3C-04

# 深層強化学習のための環境シミュレーションと自律制御ロボットの連携

宮島 優太郎 †　　李 天埼 ‡　　柴田 千尋 †　　田胡 和哉 †

東京工科大学コンピュータサイエンス学部 ‡

## 1 はじめに

近年，人工知能の分野において，Deep Learning と呼ばれる手法が大きな注目を浴びている．Deep Learning とは，多数のパラメータと層構造をもったニューラルネットワークを用いた機械学習の一手法である．本稿では，Deep $Q$-Network(DQN) とよばれる手法を用いて，ロボットの行動学習を行い，その有用性の検証を行う．

## 2 先行研究

Mnih らの DQN の手法 [1] では，強化学習の位置手法である $Q$-Learning と、Deep Neural Network(DNN) を組み合わせ，観測されるゲームの画像データから最適な行動選択を行う．環境の状態 $s$ と，行動 $a$ から求まる $Q$ 関数 $Q(s,a)$ を，DNN を用いて関数近似を行う．また，NN のモデルに Convolutional Neural Network (CNN) を利用する事で，高次元な生の画像データの学習を可能にしている．更に，一度経験した環境のデータ s をサンプルとして記憶し，繰り返し利用する Experience Reply という手法を取り入れている．本研究ではこの学習手法を，シミュレータ及びロボットの行動学習に応用する．

## 3 DQN シミュレータ及びロボットの制御

本研究では，現実の環境においてロボットが自律的に障害物を避けながら前進できることを目指す．DQN シミュレータ (図 1 左) では，カメラビューによってロボット全体と周囲の状況がわかるように撮影された画像と，前方 180 度の距離データを報酬として用いて事前学習を行う．一方，リアルな環境で動くロボット (図 1 右) では，機体に取り付けられた Android 端末から取得したカメラ画像をサーバに送信する．受信したサーバでは，DQN シミュレータを用いて行った事前学習から，ロボットの次の動作を選択し，ロボットに対して送信する．このシステムを用いて，環境シミュレータを通してどの程度適応させることができるか評価を行う．

### 3.1 Deep Q-Learning シミュレータの構成

ロボットの行動学習に DQN を応用するにあたって，学習コストの削減のため，予め DQN シミュレータ (図 2) を構築し，その上で実験を行った．本シミュレータでは，より現実に近い環境を再現するため，WebGL を用いて 3D グラフィックスで

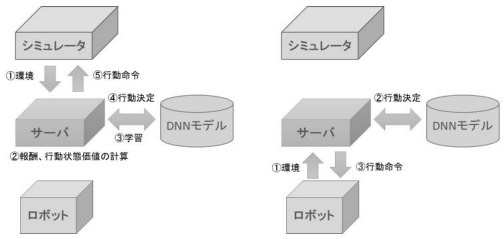


図 1: 左:シミュレータでの事前学習 右:ロボットの動作制御

実装した．シミュレータ上では，ロボットが行動を選択した結果，得られた環境データを逐次サーバに送信する．サーバサイドは，Deep Learning framwork である chainer [2] を用いて実装されており，受信したサンプルデータを元にオンライン学習及び行動選択を行う．そこで選択された行動をシミュレータにフィードバックする事で，次の環境へ遷移する仕組みである．また，本シミュレータでも，Mnih らの手法と同じく Experience Reply のアルゴリズムを実装しており，オンライン学習でありながら，教師ありのバッチ学習に近い環境で学習を行っている．

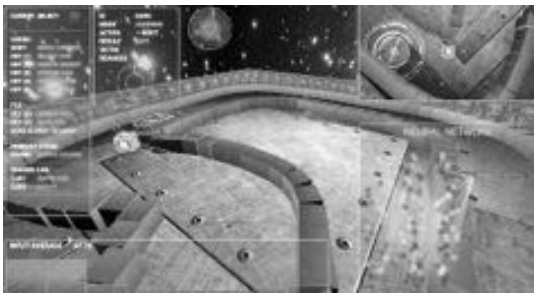

図 2: DQN シミュレータの動作画像

#### 3.1.1 画像による学習

本シミュレータでは，環境の状態 $s$ として，ロボットの頭上から見下ろしたカメラビュー (96×128) を用いる．本来高次元である画像データに対し，前処理としてグレースケール化を行い，RGB の 3 次元の画像を 1 次元に下げる．これにより，障害物や環境の色による影響を可能な限りなくし，また，色の強弱に左右されないよう，エッジ検出を行う．これを CNN の入力とし，畳込み及びプーリング変換を施す事で，行動状態価値 Q を求める．

---

Cooperation between Autonomous Robots and an Environmental SimulationSystem for Deep Reinforcement Learning

†Yutaro MIYAJIMA　‡LI Tenqi †Chihiro SHIBATA　†Kazuya TAGO

‡Tokyo University of Technology Graduate school.

†miyajima126@gmail.com　‡toukyouniversityoftechnology@gmail.com

#### 3.1.2 報酬の仕組み

本シミュレータでは，ロボットの前方 180 度に超音波センサーが取り付けられている事を想定し，そこから得られた距離データの平均値を報酬として利用する．この距離平均を 0 ~ 1 の範囲で変換し，また，ロボットが障害物に衝突した場合は，−1 の負の報酬を与える事で，報酬の範囲を −1 ~ 1 に正規化している．

#### 3.1.3 DQN の構成

本シミュレータでは，(表 1) の CNN を，状態行動価値 ($Q$ 値) を求めるための近似関数 ($\tilde{Q}(s,a)$) とする．

表 1: ネットワークの構成

| layer | patch | stride | padding | output | func |
|---|---|---|---|---|---|
| data | - | - | - | 96×128×1 | - |
| conv1 | 8×8 | 4 | 2 | 24×32×16 | - |
| BN1* | - | - | - | 24×32×16 | ReLu |
| conv2 | 4×4 | 2 | 1 | 12×16×32 | - |
| BN2* | - | - | - | 12×16×32 | ReLu |
| fc1 | - | - | - | 1×256 | ReLu |
| fc2 | - | - | - | 1×4 | ReLu |

この関数 $\tilde{Q}$ に画像データ $s$ を入力し，求まった $Q$ 値の近似値を利用して，ネットワークのパラメータを更新する．この時の更新式は，学習率 $\alpha = 0.001$, 割引率 $\gamma = 0.7$ として，以下の式を用いて，教師データ $t$ を求める．

$$t = reward + \gamma * \max_{a} \tilde{Q}(s', a) \quad (1)$$

ここで $s'$ は，1 ステップ後の画像データをあらわす．この教師データ $t$ と $\tilde{Q}(s,a)$ を差分として，勾配計算を行い，誤差逆伝搬を用いて重みの更新を行う．また，この時 Experience Reply で利用するバッチの大きさは，1 バッチ = 100 サンプルとする．関数 $\tilde{Q}$ の出力を元に行動を選択する際は，$\epsilon$-greedy 法を利用し，ランダム係数 $\epsilon$ を，学習中では 1.0 に，テストでは 0.05 に設定している．

### 3.2 ロボット

本システムに使用するロボット (図 3) には，Arduino を用いて駆動用モータを制御し，機体に取り付けられた Android 端末に内臓されているカメラを利用して前方の風景を取得する．取得したデータは，Android からサーバに対して送信さる．また，Android はサーバからロボットの行動命令（前進，後退，左旋回，右旋回）を受信し，そのデータによってロボットが動作する．

## 4 評価実験

前述した DQN シミュレータとロボットを組み合わせて，動作の検証を行う．DQN シミュレータでは２つの学習を行う．

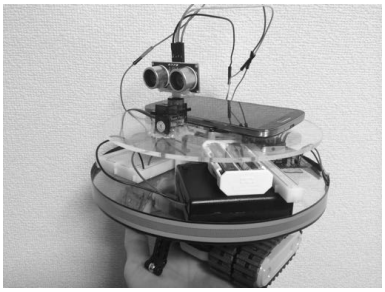

図 3: 作成したロボット

一つ目は，ロボットの頭上から撮影した画像と，ロボットから前方 180 度の距離を報酬として学習である．二つ目は，ロボットから前方 180 度の距離を用いた行動学習を行った．どちらの方法を用いても，少なくともシミュレータ上においては，衝突しないような行動を学習できることがが確認できている *．

## 5 現状の問題点と解決策

本稿では，DQN とロボットの学習を行う上で，あらかじめシミュレータ上で学習を行うことによって，ロボットの行動学習にかかる時間を大幅に削減することができた．しかし，シミュレータ上で取得される情報と，実際に動作するロボットから取得されるデータに差異があり，正しく動作することが困難である．この問題の解決策として，現実の環境をシミュレータ内の環境に近づけることや，センサの精度を向上させるなどの工夫が必要である．

## 6 まとめと今後の課題

本研究では，DQN を用いた自律制御ロボットの開発を行い，動作の検証を行った．シミュレータを用いて学習を行うことによって，多少の誤差はあるものの，実際にロボットを動かせることが確認できた．今後は，あらゆる状況に対応できるように様々な状況で学習を行うことによって，どのような状況においても自律的に判断し，対応することが課題として挙げられる．

## 参考文献

[1] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M.: Playing Atari With Deep Reinforcement Learning, *NIPS Deep Learning Workshop* (2013).

[2] Tokui, S., Oono, K., Hido, S. and Clayton, J.: Chainer: a Next-Generation Open Source Framework for Deep Learning, *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)* (2015).

*BN は Batch Normalization の略

*動作の様子：https://github.com/myamam/DQNRobot