公立はこだて未来大学 2022 年度 システム情報科学実習
グループ報告書

Future University Hakodate 2022 Systems Information Science Practice
Group Report

プロジェクト名

脳をつくるプロジェクト

Project Name

Make Brain Project

グループ名

世界モデルカー

Group Name

World Model Car

プロジェクト番号/Project No.

22-A

プロジェクトリーダ/Project Leader

中村仁　Jin Nakamura

グループリーダ/Group Leader

加藤木敦也　Atsuya Katogi

グループメンバ/Group Member

中村仁　Jin Nakamura
加藤木敦也　Atsuya Katogi
伊藤生慈　Seiji Ito
黒岩蒼太郎　Sotaro Kuroiwa
渡邉悠仁　Yuto Watanabe

指導教員

香取勇一　佐々木博昭　加藤譲　ヴラジミールリアボフ

Advisor

Yuichi Katori　Hiroaki Sasaki　Yuzuru Kato　Voldymyr Riabov

提出日

2023 年 1 月 18 日

Date of Submission

January 18, 2023

概要

近年、機械学習技術の発展により、様々な場面において人工知能の応用が進んでいる。中でも、試行錯誤を繰り返すことにより最適な行動系列を学習する手法である強化学習は、複雑な動作の実現や未知の環境への対応が可能なロボットを作成するための技術として注目を集めている。一方、従来の強化学習には、サンプル効率の悪さと汎化性能の低さという欠点が存在し、これらは自律ロボットの実現において解決すべき重要課題である。「脳型人工知能」は、脳科学の知見を利用した機械学習手法であり、人間の脳内で行われる情報処理の過程を模して作られている。

その一種に「世界モデル」がある。「世界モデル」とは、人間の脳における情報処理の認知科学的な理解の 1 つであり、観測情報から外界をモデル化する枠組みである。David Ha らは、「世界モデル」を用いて時間発展する外界の状態を予測し、その結果に応じた行動選択を可能にするモデルを、モデルベース強化学習の枠組みで実装した。これを用いたゲーム AI は従来の強化学習手法より良い結果を出した [1]。

しかし、「世界モデル」は、観測情報が高次元であるなどの理由から実環境への応用例が少ない。我々のグループのコンセプトは、脳の構造や情報処理の工学的応用の推進とした。そこで、「世界モデル」の一種である、*Dreamer* [2] を用いた AI カーを構築し、実環境内での自動運転の実現を目的とした。実機の構築にはラジコンカー向け自動運転プラットフォームである Donkey Car [3] を用いた。

しかし、「世界モデル」は、観測情報が高次元であるなどの理由から実環境への応用例が少ない。我々のグループのコンセプトは、脳の構造や情報処理の工学的応用の推進である。そこで、「世界モデル」の一種であり、*Dreamer*[2] を用いた AI カーを構築し、実環境内での自動運転の実現を目的とした。実機の構築にはラジコンカー向け自動運転プラットフォームである Donkey Car [3] を用いた。

まず、我々はシミュレーション環境内における AI カーの適切な走行獲得を目指した。この目標の達成に向けて、我々は実環境での自動運転を見据えた報酬設定を行い、我々が用意したシミュレーターで強化学習を行うことで AI カーが適切な走行を獲得できるか検証した。ただし、シミュレーション環境は、既存のシミュレーターにおける機械学習手法を教師あり学習から *Dreamer* を用いた強化学習に置換することで実装した。この結果、シミュレーション環境内において、我々が設定した報酬で適切に自動走行できることを確認した。また、別の環境でのファインチューニングにより AI カーの学習を行った結果、その環境でも適切に自動走行できることを確認した。

次に、ラジコンカーを改造し、実環境での自動運転の実現を目指した。この目標の達成に向けて、我々はシミュレーション環境内での学習により得たモデルを用いて AI カーを自動走行させてデータを取り、それを用いてシミュレーター上で学習を行うことで AI カーが実環境で適切な走行を実現できるか検証した。ただし、AI カーは、エッジ AI 用マイコンの Jetson Nano [5] や Donkey Car kit [6] を用いて実装した。この結果、シミュレーションにより得られた学習済みモデルを用いた AI カーが適切に自動走行できることを確認した。

本グループでは、実環境内での自動運転の実現に向けて活動した。その結果、適切にコースを走行する AI カーを「世界モデル」を用いて実装し、実環境内での自動運転を実現できた。今回の成果は、脳型人工知能の発展や自律型ロボット分野での実用化に貢献するものと考えている。今後は、AI カーのサンプル効率や汎化性能をさらに向上させ、実世界の様々なシーンでの応用の可能性を探っていきたいと考えている。

キーワード　世界モデル, 深層学習, 自動運転, Jetson Nano

（※文責: 黒岩蒼太郎）

Abstract

In recent years, there has been a proliferation of artificial intelligence in various settings due to advances in machine learning technology. Among these is reinforcement learning, a method of learning optimal sequences of actions through repeated trial and error, which has gained attention as a means of creating robots capable of exhibiting complex behaviors and adapting to unknown environments. However, traditional reinforcement learning suffers from low sample efficiency and low generalization performance, which are critical issues that must be addressed in the development of autonomous robots. Brain-based artificial intelligence, on the other hand, is a machine learning approach that exploits the findings of brain science and mimics the information processing processes that occur in the human brain.

One type of artificial intelligence that falls under this umbrella is "World Model", which is a cognitive science understanding of information processing in the human brain and a framework for modeling the external world from observed information. David Ha et al.[1] implemented a model that predicts the time-evolving state of the external world using "World Model" and enables action selection based on the results within the framework of model-based reinforcement learning. The use of this model in an AI game has resulted in superior performance compared to traditional reinforcement learning methods.

Despite this, "World Model" has limited real-world applications due to the high dimensionality of the observed information. As such, our group aims to promote the engineering applications of brain structure and information processing by constructing an AI car utilizing the "Dreamer" [2], a type of "World Model", with the goal of achieving automated driving in a real-world environment. To this end, we employed the Donkey Car [3], a self-driving platform for radio-controlled cars, as the physical machine.

First, we aimed to obtain appropriate driving for the AI car in the simulation environment. To achieve this goal, we set up a reward system for automatic driving in a real environment, and verified whether the AI car could acquire appropriate driving by reinforcement learning in our simulator. However, the simulation environment was implemented by replacing the machine learning method in the existing simulator from supervised learning to reinforcement learning using Dreamer. As a result, we confirmed that the AI car could drive appropriately with the rewards we set in the simulation environment. In addition, we also confirmed that the AI car could learn by fine tuning in a different environment, and that it could also drive appropriately and autonomously in that environment.

Next, we modified a radio-controlled car to achieve automatic driving in a real environment. To achieve this goal, we verified whether the AI car could achieve proper driving in a real environment by taking data from automatic driving of the AI car using a model obtained by training in a simulation environment and then learning on the simulator using the data. However, the AI car was implemented using the Jetson Nano [5] microcontroller for edge AI and the Donkey Car kit [6]. As a result, we confirmed that the AI car can properly drive automatically using the learned model obtained from the simulation. As a result, it was confirmed that the AI car using the learned model obtained from the simulation could drive appropriately in an autonomous manner.

Our group aimed to accomplish the implementation of autonomous driving within an actual environment. As a result, we were able to achieve self-driving within a real environment by implementing an AI car using "World model" that properly drives autonomously on a course. This achievement is believed to be a significant contribution to the advancement of brain-inspired artificial intelligence and its practical application in the realm of autonomous robotics. In the future, our intention is to enhance the sample efficiency and generalization performance of the AI car, as well as investigate the potential for its utilization in a variety of real-world scenarios.

Keyword World Model, Deep Learning, Automated Vehicle, Jetson Nano

（※文責: 黒岩蒼太郎）

目次

第 1 章　はじめに

1.1　背景

近年、機械学習の一つである強化学習に注目が集まっている。強化学習とは、試行錯誤を繰り返すことによって最適な方法を学習するものであり、正解データを必要とせず学習できるといった利点がある。有名な活用例としては将棋や囲碁などがある。

そこで、我々は世界モデルを用いた自動運転を実現することで世界モデルの実環境における可用性を検証した。その背景としては、既存のモデルフリー強化学習にはいくつかの欠点が存在する。その欠点の内、我々が世界モデルを利用することで次の 2 点の改善を図る。1 つ目の欠点はサンプル効率が悪いという点である。モデルフリー強化学習ではモデルベース強化学習のように自身の内部に環境を構築することを行わないため、学習に用いるデータを全て自身で用意する必要がある。そのため、サンプル効率が悪いという点はモデルフリー強化学習の欠点であると言える。2 つ目の欠点は汎用性が低いという点である。モデルフリー強化学習では先にも述べたように自身の内部に外界の環境をモデルとして構築せず、ただ自身の行動によって環境がどう変化するのかだけを入力データとして学習するため、環境そのものについて学ぶことは難しい。そのため、汎用性が低いという点はモデルフリー強化学習の欠点であると言える。

上で挙げた既存のモデルフリー強化学習が持つ２つの欠点を解決する手法として脳型人工知能に注目が集まっている。脳型人工知能が注目される理由としてはサンプル効率、汎用性ともに非常に優れていることが十分に検証されている我々の脳を数理モデル化することが欠点の解決の糸口となることが期待されているためである。我々は、脳型人工知能の一つである世界モデルに注目した。世界モデルはモデルベース強化学習の中でも学習効率に優れたモデルである。その理由は環境の圧縮された空間的・時間的表現を学習することで外界の環境を自身の内部に構築し、その環境内で迅速に自己教師あり学習ができるためである [1]。また、世界モデルは汎用性にも優れたモデルである。その理由は未来が複数存在する確率論的な状態で環境のモデルを保持するためである。そうすることによって決定論的なモデルでは得られない汎用性を獲得する。

しかし、世界モデルはシミュレーション環境で高いスコアを持つが、実環境での応用例が少ないという実態がある。その要因として実環境の不確実性の考慮が難しい点と、観測情報の次元が高すぎることが挙げられる。

（※文責: 伊藤生慈）

1.2　目的

本グループの目的は世界モデルの実環境における可用性を検証することである。1.1 節で述べたように、世界モデルには既存のモデルベース強化学習が持つサンプル効率の悪さと汎用性の低さを改善させる可能性がある。しかし、現状世界モデルを実環境に応用することに課題があり実例も少ない。このような状況の中で世界モデルを実環境に応用できる可能性を示すことには意味があるのではないかと考えた。

具体的な検証方法としては、世界モデルを用いて自動運転を実装することである。検証方法とし

て自動運転を選んだ理由は 2 つある。1 つは、シミュレーション環境で高いスコアを得ていることから、自動運転への応用についての研究が既になされていること [1]。もう 1 つは、自動運転に必要な入力情報は他のタスクに比べて限定されているためである。具体的には、カメラから与えられる外界の視覚的な情報に変化が少なく情報量が少ないことと、自動運転を実現するために必要となる車の行動の種類がステアリングの角度とスロットルの大きさのみであり、同様に情報量が少なく環境の学習が行いやすいと考えたためである。

（※文責: 伊藤生慈）

1.3 先行研究

世界モデルとは、エージェントを取り巻く環境のモデルを、観測から学習する枠組みを指す。また、深層学習の領域で近年急速に研究が進められており、今後の人工知能の鍵となるトピックとして注目されている。David Ha らは RNN ベースの世界モデルと行動決定を行うコントローラに関するいくつかの重要な概念の一部をテストする簡便なアプローチを紹介した [1]。その一部として、id Software が開発したビデオゲーム「Doom」をベースとした AI 研究プラットフォームである、VizDoom 環境や OpenAI Gym のゲーム環境の 1 つである、CarRacing-v0 環境における強化学習によるタスクの実行やその精度が示されている。

強化学習とは、ある環境内のエージェントが現在の状態を観測し、取るべき行動を決定する問題を扱う機械学習の一種である。また、エージェントは行動を選択することで環境から報酬を得ることができ、強化学習は累積報酬を最大化するような方策を学習する。

世界モデルは環境からの観測情報を圧縮する V(Vision) モデル、圧縮した表現から次の時間ステップの表現を予測する M(Memory) モデルにより構成される。また、V モデルとして変分自己符号化器（以下、「VAE」と呼ぶ。）、M モデルとして MDN-RNN が使用される。

Cae-Racing-v0 のテストの例では、VAE は Car-Racing-v0 からの観測をもとに V(Vision) モデルの学習を行った。ここで M(Memory) モデルを学習させるために、学習された V モデルを用いて、各フレームに前処理した。この前処理されたデータと、記録されたランダムアクションを用いた MDN-RNN の学習により時刻 t における状態遷移確率

$$P(z_{t+1}|\ a_t, z_t, h_t)$$

をモデル化した。ただし、a は行動、z は VAE から得た潜在変数、h は RNN の隠れ状態である。また、世界モデルを用いた、モデルベース強化学習の先行研究が他にも行われている。Danujar Harfner らは、上記の状態遷移確率に加えて、現在の状態から報酬を予測する報酬モデル

$$p(r_t|\ z_t)$$

を導入した、PlaNet(Deep Planning Network) を発表した [2]。ただし、r は報酬である。また PlaNet では方策関数を作成しておらず、クロスエントロピー法によるオンラインプランニングを行っている。また、Danujar Harfner らは前述 PlaNet をもとに、状態 s_t から行動と価値を予測する方策と価値関数を明示的にモデルとして学習する、Dreamer を発表した [4]。

方策関数のモデル：$p(a_t|\ z_t)$

価値関数のモデル：$v(z_t)$

これにより、PlaNet と同様の世界モデル上のみで、強化学習エージェントが訓練できるようになり、PlaNet と比べ、サンプル効率がさらに良くなった。Dreamer のサンプル効率の良さは Danujar Harfner ら (2020) の中で示されている。Danujar Harfner ら (2020) は A3C や D4PG といったの強化学習手法に加え、PlaNet によるモデルベース強化学習、Dreamer によるモデルベース強化学習の比較を行った。

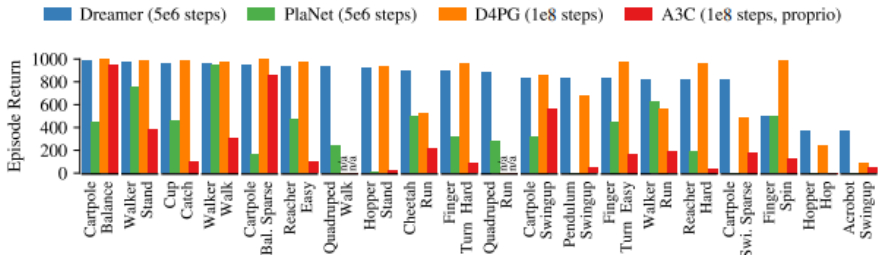

図 1.1　Dreamer は、20 タスクのベンチマークにおいて、最終的な性能、データ効率、計算時間の点で、これまでの最良のモデルフリー手法（D4PG）およびモデルベース手法（PlaNet）を上回った。(Hafner, D ら, 2019 [4])

図 1.1 は、同様の 20 個のタスクを 4 つの強化学習手法で行った際の結果である。PlaNet や Dreamer を用いた、世界モデルを使ったモデルベース強化学習では、5600 ステップ、D4PG、A3C といった既存の強化学習手法では 10^6 ステップ学習を行った際の結果である。この結果から、Dreamer が他の強化学習手法に比べて少ないサンプル数から、同等の結果が得られているといえる。また Danujar Harfner らは DreamerV2 を発表した [5]。DreamerV2 では VAE の潜在変数分布に正規分布ではなく、離散表現（カテゴリ分布に従う）にし、勾配として straight-through gradient を用いた。また、多くの世界モデルでは各画像から抽出される情報量を正則化し、汎化を容易にするために、Posterior を Prior に近づけながら正確な再構成を促す ELBO 目的関数を用いている。この目的関数は、エンドツーエンドで最適化され、Posterior と Prior はどちらかを他方に近づけることで類似性を高めることができる。しかしながら Prior がまだ正確でない場合、Posterior を Prior に近づけることに問題がある場合がある。そこで、Prior を Posterior の方へ早く動かす KL バランシングという手法をとった。これらの改良から、図 1.2 に示すように、Atari においてモデルフリー強化学習を大幅に上回る結果が得られた。

$$\text{Prior}：P(z_t|h_t)$$

$$\text{Posterior}：P(z_t|h_t, o_t)$$

ただし、o は環境からの観測情報である。

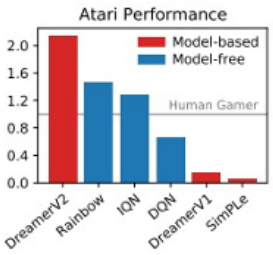

図 1.2　スティッキーアクションを持つ 55 のゲームを 2 億ステップ実行した Atari 環境におけるエージェントの人間のスコアをもとに正規化した中央値スコア (Hafner, D ら, 2020 [5])

（※文責: 渡邉悠仁）

1.4　先行研究の問題点

世界モデルには、以下のような課題が存在する。

(1). 先行研究で示されている世界モデルの利用例はいずれも、比較的次元数が少ないシミュレーション環境のみに限定されている
(2). 実環境はシミュレーション環境に比べ次元数が高い
(3). 先行研究では実環境における、世界モデルの工学的応用に関する考察が少ない

（※文責: 渡邉悠仁）

1.5　課題

1.4 節に挙げた問題点を解決するための具体策は次のとおりである。(1) の解決のため、世界モデルを用いた強化学習による自動運転を実現させ、実環境における世界モデルの可用性を検証する。ただし、システム全体としての学習を可能にする世界モデルは、自動運転が要求する要件を満たしており、これを実現させるアーキテクチャとして適切であると考えた。(2) の解決のために、実環境においてシミュレーション環境と似せた道路を作成し、シミュレーション環境で学習した重みを使ってファインチューニングを行った。このようにすることで、実環境における学習のコストが低くなると考えた。(3) の解決のために、(1) の解決で述べたように、実環境で自動運転を実現することによって、工学的応用性を検証する。

（※文責: 渡邉悠仁）

第 2 章　プロジェクト学習の概要

2.1　問題の設定

本グループは世界モデルを用いた自動運転の実現を以て、1.4 節で述べた問題点の改善を目指す。そこで、世界モデルを実装した Donkey Car を用いて、自動運転の実現を目指し、工学的応用性を検証する。また、車は行動のパターンが限られており、行動選択の学習が行いやすいため、世界モデルの実環境への応用にあたって自動運転を選択した。

（※文責: 渡邉悠仁）

2.2　課題の設定

本グループは、XiaoR Geek 製のカーキットを使用し、車載カメラを外界からの入力として車の制御を行う。また、シミュレーション環境で学習したモデルを、実環境でファインチューイングさせる。ファインチューニングを行った理由としては、シミュレーション環境下で学習したモデルを実環境でファインチューニングすることで、実環境での学習コストが低くなると考えたからである。

（※文責: 渡邉悠仁）

2.3　到達目標

2.2 節の課題を達成するために本グループでは、目標を 2 段階に分けて活動を行った。まず、1 つ目の目標は、シミュレーション環境での自動走行を目指した。具体的には自動運転プラットフォームである Donkey Car が提供するシミュレータを利用して、世界モデルを用いた強化学習により、自動運転を行わせた。2 つ目の目標は、1 つ目の目標であるシミュレーション環境で学習したモデルをファインチューニングすることで、実環境で自動走行させることを目指した。また、報酬設定は実環境でも利用できるような報酬設定をシミュレータと実環境で統一化した。

（※文責: 渡邉悠仁）

第 3 章　活動内容

本章は、活動内容の流れを簡潔にまとめたものである。詳細なログは、第 8 章「活動内容の詳細」で確認されたい。

3.1　夏季休暇までの活動

我々は、世界モデルの実環境における可用性を検証することを目標とし、その手段として世界モデルを利用して構築されている強化学習モデルを利用して AI カーの自動運転による走行を実現することを定めた。

次に世界モデルについての勉強会を行った。主に David Ha ら (2018) から世界モデルの細部までをグループ全体で確認し合いながら時間をかけて理解を共有した。

さらに、世界モデルの学習と並行して、我々が今後必要なものを発注した。初期で発注したものは、XiaoR Geek 社製のカーキットと、SD カードである。車の本体を自作せず、カーキットを購入した理由は、我々のグループが重きを置く部分は世界モデルを用いた実環境における可用性を検証することであり、車のハードウェア部分にかかる時間を省くためである。

また、我々の目標を達成する方針として、初めにシミュレーション環境内で学習させることと、使用するコードは勉強会で使用していたコードを自動運転用に改造していくことが定まった。なお、学習には画像処理に優れた GPU を搭載した PC を用いた。その後、PC のセットアップを初めに行った。PC の OS はもともと Windows であったが、我々は強化学習の研究分野で主流である Ubuntu に変更した。ディープラーニングフレームワークは PyTorch を用いた。このとき、PyTorch の都合上 Ubuntu のバージョンをダウングレードする処置を行った。その後 pyenv を使用して作った仮想環境内で必要なモジュールのインストールなどといった環境構築を行った。また、コードを実行しやすくするために、Jupyter Notebook をインストールした。最後に、勉強会で使用していたコードを問題なく実行することができ PC のセットアップが完了した。

（※文責: 伊藤生慈）

3.2　夏季休暇以降の方針とその改定

我々は夏季休暇に入る前に、8 月末までの目標設定を行った。具体的には、作業の全工程のうち 3 割を終えることを目標に掲げた。そして、8 月末になり、メンバーと話し合った結果、目標を達成できたと判断したため当初の目的を継続することとした。

その後、夏季休暇の終了直前に成果発表会を見据えた後期活動の具体的な計画を立てた。具体的な計画は 8.2 節で述べる。また、実際に計画を立てる最中、工程が不透明な作業が多かったため考えられる最悪な場合を考慮し、複数の計画を立てた。

後期活動を行っていく中、都度計画の修正を行った。加えて、このままの進行では、成果発表会までに間に合わないと判断し、従来の活動日に加えて、月曜日、木曜日、土曜日を新たに活動日として加えた。実際、成果発表会の 1 カ月前までに学習の段階に入れなかったため、今後の方針を修正する必要があると判断し、メンバーと話し合った。その結果、このまま活動を継続して実環境で

の自動運転が実現できなかったとしても、それも成果であるという結論になり、当初の目的を継続することにした。

（※文責: 加藤木敦也）

3.3　使用機材

我々が使用した PC は、サンテックス社製のノート PC である Zephyrus G15 を使用した。この PC には CPU として Ryzen9、GPU として高い性能を持つ Geforce RTX 3070 が搭載されている。そのため、我々が行ったような、画像データを使った学習などのような本来であれば時間のかかる作業を非常にスムーズに行うことができた。

Jetson Nano については、NVIDIA 社が販売するマイクロコンピュータであり、マイクロコンピュータには珍しく GPU を搭載している [6]。NVIDIA 社の強みである GPU が搭載されていることで、CUDA というアーキテクチャを用いて高速な並列処理を行うことができ、Jetson Nano 単体で画像を使用する重い処理ができるといったメリットがある。

Donkey Car は、ラジコンカーなど小型自動車向けのオープンソースの自動運転プラットフォームであり、Unity を用いて作成された、車体の重さなども反映される非常に精度の高いシミュレーション環境も提供している [3]。このシミュレータでは、様々な背景や道路がまとまって設定された環境が複数提供されている。また、強化学習を行う上で必要となる、観測データ、行動データ、報酬などの情報が設定されていた。Donkey Car は、今日でも活発に活動を行っているコミュニティが多数存在し、多くの情報を閲覧することができる。また、質問用の Discord チャンネルが用意されており、初心者でも開発しやすい環境が整備されている。

我々が発注して使用したカーキットは、XiaoR Geek 社が販売しており、Jetson Nano のメモリに OS が入った状態で納品され、Donkey Car のプラットフォームを利用できるようにカスタマイズされている [7]。カーキットから組み立てた Donkey Car を図 3.1 に示す。カーキットの内容としては車本体、Jetson Nano、DC モータ、サーボモータ、カメラ、モータドライバを含む Jetson Nano の拡張ボードである。この拡張ボードは Jetson Nano & XiaoR Geek driver Board というものであり、ネット上ではほとんど情報を得ることができず開発を進める中で障害となった。

図 3.1　カーキットより組み立てた Donkey Car

実環境での学習に使用したサーキットは、図 3.2 に示すように黒い背景布にガムテープを貼ることで自作した。サーキットの製作では、シミュレーション上で学習させたモデルを用いてファインチューニングを行うため、実環境のサーキットをシミュレーション内の環境と似せることで、より

学習しやすくなると考え、シミュレーション内の環境と似た環境のサーキットが作れるように注意した。また、報酬計算でカメラから得られる色の情報を重視しているため、報酬が得られやすいように、サーキットの色の調節にも注意した。

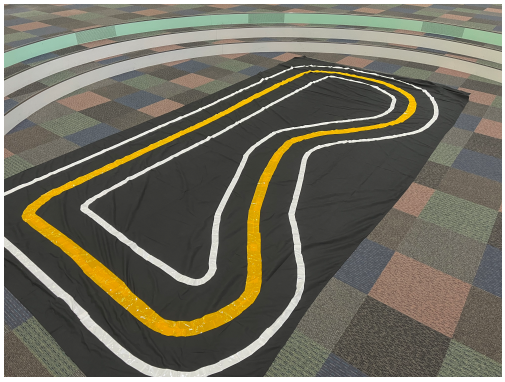

図 3.2　作成したサーキット

（※文責: 伊藤生慈）

3.4　シミュレーション環境

3.4.1　シミュレータの導入

3.3 節で述べたように、我々は Donkey Car が提供する自動運転シミュレータを使用した。初めに、シミュレータのダウンロードは GitHub にまとめられていた方法に沿って行った。しかしながら、我々が使用した Dreamer の実装コード（以下、Dreamer）は Python3.6.9 を前提に書かれていた。それゆえ、Python のバージョンを 3.7.0 に上げるにあたり、Dreamer の学習内で主に利用されるライブラリである PyTorch など、様々なライブラリのバージョン調整を行った。

（※文責: 渡邉悠仁）

3.4.2　シミュレータの利用

シミュレータと Dreamer の依存関係を整備し、我々は次にシミュレーション環境と Dreamer のインタフェース部分の調整の作業を行った。初めに、シミュレーション環境を Dreamer 内で呼び出すために、必要なクラスや関数をシミュレータのコード内を確認し、インポートを行った。次に、Dreamer の実装には入力情報として観測データ、行動データ、報酬、1 エピソードの終了フラグ情報を必要としていたため、それらをシミュレータから取り出す作業を行った。また、シミュレータが観測データを 120×160 サイズで出力する一方、Dreamer は入力情報として、64×64 サイズの観測データを必要としていたため、Dreamer 内でシミュレータから取り出した観測データを使用する際は、OpenCV による画像のリサイズを行った。次に我々は、Dreamer で学習を行う際に入力情報として使用する報酬の計算方法を考えた。シミュレータ内ではもともと、報酬を計算するアルゴリズムが実装されていたが、道路の中心からの距離など、シミュレータであるからこそできる報酬設定がなされていた。また、もともとシミュレータ内で実装されていた報酬設定で Dreamer で

の学習を行った際、600 エピソードで適切な学習が行われていたと判断したため、Dreamer の学習は 600 エピソードを基本とした。

（※文責: 渡邉悠仁）

3.4.3 シミュレーション環境における Dreamer の学習

我々は、シミュレーション環境で学習したモデルを使用して、実環境におけるファインチューニングを行うことを計画していた。そこで実環境での学習も考慮し、報酬設定は道路の中心である黄色線の RGB 値と行動データの 1 つである、スロットル値から定義することとした。また、RGB 値の報酬は、黄色線を感知するような RGB 値範囲を定め、それを満たすピクセルの割合である、*pixel_percentage* から算出した。

$$\text{報酬計算：} \frac{pixel\_percentage}{5} \times \frac{throttle}{10}$$

また、学習の際、1 エピソードの終了を定義するために、*dead_to_count* という変数を導入した。*pixel_percentage* が 0.1% 未満の時に *dead_to_count* が 0 から増えていき。*dead_to_count* が 30 カウントに達したところで 1 エピソードを終了とした。また、30 カウントの間で *pixel_percentage* が 0.1% 以上に戻った際は、カウントを減らすようにした。最後に、この学習したモデルを実環境でファインチューニングすることを考えていたため、学習が実環境と比べ容易であることから、他のシミュレーション環境による、このタスクにおけるファインチューニングの有効性の検証を行った。

（※文責: 渡邉悠仁）

3.5 実環境

3.5.1 自動運転カーのセットアップ「Jassy 版」

実環境においては、世界モデル等のソフトウェア側に注力したいという動機から、既存の Donkey Car に対応するキットを購入することとした。セットアップ作業中、キットに付属されていた Jetson Nano が起動しなくなったため、別途で用意した Jetson Nano を使用した。この別途で購入した Jetson Nano を「Jassy 版」と呼称する。「Jassy 版」では、環境構築と、我々で用意した学習用の Dreamer のコードを正常に動作させられることは確認するところまで進めることができた。ところが、プロジェクト学習終盤で、「Jassy」で我々がインストールした Jetpack ではキットに梱包されていた拡張ボードとの互換性がなく、実機のモーターの制御をすることができないことが判明した。そのため、「Jassy 版」での活動をあきらめ、後継機として「Nao 版」に作業を移行した。ただし「Jassy 版」の詳細は 8.4.1 節で述べる。

（※文責: 黒岩蒼太郎）

3.5.2 自動運転カーのセットアップ「Nao 版」

この章では、「Jassy」の後継機として新たに準備された「Nao」に関し記載する。「Jassy 版」では、XiaoR Geek 社による Donkey Car Kit の Jetson Nano に初期搭載されている、Jetpack を改造したパッケージ（以下、「改造 Jetpack」と称する）がインストールされてなかったために、Donkey Car が提供するプログラムの実行をさせることができなかった。そのため、「Nao」ではインターネット上から見つけ出し、利用することとした。

改造 JetPack と「Nao」のセットアップ

128GB の SD カードに対し、Google Drive からダウンロードした「改造 Jetpack」と、Jetson Download Center から「改造 Jetpack」のバージョンである Jetpack4.2 に対応する「Jetson Nano Developer Kit SD card image」をインストール（フラッシュ）した。様々な操作を行い最終的に上手く行ったバージョンを記載するための準備を行った。フラッシュが完了した後、ライブラリ等の大きなパッケージをインストールしたり、Dreamer コードを長時間動かしたりすることができるようなメモリを確保した。パッケージの準備として、PyTorch 等の Dreamer 用パッケージをインストールし、第 8 章に記載されている「Jassy 版」と同様な方法で依存関係を解消した。「Jassy 版」における Jetpack バージョンと「Nao 版」における Jetpack バージョンが異なることから、これまでの requirements を直接的に使用することができないたため、第 8 章で述べる「Jassy 版」で行っていた様に、地道に一つ一つのバージョンを合わせていくこととした。バージョン合わせの後、「Nao」とカメラとの通信が確立していることや、サーボモーターと DC モーターが制御されていることを確認した。また、学習に必要なデータとして、適切に観測データを保存できているのかを確認した。後者の目標に関する計画の実行として、Dreamer エージェントにカメラからの観測データを入力し、Dreamer エージェントから行動データがエラーなく出力されることと、行動データを適切に保存できていることを確認した。

Dreamer コードと「カーの各構成パーツへの通信機構」との統合

次に、Dreamer コードにおけるシミュレーション環境とのインターフェース部分を、「カーの各構成パーツへの通信機構」へと変更する作業を行った。流れに関する目標は「Jassy」と同様であるが次の 2 つに分けることができる。1 つ目に、カメラやアクチュエータを正常に動作させ、観測データを保存できるようにすること。2 つ目に、Dreamer コードを統合し、Dreamer エージェントから正常に行動データが生成されることを確認することである。前者の目標に対する計画の実行をする際にインポートについてエラーが発生した。これを解消することが困難な理由は Jetson Nano が aarch64 という CPU アーキテクチャであることやシミュレーション環境との関係、また依存関係などである。

実機におけるファインチューニング

実機におけるファインチューニングをさせるためには、学習モデルを PC から実機「Nao」に移し、そこで再度学習させる必要がある。今回我々は、「Nao」でその場で行動決定をさせる（エッジで行動を生成させる）ことが重要であると考え、図 3.3 に示すように学習させることとした。

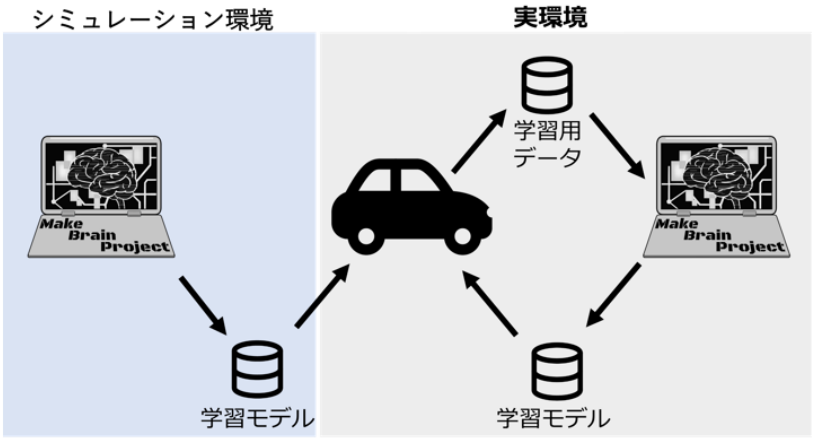

図 3.3　学習プロセス

次に、実機におけるファインチューニングをさせる。すなわち、実機を動かしデータを収集し、PC の方で学習モデルを更新することを繰り返す。この際、報酬関数を独立させ、報酬に用いるためのピクセル数が正しく計算できていることを確認した。また、その報酬に対応しているピクセルの量がシミュレーション環境と同等のものになるように調節した。ただし、実環境におけるカメラから観測したデータに基づいて報酬を計算する際、報酬とみなす RGB の範囲を広めに設定した。結果として、シミュレーション環境において制作した学習モデル（学習済みのモデル）を用いてサーキットで走らせたところ、非常に高い精度での走行をさせることができた。また、この際に収集した観測データおよび行動データを用いて報酬を計算し、PC において学習モデルを更新（ファインチューニング）させることによる、それの走行も確認することができた。

（※文責: 中村仁）

第 4 章　成果とその評価

4.1　シミュレーション環境

4.1.1　シミュレータの導入と環境構築

シミュレータの導入は GitHub にまとめられていた方法に沿って行った。ここでは、シミュレータのダウンロード方法や、利用方法がまとめられている。我々は当初 Dreamer の実装が Python3.6.9 を前提としていたため、利用していたが、シミュレータの利用に際して、Python3.7.0 以上を使う必要があった。そのため、PyTorch などのライブラリや GPU を利用するために必要な CUDA などのバージョンの調整を行った。最初は実機とのバージョン合わせを考慮して、Docker を利用した環境構築を行ったが、Docker に対する知識の不足から、作業をスムーズに進めることができなかった。最終的には、pyenv と virtualenv を利用した環境構築を行ったことにより、Python のバージョンが自由に変更できるだけでなく、仮想環境を複数つくることでアーカイブを残しつつ、実験的な作業を含めて、作業を安全に行うことができた。これらの結果として、Dreamer でシミュレータを利用することが可能となった。環境構築に関しての作業の大変さを知ることができたこと、それに関する様々な方法、解決策を調査したことは、今後にとって良い経験になったと考える。

（※文責: 渡邉悠仁）

4.1.2　Dreamer での学習結果

図 4.1 から、観測データの再構成が概ね適切に行われていることがわかり、適切な予測が行われていることが分かる。また、図 4.2 から model loss についても、500 エピソードから概ね収束していることから、適切に学習が進んでいることが分かる。ここで model loss は VAE における Posterior と Prior の間の KL ダイバージェンスと VAE のデコーダによる再構成誤差、RNN における報酬予測誤差の 3 つの和をとっている。また、学習モデルを利用した自動運転の実行時の様子から、運転や再構成に関してはよくできていると判断した。

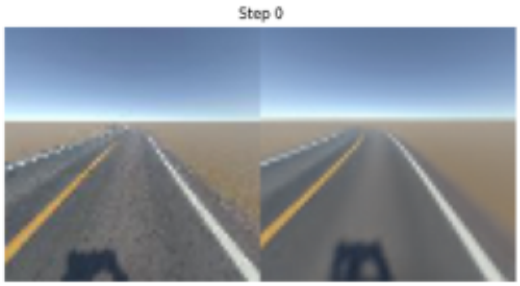

図 4.1　VAE による再構成

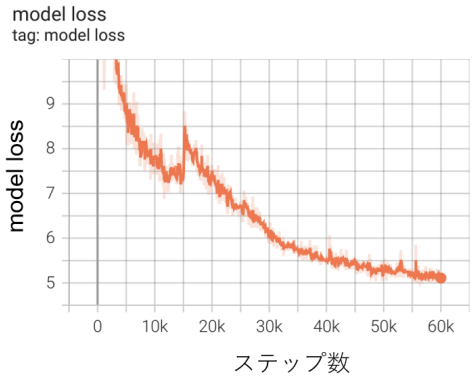

図 4.2　model loss

（※文責: 渡邉悠仁）

4.1.3　ファインチューニングの結果

図 4.3 の結果から観測データの再構成が概ね適切に行われていることがわかり、適切な予測が行われていることが分かる。しかしながら、図 4.4 から model loss については、約 50 エピソードまで概ね収束していたが、図 4.5 から 50 エピソードを超えたあたりで kl loss が急激に増加してしまっていたために増加してしまった。ファインチューニングに用いた環境が複雑であることが主な要因であると考えたが、修正には至らなかった。また、学習モデルを利用した自動運転の実行時の様子から、運転や再構成に関してはよくできていると判断した。

図 4.3　ファインチューニング後の VAE による再構成

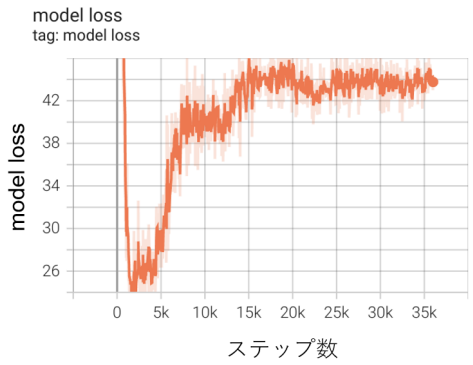

図 4.4　ファインチューニング後の model loss

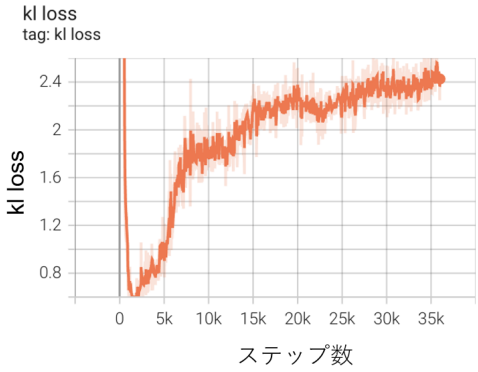

図 4.5　ファインチューニング後の kl loss

（※文責: 渡邉悠仁）

4.2　実環境

2.3 節において述べた通り、我々の目標は、実環境内での自動走行である。そのため、シミュレーション環境における学習済みモデルを、実環境下でファインチューニングすることにより実環境における自動走行を実現させることを試みた。

4.2.1　Jassy

結論から言うと、「Jassy 版」においては当初設定していた目的を達成することはできなかった。当初の予定であった拡張ボードとの接続がうまくできなかったためである。様々な原因が考えられるが、特に大きいのは次の 2 点である。

1. Xiaor Geek 社から提供されている「改造 JetPack」は、拡張ボードをすぐに使用可能にするために様々な設定が加えられていたこと
2. そのため、通常の JetPack との変更点を全て確認し自分たちの手で「Jassy 版」の設定を変更する必要があったが、それを完遂することができなかったこと

ゆえに、当初予定していた目標である実環境内での走行は「Jassy 版」では達成することができなかった。しかし、自分たちで用意した学習用の Dreamer コードを正常に動作させられることは確認できた。また、この「Jassy 版」における作業を通して、エッジコンピューティングにおける CPU のメモリ管理の重要性やハードウェアを実際に動作させることの難しさを学ぶことができた。さらに言えば、これらの作業を通して選択肢が狭まり、結果的に自分たちがやるべき作業を明確化することができた。なお、その作業もスケジュール通りには進めることができなかったが、最終的には成果につながった。以上を踏まえると、目標の達成こそできなかったが、一連の作業は次の「Nao 版」で目標を達成するための基盤になったので、「Jassy 版」での悪戦苦闘には意義があったと言える。

（※文責: 黒岩蒼太郎）

4.2.2 Nao

結論から言うと、当初からの目標を達成させることができた。まず、「Jassy 版」と同様に、自分たちで用意した学習用の Dreamer のコードを正常に動作させられることは確認できた。また、拡張ボードとの接続も順調に進ませることに成功し、学習に関する工夫や、報酬も適切に設定することができた。その結果、実環境における自動走行をさせることができた。これは、世界モデルの実環境における可用性を強く打ち出すことに成功し、今後のモデルベース強化学習における発展に寄与することができたという意義があると考えている。

加えて、これは発表 2 日前に徹夜で行ったことによる成果であった。今後やるべきことが明確であったとしても全く計画通りではなかったという点で、グループ A の学生に対しプロジェクト学習の厳しさ及び大変さを再確認させる非常に大きな契機を与えた。

（※文責: 中村仁）

4.3 中間発表会

中間発表は 2022 年 7 月 8 日に対面形式で開催された。発表と質疑応答の合計 15 分を前半、後半それぞれ 3 回ずつ行った。発表後、聴講者にアンケートフォーム用の QR コードを読み込んでもらい、発表技術と発表内容について、各項目 10 点満点の評価点数と理由やアドバイスなどを記入してもらった。

4.3.1 外部評価

アンケートの評価件数は 46 件、発表技術についての平均が 7.76 点、発表内容についての平均が 8.2 点であった。発表技術と発表内容についての評価を図 4.6 と図 4.7 に示す。評価理由の記述を一部抜粋し以下に示す。

- スライドは工夫されていましたが説明が不足しているのでは、と思われる箇所があった
- プロジェクトの名前とサービスが思っていたものと違っていた。それについての説明を最初にして欲しかった
- 専門用語が多く、内容を理解しきれない
- 声が小さくてほとんど聞き取れなかった

などの意見があった。

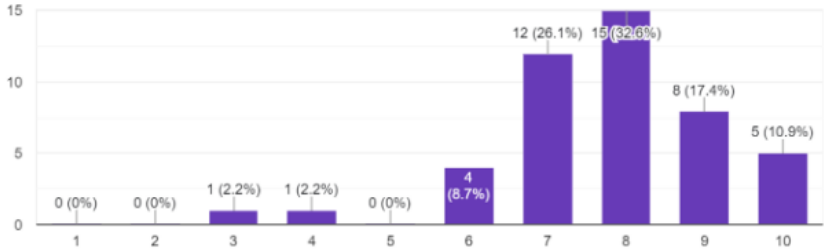

図 4.6　発表技術の評価

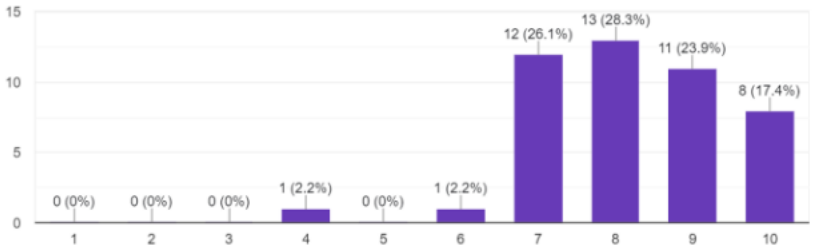

図 4.7　発表内容の評価

（※文責: 加藤木敦也）

4.3.2　総評

我々なりに世界モデルについてかみ砕いて説明したつもりだったが、VAE といった専門的な知識を用いることによって、聴講者にとって理解しにくい内容になってしまった。また、世界モデルを用いて何をしたいのかという目標が分かりにくいという指摘もあり、成果発表会で意識するよう明記した。加えて、声が聞こえにくいや小さいといった意見がいくつか見られた。発表練習が少なかったため、声量を大きくし、スライドの方を向かずに聴衆者の方を向いて発表できるように練習する機会を多くする必要があると感じた。

（※文責: 加藤木敦也）

4.4　成果発表会

成果発表会は 2022 年 12 月 9 日に中間発表会同様、対面形式で開催された。中間発表会の反省を踏まえ、プロジェクト全体で発表対策会議を複数回行い、ディスプレイやプロジェクターの配置等について話し合った。また、先生方に発表練習を見てもらい、スライドの修正を行った。

4.4.1　発表会準備及び方針会議

場所の選定や設営準備に関する会議

発表会の方針は、中間発表会の反省を活かし、グループごとではなくプロジェクト全体として進めていくこととした。これを実現させるため、プロジェクトリーダーである中村とグループリーダーである加藤木および立山が構成する「リーダー会議」が司令塔となり、企画立案と総合調整を行った。円滑な情報共有と方針制定並びに連帯確保のため、この「リーダー会議」以外のプロジェクトメンバーも参加を可能にした「成果発表会に関する対策会議（以下、「成果発表会対策会議」と呼ぶ。）」を設置し、成果発表会に関する決定を行った。10 月末から同会議を 4 回開催し、発表会の対策を総合的かつ強力に推進した。

2022 年 10 月 31 日に開催された第 1 回成果発表会対策会議では、中間発表外部評価の振り返りと今後実行する内容に関し取り扱った。この会議で、プロジェクト全体の概要をより詳細に話すことや、デザイン性のあるポスターを制作すること、発表会をより多くの方に来ていただけるようなものにするといった方針を決定することができた。また、今後の具体的なスケジュールを決めることにも成功し、これまでの想定の 2 週間ほど前倒しでスライドを制作することが決定した。この決

定を踏まえ、グループ A ではスライドの制作を前倒しですすめた。しかし、11 月の初めは Jassy 版」において時間を割いていたため、我々グループAは全体としての方針に従いつつ、自動運転を実現させるための準備を週 5 日、4,5 限に加え 25 時を超える日も多分に含めながら作業を行った。

2022 年 11 月 28 日に開催された第 2 回成果発表会対策会議では、より具体的な準備についての方針決定と細かいスケジュールの策定を行った。先生方への発表を多めにすることで練習としようと考え、3 回の発表練習をすることとなった。グループ A では、先生方への添削の依頼等の日程に合わせるため、メインポスターとスライドの完成を急いだ。先生方に対する発表の為の準備も進める必要があったが、この実行は遅れ、完成は発表会の数日前となってしまった。

2022 年 12 月 6 日開催された第 3 回成果発表会対策会議では、グループ A および B の当日に使用するスライドの統合や、各班におけるポスター、デザインポスター、動画についての方針決定とスケジュールの策定を行った。グループ A は動画の作成をすべて行うこととなった。

2022 年 12 月 8 日に開催された第 4 回成果発表会対策会議は原則全員参加とし、ポスターの印刷、設営の場所、リハーサル等についての説明を行った。同会議は、会議決定の内容が、必ずしも全てのメンバーに理解がなされていなかった等の反省あった。これにより設営に遅れが生じ、グループ A におけるリハーサルも 2 回の予定が 1 回のみとなってしまった。指揮系統の多少の乱れが存在していたが、大まかな流れはプロジェクトリーダーおよびグループリーダーにより理解されており、これに沿って設営等を進め、なんとか発表会に間に合わせることはできた。

そのため、上記 4 回の会議が行われるごとに全体としての話し合いや、グループ A 内での対応についても行われ、円滑な情報伝達をすることができたと考えている。

（※文責: 中村仁）

遠隔でのコマンドを実行

成果発表会を見据えて、パソコンから Donkey Car に SSH 接続して、遠隔からのコマンド実行を試みた。Donkey Car が正常に動作しているかを確認するために、当初はモバイルモニタを持ちながら動作確認やコマンドを実行しようと考えたが、コンセントやコードの制約があることと、作業に必要な人手が多くなってしまうという点から、SSH 接続による操作を検討した。しかし、発表前日には正常に SSH 接続ができたが、発表当日には接続できないという事態に陥った。そのため、発表中は SSH 接続をやめて、直接コマンドを打ち込んで自動走行をさせる形式をとった。その際、動作確認は必要ないため、モバイルモニタとの接続を断ち、動作させた。

（※文責: 加藤木敦也）

学習の流れに関する動画の作成

今年は対面形式で実施できたが、それでも周囲との距離を取ったりと配慮が必要だった。また、多くの人に見てもらいたいという点からディスプレイとプロジェクタをプレゼンテーションベイの両脇に設置し、人が密集しないよう配慮した。発表時間は限られているため、我々の活動を周知してもらいたいという思いから、常時モニタに動画を映した。

（※文責: 加藤木敦也）

サーキットの設置場所の検討

動画だけでなく、聴講者の移動中や質疑応答の時間に実機によるデモンストレーションを行うため、聴講者から見て、モニタをはさみサーキットを窓側に配置した。サーキットの配置場所は、先述したように VAE への入力情報に対する配慮のため、このような配置となった。

（※文責: 加藤木敦也）

4.4.2 外部評価

アンケートの評価件数は 50 件、10 点満点中発表技術についての平均が 7.72 点、発表内容についての平均が 8.38 点であった。発表技術の評価は図 4.8、発表内容の評価の評価は図 4.9 に示す。また、評価理由の記述を以下に一部抜粋する。

- 観客に目線を配りつつ、スライドの該当箇所を示しながら発表を行なっていたため非常に良いと感じた
- プレゼンテーションベイを効果的に使えていた
- 実際に作成した成果物を動かしていて分かりやすかった
- 脳との対応などの仕組みをしっかりと説明していて、わかりやすい

という意見があった一方、

- 声量はもっと大きいほうがよい
- パソコンを持ちながらの発表は不格好だった

との意見が散見された。

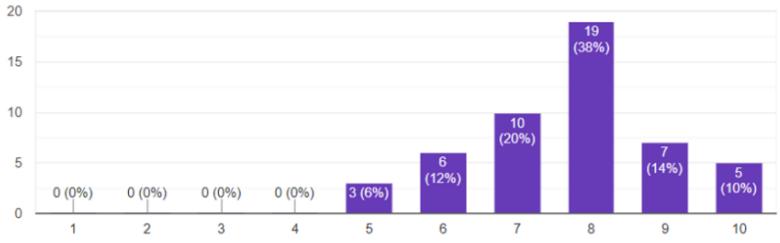

図 4.8　発表技術の評価

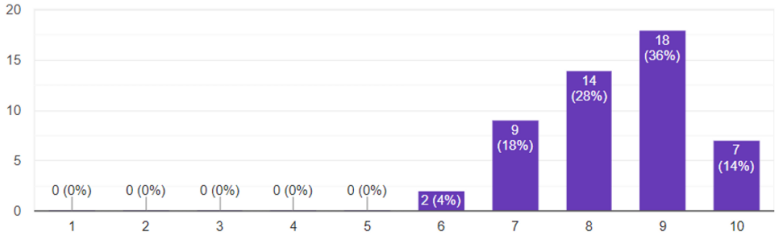

図 4.9　発表内容の評価

（※文責: 加藤木敦也）

4.4.3　総評

前半の発表時には、Donkey Car が正常に動作したため、聴講者に実機での自動運転を見てもらうことができた。しかし、後半では、カメラが故障したため、実際に動作しているところを見せることができなかった。今回はモニタとプロジェクタを活用して、作成したシミュレーション環境内と実環境での自動運転の動画を公開していたため、後半の発表時の聴講者にも我々の成果を知ってもらうことができたと思う。前期の反省を生かし、事前に会議を開いたり、先生方から添削をもらったりなど修正する機会があり、スライドについて脳との対応やグループの活動などを多くの聴講者に理解してもらうことができた。その反面、具体的な用語を可能な限り抑えたため、説明が不十分であったという意見もあった。そのため、今後誰に向けて発表するのかを意識した資料作りをしていきたいと考える。また、中間発表会の反省として発表練習が少なかったことを挙げたが、成果発表会ではその反省を生かすことができなかった。そのため、体を聴講者に向けた発表ができず、声量が小さいという事態が発生してしまった。

我々の活動を多くの方に知ってもらうということを念頭において、発表に関する活動計画を立てるよう心掛ける必要があったと感じた。

（※文責: 加藤木敦也）

4.5　グループ方針

4.5.1　夏季休暇

夏季休暇中は帰省やインターンシップにより不在のメンバーを除き、対面で参加できるメンバーのみで活動を行った。3.2 節で述べたように 8 月末までに全体の 3 割という目標を立てたことでこの期間にやるべきことが明確になり、スムーズな活動につながった。それには、週に 1 度ミーティングを行ったことが大きく貢献した。情報を共有することで活動に参加できないメンバーと話し合うことできたため、必要な情報を整理したり、やることが明確になったことで 8 月末までの目標に対しての活動を円滑に進めることができた。ただ、9 月以降の明確な目標を立てていなかったため、活動の進捗が芳しくなく、後期で行う工数が非常に多く、負担となってしまった。反省として、やることだけでなく期日についてもしっかりと話し合って決めておく必要があった。そうすることで期日を意識して活動することができ、円滑な活動につながると考える。

（※文責: 加藤木敦也）

4.5.2　後期

後期では、3.2 節で述べたように明確な活動計画を立て、かつ、進捗が芳しくない場合の計画を複数立て、時期や状況に応じて対応できるように努めた。その結果、活動が明確になり、進捗管理を円滑に行うことができた。ただし、残り期間に対して希望的観測の側面が強い計画となってしまったため、最後の方の工程が非常に困難なスケジュールとなってしまった。そのため、作業工数に見合った計画を立て、メンバーに無理を強いらない計画をたてるよう心掛けたい。

（※文責: 加藤木敦也）

第 5 章　まとめ

5.1　活動の取り組みについて

5.1.1　前期

前期の活動は、各々のやりたいテーマを考えて、そこからテーマを決定した。その後、世界モデルや使用する Dreamer コードについての勉強会を行ったり、香取教授から借用した PC の開発環境整備を行なった。また、Jetson Nano を用いて自動運転をするためにあらかじめ用意されたカーキットを購入し、Donkey Car カーの組み立てを行い、パソコンからの遠隔操作が可能かの確認を行った。中間発表会では、この Donkey Car を組み立てたことや、世界モデルについて勉強会を行ったこと、今後の活動計画などについて発表した。次に夏季休暇に向けた方針や参加状況の共有を行なった。

良かったことは、早々にテーマが決定し、方向性が決まったことだ。活動が明確になり、必要な機材を調達することができ、作業を分担しながら行うことができた。反省は、特定の個人の作業量が多くなってしまい、適切な作業分担を行うことができなかったことだ。作業工数がどの程度かかるのかを明確に把握していなかったため、このようなことが起こった。 また、先生方や先輩方への相談する回数が少なく、自分たちで調査しながら環境構築などを行ったため、環境構築に時間がかかってしまった。この反省は後期まで続いてしまい、他の人に相談することの大切さを感じた。

（※文責: 加藤木敦也）

5.1.2　夏季休暇

夏季休暇の活動は、メンバーが帰省やインターンシップなどにより、全員が集まっての活動がほとんどできなかった。ただ、夏季休暇前に参加状況を共有していたため、誰が参加するのかが把握できたことで滞りなく活動することができた。主には Dreamer で学習ができるように環境整備やプログラムの調整、Dreamer コードとシミュレーション環境とのインタフェース接続を行なった。ただ、動作はするものの、メモリの使用量が多かったりと世界モデルで一連の学習を終えることができなかった。

良かったことは、週に 1 度ミーティングを設けたことで活動進捗を共有することができたことだ。そのおかげで、都度活動の進捗を把握でき、今後考えられる進行や課題を話し合うことができ、次にやるべき作業を明確にすることができた。反省は、明確な活動計画を立てなかったことだ。具体的には 8 月末までには全体の 3 割という抽象的な目標は立てたものの、それ以降は進捗の確認を行ったのみで、いつまでに何をするのかを明確に話し合うことができなかった。

（※文責: 加藤木敦也）

5.1.3 後期

夏季休暇中の反省を生かし、後期に入る前に具体的な活動計画を立てた。また、本筋の活動計画だけでなく方針転換に対応できるように、それとは別に複数の計画を立てた。具体的な活動としては、まずは夏季休暇中に終わらなかったシミュレーション環境の整備を行った。これは、夏季休暇中の活動もあり、10 月上旬にはシミュレータを使って世界モデルによる自動運転の学習を行うことができ、その学習モデルを使用して十分自動運転ができる学習モデルが作成されたことを確認した。次にその作業と並行して、Jetson Nano の開発環境の整備を行った。多くの試行錯誤があり、Nao 版で正常に拡張ボードを利用できるようになった。その後、実環境下で適切な走行ができるように Dreamer コードを組み込んだり、報酬設定を調整した。しかし、この時点で成果発表会の 2 日前になってしまったため、満足な学習を行うことができなかった。そして、残り 2 日間で、シミュレーション環境内で作成された学習モデルを用いてファインチューニングを行った。

良かった点は、夏季休暇の反省を生かし、後期開始前までに具体的な活動計画を立てたことで活動内容が明確になったことだ。それまでは期日などを設けなかったことでこの期間にここまでやるといった目標が建てられなかったため、期日を意識した活動ができなかった。しかし、活動計画を立てることで、必然的に作業分担を行えたり、その次の作業が明確になり、スムーズな進行を実現することができた。また、計画通りにいかず、方針を転換するかどうかの話し合いを都度設けたことで、グループ内で共通認識を持つことができたと思う。

反省点は、1 つ目は成果発表会の約 1 カ月前まで拡張ボードを使用できていないことなど、情報収集が甘かったことだ。後期が始まり、 2 ヶ月近く経過しても、Donkey Car を動かすことができなかったことがその後の学習に大きく影響してしまった。また、照明の関係でカメラが得る輝度値が変化することを考慮できず、正常に走行できなかった。時間の都合上輝度値による報酬を調整できず、日中に学習ができなかったため、学習回数が少なくなってしまった。2 つ目は、実環境下での学習を成果発表会の当日まで行っていたため、十分な発表練習を行えなかったことだ。前期の中間発表会での反省として、発表練習不足が挙げられていたが、後期ではその反省を生かすことができなかった。そのため、声量が小さかったり、聴講者に体を向けずに、スライドの方を向いて発表してしまうことになった。活動と成果発表会を両立した計画を十分に立てられなかったことを反省として、綿密に準備しておくことが必要だったと感じている。

（※文責: 加藤木敦也）

5.2 成果について

本グループの目的は脳構造から工学的な応用というアプローチのもと、世界モデルの実環境における可用性を検証することであった。具体的には、世界モデルを実装した Donkey Car を用いて、自動運転の実現を目指した。そこで、達成度を明確にするために 2 つの目標を定めた。1 つ目は、Donkey Car が提供しているシミュレーション環境内での自動走行を目標とした。2 つ目は、シミュレーション環境内で作成された学習モデルを用いて、ファインチューニングを行い、実環境下での自動走行を目標とした。

1 つ目の目標であるシミュレーション環境内での自動走行を目指すための方法として、世界モデルを用いたモデルベース強化学習の 1 つである Dreamer V1 を使用して自動車エージェントを学習させた。観測情報は黒地の地面に、両脇の白線 2 本と黄色の中央線とした。学習を 600 エピソー

ド行った結果、4.1 節で述べられているように model loss がエピソード回数が増えるほど低下しているため、適切に学習できていることが分かる。また、作成された学習モデルを用いてテスト走行をした結果、適切に走行できていることから、シミュレーション環境内で自動走行を行うことができたと考える。

次に 2 つ目の目標である学習モデルを用いてファインチューニングさせ、実環境内での自動走行を試みた。Donkey Car で用意されていた教師有り学習の自動運転を行うプログラムの学習部分を世界モデルを用いた強化学習に変更したり、報酬設定を見直し、再度調整したりなどを行った結果、学習回数が少ないながらも適切な自動走行を行わせることができた。まず、シミュレーション環境内で 600 回学習させた学習モデルを用いて走行させた結果、十分な速度でサーキットを走行することができた。そして、都度走行させるたびに学習データを収集させ、そのデータからファインチューニングさせた。そのサイクルを複数回繰り返したが、シミュレーション環境下で作成された学習モデルほどの速度で走行することはなかった。また、走行も直線であれば適切に走行できたが、カーブにうまく対処できずにコース外に進んでしまうという結果になった。ただ、ファインチューニングの結果は学習の試行回数が一桁と非常に少なかったため検証が必要になる。

以上のことから、世界モデルを実装した Donkey Car で、シミュレーション環境下では適切な走行を確認できた。また、実環境下ではシミュレーション環境下で作成された学習モデルを用いた結果適切な走行ができたが、ファインチューニングの結果は学習段階といった様相で適切な走行を確認できなかった。したがって、世界モデルの実環境における可用性を完全には示せなかったが、学習の試行回数を増やすことでシミュレーション環境と同程度の走行が可能になると考える。

（※文責: 加藤木敦也）

第 6 章　今後の課題

今回はシミュレーション環境では十分な時間をかけて、ロスがしっかりと小さくなることを確認して学習モデルの作成ができたが、実環境では、時間が不足していたため、十分な学習時間を確保することができなかった。そのため、実環境での学習もシミュレーション環境と同じように、しっかりと行うことも必要である。

また、我々はシミュレーション環境、実環境ともに実際の道路のような標識や交通規則などを考慮せず、単に両脇の白線と、黄色の中央線で作成された道路を両脇の白線からはみ出さずに自動走行させることを学習させた。このようなタスクを設定した理由は実環境の情報量を制限させることで学習がしやすくなるようにするためであった。しかし、制限された簡単な環境で望んだ結果が得られただけでは、実環境において世界モデルが検証できたと言い切ることは難しく、標識や障害物などを設置し、交通規則や、急な飛び出しなどのような複雑かつ多様な環境下で学習させることが求められる。具体的には、セントラルフロリダ大学が提供している Google Street View Dataset から得られる情報をカメラ情報として入力する方法が挙げられる [8]。ただし、本当の実環境の情報では観測次元が大きくなり過ぎてしまい、学習が進まないのではないかという懸念があり、工夫が求められるのではないかと考える。また、今回使用した報酬設定の 1 つに 3.4 節で述べたようにカメラから得られる画像情報に含まれる黄色いピクセル数が全体に占める量を利用している。そのため、Google ストリートビューを利用した際、中央線が波線や白線、そもそもないといった多様な道路状況が考えられるため、それらの環境に対応させた報酬設定が必要になる。

したがって、今後の課題としては実環境において量的な面で十分な学習を行うことと、多様な道路状況や障害物、標識など、状況に応じた臨機応変な対応ができるように学習させることが挙げられる。後者の実現には高次元となることが予想される観測情報を VAE で適切な次元まで情報を圧縮できているのか確認し、もし上手く機能していないのであれば VAE の改良が求められる。また、色の情報に依存しすぎず、実環境で上手く行くような報酬設定を工夫することも重要である。

（※文責: 伊藤生慈）

第 7 章　個人の取り組み

7.1　伊藤生慈

我々のプロジェクトでは、初めに自身がシステム情報科学実習のテーマとして扱いたい内容を各々が紹介し、分野ごとに大まかなグループ分けをすることでまず 2 チームに分かれた。私が紹介した内容は機械学習を用いて実環境で何かしらの自動で動くプロダクトを作成するというものだった。チームに分かれた後は、チームごとでの進行となり、我々のチームではテーマを具体的に決定するために、各自が扱いたい内容を具体的な部分まで掘り下げて、プレゼン形式で共有し合うということを行なった。私が紹介した内容は、日常的な環境の中で適切な判断を選択できるモデルを「世界モデル」、「画像認識」、「深層生成モデル」の知識を利用して実現することであった。グループメンバー全員の意見を考慮した結果、我々のグループのテーマは世界モデルを用いた自動運転を実現することとなった。

その後、情報共有手法の統一などのグループワークで必要となってくる部分を決定したのち、世界モデルと自動運転についてそれぞれが調べ、その調べた内容を互いに共有し、質問し合うことによって、理解を深めた。私は、世界モデルについては、原著論文の翻訳を読むとともに、機械学習全般の知識不足を感じ、機械学習の初歩的な部分から勉強を行なった。時間の都合上、自動運転については、自動運転を行う際に必要となる情報は画像、ステアリング、スロットルのみで足りる程度にしか調べることができなかった。また、世界モデルの枠組みを用いて作成された「Dreamer V1」の Python コードを学習対象として輪読と同じような形式で各々の担当箇所を決め、担当者が自身の担当箇所を徹底的に勉強し、そこで得た知識をグループメンバー間で共有していった。私が主に担当した箇所は報酬を計算する部分と、行動の決定に関わる部分を担当した。

さらに、必要となる物品をメンバー間で話し合い決定し、発注を行なった。納品後、届いたカーキットに付属していた、車本体の組み立てを行なった。続いて、学習用にお借りした PC の環境構築と Jetson Nano の起動確認と環境構築を行なった。

また、勉強用に使用していたプログラミングコードを改変し、自動運転を学習できるようにし、シミュレーション環境と我々のコードを接続し、シミュレーション環境での学習ができるようにした。その後、実環境で得られるデータとコードの確認を行ない、Jetson Nano 側でも学習用のコードが実行できることを確認した。

次に、実環境でラジコンカーを動かすために、OS の破損や、モジュール間の依存関係、カーキットに付属していた Jetson Nano の拡張ボードの特異性などの様々な問題に直面し、多くの時間を費やした。この時に、同時進行で実環境にて使用するサーキットを製作した。ラジコンカーが動いた後は、学習に必要なデータを取得する部分をプログラミングコードに追加し、シミュレーション環境で製作した学習モデルを用いてファインチューニングを行なった。最初は Jetson Nano 上でデータの取得、行動決定、学習を全て行おうとしたが、スペック不足により、学習の部分だけはシミュレーションで使用した PC を用いて行なった。

（※文責: 伊藤生慈）

7.2 加藤木敦也

はじめはリーダーと書記、タイムキーパーを持ち回りでそれぞれを担当して、一回りした段階でグループリーダーを決める話を行った。私がグループリーダーに志願した理由は役職を上手にこなせなくても失敗が許される大学という環境で、グループリーダーという役職を体験したいという思いがあったため志願した。

テーマは、自己紹介のときに世界モデルという用語を知って、それを調べるうちにとても興味をもち、私のやりたいことのひとつとして発表した。他のメンバーの意見を取り入れつつ、最終的に「世界モデルを用いた自動運転の実現」というテーマに決定した。テーマが決まってからは、世界モデルに関する勉強会の計画を立てたり、Donkey Car の組立などをメンバーに指示したりすることを主に行った。画像認識や強化学習は軽く触れたことがあるから、知識が生かせるかと思ったが、世界モデルに出てくる用語で理解できるのが RNN 程度で VAE やそれ以外の用語については全く知らなかったため、勉強会を通して提供された Dreamer コードを分析して分からない部分はメンバーと話し合いながら勉強した。

中間発表会に向けた資料作成では、メンバーに作業を振り分けたり、他グループと使用するメインポスターについて話し合いながら構成を考えたりした。ただし、中間発表会までの計画を十分に立てられず、発表資料の作成指示や完成が遅れたり、先生方に見てもらう機会が少なかったりなどの反省があった。中間発表を終えて、中間提出物について考える時期となった。グループ報告書の章立てを中間発表会前に先生方に見てもらい、了承を得たため、それをもとにメンバーに作業を割り当て完成に努めた。

夏季休暇に入る前に、帰省やインターンシップにより参加できない状況が考えられるため、夏季休暇中の予定を Notion で共有してもらった。また、世界モデルを用いてシミュレーション環境内で自動運転を行わせるための環境ができていないため、夏季休暇内の目標を話し合い、決定した。夏季休暇中は週に 1 度のミーティングを開催し、情報共有を行った。メンバー全員が活動に参加できないため、活動の進捗やどこで問題が発生しているかなどの話し合いを 1 時間程度話し合った。帰省やインターンシップの都合により、夏季休暇中は 8 月上旬と 9 月下旬の参加のみとなった。具体的にやったことは、提供された Deamer コードを学習できるように環境やプログラムを整備したことや、Donkey Car が提供しているシミュレータとのインターフェース接続を行った。

夏季休暇が明ける前に、メンバーと話し合いながら後期の活動計画を立てた。後期の活動で方針が決まっていることにより活動をスムーズに行えたり、仮に計画通りに進まなかった場合の方針転換を素早く行うために計画をたてた。計画を立てた際に、このままでは世界モデルを用いた自動運転の実現が達成できないと考えたため、メンバーと話し合いながら平日と休日の一部を活動時間に割り当てた。後期では先述したように活動が明確になっているため、グループリーダーとして誰かに作業の指示をすることが減り、自分の作業に没頭することができるようになった。幾度と方針を変更するかの判断に悩まされたが、都度メンバーと話し合いながら元のテーマに沿って活動していこうということに決定した。

前期の反省を生かし、成果発表会に向けた資料作りは 1 カ月前から着々と進み、3 度先生方に見てもらうことができた。また、プロジェクト全体で発表対策会議が行われ、両グループともに情報を共有することができたことで発表会当日のモニタ等の配置を多少遅れつつも終えることができた。成果発表会当日まで活動をさせるという過酷な状況のもと、自動運転のデモンストレーションを見せられることができ、無事発表会を終えることができた。

総評として、初めてグループリーダーを務めてとても大変だった。人に対して依頼をしたり、事前に話し合いを設けたりなど考えることが多かったりと、今までにない経験を得た。成果発表会前になると、夜遅くまで残るという過酷な状況をメンバーに強いてしまったりと反省が多かった。今回グループリーダーを体験したという経験は今後の糧となり、反省を生かしていきたいと思う。

（※文責: 加藤木敦也）

7.3 黒岩蒼太郎

前期の主な活動は、テーマ設定、必要な物品の購入、「世界モデル」や強化学習に関する知識の習得、ソースコードの解析、車の組み立てや走行テスト、中間発表およびその準備である。テーマ設定に関しては、自分の興味に関連するキーワードについて調査し、それをスライドにまとめてプロジェクト全体で共有した。実環境で動くものを取り扱いたいという考えはあったものの、あまり具体的にこういうことがしたいという意見を持っていなかったが、他のメンバーの意見を聞いて自動運転に興味を持つことができた。物品の購入に関する自分の活動は、注文の重複を防いだり物品の個数の管理などを目的としたリストの作成である。「世界モデル」や強化学習に関する知識の習得に関しては、そもそも機械学習についての知識がかなり不足していることがわかったので、自分で購入した解説書などを用いて自習に取り組んだ。「世界モデル」の参考プログラムのソースコードの解析に関してはメンバーで輪読形式のように分担して実施した。しかし、自身の Python の知識不足が目立ったので、インターネットや教科書を利用しながら自身の担当部分を解析し、他のメンバーに解説した。車体の組み立てに関しては、うまくはまらないパーツを工房で削ったりしてメンバーと協力しながら完成させた。また、走行テストに関しては、Jetson nano のセットアップを担当した。中間発表の準備に関しては、ポスターの作成やスライドの準備をメンバーと協力しながら行った。しかし、パワーポイントの扱いに習熟していない、今まであまりこういったフォーマルな文書を書く経験を積んでいないといった理由から、みやすいスライドを作成したり、構成などで意見を出すことがあまりできなかった。発表会本番に関しては、個人練習の不足もあってあまり良い発表はできなかった。

後期の主な活動は、シミュレーション環境での「世界モデル」を用いた AI カーの強化学習、車体の制御に用いる Jetson nano の環境整備、ハードウェアの実装、AI カーの学習および走行、最終発表およびその準備である。シミュレーションに関しては、夏季休業中にメンバーと協力しながら初期設定を完了させたが、それ以外は基本的に他のメンバーに任せきりとなっていた。Jetson nano の環境整備に関しては、Ubuntu の設定や Python の仮想環境の整備、Python モジュールの依存関係の調整などを進めた。また、ハードウェアの実装に関しても、今回購入した車のキット独自のファイルや設定に悩まされながらも、メンバーと協力しながら 1 つずつ解決した。AI カーの学習および走行では、SSH 接続を用いることによって少しでも効率的に学習を進められるようにした。また、学習用プログラムに関しても、シミュレーションで用いていたものをメンバーと協力しながら AI カーで使えるように改変した。最終発表に関しては、前期と同様にスライドやポスターの他、発表時間中ずっと流しておく動画を作成した。発表については、中間発表のフィードバックを参考にして、声量などを改善することはできたと個人的には思う。

プロジェクト学習を通して、情報収集や知識の習得から始め、自分たちで目標を決定し、その達成に向けて計画を立案、実行して成果物を完成させるという一連の過程を経験することで、様々な学びを得ることができた。特に、このような複数人での作業ではこまめな報告や連絡が大事である

ということがよく理解することができた。自分はこまめな連絡が苦手で何度かグループに迷惑をかけてしまったので、大いに反省している。また、自分たちのよく知らない作業に関して、その作業の正当性の判断や時間配分を適切に行う難しさも感じた。実際に、自分の知識不足のせいであまり時間をかけなくて良い作業をメンバーに時間をかけさせてしまったりしたことがあったので、そこも反省点である。プロジェクト学習の活動を通して得たこれらの学びは今後の活動に大いに役立つものであると考えている。

（※文責: 黒岩蒼太郎）

7.4　中村仁

入学以来、「プロジェクト学習は社会の縮図であり非常に大変なものである」、「実機を動かすプロジェクトには終わりがない」と各所から聞かされてきた。そのため、プロジェクト学習全体として一貫した方針の決定と組織による実行が可能な体制を構築することにした。これを実現させるためには、

1. 指揮系統の確立のためのメンバーが担当する業務の確定
2. トップダウン型の意思決定方式のための組織作り
3. 空中分解を防ぐための徹底した情報共有

が不可欠であると考えた。この過程で様々な理由からプロジェクトリーダーとなった。そのため、プロジェクト初期から上記の内容に対する施策断行に邁進した。この際、グループリーダーのお二方には多くのご支援を頂きつつ進めた。全体に加えて、グループでもグループリーダーの加藤木と共に方針を練るなどの作業も行った。上記の内容は主に、高校生時代の文化祭におけるジェットコースター制作の際に担ったクラスリーダーの反省や、北の 4 大学「アントレプレナー育成講座」におけるプロジェクト学習の反省から練られたものであった。グループの性質が大きく異なることなどから上記 3 点を効果的に実現させることが難しかったが、ほぼ達成させることができ最後まで空中分解はしなかったという点で及第点は得られたのではないかと考えている。

私は、もともと世界モデルに興味があった。そのため初回の顔合わせでこれの活用を提案した。結果として「世界モデル班」が発足し、これに所属した。グループ目標の達成のために大変だった作業は、意外にも環境構築とハードウェアであった。グループ報告書に記載した通り、世界モデルを用いた機械学習以外の箇所で約 8 か月ほど苦しめられた。私は、「世界モデル」の一種である Dreamer v1 の使用をした自動運転をしよう提案した。環境構築においては、6，7 月に、Docker 上で PyTorch に GPU を使用した演算をさせることを目的として PC 環境構築を行った。Docker や CUDA、自体の学習をゼロから行い、これらを用いた環境構築を夜遅くまで行った。加えて、Dreamer コード実行のための環境、シミュレーション環境における環境、実環境における環境のためのバージョンが大きく異ならないことに配慮しつつ、情報の少ない「改造 Jetpack」に対応しているパッケージを探さなくてはならなかった。これは 8 月、9 月の活動の多くを占め、非常に大変な作業であった。また、ハードウェアに関しては、強い個性のある Jetson Nano と XiaoRgeek 社製の Donkey Car Kid が該当する。すなわち、開けない.so ファイル、ネットワーク集積回路や特有の拡張ボード等を利用するためのハードウェア等への対応として、5 か月ほど時間を消費した。確かに当初はやりたことと真反対の箇所で右往左往しているように感がしていた。しかし、振り返ってみると、本グループでソフトウェア以外の点においても学習ができたことは私にとっては

大きな収穫であったと感じている。もちろん、dreamer の学習もすることができなかったことはなく、これも良い学習の機会であったと回想している。実環境への応用をする際に、シミュレーション環境における報酬を使用することはできない。そのためこれをどのようにするかという考察を行ったが、非常に楽しい経験であった。また、入力画像のの次元が多くなりすぎるとおそらく学習が進まないのではという友人方からのアドバイスもあり、これに対応する策を練ることもできた。これも非常に面白い作業で、様々な方策を検討する過程で、自分の知らなかった機械学習手法を知る契機ともなった。このような対策も功を奏し、実環境における自動走行自体に関しては、学習自体に困難さを減らすことができたのではないかと考えている。

しかし、上記の過程から次のような反省も生まれた。第一に、初期の情報収集である。Jetson Nano や Donkey Car Kid を早めに購入し利用しようと努力していたが、まずはこれらの機材自体についての詳細を調査すべきであった。我々が実行できる情報収集は、主に、オープンソースからの情報収集と人間からの情報収集である。今回は、前者に偏りすぎており、後者が足りていなかった。ソフトウェアでは、世界モデルに関して研究している東京大学の鈴木先生、立命館大学の谷口先生らによる授業や学会発表を聴講するなどをしたが、ハードウェアに関しては、ロボット関連の研究室やその大会というに赴くことでアドバイスを受けなかった。次に、人間の限界への配慮である。夜遅くまで電気をつけることが可能になった結果、深夜までの作業が日常化した。これに伴い、深夜の帰り道において重傷者が出てしまった。そのため、健康で文化的な最低限度の生活ができるような配慮を忘れないようにしたい。

総じて、入学以来聞かされてきたことは現実となってしまったが、本プロジェクトにおける経験は、今後の活動のための大きな糧となっていると確信している。

（※文責: 中村仁）

7.5　渡邉悠仁

まずは、前期の活動での個人の取り組みについて、我々は、プロジェクトが始まった最初、やりたいことのキーワード 3 つ選択してスライドを作成することを行った。私は、AI カーに関することがやりたかったので、「パターン認識」「プランニング」「強化学習」の 3 単語を挙げた。結果として我々は、他のメンバーが提案した、世界モデルを用いたモデルベース強化学習で車を動かすことを行ったので、やりたいことに寄った内容ができたように思う。テーマがある程度固まった後は、グループでの活動が主になった。私は、機械学習やプログラミングなどの基本知識が他メンバーと比べ劣っていたため、分からないことは積極的に質問し、何もわからないということが起こらないように注意して作業を行った。前期は Dreamer の実装コードを役割分担して解説を行った。自分が担当の個所に関して、コードの写経し、一つひとつ自分の中で落とし込み、他メンバーに解説を行った。Python について 1 年次のデータサイエンス入門以来触っていなかったので、細かく調べつつの作業であった。また、グループで購入したカーキットの組み立てを行った。大きさが合わず、組み立てができない部分に関しても工房で削るなどの作業を行い、完成させることができた。我々は、車を動かすために自動運転プラットフォームである Donkey Car を利用した。テストで車を動かすために、パソコン内に Android のエミュレータをダウンロードし、Donkey Car 用のアプリケーションのダウンロードを行った。アプリケーションからの操作で観測データが保存されていることを確認した。

7 月の中間発表では、自分は主に発表用のスライドの作成を行った。発表スライドをつくるにあ

たって、世界モデルの知識の復習や、時間と分かりやすさを考慮したスライド作成をすることが大変勉強になった。発表当日では、スライドを半分に分担して発表を行った。質疑応答では、自分があまり理解していない箇所を質問されて詰まってしまった。これらの出来事と、発表後のフィードバックから、中間発表の反省を行った。　前期の総評として、周りのメンバーについていくのに必死であったが、任された仕事に関しては期日に間に合わせ、責任をもって行うことができた。しかしながら、ハード面に関しては、ほとんど他のメンバーに任せてしまったことを反省した。また、発表技術に関して、もっと練習が必要であると感じた。　次に、夏休みの活動についてである。我々のグループは夏休みに、グループリーダーが定めた日程のもと、活動を行った。加えて、毎週月曜の夜にミーティングを行い、進捗の確認、情報共有と、夏休みの目標設定を行った。夏休みの活動では、シミュレータを Dreamer 上で扱えるようにすることが主であった。この作業は、GitHub に上がっている情報や Donkey Car を使っている人がインターネット上に挙げている情報をもとに行った。結果として夏休み中に、Dreamer 上でシミュレータを利用することが間に合い、後期の活動に繋げることができた。

最後は、後期の活動についてである。まずシミュレーション側の活動について、主な自分が関わった活動は、強化学習のための報酬の設定と、行動データの調整である。報酬設定では、RGB 値と車のスピードをもとに報酬の計算を行わせた。行動の調整では、蛇行運転するように学習してしまった Dreamer の行動データに対して移動平均をとらせ、Dreamer での学習を行った。次に実機の活動について、自分が主に関わった活動は、Dreamer に行動データを生成させ、そのデータをもとに車を動かす作業、学習モデルの重みを保存した pth ファイルを異なる PyTorch のバージョン間で読み込めるように変換する作業、実際に学習させる作業である。行動データをもとに車を動かす作業として、GitHub 上にあるテンプレートをもとにカーキットに合わせたコードの改変を行った。pth ファイルの作業では、ossx ファイルに変換するなど試したのち、最終的には npy ファイルに変換し、読み込む側の PyTorch で npy ファイルを pth ファイルに変換させた。実際に学習させる作業では、データを集め pc に送り、pc で学習させ作成したモデルを Jetson に送り、再びデータを集める作業を繰り返した。

12 月の最終発表では、中間発表のスライドをもとにスライドの作成を行った。また、期限ぎりぎりで活動に進捗が出たため、最終発表の練習は十分に行えなかったため、中間発表の反省を生かしたような発表はできなかったように思えた。しかしながら、本番では実機デモができたことはとてもよかった。

1 年間の総評として、まず、環境構築に関して他メンバーに任せきりにしてしまったことが反省点である。そして、1 つ前の活動日に出ていなかった時や、報告書を書くときに、グループが何をしていたか把握するためにグループワークにおける情報共有の大切さを学んだ。また、今回あまり役割分担を行われていなかったため、手が空いてしまう時間が発生してしまった。実際にやってみないとわからなかった様々なグループワークの大変さを学べたことが、今後の他のグループ活動においても役に立つと考える。

（※文責: 渡邉悠仁）

第 8 章　活動内容の詳細

8.1　初期活動

我々はまず初めに、プロジェクト全体で顔合わせを行った後、自身がシステム情報科学実習のテーマとして扱いたい内容を個人が調べた。その内容を全員で共有し、結果として、強化学習を用いて実環境で何かしらのガジェットを動かすグループと、シミュレーション環境内でなんらかの機械学習を行なっていくグループに分かれ、前者を A グループ、後者を B グループとした。我々は A グループであるため、以下は全て A グループ内で行われた内容とする。

次に、我々は実際に扱う強化学習のモデルと、実環境で学習させて動かす具体的な対象を話し合った。強化学習のモデルについては各々が調べた上で世界モデルを活用している「Dreamer V1」を採用した。採用理由としては 2 つある。一つは、背景でも述べたように、世界モデルの特徴であるサンプル効率の良さと汎用性の高さという部分が実環境応用を見据えたときに適しているのではないかと考えたためである。もう一つは、グループメンバー内にシステム情報科学実習が行われる以前より、世界モデルを学んでいたメンバーが在籍しており、全員が触れたことのないモデルを扱うよりも効率よく、進めていけると判断したためである。実環境で学習させて動かす具体的な対象については、我々はラジコンカーの自動運転による走行を対象とした。理由としては 1.2 節でも述べたように、世界モデルはシミュレーション環境内での自動運転ですでに高いスコアを得ていること [1] と、自動運転に必要な入力情報は他のタスクに比べて限定されており、情報量が少なく、学習が比較的容易であるのではないかと考えたためである。

結果として我々は、世界モデルの実環境における可用性を検証することを目標とし、その手段として世界モデルを利用して構築されている強化学習モデル「Dreamer V1」を利用してラジコンカーの自動運転による走行を実現することを定めた。

次に世界モデルについての勉強会を行った。勉強会で行った勉強の手法としては、主に原著論文 [1] の内容から世界モデルの細部までをグループ全体で確認し合いながら時間をかけて行った。その勉強会の中で、世界モデルだけではなく強化学習全体についても学び、世界モデルのどういった点が、どのような理由で優れているのかといった部分を詳しく学ぶことができた。また、Python で実装されている Dreamer V1 のコードを見ることで、論文の内容とコードでの対応を学んだ。コードを見ていく部分では Python に対する理解が乏しかったという理由があり、難航することが予想されたため、輪読と同じような形式で各々の担当箇所を決め、担当者が自身の担当箇所を徹底的に勉強し、そこで得た知識をグループメンバー間で共有していく形をとった。

さらに、世界モデルの学習と並行して、我々が今後必要になるであろうものの発注を行った。初期で発注したものとしては、XiaoR Geek 社製のカーキットと、SD カードである。車の本体を自作せず、カーキットを購入した理由としては、我々のグループが重きを置く部分は世界モデルを用いた実環境における可用性を検証することであり、車のハードウェア部分に時間をかけたくなかったためである。加えて、様々なカーキットの中からこのカーキットを購入した理由としては、経済的に他の選択肢が少なかったという点と、Jetson Nano という GPU 付きのマイクロコンピュータが搭載されており、画像の処理が早く、エッジコンピューティングを行える可能性があったためである。カーキットは発注後、約 2 週間で納品され、世界モデルの勉強会の合間で組み立てを行っ

た。組み立て自体は動画などの資料が充実していたため、1 週間程度で完了した。Jetson Nano については、正常に起動することを確認し、付属のセットアップガイドに従って順次セットアップを完了した。この際、Jetson Nano に SSH 接続できることも確認した。尚、カーキットに付属している Jetson Nano のメモリには出荷の段階で、カーキット用の OS のようなものがメモリに入っている。

また、我々の目標を達成するにあたって、初めにシミュレーション環境内で学習させることが決まり、使用するコードは勉強会で使用していたコードを自動運転用に改造していくという方針が定まった。なお、学習には画像処理に優れた GPU を搭載した PC を担当教員より貸してもらった。その PC 内にある GPU を用いて学習させていくこととなった。その後、初めにしたことは PC のセットアップである。セットアップの完了の基準は勉強会で使用していたコードを問題なく実行できることとした。PC の OS はもともと Windows であったが、我々は強化学習の研究分野で主流である Ubuntu に変更した。初めはディープラーニングフレームワークとして TensorFlow を使う方向でセットアップを行なっていたが、ライブラリ、CUDA、ドライバ間の依存関係を解消できず、Jetson Nano との環境統一が容易であり、ライブラリの依存関係を解消する方法として Docker の利用を試み、Docker をインストールする段階で多くのエラーが発生したが、最終的にはインストールできた。しかし、我々が望むようなイメージは存在せず、イメージを自作する必要があったのだが、そのイメージの作成には Docker について深く理解する必要があることが判明したため、Docker の使用は諦めた。その後、依存関係の解消のため使用するディープラーニングフレームワークを TensorFlow から PyTorch に変更することとした。このとき、PyTorch の都合上 Ubuntu のバージョンをダウングレードする処置を行った。その後は pyenv を使用して作った仮想環境内で必要なモジュールのインストールなどといった環境構築を行い、最後にコードを実行しやすくするために、Jupyter Notebook をインストールしたのち、勉強会で使用していたコードを問題なく実行することができ PC のセットアップが完了した。

この段階では、今後の方針として、Donkey Car という自動運転プラットフォームが提供するシミュレータが利用できることを確認し、勉強会で使用していたコードを自動運転の学習ができるように改変し、そのコードをシミュレータと対応させることで、シミュレーション内で学習を完了し、そこで作成された学習モデルを用いて実環境のラジコンカーでファインチューニングを行おうと考えていた。また、PC のセットアップで前期の時間のほとんどを使ってしまった。世界モデルの実環境での学習には優に 1，2 週間以上はかかるだろうと推測しており、また、それまでにも多くの工程を要するため、8 月末までに大まかに全体の 3 割程度を達成できるかどうかを判断基準とし、達成できなかった場合方針を変えることに決めた。ただし、明確にその目標を定義したわけではなく、今後の進行で考えらえる工程を都度グループで話し合いながら決定した。方針を変更する際は世界モデルの実環境における可用性を検証する部分に重きを置いていたため、自動運転というタスクを考え直すべきなのではないだろうかという声がグループ内で上がっていた。

（※文責: 伊藤生慈）

8.2　夏季休暇以降の方針とその改定

3.2 節では述べきれなかった後期における具体的な活動計画を述べる。

8.2.1　シミュレーション環境

1. 10 月上旬にはシミュレーション環境から画像などの情報を整形し、それを入力情報として走行ができることを確認する
2. 10 月中旬にはシミュレーション環境内で、Dreamer コードによる自動運転エージェントを学習を終える。

8.2.2　実環境

1. 10 月中旬には Jetson Nano の Python ライブラリのインストールといった環境構築を終える
2. 同期間内に Donkey Car から得られる画像などの情報がシミュレーション環境とどう違うか確認する
3. 10 月末までに、Donkey Car が Dreamer コードによって学習が進むことを確認する
4. 以降は実環境での学習を行う

上記の活動計画が予定通りに進行しない場合、以下の対策を考えた。

- 問題について知見がありそうな先生や先輩に質問して対処する
- 成果発表会 1 カ月前までに学習が困難であったり、そもそも学習段階まで進まなければ今後の活動方針について話し合い、検討する

実際、成果発表会の 1 カ月前までに学習の段階に入れなかったが、このまま活動を継続して、実環境での自動運転が実現できなかったとしても、それも成果ということでいいのではという結論になり、当初の目的を継続することとした。

（※文責: 加藤木敦也）

8.3　シミュレーション環境

3.4 節で述べたように我々は、シミュレーション環境で学習したモデルを使用して、実環境におけるファインチューニングを行うことを計画していた。そのため、シミュレーション環境と実環境の報酬設定を同じにする必要があり、また、複雑な報酬設定が実環境では難しいと考えたため、シンプルな報酬設定を試行錯誤し、3.4 節のような RGB 値とスロットル値による報酬設定を行った。ところが、この報酬設定で学習させたところ、Dreamer エージェントは、道路の中心線を蛇行運転するように学習してしまった。そこで我々は、この走行結果が不適切な走行であると考え、このような学習を行った原因の考察を行った。考察の結果として、シミュレーション環境における道路にはゴールがあるため、エージェントができるだけ長い時間走行し、累積報酬を大きくするために蛇行運転することを学習してしまったと考えた。そこで我々は蛇行運転を制限するために、行動データとして、移動平均をとることを考えた。移動平均として、単純移動平均や、加重移動平均、指数平滑移動平均など様々な方法を 600 エピソードほど学習させ、検証したが、どの方法も適切に学習できていなかった。最終的には、元の行動データをそのまま利用して何度か学習のやり直しをさせ、蛇行運転を学習しなかったモデルができたところで終了とした。そして、このうまく学習した

モデルをもとに、ファインチューニングを行った。

（※文責: 渡邉悠仁）

8.4 実環境

8.4.1 自動運転カーのセットアップ「Jassy 版」

活動当初は、購入した XiaoRgeek 社製の Donkey Car キットに付属していた Jetson Nano で開発を進める予定だった。この Jetson Nano には、独自の設定が施された Jetpack が搭載されていた (以降、「改造 JetPack」とする)。しかし、この「改造 JetPack」を搭載した Jetson Nano は、Donkey Car の走行や学習に関するプログラムの流れを確認したり、実際に車をテスト走行させるなどの作業の過程で不具合が発生し、電源がつかなくなってしまった。これに際し、別の Jetson Nano を購入し、NVIDIA 社より提供されているソフトウェア開発キットの JetPack4.6.2 を搭載して、AI カーの実装に取り組んだ。以下、この Jetson Nano を便宜上「Jassy 版」と呼称する。そして、「Jassy 版」を用いて行なった作業を以下に示す。

実環境での自動運転の実現にあたって、Jetson Nano が担う役割は、主に 3 つである。1 つ目は、ファインチューニングに必要な各種データを取得し、保存することである。具体的にはカメラ画像、ステアリング、スロットル、報酬、dead 判定の有無である。ただし、dead 判定の設定に関しては、シミュレーション環境と同様にした。2 つ目は、PWM などによる車体の制御である。これらの実現に向けて、我々はシミュレーション環境で用いた Dreamer による強化学習プログラムを「Jassy 版」で実行できるようにことと、ステアリングやスロットルを制御するために用意された XiaoRgeek 社製の拡張ボードを「Jassy 版」で利用可能にすることを目的として作業を進めた。

マイコンに強いメンバーがいなかったため、カメラの制御などの初歩的なところから少しずつ学習を進めた。次に、シミュレーションで用いたプログラムの動作確認を実施した。具体的には、シミュレーションで取得される画像や、ステアリングなどの行動に関するデータと、実際に「Jassy 版」の上で取得される情報との間に齟齬がないかを、カメラ情報を取得するための関数に「Jassy 版」上で取得されるデータの同形の適当な値を持った配列を渡し、正常に動作するかを確認した。結果としては、うまくプログラムを実行することができたので、「Jassy 版」用にプログラムを改変できそうであるということがわかった。また、最初からわかっていたが、「Jassy 版」だけで学習部分のコードまで全て実行し、一からモデルを作ろうとするとメモリが足りないこともきちんと確認できた。これにより、当初から制定していた、ファインチューニングを行うことで処理をにかかる負荷を軽減し、AI カーの自動運転を実現させるという方針が妥当であると確認することができた。これらの作業の中で特に時間がかかったのは、Python モジュールの依存関係の解消およびシミュレーション環境と「Jassy 版」の環境のすり合わせである。なぜかというと、Jetson Nano は CPU アーキテクチャがシミュレーション用 PC の x86_64 と違いモジュールの整備がさほど進んでいない aarch64 であったこと、Python のバージョンを合わせるために用いた pyenv で OpenCV に不具合が生じ、正常に動作しなかったためビルドを何回か実行する必要があったことなどが理由として挙げられる。

次に、I2C 通信や PWM などを用いてステアリングおよびスロットルを制御するための拡張ボードを「Jassy 版」で使うことを目標として活動をしていたが、これは実現することはできなかった。この拡張ボードは、3.3 節に記載した通り、これ専用の XiaoRgeek 社製 Donkey Car kit に付属してきた独自のものであり、これを使用するにはこれに命令するライブラリーが必要である。これ

は、XiaoRgeek 社による Donkey Car Kit の Jetson Nano に初期搭載されている Jetpack を改造したパッケージ（以下、「改造 Jetpack」と称する）内にある。.so ファイルなどがそれに対応すると考えられる。.so ファイルは、暗号化されており開くことが難しく、米国国家安全保障会議が提供しているリバースエンジニアリングフレームワーク「Ghidra」を使用したが、解析することは困難だった。また、通常の JetPack との設定の差異についても把握が困難だった。

これを受けて、我々はこの拡張ボードを使用しない方法を模索し、調査の結果 Arudino を用いた方法でサーボモーターおよび dc モーターの制御を行い、自動運転を実現している記事に出会いそれを試した。しかし、それも失敗に終わった。

この作業が終わった段階で、このまま Jetson Nano を使い続けるか、それともドキュメントなどが充実した Raspberry Pi4 に切り替えて活動をするかをメンバー全員で協議した。その結果、Jetson Nano を用いて世界モデルベースの強化学習を行い自動運転を実現している例はほぼなかったということを踏まえ、プロジェクトの新規性を保持するために Jetson Nano を使い続けることを決断した。

我々は、失われた「改造 Jetpack」は入手ができないものと考えていた。しかし、その決定ののち、問題解決のために情報収集をした際、XiaoRgeek 社が公開している google drive 上にこれがアップロードされていることを発見したため、これを micro SD カードに焼き直した Jetson Nano に作業を引き継ぐこととした。

（※文責: 黒岩蒼太郎）

8.4.2　自動運転カーのセットアップ「Nao 版」

この章では、「Jassy」の後継機として新たに準備された「Nao」に関する、第 3 章よりも詳細なログを記載する。

「改造 Jetpack」のフラッシュ

まず、前項に述べた Jassy 版において、SD カードのメモリが非常に足りないことが発覚していたため、128GB の SD カードを用意した。同 SD カードに対し、google drive からダウンロードした「改造 Jetpack」をフラッシュした。次に、Jetson Download Center から「改造 Jetpack」のバージョンである Jetpack4.2 に対応する「Jetson Nano Developer Kit SD card image」をフラッシュした。この際に設定する Jetson Nano の名前は、同名の打ち間違いから発想を得て「Nao」と命名した。その後は様々な操作をすることになるが、5, 6 回ほどメモリや依存関係に失敗し、初期化のためフラッシュをし直した。以降、最終的に上手く行ったバージョンの流れのみを記載する。

メモリの確保とセットアップ

フラッシュが完了した後、大きなパッケージをインストールし、Dreamer コードを長時間動かすことができるようなメモリを確保した。ただし、これをしなかった場合、swap メモリが最大となったままシャットダウンし、未来永劫起動しなくなる。まず、「$sudo fdisk /dev/mmcblk0」を実行し、パーティションを削除した。その後、新しいパーティションを作成した。これによって、128GB 全てを使用できるようにした。その後、swap 領域として新たなファイルを作成し、これを有効化した。その結果、2G から 18GB まで拡張することができ、最後の学習までメモリのエラーを避けることができた。また、キーボード操作が正しくできないことを解消するため、mozc をダ

ウンロードした。

パッケージの準備

PyTorch 等の Dreamer 用パッケージをインストールした。例のように、aarch64 に対応している、Python3.6.8、CUDA10.0 にあうパッケージを探しインストール、その後依存関係を解消した。

Dreamer コードと、カーの各構成パーツへの通信機構との統合に関する目標

3.4 章に記載されている Dreamer コードにおけるシミュレーション環境とのインターフェース部分を、「カーの各構成パーツへの通信機構」へと変更する作業を行った。

流れの目標は「Jassy」と同様であるが次の 2 つに分けることができる。

1. Dreamer コードを統合する前に、「manage.py」を実行し、カメラやアクチュエータに関する Python ファイルをエラーなく動かせるようにする。
2. Dreamer コードを統合した後に、「manage.py」を実行し、エージェントが観測データに基づいて生成した行動決定により、実機が動作できるようにする。

Dreamer コードと「カーの各構成パーツへの通信機構」との統合に関する計画

前項の目標を達成させるため、テスト等も組み込んだ計画を立案した。

「『カーの各構成パーツへの通信機構』との統合に関する目標 1」に関する計画

1. import にエラーが起きていないことを確認する
2. カメラとの通信が確立しているかを確認する
3. カメラの warming が適切になされていることを確認する
4. 三角関数を用いた行動データに対応して、サーボモーターと DC モーターが制御されていることを確認する
5. カメラからの観測データを pickle ファイルに保存し、それが適切なものであるのかを確認するとともに、学習に使用することが可能であるのかを確認する（ただし、これによる報酬の生成が実際にできるのかについては確認しない）
6. 行動データ、すなわち steering および throttle データを pickle ファイルに保存し、それを確認する
7. 以上の内容のどこかで困難が発生した場合は、Jetson Nano を使用しない方法を考えるか、このまま苦しみつつ続けるかを検討する

「『カーの各構成パーツへの通信機構』との統合に関する目標 2」に関する計画

1. Dreamer エージェントのにカメラからの観測データを入力し、Dreamer エージェントから行動データがエラーなく出力されることを確認する
2. Dreamer エージェントからの行動データを pickle ファイルに適切に保存できていることを確認する

「『カーの各構成パーツへの通信機構』との統合に関する目標 1」に関する計画の実行

import についてエラーが発生した。これを解消することが困難な理由は主に次の 5 点である。1 点目に Jetson Nano が aarch64 という CPU アーキテクチャであるため、インストールできるモ

ジュールが少ないこと。2 点目に、Jetpack のバージョンが固定されており、Python3.6 しか使用できないこと。3 点目に、PC におけるシミュレーション環境と大きく異なってはいけないこと。4 点目に、依存関係が多く存在するため、これまでの点を踏まえたモジュールであっても動作しないことがあること。5 点目に、多少古いバージョンを強いられているためにエラーが発生しない場合でもモジュール自体が動作しないよう設定されている場合があること。ただし、「Jassy 版」と同様に、PC の CPU は x86 であるのに対し Jetson Nano が aarch64 という CPU アーキテクチャであることから前者の requirements を直接的に使用することができないため、上記の作業を行う以外に対処法はない。また、「Jassy 版」における Jetpack バージョンと「Nao 版」における Jetpack バージョンが異なることから、前者の requirements を直接的に使用することができないため、やはり上記の作業を行う以外に対処法はない。

対応としては例の通り、同じバージョンを使用している GitHub にアップロードされている requirements を確認したり、PIPY におけるリリース履歴を確認することで、2021 年前後に可能な限り近いものを何度もインストールとアンインストールを行い実験してみたりすることにより対応した。このような地道にこれらの過程を実行し、まずは『カーの各構成パーツへの通信機構』関連を、次に Dreamer コードを実行できるような環境を構築した。

カメラに対しては、「nvgstcapture」というコマンドを使用することで、リアルタイムでカメラ情報を取得することができ、カメラに問題がなく、カメラとの通信が確立しているかを確認することができた。また、カメラの warming が適切になされていることは以外にもすぐに確認することができた。次に、三角関数を用いた行動に対応して、サーボモーターと DC モーターが制御されていることを確認した。　学習に必要なデータとして、適切に観測データを保存できているのかを確認した。この際、画像ファイルは重いため、入出力は遅いが圧縮が可能な pickle ファイルに保存することにした。

ここまでの内容はおよそ 2022 年 11 月 11 日から 20 日までの内容であり、発表当日である 12 月 9 日まで時間が迫っている中で行われた作業である。そのため、次の「目標 2」の達成および「学習」の実行に関しては効率化が求められた。

「『カーの各構成パーツへの通信機構』との統合に関する目標 2」に関する計画の実行

Dreamer エージェントのにカメラからの観測データを入力し、Dreamer エージェントから行動データがエラーなく出力されることを確認することはでき、Dreamer エージェントからの行動データを pickle ファイルに適切に保存できていることも確認することができた。

実機におけるファインチューニングをさせるための方針

ファインチューニングをさせるためには、学習モデルを PC から実機「Nao」に移し、そこで再度学習させる必要がある。今回我々は、「Nao」で実際に行動決定をさせる（エッジで行動を生成させる）ことが重要であると考え、図 8.1 に示すように学習させることとした。

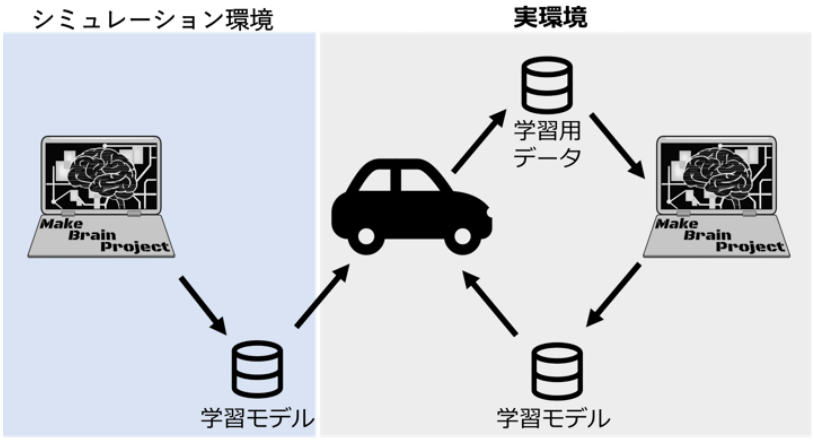

図 8.1　学習プロセス

実機におけるファインチューニングファインチューニングをさせるための計画

次のように学習させるための計画を練った。

1. 観測データから報酬を計算する機構を別に製作し、シミュレーション環境における報酬と同等な数値であることを確認する
2. サーキットを作り試走させることで、データが適切に保存され、報酬の計算も適切になされていることを確認する
3. PC に保存された PyTorch における学習モデル（.pth ファイル）を Nao に転送する
4. この重みデータを用いてエージェントに行動を決定せしめる
5. その観測データから PC の方に転送し報酬を計算させ、学習モデルを更新する
6. 2 から 4 を繰り返す

実機におけるファインチューニングをさせるための実行

結論としては効率化には失敗した。報酬計算の方法はいくつか存在している。例えば、

1. 実機で観測データが保存されるのと同時に計算してしまう方法
2. PC でファインチューニングをする際に観測データから報酬を計算する方法
3. 実機で観測データを保存した後で報酬だけを単体で計算する方法

である。当初は効率を重視し 3. とする方針であったが、コードの綺麗さ等から 1. とする方針となり、次に方針から独立して 2. を推進するグループが発生した。結果として、2. のコストが大きいだけでなく恩恵は非常に少なく、1. は学習と学習の間に報酬計算をする時間がかかってしまうため効率が悪いということから、3. となった。

報酬関数を独立させ、報酬用いるためのピクセル数が正しく計算することができることを確認した。また、その報酬に対応しているピクセルの量がシミュレーション環境と同等のものになるように調節した。ただし、実環境におけるカメラから観測したデータに基づいて報酬を計算する際、報酬とみなす RGB の範囲をシミュレーション環境よりも広めに設定した。また、Jetson Nano のパスワードの期限切れも発表会の前日に発生した。通常の作業を行っていたにも拘わらず発生したため、root 権限でパスワードを再設定しなおすなどを行った。

実行の結果

実際に、シミュレーション環境において制作した学習モデル（学習済みのモデル）を用いてサーキットで走らせたところ、非常に高い精度での走行をさせることができた。次に、この際に収集した観測データおよび行動データを用いて報酬を計算し、PC において学習モデルを更新（ファインチューニング）させ、それの走行を確認することができた。

（補足）実機の実行

実機を実行する際は「manage.py」を実行し、camera や actuator、controller 等のファイルを動かす。これにより、実機との接続が開始され、カメラの warming などの後に Dreamer コードへと進む。Dreamer による行動決定は.so ファイル等を介した DC モータ及びサーボモータに対する PWM 制御により行われる。

（※文責: 中村仁）

参考文献

[1] Ha, D., & Schmidhuber, J. (2018). Recurrent world models facilitate policy evolution. Advances in neural information processing systems, 31.

[2] Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., Davidson, J. (2019, May). Learning latent dynamics for planning from pixels. In International conference on machine learning (pp. 2555-2565). PMLR.

[3] Donkey Car, Donkey® Car - Home. https://www.donkeycar.com/（2023 年 1 月 16 日最終アクセス）

[4] Hafner, D., Lillicrap, T., Ba, J., & Norouzi, M. (2019). Dream to control: Learning behaviors by latent imagination. arXiv preprint arXiv:1912.01603.

[5] Hafner, D., Lillicrap, T., Norouzi, M., Ba, J. (2020). Mastering atari with discrete world models. arXiv preprint arXiv:2010.02193.

[6] Nvidia, "Jetson Nano 開発者キット", https://www.nvidia.com/ja-jp/autonomous-machines/embedded-systems/jetson-nano-developer-kit/（2023 年 1 月 11 日最終アクセス）

[7] XiaoRGeek, "XiaoR GEEK Donkey Car XR-F2 with Nvidia Jetson Nano Kit", https://www.xiaorgeek.net/products/xiaor-geek-donkey-car-xr-f2-with-nvidia-jetson-nano-developer-kit（2023 年 1

[8] Zamir, A. R., & Shah, M. (2014). Image geo-localization based on multiplenearest neighbor feature matching usinggeneralized graphs. IEEE transactions on pattern analysis and machine intelligence, 36(8), 1546-1558.