Image-based Reinforcement Learning with CarRacing-v0

Yih Seng Liew
School of Computer Science
University of Nottingham
hfyyl2@nottingham.ac.uk

May 10, 2022

Abstract

Reinforcement learning are often termed as the foundation that empowers the complex environments presented by autonomous driving. This work proposed the implementation of reinforcement learning algorithms such as Proximal Policy Gradient and Deep Deterministic Policy Gradient into the CarRacing-v0 environment by OpenAI Gym in hopes to produce an agent that are equipped with knowledge to complete a designated racetrack. Various research objectives have been proposed within this work; that is to investigate the difference between the performance of agents trained using different reinforcement learning algorithms and the effects of supplying the agent with observation space sourced from different RGB colour channels. A thorough discussion has been made based on the findings presented with this work.

1 Introduction

The adaptation of reinforcement learning (RL) into various industries is regarded as one of the significant breakthroughs in the field of Artificial Intelligence. It has been proven that these RL-based agents are capable of learning a complex environment by utilising rewards and punishments derived from each trial-and-error search [1]. In this project, various image-based RL techniques will be implemented to train the agent in the CarRacing-v0 environment provided by OpenAI Gym. These RL techniques will facilitate the agent in optimising the continuous problem space presented in the CarRacing-v0 environment, where the agent's primary goal is to maximise the reward signals while minimising the punishments received [2].

1.1 Aim and Objectives

The aim of this project is to develop a series of image-based reinforcement learning models that are competent in optimising the continuous problem space presented within the CarRacing-v0 environment, where an in-depth analysis of each resulting models will be made to address the research objectives proposed within this paper.

The key research objectives (questions) are:

1. To investigate how agents trained using different reinforcement learning algorithms such as Proximal Policy Optimisation (PPO) and Deep Deterministic Policy Gradient (DDPG) differ in terms of performance given the same number of training timesteps.
2. To conduct a research study on how training using a Convolutional Neural Network (CNN) that considers values from individual RGB channels found within the CarRacing-v0 environment affects the performance of the agent.

2 Related Work

In recent years, there has been a lot of emphasis on applying RL techniques to real-world problems ranging from playing strategy games like Go [3] to high-complexity problems like autonomous driving. The reason for this widespread usage of RL techniques is due to its exceptional ability in learning to adapt to a dynamic environment. Despite the widespread usage of RL techniques, this section will only discuss previous implementation of RL techniques related to OpenAI Gym environments.

Mnih et al. (2013) proposed the usage of deep CNN to address the enduring issue of RL, which is to encourage agents to perform direct learning using high-dimensional sensory inputs such as vision and speech [4]. The model proposed by the authors was proven to be successful in learning several games in Atari 2600, with the resulting agent outperforming human ability in playing these games. Nonetheless, there are similarities between Mnih et al.'s approach and this project, where both pieces of research are based on computer vision inputs. In other words, agents of RL will learn directly from pixel values retrieved from the environment's observation space.

One such paper by Zhang & Sun (2020) proposed an implementation of a CNN that aims to result in an agent that has human-like expertise in feature detections given the CarRacing-v0 environment [5]. To increase the robustness of the proposed CNN, a series of mixed neural networks that utilises sensory inputs from the agent (true speed, four ABS sensors, steering wheel position and gyroscope) and image input (pixel values) from the observation space has been implemented to accurately distinguish existing features within the environment such as the racetrack boundaries and grass.

In Fakhry (2020), an implementation of Deep Q-Network has been suggested to tackle the CarRacing-v0 environment [6]. A notable aspect of this paper is that various pre-processing methodologies were applied onto the observation space, resulting in a reduced yet concise observation space that is loaded with critical information that promotes effective Deep Q-Network training. These proposed pre-processing techniques have certainly inspired this project the importance of feature extractions in cultivating a performant agent.

3 Methodologies

3.1 Environment

Within this study, the CarRacing-v0 has been the preferred environment to demonstrate the evaluations for two of the aforementioned research objectives. Illustrated in Fig. 1 is the interface of this CarRacing-v0 environment.

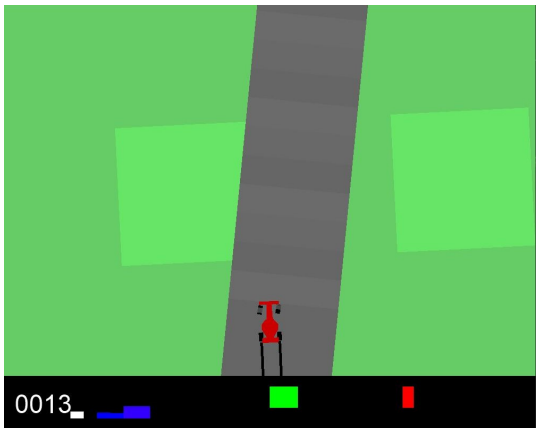

Figure 1: Interface of the CarRacing-v0 environment.

This environment contains a number of separate entities, that includes the agent represented by the red car, the observation space represented by the racetrack and the grassland, and the action space represented by the coloured bars at the bottom of the interface. Deeper into the environment,

the observation space of this environment was built using the RGB channels, which represents the red, green, and blue channels, with each channel having an image size of 96x96 pixels. As for the action space, it can be broken down into three distinct indicators represented by continuous variables, in this case, steering (with a value of -1 indicating a full left turn and +1 indicating a full right turn), gas and braking (with a value range of 0 indicating not in use to 1 indicating in full use).

The goal of this environment is for the agent to achieve a reward of 900, with each progressive frame made by the agent on the racetrack contributing to a -0.1 reward. The formula for the total reward earned by the agent during a game episode is given in Eq. 1.

$$total\_reward = 1000 - 0.1 * number\_of\_frames \tag{1}$$

A game episode has two termination criteria: the case where the agent completes every tile on the racetrack and the case where the agent accumulated a reward of -100 by cruising too far off the racetrack, resulting in a constant deduction of rewards.

3.1.1 Extraction of RGB-based Observation Space

It is noticeable that there exist a wide variety of colours present within the interface shown in Fig. 1. The primary purpose of pre-processing the observation space by extracting each individual colour channel is to trim down the input space of the RL algorithm, as it is proven in [6] that this trimming process can significantly reduce the duration of learning time required to train the agent. In the meantime, these extracted colour channels can be used to address the questions presented in the second research objective.

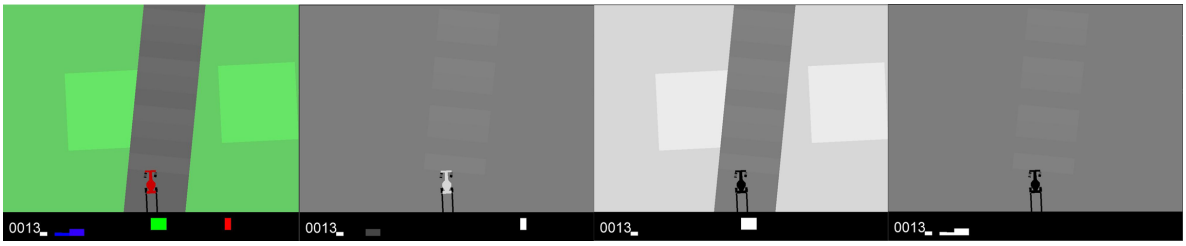

Figure 2: Illustrations of the original, red, green and blue channel after extraction (from left to right).

Referring to Fig. 2, it can be seen that results of the red and blue channel produced similar illustrations, with racetracks from both channels being represented by the faint grey boxes. The presence of these boxes is due to the design made by the environment's author onto the racetrack.

3.2 Reinforcement Learning Algorithms

3.2.1 Proximal Policy Optimisation (PPO)

The integration of PPO has been centred on the concept of simplicity, in which it leverages the amount of time required to solve complex problem space [7]. By comparing PPO to previous standard policy gradient method, PPO differs as it utilises an additional surrogate fitness function during its agent training. In other words, the training of its agent is carried out in an alternate manner, with constant switches between sampling data from the environment and optimising the surrogate fitness function. Another point worth mentioning is that PPO incorporates trust region policy optimisation (TRPO) where it effectively enhanced its generalisation ability without sacrificing performance when presented with complex problem space. Besides, it has been proven that implementation of PPO into OpenAI Gym environments such as Atari 2600 Games and Simulated Robotic Locomotion were found to be successful, resulting in an exceptional agent's performance. With that being said, it is obvious that the advantages offered by PPO are certainly the key criteria in selecting PPO as one of the RL algorithms to be evaluated within this project.

3.2.2 Deep Deterministic Policy Gradient (DDPG)

The basic idea behind DDPG is that learning of the algorithm has been conducted between the Q-function and its policy in a concurrent manner. With the term concurrent, it means the algorithm learns the Q-function by utilising both off-policy data and the Bellman equation, then applying the Q-function into the learning of the deterministic target policy [8]. This introduction of Q-function has promoted the exploratory behaviour within DDPG, ensuring that its deterministic nature can outperform its stochastic correspondent in higher-dimensional action spaces [9]. Despite being computationally intensive, DDPG has been selected as one of the RL algorithms to be compared in this project because previous research has shown that DDPG is one of the few performant RL algorithms targeting continuous action space.

3.3 Network Architecture

Within this project, a CNN architecture has been adopted to further compress the features present within the environment. The proposed CNN architecture has a total of 3 convolutional layers and specifications of each convolutional layer has been tabulated into Table 1.

Layer	Specification	Output Shape	Parameters
Convolution 1	32 filters of size 8x8 and stride of 4	[32, 23, 23]	6176
ReLU	-	[32, 23, 23]	-
Convolution 2	64 filters of size 4x4 and stride of 2	[64, 10, 10]	32832
ReLU	-	[64, 10, 10]	-
Convolution 3	64 filters of size 3x3 and stride of 1	[64, 8, 8]	36928
ReLU	-	[64, 8, 8]	-
Flatten	-	[4096]	-

Table 1: Specification of CNN architecture.

4 Implementation

4.1 Languages and Specifications

Development and training of RL agents presented within this work were performed using Python 3.7 on a CUDA-enabled GPU of Nvidia RTX 3080 that has a specification of 16GB VRAM offering 6144 CUDA Cores. The primary reason of using Python throughout the RL development process is because it offers an extensive collection of RL libraries like OpenAI Gym, Stable Baseline-3 and PyTorch. These libraries are well-equipped with necessary functions that can offer up to research usability. As for the IDE, PyCharm has been the preferred choice as it offers a range of dedicated features that assist in code debugging and performance visualisation.

4.2 Challenges

There were three main challenges faced during the implementation of this project. First of all, it is the issue of insufficient documentation made available for this CarRacing-v0 environment. The only documentation of this environment has been made available at [10], therefore it can be said that it requires a strong Python development background in order to undertake this task.

The second challenge faced concerns the shortage of computation power in training the DDPG algorithm. Various Computer Vision approaches utilising OpenCV library have been experimented to reduce computational usage by means of dimensionality reduction of the input observation space. However, these approaches did not demonstrate much improvement in terms of computational power consumption. As a result, a firm decision was made to modify the parameter settings present within the DDPG algorithm in order to support the execution of algorithm on the dedicated GPU.

The final challenge was encountered while developing a Deep Q-Learning (DQN) algorithm based on CNN. Due to the continuous nature found in the environment's action space, it complicates the implementation of DQN because DQN is more user-friendly to environments offering discrete action space.

5 Experimental Setup

5.1 Common Settings

5.1.1 Reinforcement Algorithms

As discussed previously, there are two distinct types of RL algorithms being implemented within this work, specifically the PPO and DDPG algorithm. Both RL algorithms have been developed based on the Stable Baseline-3 library featuring a range of RL algorithms accustomed for OpenAI Gym usages. Parameter settings of both RL algorithms have been kept the same to ensure fairness in evaluation of all research objectives proposed within this paper. Presented in Table 2 is the dedicated parameter settings for PPO and DDPG algorithm.

	Reinforcement Learning Algorithms	
	PPO	**DDPG**
Learning Rate	0.00003	0.00003
Policy	CnnPolicy	CnnPolicy
Timesteps	1000000	1000000
Buffer Size	-	30000

Table 2: Parameter settings for PPO and DDPG algorithm.

As discussed previously that execution of DDPG algorithm requires an extensive allocation of computational resources, the buffer size parameter of 30000 has to be pre-set to limit the size of the replay buffer. Limiting the replay buffer may cause a direct impact on the performance of the resulting agent as the trajectories of experiences stored within the agent have significantly been reduced. Meaning, the size of the subset storing the agent's experience for the purpose of "replaying" experiences has shrunk, hence restricting the agent's ability to recall past experience from older memories [11]. The replay buffer limitation may appear to be unfair, but it is undeniably the best possible solution to counter this problem because it has effectively released a portion of memory size to the GPU to facilitate agent training.

5.2 Specific Settings

5.2.1 Research Objective 1

The setup of the first research objective is fairly straightforward where parameter settings of the PPO and DDPG algorithm have been pre-set according to Table 2. The observation space of both algorithms has been pre-processed by converting the RGB layers into a single grey channel. This grey channel extraction has substantially trimmed down the size of the input space, hence promoting effectiveness to the agent's learning process. Furthermore, frame stacking procedure has been introduced onto the environment's observation space to allow the agents to utilise multiple frames during its learning. In this case, the number of frames being stacked together has been set to 4, where it allows the agent to learn based on 4 different observation spaces at once.

5.2.2 Research Objective 2

To address the second research objective, a model with a CNN architecture was implemented because studies have shown that CNN models are effective on image-based tasks. Another crucial step in

addressing this research objective is to extract each colour channel present within the RGB observation space. In this case, the layer extraction will result in three distinct layers representing the red (R), green (G) and blue (B) layer. As for the training of the agent, the PPO algorithm has been adopted as it has shown significant improvement in terms of the model performance when compared to the DDPG. Further discussion concerning the performance of PPO algorithm will be made in the Results and Discussion section.

6 Results and Discussion

6.1 Research Objective 1

The results of agents trained using two distinct RL algorithms, PPO and DDPG have been presented in Fig. 3.

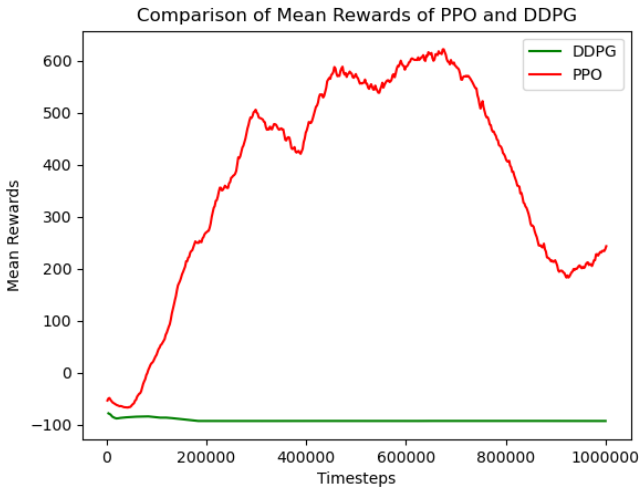

Figure 3: Comparison of mean rewards achieved by PPO and DDPG Algorithm.

To ensure a fair comparison between both algorithms, parameter settings and total timesteps used to train the algorithms have been kept constant. Referring to Fig. 3, it can be seen that there exist no significant changes in terms of mean rewards obtained by the DDPG agent. This finding has clearly implied that DDPG is not the appropriate algorithm to be applied into the training of the agent.

Scrutinising the mean rewards received by the PPO algorithm, there is a steady increase from timesteps value of 0 to approximately timesteps of 700000, followed by a downward trend towards the end of timesteps. This sudden drop in mean rewards can be elucidated using the concept of catastrophic forgetting introduced by McCloskey & Cohen (1989) where the agent starts to lose its capability for continual learning [12]. Hence, resulting in the decline of mean rewards after timesteps of 700000. Despite this, the PPO-trained agent still exhibits reasonably well performance (shown in Fig. 4), achieving its peak performance at 628 rewards and averages at 194 rewards over 100 episodes of gameplay.

6.2 Research Objective 2

From the discussion above, it can be seen that agents trained using the PPO algorithm exhibit better performance over the DDPG algorithm. Therefore, the experiments conducted in research objective 2 will be based on the PPO algorithm. Illustrated in Fig. 5 shows the mean rewards obtained by the agents trained using distinct RGB channels sourced from the observation space.

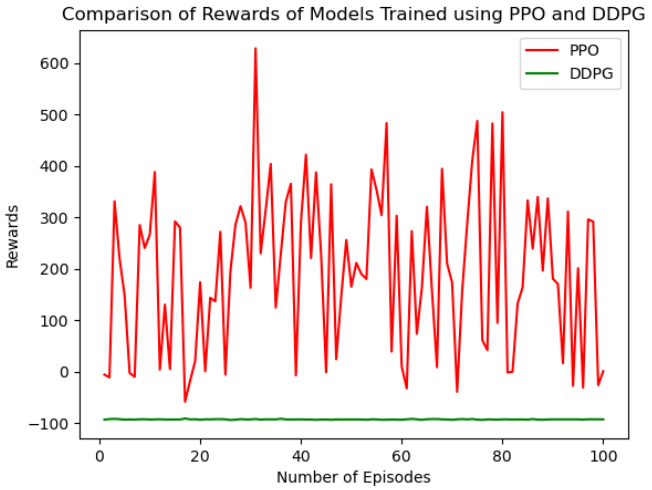

Figure 4: Comparison of rewards obtained by the trained agent after every episode.

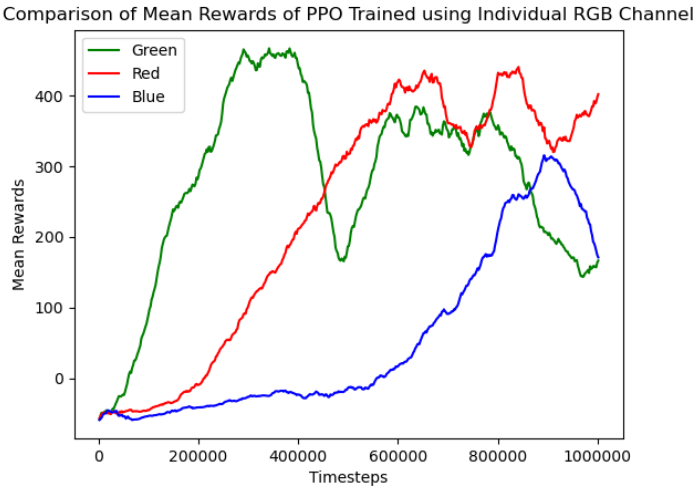

Figure 5: Comparison of mean rewards achieved by PPO and DDPG algorithm on each RGB channel.

By examining Fig. 5, it is relatively difficult to deduce that an agent trained using the green or red channel outperforms another as both graphs showed closed proximity with each other in terms of the existing trends. However, an inference has been made based on prior knowledge on the environment where green channel may be a better option in agent training as agents are assumed to be able to understand that green pixels within the environment represents grassland that cost punishment. Besides, it can be seen that the agent trained using the green channel has achieved the highest mean reward among all agents. To further verify the inference, all trained agents have been experimented with 100 episodes of unseen racetracks for an in-depth analysis over the overall performance of each agent.

The plots illustrated in Fig. 6 have proven the inference made earlier to be invalid. In contrast, the agent trained using the red channel exhibited a better performance, where it managed to reach its peak at nearly 800 rewards. Analysis on this interesting findings suggested that the agent trained on the red channel utilises the faint grey boxes (refer Fig. 2) found on the racetrack to determine the next progressive step it should take.

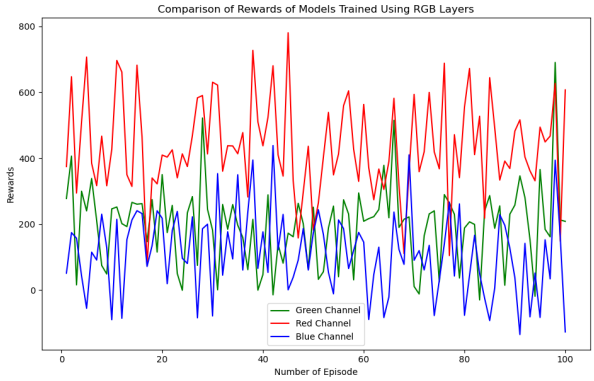

Figure 6: Comparison of rewards obtained by the trained agent after every episode.

6.3 Additional Experimentation

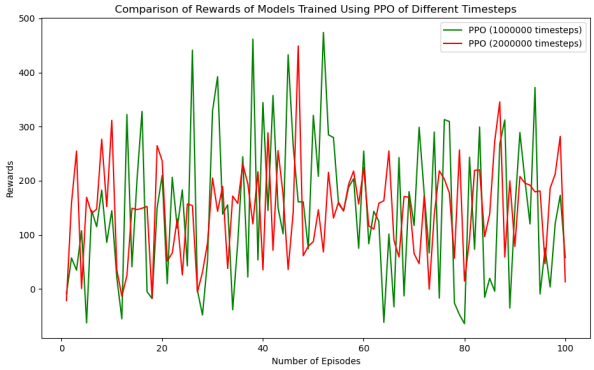

Figure 7: Comparison of rewards obtained by the agents trained using PPO of different timesteps.

With the exceptional performance exhibited by agents trained using the PPO algorithm. Extended experimentation has been conducted to investigate whether the number of timesteps used to train the agent will affect the performance of the agent and results have been illustrated in Fig. 7. As a result, it can be concluded that more timesteps used in agent training does not guarantee a better performance of an agent especially within an unseen environment.

7 Conclusion

As a summary, the work presented within this paper has covered the implementation of two distinct reinforcement learning algorithms to solve the CarRacing-v0 environment. Agents presented within this work have shown notable performances, given that proper pre-processing methodologies have been applied onto the observation space of the environment. Besides, the success of experimenting with the proposed research objectives (questions) accompanied by a series of well-established analysis is another key achievement of this project. Although the agent generally exhibits a good performance, there are situations when the agent will fail to keep itself on the racetrack. This limitation of the agent is especially obvious when the agent attempts a sharp cornering; in this case, the tendency for the agent to be zoomed off the racetrack is very high. As for the future directives, it would be great for this work to concentrate more on countering the impacts caused by the issue of catastrophic forgetting that is commonly found in reinforcement learning algorithms as it will certainly be beneficial to resulting in a performant agent. Lastly, it would also be ideal for the work to venture into more state-of-the-

art reinforcement learning algorithms, such as DeepRL which offers a much faster training time to a dynamic environment like this, producing an agent that can constantly complete the game.

References

[1] D. Mwiti, "10 Real-Life Applications of Reinforcement Learning," 11 2021. [Online]. Available: https://neptune.ai/blog/reinforcement-learning-applications

[2] R. Sutton and A. Barto, *Reinforcement Learning, Second Edition*, 2nd ed. Cambridge: The MIT Press, 2018.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 10 2017.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing Atari with Deep Reinforcement Learning," 12 2013.

[5] Y. Zhang and H. Sun, "Deep Reinforcement Learning with Mixed Convolutional Network," 10 2020.

[6] A. Fakhry, "Applying a Deep Q Network for OpenAIs Car Racing Game," 5 2020.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," 7 2017.

[8] J. Achiam, "Deep Deterministic Policy Gradient." [Online]. Available: https://spinningup.openai.com/en/latest/algorithms/ddpg.html

[9] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic Policy Gradient Algorithms," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ser. ICML'14. JMLR.org, 2014, p. I–387–I–395.

[10] A. PIERRÉ, "CarRacing-v0." [Online]. Available: https://github.com/openai/gym/blob/master/gym/envs/box2d/car_racing.py

[11] TensorFlow, "Replay Buffers." [Online]. Available: https://www.tensorflow.org/agents/tutorials/5_replay_buffers_tutorial#:~:text=Reinforcement%20learning%20algorithms%20use%20replay,%22replay%22%20the%20agent's%20experience.

[12] M. McCloskey and N. J. Cohen, "Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem," 1989, pp. 109–165.