

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 専門研究員 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。今年度はネットワークの改良のため、異なるアーキテクチャを適用したネットワークを複数試作し精度比較を行い、最適なネットワークを模索した。

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移動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。

ITX 規格の PC を搭載した。

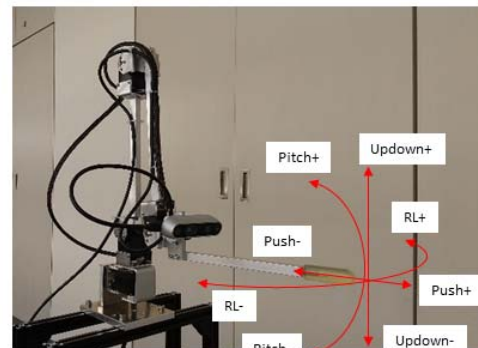


図1 ハンドアイ

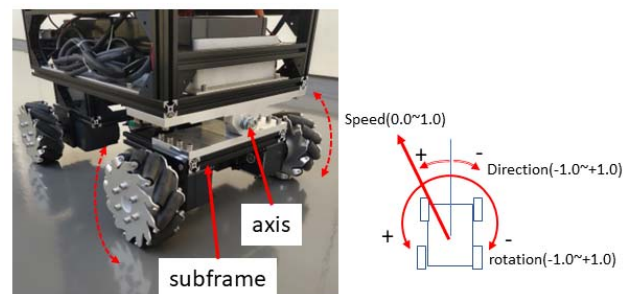


図2 メカナム駆動

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。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、ニューラルネット推論および制御プログラム実行用に

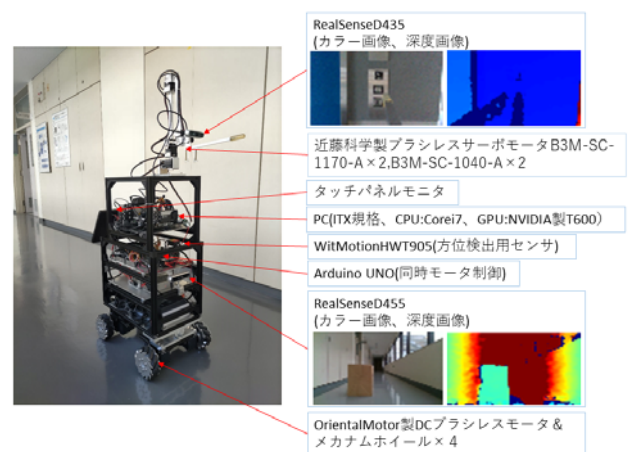


図3 ハンドアイ搭載全方向移動ロボット

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。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（表1）。

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

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）。

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(3×45×80)×5(5時刻分)	車両速度 speed(1)
車両深度画像(3×45×80)×5	車両平行移動方向 direction(1)
アームカラー画像(3×45×80)×5	車両回転速度 rotation(-1~1)
アーム深度画像(3×45×80)×5	アーム push(1)
磁気センサ X(1)×5	アーム up and down(1)
磁気センサ Y(1)×5	アーム RL(1)
地点番号(9)	アーム angle(1)
ルート選択番号(1)	アーム基本姿勢 arm position(4)
	車両、アーム制御選択 mode(1)

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いくつかの改良を実施した。

カラー画像特徴抽出のために設計した Image Convolution Unit は、これまで ResNet ブロック[6]9 つで構成していたが、新たに、Attention 機構の一種である SE ブロック[7]を追加した（図4）。

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返えし、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 Skip 機構を導入する。すなわち、現時刻 T の計算部入力に1時刻前(T-1)から4時刻前(T-4)の計算結果である4つの隠れベクトルを、RNN ブロックに向かう経路とは別に、連結した。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。

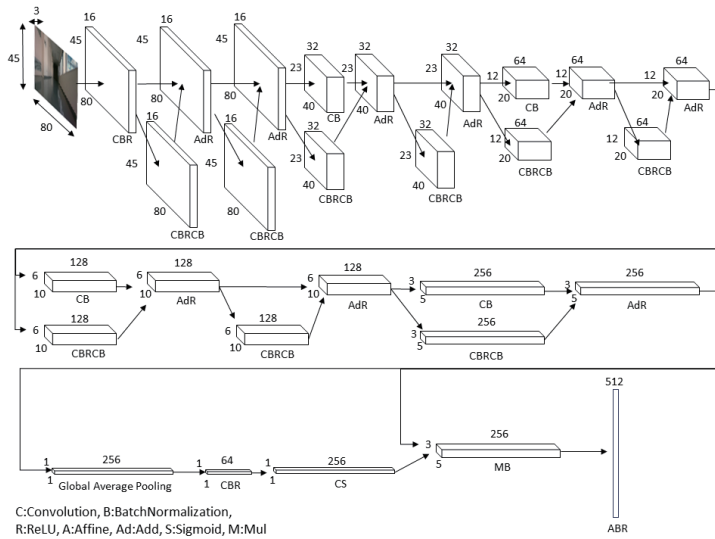


図4 Image Convolution Unit

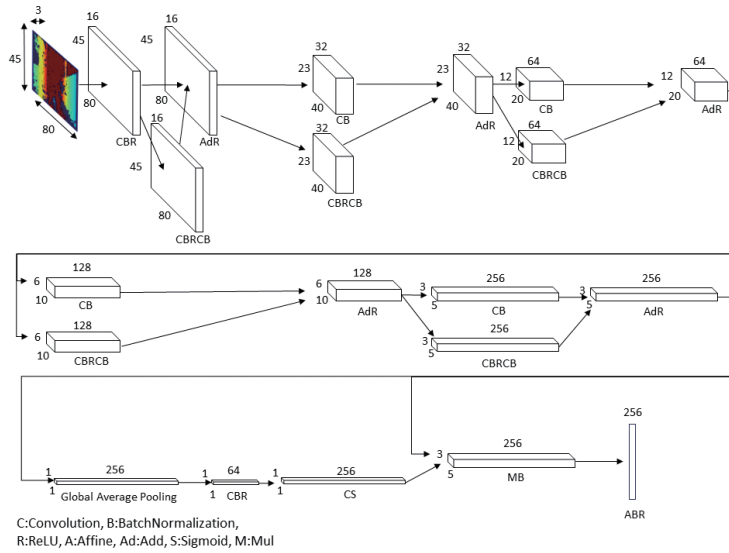


図5 Depth Image Convolution Unit

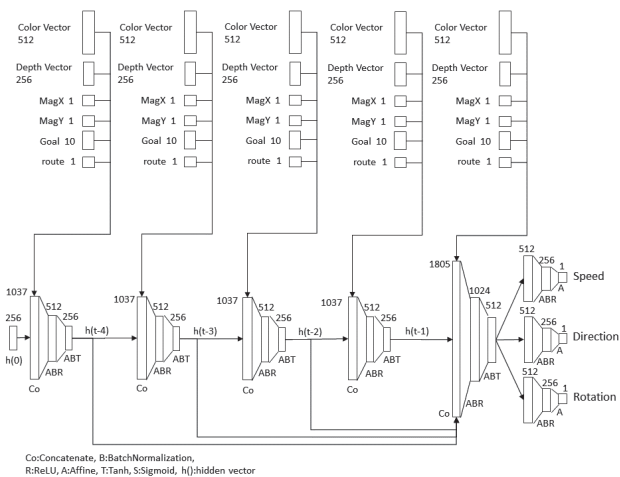


図6 Vehicle Control Unit

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）。

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。

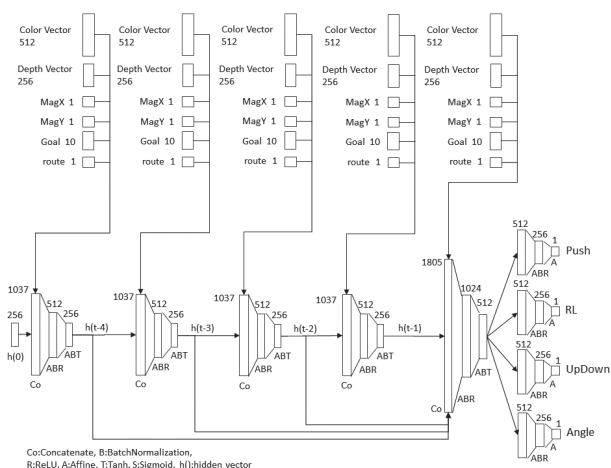


図7 Arm Control Unit

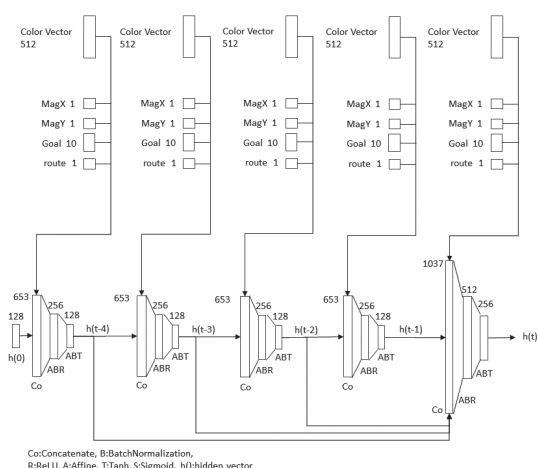


図8 Situation Unit

また、この2つの推定においては深度画像を用いずとも可能であることが実験により判明した。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)。

さらにこのネットワークで得られた車両状態ベクトル h_v とアーム状態ベクトル h_a 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。

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。これは RealSense D455 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と D455 センサで取得した深度画像のピクセル値差分を

合計し、出力する。

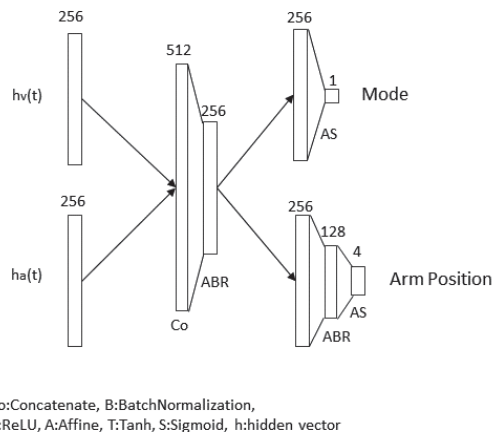


図9 PM Select Unit

後述するクライアントスレッドでは、この値を評価し、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。

	Color image (sensor_data)	Depth image (sensor_data)	Estimated depth image
Trained environment			
Untrained environment			

図10 PTD Autoencoder の出力事例

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。

設計したニューラルネットは Python サーバー上で実行し、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6～7 Hz を維持するよう調整した。

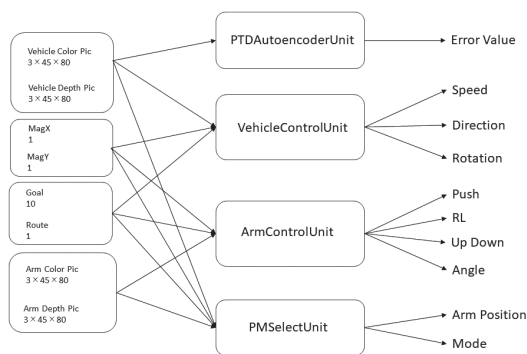


図1-1 統合ニューラルネットワーク

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1-2)。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。

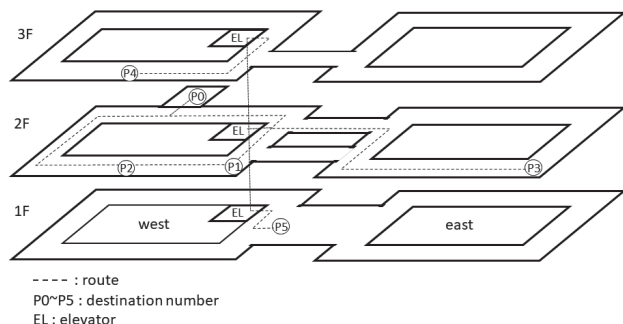


図1-2 走行ルート

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 279,000 個(約12 時間 20 分に相当) 収集し、このデータセットを、Training 用 188,000 個、Validation 用 63,000 個、Test 用 28,000 個に分割した。

ネットワークの改良が有効であるかを、4 種のネットワークでそれぞれ学習後、比較する。Vehicle Control Unit、Arm Control Unit、Situation Unitそれぞれについて、各時刻の隠れベクトル出力以降の構造を変更し検証する。

具体的には現時刻データのみから制御パラメータを

推定する Normal、昨年度までに開発した5時刻データを RNN ブロックを用いて推定する RNN、5時刻データのうち過去データ4時刻分のベクトルを現時刻処理部に入力する Skip、今回提案する RNN +Skip の4種である。学習は NVIDIA 製 TITAN RTX を搭載した PC を用いて、バッチサイズ 192、最適化アルゴリズム Adam を使用し 100epoch 実施した。

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を表5に示す。車両制御とアーム制御のモード切替推定精度についても正解確率を表6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。また表中の各制御パラメータごとに推定精度が最も良いものを太字で表している。

表3 各モデルの車両制御パラメータ決定係数(R^2)

	Speed	Direction	Rotation
Normal	0.9335	0.5259	0.7094
RNN	0.964	0.7645	0.7668
Skip	0.9624	0.767	0.7797
RNN + Skip	0.9609	0.757	0.7686

表4 各モデルのアーム制御パラメータ決定係数(R^2)

	Push	RL	UpDown	Angle
Normal	0.3	0.5093	0.5550	N/A
RNN	0.6601	0.641	0.5258	N/A
Skip	0.6913	0.6155	0.5773	N/A
RNN + Skip	0.6961	0.644	0.5844	N/A

表5 各モデルのアーム基本姿勢(pos)推定精度(%)

	Pos=0 (stay)	Pos=1 (front)	Pos=2 (left)	Pos=3 (right)
Normal	99.53	48.76	66.2	N/A
RNN	99.51	69.69	82.87	N/A
Skip	99.57	69.69	79.16	N/A
RNN + Skip	99.50	72.72	83.79	N/A

表6 各モデルの制御モード切替(mode)推定精度(%)

	Mode
Normal	99.02
RNN	99.65
Skip	99.61
RNN + Skip	99.64

表3より、車両制御パラメータの推定では時系列データを計算する3モデル(RNN、Skip、RNN + Skip)間で大きな差はない。一方、表4と表5から、アーム制御パラメータの推定にはRNNブロックとSkipを併用したモデルが最も精度が高いことが分かる。また表6からは車両とアームの制御切り替えにはRNNブロックのみを使ったモデルが最も性能が高いことが分かる。

車両制御パラメータの推定では、5時刻分の情報を並列的にとらえること、アーム制御パラメータの推定では情報の時間変化と過去情報を並列的にとらえることの両方を考慮すること、制御モードの切り替えには情報の時間変化がより重要であることをそれぞれ示していると考ええる。

表3から表6までの結果から、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。したがって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。

昨年度までに開発したRNNモデルに比べて、RNNとSkipを併用したモデルは、アーム制御パラメータ推定においては最も性能が高く、他のパラメータについても遜色ない精度があり、本システムにおいては、このモデルが最も妥当であると判断した。

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、推定精度の比較を行った結果、RNNとSkipを併用したモデルがもっとも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、Transformerに使用されているScaled Dot-Product Attentionアーキテクチャの適用を検討したい。また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や、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.