ディープラーニングを活用したロボット制御における安定性向上の研究

機械システム科　　科　　長　　**堀　江　貴　雄**

近年、ディープラーニングの活用は画像判別などにとどまらず、ロボット制御等などにも応用されている。しかしながら、ニューラルネットによる直接的な制御手法では、特定条件で誤動作することが問題とされている。そこで､本研究ではアーム搭載のメカナム移動ロボットを課題として､安定して自律制御することを目的とした。アーム制御と移動制御を組み合わせた複雑な制御を、end-to-end 学習で安定的に制御することを目標とする。これまでに実験用のアーム試作と、データセット設計、同時制御用ネットワークの設計と学習を行った。今年度はニューラルネットワークモデルの推定精度向上のため、入力画像の高精細化、入力データの拡充を行い、開発ツールとして PyTorch を用いてモデルの改良、学習、実装を行った。また走行実験によってこのモデルの有効性を確認した。

１．緒言

近年では深層学習は、機械制御への応用事例も多く報告されるようになった。そのひとつに自動走行の研究がある。

これまでにロボットの屋内の自律移動を目的に、筆者は、カラー画像、深度画像、方位データ、目的地番号、ルート選択番号の各５時刻分を入力すると、車両速度、平行移動方向、回転速度を出力するニューラルネットワークを設計した。さらに未学習環境を検出し、安全に停止させるために、カラー画像から深度画像を推定する Autoencoder を設計し、一定の閾値によって未学習環境を検出して強制停止させる手法も確立した[1-3]。これに加えてエレベータを用いたフロア階間移動を実現するため、エレベータボタンを操作可能なアームを試作し、車両とアームの制御パラメータを同時推定可能なネットワークを実現した[4]。本稿ではこのネットワークをさらに改良するため、複数のアーキテクチャを適用したネットワークを試作、性能比較を行い最適なネットワークを模索した。

２．システム構成

２．１　ハンドアイ搭載全方向移動ロボット

制御対象のロボットは、エレベータボタンを操作するための４自由度アームのハンドアイを上部に搭載している（図１）。また全方向移動のためにメカナムホイールを有している（図２）。このロボットの全体像を図３に示す。

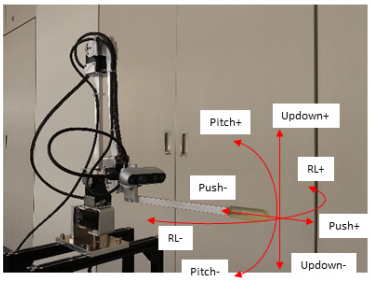

図１　ハンドアイ

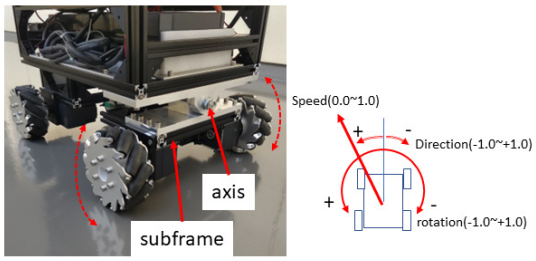

図２　メカナム駆動

アーム制御用のカラー画像、深度画像撮影用に intel 社 RealSenseD435、車両制御用のカラー画像、深度画像撮影用に Intel 製 RealSenseD455 センサ、磁気センサとして WitMotion 製 HWT905、モータ同期制御用のマイコンボードに Arduino Uno、制御プログラム実行用に ITX 規格の PC を搭載した。なお、昨年度開発モデルよりも規模の大きなニューラルネットを用いた推論を６～７Hz で実行するために､用いる GPU を NVIDIA 製 T600(4GB)から RTX A1000 (8GB)に交換した。

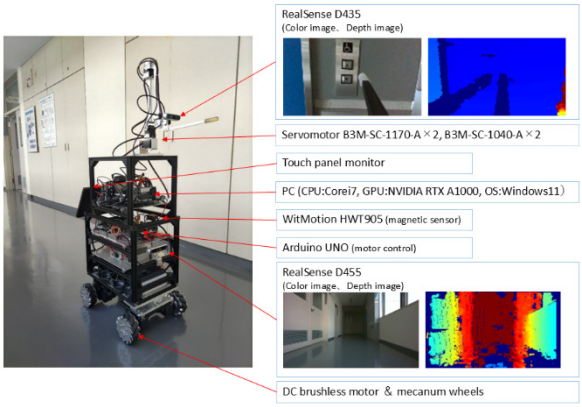

図３　ハンドアイ搭載全方向移動ロボット

２．２　操作系

台車とアームの協調動作操作を手動で行い、データセットを生成するため、ゲームパッドを使用する。ゲームパッド入力と対応する制御パラメータについて表１に示す。

表１　ゲームパッド入力と制御パラメータの割り当て関係

	右トリガーoff (車両制御モード)	右トリガーon (アーム制御モード)
左スティック 上下	前後平行移動 (directionY)と 車両速度(speedY)	アーム上下(updown)
左スティック 左右	左右平行移動 (directionX)と 車両速度(speedX)	アーム左右回転(RL)
左十字キー (上、左、右)	なし	アーム基本姿勢(pos) (正面、左、右)
右スティック 上下	なし	アーム押し引き(push)
右スティック 左右	車両回転方向と回転速度(rotation)	なし
右ショルダー	なし	アーム俯角増(angle-)
左ショルダー	なし	アーム仰角増(angle+)
A ボタン	強制データ記録	強制データ記録
X ボタン	タスク修了	なし

使用するゲームパッドには多数のボタンが用意されており、特に２つのスティックは操作性の観点から有用であるが、台車とアームを同時制御するには自由度が足りない。右トリガーボタンを、「車両・アーム切り替え」とし、２つのスティックに割り当てるパラメータを切り替えることで、解決した。

３．ネットワーク

３．１　データセット

屋内実環境でフロア間を移動させることを目的に、ロボットアームとメカナム台車の同時制御を実現するため、車両カラー画像、車両深度画像、アームカラー画像、アーム深度画像、方位センサデータ、ゴール番号、ルート番号から動作を推定するネットワークを学習させるデータセットを設計した（表２）。

今年度は入力情報拡大による推定精度の向上を期待し、入力画像サイズを昨年度の 3×45×80 から 3×90×160 に変更し、推定用パラメータにスタート階番号、スタート位置番号、ゴール階番号を追加した。

表２　車両、アーム同時制御用データセット

Input	Output
・車両カラー画像(3×90×160)×5(5 時刻分) ・車両深度画像(3×90×160)×5 ・アームカラー画像(3×90×160)×5 ・アーム深度画像(3×90×160)×5 ・磁気センサ X(1)×5 ・磁気センサ Y(1)×5 ・スタート階番号(10) ・スタート部屋番号(10) ・ゴール階番号(10) ・ゴール部屋番号(10) ・ルート選択番号(1)	・車両速度 speed(1) ・車両平行移動方向 direction(1) ・車両回転速度 rotation(-1～1) ・アーム push(1) ・アーム RL(1) ・アーム up and down(1) ・アーム angle(1) ・アーム基本姿勢 arm position(4) (0:変更なし, 1:正面, 2:左 90 度, 3:右 90 度) ・車両、アーム制御選択 mode(3) (0:車両, 1:アーム, 2:タスク終了)

３．２　ネットワーク設計

Otsubo らは、小型２輪移動ロボットによる模型道路内の走行を対象に、運転者が得る視覚情報とそのときの運転操作の関係を畳み込みオートエンコーダ等複数ネットワークを用いて学習するモデルを提案している[5]。学習済みモデルは模型の道路環境下を複数ルートで自動走行可能としており、この手法の有効性を示した。

筆者はこれまでの研究[1-4]で、５時刻分のカラー画像、深度画像と方位センサデータ、ゴール番号、ルート番号から車両制御用パラメータを推定するニューラルネットワークを実現している。このネットワークをベースに改良を実施した。

カラー画像特徴抽出のために設計した Image Convolution Unit は、ResNet18 を参考に、チャンネル数を削減するかわりに ResNet ブロックを増やし、Attention 機構の一種である SE ブロック[7]を追加した（図４）。

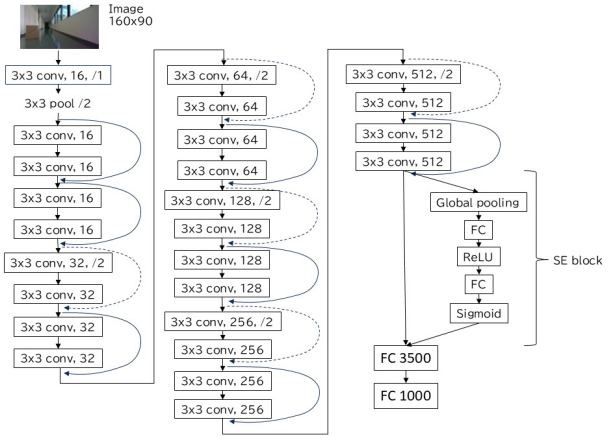

図４　Image Convolution Unit

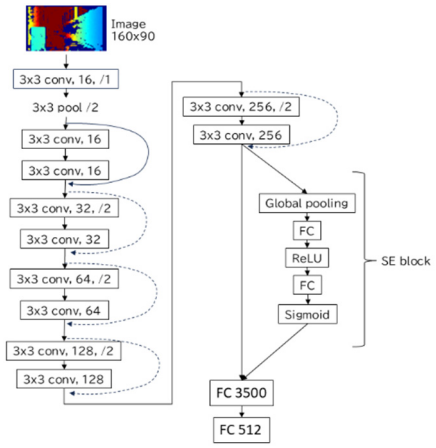

図５　Depth Image Convolution Unit

同様に深度画像特徴抽出のために設計された Depth Image Convolution Unit についても、ResNet ブロック５つと SE ブロック１つを用いたユニットに改良した（図５）。

過去に報告されたネットワークでは、これらユニットの計算結果を RNN ブロック[8]で集約したのち、次時刻の RNN ブロックに送る処理を時刻データごとに繰りかえし、車両速度、車両平行移動方向、車両回転速度をそれぞれ推定した。ここに時間方向の Skip 接続（Temporal Skip Connection）を導入する。設計した車両制御パラメータ推定用ネットワーク（Vehicle Control Unit）を図６に示す。

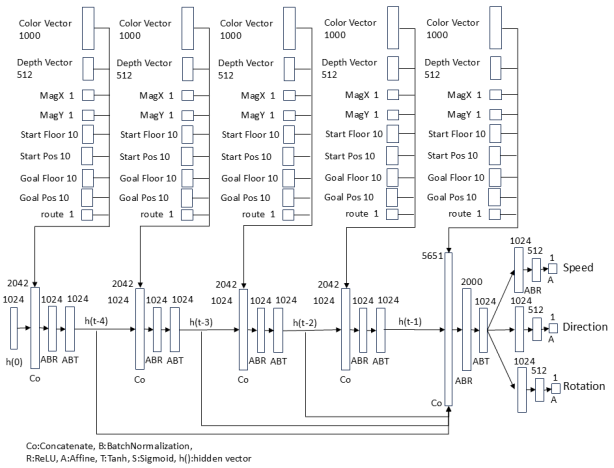

図６　Vehicle Control Unit

カラー画像を $I_{c,t}$、深度画像を $I_{d,t}$、磁気センサデータを $M_{x,t}$、$M_{y,t}$、その他数値データ（フロア番号、スタート番号など）を S とおくと、Image Convolution Unit(ICU)の出力ベクトル $F_{c,t}$ は以下の様に表現できる。

$$F_{c,t} = ICU(I_{c,t})$$

また Depth Image Convolution Unit(DCU)出力 $F_{d,t}$ は以下のように表現される。

$$F_{d,t} = DCU(I_{d,t})$$

このとき各時刻の入力特徴ベクトル X_t は以下のようにまとめられる。

$$X_t = [F_{c,t}, F_{d,t}, M_{x,t}, M_{y,t}, S]$$

過去４時刻分の隠れ状態 h_{t-1}、h_{t-2}、h_{t-3}、h_{t-4} をスキップ接続として用いて最新時刻のベクトル h_t は以下のように求める。

$$h_t = RNN(X_t, [h_{t-1}, h_{t-2}, h_{t-3}, h_{t-4}])$$

最終的な車両制御パラメータ３つは以下のとおり表現できる。

$$[y^{speed}, y^{direction}, y^{rotation}] = f_{vehicle}(h_t)$$

同様にアーム制御パラメータを推定するために、Vehicle Control Unit ベースで、最終出力のみ３つから４つに変更したネットワーク（Arm Control Unit）を設計した（図７）。

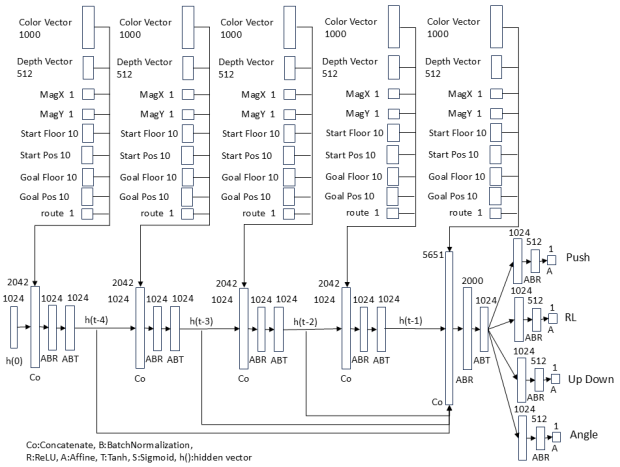

図７　Arm Control Unit

Arm Control Unit の入力値は Vehicle Control Unit と同様に、アームカラー画像、アーム深度画像、磁気センサ値、地点番号、ルート番号とした。

カラー画像の処理は Vehicle Control Unit と同じ設計の Image Convolution Unit、深度画像には Depth Image Convolution Unit を用いた。

これまでの研究から、アームの基本姿勢推定、車両とアームの制御モード選択推定は、Vehicle Control Unit および Arm Control Unit の各時刻ベクトルとは別に計算することが有効である。

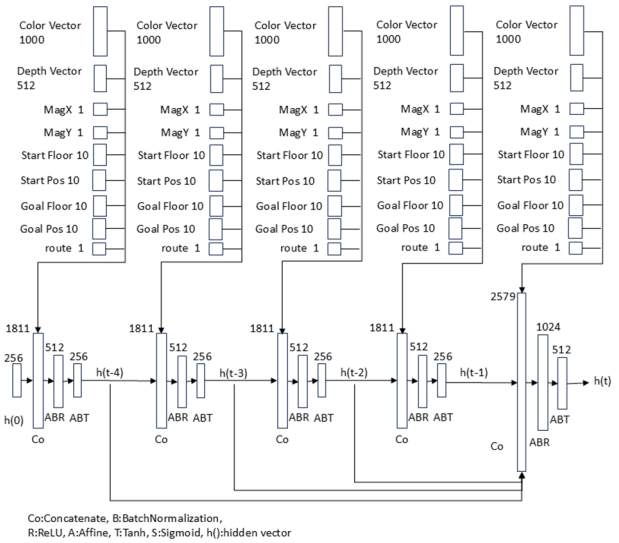

図８　Situation Unit

そこで Image Convolution Unit の出力ベクトルと、磁気センサ値、ゴール番号、ルート番号をそれぞれ５時刻分使用しベクトルを得るネットワーク（Situation Unit）を設計した（図８）。

このネットワークでも最新時刻 t における隠れ状態 h_t を推定する際、通常の RNN 入力(X_t , h_{t-1}) に加え、過去の隠れ状態 h_{t-2},h_{t-3},h_{t-4} を入力として結合し、時間方向の Skip 接続（Temporal Skip Connection）を導入している。

このネットワークで得られた車両状態ベクトル hv とアーム状態ベクトル ha を結合し、アームの基本ポジション（Arm Position）推定、車両とアームの制御モード（mode）推定をおこなう。このネットワーク（PM Select Unit）を図９に示す。

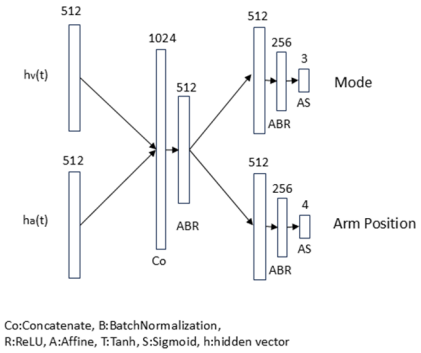

図９　PM Select Unit

また、未学習環境における安全停止機能のために開発した PTD Autoencoder[1]を追加した。

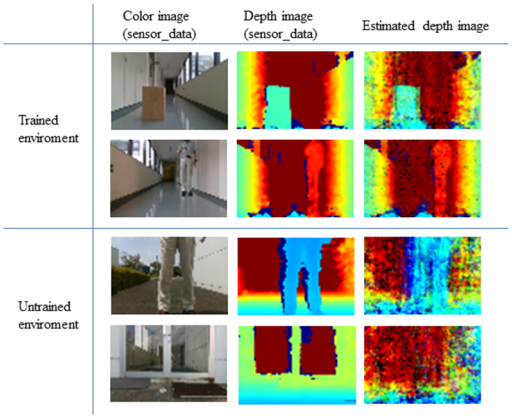

図１０　PTD Autoencoder の出力事例

これは RealSenseD455 のカラー画像から深度画像を生成する Autoencoder であり、生成した推定深度画像 D’ と D455 センサで取得した深度画像 D のピクセル値差分の合計 E を出力する。

$$E = \sum_{i,j} |D_{ij} - D'_{ij}|$$

E を評価し、閾値以上の場合にはロボットの制御値をすべて０として強制停止する。

PTD Autoencoderで深度画像を推定した事例を図１０に示す。

一番左は D455 センサで取得したカラー画像で、中央は深度画像である。右側はPTD Autoencoderがカラー画像から推定した深度画像である。１行目と２行目は学習済み環境での結果を示し、３行目、４行目は未学習の環境での結果を示している。学習済み環境での深度画像はうまく推定できている一方で、未学習環境では深度画像推定が不十分であることが確認できる。

これら Image Convolution Unit、Depth Image Convolution Unit、Vehicle Control Unit、Arm Control Unit 、 Situation Unit 、 PM Select Unit 、 PTD Autoencoderを適用して設計した車両、アームの統合ニューラルネットワークを図１１に示す。

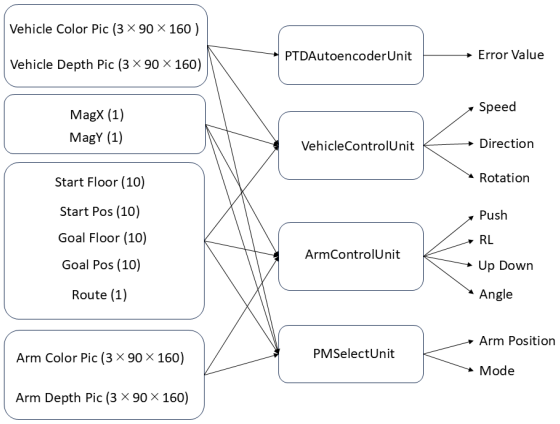

図１１　統合ニューラルネットワーク

設計したニューラルネットはPyTorchを用いてコーディングし、Pythonサーバー上で実行させ、タッチパネルを含むインタフェースと、RealSense センサ以外のデバイスは、C#クライアントで制御するようロボット内蔵PCに実装した。なお、制御周期は６～７Hz を維持するよう調整した。

４．ネットワーク評価

４．１　データセット収集と学習

３階建て屋内６地点に０～５の番号を割り振り、ゲームパッドの手動操作によって各地点から他の地点への移動を網羅的に行い、データセットを収集する。地点０は２階の事務室中央、１、２は２階西棟の廊下の壁側、３は２階東棟廊下の壁側、４は３階廊下の壁側、５は１階ロビーに設定した（図１２）。地点０、１、２、３と地点４、５間の階層間移動ではエレベータを使用するため、ボタン操作も行う。

ゲームパッド操作による手動操作により、各地点を往復するデータセットを合計 11 万 790 個（約５時間に相当）収集し、このデータセットを、Training 用 74790 個、Validation 用 24930 個、Test 用 11070 個に分割した。

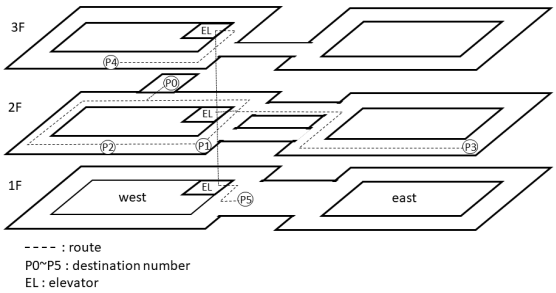

図１２　走行ルート

ネットワークの改良が有効であるかを、昨年度開発のモデル(RNN+Skip2023:データセット約 28 万個)と比較する。

学習はNVIDIA製TITAN RTXを搭載したPCを用いて、バッチサイズ 30、最適化アルゴリズム Adam を使用し 100epoch 実施した。

表３　各モデルの車両制御パラメータ決定係数(R^2)

model	Speed	Direction	Rotation
RNN + Skip (2023)	0.9609	0.757	0.7686
RNN+Skip (2024)	0.9600	0.7519	0.7053

表４　各モデルのアーム制御パラメータ決定係数(R^2)

model	Push	RL	UpDown	Angle
RNN + Skip (2023)	0.6961	0.644	0.5844	N/A
RNN + Skip (2024)	0.8226	0.8452	0.7902	N/A

学習後に Test データを用いて推論した車両制御パラメータと教師データの散布図から求めた決定係数R^2を表３に示す。同様にアーム制御パラメータについて表４に示す。

またアーム基本姿勢の教師データに対しての正解確率、制御モード（車両制御とアーム制御、タスク終了の判定）推定の教師データに対しての正解確率を表５に示す。なお、表４のAngleと表５のPos=3について

は、本実験の走行においては、データセット中に含まれなかったため評価していない。

表５　各モデルのアーム基本姿勢(pos)と制御モード切替(mode)推定精度(%)

model	Pos	Mode
RNN + Skip(2023)	99.03	99.64 (2mode)
RNN + Skip(2024)	99.66	99.73 (3mode)

表３より、車両制御パラメータの推定では 2023 モデルと 2024 モデル(RNN＋Skip 2023、RNN + Skip 2024)間で明確な差はなかった。一方、表４と表５より、アーム制御パラメータの推定では 2023 モデルに比べ、2024 モデルの精度が明確に高いことが分かる。また表５からは車両とアームの制御切り替えでは 2023 モデルと比較し 2024 モデルは高い推定精度であることが分かる。

2023 モデルのデータセット規模は 28 万個程度に対し、2024 モデルのデータセット規模は 11 万個程度と半分以下であるにもかかわらず、高い精度を実現した。2024 モデル開発で実施した入力画像の高精細化、入力パラメータの拡大 (スタートゴールの階数等)、ResNet 部の改良が特にアーム制御パラメータ推定において高い学習能力の獲得に貢献したと考える。

表３から表５までの結果から、2023 モデルではアーム制御パラメータの推定精度は車両制御パラメータ推定精度と比べ全般的に低いことが分かる。アーム制御の失敗がタスク失敗の主要因となる可能性が高く、アーム制御パラメータの推定モデル選択は車両制御パラメータよりも重要といえる。

2023 モデルに比べて、2024 モデルは、アーム制御パラメータ推定において性能が高く、他のパラメータについても遜色ない精度があり、このことから、これまでで最も高い安定性をもつ推論モデルを実現できたと考える。

５．結言

本研究では、アーム搭載移動ロボットのエレベータを使った階層間移動を模倣学習するため、まずアーム搭載移動ロボットを試作し、アーム及び車両を手動操作可能とした。次にエレベータ操作を含む階層間移動時のカメラ画像を含む各種センサデータの時系列データとその時の操作データを記録したデータセットを作成した。このデータセットを用いて、ResNet ブロック、SE ブロック、RNN ブロック、Autoencoder、Skip コネクション等のアーキテクチャを活用したニューラルネットを設計、学習し、RNN と Skip を併用したモデルが妥当であることを確認した。

今後は引き続き、RNN の改良モデルである GRU アーキテクチャや、グリッパ付きアームを搭載した移動ロボットによるハンドリングを含む移動制御、少ないデータセットで移動可能なモデルの構築、言語命令の活用および、動作安定性向上の手法についても検討していきたい。

参考文献

[1] 堀江貴雄：Neural Network Console を使用したメカナム台車制御方法の開発、 ロボティクス・メカトロニクス講演会 2020, 1A1-G10, 2020.

[2] 堀江貴雄：カラー画像と距離画像を用いた模倣学習によるメカナム台車の移動制御、 第 38 回日本ロボット学会学術講演会、 3A3-06, 2020.

[3] 堀江貴雄：機械学習を用いたロボット関連製品の制御技術の開発、長崎県工業技術センター研究報告、 No.51, pp.18-22, 2022.

[4] 堀江貴雄：ディープラーニングを活用したロボット制御における安定性向上の研究、長崎県工業技術センター研究報告、No.52, pp.13-18, 2023.

[5] Shun Otsubo, Yasutake Takahashi, Masaki Haruna, “Modular Neural Network for Learning Visual Features, Routes, and Operation Through Human Driving Data Toward Automatic Driving System,” https://doi.org/10.20965/jaciii.2020.p0368, Journal of Advanced Computational Intelligence and Intelligent Informatics Vol.24 No.3, 2020.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun “Deep Residual Learning for Image Recognition”, http://arxiv.org/abs/1512.03385, 10 Dec 2015.

[7] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, Enhua Wu, “Squeeze-and-Excitation Networks”, https://arxiv.org/abs/1709.01507, 2017.

[8] W. Zaremba, I. Sutskever, O. Vinyals,” Recurrent neural network regularization”, arXiv:1409.2329, 2015.