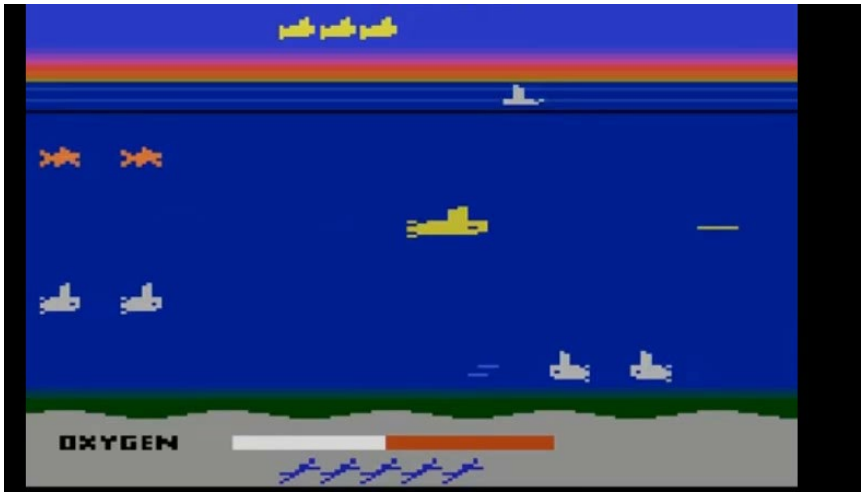
67%


画面が切り替わったり、奥行のあるゲーム

# 人間以下のスコア

Seaquest

25%




Private Eye

2%


立体的なマップであったり、必要な行動が多いゲーム

# DQNよりもほかのアルゴリズムの方がスコアが高い

Double Dunk

DQN 17%

Linear Learner 177%


報酬が入りにくく、２人の選手を操作しなければならないゲーム

# まとめ

- ビデオゲームの画面を入力としたNNがゲームプレイを学習した
- NNを用いてQ学習行う際、学習を成功させる工夫を導入した
- 49ゲームのうち29ゲームで人間レベル以上のスコアを獲得した。