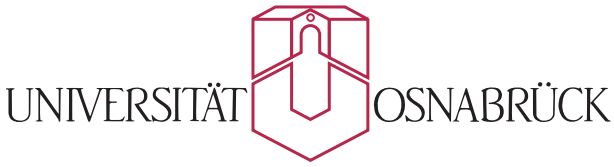


Curiosity in Deep Reinforcement Learning: Investigating applicability of recent formalizations via replication and comparison on a two-dimensional, sparse reward navigation task

by

Lukas Schüler

A thesis submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science
Department of Cognitive Science
University of Osnabrück

First Supervisor: Prof. Dr. Peter König
Second Supervisor: Viviane Clay, Ph.D.
Submitted: April 16, 2022

Eidesstattliche Erklärung

Hiermit erkläre ich, Lukas Schüler, die vorliegende Arbeit *Curiosity in Deep Reinforcement Learning: Investigating applicability of recent formalizations via replication and comparison on a two-dimensional, sparse reward navigation task* selbständig verfasst zu haben und keine anderen Quellen oder Hilfsmittel als die angegebenen verwendet zu haben.

Osnabrück, den 16.4.2022

Lukas Schüler, 975535

Affirmation Statement

I, Lukas Schüler, hereby certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

Osnabrück, April 16, 2022

Lukas Schüler, 975535

Abstract

Over the course of this bachelor's thesis, I test the three deep reinforcement learning algorithms presented in the papers *Random Network Distillation*, *Self-Supervised Exploration via Disagreement* and *Curiosity-driven exploration by self-supervised prediction* on a two-dimensional grid-world environment. Each of these algorithms uses a particular kind of intrinsic motivation to facilitate exploration, improve problem-solving and to deal with sparse reward environments. I investigate their performance as well as their susceptibility to stochastic dynamics under differing noise scenarios, established through customization of the gird-world environment. While *RND* and *Disagreement* demonstrated good performance and robustness against stochasticity, *Curiosity* failed to accomplish solving the task under the given conditions. While a good overview of the potential of these approaches could be given, to cover the complexity of the algorithms, further research needs to be done.

Acknowledgements

First and foremost I would like to thank Dirk Schüler for hooking me up with the hardware needed, giving me the opportunity to do this project with a degree of freedom I would otherwise never have been able to enjoy. I would also like to thank my friends Tom and Stefan for the fruitful discussions and the patience they showed when I again started talking about this work.

Last but not least, I would like to thank my supervisors Dr. Viviane Clay and Prof. Dr. Peter König. I wouldn't have thought it possible to learn so much over the course of one thesis.

Contents

List of Figures

List of Tables

1 Introduction

Where is the action? Nowadays, if somebody wants to engage with topics like intelligence or cognition, it is fairly difficult to get around engaging with the literature about embodied cognition. The classical view, that the mind and its implications are merely a result of neural computation, has in big parts been superseded by the stance of cognition being not only about computational processes in the brain, but also about a body and its interaction with and within the world surrounding it.

A good example for this can be seen in [Cho and So, 2018]. In the paper, the researchers investigate the relationship between arithmetic, mental imagery and hand gesture constituted through the usage of a counting frame, or abacus.

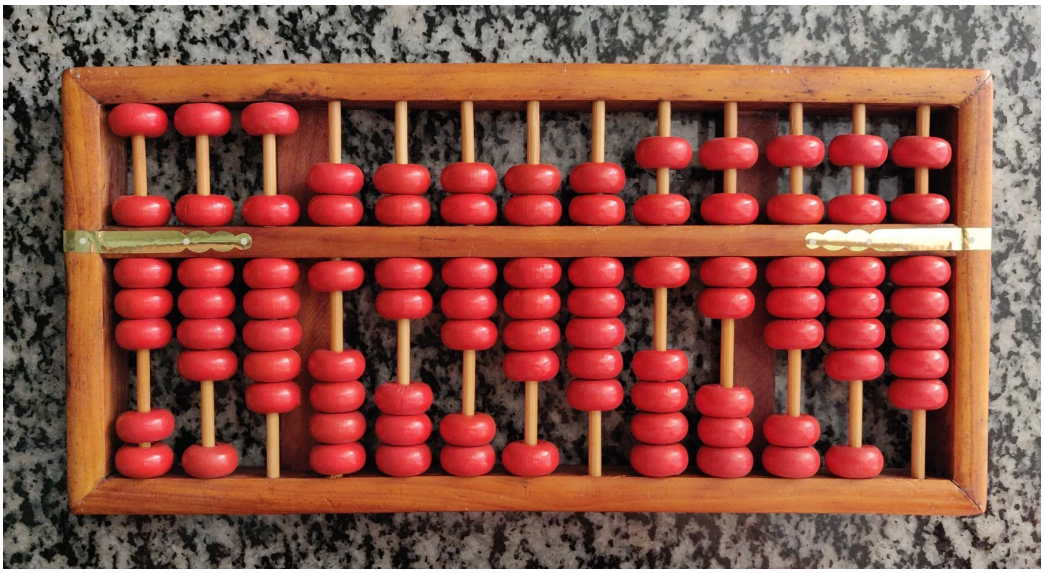


Figure 1.1: Suanpan, a traditional Chinese abacus

When children begin to learn to solve arithmetic tasks using an abacus, they manipulate a physical instance of this very object using thumb, index and middle finger. Over time, as the children's capabilities increase, they replace the counting frame with an illustration of it, while in the advanced stages of the learning process, this illustration is going to be left away as well. What remains for the stu-

dents are the in part subconsciously executed rapid hand gestures when performing, for example, multiplication of five-digit numbers in their heads. [Cho and So, 2018] suggest that these achievements can be attributed to the complement of the visual-spatial representation of a mental abacus by the motor-spatial representation induced through finger movement. When the researchers restricted the students from performing the hand gestures, performance dropped significantly.

The literature is extensive and full of examples reaching from phantom limbs [Ramachandran and Rogers-Ramachandran, 2000] and prosthetics [Murray, 2004] to meditation [Wallace, 1975] or lucidly dreaming basketball players [Erlacher et al., 2012]. Embodiment gained a lot of reception over the last decades, especially in the field of cognitive science [Engel et al., 2013], and it seems like embodied cognition arrived in all the disciplines encompassing intelligence.

However, most of artificial intelligence is solely concerned with representation learning and the formalization of world models, like for example in computer vision or natural language processing. These subfields most of the time don't include any kind of closed-loop system relying on environment interaction, like the presented literature would suggest. But taking humans as the prototype for creating a general artificial intelligence, taking embodiment into account seems unavoidable.

Luckily, [Sutton and Barto, 1998] introduced a new paradigm to the world of machine learning. This idea aims to combine the interaction of an agent in an environment with representational learning and is called Reinforcement Learning, which will be the fundament on which this thesis is going to build upon.

Reinforcement Learning For most people, the basic idea behind Reinforcement Learning, or RL, seems rather intuitive. A popular example is an infant playing with some toys on its own, without receiving any guidance. To describe it in the words of a RL-researcher: The agent, here the infant, interacts/plays with the environment at state s, constituted through its toys, yielding a reward signal like for example praise form its parents. Upon receiving a positive reward the child is inclined to repeat, to *reinforce* the behaviour, upon receiving a negative feedback, e.g. pain, the behaviour will probably be omitted in the future.

This behavioural paradigm gets formalized in the form of Markov-Decision-

Processes, or MDPs. An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the set of states, $\mathcal{A}$ the set of actions, P the transition function and where R is the function describing the reward. Lastly, γ is the discounting factor that regulates how far back in the past actions taken contributed to the reward achieved. As important the discount factor γ and the transition function P are, the focus of this thesis is going to concentrate on reward functions.

Figure 1.2 describes well a Markov Decision Process in its simplest form. At time point t the agent is situated in state S_t and undertakes action A_t. A state S_t is a complete description of the state of the world, containing every information about the world at point t. Is the agent able to observe the complete state of the environment, we say that the environment is fully observed. When the agent can only perceive a subpart of the state, we say that the environment is partially observed. Upon the agents action, the environment generates a reward $r = R(a|s)$ for the agent and the agent reaches a consequent state.

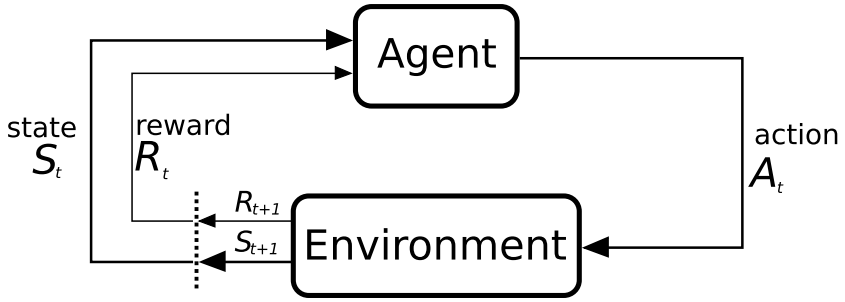


Figure 1.2: MDPs constitute a closed loop interaction system

Here the terms policy, value function and, optionally, model come to play. The policy π decides which action is taken. It defines the behaviour of the agent. And since the agent is essentially defined through its behaviour, it is not uncommon to substitute the word *policy* for *agent*.

The value function estimates the value of an input state or state-action pair, which are represented through MDPs. It does that by on the one hand taking into account proximal rewards as well as on the other hand expected rewards of possible consecutive states along a so-called trajectory or, put simply, a sequence of states.

The model describes the environments underlying dynamics, i.e. the transi-

tion probabilities between the states. If this distribution is known beforehand, the approach is referred to as model-based, the opposite being model-free.

While this extent of an overview will suffice for the scope of the work at hand, it remains only a shallow introduction to the vast field of RL and its characteristics. For a more extensive and technical introduction the curious reader is highly encouraged to engage herself with the great book by [Sutton and Barto, 1998]. Particularly chapters one to six help building a solid foundation to work on.

Deep Reinforcement Learning In the preceding section one specific subject has been left out deliberately. Namely, the manner in which the value function, responsible for estimating the goodness of a given state, takes into account possible future outcomes. This and the way how these value estimates need to be stored constitutes a key limiting factor of RL.

For environments with small, finite action and state spaces this was calculable as well as tractable using a tabular representation, simply storing a value estimate at (s_i, a_j), with i and j being indices. But when confronting large, potentially infinite action and state spaces these tables become incomputable due to the size of elements to store and the complexity of the value estimation's recursive nature. However, with the ascent of deep neural networks, or DNNs, and in particular convolutional neural networks [Krizhevsky et al., 2012], this issue could be resolved and the combination of DNNs and RL resulted in a new subfield called deep reinforcement learning, or DRL. Now, former discrete policies and value functions are replaced with deep neural networks. In the case of value estimation, these estimates are now learned by a DNN. And instead of discrete actions, another DNN returns mean and standard deviation of a normal distribution, from which then continuous actions could be sampled.

This enables DRL to work with continuous or high dimensional spaces, like images as observations or continuous movement of e.g. body parts as actions. This, of course, brings some new issues to the surface, like sample inefficiency and instability, but the algorithms investigated in this work aim at two interrelated and more pervasive challenges native to RL, and inherited to DRL.

Sparse Rewards In many environments reward can be rare, absent or, due to complexity of the environment, intractable to formulate into a clear reward function. Yet the reward signal is fundamental, since without it the agent doesn't know in which direction to update its policy. So the agent continues to take random actions with the current set of parameters until, by chance, he obtains some non-zero reward. Depending on how long it takes to get these rewards, it might take the agent extremely long to learn anything or even impede learning altogether.

Exploration and Exploitation Should the agent decide for an action that seems optimal, making the assumption that the knowledge at hand is sufficiently reliable? Or should the agent act potentially suboptimal, assuming that the experiences made might not be sound and that collecting new information could help improve on that? In simple terms, should the agent explore or exploit? The difficulty of weighing these behavioural extremes is known as the exploration-exploitation-dilemma.

Further, while exploitation is simple, from all potential rewards you choose the highest, efficient exploration is highly context dependent and a research topic on its own.

Intrinsic Motivation Combining the problems stated in the last two paragraphs, results in exploitation becoming difficult due to the scarcity of rewards as well as in the agent still inefficiently exploring the environment for a reward signal. To circumvent this [Schmidhuber, 1991] proposed the idea of an intrinsically motivated agent.

According to [Ryan and Deci, 2000] intrinsic motivation is described as being moved to do something because it is inherently enjoyable, which stands in stark contrast to being moved to do something as a step toward solving a specific problem of clear practical value.

Considering our just stated problem, it would be beneficent if our agent would not only receive reward for accomplishing a goal, but also for simply engaging with the task itself. Preferable in a way of heading for unknown things or states whose outcome it cannot predict. This kind of intrinsic motivation, circumventing sparse rewards and improving exploration, could be described as curiosity and its

technical realization constitutes the core of this thesis.

The most protruding results, representing the state of the art, were achieved by utilizingan error in prediction.

Next State Prediction To accomplish the agent striving towards states that it cannot grasp or that are not known to it, to let him act curious, researchers designed a new reward function. The novelty of a given state gets determined by letting a deep neural network try to predict this state, given the preceding state and action taken, and calculating the error between the true state and the made prediction. Since the network is trained on this error and should over time be more and more potent to predict already seen states, this discrepancy can be taken as a measure of surprise or unknownness and to function as an intrinsic, continuous reward signal. As a consequence of the agent's relationship to the reward signal, he should get incentivized to behave in a novelty approaching manner. This intrinsic reward could be added to the extrinsic reward signal or could be used on its own.

This approach is effective and simple to implement and would, at least in theory, resolve the problems around sparse rewards and exploration. Yet, it harbours a simple breach that DRL algorithms, with the aim to maximize final reward, make full use of when the opportunity is present.

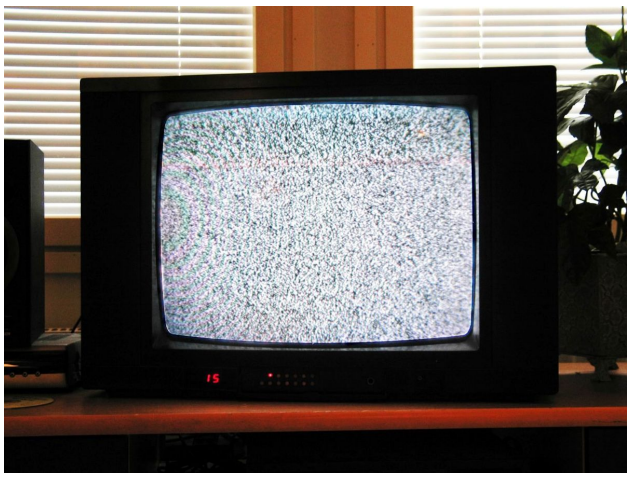


Figure 1.3: TV displaying white noise

Stochasticity Next state prediction is based on how difficult it is for the agent to predict the next state, and for simple, well predictable environments this works out well. But things that are either very complex or simply stochastic in their nature radiate a for the agent unescapable attractiveness. Be it rolling a die, observing leaves moving in the wind or the flickering of a fire. Situations like this are frequent in most real world environments, and therefore this phenomenon clearly poses a problem. The go-to example for the case of stochasticity is the white noise displayed on old TVs introduced in [Schmidhuber, 1991]. For the agent observing a television screen displaying white noise, every frame would be unpredictable resulting in the agent to never stop watching the screen. All this constitutes the infamous 'noisy-TV' problem.

The Objective The problem presented over the last pages is essential to the progress of the field and the prospect of the advances successful research would bring about, actuated researchers to design interesting methods in order to cope with the issue at hand. In the chapters following this Introduction I am going to investigate three architectures, each of which aims to solve the challenge of stochasticity, inherent to state prediction methods.

The papers to be presented illustrate impressive results, but often methods need to get massively fine-tuned for the task at hand to accomplish this degree of results. Furthermore, deep RL has the reputation of exacerbating reproducibility due to many tuneable variables with difficult to grasp impacts. For reasons of validation and fair comparison, I am going to align the architectures' configurations and investigate performance as well as generalizability through application on a common, unknown environment.

This thesis aims to answer two questions, scrutinizing the main result of each paper:

I. Does the intrinsic reward used, facilitate problem-solving on a different environment?

II. Are the algorithms still susceptible to stochasticity?

2 Methodology

This chapter documents the conception and structure of the experimental part of this work. It incorporates a detailed explanation of the algorithms under investigation, describes the testing environments and their setup and further, characterizes the experiments' designs necessary to answer the questions raised in the Introduction.

2.1 The algorithms

In this section the three algorithms in question are going to be introduced. Since they achieved remarkable results and since each of them has been published relatively recently, they are representative for the latest progress in the field of DRL considering intrinsic motivation through state prediction.

Since each of the algorithms is build upon the same foundation, it is sensible to start this section with a short introduction to this very foundation.

2.1.1 Proximal Policy Optimization

Proximal Policy Optimization, or *PPO*, is an algorithm introduced by John Schulman and colleagues in [Schulman et al., 2017] and represents the state of the art in DRL. It utilizes the actor-critic paradigm, which means that it is build from one neural network representing the policy, the actor, and from one neural network performing the value estimation, the critic. It is model-free so it has no information about the dynamics underlying the environment. *PPO* trains a stochastic policy in an on-policy way. This means that it explores by sampling actions according to the latest version of its policy, which is updated frequently.

The key idea underlying policy gradient methods, to which *PPO* belongs, is to increase the probabilities of actions that lead to higher return, and to decrease the probabilities of actions that lead to lower return, until you arrive at the optimal policy. Yet, policy gradient methods are prone to large gradients for updating the

policy, making training instable. The main idea of *PPO* is to implement a loss function limiting the size of the gradient update, so that the policy does not move too far when being updated.

While this overview will suffice for the work at hand, the ideas beneath *PPO* remain exciting and the interested reader is encouraged to engage with [Schulman et al., 2017].

2.1.2 Curiosity-driven Exploration

The first algorithm to investigate in this thesis was developed by Deepak Pathak and colleagues and was presented in [Pathak et al., 2017]. The paper is by many referred to as the *Curiosity* paper and for reasons of readability this algorithm will be, in this work, referred to as *Curiosity*.

Curiosity creates an intrinsic reward by letting a neural network try to predict the next observation, given the current observation and action taken. This is called a forward dynamics model and follows the next-state prediction paradigm stated in the Introduction. It is important to mention that not only the current state but also the following state are available to the network, because the intrinsic reward gets calculated retrospectively, after the trajectory has already happened.

Both observations are embedded inside a feature space through feature extractors, which are also neural networks. For a clearer understanding of their core idea the authors divide all possible events into three categories:

1. Things that can be controlled by the agent
2. Things that the agent cannot control but that can affect the agent
3. Things out of the agent's control and not affecting the agent

An example for category two would be a vehicle driven by another agent, and for category three the moving of leaves.

To circumvent the 'noisy-TV' problem the algorithms feature extractors are incentivized to only integrate information in their embedding about things contained in category one and two. This is accomplished by taking the embedded current and next state and then make another neural network predict the action

taken between these given states. The error of this prediction is used as a loss to train the feature extractors. The network doing this inverse prediction task is called the inverse dynamics model, and the interplay between it and the forward dynamics model is the core idea to make *Curiosity* robust against stochasticity.

But this approach might face an important issue. In [Burda et al., 2018a] researchers, including Deepak Pathak himself, investigated the performance of this algorithm on a multitude of environments and state to have found that *Curiosity* does fall for stochasticity, but only if this stochasticity is the product of the agent's action, thus belonging to category one of the distinction the researches made and thus recreating the 'noisy-TV' problem.

To shortly summarize *Curiosities* core idea: Create an intrinsic signal through next-state prediction and utilize feature extractors, trained on the loss of an inverse prediction, to exclude things not affecting the agent, from the agent's perception. Potentially, this still might lead to the 'noisy-TV' problem, when the stochasticity depends on the agents action.

2.1.3 Exploration via Disagreement

The second algorithm introduced in this work was co-developed by Deepak Pathak and presented in [Pathak et al., 2019]. It can be seen as the response to the problem stated in [Burda et al., 2018a] regarding *Curiosity's* observed pitfall for action-dependent stochasticity.

Like its predecessor it makes use of the error in next-state prediction to create an intrinsic reward signal. But to avoid susceptibility to stochastic dynamics, be it action-dependent or -independent, the algorithm does not employ an inverse model anymore. Instead, multiple forward dynamics models are applied to the next-state prediction problem and the variance calculated over their predictions is used as the intrinsic signal, while each prediction error is utilized as the loss to train the respective forward dynamics model. Like in *Curiosity* the current state as well as the subsequent state are available to the network, because the intrinsic reward gets calculated retrospectively.

The authors state that given enough stochastic input, the forward dynamic models will all converge to the mean of the distribution underlying the noise and

will thus decrease the strength of the intrinsic reward signal. The algorithm employs a randomly initialized feature extractor for performance reasons.

Besides the claims that the algorithms investigated in this thesis have in common, like enhanced performance and robustness to stochasticity, the authors of this paper also claim to have made the training process differentiable, and thus way more efficient. Unfortunately this feature is not contained in the code published and can thus not be under investigation in this work.

To conclude, the variance over predictions of multiple trained next-state predictors is used as an intrinsic reward signal. When confronted with stochastic dynamics, the predictors shall converge and the intrinsic reward will vanish. This disagreement between the forward models is the algorithms main characteristic, and it will thus be referred to it as *Disagreement*.

2.1.4 Random Network Distillation

The third and last algorithm was published in [Burda et al., 2018b] by Yuri Burda, Harri Edwards and their colleagues and is called Random Network Distillation.

Random Network Distillation, short *RND*, utilizes next-state prediction differently compared to the former two algorithms. Admittedly, *RND* uses a feature extractor to embed the next state, but it does not predict the embedded next state given the embedded current state and action, like *Curiosity* and *Disagreement* do. In contrast, *RND* generates the intrinsic reward by predicting the embedding of the next state given the non-embedded next state. Like in the preceding algorithms, this reward is calculated retrospectively. The feature extractor responsible for creating the embedding is simply a randomly initialized neural network with fixed parameters, i.e. it is not trained. The prediction error is again taken as the loss for the predictor network so that the predictor network learns to mimic the random, fixed feature extractor and thus over time ”distills” into the form of this randomly initialized feature extractor.

The authors state two reasons for why this algorithm is not susceptible to stochasticity. The first is, that the predictor network does no forward prediction, more an embedding prediction. Like said above, *RND* is not exactly utilizing next-state prediction. The second reason stated is, since the feature extractor is a

fixed network stochastic input becomes deterministic and is thus palpable for the predictor.

In a nutshell, *RND* employs a fixed, random network as a feature extractor and predicts a state embedding given the state itself, in contrast to making a prediction of the subsequent state.

2.2 Setting up

This section describes the technical challenges necessary to surpass to utilize the algorithms in the experimental setting of this work. A description of the hardware and software used can be found in Appendix II.

2.2.1 Software

This work was only able to comprise three algorithms because the developers of *Curiosity*[1], *Disagreement*[2] and *RND*[3] have published their code on GitHub[4]. Nonetheless, getting the code to properly function for this project was not trivial. Particularly dependency resolving and code adaption consumed a lot of time. The code used in this work, including the package specifications needed, can be found in the repository [FOOTNOTE] for this thesis.

Dependencies The only repository that provided a file declaring the necessary packages and their versions was the one for *Curiosity*. For the other two repositories a lot of research and trial-and-error were necessary to establish a properly functioning environment. Unfortunately the requirements given by the *Curiosity* repository, were not of too much help, due to issues with an archived repository[5], which is one of the key dependencies of the architecture. Because of that and the fact that the testing environment chosen for this thesis was, due to hard-coded network dimensions in the *Curiosity* code, challenging to integrate, I decided to

[1]Curiosity, https://github.com/pathak22/noreward-rl
[2]Disagreement, https://github.com/pathak22/exploration-by-disagreement
[3]RND, https://github.com/openai/random-network-distillation
[4]Github, https://github.com/
[5]Doom-Environment, https://github.com/openai/doom-py/issues

use an adaption of *Disagreements* code instead of *Curiosity's* original implementation. This adaption is an option supplied by the developers of *Disagreement* and *Curiosity* and has also the advantage that all architectures now use *PPO* (compare subsection 2.1.1) as a fundament, while the original implementation of *Curiosity* used A3C [Mnih et al., 2016]. This improved comparability a lot.

Adaption of variables To ensure comparability the architectures were assimilated as much as possible without distorting the subsystem responsible for the intrinsic reward signal. This assimilation process resulted in the choice of the same hyperparameters, the disabling of frame stacking for *RND* as well as changing *RND's* policy network from a recurrent neural network to a convolutional neural network, like *Curiosity* and *Disagreement* use. This possibility was provided by the developers.

[QUICK SUMMARY OF MOST IMPORTANT PARAMETERS][PAPS TABLE]

2.2.2 Hardware

Since DRL is quite demanding on hardware and a big number of experiments were needed to be conducted in the timeframe of this thesis, an external system was necessary to prevent training demands from interfering with the resource allocation necessary for development, and to assure that in the timeframe at hand a sufficient amount of data was even collectable. Information about the hardware used can be found in Appendix A

2.3 The environment

In this section the environment used to test the algorithms is introduced. Details like its setup, the action space and the reward function are explained here.

2.3.1 Structure

To appropriately answer the questions expressed in the introduction, not every environment could be utilized. Especially due to limited computing power it needed to be lightweight and fast. Further, to efficiently cover multiple experimental conditions it should on the one hand have some variability and on the other hand be customizable, while not being too complex. All this can be found in the Gym-Minigrid[6] environment. Its code is open source and available on GitHub, while it uses the API of OpenAI-Gym[7] to enhance accessibility.

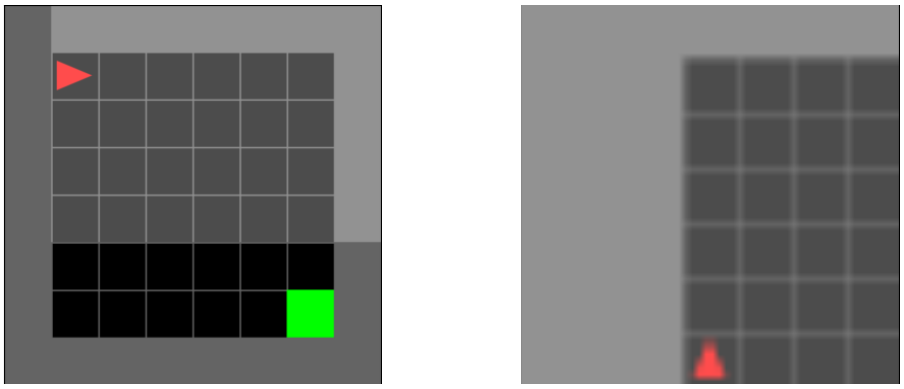
The repository contains variations of the same base environment (compare Figure 2.1) in which an agent needs to navigate in, and sometimes interact with, a $N \times M$ gridworld to reach a goal state, which is represented by a green tile.

The default observation format is not in pixels, but in a custom encoding. Further, a string describing the task to do is contained in the observation. To get rid of the mission string and to convert the view into pixel space, Gym-Minigrid offers the two wrapper-classes *ImgObsWrapper* and *RGBImgPartialObsWrapper* respectively.

Action	Code
Turn left	0
Turn right	1
Move forward	2
Pickup object	3
Drop object	4
Activate object	5

The action space is discrete and is defined by the encoding shown in the table above. For most of the environments the available sizes are (5×5), (6×6), (8×8) or (16×16) tiles. Extrinsic Reward is only obtained when reaching the goal state, what makes the extrinsic reward signal very sparse. An episode is terminated when the goal state is reached or when the maximum number of steps is passed where $steps_{max} = 10 \times envSize$. For the (8×8) scenario the number of allowed

[6] Gym-Minigrid, https://github.com/maximecb/gym-minigrid
[7] OpenAI Gym, https://github.com/openai/gym



(a) Researchers view (b) Agents view, translated into pixels

Figure 2.1: Views on the base environment

steps would then be 640. The reward attained when reaching the goal state is 1, but a penalty dependent on the number of steps taken is subtracted to incentivize learning, while in the worst case still giving a strong enough reward to enable learning. The final reward is thus $reward = 1 - 0.9 \times \frac{steps_{taken}}{steps_{max}}$ or, when surpassing the number of allowed steps, simply 0.

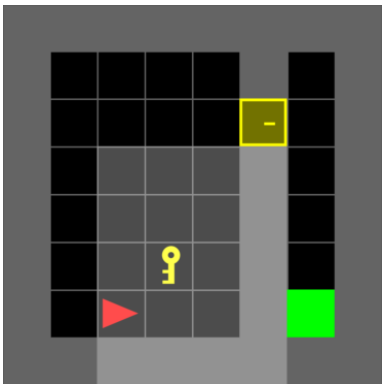


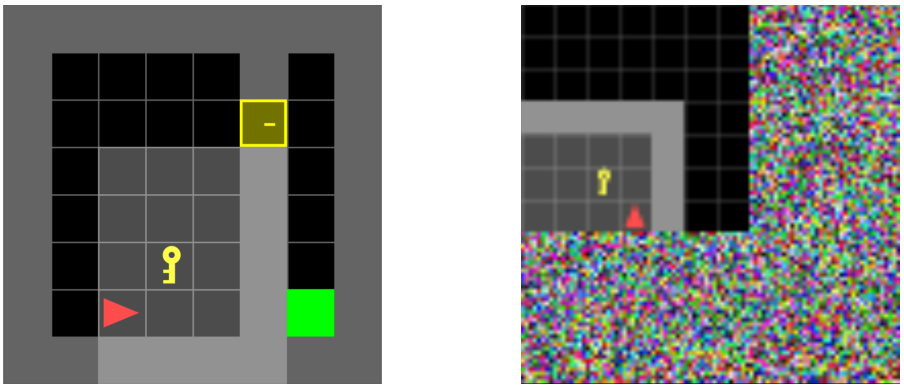
Figure 2.2: *DoorKey* environment

The task To answer the questions stated in the Introduction, I chose the *DoorKey* (compare Figure 2.2) environment. This environment is a good place to test the combination of problem-solving and intrinsic motivation. In this environment the agent needs to find a key, take the key, find a door, unlock as well as open that door and then reach a terminal position to acquire reward. To mediate between enough

complexity and the urge to keep training demands low, I decided to implement the experiments in the 8×8 version. In each trial, the position of the agent and the key on the left side of the room, the position of the vertical wall dividing the room, the position of the door in that wall were randomly assigned. This was done to avoid memorization, having the effect that there does not exist one optimal solution for the environment.

2.3.2 Extending the environment

A key result of each paper is the algorithms' robustness against stochastic dynamics. To investigate this, each algorithm was also applied to three different conditions. Each condition was extended into an action dependent and an action independent scenario. This was done to possibly replicate the results stated in [Burda et al., 2018a] and to examine whether these also apply to *Disagreement* and *RND*.



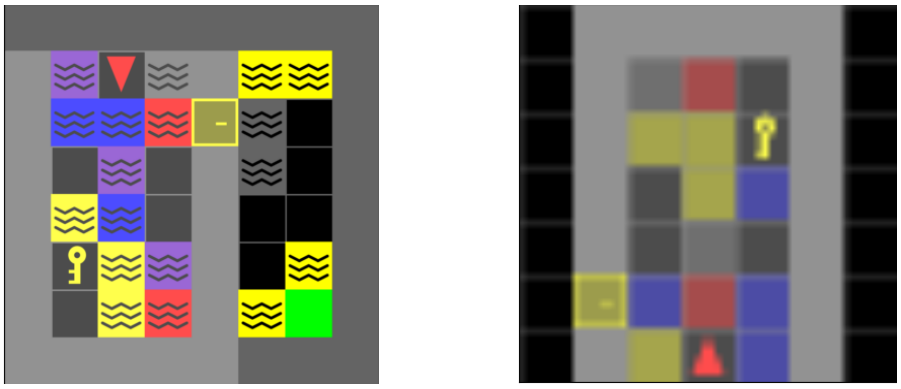
(a) Researchers view (b) Agents view

Figure 2.3: External Noise

External noise This condition together with its implementation were introduced in the *Curiosity* paper. The observation perceived by the agent gets enlarged so that the original view only occupies 60% of the new enlarged observation. The remaining 40% are replaced by noise created using Numpy's[8] random[9] module. The

[8]Numpy Python module, https://numpy.org/doc/stable/index.html
[9]Numpy random, https://numpy.org/doc/stable/reference/random/

original proposal is action-independent, meaning that the noise changes irrespective of the agent taken, after every step. I adjusted the given wrapper to also create an action-dependent scenario of this condition relating to [Burda et al., 2018a]. This means that the noise constituting 40% of the agents view is only changed when the agent takes one specific action. For the experiments undertaken this was the *activate-object* action, because using it is indispensable to solve the environment, while still, in theory, being necessary to be used the fewest and having only minimal effects on the environment.



(a) Researchers view (b) Agents view after change

Figure 2.4: Random Colours

Random Colours The reason to introduce this environment in addition to the external noise condition is to assure generalizability of the stated robustness against stochasticity and to have a scenario developed independently from the algorithms under discussion.

In this extension of the *DoorKey* environment the agents partial view is not directly manipulated but instead the composition of the environment itself. 60% of the black floor tiles were replaced with randomly coloured tiles containing three black zigzag lines. These originate from inheritance of the *Lava*-superclass and were thought to help researchers keeping the tiles apart from the goal state by adding a feature besides color. However, when transferring the observation into pixel space, this pattern is lost in translation (compare Figure 2.4b) to not influence the training process.

For the action-dependent scenario of this environment the floor gets randomly

assigned a new colouring with the same ratio of black to coloured tiles, when the action undertaken was, for the reasons stated in the external noise scenario, the *activate-object* action.

Since continuous change of observed noise is already covered in the action-independent condition of the external noise scenario, the action-independent condition of the Random Colours environment changes the noise by chance. This is done with a probability of 30% for each step taken.

Random Actions The last stochastic condition applied in this thesis addresses the effect of stochasticity in the agents actions themselves.

In the action-dependent condition, if the action activate-object is chosen, an action from the action space is randomly sampled. In the action-independent condition the action is sampled with a probability of 30% for each step taken.

Wording The terminology for the noisy environments and for the two types of stochasticity, action- dependent and -independent, under investigation are closely related and can be difficult to keep apart. To avert any confusion coming up over the course of this thesis and particularly in the presentation of the results, *external noise*, *random actions* and *random colours* will be classified as scenarios. While the terms action-dependent and action-independent will be referred to as conditions.

2.4 The experiments

Since the conditions tested in thesis were already explained thoroughly, the description of the experiments conducted only takes a few words.

2.4.1 Performance

The first question: Does adding intrinsic reward improve performance?

Extrinsic reward In this experiment the performance of each algorithm using only the ordinary extrinsic reward signal, being the equivalent of the performance

of each underlying *PPO*, gets measured. This is done to measure the degree of comparability of the algorithms. Each algorithm gets trained for 2000896 time steps in the *DoorKey* environment, where one time step is equivalent to one action undertaken.

Combining both signals To answer the question to which degree an added intrinsic reward signal facilitates the solving of sparse reward environments, the algorithms are placed within the *DoorKey* environment, using a combination of the intrinsic and extrinsic rewards. Each reward signal constitutes 50% of the total reward and each run is composed of 2000896 time steps.

2.4.2 Stochasticity

The second question: Are the algorithms robust against stochasticity?

The robustness against stochastic environment dynamics is the core statement of each algorithm. To test whether this actually holds each algorithm is confronted with two versions of stochasticity. Stochasticity influenced by the agents action and stochasticity independent of the agent's behaviour. To test this, each algorithm gets trained five times for 2000896 time steps on the external noise, random colours and random actions scenarios respectively. This whole process will be done for the action-dependent as well as the action-independent condition.

3 Results

The aim of this thesis was to investigate the impact intrinsic reward signals, implemented in three different algorithms, had on problem-solving, particularly in varying stochastic environments. In this chapter I am going to present the results obtained in this investigation. The chapter is divided into two sections, the first being intrinsic reward's impact on problem-solving and the second being susceptibility of the presented algorithms to noisy environments. Yet, the main focus of this work lays in the second section.

In both sections every environmental scenario was tested with ten runs per algorithm, each consisting of 2000896 steps. Five runs ascribed to the action-dependent condition and five runs ascribed to the action-independent condition.

The data was stored in JSON[1] format, read into a Pandas[2] dataframe, processed further and plotted using Seaborn[3]. To reduce the amount of data and increase interpretability, the data for the line plots was averaged over batches and subsequently smoothed with a running mean of window size 16, using SciPy's[4] *uniform_filter1d*. The shaded area around the line plots depicts the standard deviation.

For the swarm plots the average episode length was chosen for the y-axis, because it is, when calculating the mean over a whole run, a clearer indicator for performance than the average episode reward. This is because of the shaped reward function, where an episode taking 640 steps, one step before failure, would still generate a reward of 0.1. While this helps the algorithms learn, having the aim of intuitive interpretation of the data in mind, this would be hindering. Average episode lengths between 14 and 30 were considered optimal. In the scatter plots this is illustrated by a thick orange line.

Where applicable, the colours Red, Green and Blue were assigned to the algorithms *Curiosity*, *Disagreement* and *RND*, respectively.

[1] JavaScript Object Notation, https://www.json.org/json-en.html
[2] Pandas Python module, https://pandas.pydata.org/
[3] Seaborn Python module, https://seaborn.pydata.org/
[4] SciPy Python module, https://scipy.org/

3.1 Effects on problem-solving

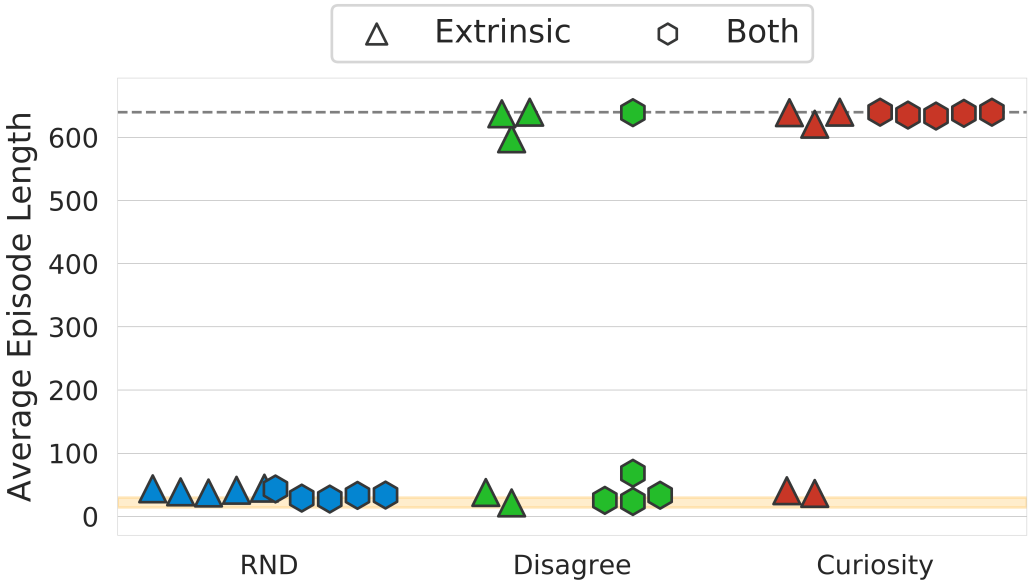


Figure 3.1: Signal-dependent performance per run in *DoorKey* environment

In Figure 3.1 the average episode length per run is plotted for each algorithm and for the reward signals in question. The dotted black line depicts the maximum number of steps allowed, while the orange area at the bottom represents the area of optimal solution. Since the environment differs from trial to trial, a single optimal value could not be determined.

Ignoring the categorization by algorithm, the data follows a bimodal distribution with centers at around 40 and 640, which means that each run either solved the task near to optimal or failed at it completely. This relation will be present throughout the Results chapter.

RND was able to solve the environment with every run close to optimal and irrespective of the reward signal used. Still, the runs using a combination of the reward signals performed marginally better.

Using the extrinsic reward only, *Disagreement* and *Curiosity* performed comparably with two almost optimal runs and three runs failing to learn to solve the environment. Using the combination of rewards though, the performance of the

two related algorithms differed significantly. While *Disagreement* increased performance, solving the environment with four of the five runs, *Curiosity* was not able to learn at all.

RND and *Disagreement* increased performance when integrating both reward signals, but for *Curiosity* learning was impeded completely. Comparing all three algorithms *RND* was able to outperform *Curiosity* as well as *Disagreement* by big margins.

For a juxtaposition of the three algorithms over time, please refer to Appendix B]

3.2 Robustness against stochasticity

This section aims to answer, whether the three algorithms are susceptible to stochastic dynamics and, if so, whether there's a type of stochasticity particularly detrimental.

The section is divided into three parts, one for each algorithm. For each algorithm extrinsic reward and the combination of the reward signals are going to be contrasted by again looking at the average episode length over the three extensions to the (8×8) *DoorKey* environment. These are *external noise*, *random actions* and *random colours*.

Subsequently, the two conditions of stochasticity, namely action-dependent and action-independent, are compared by plotting the episode reward attained over the number of frames the agent has seen. Adding this time variable allows for a more accurate look on the possible effects of stochasticity, which is not necessary for a mere comparison of total performance.

3.2.1 Random Network Distillation

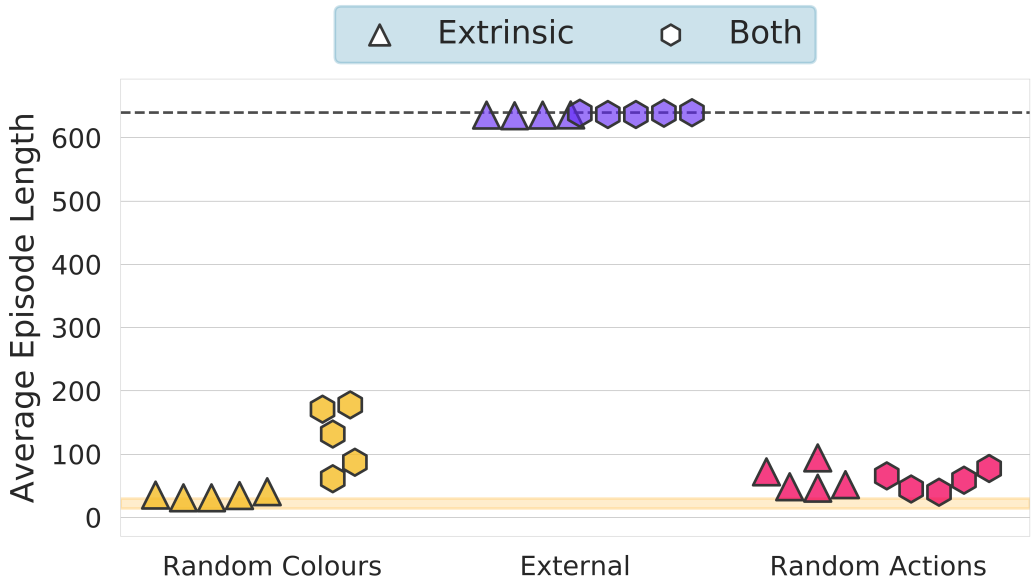


Figure 3.2: *RND's* signal-dependent average performance over action-independent stochastic scenarios

Action-independent Under the action-independent condition *RND* was able to solve the *random colours* and *random actions* scenarios, irrespective of the reward signal applied. Still, in the *random colours* scenario of the *DoorKey* environment, *RND's* underlying *PPO* outperformed *RND* significantly over all runs(compare Figure 3.2) For the *random actions* scenario this difference was small enough to assign them equal performance.

In the *external noise* scenario one run had to be excluded, due to an error while recording. Here, *RND* was not able to learn to solve the environment, regardless of the reward signal implemented. Still, all runs using only extrinsic reward achieved minimally better results than each of the runs performed using the combination of the rewards.

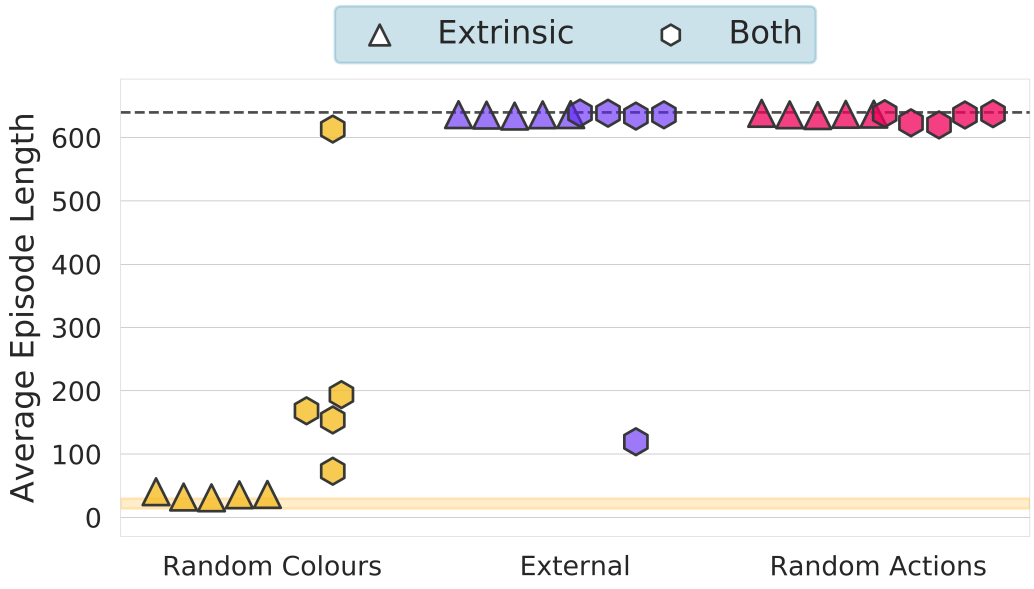


Figure 3.3: *RNDs* signal-dependent average performance over action-dependent stochastic scenarios

Action dependent The most protruding change from Figure 3.2 to Figure 3.3 is *RNDs* performance in the *random actions* scenario. Irrespective of the reward, *RND* was not able to overcome the *DoorKey* environment. In the *external noise* scenario, *PPO* failed in each run, whereas *RND* accomplished to learn to solve the environment with a single run. Only in the *random colours* scenario *RND* as well as its underlying *PPO* managed to solve the environment reliably. *PPO* achieved optimal performance over each run, whereas for *RND* one run failed and only one run came close to optimal performance. The remaining three centered around 190 steps per episode, which is admittedly not close to optimal, but sufficed to be classified as successes.

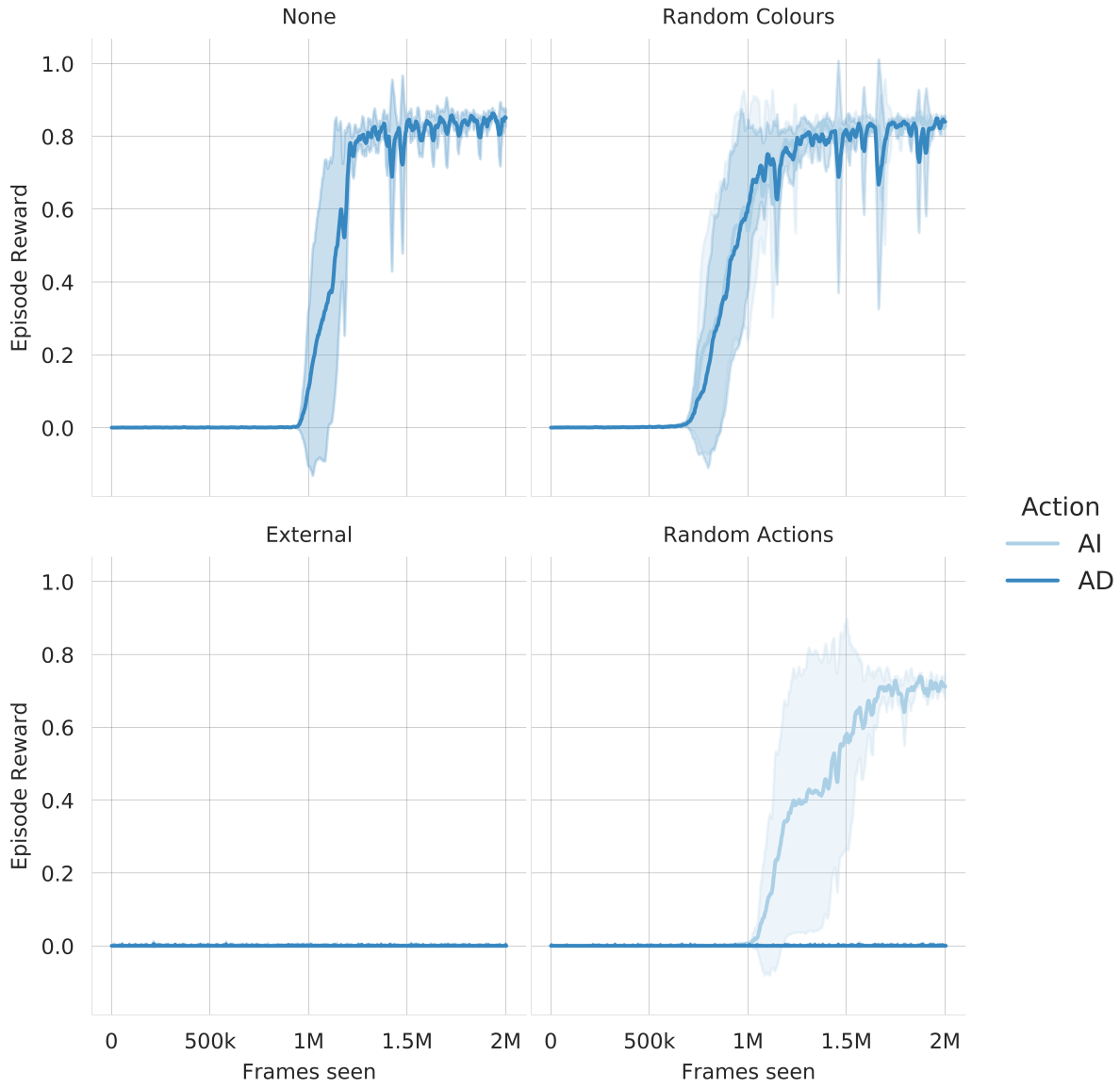


Figure 3.4: *RND's* episode reward over frames seen using only extrinsic reward

Juxtaposition Under the *external noise* scenario, switching between action-dependent and action-independent stochasticity had no impact on *RND's PPO* (compare Figure 3.4). In both cases *PPO* was not able to solve the environment in any run. For the *random colours* scenario switching between action-dependent and action-independent stochasticity made no difference, either. But on the con-

trary, in both scenarios *PPO* managed to solve the environment optimally with each run, resulting in the two line plots being almost indistinguishable. Lastly, for the *random actions* scenario the effects of the two stochasticity conditions do differ. While *PPO* was capable of solving the environment under the action-independent condition, learning did not occur being confronted with the action-dependent version. In relation to the stochasticity scenario *None*, where *PPO* was not marooned to any type of stochasticity, all runs of *random colours* and the action-independent condition of *random actions* performed comparably. The remaining, failed entirely in solving the environment.

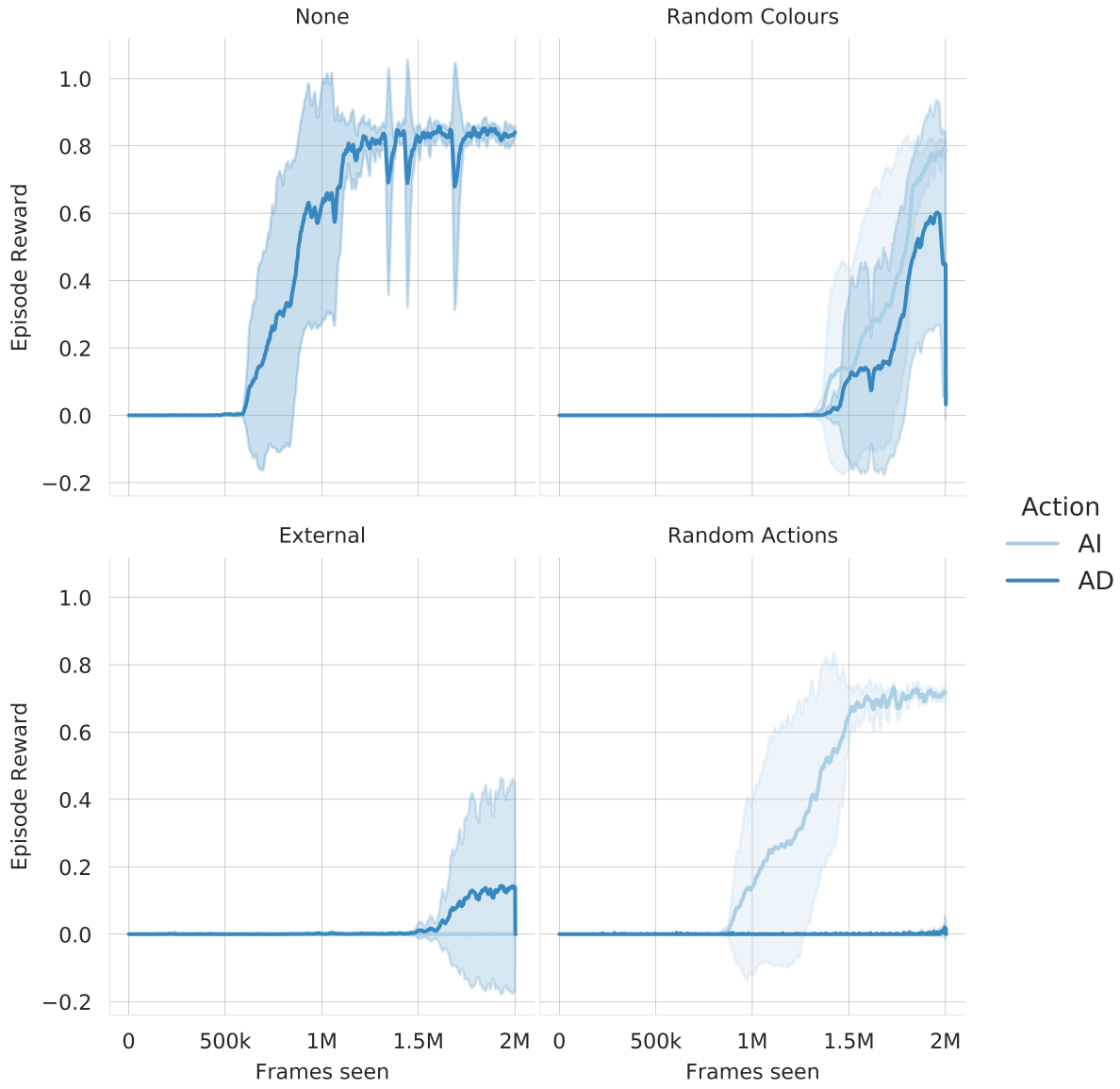


Figure 3.5: *RND's* episode reward over frames seen using both rewards

Figure 3.5 shows the juxtaposition of the action-dependent and -independent stochasticity conditions when *RND* utilized both rewards. Taking the absence of stochasticity, represented by *None*, as a baseline, *RND* performed worse under each condition, irrespective of action dependent or independent noise. However, under both conditions in *random colours* and under the action-dependent condition in *random actions* learning was still successful. Under *external noise* first

sign of learning could be seen for the action-dependent condition, whereas for the action-independent version no learning occured at all.

3.2.2 Curiosity-driven Exploration

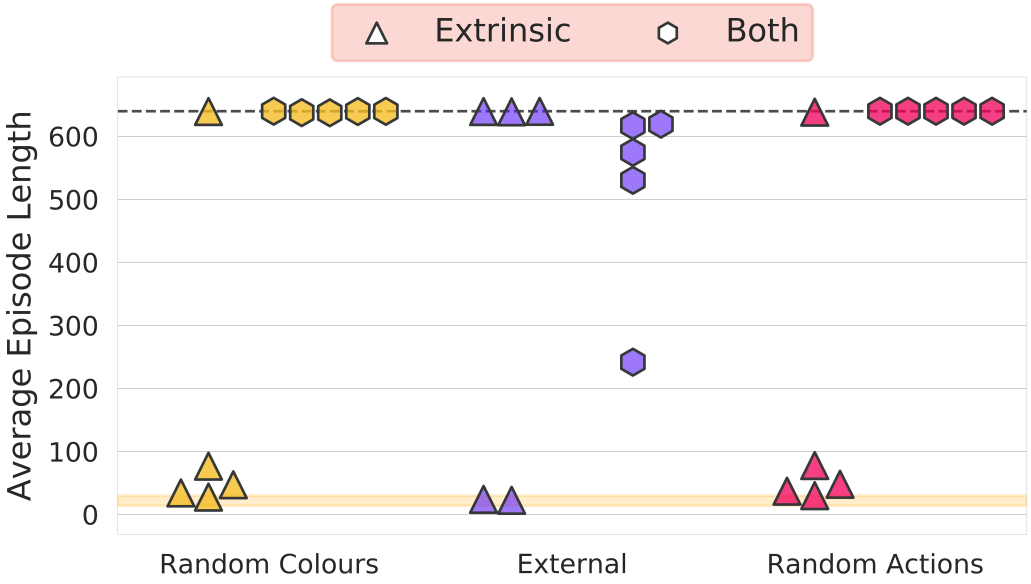


Figure 3.6: Signal-dependent average performance over action-independent stochastic scenarios of *Curiosity*

Action independent As seen before in Figure 3.1 the data in Figure 3.6 is bimodally distributed with its centers at around 640 and 50. Except for the runs performed in the *external noise* scenario using both rewards, each run either solves the environment very good or not at all. The performances in the *random actions* and the *random colours* scenarios look identical, but an inspection of the underlying dataframe shows that they differ by a few steps each. In both scenarios *Curiosity's* underlying *PPO* managed to solve the environment with four runs, while one run failed completely. Adding the intrinsic reward however, the algorithm was not able to achieve any learning. This can be compared to Figure 3.1.

In the *external noise* scenario the underlying *PPO* achieved underoptimal performance in two cases, but failed three runs entirely. By simply looking at the mean performance of a run, the results achieved under *external noise* using both

rewards are difficult to interpret. However, when considering the respective plot in Figure 3.9, one can see that these means originate from early signs of learning. But despite this first learning, the results are far away from reliably and frequently solving the task at hand, wherefore they, except for the run at around 230, were not classified as successes.

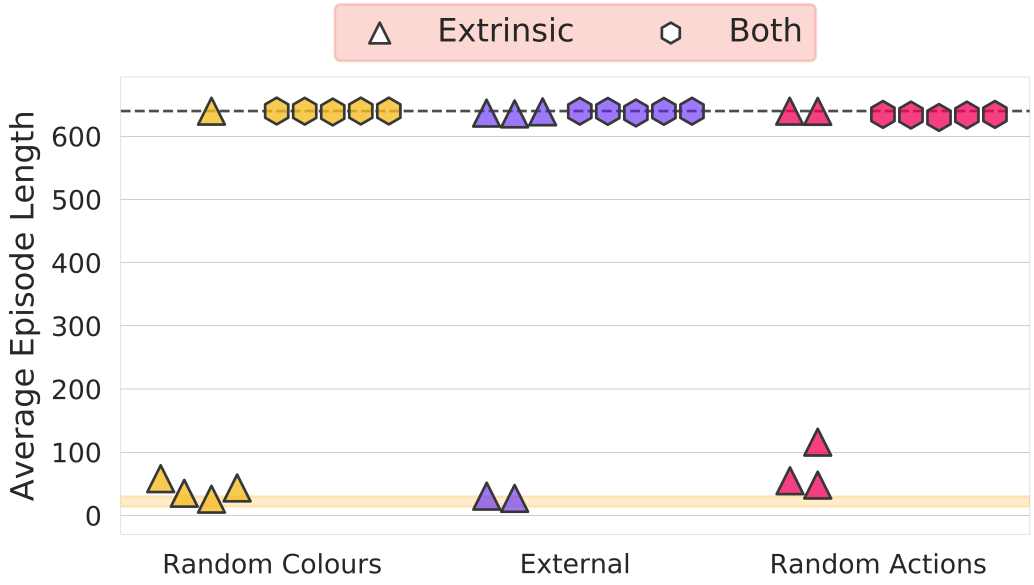


Figure 3.7: Signal-dependent average performance over action-dependent stochastic scenarios of *Curiosity*

Action-dependent The most protruding result of the action-dependent condition shown in Figure 3.7 is that *Curiosity* failed at each run where both rewards were utilized. This follows the trend shown in Figure 3.1 and Figure 3.6. However, *Curiosity's PPO* managed to achieve two, three and four optimal or close to optimal runs for the *external noise*, *random actions* and *random colours* scenarios, respectively.

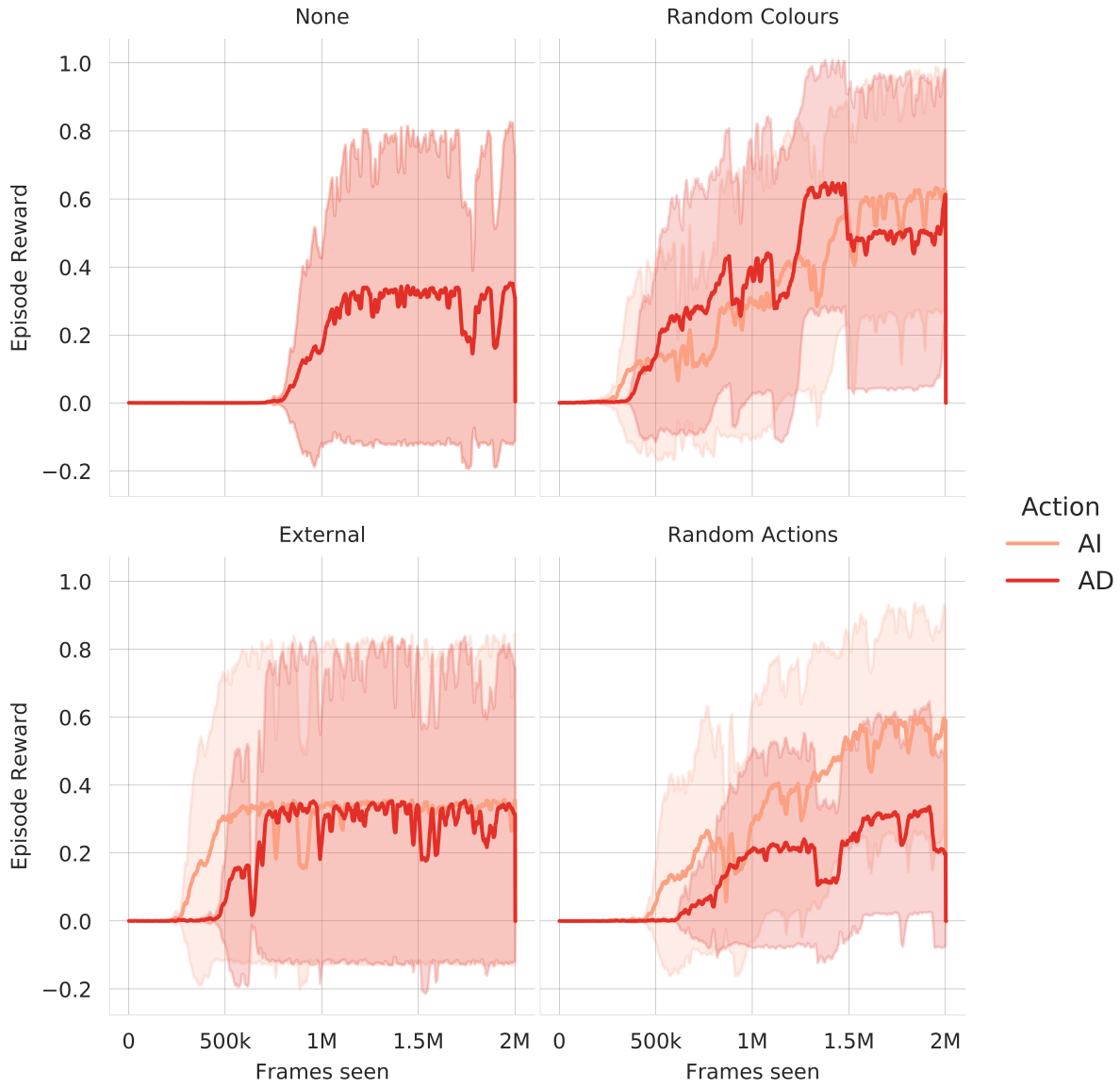


Figure 3.8: *Curiosity's* episode reward over frames seen using extrinsic reward only

Juxtapositions Figure 3.8 shows that when taking the *None* scenario as the baseline, on average, the *PPO* did only perform worse in the action-dependent *random actions* scenario, which it still, on average, learns to solve. Further, except for the *random actions* scenario, it did not make a significant difference for *PPO* whether the noise applied was depended on the action taken or not. In the

random actions scenario, *PPO* performed better in the action-independent condition.

This improves validity of the subsequent comparison of the utilization of both rewards, since the potential confound of the combination of environment and stochasticity condition being in itself harder to solve thus falls away. Only in the *random actions* scenario the mean performances of the stochasticity conditions differ. Here the action-independent condition performs better, while also starting earlier to learn.

The big standard deviations can be derived from different onsets in learning for each run as well as from the bimodal distribution depicted in Figure 3.1,Figure 3.6 and Figure 3.7.

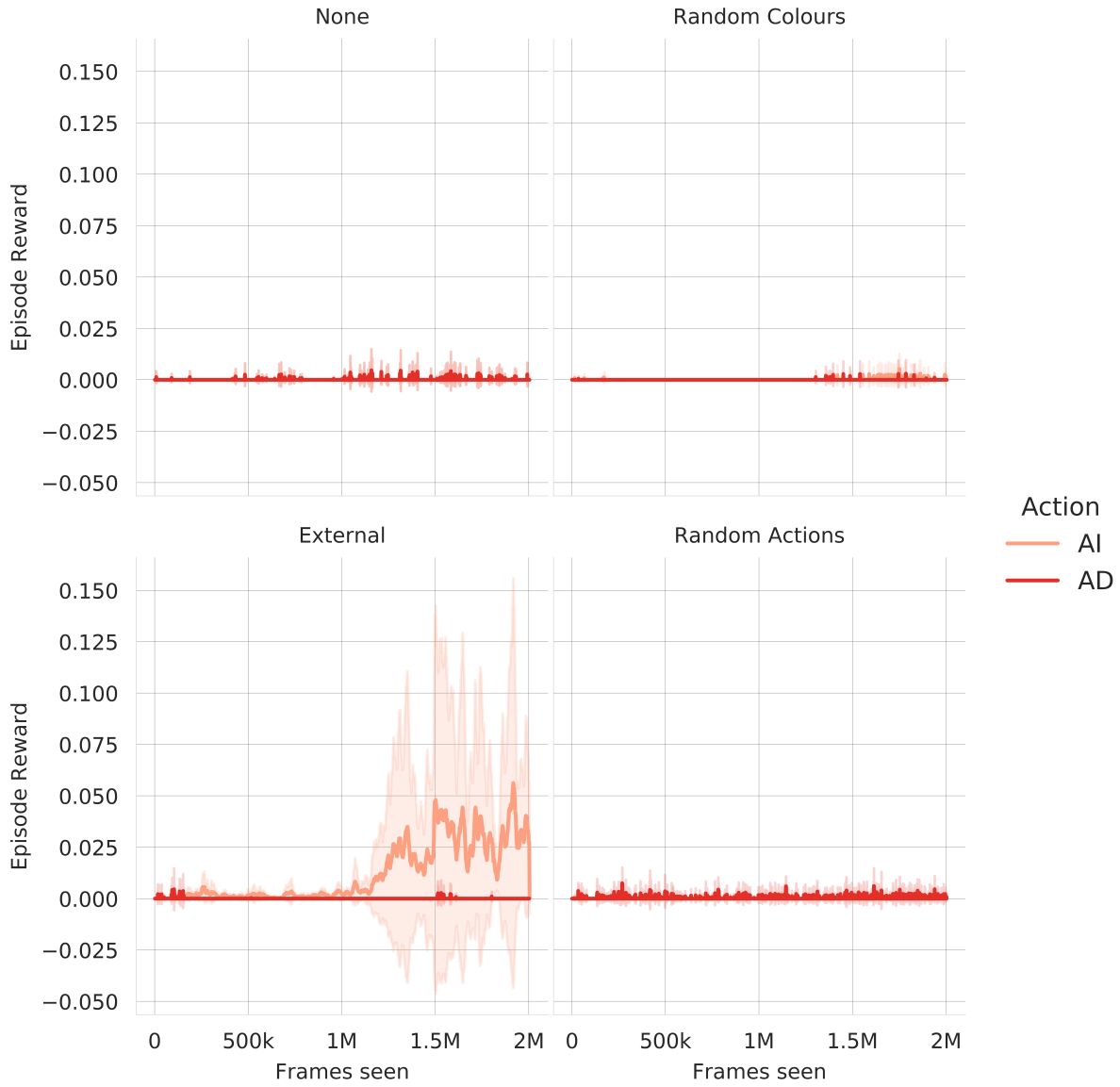


Figure 3.9: *Curiosity's* episode reward over frames seen using both rewards

Figure 3.9 complements Figure 3.6 and Figure 3.7 by two things. The first being the already mentioned onset of learning in the action-independent *external noise* scenario. Secondly, in contrast to the means shown in the foregone scatter plots, a difference in the failure of each scenario can be seen. In the *random actions* scenario, miniscule (mind the changed limit of the y-axis) rewards are obtained regularly, while for *random colours* and *external noise* over long periods

not a single reward could be attained.

3.2.3 Exploration via Disagreement

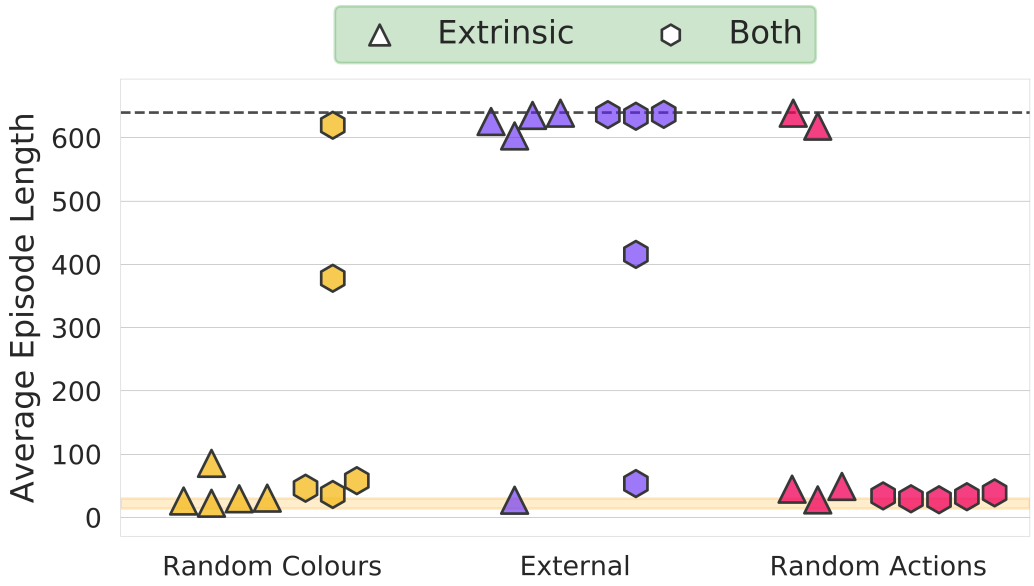


Figure 3.10: *Disagreement's* signal-dependent average performance over action-independent stochastic scenarios

Action independent Figure 3.10 shows the results obtained by *Disagreement*, the last algorithm to be debated. In the *random colours* scenario *Disagreement's PPO* accomplished solving the environment optimally in almost every run. The run with the worst performance only reached near-optimal performance. *Disagreement*, utilizing both rewards, performed less with three near-optimal, one failed and one run that had started to learn. Under *external noise* both *Disagreement* and its *PPO* only achieved solving the environment in one run. *PPO* failing four runs and *Disagreement* three. The one remaining run of *Disagreement* achieved a mean episode length between the two extremes, and will be categorized as failed[WILLKÜR]. While in the *random actions* scenario *Disagreement* performed optimally, it was only able to solve the environment in three runs when integrating just the extrinsic signal.

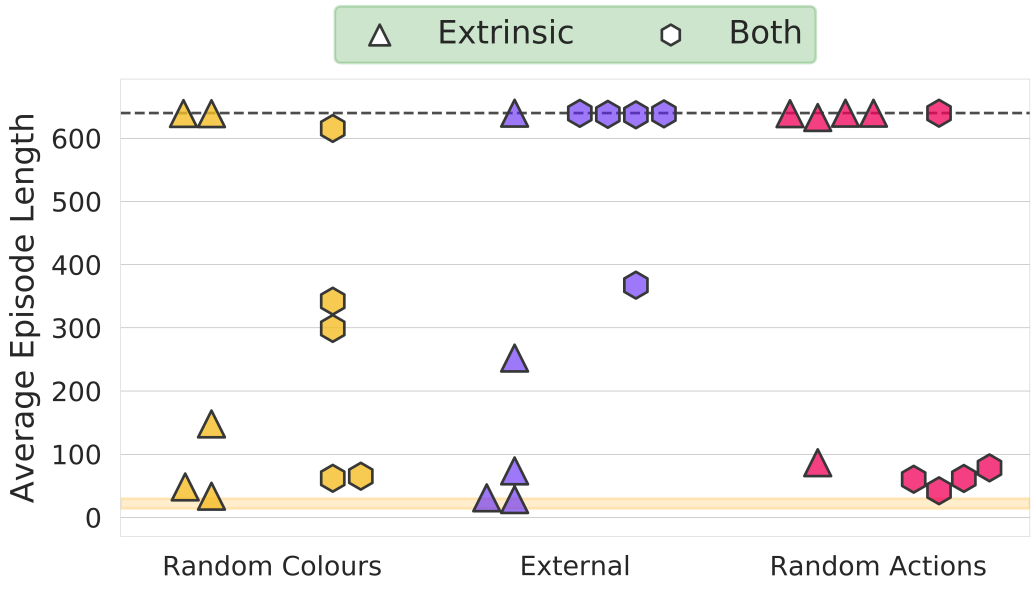


Figure 3.11: *Disagreement's* signal-dependent average performance over action-dependent stochastic scenarios

Action dependent For the action-dependent condition the data obtained is more spread than for its independent counterpart(compare Figure 3.10).

In the *random colours* scenario *Disagreements PPO* accomplished solving the environment three times, while failing two runs. *Disagreement* solved the environment with two runs near-optimal, failed with one time and achieved first ongoing learning with two runs. Comparing the total mean of the two reward signals, *Disagreement* performed on average better than *PPO* with a 276 and 302 as mean average episode lengths respectively.

For the *external noise* scenario things were more obvious. While the underlying *PPO* solved the environment in three runs optimally, *Disagreement* could only exhibit partial success with one run, failing the remaining four completely.

In the last scenario in question, *random actions*, the performance on each reward signal presents the o*PPO*site to the other. *PPO* failed four runs entirely and succeeded with one near-optimally, whereas *Disagreement* succeeded with four runs near-optimal and failed one run entirely.

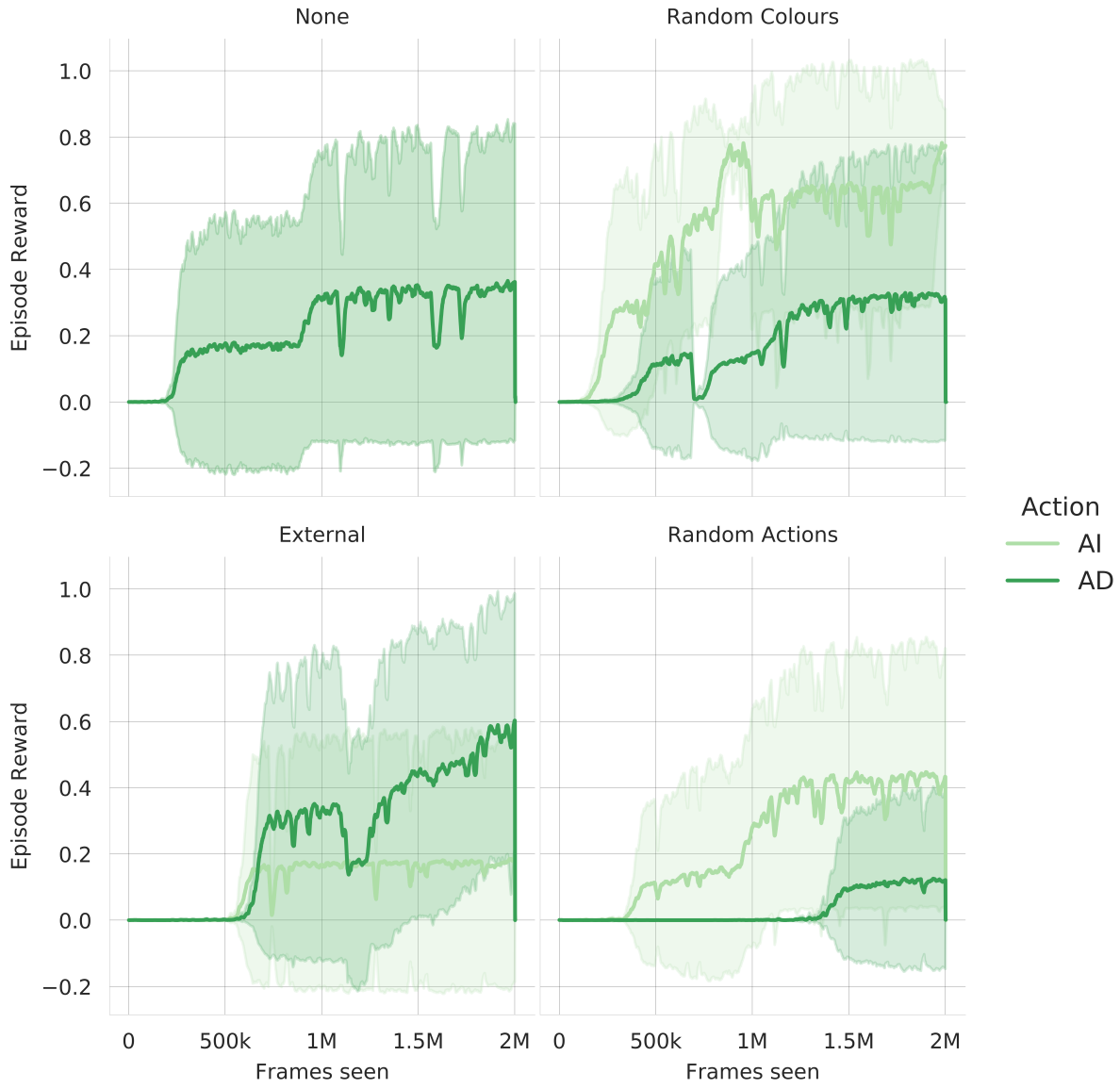


Figure 3.12: *Disagreement's* episode reward over frames seen using extrinsic reward only

Juxtaposition When comparing the noisy scenarios and *None* in Figure 3.12 for *Disagreements PPO*, one can see that for *random colours* the action-dependent version performed a bit worse than baseline, while the action-independent condition actually performed better. The same relation can bee seen for *random actions*, although here the action-dependent condition performed significantly worse

than baseline, with a lower mean performance and a later onset in learning. For the *external noise* scenario the relation shown in *random colours* is inverted. Here the action-independent condition performed significantly worse than baseline, whereas the action-dependent version achieved a perceivably higher average reward.

While for *random actions* and *random colours* the average performance of the action-dependent stochasticity condition was worse, compared to its action-independent counterpart also the learning onset was delayed. In the *external noise* scenario, the action-dependent condition performed on average drastically better, with a final mean reward three times as high.

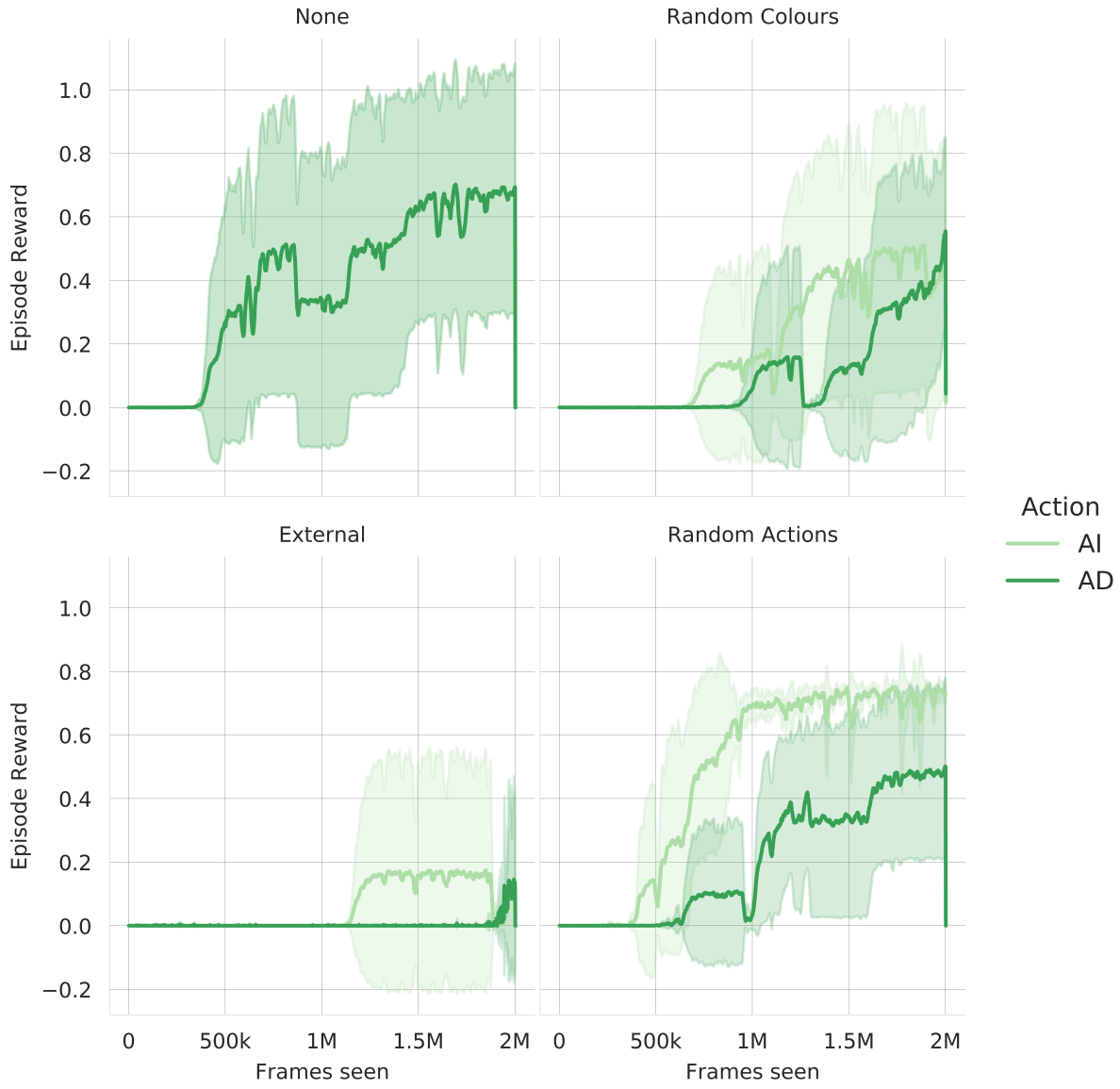


Figure 3.13: *Disagreements's* episode reward over frames seen using both rewards

Figure 3.13 represents the last juxtaposition done in this chapter. The action-independent version of *random actions* is the only situation in which *Disagreement* performed better than baseline. Under every other kind of stochasticity *Disagreement* performed worse. Particularly the *external noise* scenario was difficult to solve for the algorithm. Further, the average performance of the action-dependent condition was for each noisy scenario worse than the action-independent

condition. Here, the shape of the plots stayed the same, while in addition to the lower average performance, the learning onset was delayed as well.

4 Discussion

The work documented here tried to answer two main questions: Does the intrinsic reward, implemented by the algorithms presented, facilitate problem-solving? And, are the algorithms using this intrinsic reward robust against stochasticity?

4.1 Findings

Random Network Distillation When testing for problem-solving in the non-stochastic *DoorKey* environment, *RND* was able to reach optimal performance in each run. But the *PPO* implementation underlying *RND*, using only the extrinsic reward, managed to do so as well, leaving no space for improvement. In the experiments around stochasticity, *RND* did not perform better than its *PPO*, either. *RND's PPO* implementation seems to be very efficient and thus, no significant difference could be recognized. Contrasting this with the performance of the other algorithms' *PPO*, this result suggests, that tuning of *PPO*, besides the in the paper stated extension to two value-heads, might have been undertaken.

When testing *RND* for robustness three main conclusions could be obtained. The first being, that the *external noise* scenario, action dependent as well as independent, was not solvable for *RND's* base *PPO* at all. Thus, *RND's* equally bad performance might be traced back to the inability of its base, leaving us with no clear assertion to make about *RND's* susceptibility to this stochastic scenario. Here the difference between the *PPO* implementations must be underlined. While *external noise* being difficult to solve, the base *PPO* that *Curiosity* and *Disagreement* share, was able to learn in this scenario on multiple occasions. The second conclusion, however, was that *RND*, in contrast to its *PPO*, was susceptible to the noise established in the *random colours* and the *random actions* scenarios. Nonetheless, it was robust enough to learn to solve the task reliably. The last conclusion drawn from the data presented in the results, is that for the *random colours* and the *random actions* scenarios, *RND* performed worse being confronted with stochastic dynamics that were dependent on its actions.

So, under the conditions tested *RND* had, in relation to its base algorithm, no advantage through the integration of an intrinsic reward signal. Yet, under the conditions tested it achieves the highest performance compared to the other two algorithms. *RND* did perfrom

Curiosity-driven Exploration The results surrounding *Curiosity* stand in stark contrast to the other two algorithms. Implementing both reward signals resulted in *Curiosity* being unable to solve the task in almost any testing scenario, while its base *PPO* still managed to perform quite well over the different tests. The only condition were *Curiosity* managed to learn, was the action-independent condition of the *external noise* scenario. The reasons for this failing could range from too short training time over susceptibility to initialization of network parameters over sensibility to hyperparameters to perhaps simply an error in the code. Upon analysis, I investigated the setup and code used, and did, on thorough observation, not find any mistakes. Looking at recordings of the agent, did not reveal anything suspicious, either. These results are unexpected, but the difficulty of reproducibility in deep RL is an important issue for which, still proper methods and standardization are needed [Henderson et al., 2017].

Since the range of possibilities makes pinpointing the exact cause of this go beyond the scope of this thesis, however, I need to conclude that the data gathered suggests that for *Curiosity* the added intrinsic reward signal does not improve problem-solving under the tested conditions. Further, the data suggests that *Curiosity* might be highly susceptible to each of the stochastic conditions tested. This stands in stark contrast to the work done in [Burda et al., 2018a] as well as the original paper [Pathak et al., 2017], but for proper clarification more testing is necessary.

Exploration via Disagreement Lastly, the conclusions drawn from the data generated by *Disagreement*, the successor to *Curiosity*.

The first thing to be said is, that *Disagreement* does not demonstrate the issues encountered when investigating *Curiosity*. Viewing the problem-solving capabilities of *Disagreement*, it could clearly be observed that the addition of the intrinsic reward signal significantly increased performance.

The results regarding stochasticity are not that simple to grasp, since *Disagreement* performed quite diversely in each noisy scenario. Firstly, it can clearly be stated, that due to no change in performance when confronted with the uncertainty constituted by *random actions*, the data suggests that *Disagreement* is not susceptible to that kind of stochasticity.

In the *random colours* scenario for both stochasticity conditions *Disagreement* performed worse than its *PPO* base, but still managed to reliably learn to frequently attain extrinsic reward. Independent of stochasticity condition, *Disagreement* performed worst being confronted with the *external noise* scenario.

Disagreement performs, like *RND*, worse being confronted with action-dependent stochasticity instead of the independent condition. This is similar to the observations made in [Burda et al., 2018a] regarding *Curiosity* and suggests that the susceptibility to action-dependet noise must not necessarily be related to the design flaw stated by [Burda et al., 2018a] in *Curiosity's* inverse dynamics approach.

While being susceptible to the above mentioned types of stochasticity, *Disagreement* nonetheless remains robust and reliably learns to solve the task investigated.

4.2 Implications and limitations

Implications The work documented in this thesis contributes to a more differentiated and realistic understanding of established deep reinforcement learning algorithms. Papers often pick the best of their results while leaving out little events leading not to supporting evidence. By applying these algorithms to a small custom toy-environment with many differing scenarios, this work accomplished to demonstrate some of these events, like the high efficiency of *RND's* PPO implementation, *Curiosity's* issue when integrating both reward signals. Furthermore, this thesis is a good demonstration of the difficulty of reproducing results in DRL. DRL methods often suffer from high instability [Henderson et al., 2017] and even for simple toy environments this can apply. Lastly, when comparing the results gathered, one algorithm clearly emerges victorious. Despite the simplicity of the approach, *RND* was able to achieve high performances while being more robust toward stochasticity than the other two. But as the results from the *external noise*

scenario suggest, even here future work needs to be done.

Limitations The results obtained are mostly limited to the magnitude of the experiments I was able to perform within the frame of a bachelor's thesis. The algorithms investigated need to be tested more often, longer and more environments to properly grasp their merits and drawbacks. While I was able to give a good overview of the state-of-the-art, with the sample sizes at hand no statistical significant statements could be made.

4.3 Outlook

Deep reinforcement learning is a promising paradigm, able to achieve great results. Yet, still a lot of research needs to be done. For the approach of intrinsic motivation, many things remain open. For example research regarding the explorative behaviour constituted through these intrinsic rewards, might be fruitful. Particularly when contrasting with how humans explore.

Further, the weighing of the two reward signals needs to be investigated. In this work the balance was constantly fifty/fifty. However, mediating between differing demands is necessary to solve more complex environments, probably even in a hierarchical manner.

Lastly, regarding DRL itself, new methods need to be established to assure productivity and reproducibility for future achievements. The papers [Henderson et al., 2017] and [Burda et al., 2018a] set both a great example for how this research should look.

4.4 Conclusion

In this study, I tested three deep reinforcement learning algorithms, which utilize different kinds of intrinsic motivation, on a two-dimensional grid-world environment to investigate their performance using two reward signals as well as their susceptibility to stochastic dynamics under differing noisy scenarios. While *RND* and *Disagreement* demonstrated good performance and robustness against stochastic-

ity, *Curiosity* failed to accomplish anything in the given conditions. While a good overview of the potential of these approaches could be given, to cover the complexity of the algorithms, further research needs to be done.

Bibliography

[Burda et al., 2018a] Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. (2018a). Large-scale study of curiosity-driven learning. In *arXiv:1808.04355*.

[Burda et al., 2018b] Burda, Y., Edwards, H., Storkey, A., and Klimov, O. (2018b). Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*.

[Cho and So, 2018] Cho, P. S. and So, W. C. (2018). A feel for numbers: The changing role of gesture in manipulating the mental representation of an abacus among children at different skill levels. *Frontiers in Psychology*, 9.

[Engel et al., 2013] Engel, A. K., Maye, A., Kurthen, M., and König, P. (2013). Where's the action? The pragmatic turn in cognitive science. *Trends in Cognitive Sciences*, 17(5):202–209.

[Erlacher et al., 2012] Erlacher, D., Stumbrys, T., and Schredl, M. (2012). Frequency of lucid dreams and lucid dream practice in german athletes. *Imagination, Cognition and Personality*.

[Henderson et al., 2017] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2017). Deep reinforcement learning that matters.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*.

[Mnih et al., 2016] Mnih, V., Puigdomènech Badia, A., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. *CoRR*.

[Murray, 2004] Murray, C. D. (2004). An interpretative phenomenological analysis of the embodiment of artificial limbs. *Disability and Rehabilitation*.

[Pathak et al., 2017] Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*.

[Pathak et al., 2019] Pathak, D., Gandhi, D., and Gupta, A. (2019). Self-supervised exploration via disagreement. In *International Conference on Machine Learning (ICML)*.

[Ramachandran and Rogers-Ramachandran, 2000] Ramachandran, V. S. and Rogers-Ramachandran, D. (2000). Phantom limbs and neural plasticity. *Archives of Neurology*.

[Ryan and Deci, 2000] Ryan, R. M. and Deci, E. L. (2000). Intrinsic and extrinsic motivations: classic definitions and new directions. *Contemporary Educational Psychology*.

[Schmidhuber, 1991] Schmidhuber, J. (1991). A possibility for implementing curiosity and boredom in model-building neural controllers. *Proceedings of the International Joint Conference on Neural Networks*.

[Schulman et al., 2017] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. The MIT Press.

[Wallace, 1975] Wallace, R. K. (1975). Physiological effects of transcendental meditation. *Science*.

A Appendix One

A.1 Hardware

Asus Notebook - Intel I7 CPU - 12 GB DDR4 RAM - 1000 GB SSD - NVIDIA GPU (not involved here)

Dell Notebook - Intel I7 CPU - 16 GB DDR4 RAM - 500 GB SSD - NVIDIA GPU (not involved here)

A.2 Hyperparameters

These Hyperparameters were adjusted to enable clear comparability

	RND	Disagree	Curiosity
gamma	0.99	0.99	0.99
cliprange	0.1	0.1	0.1
lambda	0.95	0.95	0.95
minibatches	4	8	8
policy	rnn	cnn	cnn
lr	0.0001	0.0001	0.0001
ent_coeff	0.001	0.001	0.001
nepochs	4	3	3
nsteps	128	128	128
num_dynamics	-	5	5
num_env	32	128	128
var_output	-	True	True
ext_coeff	2	0	0
int_coeff	1	1	1

Table A.1: Default Hyperparameters

	RND	Disagree	Curiosity
gamma	0.99	0.99	0.99
cliprange	0.1	-	-
lambda	0.95	0.95	0.95
minibatches	8	8	8
policy	cnn	cnn	cnn
lr	0.0001	0.0001	0.0001
ent_coeff	0.001	0.001	0.001
nepochs	4	4	4
nsteps	128	128	128
num_dynamics	-	5	1
num_env	8	8	8
var_output	-	True	True
ext_coeff	1	1	1
int_coeff	1	1	1

Table A.2: Hyperparameters after adaption

B Appendix Two

When the first draft of this thesis contained about 80 pages, I realized that a lot of the work done over the last months was more due to curiosity than to expedient thinking. Because I value quality over quantity, I decided to slim the thesis down and put things being neither productive to answer the questions asked nor sound in their scientific manner in this appendix. They are still highly interesting and might someday motivate an interested reader to further scientific work.

This chapter presents everything that's out of my thesis' scope, but of which I could not keep my hands off.

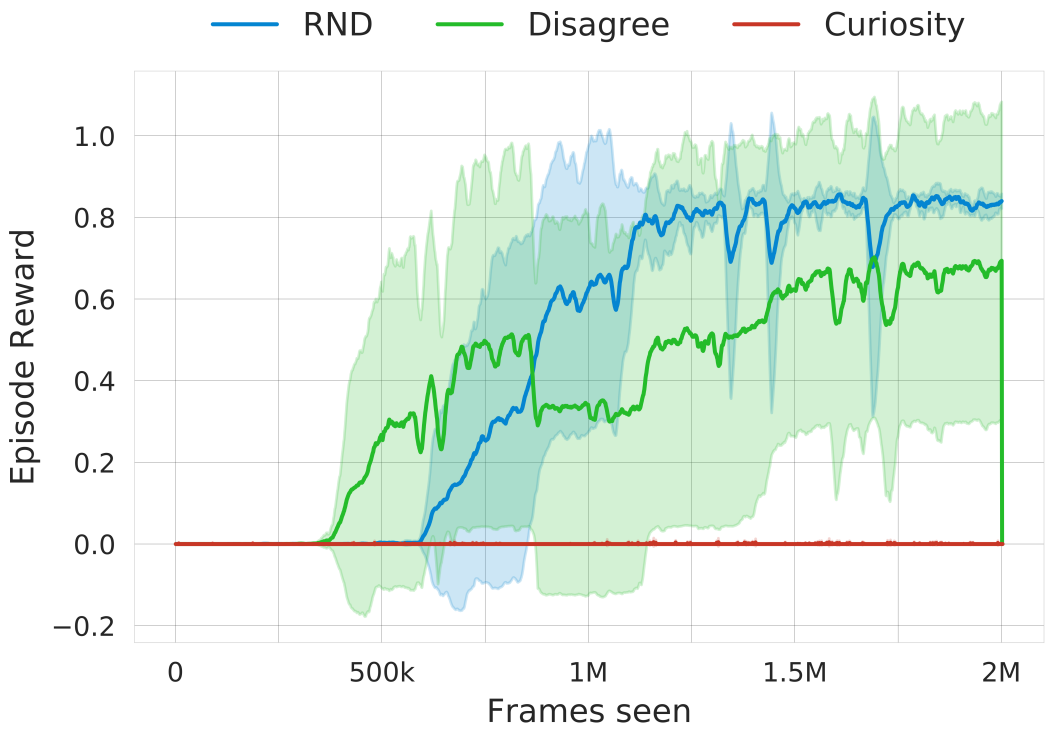


Figure B.1: Performance comparison using both reward signals

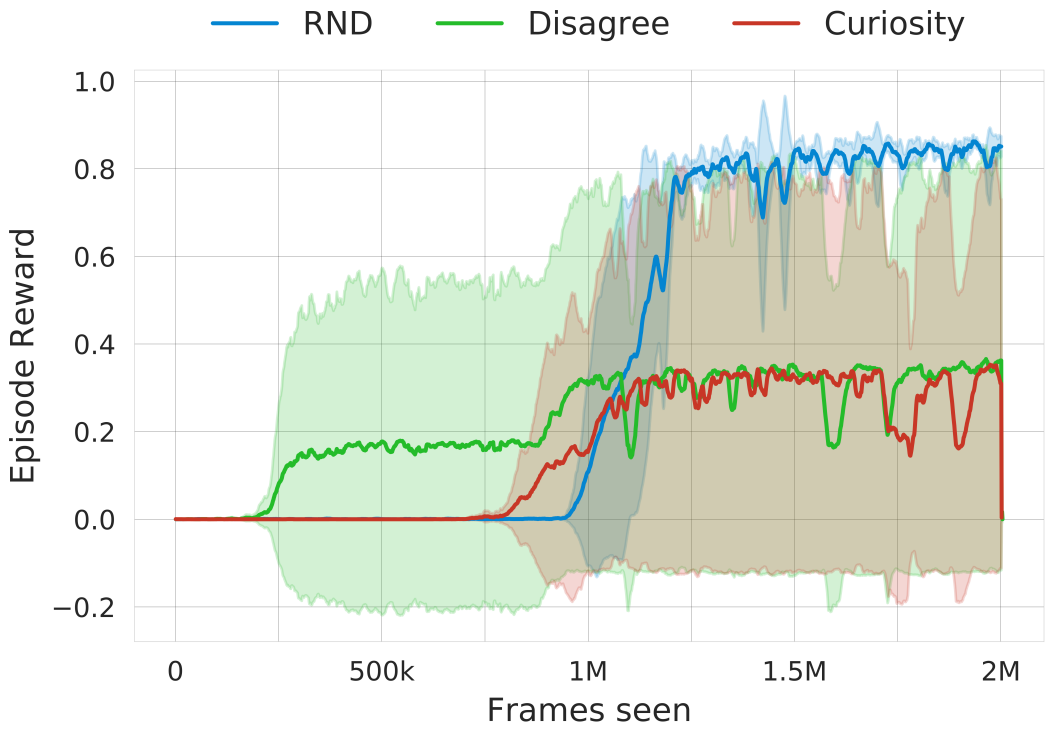


Figure B.2: Performance comparison using only extrinsic reward(PPO)

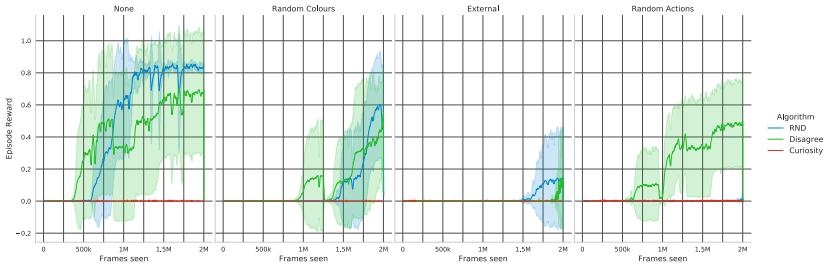


Figure B.3: Ad both

B.1 Exploration

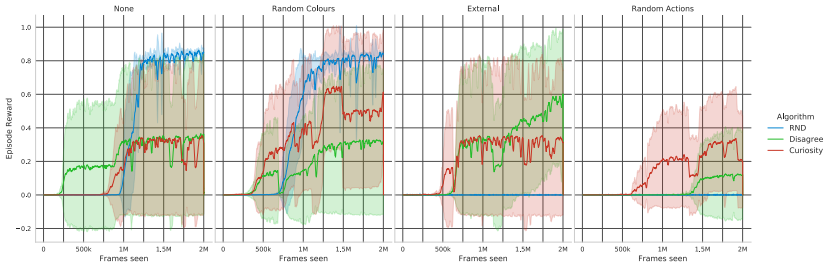


Figure B.4: Ad extrinsic

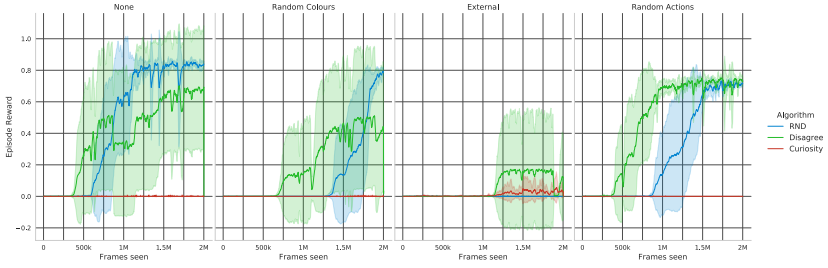


Figure B.5: AI Both

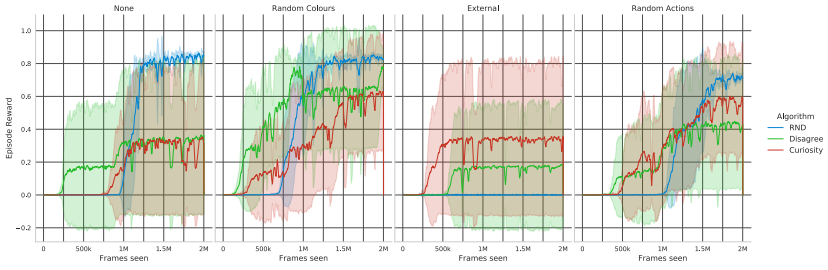


Figure B.6: AI Intrinsic

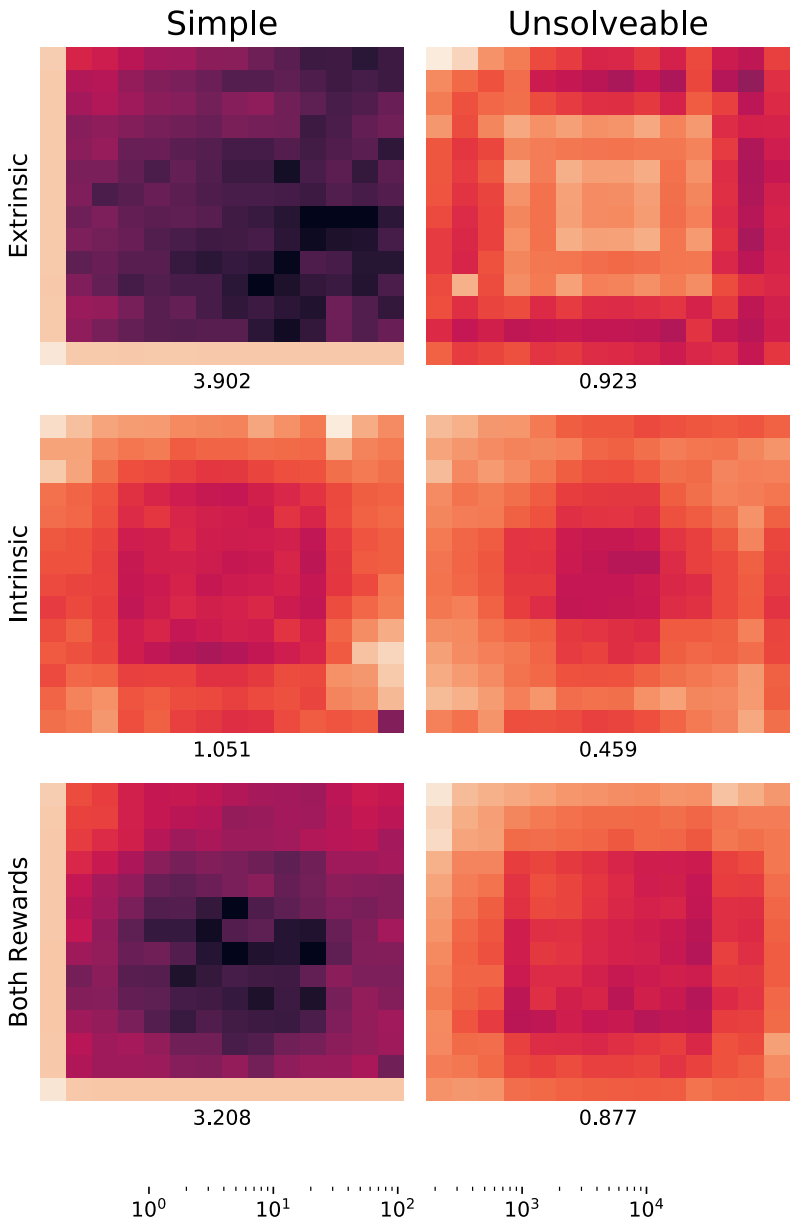


Figure B.7: Disagree Log

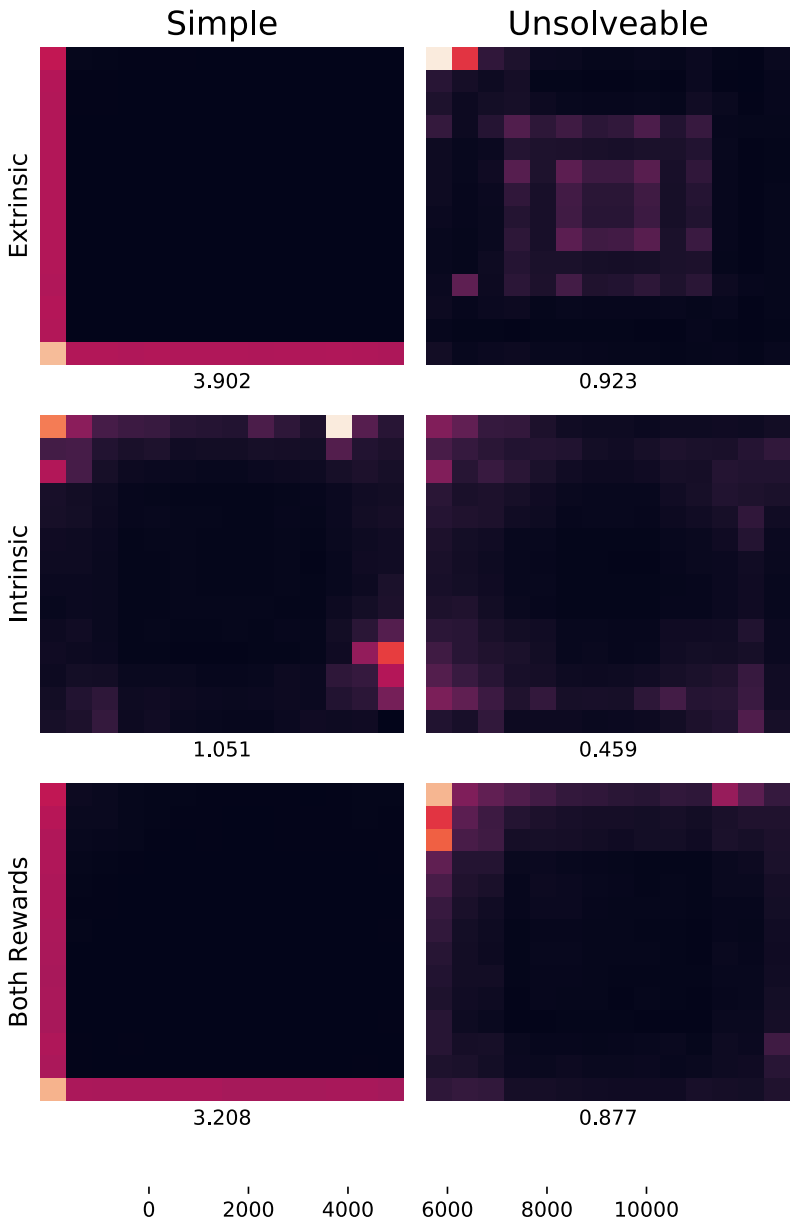


Figure B.8: Disagree

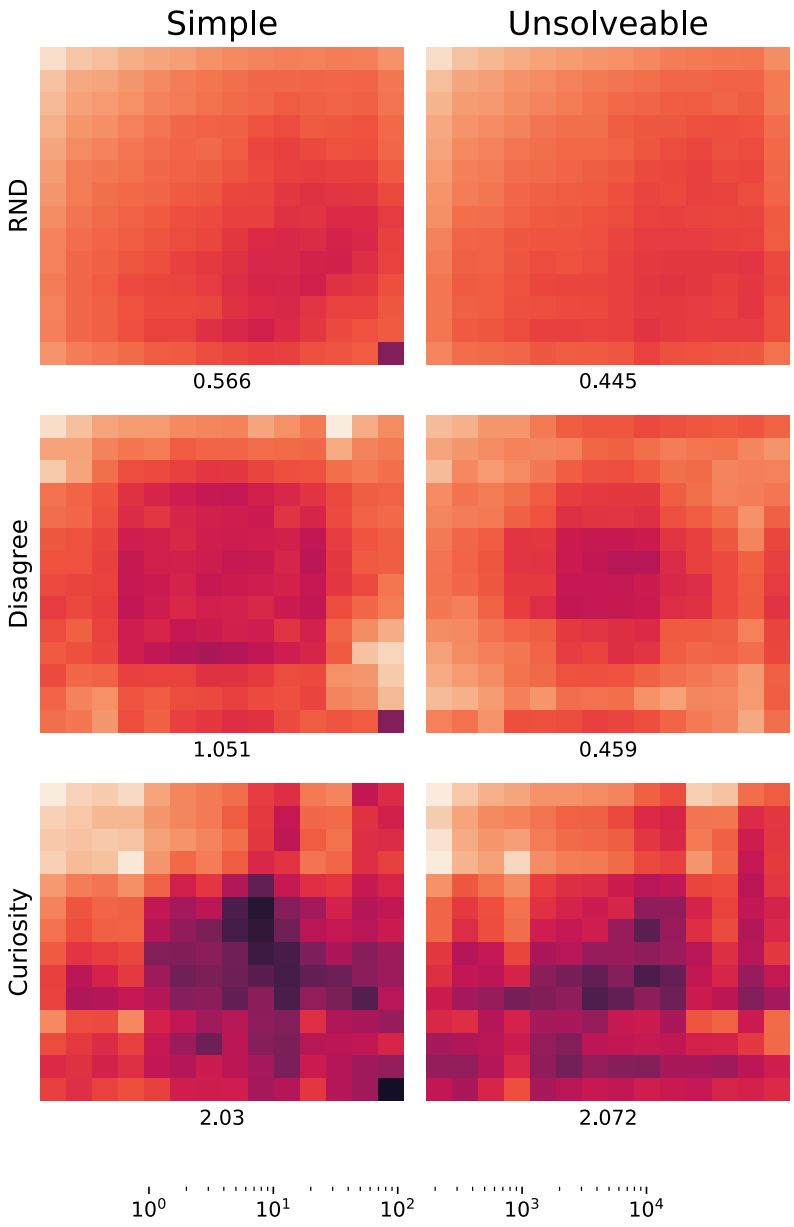


Figure B.9: Intrinsic Log

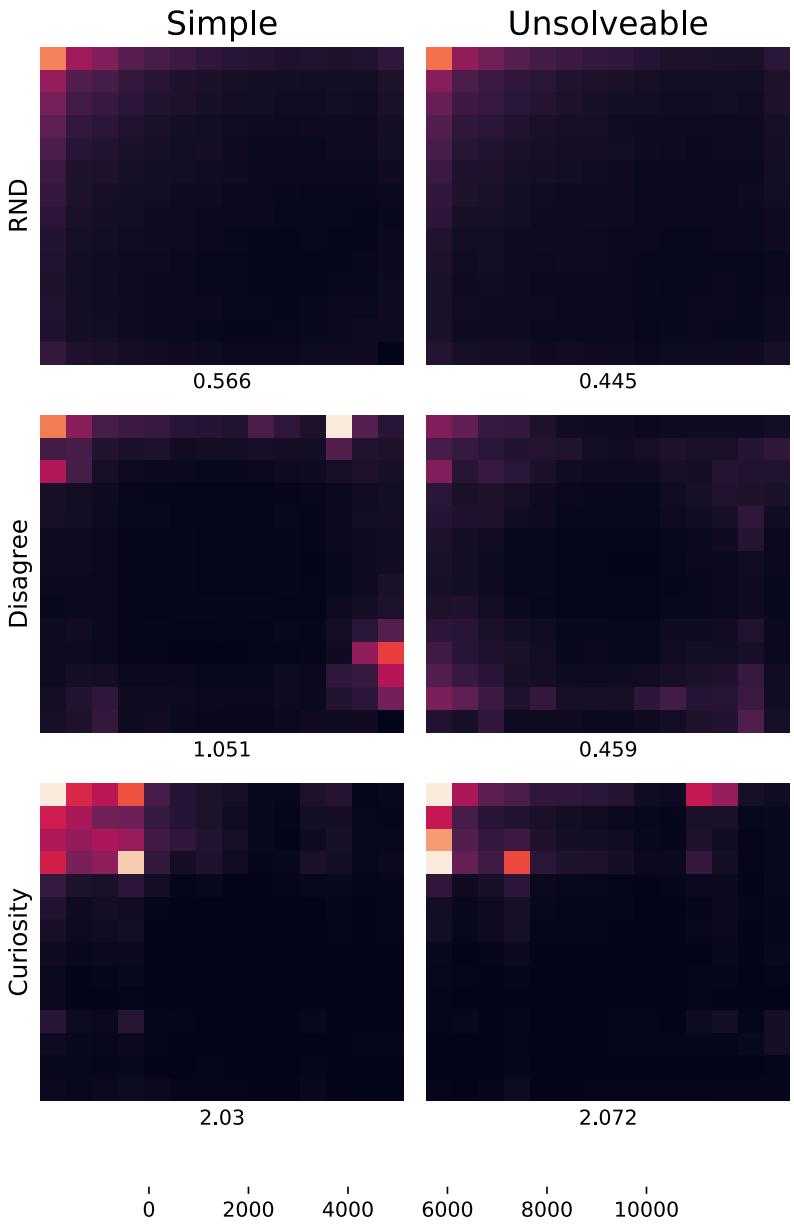


Figure B.10: Intrinsic

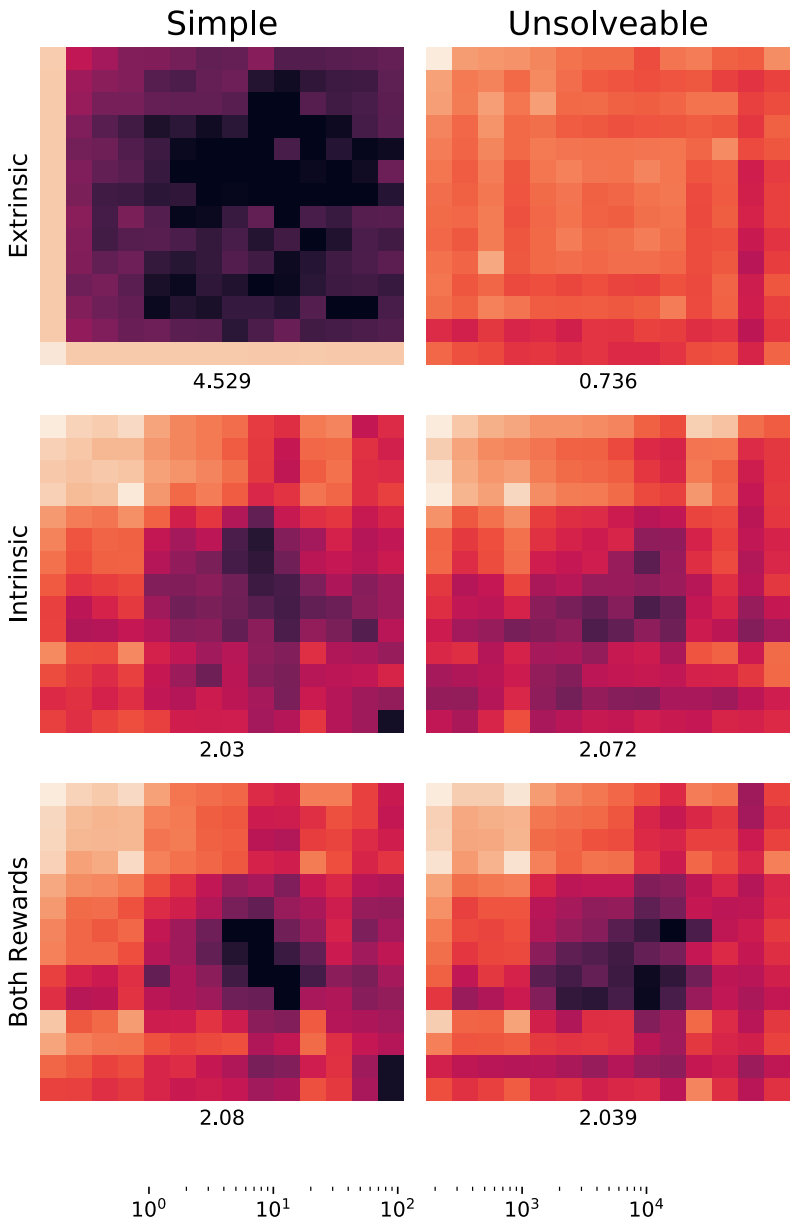


Figure B.11: Novar Log

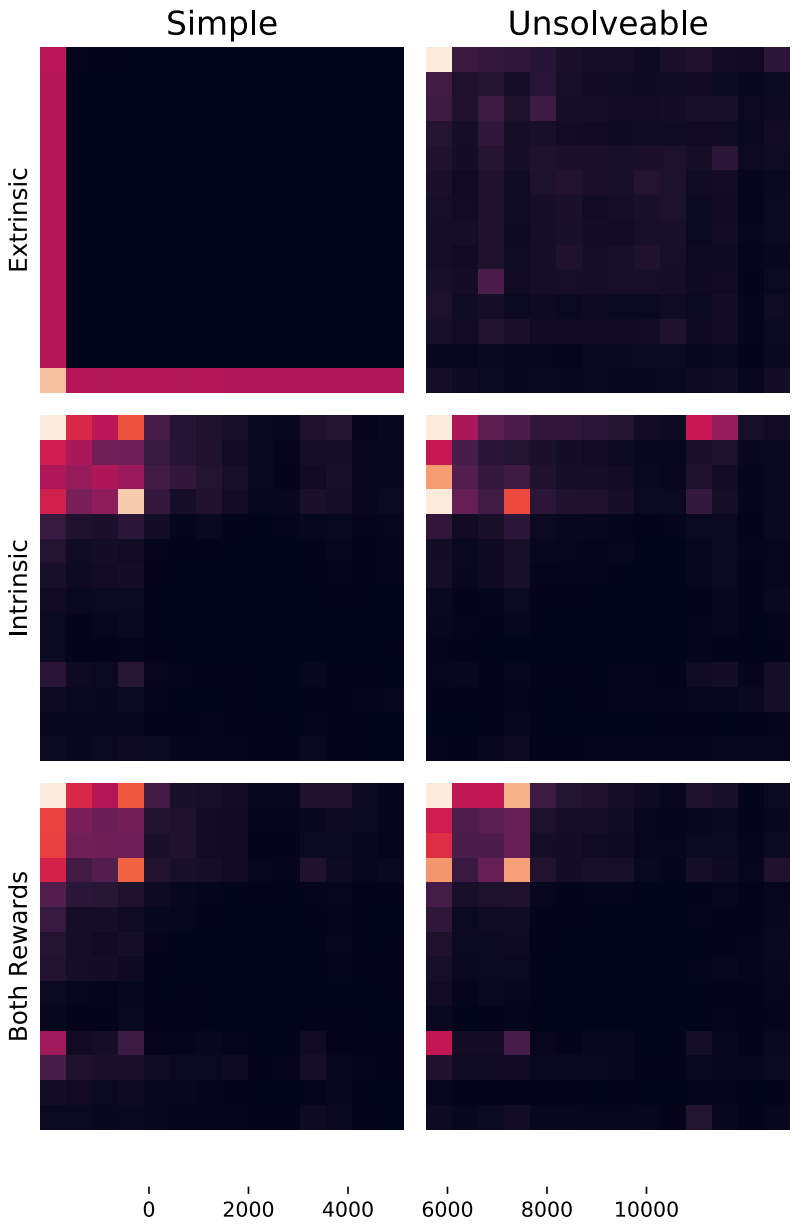


Figure B.12: Novar not Solved

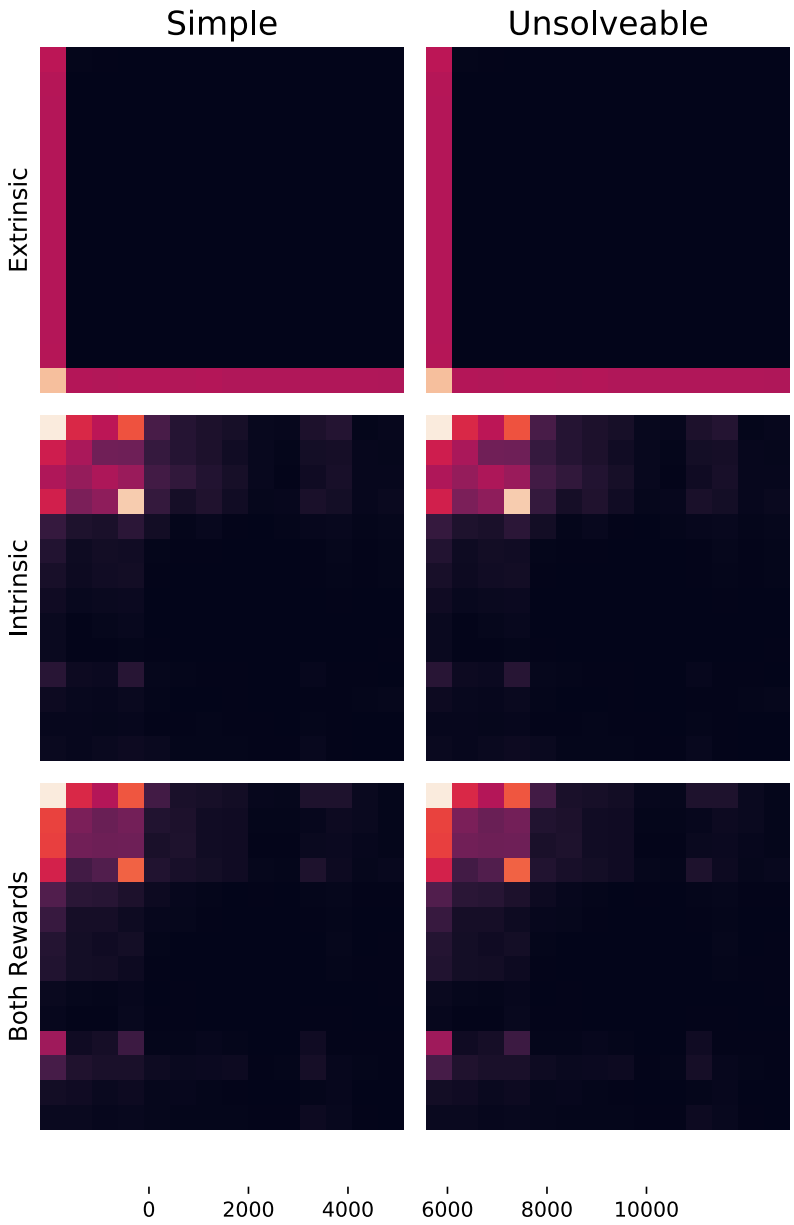


Figure B.13: Novar simple

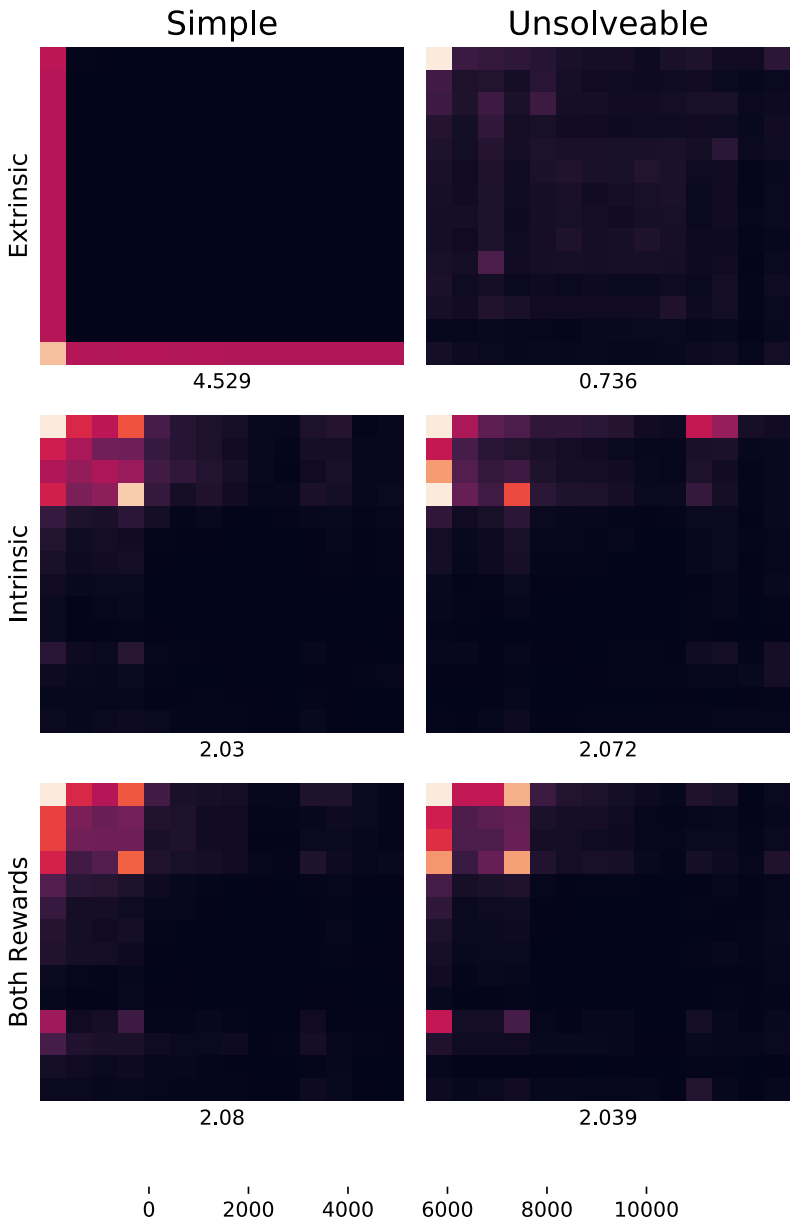


Figure B.14: Novar

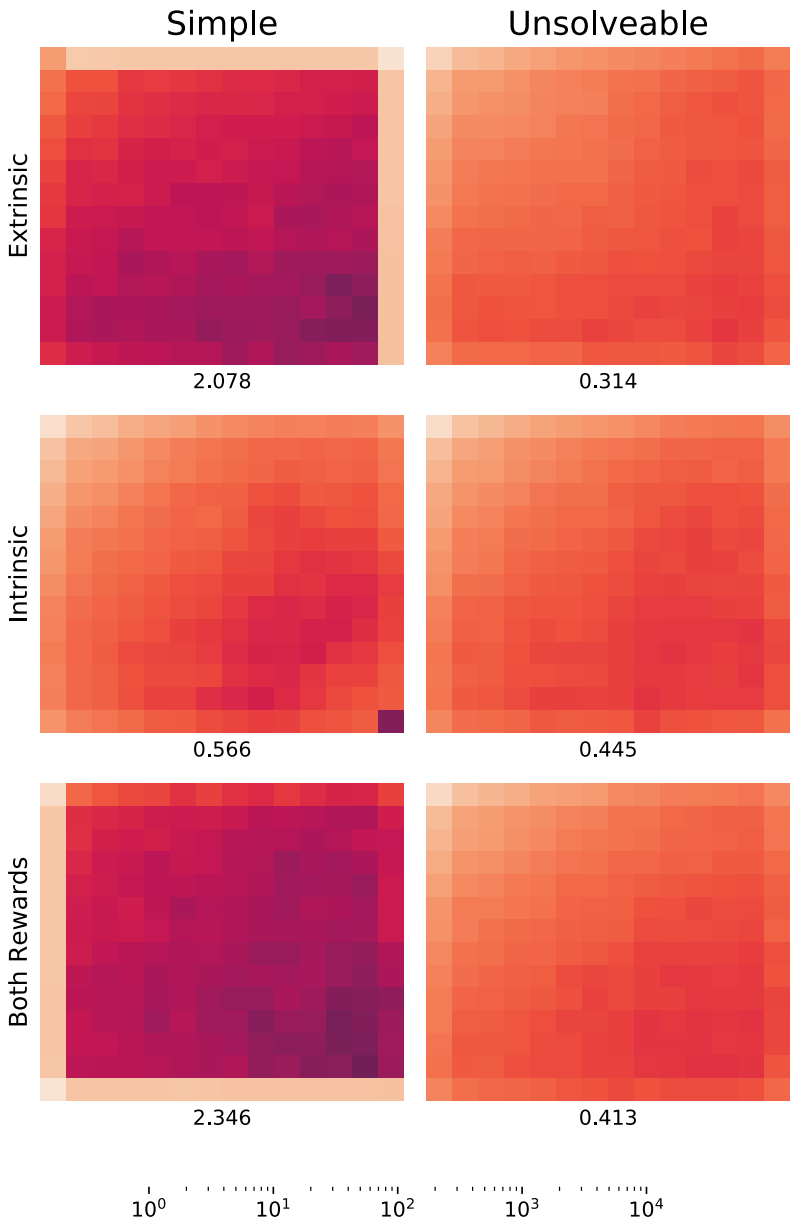


Figure B.15: RND Log

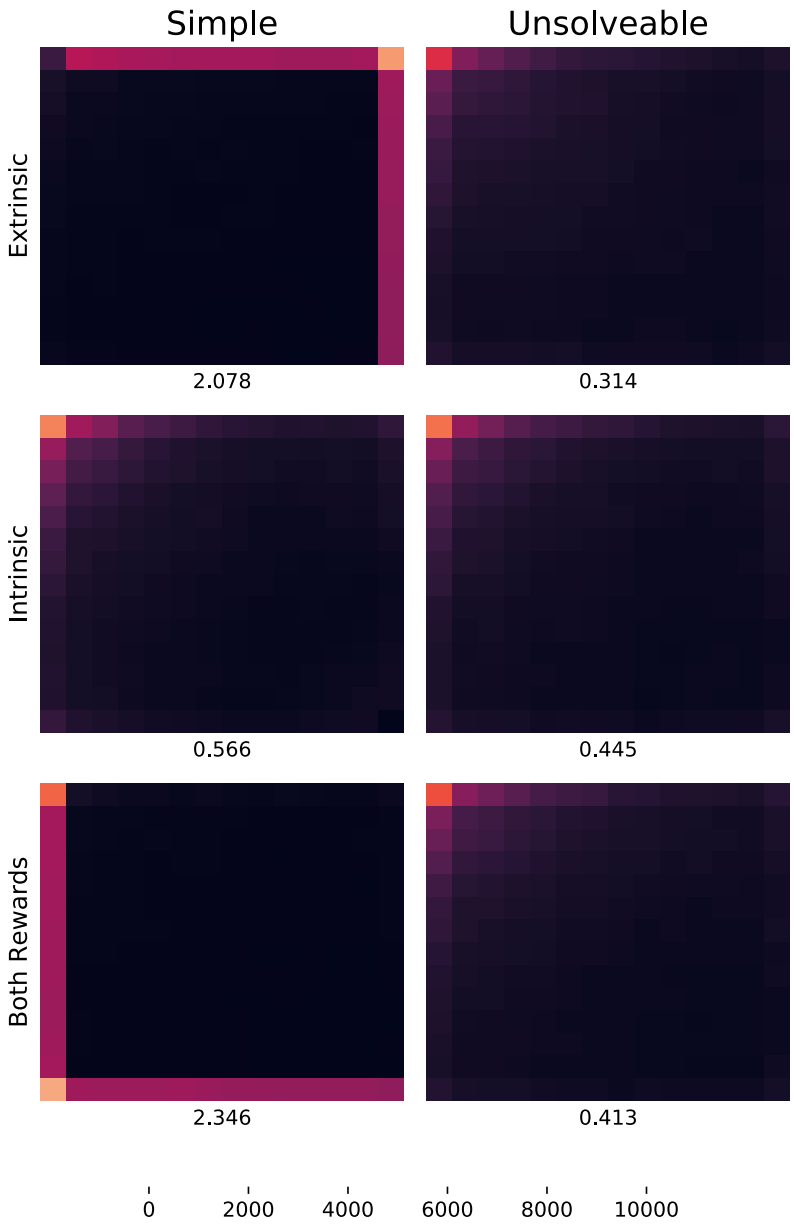


Figure B.16: RND