

Explainable Reinforcement Learning via Rule Extraction in Complex Visual Environments

Anthony Manchin

December 29, 2022

Thesis submitted for the degree of

Doctor of Philosophy

in

Computer Science

at The University of Adelaide

Faculty of Engineering, Computer and Mathematical Sciences

School of Computer Science



THE UNIVERSITY
of ADELAIDE

Abstract

Deep neural networks have allowed for significant advances within the field of reinforcement learning and autonomous agents. However, in contrast to traditional approaches such as expert systems and hand-crafted control systems, deep neural networks introduce a large amount of ambiguity regarding the decision making of an autonomous agent. Understanding the decision-making process of any autonomous agent is crucial for applications where trusted autonomy is not only paramount, but required before an agent can be deployed. In this thesis, we focus on the following problems of explainability and rule extraction from autonomous agents. 1) How does the neural network architecture impact the performance and the explainability of an agent trained using reinforcement learning. 2) Can rules be defined and extracted from observations of an autonomous agent trained using reinforcement learning. 3) Can complex rules be derived from multiple partial observations of an autonomous agents.

For the first problem we investigate the common neural network architectures used in reinforcement learning and how attention mechanisms have been used to improve performance in prior works. We devise a novel spatial temporal attention-based approach that allows the agent to learn where it should focus its attention in contrast to previous works which favoured constraining networks with guided attention mechanisms.

For the second problem we propose a formal definition of a rule for trajectories

consisting of state and action pairs. We show that under this definition, rules are extractable using unsupervised learning techniques. Additionally, we investigate the impact of neural network design on an autonomous agent's ability to learn rules.

For the third problem we introduce a novel method for multi-sequence-to-sequence based tasks that require visual induction and translation. This method allows us to observe multiple partial visual observations of an agent and extract the over-arching rule set that defines the agent's behaviour. We also show that this method is robust with respect to noisy signals.

Declaration

I certify that this work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree.

I acknowledge that copyright of published works contained within this thesis resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Signed:

Date: 27/7/2022

Preface

This thesis was written at the School of Computer Science, The University of Adelaide. The main parts of the thesis are based on the following published/submitted papers in which I am the primary author:

1. A. Manchin, E. Abbasnejad, and A. van den Hengel, "Reinforcement learning with attention that works: A self-supervised approach," *In Proceedings of the International Conference on Neural Information Processing 2019*, pp. 223–230.
2. A. Manchin, J. Sherrah, Q. Wu, A. van den Hengel, "Noise Robust Visual Program Synthesis"; Submitted to IEEE/CVF Conference on Computer Vision and Pattern Recognition 2022, under review.

Dedication

I dedicate this thesis to my Father.

Acknowledgements

Firstly, I would like to thank my supervisors, **Professor Anton van den Hengel** and **Dr. Anthony Dick**. During my study I feel I have enjoyed a level of academic freedom not often afforded to other students. The level of trust they placed upon me, and the ability to collaborate so broadly with other researchers has meant a great deal to me during my PhD. Many of the unique and enriching experiences I have had the pleasure of enjoying over the last few years have stemmed directly from this trust. Add to this the support and guidance that was offered when needed, and I am truly grateful.

I would also like to single out **Dr. Ehsan Abbasnejad**. Your support throughout my PhD, whether we were actively collaborating or not, has been greatly appreciated. I'm sure many people have heard me vent my frustrations over the years, however, not many have done so with a smile on their face and honest discussions on the topics. Your guidance and encouragement during the first twelve months of my PhD helped me to develop the skills that have allowed me to get here today. Thank you.

I have also had the pleasure of making many great friends here in Adelaide. I would especially like to thank **Thomas Rowntree, Rafael Felix, and Michele Sasdelli** for making me feel welcome from the moment I arrived. If I were to name everyone that has made my PhD memorable, I would surely miss someone. I think this is a wonderful reflection on the quality of people AIML attracts, so I

will simply say this. It has been a pleasure working and making friends with so many wonderful people at AIML, and thank you all.

I would like to thank my family for all their support and encouragement throughout the years. In particular, I would like to thank my father **Paul Manchin**. Your constant encouragement and support in everything I have ever decided to do has given me the confidence and ultimately the ability to tackle any challenge. I could never thank you enough for the guidance and inspiration you have provided me over the years, which has lead me to where I am today, so thank you.

Lastly, I would like to thank my fiancé **Priscilla Jack**. Your unwavering love and support has meant the world to me, and I only stand here now as I have you by my side. While this PhD journey has been longer and harder in many ways than we first imagined, we have made it together. Words cannot express my gratitude for all you have done, for all your patience, for all your love. From the bottom of my heart, thank you.

Table of Contents

Abstract	iii
Declaration	v
Preface	vii
Dedicationviii
Acknowledgements	ix
Contents	xi
Chapter 1 . Introduction	1
1.1 Overview	1
1.2 Motivations	6
1.3 Contributions	9
1.4 Notation	10

1.5	Outline	11
Chapter 2 . Literature Review		17
2.1	Deep Reinforcement Learning	17
2.2	Attention based Reinforcement Learning	20
2.3	Neural Network Rule Extraction	23
2.4	Visual Program Synthesis	26
Chapter 3 . Reinforcement Learning with Attention that Works: A Self-Supervised Approach		34
3.1	Introduction	37
3.2	Related Work	39
3.2.1	Reinforcement Learning Network Design in Video Games	39
3.2.2	Attention in Reinforcement Learning	40
3.3	Methods	43
3.3.1	Markov Decision Process Formulation	43
3.3.2	Policy Optimisation	44
3.3.3	Non-Local Neural Networks	45
3.3.4	Network Architecture	46
3.4	Experiments	48
3.4.1	Validation Methodology	48
3.4.2	Performance results	49
3.5	Conclusions and Future Work	52
Chapter 4 . Unsupervised rule discovery with autonomous agents in visually complex environments		57
4.1	Introduction	59
4.2	Related Work	62

4.2.1	Policy Summarisation	62
4.2.2	Rule Extraction	64
4.3	Methods	65
4.3.1	Visual Evaluation	65
4.3.2	Rule Definition and Extraction	67
4.4	Experiments	68
4.4.1	Environments	68
4.4.2	Model Architecture	70
4.4.3	Results	71
4.5	Conclusion and Future Work	80
Chapter 5 . Deep Inductive Reasoning for Video to Program Translation. .		85
5.1	Introduction	88
5.2	Literature Review	91
5.2.1	Program Induction	92
5.2.2	Intrinsic Motivation	92
5.2.3	Program Synthesis	93
5.3	Method	94
5.3.1	Program Generation	94
5.3.2	VT4 Model	97
5.4	Experiments	102
5.4.1	Dataset and Metrics	102
5.4.2	Overall Performance	104
5.4.3	Noise Ablation Study	106
5.5	Conclusions and Future Work	107
Chapter 6 . Conclusion and Future Directions.114
6.1	Summary of the Contributions	114

- 6.2 Limitations and Future Directions 116
 - 6.2.1 Explainable Reinforcement Learning 116
 - 6.2.2 Program Synthesis 117
 - 6.2.3 Explainable Reinforcement Learning Via Program Synthesis 118
 - 6.2.4 Final Remarks 118

List of Tables

3.1	Experiment Results	51
3.2	Direct comparison between baseline PPO and SAN. Clearly demonstrating that our proposed method is capable of improving performance in 50% of tested environments.	51
5.1	An exact comparison of our results compared to the results of previously published works.	104
5.2	Exact and Alias program accuracy's for varying levels of perception noise.	107

List of Figures

1.1	A diagram of the Agent-Environment feedback loop for reinforcement learning.	2
1.2	A diagram of a Markov Decision Process	3
1.3	A collage of screenshots from 25 Atari environments available through the Arcade Learning Environment	4
3.1	A diagram of the convolutional neural network with embedded self-attention	46
3.2	Experiment Results	50
4.1	Baseline action and state trajectory plots for Demon Attack	72
4.2	Our approach action and state trajectory plots for Demon Attack	72
4.3	Baseline action and state trajectory plots for MsPacman	73
4.4	Our approach action and state trajectory plots for MsPacman	73
4.5	Example of temporal attention	76
4.6	Example of multiple points of foci	77
4.7	Examples of disappearing enemies in MsPacman	78
4.8	Attention map comparison between baseline and our method	79
5.1	An illustration of the task of visual program synthesis on the game 'Genshin Impact'	89

5.2	A Domain specific language program example from the Vizdoom environment. Examples of all component types are present.	97
5.3	A complete diagram of the VT4 architecture showing the semantic encoder and the program generator modules	98
5.4	A diagram of the Program Generator Module	100
5.5	A comparison of the ground truth program and a synthesised program from our VT4 model.	105
5.6	A plot comparing our results with prior works	105
5.7	A plot showing the impact of perception noise on our method . . .	107

CHAPTER 1

Introduction

In this chapter we provide an overview and motivation for the work presented in this thesis. We then summarise our key contributions and conclude with an outline for the layout of this thesis.

1.1 Overview

The field of Reinforcement Learning (RL) focuses on the science of autonomous agent decision making. The challenge it seeks to solve is to produce an agent that is capable of self-learning via its own experiences to maximise some form of reward. Like other subfields of machine learning, reinforcement learning is characterised by the learning problem, not the learning model [Sutton and Barto, 2018]. A reinforcement learning problem requires an agent, an environment, and a goal, and any method that can solve this problem may be considered a reinforcement learning method. By design, a reinforcement learning problem provides a framework which allows goal-directed learning tasks to be numerically optimised via computational approaches. This framework defines the interactions between a learning agent and its environment through the concepts of states,

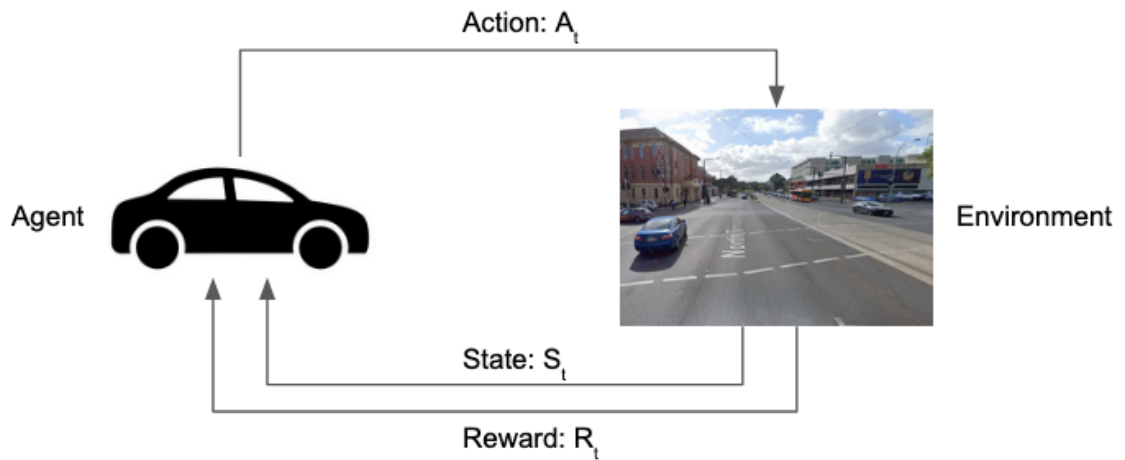


Figure 1.1: A diagram of the Agent-Environment feedback loop for reinforcement learning.

actions, and rewards. An agent is free to interact with the environment through select actions and receives from the environment a representation of the state along with a reward signal. A depiction of this framework is shown in Figure 1.1.

An important aspect of this framework is that it allows for problems to be modelled as a Markov Decision Processes [Markov, 1957] if the task satisfies the Markov property. The condition of which is met if the next state from the environment is only dependent on the current state (and action of the agent) and is independent of all previous states. A full derivation is provided in section 3.3, while Figure 1.2 illustrates the process. Many real-world problems from various fields can be represented as a Markov Decision Process including robotics, finance, manufacturing, and primary production. However, given the requirement of agent-environment interaction, studying these problems directly is often impractical. Instead, researchers often turn to various games as a stand-in as they can easily be modelled as a Markov Decision Process.

An early example of a successful reinforcement learning agent was that of

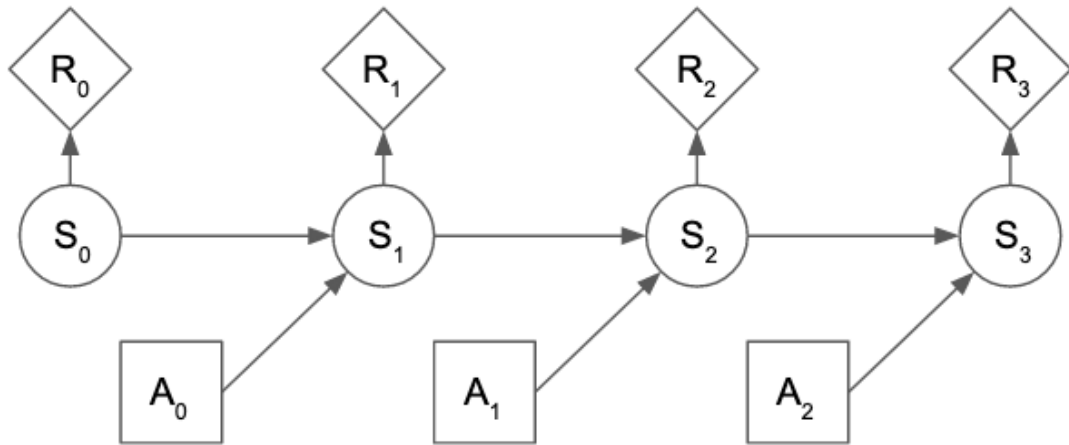


Figure 1.2: A diagram of a Markov Decision Process

TD-Gammon [Tesauro, 1995], which at the time of its release was able to challenge the top human players in the world at the game of backgammon. However, the success of TD-Gammon was largely due to human crafted domain-dependent features that allowed for accurate value function estimates. This approach of hand crafting features to represent the state of the environment was unable to produce the same level of results across other games such as Checkers, Chess, or Go.

In 2012 the Arcade Learning Environment (ALE) was released as both a challenge problem and a platform for evaluating domain-independent reinforcement learning agents [Bellemare et al., 2013]. Built on top of an open-source Atari 2600 emulator, ALE presents researchers with access to hundreds of game environments. The sheer diversity of games and complexity of the image-based state representations largely prohibits hand crafting inputs and poses a significant challenge for researchers.

In 2013 [Mnih et al., 2013] proposed Deep Q-Network (DQN) as a solution for ALE. It was the first reinforcement learning algorithm to successfully perform across multiple environments without adjustments to either the model architec-

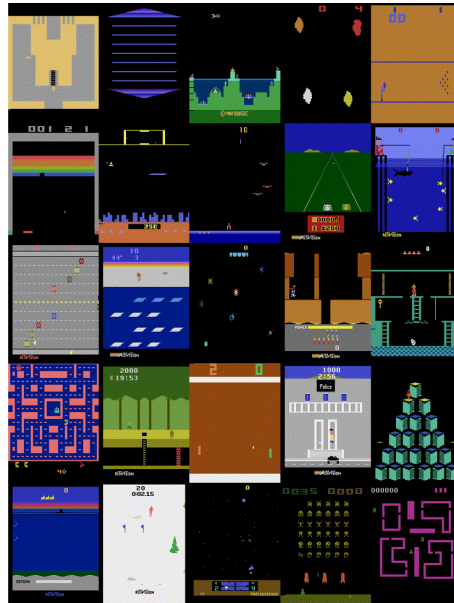


Figure 1.3: A collage of screenshots from 25 Atari environments available through the Arcade Learning Environment

ture or learning algorithm. DQN combined deep learning with Q-learning by using convolutional neural networks [Lecun et al., 1998] to approximate the value function. Originally applied to seven games, it was able to outperform all previous approaches in six environments and achieve superhuman results on three. Further experiments on a larger set of 49 environments [Mnih et al., 2015] showed that DQN was able to perform at or above human level in 29 environments and outperform all previous approach in 44. This achievement has since inspired many breakthroughs in deep reinforcement learning research, upon which ALE has been a vital and robust proving ground.

A popular area of focus for reinforcement learning research has always been that of the learning algorithm. Many new algorithms of recent years have been empirically validated using ALE. These algorithms can be split into two types: on-policy and off-policy. The distinguishing feature between these two types is that on-policy methods estimate the value of, and optimise a policy whilst using

it to control an agent, while an off-policy method separates these two functions. In an off-policy setting, the policy that is used to generate the behaviour of the agent may be completely unrelated to the policy that is evaluated and improved. As DQN is an off-policy method, extensions to this algorithm have either taken advantage of the separation of functions [Schaul et al., 2015, Wang et al., 2015], or sought to address known limitations [van Hasselt et al., 2015, Fortunato et al., 2018]. These extensions were unified by [Hessel et al., 2017] which showed that they could all be applied simultaneously and lead to significant improvements in performance.

Deep neural networks have also been successfully combined with on-policy methods and validated with ALE. By adopting the same neural network architecture as originally used by [Mnih et al., 2015] various actor-critic based methods have achieved state-of-the-art results in numerous studies [Mnih et al., 2016, Haarnoja et al., 2018]. These methods involve using an actor to select an action, while feedback from a critic is used to update the actor’s policy. Whilst separate networks can be used for both the actor and the critic, it is common practise to use a single shared network with two different heads. However, on-policy methods often experience slow sample rates due to the requirement to sample from the policy directly. This can also lead to high variance and instability during parameter updates. Methods such as trust-region policy updates [Wang et al., 2016, Schulman et al., 2015] are a common way to address the instability issue, although they do not offer any benefit with respect to training time. Addressing both issues [Schulman et al., 2017] proposed Proximal Policy Optimisation which alternates between generating experience and optimising a ‘surrogate’ objective function. We refer the reader to section 3.3 for a full derivation.

As a result of this success, applications of these algorithms into real world

scenarios have been increasing in recent years. Driverless cars for example, is one area in which reinforcement learning is heavily being researched. However, the combination of unstructured state representations and the application of deep neural networks within reinforcement learning has largely come at the cost of the explainability of the agent. The outputs of deep neural networks can be difficult to explain, and the more complex or convoluted the network, the more difficult the task. This presents major problems in situations where the explanation of the decision is just as important as the decision itself.

1.2 Motivations

For reinforcement learning models to be widely adopted, the arising issues surrounding the explainability of their decision making needs to be addressed. This lack of transparency has been the cause of much debate around driverless cars and other areas where trusted autonomy is essential. An ethical thought experiment (often referred to as the Trolley Problem [Foot, 1967] regarding what actions a driver should take in different scenarios is often raised as a part of the wider discussion on driverless cars. The significance of this cannot be disparaged as legislation in many countries is increasingly focusing on assigning responsibility to the manufacturers or engineers for the outcomes of automated processes. Systems that are unable to explain their decisions may find themselves at risk of being in breach of these laws.

One approach to providing explainability for neural networks is to define its behaviour as a set of rules. While researchers have sought to extract rules from simple neural networks [Sato and Tsukimoto, 2001], and recently some deep neural networks [Zilke et al., 2016], a very limited amount of work has been done

to extract rules from, and explain, deep convolutional neural networks used in a reinforcement learning setting. This problem is a lot harder to solve as it requires considering additional factors that models which operate in static environments do not need to address. Specifically, recently successful on-policy reinforcement learning approaches utilise a value function that attempts to predict how good or 'valuable' the current state is to be in. As this value function is commonly approximated by the same convolutional neural network that is used for the policy, understanding the reasoning behind the decisions of the agent is difficult. For example, in a shared network setup, when an agent decides to go 'left', it is currently impossible to determine if it is doing so because it believes it will receive an immediate reward, or if it believes that by doing so it will increase the probability of some future reward. This makes the extraction of semantic reward-linked rule extraction an ill-posed problem. For this reason, we are interested in defining relationships between states and actions (not state-action pairs and rewards). This allows us to focus on what the policy does in practise, as opposed to what the policy may be trying to achieve. As such, we consider the architecture of the policy fundamental to both the performance of the model and its explainability. We set out to shed light on the topic of explainable reinforcement learning and rule extraction in complex visual environments.

Whilst policies consisting of convolutional and fully connected layers have achieved great results, they effectively separate the tasks of learning local connectivity and global connectivity between the different layers. However, the ability to relate various spatial features with others regardless of position in both time and space, is often critical for many tasks. For this reason, we find the separation of duties counter intuitive. If the relationship between spatial features is just as important as the features themselves, would it not be better to learn these embed-

dings simultaneously? This question motivates us to explore recently proposed neural architectures known as Transformer (also known as Attention) networks. These networks learn mappings between a query and key-value pairs [Vaswani et al., 2017] and have recently been shown to improve spatial-temporal reasoning in action detection tasks [Wang et al., 2018]. With recent advances in computing capabilities, neural architectures that were previously infeasible are quickly becoming viable options. This allows us to propose policy networks that incorporate global connectivity in the convolutional layers via global attention mechanisms, as opposed to only the fully connected layers. We believe this improved ability to learn global-relational embeddings throughout the network should result in not only better performance of a trained agent, but also more distinguishable mappings between states and actions. This directly relates to our desired goal of producing more explainable DRL agents.

Furthermore, as we believe rules are a constrained definition of a mapping between states and actions, it should be possible to produce a general definition for a rule in terms that are agnostic to the agent or the environment. A generalised rule definition such as this has the potential to form the basis for improved explainability for any autonomous system. If this is indeed the case, then we should also be able to leverage the global connectivity of attention-based networks to extract rules sets that define other types of autonomous systems by simply observing their behaviour. This motivates us to also explore the problem of rule extraction from other types of autonomous systems operating in visually complex environments.

1.3 Contributions

In this thesis we present several different contributions to the field of explainability for autonomous agents as outlined below.

- We investigate the impact of the neural network architecture on autonomous agents trained using state-of-the-art reinforcement learning techniques. Here we are able to make a novel contribution regarding the integration of self-attention mechanisms with convolutional networks for the purpose of improving global connectedness throughout the embedding space for spatio-temporal reasoning in deep reinforcement learning agents. We explore the impact of various designs across multiple visually complex environments and are able to show that not only does our contribution improve the explainability of an agent through improved rule extraction, but it also provides an increase in sample efficiency and overall performance.
- We propose a novel method of defining and extracting rules from autonomous agents wherein one does not have access to a ground-truth mapping from inputs to outputs. In particular, we show that it is not sufficient to simply consider the relationships between single state-action instances when extracting or defining the rules learnt by an autonomous agent. Instead, one must consider a trajectory of state-action pairs over a period of time.
- We propose a multi-sequence-to-sequence based approach for generating executable programs from multiple demonstrations of an autonomous agent following deterministic policies in visually complex environments. This work focuses on extracting a structured rule set wherein no single series of observations of the agent contains all the relevant information to achieve

the task. Our approach utilises the creation of a visual language paired with transformer based networks which achieves significant improvements in the explainability of the agent over prior state-of-the-art methods through improved rule extraction, while drastically simplifying the model in the process.

1.4 Notation

Here we introduce a summary of the notation used in this thesis.

t	discrete time step
T	final time
s_t	state at t
a_t	action at t
r_t	reward at t
\mathcal{S}	set of all states
\mathcal{A}	set of all possible actions
\mathcal{P}	state transition probability matrix
\mathcal{R}	expected reward function
π	policy (decision making rule, program)
θ	vector of parameters
π_θ	policy guided by parameter vector θ
$\pi_\theta(a_t s_t)$	probability of taking action a in state s at time step t under policy π_θ
δ_t	temporal-difference error at t
γ	discount-rate parameter
λ	smoothing parameter
τ	sequence of vectors

\mathcal{T}	set of sequences τ
\mathfrak{T}	set of all sets of sequences \mathcal{T}
ψ	integer tokens
Ψ	set of all integer tokens
μ	boolean value

1.5 Outline

In this section we outline the structure of this thesis.

In Chapter 1 we present an overview, motivations, and contributions provided by this thesis.

In Chapter 2 we provide a literature review of previously published works in the field of Explainable Reinforcement Learning, and the related fields of Rule Extraction and Program Synthesis.

In Chapter 3 we propose a novel combination of spatio-temporal self-attention with deep reinforcement learning. We evaluate our contribution in a suite of visually complex and dynamic environments against prior state-of-the-art models. We show our model is able to exceed prior works and achieve new state-of-the-art.

In Chapter 4 we introduce a formal definition for ‘rules’ as learnt by deep reinforcement learning agents (especially those operating in visually complex environments). We provide a straight forward methodology for discovery and extraction of rules under our proposed definition. Additionally, we study the impact of our previously proposed self-attention integration has with respect to rule discovery.

In Chapter 5 we propose a multi-sequence-to-sequence approach for program synthesis from a diverse set of visual observations. We explore the challenges

involved with specification generation when dealing with a visual domain, and propose a novel solution that learns to simultaneously summarise and translate visual information into an executable program. Additionally, we provide a noise ablation study to stress test our framework and show that it is highly robust to noisy perceptions.

In Chapter 6 we conclude the contributions within this thesis and discuss future works.

Bibliography

- [Bellemare et al., 2013] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.
- [Foot, 1967] Foot, P. (1967). The problem of abortion and the doctrine of the double effect. *Oxford Review*, 5:5–15.
- [Fortunato et al., 2018] Fortunato, M., Azar, M. G., Piot, B., Menick, J., Hessel, M., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. (2018). Noisy networks for exploration. In *International Conference on Learning Representations*.
- [Haarnoja et al., 2018] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Dy, J. G. and Krause, A., editors, *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR.
- [Hessel et al., 2017] Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2017). Rainbow: Combining improvements in deep reinforcement learning. cite arxiv:1710.02298Comment: Under review as a conference paper at AAI 2018.

- [Lecun et al., 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- [Markov, 1957] Markov, A. A. (1957). Theory of algorithms. *Journal of Symbolic Logic*, 22(1):77–79.
- [Mnih et al., 2016] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA. PMLR.
- [Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.
- [Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.
- [Sato and Tsukimoto, 2001] Sato, M. and Tsukimoto, H. (2001). Rule extraction from neural networks via decision tree induction. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, volume 3, pages 1870–1875 vol.3.

- [Schaul et al., 2015] Schaul, T., Quan, J., Antonoglou, I., and Silver, D. (2015). Prioritized experience replay. cite arxiv:1511.05952Comment: Published at ICLR 2016.
- [Schulman et al., 2015] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France. PMLR.
- [Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.
- [Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.
- [Tesauro, 1995] Tesauro, G. (1995). Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68.
- [van Hasselt et al., 2015] van Hasselt, H., Guez, A., and Silver, D. (2015). Deep reinforcement learning with double q-learning. cite arxiv:1509.06461Comment: AAAI 2016.
- [Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

- [Wang et al., 2018] Wang, X., Girshick, R., Gupta, A., and He, K. (2018). Non-local neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7794–7803.
- [Wang et al., 2016] Wang, Z., Bapst, V., Heess, N., Mnih, V., Munos, R., Kavukcuoglu, K., and de Freitas, N. (2016). Sample efficient actor-critic with experience replay. *CoRR*, abs/1611.01224.
- [Wang et al., 2015] Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. cite arxiv:1511.06581Comment: 15 pages, 5 figures, and 5 tables.
- [Zilke et al., 2016] Zilke, J. R., Loza Mencía, E., and Janssen, F. (2016). Deepred – rule extraction from deep neural networks. In Calders, T., Ceci, M., and Malerba, D., editors, *Discovery Science*, pages 457–473, Cham. Springer International Publishing.

CHAPTER 2

Literature Review

In this chapter we first review the literature around attention incorporated neural network policy designs for reinforcement learning. We then describe and highlight the gaps in current research with respect to neural network rule extraction and program synthesis from visual stimuli.

2.1 Deep Reinforcement Learning

The Arcade Learning Environment poses a significant challenge for reinforcement learning agents. The unstructured state representation consisting of video frames and the requirement for the learning algorithm and architecture to remain unchanged across a diverse set of games largely prevents previously popular techniques of hand-crafting domain specific input features. This combination of challenges, along with technical advances and increases in GPU computing capabilities [Krizhevsky et al., 2012] lead to the development of the modern era of deep reinforcement learning.

The first algorithm able to outperform human baselines across a range of games within ALE was DQN [Mnih et al., 2013]. The approach combined deep convo-

lutional neural networks with Q-learning [Watkins and Dayan, 1992] to learn a policy solely on video input. Although shallow neural networks had been used previously to estimate the value function in model-free reinforcement learning algorithms such as TD-Gammon [Tesauro, 1995], attempts to replicate its success across different domains had been unsuccessful. DQN differed from TD-Gammon in several ways. Firstly, DQN approximated the action-value function $Q(s, a)$ while TD-Gammon approximated the state value function $V(s)$. Secondly, TD-Gammon learnt in an on-policy manner directly from self-play games, while DQN takes an off-policy approach which allows for previous transitions to be randomly resampled using an experience replay mechanism [Lin, 1992]. This resampling smooths the training distribution across many previous behaviours and had previously only been combined with Q-learning while using low-dimensional state representations by [Lin, 1992]. The deep convolutional neural network architecture used in the seminal work by [Mnih et al., 2013] took as input a stack of the four previous timesteps so that temporal information would be included in the state representation. This architecture along with the process of stacking video frames became standard practice for papers that expanded DQN, along with other reinforcement learning algorithms that were applied to ALE.

Other reinforcement learning approaches were also successfully combined with deep convolutional neural networks and empirically evaluated using ALE. Using the same base neural network architecture as proposed by [Mnih et al., 2013], and [Mnih et al., 2016] was able to demonstrate that on-policy actor-critic methods (a type of policy gradient method) could also surpass human baselines. Asynchronous Advantage Actor-Critic (A3C) maintains both a policy $\pi_{\theta}(a_t|s_t)$ (actor) and an estimate of the value function $V_{\theta}(s_t)$ (critic) with both functions making use of the same base network parameters in practice, with a softmax

output used for the action policy and a linear output used for the value function. The critic learns to estimate the value of a state based on the expected sum of discounted future rewards. The error of this prediction, known as the Advantage, is then used to guide the policy towards selecting actions that lead to better states. Additionally, [Mnih et al., 2016] were able to incorporate the entropy of the policy into the objective function as originally proposed by [Williams and Peng, 1991] to increase exploration and discourage premature convergence to suboptimal deterministic policies. Empirical results showed that when using the model architecture originally proposed by [Mnih et al., 2013] A3C was comparable to many similarly trained DQN based networks. However, the addition of a Long Short-Term Memory (LSTM) component to the network resulted in superior results across the majority of environments. Furthermore [Mnih et al., 2016] highlight the significance of architecture improvements and the possibility for future work in this area.

One drawback to optimising policy gradient methods is the requirement to sample transitions directly from the policy. This results in slow training times as taking multiple optimisation steps from the same trajectory often leads to divergence. To address this [Schulman et al., 2017] proposed Proximal Policy Optimisation (PPO) which allows for multiple parameter updates to be taken from the same trajectory. Sharing some similarities with Trust Region Policy Optimisation (TRPO) [Schulman et al., 2015], PPO optimises a surrogate objective. The advantage of this is it allows for pessimistic bound to be taken between a clipped and an unclipped objective. A detailed derivation is given in section 3.3. The final objective is further augmented to include the loss for the value function for actor-critic models, along with an entropy bonus to encourage exploration of the state space. Empirical results from experiments in ALE showed that PPO

implemented with the same architecture as proposed by [Mnih et al., 2013] can match that of other policy gradient methods whilst achieving a higher sample efficiency.

While the significance of these improvements across various reinforcement learning algorithms cannot be overstated, the goal of ALE was not to diminish the importance of the learning architecture. Many augmentations that have allowed for improvements in the field of deep reinforcement learning such as experience replay [Lin, 1992], conservative policy optimisation [Kakade and Langford, 2002], and entropy bonuses [Williams and Peng, 1991] had been shown to work with shallow neural networks in constrained domains. The success of a reinforcement learning agent, regardless of its learning algorithm, is bound by its ability to accurately estimate some form of value function. TD-Gammon was successful because its neural network architecture was able to learn useful relations from the state representations supplied. Similarly, the above discussed learning algorithms have been able to perform well in ALE due the ability of convolutional neural networks to learn useful embeddings from unstructured visual inputs. Despite scope for improvement in this area being correctly identified by [Mnih et al., 2016], this area of research has received less attention than the former in recent years.

2.2 Attention based Reinforcement Learning

One promising direction for improving the neural network architecture for deep reinforcement learning agents is to incorporate attention mechanisms. Proposed by [Vaswani et al., 2017], attention mechanisms learn mappings between queries and key-values which allows for global or local relations to be learnt. Although originally developed for use with recurrent neural networks in machine translation

tasks, attention mechanisms can be applied in conjunction with convolutional neural networks.

An early combination of attention mechanisms with reinforcement learning was proposed by [Sorokin et al., 2015], which studied the effects of adding both soft and hard attention to a neural network policy. The difference between soft and hard attention is that soft attention calculates the context vector as a weighted sum of its inputs, while hard attention uses an 'attention score' to select a single input for the context vector. Sorokin *et al.* tested their approach on a network that consisted of a convolutional network that was paired with a recurrent neural network (RNN). The RNN allowed them to use just a single image as input into the CNN, as it would be responsible for encoding and remembering the relevant temporal and dynamic information of the state. The attention module received spatial information from the CNN and temporal information from the RNN. Despite testing both soft and hard attention integration's, Sorokin *et al.* only achieved increased performance on two out of five evaluated ALE domains. Additionally, the separation of spatial and temporal information makes the network harder to understand as the temporal information is encoded in latent space and therefore unable to be visualised.

While investigating navigation policies, [Choi et al., 2017] also proposed combining attention mechanisms with reinforcement learning. Their approach employed a Multi-focus Attention Network which used multiple parallel attention modules. This worked by segmenting the input, with each parallel attention layer attending to a different segment. To evaluate this framework the authors developed a custom synthetic grid-world environment and testing their model against a standard implementation of DQN. By decomposing the input into partial states, their model was able to compute multiple parallel attention heads. With

this approach they showed a 20% increase on sample efficiency over the baseline. However, the segmentation of the input was only possible due to the carefully constructed environment in which their idea was evaluated on. It is unclear that the same methodology could be applied in environments that involve complex visual inputs.

Taking inspiration from the way humans play games, [Zhang et al., 2018] proposed an attention-guided imitation learning framework. A dataset was created depicting where humans would look when playing a game before a model was trained to replicate these 'gaze heat maps'. The input would then be augmented with this additional information before being passed into a DQN model. While this method of attention fundamentally differs from soft and hard attention mechanisms, it is still worth noting. During evaluation Zhang *et al.* were able to show that their method was able to strongly outperform Imitation learning models across a total of eight environments. When compared to a standard DQN model, their approach significantly under performed in four out of eight environments, achieved similar performance in two other environments, and surpassed the baseline only twice.

Also inspired by the way humans perceive their environments, [Yuezhong et al., 2018] proposed an approach to construct attention maps based on the optical flow between sequential frames. The approach was based upon the Broadbent filter model [Broadbent, 1958] and combined the calculated attention map with the output of the last convolutional layer in the policy's network. Evaluated on a toy problem referred to as 'Catch' by the authors an agent was required to learn how to catch a falling ball. In this setting the addition of optical flow attention provided no benefit over the baseline DQN model. In a second variation, noise was added to the background to obfuscate the position of the ball as it fell. This increased

the significance of the optical flow attention map in the model and resulted in improved performance over the baseline. Evaluations on four domains from ALE were unable to demonstrate significant improvements in either performance or sample efficiency.

From evaluating these works we can identify commonalities and gaps that have not been addressed. A familiar theme is the use of guided attention mechanisms which actively enforce preconceived human notions of importance on the policies. Second is the heavy use of local attention mechanisms. Our proposed contributions in Chapter 3 address these limitations integrating a non-local (global) form of self-attention that is not bound or directed by human input.

2.3 Neural Network Rule Extraction

Neural network rule extraction methods can be divided into three main categories: pedagogical, decompositional, and eclectic. Pedagogical rule extraction is generally model-agnostic and treats the neural network as a black-box. The goal of pedagogical rule extraction is to define a set of rules that maps the input of a neural network to its output by approximating its global function. A popular approach to this is Validity Interval Analysis (VI-Analysis) [Thrun, 1995]. VI-Analysis seeks to determine intervals for input features which trigger changes in the activations of a neural networks output layer. The found intervals are then converted to rules in the form of *'if x then y'*. Thrun demonstrates his algorithm by extracting rules from a neural network representing the XOR problem, and an example of a continuous robot arm.

Another popular pedagogical approach known as Sampling generally involves creating an artificial training set, which is then used to feed a rule-based learning

algorithm to generate rules that mimic the original neural networks behaviour. In particular, [Craven and Shavlik, 1996] proposed the algorithm Trepan, which seeks split points in the training data to separate instances of different classes. Although similar to the C4.5 algorithm [Quinlan, 1993], it includes additional M-of-N style split points, an ability to sample deeper points in the tree, and instead of following a depth-first strategy it uses a best-first tree expansions strategy instead. More recently, [Sethi et al., 2012] proposed KDRuleEx which bears similarities to Trepan. However, unlike the work of [Craven and Shavlik, 1996], new training examples are produced using a genetic algorithm which results in a decision table before being transformed into *if-then* rules.

The last type of pedagogical rule extraction method we wish to highlight here is the algorithm known as RxREN which was proposed by [Augasta and Kathirvalavakumar, 2012]. The algorithm works in two stages. The first stage is pruning neurons which prove to be insignificant to a classification decision. The second stage then uses the ranges discovered in stage one to determine rules for each classification and refines these rules through the process of rule pruning and updation.

In contrast to pedagogical approaches, decompositional methods extract rules from neural networks at the neuron level. Typically, rules are extracted for each neuron in a network before aggregation techniques are used to form a composite rule base that describes the neural network as a whole [Andrews et al., 1995]. Examples of this include [Fu, 1994] and [Tsukimoto, 2000] who both present layer-by-layer decompositional algorithms that extract *if-else* rules for every single neuron in a network. More recently [Zilke et al., 2016] proposed the algorithm DeepRED, an extension of the CRED algorithm [Sato and Tsukimoto, 2001], which itself is an extension of the C4.5 algorithm. The DeepRED algorithm starts by

using the C4.5 algorithm to create decision trees for all the split points on the activations of the last hidden layer in a neural network. After this they refer to the next hidden layer in the network and so on and so forth until they have generated a rule sets that describe each layer by their respective preceding layer. After this is completed, they then merge all the rule sets into a single rule set that can describe the relationship between inputs and the output classifications.

Eclectic approaches incorporate both decompositional and pedagogical elements to extract rules. Early work in this area was performed by [Tickle et al., 1994] who proposed the Decision Detection (DEDEC) algorithm. This algorithm worked in two steps. First it would identify the dependencies between the inputs and outputs of a neural network according to the node activations. Second, it would learn a symbolic representation in a pedagogical manner. This approach of first identifying characteristics of the network before learning a symbolic representation was influential on other eclectic methods that followed.

An example is Rule Extraction from Artificial Neural Networks (REANN) proposed by [Kamruzzaman and Islam, 2010]. Their approach included first adding, then subtracting nodes from a network based upon the performance during training. Once the network was pruned to a simplified state, the activation values of hidden nodes were discretised via a heuristic clustering algorithm. This then allowed for 'if-else' style rules to be extracted by clustering activation's based on the classification outputs and the decision boundaries that separate them. Another is Hierarchical and Eclectic Rule Extraction via Tree Induction and Combination (HERETIC) [Iqbal, 2011]. This approach utilised steep sigmoid activation functions to approximate a step function. The purpose of this was so that each activation would essentially be binary allowing for a simple discretisation process. From here the authors use the C4.5 algorithm [Quinlan, 1993] to generate

the decision tree before using the espresso logic minimisation algorithm [Brayton et al., 1984] to produce simplified rules in a disjunctive normal form.

While there may be significant differences between the three different ideological approaches, the above works all share common traits. First, they all operate on simple neural networks with at most a few hidden layers. Second, the inputs to these neural networks only consist of 1D structured feature arrays. Our work differs by focusing on extracting rules from more complex neural networks which include convolutional layers and attention mechanisms. Our inputs are also multi-dimensional unstructured arrays in the form of multiple RGB images which makes feature level decompositional approaches impractical. The methods we propose in Chapter 4 are to the best of the authors knowledge not only novel, but also the first work aimed at extracting rules from complex deep neural networks that utilise unstructured RGB inputs for the purpose of learning a policy via reinforcement learning.

2.4 Visual Program Synthesis

The field of Visual Program Synthesis is relatively new, with the first problem in this area being proposed by [Sun et al., 2018]. Having concluded that the body of published research focused on traditional Program Synthesis is not applicable to situations where the derivation of specifications is required from visual observations, Sun *et al.* proposed a dataset addressing this. Specifically, state-action trajectories of autonomous agents with deterministic policies were generated and collected. The policies consisted of program constructed from a domain specific language (DSL). We find this to be an interesting problem and an extension of our previous work in the following way. To extract the rules from observations of an

agent, a correlation between states and actions is required to be learnt. Also, as the underlying programs were non-trivial, a model would need to piece together multiple observations to extract these rules. In many ways, this is an extension of our own rule extraction from state-action correlations.

Along with the dataset Sun *et al.* proposed a LSTM based encoder-decoder network. The network would first process each demonstration (trajectory of states) creating a unique embedding for each one. Then, the model would condense these embeddings into a single compact vector representation via a combination of average pooling and a relation network [Santoro et al., 2017]. While this approach set a respectful baseline on this new problem, its summarisation module was computationally expensive. This is something that Duan *et al.* noticed and addressed with their model 'Watch Reason Code' [Duan et al., 2019]. By replacing the summariser proposed by Sun *et al.* with a deviation-pooling method, Duan *et al.* were able to reduce the computational complexity, and therefore training time, while achieving similar results. Additionally, they included a multi-pass decoder which increased the accuracy of their model. However, it is reasonable to suspect that had this multi-pass decoder been applied to the model of Sun *et al.*, it too would have received a slight increase in performance.

Dang-Nhu [Dang-Nhu, 2020] took a different approach to this problem. Instead of using a LSTM encoder-decoder, Dang-Nhu uses a CNN with a perception head and an action head to generate the specifications for a rule-based synthesizer. As rule-base solvers are very sensitive to even small amounts of input noise [Devlin et al., 2017], Dang-Nhu proposed using a dynamic filtering method to ignore demonstrations based upon the level of confidence of the CNN's predictions. With this Dang-Nhu was able to surpass the performance of the previous works of Sun *et al.* and Duan *et al.*. The approaches of Sun *et al.* and Duan *et al.*, differ significantly

compared to the approach of Dang-Nhu and it is in this difference that we reveal a gap in the current literature. Sun *et al.* and Duan *et al.* both compress the problem via summarisation to a single sequence-to-sequence problem, while Dang-Nhu's method requires summarisation via a dynamic filtering due to inherent noise sensitivity. The question then becomes, can we create a model that is able to learn this summarisation whilst simultaneously being robust to noisy inputs? This question inspired our work in Chapter 5 where we propose a framework that treats the problem as a multi-sequence-to-sequence task and is capable of simultaneous summarisation and translation while being robust to noisy inputs.

Bibliography

- [Andrews et al., 1995] Andrews, R., Diederich, J., and Tickle, A. B. (1995). Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-based systems*, 8(6):373–389.
- [Augasta and Kathirvalavakumar, 2012] Augasta, M. G. and Kathirvalavakumar, T. (2012). Reverse engineering the neural networks for rule extraction in classification problems. *Neural processing letters*, 35(2):131–150.
- [Brayton et al., 1984] Brayton, R. K., Hachtel, G. D., McMullen, C., and Sangiovanni-Vincentelli, A. (1984). *Logic minimization algorithms for VLSI synthesis*, volume 2. Springer Science & Business Media.
- [Broadbent, 1958] Broadbent, D. E. (1958). Perception and communication.
- [Choi et al., 2017] Choi, J., Lee, B.-J., and Zhang, B.-T. (2017). Multi-focus attention network for efficient deep reinforcement learning. *ArXiv*, abs/1712.04603.
- [Craven and Shavlik, 1996] Craven, M. and Shavlik, J. (1996). Extracting tree-structured representations of trained networks. In Touretzky, D., Mozer, M. C., and Hasselmo, M., editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press.
- [Dang-Nhu, 2020] Dang-Nhu, R. (2020). Plans: Robust program learning from neurally inferred specifications. *ArXiv*, abs/2006.03312.

- [Devlin et al., 2017] Devlin, J., Uesato, J., Bhupatiraju, S., Singh, R., rahman Mohamed, A., and Kohli, P. (2017). Robustfill: Neural program learning under noisy i/o. In *ICML*.
- [Duan et al., 2019] Duan, X., Wu, Q., Gan, C., Zhang, Y., Huang, W., van den Hengel, A., and Zhu, W. (2019). Watch, reason and code: Learning to represent videos using program. In *Proceedings of the 27th ACM International Conference on Multimedia, MM '19*, page 1543–1551, New York, NY, USA. Association for Computing Machinery.
- [Fu, 1994] Fu, L. (1994). Rule generation from neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(8):1114–1124.
- [Iqbal, 2011] Iqbal, R. A. (2011). Eclectic extraction of propositional rules from neural networks.
- [Kakade and Langford, 2002] Kakade, S. and Langford, J. (2002). Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning, ICML '02*, page 267–274, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- [Kamruzzaman and Islam, 2010] Kamruzzaman, S. M. and Islam, M. M. (2010). Extraction of symbolic rules from artificial neural networks. *CoRR*, abs/1009.4570.
- [Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, NIPS'12*, page 1097–1105, Red Hook, NY, USA. Curran Associates Inc.

- [Lin, 1992] Lin, L.-J. (1992). *Reinforcement Learning for Robots Using Neural Networks*. PhD thesis, USA. UMI Order No. GAX93-22750.
- [Mnih et al., 2016] Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In Balcan, M. F. and Weinberger, K. Q., editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA. PMLR.
- [Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.
- [Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- [Santoro et al., 2017] Santoro, A., Raposo, D., Barrett, D. G. T., Malinowski, M., Pascanu, R., Battaglia, P. W., and Lillicrap, T. P. (2017). A simple neural network module for relational reasoning. In *NIPS*.
- [Sato and Tsukimoto, 2001] Sato, M. and Tsukimoto, H. (2001). Rule extraction from neural networks via decision tree induction. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 3, pages 1870–1875. IEEE.
- [Schulman et al., 2015] Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. (2015). Trust region policy optimization. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France. PMLR.

- [Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.
- [Sethi et al., 2012] Sethi, K. K., Mishra, D. K., and Mishra, B. (2012). Kdruleex: A novel approach for enhancing user comprehensibility using rule extraction. In *2012 Third International Conference on Intelligent Systems Modelling and Simulation*, pages 55–60.
- [Sorokin et al., 2015] Sorokin, I., Seleznev, A., Pavlov, M., Fedorov, A., and Ignatova, A. (2015). Deep attention recurrent q-network. *ArXiv*, abs/1512.01693.
- [Sun et al., 2018] Sun, S.-H., Noh, H., Somasundaram, S., and Lim, J. (2018). Neural program synthesis from diverse demonstration videos. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4790–4799. PMLR.
- [Tesauro, 1995] Tesauro, G. (1995). Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68.
- [Thrun, 1995] Thrun, S. (1995). Extracting rules from artificial neural networks with distributed representations. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems (NIPS) 7*, Cambridge, MA. MIT Press.
- [Tickle et al., 1994] Tickle, A. B., Orłowski, M., and Diederich, J. (1994). Dedec: decision detection by rule extraction from neural networks. *QUIT NRC*.
- [Tsukimoto, 2000] Tsukimoto, H. (2000). Extracting rules from trained neural networks. *IEEE Transactions on Neural networks*, 11(2):377–389.

- [Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.
- [Watkins and Dayan, 1992] Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.
- [Williams and Peng, 1991] Williams, R. J. and Peng, J. (1991). Function optimization using connectionist reinforcement learning algorithms. *Connection Science*, 3:241–268.
- [Yuezhang et al., 2018] Yuezhang, L., Zhang, R., and Ballard, D. H. (2018). An initial attempt of combining visual selective attention with deep reinforcement learning. *ArXiv*, abs/1811.04407.
- [Zhang et al., 2018] Zhang, R., Liu, Z., Zhang, L., Whritner, J. A., Muller, K. S., Hayhoe, M. M., and Ballard, D. H. (2018). Agil: Learning attention from human for visuomotor tasks. In *ECCV*.
- [Zilke et al., 2016] Zilke, J. R., Loza Mencía, E., and Janssen, F. (2016). Deepred – rule extraction from deep neural networks. In Calders, T., Ceci, M., and Malerba, D., editors, *Discovery Science*, pages 457–473, Cham. Springer International Publishing.

CHAPTER 3

Reinforcement Learning with Attention that Works: A Self-Supervised Approach


The work contained in this chapter has been published as the following paper:

Manchin A., Abbasnejad E., van den Hengel A. (2019) Reinforcement Learning with Attention that Works: A Self-Supervised Approach. In: Gedeon T., Wong K., Lee M. (eds) Neural Information Processing. ICONIP 2019. Communications in Computer and Information Science, vol 1143. Springer, Cham.

Statement of Authorship

Title of Paper	Reinforcement Learning with Attention that Works: A Self-Supervised Approach
Publication Status	<input checked="" type="checkbox"/> Published <input type="checkbox"/> Accepted for Publication <input type="checkbox"/> Submitted for Publication <input type="checkbox"/> Unpublished and Unsubmitted work written in manuscript style
Publication Details	Manchin, A., Abbasnejad, E., van den Hengel, A. (2019). Reinforcement Learning with Attention that Works: A Self-Supervised Approach. In: Gedeon, T., Wong, K., Lee, M. (eds) Neural Information Processing. ICONIP 2019. Communications in Computer and Information Science, vol 1143. Springer, Cham. https://doi.org/10.1007/978-3-030-36802-9_25


Principal Author


Name of Principal Author (Candidate)	Anthony Manchin		
Contribution to the Paper	<ul style="list-style-type: none">- Development of the main idea of the paper- Implementing and conducting the experiments- Writing and revising of the paper		
Overall percentage (%)	80%		
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.		
Signature		Date	20 July, 2022

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

- the candidate's stated contribution to the publication is accurate (as detailed above);
- permission is granted for the candidate to include the publication in the thesis; and
- the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Ehsan Abbasnejad		
Contribution to the Paper	<ul style="list-style-type: none">- Help with the development of the idea- Help writing, revision, and discussions		
Signature		Date	July 22, 2022

Name of Co-Author	Anton van den Hengel		
Contribution to the Paper	<ul style="list-style-type: none">- Help writing, revision, and discussions		
Signature		Date	July 22, 2022

Please cut and paste additional co-author panels here as required.

Abstract

Attention models have had a significant positive impact on deep learning across a range of tasks. However, previous attempts at integrating attention with reinforcement learning have failed to produce significant improvements. Unlike the selective attention models used in previous attempts, which constrain the attention via preconceived notions of importance, our implementation utilises the Markovian properties inherent in the state input. We propose the first combination of self-attention and reinforcement learning that is capable of producing significant improvements, including new state of the art results in the Arcade Learning Environment.

3.1 Introduction

Research on reinforcement learning has seen accelerating advances in the past decade. In particular, methods for deep RL have made tremendous progress since the seminal work of [Mnih et al., 2015] on Deep Q-Networks. A number of different approaches have progressed the state of the art on simulated tasks - in particular in the Arcade Learning Environment [Bellemare et al., 2013] - to, or above that of human players. The major focus of attention of many of these methods is the learning and optimization of the policies. However, one thing that all these approaches have in common is that they process the raw input data through a convolutional neural network.

Regardless of the chosen policy optimisation method, the underlying neural network is responsible for interpreting and encoding useful representations of the input state. While significant efforts have focused on methods for policy optimisation, the techniques for encoding the observations have received less attention. While many methods thus use generic, off-the-shelf CNN architectures, we instead focus on this as the main subject of study.

Taking inspiration from other areas of deep learning, we investigate the benefits of incorporating self-attention into the underlying network architecture. Attention models have been applied with remarkable success to complex visual tasks such as video and scene understanding [Han, 2018, Shi et al., 2018, Fang et al., 2018], natural language understanding (including machine translation) [Bahdanau et al., 2014, Zhao and Zhang, 2018], and generative models using generative adversarial networks (GANs) [Xu et al., 2017, Kastaniotis et al., 2018]. Although previous attempts to integrate attention with RL have been made, these attempts have largely used hand-crafted features as inputs to the attention model [Yuezhang

et al., 2018, Zhang et al., 2018]. Although the spatial and temporal information contained within the state input (when using stacked frames) is sufficient to satisfy the Markov principle, the hand crafted features often chosen by these approaches do not satisfy the Markov principle.

For this reason we observe the work from [Wang et al., 2017], and their goal of improved attention through space, time, and space-time by combining self-attention with non-local filtering methods. The benefit of self-attention is the ability to compute representations of an input sequence by relating different positions of the input sequence. Their implementation achieves state of the art results on the Kinetics [Kay et al., 2017] and Charades [Sigurdsson et al., 2016] datasets. However, the datasets they considered are large-scale video classification problems where the changes in the input from time to time are minimal. In addition, the neural network is in a passive environment that does not require interaction. None the less, these challenges require spatial and temporal reasoning abilities that would be very useful for a reinforcement learning agent to possess. Taking inspiration from this work, we capitalise on the Markovian principle of the state input and propose a novel implementation of self-attention within the classical convolutional neural network architecture, as used by [Mnih et al., 2015]. The contributions of our paper are as follows.

- We provide a spatio-temporal self-attention mechanism for reinforcement learning and demonstrate that the network architecture has significant benefits in learning a good policy
- We present state-of-the-art results in the Arcade Learning Environment [Bellemare et al., 2013]. Our approach significantly outperforms the baseline across a number of environments where the agent has to attend to multiple opponents and anticipate their movements in time.

3.2 Related Work

3.2.1 Reinforcement Learning Network Design in Video Games

Current state-of-the-art approaches for reinforcement learning agents evaluated in the Arcade Learning Environment (ALE) are built on top of the original network architecture proposed by [Mnih et al., 2015]. Alterations to this underlying network architecture have included the implementation of recurrent neural networks (RNN). [Hausknecht and Stone, 2015] proposed replacing the fully connected layer, following the output of the last convolutional layer of the network with an LSTM. This allowed for a single frame input to be used, as opposed to sequentially stacked frames, with the LSTM integrating temporal information. [Oh et al., 2016] also proposed using recurrent networks with their Recurrent Memory Q-Network (FRMQN). This memory-based approach used a mechanism based on soft attention to help read from memory and was evaluated with respect to solving mazes in Minecraft (a flexible 3D world). In comparison to [Hausknecht and Stone, 2015] and [Oh et al., 2016] we utilise sequentially stacked frames as input and augment the network with an attention model which is able to demonstrate improved temporal reasoning.

Having also noticed the heavy reliance on convolutional neural networks for reinforcement learning agents within ALE, [Wang et al., 2015] proposed a new neural network architecture specifically for model free learning. The architecture (known as Dueling DQN), still utilised the convolutional neural network as originally proposed by [Mnih et al., 2013], however, instead of a single sequence of fully connected layers after the convolutional layers, Dueling DQN used two sequences. This allowed for the advantage and value functions estimates to be separated

before being combined to produce a single output Q-function. The novelty of this architecture proved to be useful in situations where the value of a given action is either of little or great importance. An example of this is the environment Enduro, where the choice of action only matters when the agent needs to avoid a collision. The separation of the advantage and value functions allows the value stream to learn to pay attention to the road, while the advantage stream learns to only pay attention when there are cars immediately in front of the agent.

3.2.2 Attention in Reinforcement Learning

[Sorokin et al., 2015] studied the effects of adding both ‘soft’ and ‘hard’ attention mechanisms to a CNN-LSTM network trained using DQN. In this approach the authors inserted the attention mechanism in between the CNN and the LSTM. The attention mechanism consists of two fully connected layers followed by a softmax activation and takes as input the flattened output of the final convolutional layer and the previous hidden state of the LSTM in order to compute a context vector. When trained in a soft manner, the attention mechanism outputs the context vector as a weighted sum of its input vectors which correspond to the features extracted by the CNN for different image regions. When trained in a hard manner, the attention mechanism only samples a single feature location at each time step. The selection of this location requires the inclusion of an additional attention policy π_g which is trained using the algorithm REINFORCE [Sutton et al., 2000]. As the attention policy decides which features should be focused upon, training both the attention policy and the underlying CNN simultaneously would likely lead to either divergence or extremely long training times. To counter this [Sorokin et al., 2015] preinitialised the underlying CNN with weights obtained from training an agent using the soft attention approach. Testing in five different ALE games

revealed that the soft attention implementation outperformed both the baseline and the hard implementation in two and five games respectively. Although this approach indicated some potential performance improvements under certain conditions, experiments were limited with results showing no systematic performance increases. In contrast, our work explores a different form of self-attention, and we demonstrate significant benefits in both performance and interpretability for the resulting policy.

[Choi et al., 2017] also proposed combining attention with reinforcement learning for navigation purposes. This approach employed a Multi-focus Attention Network (MANet) which used multiple parallel attention modules. This worked by first segmenting the input into K partial states which were then passed through a CNN. The outputs for each partial state was then passed through parallel attention layers which is trained using a soft approach. To prevent multiple attention layers attending to the same partial states, the authors explore two regularisation methods, entropy and distance. The entropy and distance terms were defined as $R_e = \lambda_e \times \sum_n ||A^n \times \log A^n||$ and $R_d = \lambda \times \exp(-\sum_n m(A^n - A^m)^2)$ respectively and were added to the loss function. The authors designed two custom game environments to evaluate their proposed methodology. The first containing a single agent task and the second consisting of multiple agent cooperation. The reasoning for the second task is that the input segmentation into partial states could also be used to allow a network to learn how to pass important information between multiple agents in a cooperative setting. In both settings MANet was able to outperform the baselines with improved sample efficiency.

[Zhang et al., 2018] proposed Attention Guided Imitation Learning (AGIL) framework with the main idea being to train an agent to attend to the input in the same fashion a human would. To achieve this, a gaze network is first trained

in a supervised manner from human gaze data. The network consists of three parallel channels which are made of convolutional and deconvolutional layers. Each channel takes a different input with the first being four sequentially stacked frames (images), the second takes the optical flow information, and the last channel is fed a saliency map computed using the Itti-Koch model [Itti et al., 1998]. The outputs from these channels are averaged together before a softmax is applied to predict the gaze locations. After this network is trained, it is used to augment the input that is fed to the agents policy. The agents policy architecture includes two parallel convolutional channels, the outputs of which are then averaged together and passed through a fully connected layer to predict an action. The first convolutional channel takes as input the processed frames from the environment, while the second convolutional channel takes as input the input frames augmented with the output from the gaze network. The results showed that this inclusion of a human gaze attention network was able to outperform standard imitation learning in eight different games from ALE.

More recently [Yuezhang et al., 2018] proposed a model based upon the Broadbent filter model [Broadbent, 1958]. This approach uses the optical flow calculated between two frames to construct an attention map, which was then combined with the output of the last convolutional layer in their network. The authors evaluated their network on a toy problem named 'Catch', originally inspired by [Mnih et al., 2014], a modified version of the problem, and four games from ALE. The modified version of Catch involved adding artifacts to the background that closely resembled the falling square in the game. This made relying on the spatial information provided by the single frame input difficult. In this setting the additional optical flow information proved to be useful to the agent, resulting in improved performance over the baseline model. Testing in ALE was unable to

produce similar improvements which is likely a result of the stacked frames used as input allow the network to learn important optical flow information directly.

3.3 Methods

3.3.1 Markov Decision Process Formulation

A Markov Decision Process formally describes a fully observable environment for reinforcement learning, and is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ where,

- \mathcal{S} is a set of states
- \mathcal{A} is a set of action
- \mathcal{P} is a state transition matrix, where $\mathcal{P}_{s,s'}^a = P(S_{t+1} = s' | S_t = s, A_t = a)$
- \mathcal{R} is a reward function, where $\mathcal{R}(s) = \mathbb{E}(R_{t+1} | S_t = s, A_t = a)$

Critically, this formulation is reliant upon the validity of the Markov Property which states that the probability of the next state is dependent only upon the current state, and independent from previous states.

The ultimate goal, given a MDP, is to derive a policy π which will maximise the cumulative discounted reward from the environment.

$$\mathbb{E}[\sum \gamma^t \mathcal{R}(s_t, s_{t+1})] \quad (3.1)$$

Where γ is a hyper-parameter between 0 and 1 which discounts future rewards. This encourages behaviour that is more likely to exploit immediate reward as future rewards are often less certain.

3.3.2 Policy Optimisation

We use an Actor-Critic network architecture, optimised by Proximal Policy Optimisation (PPO) [Schulman et al., 2017] to train our agents over traditional DQN baselines due to its wall clock training time and improved general performance. In particular we utilise the open-source implementation OpenAI Baselines [Dhariwal et al., 2017].

The objective function used is defined by,

$$L_t^{CLIP+VF+S}(\theta) = \mathbb{E}[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)] \quad (3.2)$$

where,

- c_1 is the value function coefficient
- c_2 is the entropy coefficient
- S is the entropy bonus
- $L_t^{CLIP}(\theta)$ is the policy loss
- $L_t^{VF}(\theta)$ is the value function loss

The value function loss $L_t^{VF}(\theta)$ is defined as the squared-error loss $(V_\theta(s_t) - V_t^{targ})^2$, while the policy loss $L_t^{CLIP}(\theta)$ is given by,

$$L_t^{CLIP}(\theta) = \mathbb{E}[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t)] \quad (3.3)$$

Where $r_t(\theta)$ is the probability ratio given by,

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad (3.4)$$

and \hat{A}_t is a truncated version of the generalised advantage estimation, which when $\lambda = 1$:

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \dots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \quad (3.5)$$

where,

$$\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t) \quad (3.6)$$

A full derivation can be found in [Schulman et al., 2017]

3.3.3 Non-Local Neural Networks

The variant of self-attention used in our experiments was first proposed by [Wang et al., 2017] as a flexible building block for use with convolutional neural networks. In particular, the generic non-local operation can be defined as,

$$y_i = \frac{1}{C(x)} \sum_j F(x_i, x_j) G(x_j) \quad (3.7)$$

This formulation allows for a response to be computed for an output position, indexed by i , across all possible positions, indexed by j . The output position itself can be in space, time, or spacetime while x is the input signal (e.g. image, video, sequence) and y is the output signal. F is a pairwise function, which calculates a relationship between i and all j , while a representation of the input signal at position j is computed by the function G . This response can be normalised by the factor $C(x)$ which in practise is equal to the number of positions in x .

Unlike a fully connected layer which uses learnt weights to compute a response, a non-local operation uses the relationship between different locations to compute

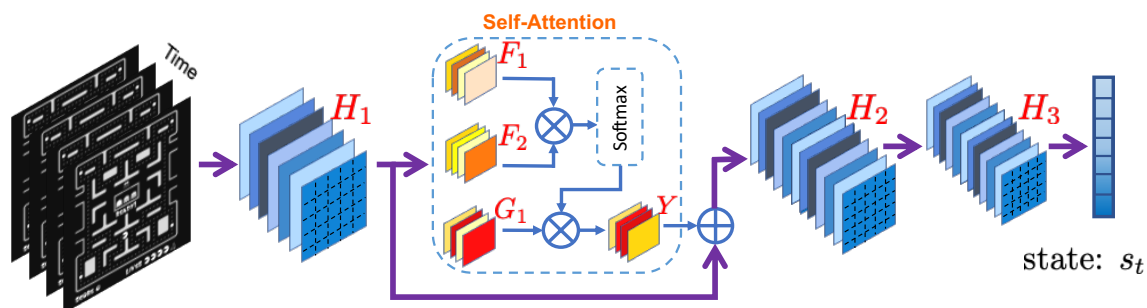


Figure 3.1: Overview of the proposed architecture. We introduce a self-attention module within the CNN used to process the input observations. The resulting policy benefits significantly from the capability of selective attention over space and time.

a response. Additionally, a non-local block can support variable sized inputs, whilst maintaining the input size in the output. A diagram of the non-local block implemented as self-attention can be seen in fig.3.1.

3.3.4 Network Architecture

We build upon the common practice of using a convolutional neural network to encode input observations into a state representation. Our main contribution is to incorporate the above described self-attention mechanism over space and time to better equip an agent with spatial and temporal reasoning. We specifically describe the implementation of self-attention used in our approach (see Fig.3.1), as originally proposed by [Wang et al., 2017]. The self-attention mechanism operates as follows. F_1 , F_2 , and G_1 are all 1×1 convolutions. The outputs of F_1 and F_2 are matrix multiplied together before passing through a *Softmax* activation, which is then matrix multiplied by the output from G_1 . This is then passed through Y which is also a 1×1 convolution, before being added back into the original input.

We specifically describe six instantiations of our general approach. Owing to the empirical nature of current research in deep learning, we conducted a thorough

exploration of possible implementations of self-attention. Different domains have previously shown to be better addressed with different, sometimes conflicting implementation choices [Yuezhang et al., 2018, Sorokin et al., 2015, Choi et al., 2017]. It is important to consider the spatial and temporal dimensions of the data, and maintain the possibility of attending to different parts of the input across the layers of the network. Our six proposed instantiations are described as follows (see Fig.3.1).

- Self-Attending Network (SAN): Self-attention between convolutional layers 'H1' and 'H2'. This approach focuses on how attention interacts with the input in the lowest level of the network.
- Strong Self-Attending Network (SSAN): Multiplying the output of the last convolutional layer in the self-attention component ('Y') by a factor of two (thereby increasing the influence of attention on the network).
- Self-Attending Double Network (SADN): Self-attention between convolutional layers 'H1' and 'H2', 'H2' and 'H3'. Since the higher-level layers learn the semantics and higher-level abstractions, we intend to evaluate how attention changes the performance when applied to these layers.
- Strong Self-Attending Double Network (SSADN): Multiplying the outputs for both self-attention components by a factor of two.
- Pure Self-Attending Network (PSAN): Passing only the output of the self-attention forward, removing the addition of the previous convolutional layer output. This approach investigates the performance of the agent when only the 'pure' sequence representations learnt by the self-attention component are passed forward in the network.

- Pure Self-Attending Double Network (PSADN): Self-attention between convolutional layers 'H1' and 'H2', 'H2' and 'H3', while passing only the output of the self-attention forward.

3.4 Experiments

3.4.1 Validation Methodology

Implementation

The Arcade Learning Environment is a well-established baseline which allows us to critically evaluate the effects of our proposed architecture modifications. We use Proximal Policy Optimisation [Schulman et al., 2017] to train our agents over traditional DQN baselines due to its wall clock training time and improved general performance. In the interest of comparability, the open-source implementation from OpenAI 'Baselines' was utilised [Dhariwal et al., 2017]. In order to objectively identify the effects of the additional attention model, the standards set by [Mnih et al., 2015] were followed. This included pre-processing of the input image from a single 210x160 RGB image to a stack of four 84x84 grey-scale images. 'No-Op' starts were also used which prevents the agent from taking an action at the start of each game for a random number (maximum thirty) of time-steps.

Performance evaluation

In order to evaluate our agents, we randomly seed each different architecture for a total of three times across ten different Atari games. In terms of standard training times for bench marking, [Mnih et al., 2015] [Schulman et al., 2017] [Hausknecht and Stone, 2015] [Horgan et al., 2018] show variations between 40M to 16B+

frames. We train each model for a total of 40M time-steps, which is equivalent to 160M frames. This is in line with the evaluation methodology as presented by [Fortunato et al., 2017]. Performance is evaluated by the maximal score achieved (after averaging) during training.

3.4.2 Performance results

By integrating attention into the underlying neural network, new state of the art results for Demon Attack were achieved. In fact, all but one implementation was able to significantly improve against the baseline, along with surpassing the previous state of the art results for Demon Attack. Additionally, SAN was able to produce significant improvements in both Asterix and MsPacman. Impressively SAN is able to surpass the previously highest score reported using a policy gradient method for Demon Attack, MsPacman, Bowling, Freeway and Frostbite [Fortunato et al., 2017, Wu et al., 2017].

Table 3.1 shows the maximal score after averaging over three random seeds during training for 40M time-steps. From this we can observe that integrating self-attention, in one form or another, led to an increase in performance across 60% of environments tested. Table 3.2 also shows our proposed network, SAN, improved results in 50% of environments when directly compared to the baseline PPO agent. While Fig. 3.2 shows the training curves for each network across all ten environments. This allows us to visually see the increased sample efficiency self-attention provides in environments such as Demon Attack, MsPacman, Asterix, and Frostbite.

Although single implementations of attention such as SAN and SSAN were able to achieve higher rewards across more environments than other 'double' implementations, it is clear in Fig. 3.2 and Table 3.1 that SADN is able to outperform

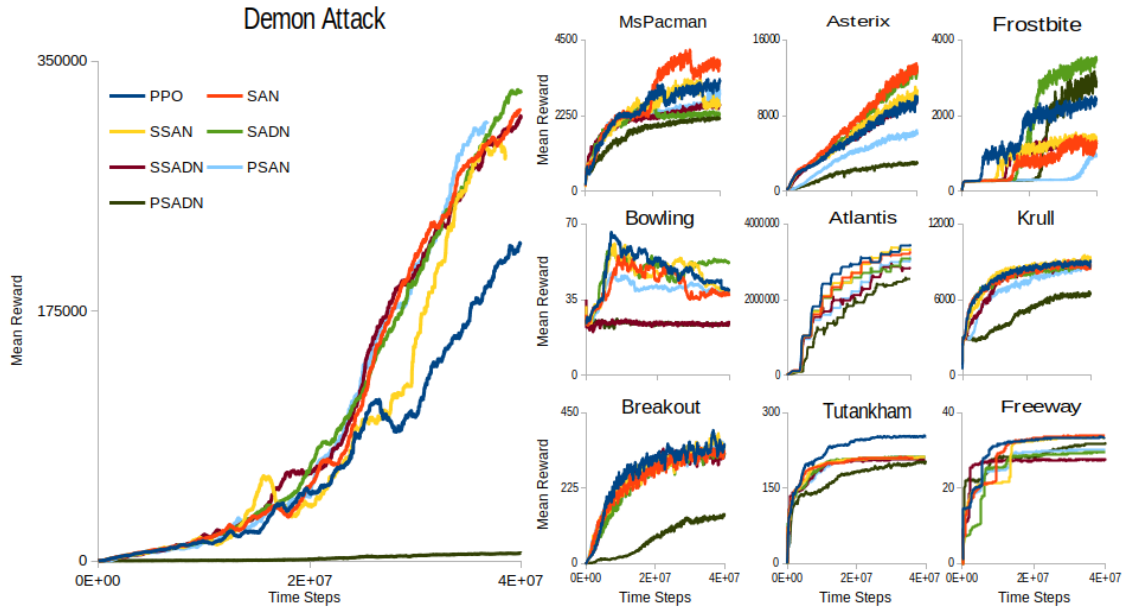


Figure 3.2: Learning curves of all variants compared to the baseline on all ten environments tested. Agents are trained for a total of 40M timesteps, with results averaged over three random seeds. Here we can see the clear advantage of self-attention is able to provide with respect to sample efficiency.

SAN in Demon Attack, Frostbite, and Bowling. This provides support for the idea that attention in general is beneficial to the network.

	PPO	SAN	SSAN	SADN	SSADN	PSAN	PSADN
MsPacman	3342.15	4217.3	3351.4	2638.3	2681.5	3118.6	2252.36
Tutankham	254.57	211.04	211.14	212.07	208.15	211.60	205
Freeway	33.48	33.93	33.58	29.66	27.73	30.21	31.87
Atlantis	3.45M	3.27M	3.39M	3.09M	2.96m	3.12M	2.58M
Krull	9102	9136	9535	8912	9191	9171	6668
Demon Attack	222650	315727	294359	329837	311438	307539	5510
Bowling	66.33	55.21	60.99	57.52	35	47.49	26.26
Frostbite	2503	1481	1552	3554	1426	1036	3173
Asterix	10121	13556	11069	13253	9815	6461	3106
Breakout	398.10	353.47	388.89	345.07	354.81	376.45	147.81

Table 3.1: Maximal score achieved by each implementation, averaged over three random seeds and trained for 40M time-steps. These results clearly demonstrate the improved performance of multiple self-attention variants.

Environment	PPO	SAN
MsPacman	3342.15	4217.3
Tutankham	254.57	211.04
Freeway	33.48	33.93
Atlantis	3445922	3272823
Krull	9102	9136
Demon Attack	222650	315727
Bowling	66.33	55.21
Frostbite	2503	1481
Asterix	10121	13556
Breakout	398.10	353.47

Table 3.2: Direct comparison between baseline PPO and SAN. Clearly demonstrating that our proposed method is capable of improving performance in 50% of tested environments.

3.5 Conclusions and Future Work

We evaluate the benefit of incorporating self-attention into the underlying neural network architecture with direct access to the spatial and temporal information from the state input. We directly compare this approach to the classic architecture first proposed by [Mnih et al., 2015]. Our results clearly indicate that the addition of attention to the network is beneficial, and lead to significant improvements in sample efficiency across 60% of tested environments. Of particular note is the performance in the environment Demon Attack, where the addition of self-attention resulted in state-of-the-art results, far exceeding the previous reported benchmark.

Future work will seek to further investigate why attention was more beneficial in some environments compared to others, along with further testing of the proposed architecture with different optimisation techniques, including DQN methods.

Bibliography

- [Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.
- [Bellemare et al., 2013] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.
- [Broadbent, 1958] Broadbent, D. E. (1958). Perception and communication.
- [Choi et al., 2017] Choi, J., Lee, B., and Zhang, B. (2017). Multi-focus attention network for efficient deep reinforcement learning. *CoRR*, abs/1712.04603.
- [Dhariwal et al., 2017] Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. (2017). Openai baselines. <https://github.com/openai/baselines>.
- [Fang et al., 2018] Fang, S., Xie, H., Zha, Z.-J., Sun, N., Tan, J., and Zhang, Y. (2018). Attention and language ensemble for scene text recognition with convolutional sequence modeling. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, pages 248–256, New York, NY, USA. ACM.
- [Fortunato et al., 2017] Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. (2017). Noisy networks for exploration. *CoRR*, abs/1706.10295.

- [Han, 2018] Han, Y. (2018). Explore multi-step reasoning in video question answering. In *Proceedings of the 1st Workshop and Challenge on Comprehensive Video Understanding in the Wild, CoVieW'18*, pages 5–5, New York, NY, USA. ACM.
- [Hausknecht and Stone, 2015] Hausknecht, M. J. and Stone, P. (2015). Deep recurrent q-learning for partially observable mdps. *CoRR*, abs/1507.06527.
- [Horgan et al., 2018] Horgan, D., Quan, J., Budden, D., Barth-Maron, G., Hessel, M., van Hasselt, H., and Silver, D. (2018). Distributed prioritized experience replay. *CoRR*, abs/1803.00933.
- [Itti et al., 1998] Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259.
- [Kastaniotis et al., 2018] Kastaniotis, D., Ntinou, I., Tsourounis, D., Economou, G., and Fotopoulos, S. (2018). Attention-aware generative adversarial networks (ata-gans). *CoRR*, abs/1802.09070.
- [Kay et al., 2017] Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., and Zisserman, A. (2017). The kinetics human action video dataset. *CoRR*, abs/1705.06950.
- [Mnih et al., 2014] Mnih, V., Heess, N., Graves, A., and Kavukcuoglu, K. (2014). Recurrent models of visual attention. *CoRR*, abs/1406.6247.
- [Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. cite arxiv:1312.5602Comment: NIPS Deep Learning Workshop 2013.

- [Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.
- [Oh et al., 2016] Oh, J., Chockalingam, V., Singh, S. P., and Lee, H. (2016). Control of memory, active perception, and action in minecraft. *CoRR*, abs/1605.09128.
- [Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.
- [Shi et al., 2018] Shi, J., Zhang, H., and Li, J. (2018). Explainable and explicit visual reasoning over scene graphs. *CoRR*, abs/1812.01855.
- [Sigurdsson et al., 2016] Sigurdsson, G. A., Varol, G., Wang, X., Farhadi, A., Laptev, I., and Gupta, A. (2016). Hollywood in homes: Crowdsourcing data collection for activity understanding. *CoRR*, abs/1604.01753.
- [Sorokin et al., 2015] Sorokin, I., Seleznev, A., Pavlov, M., Fedorov, A., and Ignatova, A. (2015). Deep attention recurrent q-network. *CoRR*, abs/1512.01693.
- [Sutton et al., 2000] Sutton, R. S., McAllester, D. A., Singh, S. P., and Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063.
- [Wang et al., 2017] Wang, X., Girshick, R. B., Gupta, A., and He, K. (2017). Non-local neural networks. *CoRR*, abs/1711.07971.

- [Wang et al., 2015] Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., and de Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. cite arxiv:1511.06581Comment: 15 pages, 5 figures, and 5 tables.
- [Wu et al., 2017] Wu, Y., Mansimov, E., Grosse, R. B., Liao, S., and Ba, J. (2017). Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5279–5288. Curran Associates, Inc.
- [Xu et al., 2017] Xu, T., Zhang, P., Huang, Q., Zhang, H., Gan, Z., Huang, X., and He, X. (2017). AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks. *CoRR*, abs/1711.10485.
- [Yuezhang et al., 2018] Yuezhang, L., Zhang, R., and Ballard, D. H. (2018). An initial attempt of combining visual selective attention with deep reinforcement learning. *CoRR*, abs/1811.04407.
- [Zhang et al., 2018] Zhang, R., Liu, Z., Zhang, L., Whritner, J. A., Muller, K. S., Hayhoe, M. M., and Ballard, D. H. (2018). AGIL: learning attention from human for visuomotor tasks. *CoRR*, abs/1806.03960.
- [Zhao and Zhang, 2018] Zhao, S. and Zhang, Z. (2018). Attention-via-attention neural machine translation.

CHAPTER 4

Unsupervised rule discovery with autonomous agents in visually complex environments

The work contained in the chapter is based upon work that was partially sponsored by an industry partner, wherein publication has not been pursued as a result of the sponsorship.

Abstract

While modern deep reinforcement learning has produced outstanding improvements over previous handcrafted agents in many fields, this has largely come at the cost of explainability. This has created areas of concern around the deployment of these algorithms, especially in fields which require an explanation in conjunction with a decision. In this chapter we propose a novel approach to extending the explainability of a deep reinforcement learning agent via state-action trajectory clustering for rule extraction. Our approach is post hoc model agnostic (pedagogic), and does not require the pre-existence of semantic labels. We evaluate our formulation in visually complex environments from the Atari gaming suite whilst simultaneously evaluating the impact of incorporating attention mechanisms with the underlying neural policy. Through these experiments we can show that not only are we able to detect the existence of rules without semantic labels with our approach, but that the incorporation of attention mechanisms with the convolutional layers of the network lead to improve rule discovery. Finally, we extend our analysis by performing a visual inspection of the underlying policies ‘focal attention’ by extending the Grad-CAM technique to work in a reinforcement learning setting.

4.1 Introduction

Explainability in artificial intelligence, or XAI as it is also known, is a growing area of interest to many researchers. This is largely due to the ever-increasing proliferation of deep learning techniques powering modern AI models. Unlike traditional AI methods that were largely interpretable, deep learning models can be much more difficult to interpret. Additionally, it is a lot easier to explain a small or shallow model, than it is to explain a deep model. An example of this can be demonstrated by considering a PID controller. We can model a PID controller as a single layer neural network that has three neurons and is hand tuned (without back-propagation). The controller is tuned such that the first neuron interacts with the input signal (generally this is an error reading at time t) in a 'Proportional' manner, the second in a manner which relates the signal and the time together, 'Integral', and the third relates the change in signal over time, the 'Derivative'. A model such as this is both simple and easily explainable from both a mathematical level, and a semantic level. We can describe how the model works with a high-level explanation confidently, whilst being mathematically assured that the outputs of our hidden layer of neurons can only impact one single output neuron in a manner that is both deterministic and completely defined. One may consider this (albeit slightly stretched) hypothetical neural network PID model as a highly interpretable and explainable AI model.

The question now becomes, what happens when we use back-propagation to tune the model, and instead of one hidden layer we have one hundred? Additionally, our hidden neurons are now made up of weights and biases ($w + b$), instead of coefficients (k_p, k_i, k_d). Our inputs too are more complicated, instead of a single numerical value we have a multi-dimensional array of values. The problem very

quickly becomes infinitely large, as an adjustment to even a single input feature may create large changes in a model's output due to the flow on effects of different activation's in the network as a result of the changed input. This point in particular has driven an entire field of research known as 'Adversarial Machine Learning', where researchers have investigated the resistance of popular deep learning models to single feature attacks. [Huang et al., 2011, Kurakin et al., 2016, Su et al., 2019]. From this we can also clearly see that even if we derive mathematical bounds for each hidden neuron in a relatively constrained multi-layer neural network and define which combinations of input features trigger positive activation's, the vast degree of interconnectedness of a deep neural network is just too complicated to entirely grasp for a human at this level.

While XAI researchers have been devising methods to explain deep learning models with varying approaches that tend to range from intrinsic model specific, to post-hoc model agnostic methods [Vilone and Longo, 2020], the explainability of reinforcement learning agents (with neural network policies) has received little to no attention. Explainability of reinforcement learning, or XRL, contains all the problems associated with XAI and then some. Unlike a standard object detector or classifier which has been trained with ground truth labels and has no ability to interact with its input, a RL agent must learn about its environment without the guidance of an oracle signal, and its decisions often directly impact its future inputs. Furthermore, the RL policy is optimised in a manner that is conditioned upon the validity of the Markov assumption [Markov, 1957]. When an object classifier identifies a dog in an image, one may conclude that this decision is due to the combination of detected latent features. However, when a RL agent takes a particular action, it may not be clear as to why it has decided to choose this action. While it is true that the the output has been triggered by a combination of latent

space features and activation's, the reason of why is not so easy to answer. Is it because the agent believes it will receive an immediate reward from this action? Or is it because the agents value function believes that by doing so it will increase the probability of receiving some future discounted reward? Additionally, what can we tell about the agents understanding of the environment? For example, when humans learn new skills or tasks, we build rule sets to govern how we should apply these skills or accomplish these tasks. Does an RL agent also build a rule set? And if so, what does this look like, and how can we tell?

To begin the process of understanding or explaining a RL agent, we believe it is vital to be able to answer these types of questions. In this paper we focus on investigating and answering the question of whether a RL agent is able to learn rules which it can then apply in order to gain an advantage or receive rewards from an environment with. Additionally, we investigate the impact recent breakthroughs in RL architecture design impact rule discovery and provide a visual analysis of how this impacts feature attention maps.

The contributions of this chapter are as follows.

- We provide a formal definition for the classification of learnt rules by autonomous agents trained using deep reinforcement learning in visually complex environments. Additionally, we demonstrate the existence and application of such rules in trained policies.
- We provide an analysis of current state-of-the-art policy design choices that implement attention mechanisms for reinforcement learning with respect to rule discovery. We show that the addition of attention mechanisms with a neural network policy results in increased rule discovery.
- We provide visualisation of a policies feature attention map that sheds new

light on the policy learnt by an agent when equipped with non-local neural networks (attention mechanisms). This provides evidence of temporal attention and improved spatial awareness in networks that incorporate a self-attention mechanism over policies that do not.

4.2 Related Work

The two closest fields of work to our own would be policy summarisation, and rule extraction. We provide a summary and comparison of related works from these fields below.

4.2.1 Policy Summarisation

Policy summarisation methods seek to develop frameworks that can explain a deep reinforcement learning (DRL) agent via a high-level explanation (typically natural language). These frameworks are similar to the pedagogical rule extraction-based approaches in that they generally seek to link the inputs to the model with the outputs, while allowing for the internal neural network to be treated as a black-box. However, unlike pedagogical rule extraction, they do not necessarily seek to extract provable logical conditions such as "if else" scenarios.

[Fukuchi et al., 2017] proposed the Instruction-based Behaviour Explanation model (IBE). This model consists of two main parts. The first part learns a mapping from $(s_t, a_t) \rightarrow s_{t+1}$, while the second part takes a signal provided by a human expert and learns a mapping from $\Delta s_t \rightarrow m$, where m is an explanation as provided by a human expert. Although this approach is technically post-hoc model agnostic, due to the requirement that a human expert provide an explanation and subsequent mapping for state transitions, the scalability of this approach is

limited. Additionally, the highly constrained environment that this methodology was evaluated upon (Lunar Lander [Brockman et al., 2016]) resulted in the human expert providing a total of only three explanations. These consisted of "Fall to the left", "Fall straight down", and "Fall to the right".

In a similar vein, [Hayes and Shah, 2017] proposed a framework that could be used for both DRL and hard coded robots. The framework was designed to increase the level of transparency regarding a robot's behaviour in collaborative settings. A user could issue a natural language query to receive a natural language response that would explain the behaviour of the policy in question. Their approach was designed to facilitate the operational goals outlined in [Vessey, 1985]. Their proposed solution consisted of a composition of functions " $f = \text{Compose_summary} \circ \text{Summarize_attributes} \circ \text{Resolve_states} \circ \text{Identify_question}$ " [Hayes and Shah, 2017]. By using communicable predicates to associate states and actions with natural language descriptors, they were then able to train a model via a Markov decision process to model the domain and policy of the agent.

[Verma et al., 2018] proposed a framework called "Programmatically Interpretable Reinforcement Learning" (PIRL). This framework involves first training a Deep Reinforcement Learning (DRL) agent to act as a guide for their "Neurally Directed Program Synthesis" algorithm as it searches for a programmatic policy that can approximate the behaviour of the DRL agent. While the outcomes of this approach were able to produce a policy, which was intrinsically more interpretable, they were not able to match the level of performance of the guiding DRL agent. Additionally, this approach requires a high level of knowledge about the environment and action space of the agent to craft a domain specific language capable of bounding all possible interactions between the agent and the environment.

4.2.2 Rule Extraction

Pedagogical rule extraction is similar to that of policy summarisation in that it aims to develop a mapping from input to output while treating the network in between as a black box. However, pedagogical rule extraction methods attempt to define "if else" style relationships between the inputs and outputs. Works published in this area have largely focused on relatively simple (generally single layer) neural networks. An example of this is input-to-output through 'if-else' mapping is Validity Interval Analysis (VI-Analysis) [Thrun, 1995], which seeks to determine the interval ranges within input data which result in the same output. These intervals form the bounding conditions for the discovered rules. The idea of splitting rules according to discovered intervals in training data was also used by [Craven and Shavlik, 1996] and [Augasta and Kathirvalavakumar, 2012]. However, practical applications of these approaches are severely limited to networks that take as input simple structured data, and in their current forms could not be applied to areas involving computer vision.

A decompositional approach to extracting rules from neural networks generally works by evaluating the activation points at the neuron level of the network. Until recently a lot of work in this field focused on simple, single layer neural networks [Sato and Tsukimoto, 2001, Quinlan, 1993]. The first to extend this idea to a deep neural network was [Zilke et al., 2016], who built upon the ideas proposed by [Sato and Tsukimoto, 2001]. Their model, which they named DeepRED, used the C4.5 algorithm to extract decision trees for each class output of a network, before processing every hidden layer in a descending order. Once rules are extracted for each layer of the network, they are then merged together to form a rule set that describes the relationship between input and output as it passes through each

hidden layer.

The first known work combining these two styles of approach was the Decision Detection algorithm which was proposed by [Tickle et al., 1994]. It would first identify dependencies between the inputs and outputs according to the neuron activations within the network, before learning a symbolic representation in a pedagogical manner. A similar model was also proposed by [Kamruzzaman and Islam, 2010], which also identified dependencies between inputs and outputs before generating pedagogical style rules. In particular, the authors proposed steep sigmoid activation functions that in practice approximated step functions whilst maintaining differentiable properties. This allowed for a simplified discretisation process upon which 'if-else' rules could be extracted.

While these approaches have all made significant contributions to the challenge of extracting rules from a neural network, they all become impractical when applied to the deep neural networks commonly in use today. Additionally, the requirement for a simple and generally structured input is a very prohibitive factor to applying these within computer vision scenario.

4.3 Methods

In this section we outline the visualisation techniques along with our proposed definition for rule extraction. We train the reinforcement learning agents using the same methodology and optimisation techniques as outlined in 3.3.

4.3.1 Visual Evaluation

As we are interested in developing more explainable deep reinforcement learning agents, the ability to visualise what aspects of the inputs the agent is using to

make a decision is also important. We first considered the visualisation technique proposed by [Greydanus et al., 2017] which involves blurring a part of the input and evaluating the effect on policy performance in order to determine attention at that location. However, as this would be computationally exhaustive to implement with a sequential series of input frames stacked together, we implement a Grad-cam [Selvaraju et al., 2016] inspired approach similar to [Weitkamp et al., 2019]. We produce action-discriminative activation maps using the gradients back-propagated with respect to the chosen action. Global average pooling is performed over the gradients to determine the neuron importance weights, α_k^a of action a , for the last activation layer k in our network.

$$\alpha_k^a = \overbrace{\frac{1}{Z} \sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial h^a}{\partial L_{ij}^k}}_{\text{gradients}} \quad (4.1)$$

Where h^a is the score for action a prior to the softmax. The gradients are then element-wise multiplied by the forward pass activation's of the final activation layer L^k before passing through a ReLU activation, revealing L^a , the weighted action activation map.

$$L^a = \text{ReLU}\left(\sum_{k=1}^K \alpha_k^a L^k\right) \quad (4.2)$$

This activation map is then bilinearly extrapolated to the size of the input frame and overlaid producing accurate indications of visual attention with respect to decision making.

4.3.2 Rule Definition and Extraction

While many previous works have set about defining a rule as a natural language conditional argument in static classification settings, we believe that this merely provides a semantic label for a rule in a reinforcement learning setting. Due to the temporal dimension involved with reinforcement learning, an agent must learn that its actions at time t in state s_t directly effect the probabilities of s_{t+1} . From this we can see that if we look at this from the perspective of an "if else" point of view, we would be essentially saying that *if* this state, *then* take this action. However, our current state is the result of our previous action, which itself was predicated on our previous state. As we can see, in a dynamic temporal environment what we are defining as a rule is a set of consistently correlated state-action pair trajectories that lead to a desirable outcome. We can define this as,

$$Rule = Corr(\tau_s, \tau_a) \quad (4.3)$$

Where τ_s and τ_a are a trajectory of states and actions respectively. This definition allows us to evaluate the emergence of learnt rules in a DRL agent, which operates in visually complex environments. This was previously unattainable with prior methodologies, in part due to their abstraction of a rule to a semantic label.

While we have modelled this problem so far as a fully observable Markov Decision Process, the reality is slightly different. A DRL agent that is fed a stack of images as its input needs to learn where to look both spatially and temporally to extract the information that is known as the 'state'. As there is currently no way to accurately know if an agent has processed this information sufficiently to extract all the information that describes the entire 'state' of the environment, the

more accurate description here is that the agent processes the input to produce an observation of the state. As a result, two similar inputs may lead the agent to encode two very different observations, depending upon where the agent attends to the input. For this reason, we take the final layer before the logit layer as our observation of the state and use that in our analysis.

To evaluate the existence of learnt rules we sample observation and action trajectories of length m from a trained DRL agent and use k-means clustering on each independently to detect similarities over time. By using the elbow method, we are able to determine the number of clusters for our observations and then use the same number for our action clusters. We then use t-SNE dimensionality reduction on the observations to reduce them from 512 dimensions to two. This allows us to visualise the clusters found via our k-means clustering. In order to determine a correlation between the observations and actions we label each observation trajectory according to the cluster it has been assigned, and label each corresponding action trajectory with the same colour. This approach allows us to see not only how similar different observation trajectories are to each other, but also how similar their corresponding action trajectories are as well.

4.4 Experiments

4.4.1 Environments

We evaluate our approach in the Arcade Learning Environment [Bellemare et al., 2013], a purpose-built suite of discrete-time environments for bench-marking reinforcement learning algorithms. Specifically, we test our theory in two distinct environments. The first being 'Demon Attack', which involves a high level of stochastic behaviour from enemy agents in the environment. The second is 'Ms

Pacman', which contrary to the first environment contains enemies that operate in a much more deterministic manner. We observe the accepted standards set by [Mnih et al., 2015] with respect to processing of the observations obtained from the selected environments, which we detail below.

Pre-processing

As the environments are discrete-time, a new observation is only returned after an action (decided by $\pi(s_t)$) is taken by the agent. The observation in question is a 210x160 pixel RGB image of the environment. This observation is processed to represent the state of the environment by the following process.

- The observation is cropped from 210x160 to 84x84 pixels
- The observation is then converted from RGB to Gray-scale
- If this is the first observation from the environment, it is copied four times and stacked together to create an input tensor of size 84x84x4
- After four time-steps (for which the previously chosen action is repeated), a new observation is returned, cropped, converted, and stacked on the end of tensor. The oldest observation is removed from the tensor, maintaining the 84x84x4 shape.

As the Arcade Learning Environment is an emulator for interfacing with emulated Atari 2600 games, it is important to consider the memory limits of the original system. In particular, games would often alternate the display of certain features on and off, which although not discernible to the human eye, is discernible to a time-discrete agent and can potentially create a partially observable environment. In order to minimise this from happening, when returning a new observation

(O_t) , it is compared with the previous observation (O_{t-1}) , with the observation containing the maximum amount of information chosen as the observation to return.

4.4.2 Model Architecture

We test our definition and theory of a rule on two different model architectures. The first being the standard architecture as originally proposed by [Mnih et al., 2015]. This architecture consists of a total of three convolutional layers followed by a few fully connected layers. As we are using an Actor-Critic model our network has an Actor head and a Critic head. The second architecture we test is the Self-Attending Network (SAN) as proposed by [Manchin et al., 2019]. The SAN model incorporates a form of self-attention known as a non-local neural network [Wang et al., 2017] between the first two convolutional layers of the model.

As shown in figure 3.1, the input (consisting of four grayscale observations sequentially stacked together) are fed into a convolutional layer H_1 . The output of H_1 is then feed into the self-attention mechanism where,

- F_1 , F_2 , and G_1 are 1x1 convolutional layers with half the number of filters as H_1 .
- The outputs of F_1 and F_2 are matrix multiplied together before passing through a *Softmax* activation.
- This output is then matrix multiplied with the output of G_1 .
- Y_1 is a 1x1 convolutional layer with the same number of filters as H_1
- The output is added back into the original output of H_1 before passing into H_2 .

The output then passes through the convolutional layers H_2 and H_3 respectively, before being flattened and passed through fully connected layers.

4.4.3 Results

Rule Extraction

Having defined a rule previously in equation 4.3 as the consistent correlation between a trajectory of states and actions, we can now set about searching for the existence of these rules. Firstly, we gather state and action recordings from each DRL agent in their respective environments. From there we can create the state and action trajectories by concatenating a total of ten timesteps together. K-means clustering was then performed over the state trajectories to obtain the labels. The state trajectories and the action trajectories were then plotted using t-SNE, with both plots coloured according to the labels from the K-means clustering of the states. With this approach, and under this definition of a rule, the existence of such a rule would be evident if clear clusters are apparent and share the same label across both state and action trajectory plots.

From these plots (figures 4.1, 4.2, 4.3, and 4.4) we are able to observe a few interesting patterns. Firstly, both DRL agents in the Ms Pacman environment appeared to have been able to learn tighter more constrained rules than the agents operating in the Demon Attack environment. We believe this may be due to the difference in nature of both environments, specifically the level of stochastic behaviour displayed by the respective enemies. However, it is interesting to note that the state trajectories from the Demon Attack environment appear to form tighter clusters than that of the state trajectories from the Ms Pacman environment. Again, we believe this is due to the nature of each environment. In Demon Attack,

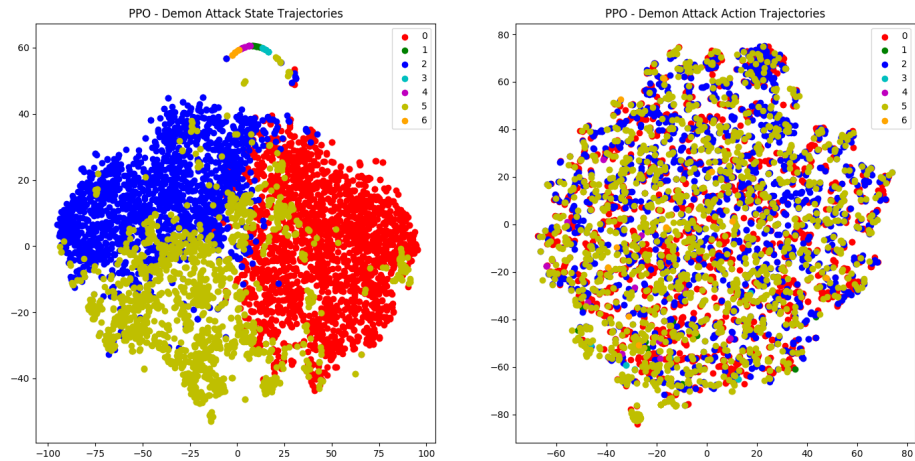


Figure 4.1: t-SNE plot of Demon Attack state trajectories (left), and action trajectories (right) trained using the baseline approach (PPO). Labelled according to k-means clustering performed on state trajectories.

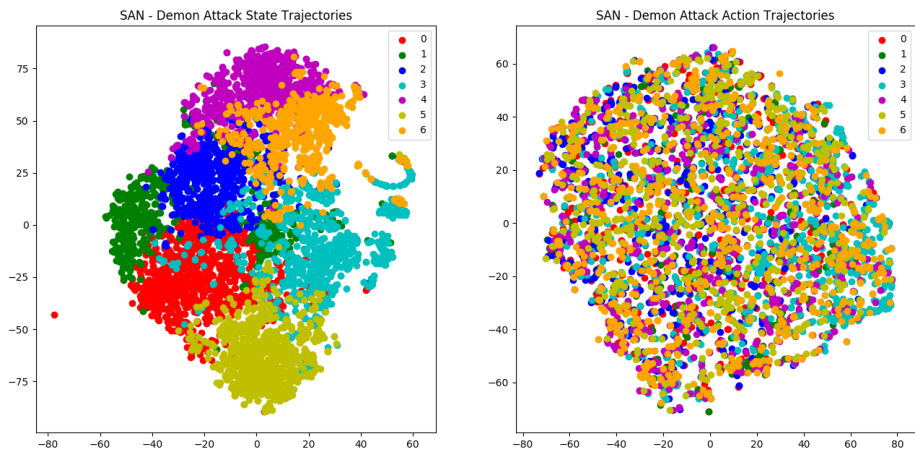


Figure 4.2: t-SNE plot of Demon Attack state trajectories (left), and action trajectories (right) trained using with self-attention (SAN). Labelled according to k-means clustering performed on state trajectories.

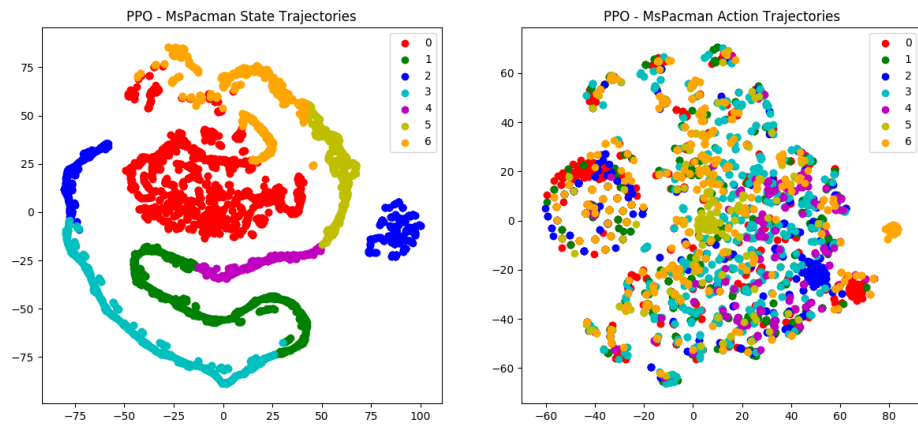


Figure 4.3: t-SNE plot of MsPacman state trajectories (left), and action trajectories (right) trained using the baseline approach (PPO). Labelled according to k-means clustering performed on state trajectories.

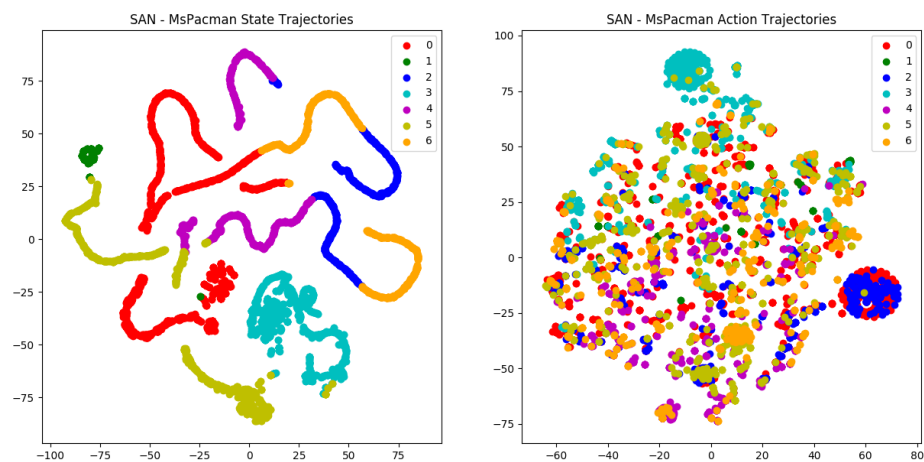


Figure 4.4: t-SNE plot of MsPacman state trajectories (left), and action trajectories (right) trained using with self-attention (SAN). Labelled according to k-means clustering performed on state trajectories.

as the agent progresses through the game it encounters new types of enemies. However, in Ms Pacman the enemies remain constant throughout. Additionally, no trained agent on Ms Pacman was ever able to complete the first two levels, which meant that the agents never encountered the change in level design that occurs on level three.

Now we investigate the impact of architecture design by comparing the results of a standard implementation (PPO) with the SAN model for each environment. Looking at figures 4.1 and 4.2 it is clear that the SAN model was able to distinguish clearer differences between states as it progressed through the game. However, both action trajectory plots appear quite noisy. We believe that this is due to the fact that although the enemies are changing in appearance as the game progresses, the strategy required to kill these enemies changes only slightly.

With respect to the Ms Pacman environment, we are able to identify clear correlations between state and action trajectories for both the standard PPO model and the SAN model. This is observable via the correlated state and action trajectories in figures 4.3 and 4.4. While both methods do ultimately learn rules which we are able to observe, the SAN model produces a more clearly defined set, in comparison to the standard PPO model.

Visual Analysis

By comparing the visualisation results of the baseline PPO agent against the SAN implementation, a number of insights were produced. These insights include an increased ability to track and attend to multiple targets, a better understanding of spatial information from the state input, and temporal attention in situations of partial visibility being displayed by the SAN agent and not seen in the baseline. Additionally, these situations of partial visibility change the underlying structure

of the problem from a Fully Observable Markov Decision Process (FOMDP), to a Partially Observable Markov Decision Process (POMDP). This significantly increase the difficulty of the problem the agent is trying to solve.

Figure 4.8a shows the baseline PPO agent paying a very small amount of attention to its surrounding area. While Figure 4.8b shows the SAN agent demonstrating attention of its nearby surroundings. It is also clear that the SAN agent is attending to the other side of the map, where enemies can possibly appear from. This level of spatial understanding was common for the SAN agent, however, was not observed from the baseline agent. Agents trained with SAN also show a higher level of ability when it comes to tracking multiple enemies or points of interest. Figure 4.6 shows the ability of agents trained using SAN to focus on multiple enemies in different environments. This is crucial for any agent to succeed, as was illustrated by [Greydanus et al., 2017] who showed examples of agents failing to attend to nearby enemies resulting in poor performance.

Referring to Figure 4.5, which shows a sequential time series of four frames, it can be seen that the enemy located in the center bottom of the map is only visible in frames (b) and (c). However, in frame (d) the agent attends to the spot where the enemy was last visible. As the agent has all four of the frames as input it can see the enemy in that location, just not on that frame. This is a trait that was only observed in agents trained using self-attention and is the first visualisations of temporal attention of a RL agent. Importantly the ability to 'look through time' for information is a big advantage in situations of partial observability.

It should be noted that the missing enemies in MsPacman were a surprise to the authors. No other game tested showed any sign of missing information yet was a clear and consistent problem in MsPacman. Best practices were followed with all implementations using the standard 'MaxandSkip' environment wrapper, unal-

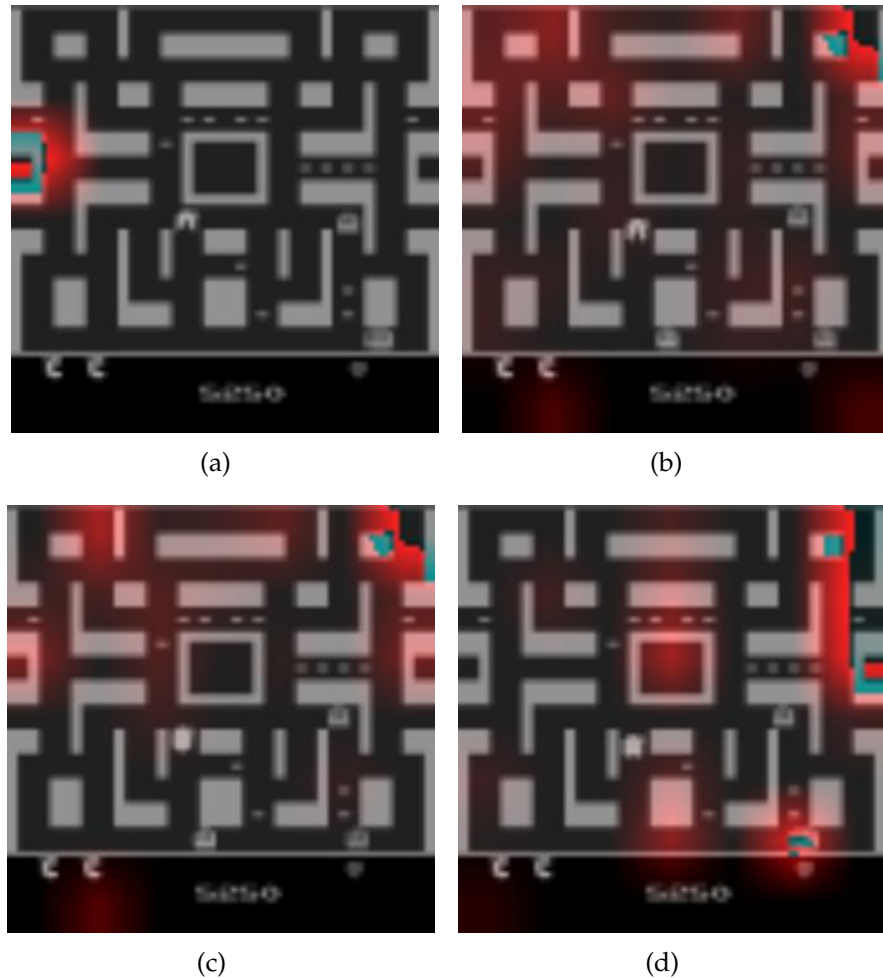


Figure 4.5: Here we can see that the agent is able to attend to multiple targets along with using temporal information to compensate for the missing enemy in the most recent frame. The attention also appears to bounce around, as the agent constantly searches the map. As the agent only has temporal information relating to four time steps, this is reasonable behaviour. The hard focus in the top corner is likely related to the fact that the 'super candy' appears in this spot, and like the enemies displays a blinking pattern. As the collecting of this reward has particular benefits to the agent it is not surprising to find the agent attending to this spot. Areas attended to by the agent for decision making are shown by adding the information from the activation map to the red channel of the image. This appears as either red or cyan and allows for the information in these areas to remain visible to the reader.

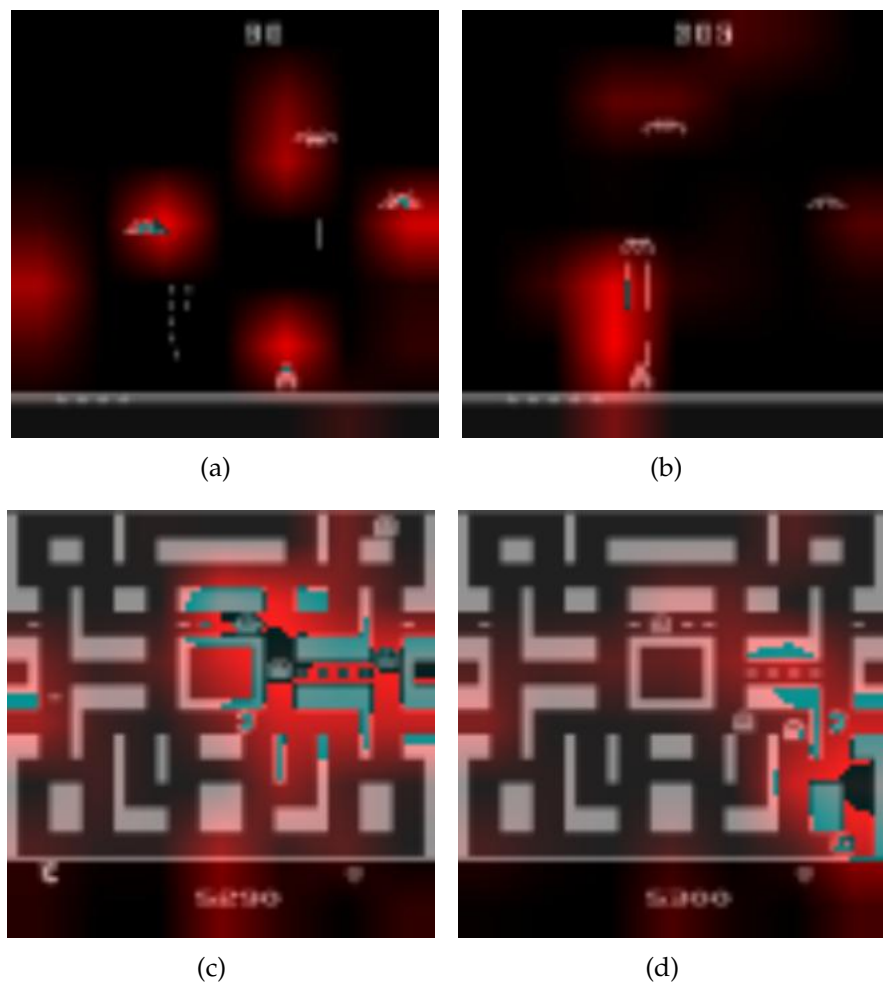


Figure 4.6: Examples of a trained SAN agent focusing on multiple enemies in different environments. a,b) Demon Attack. c,d) MsPacman. Areas attended to by the agent for decision making are shown by adding the information from the activation map to the red channel of the image. This appears as either red or cyan, and allows for the information in these areas to remain visible to the reader.

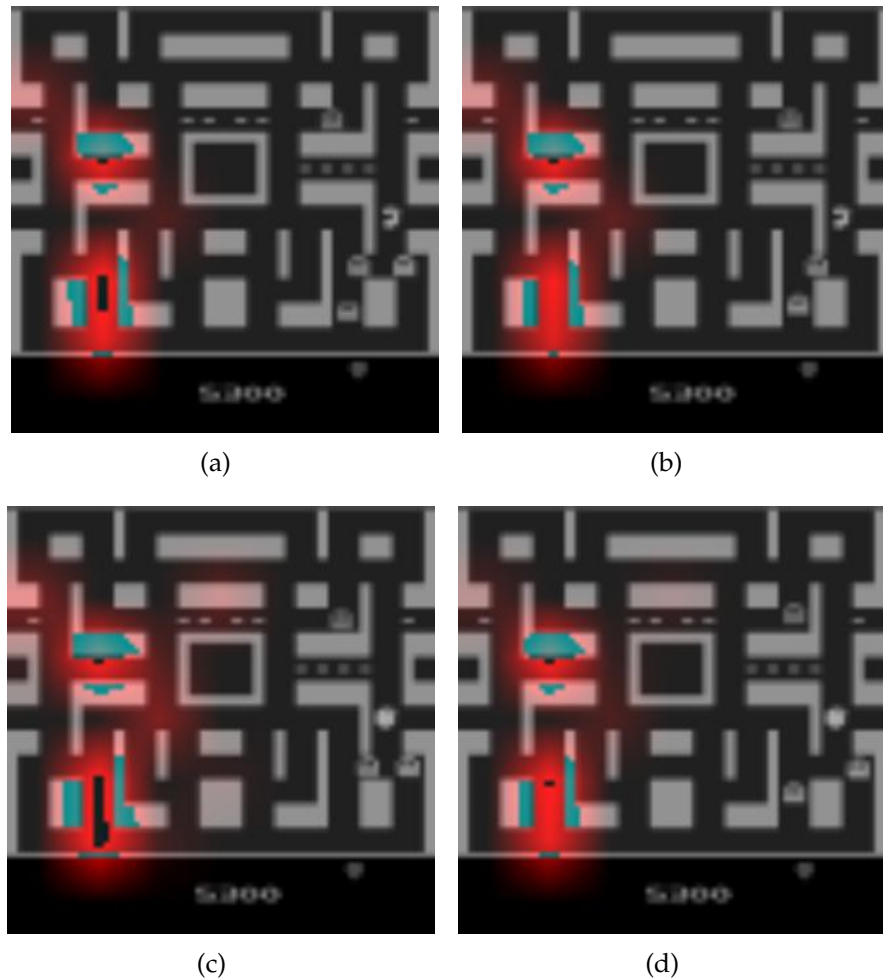


Figure 4.7: Examples of disappearing enemies in MsPacman. Enemies are located in the bottom right corner and appear sporadic. This clearly demonstrates the input constitutes a POMDP Areas attended to by the agent for decision making are shown by adding the information from the activation map to the red channel of the image. This appears as either red or cyan and allows for the information in these areas to remain visible to the reader.

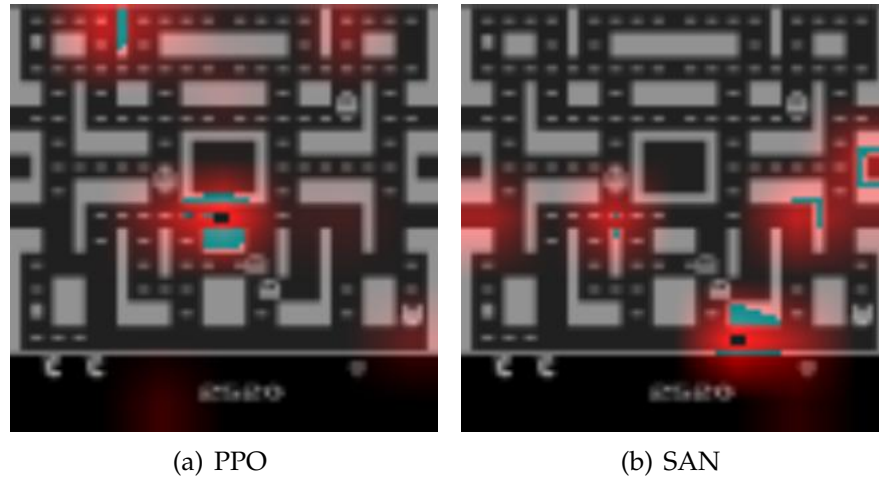


Figure 4.8: a) Baseline PPO agent failing to attend to areas of potential interest. b) SAN agent attending to nearby pathways, along with demonstrating spatial awareness by attending to the potential entry point on the other side of the map. This awareness was commonly seen in agents with self-attention. Areas attended to by the agent for decision making are shown by adding the information from the activation map to the red channel of the image. This appears as either red or cyan and allows for the information in these areas to remain visible to the reader.

tered as supplied from OpenAI ‘baselines’ [Dhariwal et al., 2017]. This wrapper is designed to ensure the environment conforms as a FOMDP, however it is clear that in this instance the environment is a POMDP. As Atari 2600 games have a tendency to display blinking sprites, due to the memory availabilities of technology at the time, the ‘MaxandSkip’ environment wrapper essentially compares two frames, separated by a single time step, and choose the one with the most information. However, it would appear that in the case of MsPacman, different enemies are blinking on and off during different frames. This would appear to result in the environment wrapper choosing the best of a bad situation.

Interestingly MsPacman is one of the few environments left where reinforcement learning has not yielded an agent capable of surpassing the average human level of performance. It is possible that this is due to the POMDP nature of the environment. Observations towards this can be seen in Fig. 4.7 where enemies

appear to jump sporadically but are in fact different enemies blinking on and off at different frames. We also observed scenarios where enemies disappeared for a total of eight frames prior to the agent making contact, resulting in the death of the agent. As the agent only takes in a history of four frames, it was impossible for the agent, even with temporal attention to avoid this situation.

4.5 Conclusion and Future Work

The problem of extracting rules from simple neural networks has been studied from a pedagogical, decompositional, and eclectic viewpoint. However, study into deep neural networks, and even more specifically, deep reinforcement learning agents has been limited or non-existent. In this chapter we proposed a mathematical definition for a rule that transcends the semantics of a natural language label often employed to describe rules. We evaluate our definition by evaluating it against DRL agents across multiple visually complex environments, something previously not possible with prior definition and methods. Additionally, we test and show that DRL agents that incorporate attention mechanisms into their underlying network architecture can learn more defined rule sets.

The results from these experiments reveal interesting questions for future work to answer. In particular, although our method was able to determine the existence of learnt rules, it is not able to convert them into an explainable natural language form. While previous work in this area has involved obtaining human provided labels for such things, an interesting question would be if an explanation could be provided without a human in the loop.

Bibliography

- [Augasta and Kathirvalavakumar, 2012] Augasta, M. G. and Kathirvalavakumar, T. (2012). Reverse engineering the neural networks for rule extraction in classification problems. *Neural processing letters*, 35(2):131–150.
- [Bellemare et al., 2013] Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.
- [Brockman et al., 2016] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540*.
- [Craven and Shavlik, 1996] Craven, M. and Shavlik, J. (1996). Extracting tree-structured representations of trained networks. In Touretzky, D., Mozer, M. C., and Hasselmo, M., editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press.
- [Dhariwal et al., 2017] Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. (2017). Openai baselines. <https://github.com/openai/baselines>.

- [Fukuchi et al., 2017] Fukuchi, Y., Osawa, M., Yamakawa, H., and Imai, M. (2017). Autonomous self-explanation of behavior for interactive reinforcement learning agents. *Proceedings of the 5th International Conference on Human Agent Interaction*.
- [Greydanus et al., 2017] Greydanus, S., Koul, A., Dodge, J., and Fern, A. (2017). Visualizing and understanding atari agents. *CoRR*, abs/1711.00138.
- [Hayes and Shah, 2017] Hayes, B. and Shah, J. A. (2017). Improving robot controller transparency through autonomous policy explanation. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 303–312.
- [Huang et al., 2011] Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., and Tygar, J. D. (2011). Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58.
- [Kamruzzaman and Islam, 2010] Kamruzzaman, S. M. and Islam, M. M. (2010). Extraction of symbolic rules from artificial neural networks. *CoRR*, abs/1009.4570.
- [Kurakin et al., 2016] Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.
- [Manchin et al., 2019] Manchin, A., Abbasnejad, E., and van den Hengel, A. (2019). Reinforcement learning with attention that works: A self-supervised approach. In Gedeon, T., Wong, K. W., and Lee, M., editors, *Neural Information Processing*, pages 223–230, Cham. Springer International Publishing.
- [Markov, 1957] Markov, A. A. (1957). Theory of algorithms. *Journal of Symbolic Logic*, 22(1):77–79.

- [Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.
- [Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
- [Sato and Tsukimoto, 2001] Sato, M. and Tsukimoto, H. (2001). Rule extraction from neural networks via decision tree induction. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No.01CH37222)*, volume 3, pages 1870–1875 vol.3.
- [Selvaraju et al., 2016] Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., and Batra, D. (2016). Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR*, abs/1610.02391.
- [Su et al., 2019] Su, J., Vargas, D. V., and Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841.
- [Thrun, 1995] Thrun, S. (1995). Extracting rules from artificial neural networks with distributed representations. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems (NIPS) 7*, Cambridge, MA. MIT Press.
- [Tickle et al., 1994] Tickle, A. B., Orłowski, M., and Diederich, J. (1994). Dedec: decision detection by rule extraction from neural networks. *QUT NRC*.

- [Verma et al., 2018] Verma, A., Murali, V., Singh, R., Kohli, P., and Chaudhuri, S. (2018). Programmatically interpretable reinforcement learning. In *ICML*, pages 5052–5061.
- [Vessey, 1985] Vessey, I. (1985). Expertise in debugging computer programs: A process analysis. *Int. J. Man Mach. Stud.*, 23:459–494.
- [Vilone and Longo, 2020] Vilone, G. and Longo, L. (2020). Explainable artificial intelligence: a systematic review. *arXiv preprint arXiv:2006.00093*.
- [Wang et al., 2017] Wang, X., Girshick, R. B., Gupta, A., and He, K. (2017). Non-local neural networks. *CoRR*, abs/1711.07971.
- [Weitkamp et al., 2019] Weitkamp, L., van der Pol, E., and Akata, Z. (2019). Visual rationalizations in deep reinforcement learning for atari games. *CoRR*, abs/1902.00566.
- [Zilke et al., 2016] Zilke, J. R., Loza Mencía, E., and Janssen, F. (2016). Deepred – rule extraction from deep neural networks. In Calders, T., Ceci, M., and Malerba, D., editors, *Discovery Science*, pages 457–473, Cham. Springer International Publishing.

CHAPTER 5

Deep Inductive Reasoning for Video to Program Translation

The work contained in this chapter is in submission as the following paper:

Manchin, A., Sherrah, J., Wu, Q., Van den Hengel, A., Deep Inductive Reasoning for Video to Program. In *Submission CVPR2022*

Statement of Authorship

Title of Paper	Program Generation from Diverse Video Demonstrations
Publication Status	<input type="checkbox"/> Published <input type="checkbox"/> Accepted for Publication <input checked="" type="checkbox"/> Submitted for Publication <input type="checkbox"/> Unpublished and Unsubmitted work written in manuscript style
Publication Details	Manchin, A., Sherrah, J., Wu, Q., Van den Hengel, A., Deep Inductive Reasoning for Video to Program. In Submission BMVC2022

Principal Author

Name of Principal Author (Candidate)	Anthony Manchin
Contribution to the Paper	- Development of the main idea of the paper - Implementing and conducting the experiments - Writing and revising of the paper
Overall percentage (%)	80%
Certification:	This paper reports on original research I conducted during the period of my Higher Degree by Research candidature and is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in this thesis. I am the primary author of this paper.
	Date 20 July 2022

Co-Author Contributions

By signing the Statement of Authorship, each author certifies that:

- the candidate's stated contribution to the publication is accurate (as detailed above);
- permission is granted for the candidate to include the publication in the thesis; and
- the sum of all co-author contributions is equal to 100% less the candidate's stated contribution.

Name of Co-Author	Qi Wu
Contribution to the Paper	- Help with the development of the idea - Help writing, revision, and discussions
	Date 20 Jul 2022

Name of Co-Author	Jamie Sherrah
Contribution to the Paper	- Help with the development of the idea and discussions
	Date 22 Jul 2022

Please cut and paste additional co-author panels here as required.

Name of Co-Author	Anton van den Hengel
Contribution to the Paper	- Discussions regarding the ideas and revisions
Signature	Date 26 Jul 2022

Abstract

The ability to use inductive reasoning to extract general rules from multiple observations is a vital indicator of intelligence. As humans, we use this ability to not only interpret the world around us, but also to predict the outcomes of the various interactions we experience. Generalising over multiple observations is a task that has historically presented difficulties for machines to grasp, especially when requiring computer vision. In this paper, we propose a model that can extract general rules from video demonstrations by simultaneously performing summarisation and translation. Our approach differs from prior works by framing the problem as a multi-sequence-to-sequence task, wherein summarisation is learnt by the model. This allows our model to utilise edge cases that would otherwise be suppressed or discarded by traditional summarisation techniques. Additionally, we show that our approach can handle noisy specifications without the need for additional filtering methods. We evaluate our model by synthesising programs from video demonstrations in the Vizdoom environment achieving state-of-the-art results with an increase of 11.75% program accuracy on prior works.

5.1 Introduction

Inductive reasoning involves using logic to extract general rules from multiple observations and is a skill that is widely viewed as an indicator of intelligence. Humans (and many species of animals) are known to possess the ability to observe demonstrations and subsequently use inductive reasoning to acquire new knowledge and skills without requiring explicit instruction. An example of this behaviour would be children learning to play video games, wherein one is able to observe another player, and learn the rules of the game without having to play the game themselves.

Additionally, children are also able to abstract and generalise information they have induced from one video game and apply that same logical rule set to a completely different virtual domain. An example of this may be that someone observes that a red cross symbol indicates a health related bonus in one game, and subsequently uses that information to infer that similar symbols in a different game are also related to player health. Implying general rules from complex observations however has proven to be a difficult task for machines. In-fact until recently, very little work had been published on extracting general rules from a diverse range of visual observations. Advances in both computer vision and machine learning techniques however are changing this.

To help excel the study of inductive reasoning with artificial intelligent systems we seek to create a model that can generate executable code by inferring the specifications from multiple visual demonstrations. The goal of creating a model capable of generating executable code has long been a dream of artificial intelligence researchers [Waldinger and Lee, 1969, Gulwani, 2011]. However, until recently research has primarily focused on the generation of code, *given* the desired

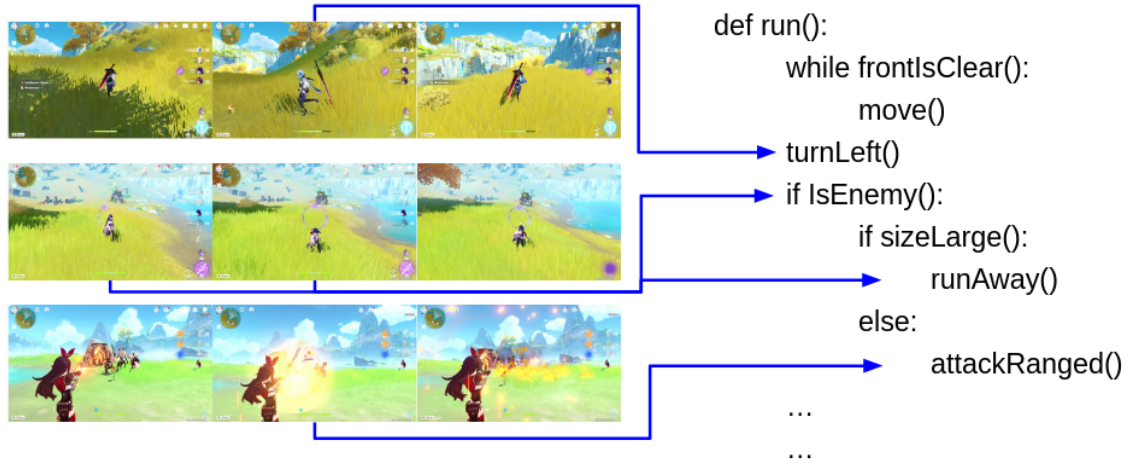


Figure 5.1: An illustration of the task of visual program synthesis on the game ‘Genshin Impact’. Humans can infer a general rule set from simply observing examples of game play. The task of visual program synthesis involves training a machine that can synthesis a program that correctly captures a rule set simply from watching visual demonstrations of agent.

specifications [Jha et al., 2010, Alur et al., 2013]. However, the problem becomes much more difficult when the model is also required to infer the specifications for itself. This task, originally proposed by [Sun et al., 2018], presents a unique challenge as it requires a model to accurately detect the relevant semantics of a demonstration, understand the relationships between demonstrations, define a set of specifications that captures this information, and finally generate a program that satisfies these specifications.

Previous approaches [Sun et al., 2018, Duan et al., 2019] have considered framing this as a sequence-to-sequence task and have utilised the once popular recurrent neural network architecture of long short term memory units as the backbone of their models. However, these approaches were restricted by the limitations of these recurrent based models to handle long sequences. This limiting factor resulted in both [Sun et al., 2018, Duan et al., 2019] using summarised latent space representations, which undoubtedly limited their models ability to completely

capture the entire specification constraints. Meanwhile [Dang-Nhu, 2020] sought to use a rule-based solver to generate the desired code, which receives its specifications from a neural model. As rule-based solvers are very susceptible to noise, [Dang-Nhu, 2020] proposed using a dynamic filtering method to de-noise specifications.

To address these limitations, we propose ‘video-to-text transfer transformer’ (VT4). Taking inspiration from the works of [Raffel et al., 2019, Sun et al., 2019a] we leverage the ability of attention based transformer networks to handle long and disjointed sequences, removing the need to summarise features. Specifically, we formulate a ‘visual language’ which encapsulates the relevant semantic information from each demonstration. We then simultaneously feed all the demonstrations (represented in a visual language) into our encoder-decoder transformer network which learns to summarise and translate the demonstrations into an executable program. Additionally we show that our approach is able to handle noisy specifications without the need for additional filtering methods. We evaluate our model in a partially observable virtual environment (Vizdoom) [Kempka et al., 2016] and demonstrate that our approach is able achieve state-of-the-art results by out-performing previous summarisation and rule-based approaches. We observe a relative increase of 15.45% and 11.75% over summarisation and rule-based approaches respectively.

Contributions

- We present video-to-text transfer transformer for the task of executable program generation from video demonstrations. Addressing the limitations of previous summarisation and rule-based approaches, VT4 can generate programs from a long, disjointed set of demonstrations without the need for

a summarised representation of the average demonstration.

- We evaluate the effects of noisy specifications on program accuracy and show that our model is robust to significant levels of erroneous detections. State-of-the-art results were achieved with a 10% error rate for perception primitives. Significantly out-performing previous rule-based models which strongly rely on dynamic filter for noise reduction.
- We evaluate our model’s ability to generate programs from partially observable, visually complex demonstrations. We achieve increases of 9% and 11.75% for exact and aliased program accuracy relative to previous state-of-the-art. This translates to absolute increases of 5.3% and 7.7% respectively and represents the largest single increase in performance on this task to date.

5.2 Literature Review

The task of video understanding [Lin et al., 2019, Wu et al., 2019] can be viewed as a subset task of program generation from video demonstrations (PGfVD). As with PGfVD, the understanding of videos requires the ability to extract and understand the correlations between events and features. This is often achieved through models that can perform tasks such as action and perception recognition [Kay et al., 2017]. However, unlike PGfVD, video understanding aims to describe *what* has been observed in a single demonstration, not *why* something has happened. For the most part, this is a straightforward translation task that can benefit from a large amount of acceptable ambiguity [Dong et al., 2019]. This is largely because for the task of video translation, multiple captions may be appropriate and considered semantically correct. However, for the task of PGfVD, the video demonstration may only display a single component of the overall rule set that one is trying to

learn. Additionally, the task of PGfVD has much lower levels of ambiguity, as slight changes to a program can result in greatly different outcomes.

5.2.1 Program Induction

The task of extracting algorithmic representations from observations is known as program induction. Various works have contributed to this field with a diverse set of approaches aimed at solving this problem. [Graves et al., 2014, Kaiser and Sutskever, 2016, Kurach et al., 2016] use memory based approaches such as Turing machines while [Ling et al., 2017] adopt end-to-end networks to solve and explain algebraic word problems. However, in contrast to these approaches, we aim to generate a fully defined executable program in a domain specific language.

5.2.2 Intrinsic Motivation

The study of intrinsic motivation, or sometimes known as curiosity, in the field of reinforcement learning can also be seen as related work. The goal of intrinsic motivation is for an agent to learn about the environment in such a way as to actively seek out new and novel scenarios to explore. To do this the agent attempts to learn a mapping from states to actions in order to predict the novelty of taking certain actions at different states. Works from [Burda et al., 2018a, Burda et al., 2018b] have shown that learning this mapping can be beneficial to the performance of an agent, and its ability to generalise over multiple domains. These results would infer that intrinsic motivation helps the policy to learn general rules regarding the dynamics of the environment, or simple cause and effect relationships.

5.2.3 Program Synthesis

The aim of program synthesis is to generate a program that captures the underlying logic of given examples. Typically, this task has restricted the programs to simple domain specific languages and has involved producing an abstract syntax tree. Examples of this work include [Parisotto et al., 2016], who proposed using Recursive-Reverse-Recursive neural networks (R3NN) for string transformation. Other work has paired neural models with search algorithms and rule based solvers [Balog et al., 2016, Dang-Nhu, 2020]. Additionally reinforcement learning has also been investigated as a possible way to solve the task of program synthesis [Bunel et al., 2018, Simmons-Edler et al., 2018].

However, most of the work in this field does not consider the task of synthesising programs from visual observations. [Sun et al., 2018] identified this and proposed the task of generating a program from observing a diverse range of visual demonstrations. To achieve this goal [Sun et al., 2018] proposed using a sequence-to-sequence LSTM model. Their model consisted of convolutional neural network which fed an LSTM network encoded video frames. [Sun et al., 2018] introduced a combination of average pooling and a relational network to summarise the encoded demonstrations, which were subsequently passed to a LSTM decoder. [Duan et al., 2019] aimed to improve the computational efficiency of [Sun et al., 2018] approach by introducing a deviation-pooling summariser to replace the relation network. Additionally, [Duan et al., 2019] proposed using multiple decoding layers to refine the accuracy of the generated program which ultimately resulted in a slight improvement in performance.

[Dang-Nhu, 2020] took a different approach to this problem, proposing a hybrid model which combined a neural network to extract specifications and

a rule-based solver to generate the program. In particular, [Dang-Nhu, 2020] proposed using a convolutional neural network to encode the video frames, and two different decoder layers to predict the perceptions and actions observed in the videos. [Dang-Nhu, 2020] correctly identifies the sensitivity of rule-based solvers to input specification noise [Devlin et al., 2017], and proposed a dynamic filtering method to ignore certain demonstrations based on the confidence level of the neural networks predictions. The inclusion of a dynamic filter allowed [Dang-Nhu, 2020] to surpass the performance of previous LSTM based program generators.

5.3 Method

This section first presents a formal definition for the task of generating programs from video demonstrations, as originally proposed by [Sun et al., 2018], before describing the proposed VT4 model, and our contributions in detail.

5.3.1 Program Generation

The goal of generating a program given k video demonstration can be considered a multi-sequence-to-sequence task. A domain specific language (DSL) is used to define a program which consists of perception primitives, action primitives and control flow statements. Action and perception primitives define the way an agent can interact with and perceive the environment respectively. The control flow statements of the DSL language include while loops, repeat and if/else statements, and simple logic operations.

Program

A program π is defined as a deterministic function, which given an input of state $s \in \mathcal{S}$ at time t , returns an action $a \in \mathcal{A}$.

$$\pi(s_t) = a_t \quad (5.1)$$

For this task we limit the parameters of the program to a vectorised DSL, which we denote as $\theta \in \Theta$. The parameters are what would typically be referred to as the ‘code’ of the program.

Demonstrations

A demonstration is defined as a sequence of state-action pairs sampled from $\pi(\theta)$

$$\tau = ((s_1, a_1), (s_2, a_2), \dots, (s_T, a_T)) \quad (5.2)$$

However, there is no guarantee that any particular demonstration generated by an agent following a program $\pi(\theta)$ will provide examples of all control flow statements present in $\pi(\theta)$. If we consider the case where an agent following $\pi(\theta)$ (see Figure 5.2) does not encounter a state wherein the perception primitive ‘*HellKnight*’ is *True*, there is no way to induce the existence of that control flow statement from that demonstration. Therefore it is necessary to observe a set of demonstrations $\mathcal{T} = (\tau_1, \tau_2, \dots, \tau_k)$ where $\mathcal{T} \subset \mathfrak{T}$ that contains transition examples of all control flow statements in $\pi(\theta)$.

Perceptions

Given the conditional structure of the DSL used to define the parameters θ for program π , we employ the use of perception primitives to simplify the high dimen-

sionality of states $s \in \mathcal{S}$. Each perception primitive μ is given as a Boolean value, with q possible perception primitives. This allows for a high level representation of the state to be given by the vector $s = (\mu_1, \mu_2, \dots, \mu_q)$.

Actions

For an agent interacting in discrete time steps with a deterministic environment of previously defined states $s \in \mathcal{S}$, we can define a finite set of possible actions \mathcal{A} . This allows us to model the deterministic transition between states as $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$. In our case we only have one possible action per transition, and as such we can represent an action primitive as a one hot tensor of length m , where m is equal to the total number of possible actions.

Visual Language

Having defined both perception and action primitives, we can use these to define a visual language of semantic tokens $\psi \in \Psi$ with a deterministic function $F(x, y)$ such that,

$$\psi_t = F(s_t, a_t) \tag{5.3}$$

This approach of tokenizing high level semantic information for ease of use with transformer models has been shown to be very effective by [Sun et al., 2019b]. With this we can also substitute our transition tokens into equation 5.2 giving,

$$\tau = (\psi_1, \psi_2, \dots, \psi_T) \tag{5.4}$$

```

def run():
    while isTargetDemon:
        moveForward()
    if isTargetRevenant:
        moveRight()
    else:
        if isTargetHellKnight:
            shoot()
        else:
            moveLeft()

```

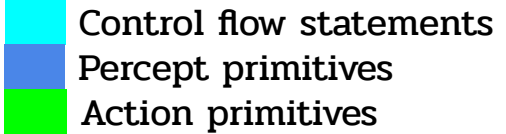


Figure 5.2: A Domain specific language program example from the Vizdoom environment. Examples of all component types are present.

Supervision Signals

We abide by the constraints originally proposed by [Sun et al., 2018] and assume that action and perception labels are absent during testing. However, we form the same conclusion as [Sun et al., 2018, Duan et al., 2019, Dang-Nhu, 2020] that during training the action and perception signals are required for the model to learn how to generate the I/O specifications of the program.

5.3.2 VT4 Model

Our VT4 model can be separated into two main sections; i) the semantic encoder and ii) the program generator network. Figure 5.3 gives an overview of the VT4 model. We approach this problem of generating an executable program from video demonstrations as a combined translation and summarisation task. In contrast to previous approaches which simplify the problem to a straight sequence-to-sequence task, (by creating a summarised expression of the multiple demonstrations) our approach frames the problem as the multi-sequence-to-sequence task

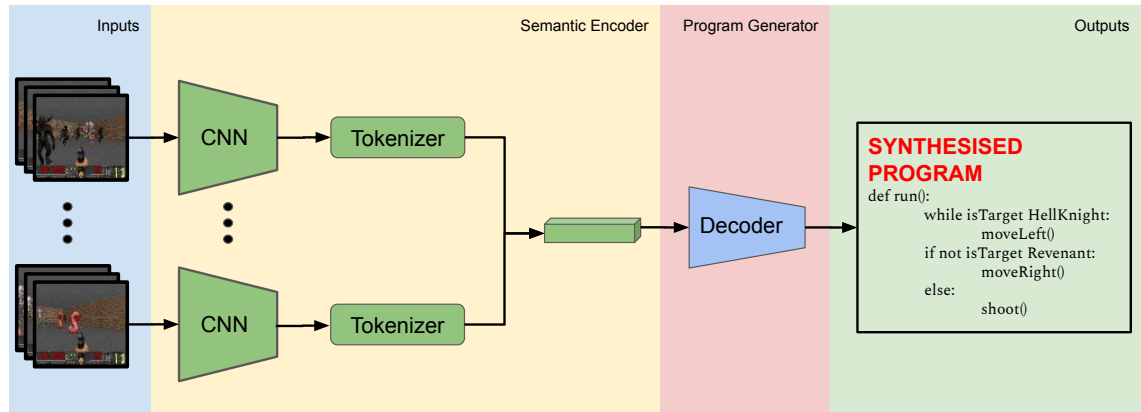


Figure 5.3: A complete diagram of the VT4 architecture showing the semantic encoder and the program generator modules. The semantic encoder takes as input raw videos and processes them into a visual language of semantic tokens. These tokens are then passed to the program generator which learns to extract the underlying rule set from the demonstrations and generate a program accordingly.

that it is. This eliminates the inherent probability of information loss associated with summarising multiple diverse demonstrations.

Semantic Encoder

The Semantic Encoder itself can be separated into two parts: a neural module and a tokenizer. The neural module is a multi-layer convolutional neural network (CNN) which learns to detect the perception primitives present in each frame, and the actions taken between frames. In theory this module could contain one CNN with two different output heads, each predicting either the actions or perceptions. For the sake of simplicity this is how we present our model in figure 5.3 However, in practise we use two distinct networks for each task. This choice was made purely to simplify the process of altering the perception prediction accuracy for the noise-ablation study. The action prediction network consists of a five-layer convolutional neural network with two fully connected layers, and is trained from scratch. The perception prediction network utilises a pre-trained Efficientnet

model [Tan and Le, 2019].

Given a set of video demonstrations $\mathcal{V} = \{v_i\}_{i=1}^k$, we desire the corresponding perceptions p and actions a sequences for each demonstration. This is a straight forward problem which is modelled as;

$$p_{i,j} = MLP(CNN(v_{i,j})) \quad (5.5)$$

$$a_{i,j} = MLP(CNN(v_{i,j}), CNN(v_{i,j+1})) \quad (5.6)$$

Where i and j refer to the i^{th} demonstration and j^{th} frame. While it is possible to predict all the observable perceptions from a single frame $v_{i,j}$ with purely spatial information, this is not the case for predicting actions. To predict the action taken at any time t temporal information in the form of a minimum of two frames is required. Thus, we concatenated sequential frames together as shown in equation 5.6 for action prediction.

Having now obtained the predicted actions $a_{i,j}$ and perceptions $p_{i,j}$ for each frame of every video, we are able to use these to create semantic tokens which completely encapsulates all the required information from each frame. We achieve this by passing our predictions through the second part of the semantic encoder which we refer to as a tokenizer. The tokenizer itself is a deterministic function that takes as input the predicted action and perceptions of each frame individually.

$$\psi_{i,j} = F(p_{i,j}, a_{i,j}) \quad (5.7)$$

As described in section 5.3.1, our perception prediction is a multiclass prediction problem given as $p_{i,j} = (\mu_1, \mu_2, \dots, \mu_q)$, while our action prediction returns a single class prediction. Our tokenizer first concatenates our predictions into a

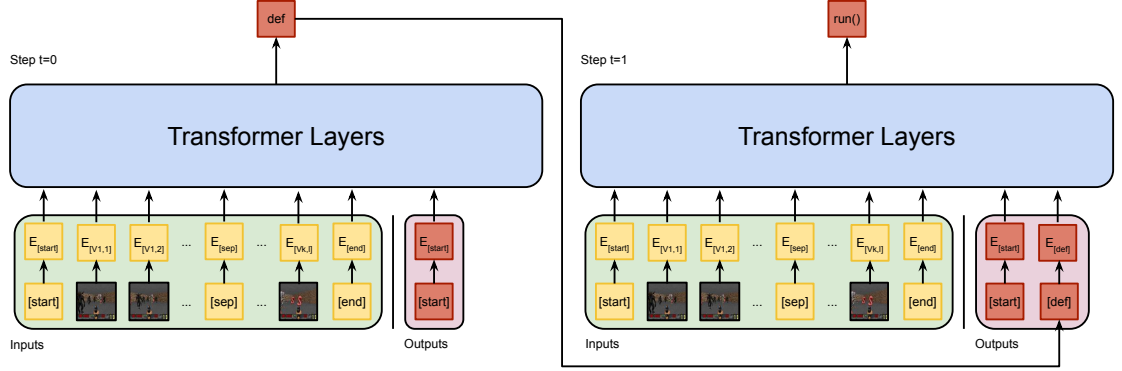


Figure 5.4: The Program Generator Module. The program generator consists of an encoder-decoder transformer network which receives as input the visual language tokens from the semantic encoder. The network iteratively decodes the program in a sequential fashion.

single tensor pa , before calculating the finite sum of a basic power series.

$$\psi_{i,j} = \sum_{n=0}^n pa_n \times 2^n \quad (5.8)$$

Our demonstrations are now encoded into a sequence of semantic tokens which encapsulates all the semantic information derived from the original video frames.

While prior works have decided to summarise the multiple demonstrations into a single average representation, we choose not to pursue this path. Instead, we pass every tokenised demonstrations into our program generator network as a set of disjoint sequences. This approach of creating ‘visual words’ [Sun et al., 2019b] allows us to take inspiration from the field of natural language processing and concatenate our demonstrations together separating the different demonstrations with special tokens. $V = (\langle start \rangle, \psi_{1,1}, \psi_{1,2}, \dots, \psi_{1,l} \langle sep \rangle, \dots, \psi_{k,l}, \langle end \rangle)$

Program Generator Network

Due to the wide success of Transformer based architectures [Vaswani et al., 2017] in various settings such as machine translation [Wang et al., 2019, Ahmed et al., 2017], natural language processing [Raffel et al., 2019, Devlin et al., 2018], and even more recently image classification [Dosovitskiy et al., 2020], we hypothesise that they should also be highly effective for the task of program generation. As such our program generator is an encoder-decoder transformer network. By utilising the insights of [Sun et al., 2019b] and converting our inputs into a visual language, we can leverage pre-trained language models. We choose to leverage a pre-trained ‘Text-to-Text Transfer Transformer’ (T5) network as proposed by [Raffel et al., 2019].

The T5 model is particularly well suited to our task as it specifically (as the name suggests) casts all tasks as a text-to-text task. Due to this we can leverage pre-trained features from upstream natural language tasks such as summarisation and translation. The T5 implementation closely follows the originally proposed architecture of [Vaswani et al., 2017]. The encoder consists of a stack of ‘blocks’, wherein each block is comprised of a self-attention, layer normalisation, and a feed-forward network. The decoder has a similar structure, except that it includes an additional standard attention mechanism. This standard attention mechanism is applied after the self-attention layer and attends to the output from the encoder. Figure 5.4 gives a high level overview of generative sequence of the transformer network, while for full details we refer the reader to [Raffel et al., 2019] and [Vaswani et al., 2017] respectively.

5.4 Experiments

In this section we present the experimental evaluation of our model. We include an overview of the dataset and metrics used for this evaluation. We then go on to discuss the results of our experiments along with providing a noise-ablation study to evaluate our model’s ability to handle noisy inputs.

5.4.1 Dataset and Metrics

Dataset

We evaluate our model with the Vizdoom Program Dataset [Sun et al., 2018] which has the following structure. For every program label there exists multiple video demonstrations of an agent following said program in the deterministic virtual environment known as Vizdoom [Kempka et al., 2016]. For every demonstration (of length T), there exists action and perception labels of lengths $T - 1$ and T respectively. We conform with the previously established convention set by [Sun et al., 2018] and utilise a total of twenty-five demonstrations per program during testing and training. We also adhere to the assumption that action and perception labels are only available during training, and that at test time we only have access to the video demonstrations of the agent. The dataset contains 80,000 programs for training, and 8,000 programs for testing.

Metrics

As the problem of verifying that two programs are in fact equal is an intractable problem, we evaluate the accuracy of our model by comparing the synthesised parameters $\hat{\theta}$ with the instantiated parameters of the ground truth program θ^* . In

particular we evaluate the *exact accuracy* and the *aliased accuracy*. We break with the previous naming convention of these accuracy measures (originally established by [Sun et al., 2018] as *sequence* and *program* accuracy) for the sake of clarity.

Exact Accuracy

We consider a program to be an exact match if, and only if, the synthesised parameters $\hat{\theta}$ is an exact match to that of the instantiated ground truth parameters θ^* . This is formally expressed as,

$$Acc_{exact} = \frac{1}{N} \sum_{n=1}^N 1_{exact}(\hat{\theta}, \theta^*) \quad (5.9)$$

Aliased Accuracy

While the exact accuracy is a simplistic measure of the performance of our model, it does not account for the ambiguity of the program space. Take for example two different yet semantically identical pieces of code; `repeat (2) : (move ())` and `move () move ()`. Here it is obvious that both pieces of code are semantically identical, however they would not be counted as the same given the definition of exact accuracy. As identified by [Sun et al., 2018], it is possible to exploit the simplistic syntax of our DSL and enumerate multiple variations of the code following a set of defined rules. Examples of this include decomposing control flow statements such as if/else statements, and unfolding repeat statements. With this we can formally express our accuracy as

$$Acc_{alias} = \frac{1}{N} \sum_{n=1}^N 1_{alias}(\hat{\theta}, \theta^*) \quad (5.10)$$

Model	Exact	Aliased
demo2program [Sun et al., 2018]	53.2	62.5
watch-reason-code [Duan et al., 2019]	55.8	63.4
PLANS (dynamic) [Dang-Nhu, 2020]	58.8 \pm 0.6	65.5 \pm 0.6
VT4 (ours)	64.1	73.2

Table 5.1: An exact comparison of our results compared to the results of previously published works.

5.4.2 Overall Performance

We display the results of our evaluation on the Vizdoom benchmark in table 5.1. We obtain the values for the baselines from prior published works, and report best obtained performance. We strictly adhere to the experimental settings originally proposed by [Sun et al., 2018] to provide a fair comparison. Our VT4 model significantly improves on the exact and aliased program accuracy of the prior state-of-the-art with relative 9% and 11.75% increases and 5.3% and 7.7% absolute increases respectively. These results provide clear evidence of the capabilities of our model to simultaneously perform summarisation and translation of disjointed sequences. These results also provide empirical evidence to support the use of visual languages to describe the semantics of complex scenes for tasks requiring translation. Additionally, we observe the expected result of higher accuracy for aliased programs compared to exact programs as consistently observed across all prior works. We also investigate claims by [Duan et al., 2019] and [Dang-Nhu, 2020] that the original algorithms used by [Sun et al., 2018] fails to recognise all semantically identical programs. In particular are cases such as the example shown in figure 5.5. While we find these claims to be accurate, we report our results using the same metrics to keep our comparison fair. However, we strongly agree with [Dang-Nhu, 2020] on the need to improve this metric in future work.

<p>GROUND TRUTH</p> <pre>def run(): while isTarget Hellnight: moveLeft() if isTarget Revenat: shoot() else: moveRight()</pre>
<p>SYNTHESISED PROGRAM</p> <pre>def run(): while isTarget Hellnight: moveLeft() If not isTarget Revenat: moveRight() else: shoot()</pre>

Figure 5.5: A comparison of the ground truth program and a synthesised program from our VT4 model. While these two programs are semantically equal, the algorithm used by [Sun et al., 2018] considers these as different. To keep evaluation fair we report results using the same base code wherein this situation is considered a ‘failure’ case.

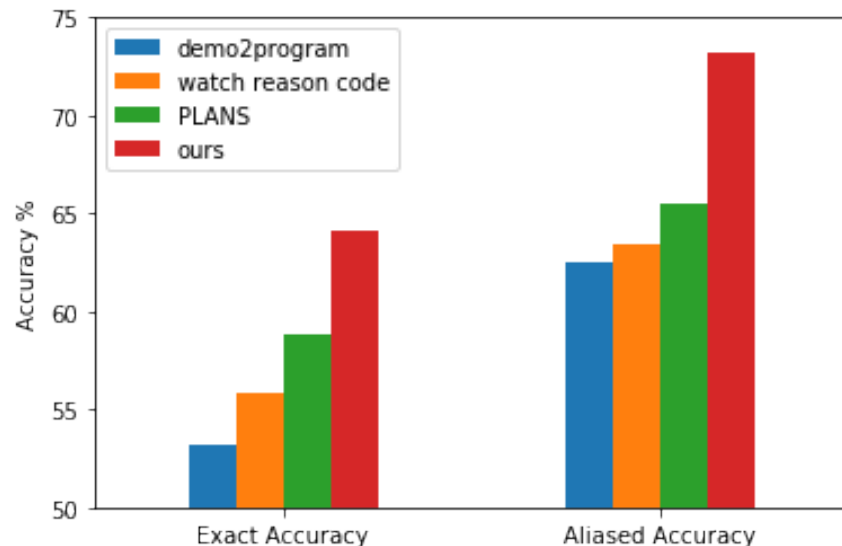


Figure 5.6: A plot demonstrating that our model is able to achieve an Exact Accuracy comparable to previous approaches Aliased Accuracy, while significantly out performing on the latter.

5.4.3 Noise Ablation Study

While our model is clearly capable of inductive reasoning over multiple demonstrations, we consider the implication of noise with respect to its ability to accurately generate programs. Previous approaches that utilise rule-based solvers [Dang-Nhu, 2020] have been highly sensitive to noise. In-fact, even with high levels of perception accuracy the PLANS model required dynamic filtering. This indicates that even the slightest amount noise in the input causes the PLANS model to underperform.

To test our model’s ability to deal with noise in its input we devise a noise ablation study. By separately predicting the actions and perceptions with two distinct networks we can vary the accuracy of the perception predictions independently of the actions. Our semantic encoder in this setup has independent action and perception encoders. Our action encoder easily obtains an accuracy of 98% with a simple five layer convolutional neural network (as described in section 5.3.2). We train a perception encoder with the same architecture which achieves an accuracy of 79.9%. We then utilise a pre-trained object detection network (efficientnet [Tan and Le, 2019]) to improve this result. This time our prediction encoder achieves an accuracy of 90.2%. Our approach to utilise predicted perception primitives and actions to generate a visual language allows us to evaluate a ‘perfect case’ scenario. As we have access to ground-truth labels for these perception primitives and actions, we can use these directly to generate our visual language tokens. Doing so artificially creates a scenario in which our model has effectively no input noise (excluding noise from mislabelled data).

Our results from this study are presented in table 5.2. These results clearly show our models capacity to not only reason over multiple demonstrations, but to

Perception Noise	Exact	Alias
0%	66.0	75.1
10%	64.1	73.2
20%	62.7	71.6

Table 5.2: Exact and Alias program accuracy’s for varying levels of perception noise.

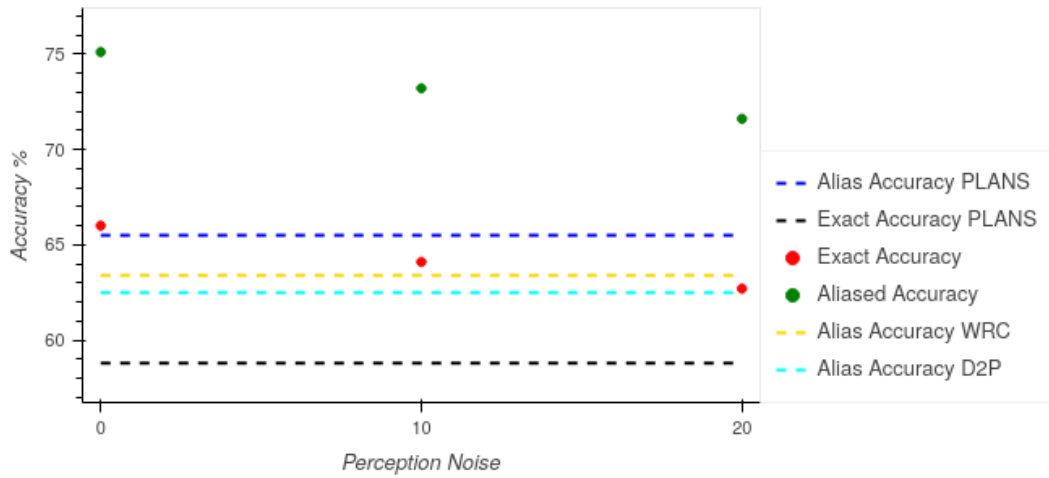


Figure 5.7: A plot showing the impact of perception noise on our method. We compare the results against previously reported accuracy metrics, clearly showing that even with high levels of perception noise (20%) our method is able to outperform prior state-of-art results.

handle contradictory or noisy signals. In contrast to rule-based solvers, which as shown by [Dang-Nhu, 2020] are strongly reliant on pre-defined filtering heuristics, our VT4 model is able to learn its own heuristics. These results also show that our model was able to exceed the previous state-of-the-art while enduring a 20% error rate in perception predictions.

5.5 Conclusions and Future Work

The task of synthesising a program from multiple visual demonstrations involves solving many different tasks. These include learning spatial-temporal relationship

from visually complex inputs and successfully translating these into a logical sequence in a different domain. Previous attempts at solving this problem have relied upon summarised representations of the spatial-temporal relationships and have been highly sensitive to input noise. In this paper we propose a video-to-text transfer transformer network that is able to perform multi-sequence-to-sequence translation without requiring summarised spatial-temporal embeddings. Our method is also highly robust to input noise, which is a problem that caused significant challenges for previous methods. On top of this, we achieve the largest increase in performance to date on this task.

Future work in this area will aim to investigate aspects of our approach that could be improved. These include researching methods for streamlining the visual language model, including end-to-end techniques. Additionally, another area for potential research would be to expand this work into areas and datasets that do not contain curated labels. Examples of this may include reinforcement learning agents trained in visually complex domains or driverless cars.

Bibliography

- [Ahmed et al., 2017] Ahmed, K., Keskar, N. S., and Socher, R. (2017). Weighted transformer network for machine translation. *CoRR*, abs/1711.02132.
- [Alur et al., 2013] Alur, R., Bodik, R., Juniwal, G., Martin, M. M. K., Raghothaman, M., Seshia, S. A., Singh, R., Solar-Lezama, A., Torlak, E., and Udupa, A. (2013). Syntax-guided synthesis. In *2013 Formal Methods in Computer-Aided Design*, pages 1–8.
- [Balog et al., 2016] Balog, M., Gaunt, A., Brockschmidt, M., Nowozin, S., and Tarlow, D. (2016). Deepcoder: Learning to write programs.
- [Bunel et al., 2018] Bunel, R., Hausknecht, M. J., Devlin, J., Singh, R., and Kohli, P. (2018). Leveraging grammar and reinforcement learning for neural program synthesis. *CoRR*, abs/1805.04276.
- [Burda et al., 2018a] Burda, Y., Edwards, H., Pathak, D., Storkey, A. J., Darrell, T., and Efros, A. A. (2018a). Large-scale study of curiosity-driven learning. *CoRR*, abs/1808.04355.
- [Burda et al., 2018b] Burda, Y., Edwards, H., Storkey, A. J., and Klimov, O. (2018b). Exploration by random network distillation. *CoRR*, abs/1810.12894.
- [Dang-Nhu, 2020] Dang-Nhu, R. (2020). Plans: Robust program learning from neurally inferred specifications. *ArXiv*, abs/2006.03312.

- [Devlin et al., 2018] Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.
- [Devlin et al., 2017] Devlin, J., Uesato, J., Bhupatiraju, S., Singh, R., Mohamed, A.-r., and Kohli, P. (2017). Robustfill: Neural program learning under noisy i/o. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 990–998. JMLR.org.
- [Dong et al., 2019] Dong, J., Gao, K., Chen, X., Guo, J., Cao, J., and Zhang, Y. (2019). Not all words are equal: Video-specific information loss for video captioning. *CoRR*, abs/1901.00097.
- [Dosovitskiy et al., 2020] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale.
- [Duan et al., 2019] Duan, X., Wu, Q., Gan, C., Zhang, Y., Huang, W., van den Hengel, A., and Zhu, W. (2019). Watch, reason and code: Learning to represent videos using program. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, page 1543–1551, New York, NY, USA. Association for Computing Machinery.
- [Graves et al., 2014] Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing machines. *CoRR*, abs/1410.5401.
- [Gulwani, 2011] Gulwani, S. (2011). Automating string processing in spreadsheets using input-output examples. *SIGPLAN Not.*, 46(1):317–330.

- [Jha et al., 2010] Jha, S., Gulwani, S., Seshia, S. A., and Tiwari, A. (2010). Oracle-guided component-based program synthesis. *ICSE '10*, page 215–224, New York, NY, USA. Association for Computing Machinery.
- [Kaiser and Sutskever, 2016] Kaiser, L. and Sutskever, I. (2016). Neural gpu learn algorithms. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.
- [Kay et al., 2017] Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., and Zisserman, A. (2017). The kinetics human action video dataset. *CoRR*, abs/1705.06950.
- [Kempka et al., 2016] Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaskowski, W. (2016). Vizdoom: A doom-based AI research platform for visual reinforcement learning. *CoRR*, abs/1605.02097.
- [Kurach et al., 2016] Kurach, K., Andrychowicz, M., and Sutskever, I. (2016). Neural random-access machines. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.
- [Lin et al., 2019] Lin, J., Gan, C., and Han, S. (2019). Tsm: Temporal shift module for efficient video understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.
- [Ling et al., 2017] Ling, W., Yogatama, D., Dyer, C., and Blunsom, P. (2017). Program induction by rationale generation: Learning to solve and explain algebraic word problems. *CoRR*, abs/1705.04146.

- [Parisotto et al., 2016] Parisotto, E., Mohamed, A., Singh, R., Li, L., Zhou, D., and Kohli, P. (2016). Neuro-symbolic program synthesis. *CoRR*, abs/1611.01855.
- [Raffel et al., 2019] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.
- [Simmons-Edler et al., 2018] Simmons-Edler, R., Miltner, A., and Seung, H. S. (2018). Program synthesis through reinforcement learning guided tree search. *CoRR*, abs/1806.02932.
- [Sun et al., 2019a] Sun, C., Myers, A., Vondrick, C., Murphy, K., and Schmid, C. (2019a). Videobert: A joint model for video and language representation learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7463–7472.
- [Sun et al., 2019b] Sun, C., Myers, A., Vondrick, C., Murphy, K., and Schmid, C. (2019b). Videobert: A joint model for video and language representation learning. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7463–7472.
- [Sun et al., 2018] Sun, S.-H., Noh, H., Somasundaram, S., and Lim, J. (2018). Neural program synthesis from diverse demonstration videos. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4790–4799. PMLR.
- [Tan and Le, 2019] Tan, M. and Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR.

- [Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u., and Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17*, page 6000–6010, Red Hook, NY, USA. Curran Associates Inc.
- [Waldinger and Lee, 1969] Waldinger, R. J. and Lee, R. C. T. (1969). Prow: A step toward automatic program writing. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence, IJCAI'69*, page 241–252, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- [Wang et al., 2019] Wang, Q., Li, B., Xiao, T., Zhu, J., Li, C., Wong, D. F., and Chao, L. S. (2019). Learning deep transformer models for machine translation. *CoRR*, abs/1906.01787.
- [Wu et al., 2019] Wu, C.-Y., Feichtenhofer, C., Fan, H., He, K., Krahenbuhl, P., and Girshick, R. (2019). Long-term feature banks for detailed video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Conclusion and Future Directions

In this thesis, we investigated the explainability of deep reinforcement learning and other autonomous agents via rule extraction, and the impact that self-attention mechanisms have on a policy with respect to learning rules in a visually complex environment. In this chapter, we will summarise the key contributions of our research and discuss the future direction for this research.

6.1 Summary of the Contributions

In Chapter 3, We describe the benefit of incorporating spatio-temporal self-attention into the underlying neural network architecture of a deep reinforcement learning agent. We evaluate our architecture in various forms across a visually complex suite of environments, while directly comparing our approach to the classic architecture first proposed by [Mnih et al., 2015]. Our results clearly indicate that the addition of attention to the network is beneficial, and lead to significant improvements in sample efficiency across 60% of tested environments. Of note is the performance in the environment Demon Attack, where the addition of self-attention resulted in state-of-the-art results, far exceeding the previous reported

benchmark.

In chapter 4, we proposed a mathematical definition for a rule that transcends the semantics of natural language labels which are often employed to describe rules. We evaluate our definition by evaluating it against deep reinforcement learning agents across multiple visually complex environments, something previously not possible with prior definition and methods. Additionally, we test and show that deep reinforcement learning agents that incorporate attention mechanisms into their underlying network architecture can learn more defined rule sets.

in Chapter 5 we describe the challenge of synthesising a program from observations of an autonomous agent. We highlight the shortcomings of prior works which focus heavily on summarising state information or are alternatively highly sensitive to noise in their self-obtained specifications. We successfully overcome both issues with our novel multi-sequence-to-sequence approach that learns to summarise and translate simultaneously. Our proposed model is also highly robust to noisy inputs. Not only does it achieve significant improvements over prior state-of-the-art results, but our ablations study was also able to show that it can do so with high levels of noise in its perception predictions.

Overall, this thesis focuses on the explainability of deep reinforcement learning and other autonomous agents. To achieve this, the work presented focuses on revealing the importance of the underlying architecture of a deep reinforcement learning model while proposing novel designs capable of outperforming the traditional approaches. We then propose a novel and mathematical inspired definition for rule extraction and investigate the impact that spatio-temporal self-attention mechanisms have on a policy's ability to learn rules in visually complex environments. From there, we extend our focus to extracting entire executable programs from a diverse range of visual observation produced by autonomous

agents. Wherein one must not just extract a single rule, but instead an entire set of coherent and logical rules that are highly interpretable. As a result of this work, we believe this thesis delivers numerous novel ideas and insights into the fields of explainable reinforcement learning, rule extraction, and program synthesis alike.

6.2 Limitations and Future Directions

Although this thesis has made several considerable contributions to the fields of explainable reinforcement learning via deep convolutional neural network rule extraction, and program synthesis, we acknowledge that there are still open problems in these fields, and future directions of research as a result of our contributions.

6.2.1 Explainable Reinforcement Learning

In Chapter 3, we introduce a novel spatio-temporal self-attention architecture for deep reinforcement learning agents. In Chapter 4, we defined and extracted learnt rules from trained policies that correlate state and action trajectories. While it is the first work of its kind (as far as the author is aware) to map visually complex, temporally, and unstructured input to a policy into rules dependent upon its actions, one issue with our approach is that it does not provide an easily interpretable output (such as a natural language expression). Our method only allows for the existence of rules to be discovered, not for them to be explained in a manner that would make sense to a common person. While we believe that our contributions are a big step in the right direction for explainable reinforcement learning, future work should investigate how to explain rules once they have been discovered. We propose that future work could investigate the correlated state action trajectories while possibly incorporating natural language models that are

able to link detections to actions during these periods.

Secondly, although we demonstrate and highlight the importance of neural network architecture in both Chapter 3 and 4, our approach to rule extraction is pedagogical in nature. While we can verify through our experiments (Chapter 3) and rule extraction model (Chapter 4) that our additions of spatio-temporal self-attention to a policy’s neural network are beneficial, a decompositional or eclectic based approach to rule extraction or rule explanation may shed more light onto why these additions are so beneficial. For example, future work may focus on investigating the validity intervals of activations between state-action trajectories that are classified as rules, and those that are not. This may shed light on why many agents seem to fail after a certain point in a game, even though there appears to be no real shift in the dynamics, enemies, or objectives of the agent.

6.2.2 Program Synthesis

In Chapter 5, we explore the problem of program generation given a diverse range of video demonstrations. This is a unique and difficult task as it centers around the intersection of multiple domains including explainability, rule extraction, and program synthesis. Our proposed multi-sequence-to-sequence approach was able to achieve significant performance increases over prior works however, our model does have some limitations. One is the intermediary generation of a visual language that creates semantic represents of the state-action pairs generated by the agent and observed in the video demonstrations. Due to the way we generate this language, we are unable to train our model completely end-to-end. Although this does not cause any problems given the relatively constrained visual perception space of the test environment (VizDoom), our method may have limits on its scalability in more complex domains. To address this limitation future work

would involve developing an end-to-end model where the latent space features of the perception model are used in place of the simplified visual language employed in our approach.

6.2.3 Explainable Reinforcement Learning Via Program Synthesis

A potential area forward for our research would be in the unification of our contributions into a framework that can observe a deep reinforcement learning agent (operating in a visually complex environment) and subsequently output a completely interpretable natural language program. The program itself should be derived from the correlations between state-action trajectories, and as such it could even identify when it does not know what to do. The potential for this line of research has come about as a direct result of the contributions made within this thesis. As far as the author is aware, there is currently no one working on this problem, and for that matter, limited research on the extraction of rules from agents that operate in visually complex domains.

6.2.4 Final Remarks

Despite the limitations and potential areas of future research discussed above, this thesis has provided significant contributions to the fields of Explainable Reinforcement Learning and Program Synthesis. The empirical results achieved by our proposed models in Chapters 3 and 5 were able to achieve state-of-the-art results at the time of their publication. While our contributions in Chapter 4 not only shed new light on the impact that spatio-temporal self-attention mechanism have on a policy's ability to interpret its input, but also provide a definition for

rules learnt by deep reinforcement learning agents. To the best of the authors knowledge, this problem has so far been overlooked, and the proposed definition has the potential to open the field to further research.

Bibliography

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.