Leibniz Universität Hannover

HÁSKÓLINN Í REYKJAVÍK
REYKJAVIK UNIVERSITY

Institute of Photogrammetry and Geoinformation
Leibniz Universität Hannover

Center for Analysis and Design of Intelligent Agents
Reykjavik University

Masterarbeit

Task-environments for evaluating learning and autonomy in AI architectures

Leonard M. Eberding

Hannover, 10. März 2020

Erstpüfer:	Prof. Dr.-Ing. habil. Christian Heipke
Zweitprüfer	Prof. Dr. Kristinn R. Thórisson

Declaration of Authorship

I hereby declare that I am the sole author of this master thesis and I certify that, to the best of my knowledge, I have not used any sources other than those listed in the bibliography and identified as references. I further declare that I have not submitted this thesis at any other institution in order to obtain a degree.

Leonard M. Eberding

Abstract

Since the field of artificial intelligence (AI) was first given that name in 1956, different methodological research approaches have resulted in a split into two branches, ”narrow” artificial intelligence (AI) and artificial general intelligence (AGI). Different approaches with different definitions of the same terms has lead to further divergence over the past decades. One of the problems persisting is the unavailability of methods for measuring generality of narrow AI systems. While many researchers have claimed that their reinforcement learning algorithm shows generality, most of these claims lack any measure of or support. Some have argued that a task-theory which describes the complexity and difficulty of tasks might give an insight into the systems generality. Before such a task-theory exists, however, a better insights into the chosen methodologies and evaluation of its strengths and weaknesses is key to assess the progress of both, narrow and AGI research. The work presented here proposes a novel evaluation platform SAGE (Simulator for Autonomy & Generality Evaluation) which has two main goals. First, with this platform the gap between general and narrow machine intelligence (GMI and NMI, respectively) can be bridged by providing a single platform on which both, NMI and GMI aspiring systems, can be evaluated. Second, by providing full adjustability of most of the complexity dimensions of task-environments proposed in the literature, different levels of generality can be tested and evaluated, assessing the progress towards ”thinking machines” (Turing, 2009). In this work the requirements of a general evaluation platform have been identified, an insight is given into the implementation strategy of SAGE and how these requirements were met, and first results of evaluating two different deep reinforcement learners are being presented. The importance of parameter isolation is further discussed and the possibilities of SAGE for future application for a broad range of AI systems is presented.

Kurzfassung

Seitdem das Feld der knstlichen Intelligenz (engl. artificial intelligence, AI) das erste mal 1956 so bezeichnet wurde haben unterschiedliche Herangehensweisen in der Forschung in eine Aufspaltung in zwei Zweige gefhrt. ”Eingeschrnkte” knstliche Intelligenz (AI) und generelle knstliche Intelligenz (AGI). Unterschiedliche Ansätze mit unterschiedlichen Definitionen der gleichen oder ähnlicher Begriffe führten in den letzten Jahren zu einer Verstärkung dieser Kluft zwischen den Forschungsfeldern. Ein bestehendes Problem ist, dass die Generalität eines Systems bis heute nicht messbar ist. In der Literatur der letzten Jahre wurde mehrfach behauptet, dass ein neu entwickelter Reinforcement-Learning-Algorithmus Generalität aufweist, allerdings konnte diese Behauptung kaum oder gar nicht mit gezielten Evaluationen belegt werden. Von vielen Wissenschaftlern und Wissenschaftlerinnen wurde argumentiert, dass eine ”Task-Theory” (dt. Aufgaben-Theorie) notwendig ist um die Komplexität und die Schwierigkeit von verschiedenen Aufgaben bestimmen zu können und somit einen Einblick in die Generalität des Systems, welches diese Aufgaben bewältigt, zu erhalten. Da solch eine Theorie bisher nicht beschrieben ist muss vorerst eine andere Möglichkeit für einen Einblick in die Vorteile und Nachteile verschiedener Methodologien entwickelt werden. Solch ein Einblick in die AI Architekturen spielt eine Schlüsselrolle in der Beurteilung des Fortschritts beider Zweige, AI, und AGI, im Bezug auf die Generalität von Systemen. Im Rahmen dieser Arbeit wird eine neuartige Evaluierungsplattform SAGE (Simulator for Autonomy & Generality Evaluation) vorgestellt deren zwei Hauptziele sind zum einen die Kluft zwischen den beiden Zweigen des Forschungsfelds zu überbrücken, indem eine einzige Plattform bereitgestellt wird, welche sowohl AI, als auch AGI Systeme evaluieren kann und zum anderen durch die hohe Anpassbarkeit fast aller bis heute beschriebenen Komplexitätsdimensionen von Task-Environments (dt. Aufgabenumgebungen) eine Evaluierung des Levels der Generalität möglich zu machen. Durch diese beiden Möglichkeiten kann der Fortschritt in Richtung ”denkender Maschinen” (Turing, 2009) ermittelt werden. In dieser Arbeit werden die Anforderungen an eine generelle Evaluierungsplattform identifiziert und ein Einblick in die Implementierungsstrategie von SAGE gegeben. Hierbei wird die Art und Weise, wie die Anforderungen erfüllt wurden beschrieben. Zudem werden erste Evaluierungsergebnisse zweier Deep-Reinforcement-Learner vorgestellt. Anhand dieser Ergebnissen wird die Bedeutung der Parameterisolierung für die Anpassung von Komplexitätdimensionen weiter diskutiert und die Möglichkeiten, welche SAGE sowohl für das Feld von AI, als auch AGI bietet, vorgestellt.

Acknowledgement

I would like to show my gratitude to my two supervisors, Prof. Kristinn R. Thórisson for the ongoing support during my thesis, the constant help whenever needed, and the inclusion into his team from the very beginning, and Prof. Christian Heipke, without whom neither my stay at the CADIA institute at the Reykjavik University, nor this thesis could have been done. I want to thank both of them for going out of their way to help me follow this research path and helping me to smooth the path towards this thesis.

I further want to thank Arash Sheikhlar and Hjrleifur Henriksson Rrbeck for the constant exchange of ideas and the ongoing discussions we had about the importance of the research done and the research possibilities that opened up during my time at the CADIA institute.

Not to forget both, Árni Dagur, and Kaashika Prajaapat with whom I was able to discuss and investigate new ideas in the field of task-theory and AI evaluation, whenever we met.

This project has been made possible in part by a grant from the Cisco University Research Program Fund, an advised fund of Silicon Valley Community Foundation. The work was further supported in part by the Icelandic Institute for Intelligent Machines.

Contents

1 Introduction

In order to assess the progress of any research field an evaluation of novel developments is necessary. Further, in the field of AI, good evaluation methods that allow comparisons between systems and research teams are necessary to guide the research community in evaluating advantages and flaws of different algorithms and methodologies in a variety of scenarios, thus making an informed decision on the usability of different architectures possible. Further more, proper evaluation methods can (a) gauge research progress by measuring the difference in performance of different versions of the same architecture, giving insight into the limitations and the potential of the chosen architecture and additions, modifications, and/ or extensions; (b) lead to better understanding of the chosen methodology in comparison with other AI methodologies by testing different AI architectures on the same task; and (c) give the expert an overview of the possible applications of the AI architecture by testing it on different tasks and in different environments. More good reasons for AI evaluation are described in the literature (e.g. Bieger et al. (2016)) and often focus on evaluating the generality or ”intelligence” of a system. To date, many evaluation methods for general intelligence aim towards testing either exclusively human intelligence, like the IQ test, or at least human-level intelligence, like the Winograd's Schema Challenge (Levesque et al., 2012), the Lovelace Test 2.0 (Riedl, 2014), or the Toy Box Problem (Johnston, 2010). The purpose of general intelligence evaluation, however, is not only to evaluate if the system shows a general intelligence, but also at which level it does so. Other evaluation strategies are often very domain specific, like the general game playing evaluation (Świechowski et al., 2015) which focuses only on video games, or tests like the Wozniak Coffee Test or the Turing Test (Oppy and Dowe, 2019) which primarily focus on human social conventions, skills, and experiences. While artificial general intelligence (AGI) aims towards an intelligence comparable to the of humans, the term intelligence by itself does not include human-like knowledge, experience, skills, or social conventions. For an evaluation of general intelligence a definition of generality and intelligence is therefore needed in order to do proper evaluation of the level of generality of a system.

To define a concept like ”intelligence” has proven to be a difficult task by itself. More than 70 different definitions exist at to date and are strongly debated (Legg et al., 2007; Wang, 2007). This debate about general intelligence might have its source in the different understanding of ”generality” between fields and traditions of research. Whilst in mathematics something ”more general” can be applied to more, different, problems, in physics a ”more general” model means one which describes the observed entity better or more broadly than a less general model. When talking about general machine intelligence (GMI) in comparison to narrow machine intelligence (NMI) the focus should be put on the second, physical, description of generality. Therefore the more general an agent is, the better it can generalise its knowledge in order to describe the task and environment in a way which represents causal relations, which in turn allow it to act more intelligently in the world. Additionally, the autonomy of the agent is of crucial interest. Intelligence can emerge, if the system can use the identified causal relations to achieve the goal (or multiple goals) of a task (or multiple tasks). Therefore, important aspects of a system's intelligence can be seen as the autonomy and the generality of the agent, or differently put: The system's capability to identify causal relations (generality) and exploit them in order to achieve a goal (autonomy). Additionally, Pei Wang's definition of intelligence Wang (2019) emphasizes an assumption of insufficient knowledge and resources that any intelligence must take into account.

To evaluate specific internal workings of different AI architectures calls for novel evaluation platforms. Current AI evaluation approaches test for performance scores of single tasks without changes in the task or environment during runtime or a multitude of tasks with similar constraints, like different video games. Further, most evaluation platforms do not include stochasticity of environment or task variables which can lead to wrong conclusions about the necessary sophistication or intelligence of AI architectures in order to solve the task (see e.g. Bellemare et al. (2015) for necessity of stochasticity control). Lastly, it has been shown that evaluation strategies like the ALE (Bellemare et al., 2013) do not necessarily give an insight about the progress of AI, but rather only make a comparison of architectures (Hernández-Orallo et al., 2017) without informing about their level of generality, intelligence potential, or learning abilities. To evaluate a system on generality, intelligence, and learning abilities, an *adaptable evaluation platform* is called for which makes isolation of single complexity changes, like stochasticity or task changes, possible.

To date, many evaluation methods evaluate a system's capabilities using a single performance measure, where a series of (relative) measurements would be better suited to separate between GMI and NMI. With the "Simulator for Autonomy & Generality Evaluation" (SAGE) a new platform for AI evaluation is introduced, which aims towards bridging the gap between NMI and GMI research. Further, with a platform like SAGE learning methods and knowledge generalisation of AI systems can be evaluated showing flaws and advantages of different approaches of AI methodologies. SAGE is a simulation tool for task environments based on the idea to break the tasks and the environment into variables and transition functions. Variables include observables, unobservables (hidden), manipulatables, and non-manipulatables. Transition functions represent the change of those variables over time and/ or by interaction of the system with the environment (Thórisson et al., 2015, 2016). Task-environments can be constructed with different, independent characteristics and levels of complexity. This includes causal relations, statistical relations, determinism or stochastic behaviours, hidden variables and observable variables, distracting variables, noise models, dynamic or static environments, task changes and much more. This way, by increasing the levels of complexity on different dimensions, it is possible to bridge the gap between NMI and GMI. While NMIs might start at the lower end of complexity levels they can be increased and/ or changed periodically in order to find the systems moment of failure or situations in which the system shows unwanted behaviour. From this conclusions can be drawn about the lacking abilities which would be necessary in order to solve such tasks with higher complexity.

The design of SAGE is based on a model-view-controller-agent (MVC-A) paradigm. The MVC (model-view-controller) paradigm is well known in computer science for creating simulation systems, including those that connect with the physical world in realtime. For evaluating GMI systems it is important to have flexible control over how time progresses. This model is extended for the purpose of evaluating a broad range of AI systems with an additional conceptual component for the AI agent(s) ("A" for "agent") that interact with a task-environment. By dividing these parts into sub-processes it is possible to adjust parameters in the controller, model, and agent independent of the others making isolated complexity changes possible. The controller simulates the environment and transitions it between time-frames in response to the agents actions or the environments internal dynamics, making adjustments in the internal workings of the task-environment possible. The model provides adjustability options for observable data, actions, or the hiding (exposing) of data from (to) the agent. The agent can be any machine learning algorithm which is capable of processing the observable data provided by the model. Besides making independent adjustments of complexity dimensions possible the division into own processes makes physical division of model, controller, and agent possible and therefore gives the opportunity to evaluate the resource management of the agent without interference from the evaluation platform. This is especially important in the concept of GMI (insufficient resources).

In this thesis the focus is not on the methodologies of AI architectures or their specific advantages, but rather on the possibilities the SAGE platform offers for the research community in order to test their methodologies and use evaluation results to assess their progress towards GMI. For this the requirements and possibilities of a NMI-GMI bridging evaluation platform are introduced and an implementation with first results of NMI evaluation presented. An actor-critic and a double-deep-Q-learner are used to demonstrate some of the features of the SAGE platform and to validate its design. At the time of writing of this thesis a paper was submitted to the AGI

Conference 2020 with the same topic (Eberding et al., 2020), based on the work described here. This thesis presents more in-depth research research results and describes SAGE in more detail.

The thesis is structured as follows: In section two a brief overview of the term "artificial intelligence" and the differences between "narrow" and "general" AI is shown before introducing the importance of a task-theory for AI evaluation and continuing to a short description of current methodologies of reinforcement learning algorithms (RL). Then a description of current approaches of AI/ RL evaluation is given and the necessity for a new evaluation methodology provided. In section three the requirements for such a novel evaluation strategy are introduced. Further an implementation strategy is formed to realise an evaluation platform which is able to bridge the gap between NMI and GMI by making an evaluation of both possible and assessing the progress towards GMI. In section four SAGE's implementation is introduced going into detail of implementation of the requirement profile introduced in section three. Section five shows first results, as a proof of concept, of evaluating current RL algorithms before concluding the work in section six, discussing the first results and their importance for the AI research community, and giving an outlook on future work on and with SAGE.

2 Related Work

2.1 Artificial Intelligence

There exists no clear definition of intelligence to this date even though intelligence and its definition has been subject of debates for decades (Wang, 2007). There exist at least 71 different working definitions on intelligence at the moment of writing, 18 of those alone in the field of AI research (Legg et al., 2007). Even looking back at the founders of the modern AI different opinions about what the essence of intelligence and therefore artificial intelligence is and how it should be defined can be found (McCarthy, 1988; Minsky, 1988; Newell and Simon, 1976; Wang, 2019). This leads to confusion amongst AI researchers resulting in different approaches towards AI, all under the same term: "intelligence". By the systems behaviour and their main focus it is possible to divide those approaches into three main categories. Systems which behave like the human mind, Systems which can solve problems previously only solvable by the human mind, and Systems which show the same or similar functions to those of the human mind (Wang et al., 2018).

Systems which behave like the human mind are sought to have a similar structure as the human brain. The research in this approach of artificial intelligence assumes that any intelligence has a foundation in the biological neural network of the brain. This approach is closely related to neuroscience and rises and falls with the progress of neuroscience itself (Wang, 2019).

The second category only focuses on the problem solving capabilities of the AI architecture. Neuroscience does not play a role in this branch of AI research, neither does psychology or cognitive science. This is the currently most common approach to create artificial intelligence and includes neural networks, deep learning, game playing, and most current publications in AI (Wang et al., 2018) (See as examples (LeCun et al., 2015; Luger, 2005; Russell and Norvig, 2016)).

Researchers aiming towards the third category on the other hand neither build their research in neuroscience, but rather in the study of behaviour of intelligent biological agents, nor do they only aim towards problem solving capabilities in AI architectures. This includes for example the well known Turing test (Oppy and Dowe, 2019), the "Wozniak coffee test" suggested by the Apple co-founder Steve Wozniak in which the system needs to go into an unknown house and make coffee, or the "graduation test" proposed by Goertzel et al. (2012) in which the system has to graduate (virtual-world or robotic) preschool. None of these tests include a necessity to model the human brain or base the assumptions on biological foundations but rather see the behaviour of the system as a measurement of intelligence.

Another way to categorise AI architectures is by the similarity to the human mind. It is similar to the behavioural categorisation but gives a better insight into how intelligence is thought to emerge from a system. The similarity categories include structure, performance, capacity, function, and principle (Wang, 2007).

In recent years this differentiation between the opinions of what a intelligent system needs in order to be seen as ”human-like” has lead to the introduction of a new term, ”AGI” (Pennachin and Goertzel, 2007; Wang et al., 2018). Differentiating between ”narrow” or ”weak” AI and ”general” or ”strong” AI divides the field of AI research between the second and third categories. A general intelligence must not only solve problems, but rather show a human-like behaviour and most importantly is not constructed to solve a single problem, but rather autonomously solve a variety of different tasks in a variety of different environments (Pennachin and Goertzel, 2007). The category of an artificial intelligence founded in the biological properties of the human mind would be part of the ”general” intelligence research (Wang et al., 2018) and is therefore not further differentiated. From this division of the term ”intelligence” follows a further division amongst the AI research community. In order to accelerate progress in the field of AI this gap should be bridged. This would allow researchers from both communities to compare their architectures and isolate flaws and advantages of both approaches and use strategies from both fields in order to achieve the goal of this research as described in the early days of AI (cf. McCarthy et al. (2006); Turing (2009)).

2.1.1 ”Narrow” AI

”Narrow” AI has been subject of many studies in the past years. It achieved impressive results in the past decades (Wang et al., 2018) including Deep Blue beating the chess world champion in 1996 (Campbell et al., 2002), AlphaGo (Silver et al., 2016) or the general advance in deep neural networks (LeCun et al., 2015). With the introduction of neural networks in RL it was possible to develop learners for continuous state spaces. RL algorithms were soon available which could play Atari 2600 games with a super-human performance (Mnih et al., 2015). This lead to the belief that deep neural networks are a first step towards a general intelligence, as can be seen by the title of the Atari 2600 evaluation platform ”The Arcade Learning Environment: An Evaluation Platform for General Agents” (Bellemare et al., 2013). However these learners are built on the assumptions, that intelligence is the capability of a system to solve problems, rather than to create general models of the environment. Thus they are mainly focusing on the strength of the architecture in its designated tasks. The current approach to create ”stronger” weak AI systems does not necessarily lead towards a general intelligence since generality does not only include the strength in one particular field/ task, but rather the breath of applicability of the system in a variety of tasks and domains without human interference (Goertzel et al., 2012; Wang et al., 2018). ”Narrow” AI can be seen as a part of computer science without many connections to cognitive science, psychology or neurobiology.

2.1.2 General AI

"General" AI has been the origin of research in artificial intelligence. A general AI is what Turing had in mind, when he wrote his paper on "thinking machines" (Turing, 2009), or McCarthy's research proposal in 1955 (McCarthy et al., 2006). Those attempts, however, turned out to be fruitless leading the majority of AI researchers into a more promising direction with limitation on single problems or single cognitive functions. This lead towards an understanding of intelligence as a single competency rather than a collection of of related capabilities (Wang et al., 2018). In contrast to this exists the field of AGI, aiming towards a human-like intelligence which is able to solve a variety of tasks in a variety of environments and domains (Adams et al., 2012; Goertzel et al., 2012).

Another main difference between general and narrow AI systems is their relationship to other sciences. While narrow AI can be mainly seen as a part of computer science, AGI leans towards cognitive science, neurobiology and psychology to better understand the human mind (cf. Goertzel (2009); Goertzel et al. (2012); Wang (2007)).

To identify what needs to be achieved in artificial general intelligence the underlying structures of the human mind have been analysed. Cognitive science has discovered different structures and dynamics describing those underlying principles (Goertzel et al., 2012). By designing architecture diagrams of human intelligence many researchers in the field of cognitive science have helped to create a better understanding of the human mind (cf. Baars and Franklin (2009); Sloman (2001)). These architecture diagrams of the human mind give the AGI researchers a framework and context with which their architecture can be compared. Thus cognitive architecture requirements could be described for an AGI system which need to be fulfilled in order to claim generality (Goertzel et al., 2012).

The abilities of AGI architectures in comparison of narrow AI architectures are fundamentally different as well. Due to the nature of narrow AI as a problem solving system the architecture can be designed to solve this problem without respect to different environments or tasks. Therefore the design is constructed by humans and tuned to behave in a certain way in expected situation. They mainly rely on integration of single competencies until the task can be reliably solved by the system itself (Thórisson, 2012). AGI architectures on the other hand aim towards a goal of autonomous, cumulative, life-long learners which can adapt to their surrounding as necessary and achieve not only selected goals but rather any solvable task it is presented with (Goertzel et al., 2012; Thórisson et al., 2019). Any AGI system always underlies the assumption of insufficient knowledge and resources, meaning, that the systems always works under restrictions. Those restrictions include finitness of its information processing capabilities, real-time and incompleteness of their knowledge (Wang, 2007).

Cumulative learning is a central aspect of any autonomous AGI system. Due to the assumption of insufficient knowledge and resources and the necessity of an AGI system to constantly adapt to its surrounding it is possible to define different aspects or dimensions of cumulative learning to measure a systems performance as a cumulative learner. These include the systems memory management, temporal capacity and granularity and generality of the learning/ generalisation of knowledge (Thórisson et al., 2019).

Transfer learning General intelligent systems must be able to deal with novel situations during their lifetime. This novelty is in relation to the learner's knowledge of the task-environment it was deployed in before. A possibility to handle novelty can be to use previously acquired experience that seem familiar as described by Sheikhlar et al. (2020). This is well described in psychology by the canonical concept of transfer of learning (Kaptelinin and Nardi, 2006). Transfer learning is described in various machine learning theories to date. The main goal is to increase the systems learning rate by simultaneously increasing its flexibility. In deep neural networks for example a scheme was implemented, in which a human programmer selects certain layers of a trained network and reuses them in order to train another network in a related domain. Transfer learning in reinforcement learning was used in order to train the system on one task and repurposing it in a similar one (Taylor and Stone, 2009; Xie et al., 2016; Yosinski et al., 2014). This however needs human interaction for choosing the tasks and making analogies, In GMI an autonomous form of transfer learning is necessary in order to deal with the complexity of task-environments that can emerge over the life-time of a learner (Sheikhlar et al., 2020).

2.2 Task Theory, Environments, and Agents

An AGI system must be able to cope with a variety of tasks in a variety of environments (Goertzel et al., 2012). While for example P. Wang's theory of intelligence (Wang, 2019) takes a stance in isolating key properties of intelligence, and attempts to explain it in all aspects, such a definition is still a work in progress for tasks and environments. This makes any claim, that any AI architecture can cope with a variety of tasks dependent on the mainly ad-hoc and case specific tasks and environment selection of the expert (Thórisson et al., 2016). To assess the progress towards autonomous generality a description of tasks, environments, and their interconnection, task-environments, is needed, as well as of their complexity and similarity. This could be done in a similar fashion, as the definition of intelligence, by isolating key properties of tasks and environments in order to describe tasks in all aspects. Thórisson et al. (2015) proposes such a description by dividing task-environments into complexity dimensions. By Sheikhlar et al. (2020) something similar was done for the description of task-environment similarity. However, both the description of task complexity, as well as a description of the similarity of tasks still needs to be confirmed. Without such theories and definitions crucial aspects of general intelligence (e.g. cumulative learning, life-long learning, or transfer of learning) cannot be assessed, making a validation of the progress dependent on experts opinions instead of objective evaluation methodologies (Bieger et al., 2016; Thórisson et al., 2015, 2016). A confirmation of the descriptions of Sheikhlar et al. (2020) and Thórisson et al. (2015) could for example be done by integrating their key properties into an evaluation platform and making them adjustable by the examiner.

2.2.1 Tasks and Environments

Tasks describe the transformation of the environments state into a defined goal state while avoiding (defined) failure states. This transition towards (un-)desired states can be achieved by an agent interacting continuously or intermittent with the environment. Any instance of a task includes the environment which is to be transformed by the agent towards the described goal state. Therefore the term ”task-environment” is used to describe any instance of a task in an environment (Thórisson et al., 2015, 2016). Task-environments have certain properties which define their complexity and describe their similarity to other tasks (Thórisson et al., 2015). Those properties are essential to define test and evaluation environments to evaluate any agent on different difficulties of tasks, as well as their ability to transfer learning and accumulate knowledge over time when certain task similarities can be exploited. Due to the different approaches of AI research tasks and their environments are of high importance in order to evaluate the generality and autonomy of any AI system. As argued by many (Adams et al., 2012; Hernández-Orallo et al., 2017; Thórisson et al., 2015) a more flexible tool is needed, which allows construction of appropriate task-environments. Thórisson et al. (2015) list 11 dimensions that ideally should be controllable by a creator of a task-environment for measuring intelligent behaviour; Russell and Norvig (2016) present a comparable subset of seven dimensions. The environment can be categorised along different dimensions, namely determinism (cf. Bellemare et al. (2015) regarding the importance of noise control), staticism, observability, agency, knowledge, episodicity, and discreteness. TE properties include next to the seven environment properties ergodicity, asynchronicity, controllability, number of parallel causal chains, and periodicity (Russell and Norvig, 2016; Thórisson et al., 2015).

These properties should be adjustable by the evaluator in order to test a systems capabilities of solving more complex tasks. Reflexively these evaluations of agents can provide a better insight into difficulty and similarity of seemingly unrelated tasks. In order to examine tasks, environments, or agents it is necessary to isolate the aspect under inspection from the other two. By for example changing the task independent of the environment and the agent its impact on the agents performance can be measured and therefore might lead to conclusions regarding the theory of tasks.

Tasks Some effort has been made to describe the difficulty of tasks in Levi's Search, but they only use computational complexity for the description of tasks (Hernández-Orallo, 2015). However, to describe tasks in a way to use in the evaluation of generality more than computational complexity is necessary. By introducing transfer of learning for example, previously gathered knowledge and its applicability must be included in the task description.

Any ”narrow” intelligence is by definition designed to address only the tasks it has been designed for, enabling the examiner to evaluate the system's performance on these specified tasks. This assumption does not hold for any ”general” intelligence. Without having information about the tasks the AGI architecture has been tested on, the examiner can neither make any statements about the generality of the system nor is any prediction of success in unknown tasks possible. To evaluate a general intelligence a task theory is necessary which describes not only the difficulty of tasks but rather their complexity in different, task-environment related, dimensions. Such a task theory does not exist at the moment, however the for AGI evaluation necessary parameters can be described (Bieger et al., 2016; Thórisson et al., 2015, 2016). Until such a task-theory exists the evaluation should target most or all task-environments properties. This in return might provide an insight into task-environments and their descriptive parameters for better understanding of complexities and difficulties of different task-environments.

Environment The environment creates boundaries for the task. Only by defining states in the environment a task becomes (un-)solvable. For a description of the environment a discrimination between controller, agent, body and environment is done. The agent is divided into the controller and the body. The controller includes the AI architecture while the body represents the embodiment in which the controller is placed in order to perform actions on the environment. The body is therefore the intersection of environment and agent (Poole and Mackworth, 2010). There exist different arguments in favour of including the body into the environment. When for example the body of an agent is moved in its environment the environments state of ”position of robot” is changed accordingly (Russell and Norvig, 2016). The body includes the sensors of the system and passes them to the agent in the form of percepts. The same with actuators which are used in order to act on the environment and commands telling the actuators how to move (Poole and Mackworth, 2010). This point of view is in accordance with psychology and therefore chosen in order to describe the parameters of the embodiment as part of the environment (Pomerantz, 2006).

2.2.2 Agents

As described by Poole and Mackworth (2010) agents are systems, which act in an environment. Agents can have different forms, designs, and architectures. Examples for agents in general are persons, robots, animals, computer programs or even corporations. An agent consists of the body and the controller. If the body is a physical body it is called the embodiment of the agent (e.g. robots) Poole and Mackworth (2010). Agents can receive measurements of the environment through sensors which are part of the body of the robot. Such stimuli are forwarded to the controller as percepts. On the other hand, agents can act on the environment with actuators (or effectors). The actuators are part of the body and the controller can interact with them using commands. Both stimuli and actions underlie environment properties like noise, unreliability, and speed. Percepts and commands on the other hand are part of the agents internal controller and therefore underlie the controllers capabilities like memory, energy consumption, and calculation speed (Poole and Mackworth, 2010). For example the stimuli for an agent equipped with a camera is the light hitting the cameras sensor. This light is converted into pixel information of an image, this image is an example for a percept passed to the controller. Percepts can consist of higher-level features like lines, edges, clusters etc (Poole and Mackworth, 2010). An example for an action is the acceleration of a robot. The action is created by passing a command to a motor controller which in turn adjusts the voltage for the motor. This voltage adjustment and the final acceleration can be noisy or include slippage of the wheel, while the command itself does not. When talking about NMI or GMI systems the controller of the agent is the main focus of attention (Poole and Mackworth, 2010).

2.3 Reinforcement Learning

Current approaches to create an AI system which can autonomously solve tasks in arbitrary environments focus strongly on reinforcement learning (RL) strategies. A significant achievement in this field came with the introduction of deep neural networks in reinforcement learning, making it a successful branch of AI-research (Mnih et al., 2015). RL is a branch of AI which aims towards the development of goal-driven learning and decision-making. This approach is well described in literature, for example by Sutton and Barto (2018) from which the following chapter is derived. The idea of reinforcement learning is based on the assumption, that any intelligent system can learn by interacting with its environment. It is distinguished from other AI branches like supervised or unsupervised learning by (a) the goal-driven nature of RL algorithms which lead to a long term interaction with the environment, (b) the absence of an expert telling the agent about the desirability of states, and (c) the absence of complete models of the environment. The AI system generates a map of state-action combinations by applying reinforcement learning strategies to task-solving problems. With any interaction with the environment a reward signal is sent to the agent giving it information about the desirability of the newly reached state. This way a policy can be derived by the learning system by assigning values to state-action pairs. As described in Sutton and Barto (2018), reinforcement learning algorithms can be divided into four sub-elements: policy, reward, value function and optionally a model of the environment:

Policy The policy is the core of a reinforcement learner. It is the mapping of the current state to the best next action. The policy can be any mapping function, from simple look-up tables to sophisticated search algorithms or other expensive calculations. In deep reinforcement learning the policy can be learned by deep neural networks (DNN) which map the current state to the best action in regards of the highest achievable reward (Sutton and Barto, 2018).

Reward The reward signal is the information about the desirability of a state. In each new state the agent receives such a reward signal as a single number which gives the agent information about whether the newly achieved state is good or bad. The agents sole goal is to maximise the amount of gathered reward over a complete (long) run. Rewards are the main reason for changes of policy. If after following the policy a better (worse) state is reached the policy of choosing the same action in the previous state the next time it is reached may be supported (opposed). Thus the chance of choosing this action is increased (decreased) (Sutton and Barto, 2018).

Value Function While rewards show immediate results, the value function includes information about the long run. It includes the accumulated reward possible to achieve following the current (or next) state. Therefore the value function is the agents possibility to overcome local reward maxima and aim for global reward maximisation (Sutton and Barto, 2018). In order to achieve the global maximum it must first be discovered to include it into the value function. This leads to the well described exploration-exploitation dilemma (cf. Ishii et al. (2002); Thrun (1992)).

Model The model makes inferences of the environment possible by representing the environment transitions. It is used for planning by considering future states without having to experience them first and therefore help maximising the reward over time. RL methods which include a model are called model-based methods, other, simpler, RL methods which rely on trial-and-error approaches are called model-free methods (Sutton and Barto, 2018).

2.3.1 The Reinforcement-Learning Problem

Sutton and Barto (2018) describe the reinforcement-learning problem as the interaction of the agent with the environment. In each time-step $t = t_1, t_2, t_3, ...$ the agent receives information about the environment's (including the agent's) current state $s_t \in S$ where S is the set of all possible states, chooses an action $a_t \in A(s_t)$ with $A(s_t)$ being the set of possible actions in state s_t and thereby transitions the environment into a new state s_{t+1}. With the transition to the new state the agent receives a reward signal $r_t \in R \subset \mathbb{R}$. From this reward and transition the agent can create a mapping of the desirability of newly encountered states to previously chosen actions. This policy is denoted as $\pi_t = \pi_t(a|s)$ and describes the probability to choose action a when state s is the current state. RL methods define the change of the policy in accordance to experienced rewards in new states.

In order to maximise the rewards over the full run, rather than just a single state, the expected return G is defined for example as

$$G_t = R_{t+1} + R_{t+2} + ... + R_{t+n}. \tag{1}$$

However, to include both finite and infinite/ continuing tasks (open ended) a discounted return is introduced:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k \cdot R_{t+k+1} \tag{2}$$

γ is the discount factor $0 < \gamma < 1$ and defines the importance of later rewards for current actions. A low γ corresponds to maximisation of immediate rewards, a high γ results in a stronger representation of future rewards in the accumulated return. Additionally, by introducing this discount factor the return is upper bounded, as long as $\gamma < 1$ and the rewards sequence $\{R_k\}$ is bounded (Sutton and Barto, 2018).

To maximise this accumulated/ discounted return is the main objective of any reinforcement learner. The decisions leading to this maximal return is represented by the policy of the agent. Since the policy represents a mapping of states to actions certain assumptions must be made about the kind of states received by the agent in order to make such a policy generation possible. One of them is the assumption of states having the Markov-Property and therefore the whole task being a Markov-Decision-Process.

2.3.2 Markov-Decision-Processes

As Sutton and Barto (2018) describe in the book ”Reinforcement Learning: An Introduction” a Markov-Decision-Process (MDP) is a (reinforcement learning) task, which satisfies the Markov-Property. The Markov property is a property of the state signal passed to the agent and informally can be defined as the property, that any given state s_t of a task includes all relevant information of the history of all states leading to the current state which are necessary in order to make a new decision. This does not mean, that all past states are combined within the current state signal, but rather, that all *relevant* information from the past states is available (Sutton and Barto, 2018). For example in physics a ball's position, velocity and acceleration (s_k) describe the future path of the ball perfectly without including information about where it came from. Therefore the past states (position, velocity, and acceleration at time $t_{0...k-1}$) are not included in the state signal, but all relevant information for predicting future states is included. The same holds for any moment in a chess play. While the previous moves cannot be reconstructed from the current board all necessary information for future moves is included in the current state of the board. This Markov-Property can be mathematically defined. When looking at the most general case, in which the environment at $t + 1$ is dependent on all previous states, actions, and rewards the probability of achieving a rewards r_{t+1} and transitioning the state of the environment into s_{t+1} is

$$Pr(r_{t+1} = r, s_{t+1} = s|s_0, a_0, r_1, s_1, a_1, r_2, ..., s_{t-1}, a_{t-1}, r_t, s_t, a_t). \tag{3}$$

When the reinforcement task is a Markov-Decision-Process, however, the Markov-Property defines, that

$$Pr(r_{t+1} = r, s_{t+1} = s|s_t, a_t). \tag{4}$$

From this it follows, that a process is a Markov-Decision-Process, if for all states S (3) equals (4) (Sutton and Barto, 2018).

Value Function in MDPs As described previously, the policy of an agent π is the mapping from a state s and an action a to the probability $\pi(a|s)$ of choosing action a, when in state s. The value $v_\pi(s)$ is the expected return when in a state s and following the policy π (Sutton and Barto, 2018). It can be described as:

$$v_\pi(s) = E[G_t|s_t = s] = E_\pi[\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1}|s_t = s] \tag{5}$$

The action-value function q is similar to the value function, but does not only depend on the state s, but also on the chosen action a:

$$q_\pi(s, a) = E[G_t|s_t = s, a_t = a] = E_\pi[\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k+1}|s_t = s, a_t = a] \tag{6}$$

From the value function the Bellman equation for v_π can be derived:

$$v_\pi(s) = E[G_t|s_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a) \left[r + v_\pi(s')\right] \tag{7}$$

This Bellman equation can be learned/ solved by experience by adjusting the (action-)value function of a past state s when the reward and new state r, s' is observed by the agent.

When talking about the maximisation of the expected result the main objective is the maximisation of the value function. This leads to the optimal value-function and action-value function $v_*(s)$ and $q_*(s, a)$, respectively.

$$v_*(s) = \max_\pi v_\pi(s), \qquad q_*(s, a) = \max_\pi q_\pi(s, a) \tag{8}$$

These optimal value-function(s) must not be a single function, but can be multiple functions, however, all optimal policies are meant to be included in $v_*(s)$. All v_* share the same optimal action-value function q_* (Sutton and Barto, 2018).

$$q_*(s, a) = E[r_{t+1} + \gamma v_*(s_{t+1})|s_t = s, a_t = a] \tag{9}$$

From this follows for the Bellamn equation of $v_*(s)$:

$$v_*(s) = \max_{a \in A(s)} q_{\pi_*}(s, a) = \max_{a \in A(s)} \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_*(s')\right] \tag{10}$$

The Bellman equation for the optimal value-function is finite for finite MDPs and can be solved in a variety of ways. If the optimal (action-)value-function is known the optimal policy can be derived by a one step search. The action leading to the best expected return of v_* is the best possible action in state s (Sutton and Barto, 2018). In high space dimensional continuous tasks the calculation of every state action pair is unfeasible. This problem was overcome by introducing deep neural networks (DNNs) to reinforcement learning Li (2017). The actor-critc and the double-deep-q learner are two examples of such deep reinforcement learners using DNNs in order to estimate the optimal policy or optimal aciton-value function, respectively.

2.3.3 Actor-Critic

Two RL algorithms are introduced which were used to test the newly developed evaluation platform SAGE. The actor-critic deep reinforcement learning algorithm (AC) (Konda and Tsitsiklis, 2000) is an On-Policy learner, which uses DNNs to maximise the return of a task. Its advantage in low-variance optimal policy search

has been a reason for it to be used in many real life applications (Grondman et al., 2012).
The AC consists of two parts, the actor, and the critic. The actor represents the policy and the critic the value function. The critic is called *critic*, because it criticises the actor's action and adjusts the value function to the reward and new state, whereas the actor uses information of the current state and its policy in order to decide on a new action. After each action the critic evaluates, if the action lead to better or worse results than expected resulting in the temporal difference (TD) error δ_t

$$\delta_t = r_{t+1} + \gamma v_t(s_{t+1} - v(s_t)). \tag{11}$$

A positive δ_t means a more successful return, a negative δ_t a worse than expected return. This can be used accordingly to adjust the actors policy increasing or reducing probabilities of the chosen action. The TD error is further used to update the critic's value function. Therefore the TD error δ_t drives all learning of the actor-critic reinforcement learner (Grondman et al., 2012).

2.3.4 Double-Deep-Q

In deep-Q-learning the action-value function is directly approximated, rather than searching for the best policy. The policy is derived over the Q-function and evaluated using TD methods.

$$Q(s, a) = Q(s, a) + \alpha[r_t + \gamma \cdot \max_{a'} Q(s', a') - Q(s, a)] \tag{12}$$

with the learning rate α, the reward r_t, and the discount factor γ. The derived policy corresponds to choosing the action with the highest Q-value in a state s. The Q-values are arbitrarily (e.g. zeros) initialised and converge via sampling of the environment by interaction.
Due to taking the maximum Q-value of the future state it can lead to systematic overestimation and therefore it can introduce a maximisation bias in learning. To avoid this a second neural network, the target network, is introduced as a q-value evaluator leading to a double-deep-q learner (Van Hasselt et al., 2016). The first network is responsible for choosing the action and the second for action evaluation. The target network is updated with a update-factor τ (rate of averaging) periodically by the computed networks weights of the source network. This is done by copying the weights multiplied by the update-factor of the source model to the target model (Van Hasselt et al., 2016). The double-deep-Q-learner is an Off-Policy learner by directly approximating the action-value function and choosing its policy by the Q-value of actions.

2.4 AI Evaluation

Evaluation of AI systems is key to better understand the strengths and weaknesses of different AI architectures. A progress towards an AGI architecture can only be assessed by evaluating the performance and cognitive abilities of different AI systems (Thórisson et al., 2015). Evaluation methodologies to evaluate ”narrow” AI systems are well described. Most commonly performance and abilities are described by a single value. While this makes comparison between different architectures easy, a single value cannot provide information about the cognitive abilities of the system. For example important aspects of intelligence, like life-long, cumulative learning or transfer of learning cannot be assessed using such a simple performance measurement (Thórisson et al., 2015). Another possibility of AI evaluation are well described tests, like the ”Turing-Test” (Oppy and Dowe, 2019), the ”Wozniak-Coffee-Test” (Goertzel et al., 2014), or other anthropological centred tests (cf. Goertzel et al. (2014)). These test have some main disadvantages. First, any test designed by humans to evaluate human-like intelligence is by definition anthropomorphic, meaning that for example the Turing-Test might rather test humanness, than intelligence (Legg et al., 2005). Secondly, these sort of tests all require the agent to be equipped with certain sensors and actuators. They might therefore test the usage of sensor-actuator combinations, but fail to describe the cognitive abilities of the system (Pennachin and Goertzel, 2007). Thirdly any of these tests show the same problems, which follows AI research since decades. Rather than testing for intelligence they test for problem solving of a single task and therefore can be described and implemented by the designer of the system. Any evaluation which aims towards evaluating autonomous generality must therefore test the agents abilities on different tasks and environments, rather than the same task, like making coffee.

There exist many different evaluation platforms for AI architectures and reinforcement learners. Evaluation platforms generally provide a task-environment in which the agent is placed. The evaluation platform exposes the agent to its current state by providing observations and rewards for the agent to process and accepts actions from the agent in return. Different approaches can be seen to generate such task-environments (cf. Hernández-Orallo et al. (2017)). Among others:

OpenAI Gym provides different task environments for evaluating AI systems. The available task environments include classic control problems, like the inverse pendulum, toy texts, 2D computer games (similar to Atari games), and 3D robot simulations, among others. Due to its open-source interface more task-environments have been implemented, like a Gazebo simulation of physical robots (Zamora et al., 2016). OpenAI Gym mainly aims to evaluate learning performance of reinforcement learners on (Partially-Observable-)Markov-Decision-Processes (Brockman et al., 2016; Hernández-Orallo et al., 2017).

The *Malmo* project by Microsoft uses video game emulation for AI evaluation. The evaluator can generate different tasks in a Minecraft world for the AI system to solve. Those tasks include navigation, survival, collaboration and problem solving. The Malmo platform can support research in robotics, computer vision, reinforcement learning, planning, multi-agent systems, and related areas (Johnson et al., 2016).

The *arcade learning environment* (ALE) uses Atari 2600 games for evaluation of reinforcement learners. Systems can have access to the screen and/ or the memory of the Atari emulator. According to the developers of this evaluation platform it is possible to evaluate not only the problem solving capabilities, but also model learning, model-based learning, transfer learning and imitation learning. The evaluation platform consists of 55 different Atari 2600 games. The observation space is 160 times 210 pixels for the screen and 128 byte for the RAM. Additionally 18 different actions are available for the agent to choose. A limitation of the ALE is the number of games, as well as the deterministic environment of all games, leading to overspecialisation (Bellemare et al., 2013; Hernández-Orallo et al., 2017). The authors of the original paper showed in 2015, that a rather simple tree learning algorithm called ”Brute” can solve some of the arcade games better than state-of-the-art machine learning algorithms, showing the importance of stochasticity in task-environments (Bellemare et al., 2015). They therefore later introduced so called ”sticky-actions” to break the deterministic property of the ALE (Machado et al., 2018).

Another approach using game playing is the *DeepMind Lab*. 3D games with first person perspective are chosen to emulate the task-environment. The research surrounding DeepMind Lab is aimed at 3D reconstruction of raw pixel data, fine motor dexterity, planning, navigation, strategy, and autonomous learning on what task to perform by exploring the task-environments (Beattie et al., 2016).

Other evaluation platforms include ViZDoom (Kempka et al., 2016), Facebooks Torch-Craft (Synnaeve et al., 2016), Facebooks CommAI-env (Mikolov et al., 2016), OpenAI Universe[1], and AI2-THOR (Kolve et al., 2017).

Many evaluation methods have focused on (general) game playing using the ability to play games as an indicator for the systems sophistication. Using psychometric evaluation like item response theory (IRT) it was shown that the difference of per-

[1] https://github.com/openai/universe last visited 7th of March 2020

formance score between different ML techniques does not necessarily correlate with the systems level of abilities. Thus a simple performance rating like achieved game score cannot describe the progress of AI by itself (Hernández-Orallo et al., 2017). By evaluating the ability to handle TE property changes over different learners a conclusion can be drawn on the abilities of the learner in regards to autonomous generality. Such conclusions should be accompanied by evaluation strategies like IRT to show the significance of the progress. By isolating and adjusting single parameters of the TE and testing on different learners it is furthermore possible to describe task difficulties in relation to the properties of the TE.

In accordance with many working definitions of A(G)I, an evaluation platform should provide the possibility to introduce novelty at any moment of the evaluation process. Such novelties could include for example unknown states, unknown transition functions, or unknown mechanics influencing the outcome of known state-action combinations. Without these novelties a proper evaluation of the learning – and thus a systems autonomous generality – cannot be performed (Bieger et al., 2016). Such changes in the task-environment can provide evidence of a systems pragmatic understanding of the causal relations between factors in the environment, arguably an important aspect of any GMI (cf. Thórisson et al. (2016)). This should be accompanied by a proper task theory, that enables the comparison of a variety of tasks and environments in order to draw conclusions on the systems performance on a wide range of tasks and environments, one of the prerequisites of AGI. While some of the presented evaluation platforms provide different sorts of novelty, like changing the played game or introducing changes in the environment in the Minecraft world, none of them gives the examiner the ability to change any complexity dimension during runtime. For example hiding variables, stochasticity, or dynamics changes cannot be done without changing the source code of the evaluation platform.

3 SAGE – Simulator for Autonomy & Generality Evaluation

The evaluation platform SAGE (Simulator for Autonomy & Generality Evaluation) is developed in order to provide flexible task-environments with different levels of complexity to evaluate artificial intelligence systems on autonomy and generality. One of the key properties of SAGE is that it can evaluate NMIs and GMI-aspiring systems, providing a bridge for the communities surrounding narrow and general artificial intelligence research. It follows a path laid out previously by Bieger et al. (2016); Thorarensen (2016), and Thórisson et al. (2015) for evaluation of GMI. The approach of SAGE enables the evaluator to test a systems capabilities of addressing novel things. This can be done by introducing new variables, hide existing observables, change the task, and/or introduce randomness into existing tasks. These changes can possibly have unknown transition functions, and/or unknown relations to other variables. Either of these can show similar or novel behaviour in regards to the previously learned variables. The response of a system to such changes in the task-environment can lead to conclusions about the agents generality and/or autonomy.

3.1 Assumptions

In order to create an evaluation platform which evaluates NMIs and GMIs for their generality and/ or autonomy assumptions about the nature of generality and autonomy are necessary. Wang (2019) defines intelligence as the ability to adapt to novel situations under the assumption of insufficient knowledge and resources. This, however, does not clarify the term "adaptation", "autonomy", or "generality". Adaptation can be seen as general autonomy (or autonomous generality). This lead to the introduction of two working definitions which can be used in order to describe an agents adaptivity.

Generality of Agents When evaluating generality of an agent a working definition of what is meant by generality must be proposed. Generality can have different meanings depending on the background of the researchers. While in mathematics generality is seen as "applicable to as many problems as possible" in physics a general model describes the world better, than a less general one. Because of this differentiation is important to clarify that the generality evaluation of SAGE aims towards generality in the latter sense. A system which can show a *pragmatic understanding* of its surroundings and extract causal relations from the environment is seen as more general, than one which simply maps states to actions. In essence the description of generality is an agents capability to identify cause-effect-chains in its surroundings in connection with its own actions.

Autonomy of Agents Secondly autonomy of an agent must be investigated. The autonomy of an agent corresponds to its ability to use (exploit) cause-effect-chains in order to achieve a goal. These chains might be implemented by a designer (e.g. human programmer) or might come from identifying causal relations due to the generality of the learner.

3.2 Requirements

To evaluate an agent on its generality and autonomy, as well as bridging the gap between NMI and GMI research certain requirements must be met. The following requirements have been integrated into the SAGE platform:

1. Evaluation of NMI and GMI-aspiring systems and make them comparable.
2. Flexible, easy, and fast generation of task-environments for evaluation.
3. Inclusion and adjustability of complexity dimensions (cf. Thórisson et al. (2015)) before and during training.
4. Simulation of a physical 3D world for high tasks with highest complexities, including sensor systems, embodiment of the agent, and noise models.
5. Additional for GMI aspiring systems: physical division of evaluation platform and learner for evaluation of resource management to fulfil the assumption of insufficient knowledge and resources.

These aspects are necessary in order to speak of an evidence of generality of an agent. If the system can cope with them a further increase of complexity can be done by introducing the following:

6. *Observability* - By changing the observability of variables (even during run time) the systems ability to adapt to different, novel observations can be evaluated. This includes hiding previously observable variables from the agent or make previously hidden variables observable.
7. *Episodicness* - By introducing complex causal chains a sequential environment can be generated to evaluate the systems capabilities to extract causal relationships between past actions and current states.
8. *Number of causal chains* - Not only the episodicness is of interest, but also the degree of complexity in sequential environments. Parallel causal chains test for the ability to differentiate between correlation and causation.

9. *Agency* - Lastly by introducing a number of agents to the same task-environment the ability to interact with other agents and developing a goal oriented strategy for multi-agent systems can be evaluated. Other agents introduce the highest degree of unknown environments by changing their own strategies constantly while learning a task and its environment.

3.3 Task-Environment Properties

The by Thórisson et al. (2015) described eleven complexity dimensions of tasks environment are mostly covered in SAGEs architecture to evaluate the agents generality, as well as its capability to cope with different complexity dimensions. Determinism, staticism, observability, knowledge, episodicity, and discreteness can be changed before or during learning/ evaluation. Reproucibility is provided by randomisation seeds in noise control and sensor models.

Determinism The stochasticity of the task-environment can be adjusted in the observables, the actions, and in the environment dynamics. This enables the evaluator to evaluate the systems capabilities to cope with sensor noise or imprecise actuators independent of the environment itself. It also helps the evaluator to understand better, how noisy dynamics of the environment influence learning of the agent.

Staticism By making environmental variables (like for example gravity or dynamics parameters) changeable during run time static environments can be changed into dynamic ones on-the-fly or the dynamism of an environment can be increased or decreased by the evaluator at any given time.

Observability Any variable can be made observable or hidden at any time of the learning/ evaluation process. This can test the agents capability to cope with novel observations if made observable or for example sensor failure if made hidden. This also helps to analyse the crucial variables of a task. If a learner can do as good with a variable observed, as without, assumptions can be made about the importance of this variable, supporting task theory.

Knowledge The knowledge of the learner can be changed in two different ways. Either by providing knowledge to the learner by design or by training it on a different task in advance. The second gives the possibility to evaluate the capabilities of a learner to do for example transfer learning.

Episodicity The task-environments can be changed in order to provide either asynchronicity between the agent and the environment or a ”game-like” step by step training in which the environment only changes after an agent has decided on an action. This is important in order to evaluate a systems learning rate (how much data is needed in order to perform the task) or, if asynchronous, a GMI aspiring system under the assumption of insufficient time, under which the time taken in order to choose an action can have large influences on the outcome of the action.

Discreteness The continuity of the task-environment can be changed by adjusting the resolution of observations, simulating sensor accuracy. Changing the continuity of time can simulate different sensor frequencies.

3.4 MVC-A Approach

SAGEs task-environment simulation is done using a Model-View-Controller-Agent (MVC-A) approach (See figure 1). The different parts of this architecture (model, view, controller, and agent) are implemented as ROS2[2] (Quigley et al., 2009; Stanford Artificial Intelligence Laboratory et al., 2018) nodes using ROS2 as a middle-ware for inter-process communication. By using ROS2 as a middleware the process communication is done using network connections (UDP and TCP) and therefore makes a division of task-environment and learner to two different machines possible, fulfilling one of the requirements set for the SAGE platform. Further it provides an interface to the Gazebo (Koenig and Howard, 2004) simulation environment making 3D simulation of robots in complex worlds including different sensor models possible. Thereby a second requirement is met. The different task-environment properties and the remaining requirements are introduced in the different parts of the MVC-A architecture.

Model The ROS2 model node is used to store the current state of the environment (observables, non-observables, mainpulables, time (and energy)) and provides interfaces to the agent and the controller. In order to support reinforcement learning algorithms it also includes the reward calculations dependent on the current state of the environment. It passes this information to the agent by broadcasting its current state and, depending on the episodicness of the environment, passes the state to the controller either asynchronously (immediatley after receiving the new state from the controller) or only after it received action information from the agent.
Further it provides the possibility to change the stochasticity and the discreteness of

[2] Version: Eloquent Elusor, documentation: https://index.ros.org/doc/ros2/ (last accessed 7th of March 2020)

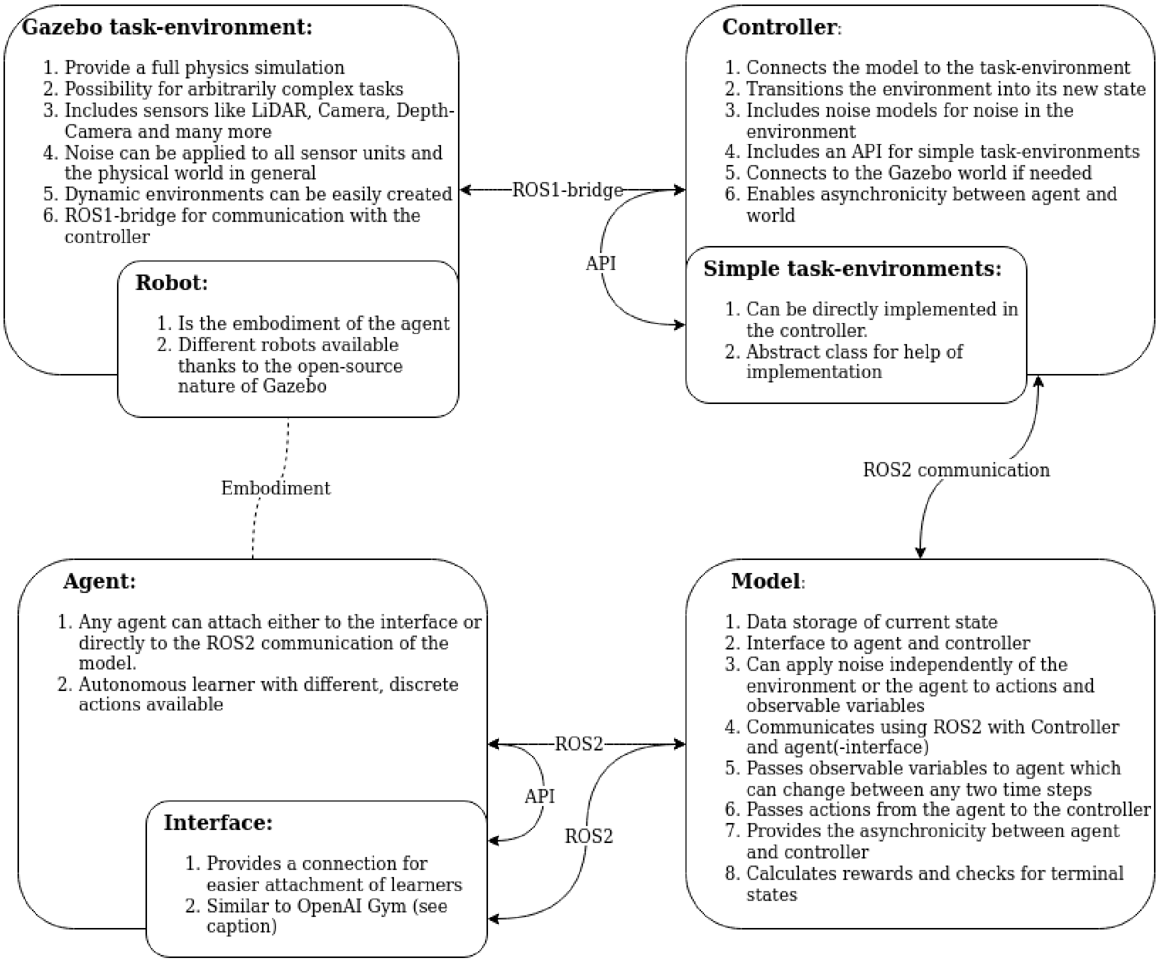
Gazebo task-environment:
1. Provide a full physics simulation
2. Possibility for arbitrarily complex tasks
3. Includes sensors like LiDAR, Camera, Depth-Camera and many more
4. Noise can be applied to all sensor units and the physical world in general
5. Dynamic environments can be easily created
6. ROS1-bridge for communication with the controller
Robot:
1. Is the embodiment of the agent
2. Different robots available thanks to the open-source nature of Gazebo
Controller:
1. Connects the model to the task-environment
2. Transitions the environment into its new state
3. Includes noise models for noise in the environment
4. Includes an API for simple task-environments
5. Connects to the Gazebo world if needed
6. Enables asynchronicity between agent and world
Simple task-environments:
1. Can be directly implemented in the controller.
2. Abstract class for help of implementation
ROS1-bridge
API
Embodiment
ROS2 communication
Agent:
1. Any agent can attach either to the interface or directly to the ROS2 communication of the model.
2. Autonomous learner with different, discrete actions available
Interface:
1. Provides a connection for easier attachment of learners
2. Similar to OpenAI Gym (see caption)
Model:
1. Data storage of current state
2. Interface to agent and controller
3. Can apply noise independently of the environment or the agent to actions and observable variables
4. Communicates using ROS2 with Controller and agent(-interface)
5. Passes observable variables to agent which can change between any two time steps
6. Passes actions from the agent to the controller
7. Provides the asynchronicity between agent and controller
8. Calculates rewards and checks for terminal states
ROS2
API
ROS2

Figure 1: Flowchart for illustration of the different components and their interactions. The controller provides the simulation of the task-environment either itself or by connecting to a Gazebo (Koenig and Howard, 2004) world. The model node provides the system with a data storage and an environment independent noise system. It passes the actions from the agent to the controller and the current state from the controller to the agent. The agent is the learner to be tested and the interface gives the possibility to easily connect existing learners by providing a similar interface as OpenAI Gym (Brockman et al., 2016).

the stored data. Noise and discreteness models of observations are only added when broadcasting to the agent. If the models are applied to the environment it only applies them when broadcasting to the controller. This makes independent evaluation of different noise models possible.

In the model is further stored which variables are observable. Only those are passed

to the agent and can be changed between any two episodes of a task including during evaluation.
Theoretically it is possible to attach any number of agents to the provided interface, this has, however, not been tested, yet.

Controller The controller provides the core calculation of any tasks done or, if Gazebo is used, provides the connection to the simulated 3D environment. It is designed to be a ROS2 node which communicates with the model node to receive the current state of the environment, agent, and chosen actions in order to calculate the transitions accordingly. The transition calculations are triggered by incoming messages from the model and therefore run asynchronous or synchronous to the agents decision making.
Tasks can be easily added and switched between any two epochs of the environment using parameter files in order to specify which task should happen when. This, in theory, makes complex task chains possible in which the agent must reach the goal of the first task before continuing onto the next. This is, however, not yet implemented, but the necessary interfaces are provided.
If Gazebo is used for simulation of the environment the controller connects to the Gazebo-ROS interface and passes information from the model, and therefore agent, on to the simulation. Validation checks of actions can however be done in the controller making for example simulation of actuator failure possible by prohibiting certain actions. This also can be changed during evaluation the same way, as changing tasks (actuator failure and data prohibition would be a different task than with a fully functioning body).

Agent The agent consists of two parts (if evaluating current RL algorithms), the agent itself and an interface to the model. The interface receives ROS2 messages broadcast by the model and processes them in order to pass to the learner. It also stores the history of agents actions, as well as model messages in order to save them regularly for later analysis. It further resets the learner if evaluation of a novel task without previous knowledge is wanted and lets the agent know, when a task is done (either failed or goal reached). The agent itself can be any RL algorithm able to deal with the provided data. It has been tested on two different deep reinforcement learners, an actor-critic-learner, and a double-deep-q-network learner. If other learners (including GMI-aspiring ones) are attached the evaluator has the choice of either using the interface provided or to attach directly to the ROS2 communication, implementing their own message to data processing.

View The view currently implemented is either the 3D simulation of Gazebo, or the ROS2 built in rqt_plot. Rqt_plot provides an easy to use monitoring of published observables by attaching to the ROS2 communication network. This way it allows remote monitoring of the agents current progress of the task and therefore a better understanding of the decision making of the agent. When using Gazebo for simulation it provides a 3D environment by itself which can be used to monitor the agents progress.

Using this approach of ROS2 communication between the different parts of the MVC-A architecture brings main advantages. (a) The previously described evaluation possibility of resource management; (b) The real-time processing and asynchronous calculations if necessary; (c) The possibility to attach any number of agents to the task-environment simulation; (d) The possibility to run multiple SAGE platforms at the same time by using ROS internal namespacing, speeding up the generation of statistically relevant data; and (e) the independence of model-to-agent and model-to-controller communication making changes to both easier.

4 Evaluation Methodology – Proof of Concept

In order to show the capabilities of SAGE a proof of concept evaluation of two different learners was done. An actor-critc deep reinforcement learner (AC)[3] (Konda and Tsitsiklis, 2000) and a double-deep-Q-network learner (DDQ)[4] (Van Hasselt et al., 2016) were used for this proof of concept. For this they were evaluated on the inverted pendulum task derived from the OpenAI gym's (Brockman et al., 2016) cart-pole-v-0 task[5]. In order to reduce outlying performances all tests have been done 40 times per learner and the median of those results was calculated and is presented in the result section.

4.1 Inverted Pendulum Task

In the inverted pendulum task the agent has the goal to balance a pole on a cart upright over as many iterations as possible (See figure 2). The settings that were used in the SAGE platform are similar to the ones used in OpenAI gym. The mass of the cart is $M = 1.0$ kg, the mass of the tip of the pole is $m = 0.1$ kg, the length of the pole is $l = 0.5$ m, and the gravity-constant used is $g = 9.81\frac{m}{s^2}$. The learner gets a reward signal of $r = +1$ if the pole's angle difference to the upright position is in $\Theta \in [-12°, 12°]$ and the position difference of the cart is in the interval $x \in [-2.4m, 2.4m]$. Otherwise the reward signal is $r = -1$ and the episode is failed, restarting a new episode from the upright position.

The differential equations of the transition of the environment ($s_o \rightarrow s_n$ is calculated iteratively with $\delta t = 0.001s$, the model is updated in $t = 0.02s$ steps and passes the data to the agent. The new state s_n is calculated by the following set of equations:

$$\ddot{\Theta}_n = \frac{((M+m)\cdot g\cdot sin(\Theta_o) - (cos(\Theta_o)\cdot(F + m\cdot l\cdot\dot{\Theta}_o^2\cdot sin(\Theta_o)}{\frac{4}{3}\cdot(M+m)\cdot l - m\cdot l\cdot cos^2(\Theta_o)} \tag{13}$$

$$\ddot{x}_n = \frac{(F + m\cdot l\cdot(\dot{\Theta}_o\cdot sin(\Theta_o) - \ddot{\Theta}_n}{M+m} \tag{14}$$

$$\dot{x}_n = \dot{x}_o + \ddot{x}_n\cdot\delta t \tag{15}$$

$$x_n = x_o + \dot{x}_n\cdot\delta t \tag{16}$$

$$\dot{\Theta}_n = \dot{\Theta}_o + \ddot{\Theta}_n\cdot\delta t \tag{17}$$

[3] Adapted from https://github.com/gearsuccess/Reinforcement-learning-1 (last visited 7th of March 2020) and (Kostrikov, 2018)

[4] Adapted from https://github.com/jbocinsky/Big_Data_Project/blob/master/ (last visited 7th of March 2020)

[5] https://gym.openai.com/envs/CartPole-v0/ (last visited 7th of March 2020)

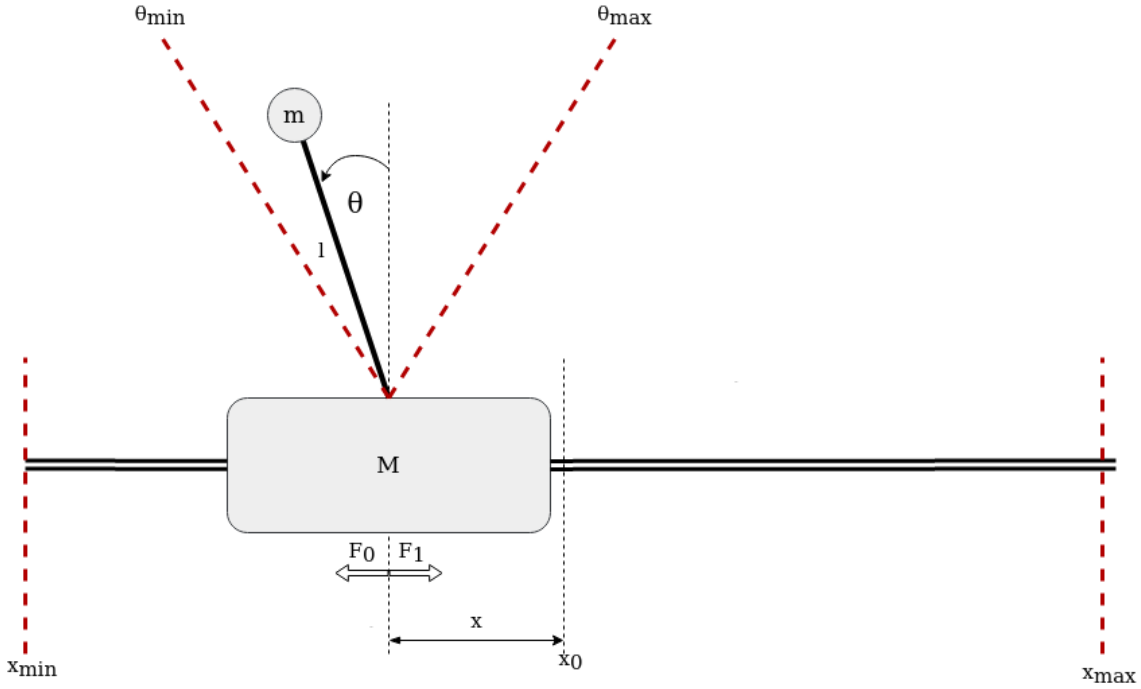

Figure 2: Illustration of the cart-pole/ inverted pendulum task as implemented in SAGE derived from Brockman et al. (2016). $F_0 = -10N$, $F_1 = 10N$ possible actions the agent can apply; x position of the cart; Θ angle of the pole; $M = 1.0kg$ mass of the cart; $m = 0.1kg$ mass at the tip of the pole; and $l = 0.5m$ the length of the pole. The whole task underlies gravity $g = 9.81\frac{m}{s^2}$. The goal space is constraint by Θ_{min} and Θ_{max} for the angle of the pole and x_{min}, x_{max} for the position of the cart. While the pole and cart is kept in the goal space a reward of $r = +1$ is sent to the agent, otherwise $r = -1$ and the task is reset to $\Theta = 0.0°$, $x = 0.0m$.

$$\Theta_n = \Theta_o + \dot{\Theta}_n \cdot \delta t \tag{18}$$

The default settings of task's complexity with which the different changes in complexity are compared are as follows:

- Observation space: The agent can observe x, $\dot{x}$, Θ, $\dot{\Theta}$.
- Action space: The agent has two different action options: $F \in \{-10N, +10N\}$ for moving the cart sideways.

- Stochasticity: Observations, actions, and environment dynamics are completely deterministic.

Further a random policy agent was tested to generate a comparable minimum performance. This agent chooses actions in every time-step completely at random.

4.2 Tests

In order to test the agents generality a set of different evaluations was run on the inverted pendulum tasks with the two learners:

1. The noise on all observations was increased. This was done between evaluation tests, giving the learner the ability to learn the noisy data from beginning. This was done in order to examine the learners capability to cope with sensor noise without changing the determinism of the environment.

2. The noise on all actions was increased, again between evaluation tests. This was done to better understand the influences of actuator imprecision on task performance, as well as to analyse the agent's policy's dependencies on action values in this particular task.

3. The noise on the observable variables, but in the environment dynamics, was increased between evaluation tests. This makes an evaluation on how uncontrollable noises influence the performance possible.

4. Only a single observable was randomised by the maximum noise applied in 1.. This can give insight in the importance of variables and their accuracy to achieve a high performance on the task.

5. One observable was randomised with extremely high noises (standard deviation of 10 times the commonly observed maximum value of the variable) during a full training/ evaluation cycle.

6. One observable was hidden during a full training/ evaluation cycle. This in combination with 5 can give insight on the generality of the agent. Strongly randomised variables should have a similar impact on the learner's performance, as if it were hidden, if a generalisation takes place.

7. The forces applied by the agent were inverted after training and re-inverted to normal later. This test can give insights in the agent's capability to extract cause-effect relations and its ability of autonomously transfer knowledge of a similar task.

5 Evaluation Results

The results of evaluating a DDQ learner and an AC learner on the tests described in 4.2 show the differences of these two learners and their capabilities to cope with different complexity levels and dimensions. The two learners hyper-parameters and network properties were as follows:

AC

The AC algorithm used to solve the inverted pendulum task had the following properties:

- Two networks, each with dense layers. The actor network with the number of observables as input dimension, one hidden layer with 256 neurons and the number of actions available as output dimension. The critic with the same input, 256 neurons on the hidden layer, but only one output.
- As optimiser the Adam optimiser was used (Kingma and Ba, 2014).
- A learning rate of $7.5 \cdot 10^{-4}$ was chosen and
- a discount factor of $\gamma = 0.99$.

DDQ

The DDQ algorithm used to solve the inverted pendulum task had the following properties:

- Two networks, each with dense layers. Both networks had the number of observables as inputs, a hidden layer with 64 neurons and an output-dimension of the number of available actions.
- As optimiser the Adam optimiser was used (Kingma and Ba, 2014).
- A learning rate of 0.001 was chosen,
- a discount factor $\gamma = 0.99$,
- a start exploration rate of 1.0 with a decay of 0.99 and a minimal exploration rate of 0.01.
- to update the target network an update factor of $\tau = 0.125$ was chosen.

These parameter settings are not necessarily optimal for this task, but since the evaluation was done in order to proof the concept of the SAGE platform, and not to create single performance measures, the properties have not been optimised in a time consuming matter.

5.1 Stochasticity

An evaluation of the DDQ and the AC learner show that different kinds of noise have different impacts on the learners possibilities of (a) learning the task and (b) achieve a high performance score.

5.1.1 Observations

Stochastic observations lead to worse performance of both the DDQ learner and the AC learner. While low noise levels seem to influence the learning rate of the learner, high noise levels show decreased learning rate and performance loss in both evaluated learners.

Full Observation Noise In this scenario the noise applied to the observations was applied on all observable variables by adding randomly drawn values from a normal distribution with $\mu = 0.00$ and $\sigma = x\%$ to the variables value. The percentage represents the percentage of a variables commonly occurring maximal magnitude of values when the task is failed.

$$x_{max} = 2.4m; \quad \dot{x}_{max} = 2.4\frac{m}{s}; \quad \Theta_{max} = 0.21rad; \quad \dot{\Theta}_{max} = 0.4\frac{rad}{s} \qquad (19)$$

The standard deviation was increased in 10 % steps from 0-100 % to evaluate if any learning still takes place. The resulting performance was compared to a random policy agent which chose a random action in every time-step. Figure 3 shows the results of the evaluation of both learners. The results show, that for the AC, noise levels between 0-40 % have only little influence on the learners performance, but change the learning rate of the task. Observation noise has higher impact on the performance of the DDQ learner. This might, however, be a problem of DDQ learner's late convergence on policies in comparison to the AC. Both learners, however, show drastic performance loss when the noise level is increased to more than 60 %. But even when noise levels of 100 % of the goal state are applied both learners still show better performance, than a random policy. Therefore relevant information could still be extracted and used in order to increase performance. Therefore it can be stated, that stochastic observations, while having an influence on the learning rate, do not lead to wrong policies and critical failure like misinterpreting observed data.

Single Variable Noise In order to test the importance of variables to complete the task two different variables were chosen which were randomised by the previously described 100 % level of noise. The two chosen variables are the angle of the pole Θ and the velocity of the cart $v = \dot{x}$. The results show, that the randomisation of Θ has less influence, than the randomisation of $\dot{x}$. This is in contradiction to the expected

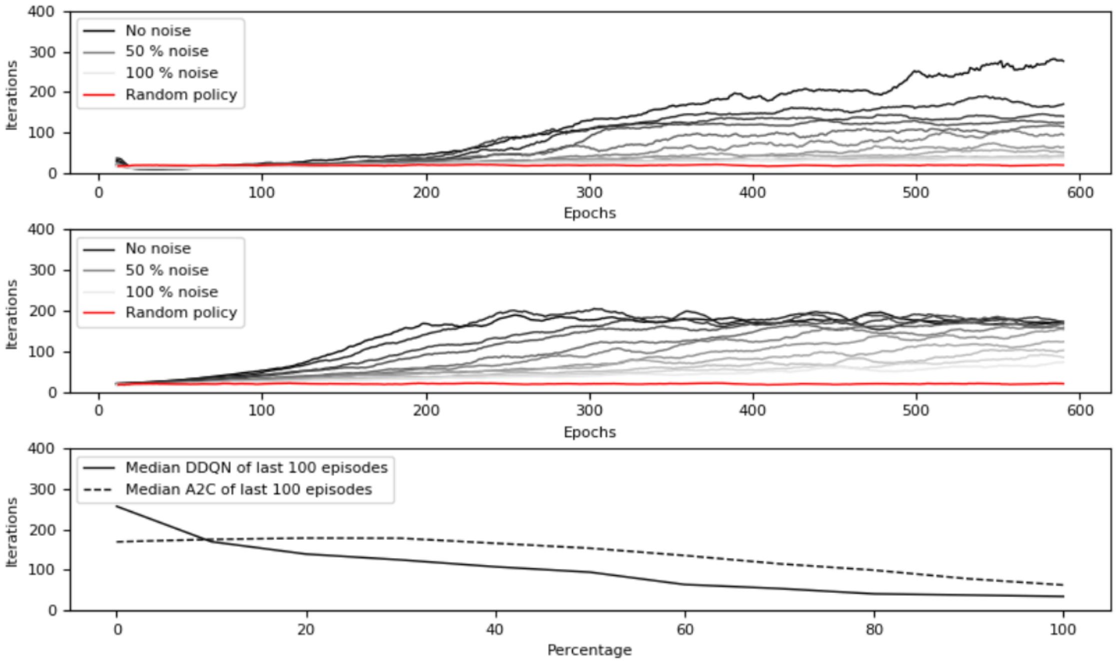

Figure 3: Evaluation of a DDQ and a AC learner on their capabilities to cope with noise on observed data. Noise applied on all observables in percentage of the goal state of the inverted pendulum task (or commonly occurring maximum values in case of goal independent variables $\dot{\theta}$ and $\dot{x}$. Random policy for comparison. Tests were done 40 times and median of each episode is plotted. Data was smoothed with a running mean with window width 10 for better visualisation. Top: DDQ learner; Middle: AC learner; Bottom: Comparison of the median of different noise levels of the performance after training (median of last 100 episodes from DDQ/ AC learner).

result. Since Θ is part of the goal constraints, other than $\dot{x}$ it was assumed, that its randomisation has a higher impact on the performance. For both Θ, and $\dot{x}$ the conclusion can therefore be drawn, that their accuracy by itself has little influence of the learner's final performance. The other variables seem to include enough relevant information to achieve high performance scores on the task. The results of this test can be seen in figure 4. In this test only the AC was evaluated and therefore no comparison between the DDQ and the AC learner is available.

5.1.2 Dynamics

Noise on the environments dynamics have a stronger impact on the performance, than noise on the observations. The same noise percentage definition as with the observa-

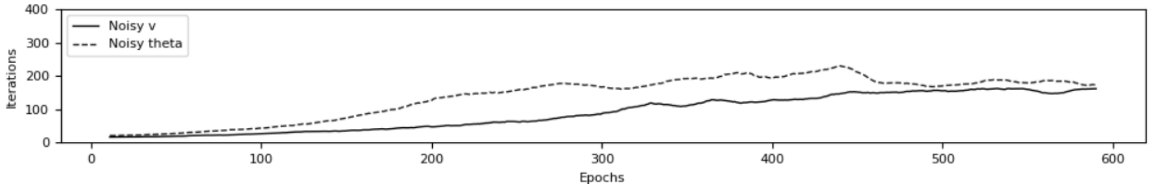

Figure 4: Evaluation results of the AC with a single stochastic variable. Comparison of learning speed and final performance with randomisation of $v = \dot{x}$ and Θ. Randomisation by normal distribution with $\mu = \dot{x}$ and $\sigma = 2.4m$ for $\dot{x}$, and $\mu = \Theta$ and $\sigma = 12°$ for Θ, respectively. Learning speed with randomised $\dot{x}$ is slower, than with randomised Θ. All other observables are true values from the environment simulation. Tests were done 40 times and the median of each episode is displayed. For better visualisation a running mean with window width 10 was applied.

tion noise test has been used. Results show, that 1-10 % of noise on the environments dynamics have a similar impact, as 10-100 % of noise on the observations. This can have an agent independent explanation: Noise on the dynamics can result in a fail of the task independent of the agent's chosen action. However this must not make up for a factor of 10 in the noise percentage before performance drops drastically. Another influencing factor can be the noise propagation when applying noise to $\dot{x}$ and $\dot{\Theta}$. Lastly the environment noise might have an influence on the Markov-Decision-Process, by removing the constraint, that any future state is only dependent on the current state. By applying state independent noise to the dynamics this assumtion might fail. This, however, must be tested by further evaluations.
This observation of impact of noise in the dynamics of the environment agrees with the by Bellemare et al. (2015) identified problems with the ALE evaluation platform. Randomness in the environment is necessary in order to evaluate a learners generality. The results of this test show similarity to the results of applying noise to the observations. The AC learner copes better with low noises in regards of the final performance, the DDQ learner, however, shows close to no deterioration of learning speed when low noise levels are introduced unlike the AC. With more than 3 % noise both AC and DDQ learner show similar final performance with faster learning speed of the DDQ learner. While the DDQ learner still can extract relevant information at 10 % noise levels, the AC learner degrades to almost the same performance, as a random policy agent. Therefore the conclusion can be drawn, that, while noise on the environment impacts learning stringer, than noise on the observations this still does not lead to wrong policies of the learner. This is also a conclusion that can be used in real world applications. A noise reduced environment (if possible) is more important for high performance scores, than high accuracy sensors.

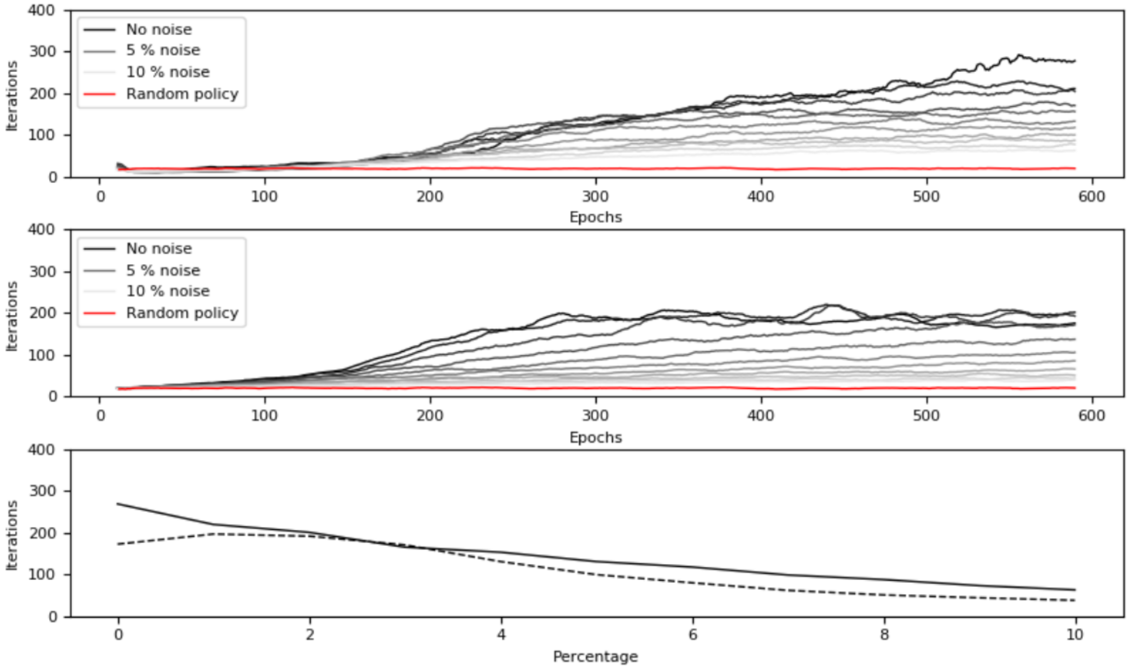

Figure 5: Evaluation of a DDQ and a AC learner on their capabilities to cope with noise on the environments dynamics. Noise applied on all observables in percentage of the goal state of the inverted pendulum task (or commonly occuring maximum values in case of goal independent variables $\dot{\theta}$ and $\dot{x}$. Random policy for comparison. Tests were done 40 times and median of each episode is plotted. Data was smoothed with a running mean with window width 10 for better visualisation. Top: DDQ learner; Middle: AC learner; Bottom: Comparison of the median of different noise levels of the performance after training (median of last 100 episodes from DDQ/ AC learner).

5.1.3 Actions

Applying noise on the action's values applied by the agent to the environment has no significant impact on the learners performance. Figure 6 shows, how increasing the noise leads to almost identical results in both the AC, and the DQN learner. This can be due to the simplicity of the task. Since the policy does not change no matter which force is applied, as long, as they are not inverted, noise on the action has no impact on the learning of the agent. In more complex tasks this can be different. The policy of the inverted pendulum task, however, is simple enough to always follow the policy "Apply force to the opposite direction of the direction of the pole". This is always true and always leads to the best results, no matter how strong the applied action is, as long as it is not negative. The impact of force inversion can be seen in section 5.3.

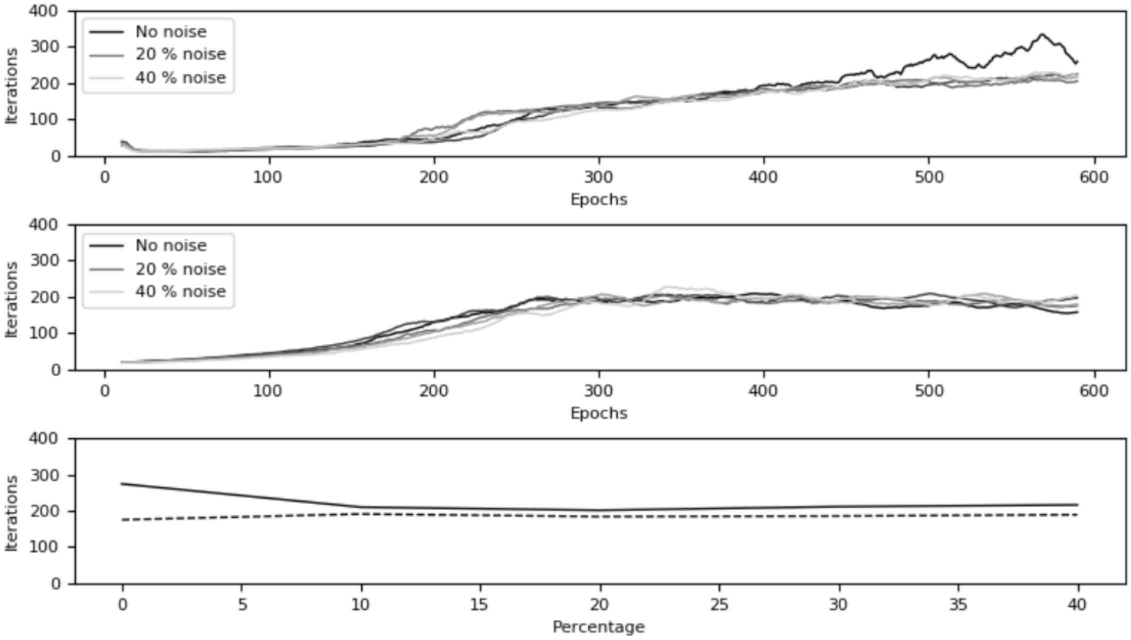

Figure 6: Evaluation of the DDQ and AC in regards of their capabilities to cope with actuator imprecision. Noise applied as previously described. Top: DDQ; Middle: AC; Bottom: Median of achieved performance of the last 100 episodes after training.

5.2 Hidden variable

In order to evaluate the importance of accuracy and stochasticity further, than randomising a single observable variable by 100 % noise a test was created in which a single variable ($\dot{x}$) was randomised with 1000 % noise. For comparison of the learner's performance a second test was done in which the observable was hidden instead of randomised. With this the generalisation of knowledge can be evaluated. A general learner should be able to identify the randomness of the variable and therefore the lack of relevant information and treat the observation as if the variable does not exist. The results from the evaluation of the DDQ learner plotted in figure 7 show, however, that this is not the case. Random variables have an impact on the learner's performance, making it worse, than with a hidden variable. This test was only done on $\dot{x}$ and might be different on other variables. However the conclusion that can be drawn is that the DDQ learner does not create an importance ranking of variables by which random variables could be neglected in decision making. Therefore the usage of the RL algorithms must be supervised by a human designer, who identifies irrelevant variables and hide them from the learner. This might be one of the reasons, why bias in AI exists. Making it necessary to have a human developer identifying relevant data also includes the human's bias of "what is necessary data and why". The test results only include the DDQ learner due to the problem, that too high noise levels resulted in negative probabilities in the AC's networks leading to total failure of the learner.

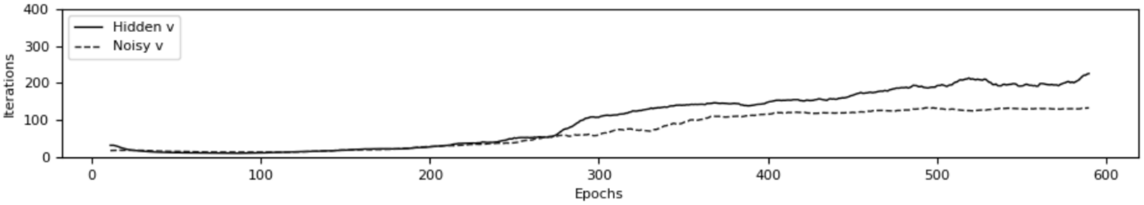

Figure 7: Test of the DDQ algorithm to cope with a hidden variable vs. the same variable randomised (in observation). The variable $v = \dot{x}$ was chosen to be hidden or randomised.

5.3 Task Inversion

This test shows an agents capability to extract causal relations and use them in order to increase learning speed. By inverting the task's actions after training on the original action-set the time needed to re-learn the novel task is an indicator for the learners generalisation capabilities, as well as its capabilities of transfer learning (TL). The first phase of training on the original task can be seen as the source task in a TL task description, the second phase, after inverting the actions, can be seen as the target task. Sheikhlar et al. (2020) describes the necessary similarity calculation of variables, states, causes, effects and transitions in order to exploit them to increase the learning rate of the agent. By inverting the forces the similarities in state and variables stay the same in both source and target task, but the effects and therefore transitions of the task differ drastically. A learner capable of TL can use this information and find the similarity of effect of the previously learned other action in order to map the new task to the old one. A learner which simply produces state-action mappings, however, can be fooled into choosing the wrong action repeatedly.

The AC learner's results displayed in figure 8 show long relearning periods before reaching previous performance after inversion. This shows, that the causal relation between an action and the impact of it in the environment was probably not extracted and that the level of TL is low in the learning algorithm. The quadrupled time before reaching previous performance also shows, that the source task has a strong negative influence on the target task. Further by re-inverting to the original task after learn-

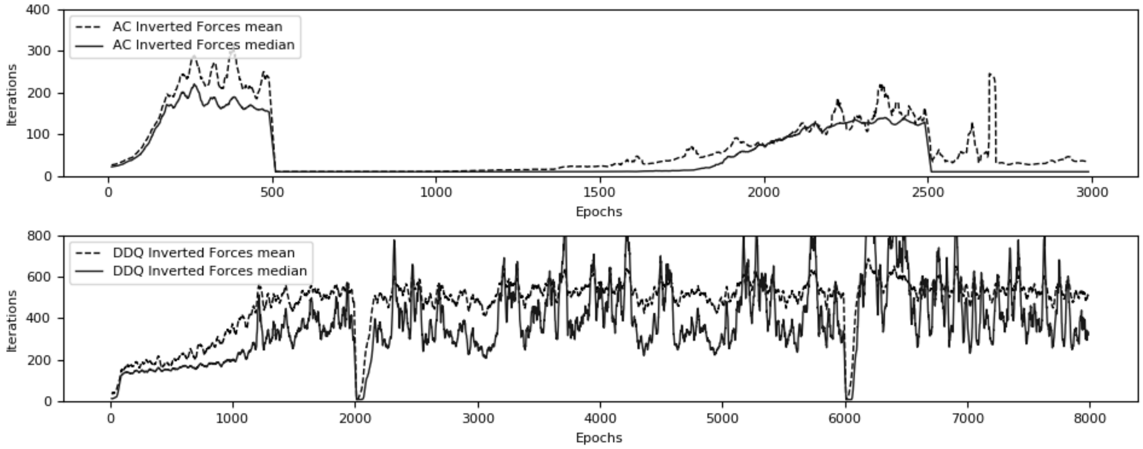

Figure 8: Inversion of forces of the inverted-pendulum task after training of the AC/ DDQ learner. Mean and median of each episode plotted. Running mean applied to the figure for better visualisation with window width 10. Top: AC leaner, forces inverted after 500 episodes of training and re-inverted to the original version after another 2000 episodes. Bottom: DDQ learner, forces inverted after 2000 episodes of training and re-inverted after another 4000 episodes. Longer learning periods for the DDQ were chosen to test, if longer periods have an impact on the performance after inversion.

ing the target task the AC has to relearn the source task. This means, that critical forgetting of the source task happened when learning the target task.

Additionally the inspection of the mean, rather than the median, of the 40 trials, shows, that the previously learned knowledge was not fully forgotten. In figure 8 it can be seen, that after re-inverting the task to the original training set high performance scores were achieved in some runs. The conclusion drawn from this is, that those phases of sudden high-performance episodes can be seen when the learner encounters a state which was not discovered during re-learning of the target task. The agent can therefore use its policy from the first learning phase and thereby achieve high performance scores. Therefore after learning a similar task in addition to the source task the examiner can make no predictions on the outcome of an episode without knowledge about which states were discovered during re-learning. This makes an argument for the unusability of an AC after re-training on a novel task with high state similarities. Further tests must be done to confirm this theory and argument.

The DDQ learner, on the other hand, shows promising results in generalisation and TL. After inverting the forces the DDQ learner takes only a few episodes (50-100) before reaching the previous performance. This shows, that the previously learned source task has no negative effects on the learner (other than on the AC) but rather increases learning speed. The same happens after re-inverting the actions to the original setup. This might have two reasons: (a) the DDQ learner is a transfer learner and uses causal relations and similarity of effect states in order to adjust the policy; or (b) the task shows to little complexity for a DDQ learner, leading to a ceiling effect.

6 Conclusions & Future Work

The results presented in section 5 show some of the possibilities the SAGE evaluation platform provides for in depth evaluation of RL methodologies. With results like those presented in section 5.3 or 5.2 a conclusion on the generality of an agent can be drawn and its capabilities to do TL can be assessed. Both are clear indicators for generality of an agent. In order to support these findings further tests are necessary and a comparison table of different learners should be created. As a first proof of concept these results show the possibilities a platform like SAGE gives to the research community.

Further not only information about the systems advantages and flaws can be extracted from the evaluation results, but also the variables' influences for the task-environment itself. The application of noise on the action for example shows, that the policy of the task seems to be independent of the force applied to the cart. This is in line with the expectation, since the task has such low complexity it can be described by only a few words: ”Push in the opposite direction of the falling pole.”. This would be in line with independent action value and policy. When it comes to the randomisation of a single variable, however, such an intuitive description seems to fail. Apparently the noise on the observation of the angle of the pole has less impact on the performance, or learning rate, than the randomisation of the velocity. Therefore the velocity seems to play a more important role for learning, than the angle, even though the velocity is not constraint by the goal space. This information about the importance of variables could be used for the development of a task-theory, as described in section 2.2.1.

Investigating the performance of random variables vs. hidden variables gave a better insight in the importance of the human designer for current AI architectures. An agent which is not capable of identifying random variables and removing them from the decision process cannot be expected to process large amounts of data with distracting variables. Currently a human designer seems to be necessary in order to remove distractions. This however puts the human bias of identifying distracting variables into the system which leads rather to the support of expected results, than new correlations or causal relations which might be discovered in data unchanged by human designers. This is a strong argument for why current approaches of AI still show only little generality.

The conclusion that can be drawn from the evaluation of the actor-critic (AC), and the double-deep-Q (DDQ) learner is, that the DDQ learner shows more generality, than the AC. The DDQ learner showed no critical failure in any of the conducted tests, while the AC failed at highly random variables leading to fatal errors. This shows the strong influence such high noise levels have on the learning. Further the AC was not able to compensate the inversion of the task's forces leading to very long relearning phases which lead to the conclusion, that no knowledge generalisation took place. This in comparison with the DDQ learner, which only took a short time to compensate the force inversion shows the higher level of generality of the DDQ learner. Lastly a conclusion can be made from the relevant information extracted from variables with strong noises both in the environments dynamics, and in the observations. The DDQ learner was able to extract more information on the same noise levels, than the AC resulting in a better performance. This is an indicator, that the information extraction, and therefore the learning of a task, is better in the DDQ learner.

Summarised it can be said, that SAGE offers a platform to continuously monitor the progress of AI research towards GMI by offering full adjustability on any task-environment dimensions required to test for generality. SAGE further includes the possibility to change action and observation space during evaluation. This however could not be tested, since all current AI architectures, that could be implemented, did not support a change of either observation or action space size during run-time. This leads to a strong argument against the autonomy of the agent. Without human interference changes like for example sensor failures or actuator failures cannot be compensated by the system alone. A GMI, however, should be able to not only adjust its observation space to novel methods of observing the environment, but also adjust its action space in order to use causal chains previously unknown to achieve a goal. This can be summarised under the lack of action grouping. Without changing the action space every decision must be converted into at the initialisation defined actions, rather than creating bundles which could be presented as single action outputs.

In the future more tests with more different RL algorithms should be done in order to support the claims made in this thesis. GMI apsiring architectures could not be evaluated, yet, due to two problems: (a) only few GMI aspiring systems exist to date; and (b) the setting up of the learner so that it can attach to the SAGE platform is non-trivial and time consuming. However the amount of adjustabilities provided by SAGE should make the evaluation of GMI aspiring systems possible. It is planned to test this on two GMI aspiring systems, AERA (Nivel et al., 2013) and NARS (Wang, 2006) in order to confirm the GMI evaluation possibilities of SAGE and to compare those GMI aspiring system's generality with current RL strategies.

A range of performance data of both NMI and GMI systems is planned for the future. This might make the identification of important aspects for GMI architectures possible and give the AI research community an insight into the complexity dimensions of tasks making conclusions about the difficulty possible. This in combination with for example psychometric evaluation like IRT can make the identification of new milestones for AI research possible leading to further progress in this field.

Bibliography

Adams, S., Arel, I., Bach, J., Coop, R., Furlan, R., Goertzel, B., Hall, J. S., Samsonovich, A., Scheutz, M., Schlesinger, M., et al. (2012). Mapping the landscape of human-level artificial general intelligence. *AI magazine*, 33(1):25–42.

Baars, B. J. and Franklin, S. (2009). Consciousness is computational: The lida model of global workspace theory. *International Journal of Machine Consciousness*, 1(01):23–32.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., Legg, S., and Petersen, S. (2016). Deepmind lab. *arXiv preprint arXiv:1612.03801.*

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.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. (2015). The arcade learningenvironment: An evaluation platform for general agents(extended abstract). In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, page 41484152.

Bieger, J., Thórisson, K. R., Steunebrink, B. R., Thorarensen, T., and Sigurardóttir, J. S. (2016). Evaluation of general-purpose artificial intelligence: why, what & how. *Evaluating General-Purpose AI.*

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *arXiv preprint arXiv:1606.01540.*

Campbell, M., Hoane Jr, A. J., and Hsu, F.-h. (2002). Deep blue. *Artificial intelligence*, 134(1-2):57–83.

Eberding, L. M., Sheikhlar, A., and Thórisson, K. R. (submitted in 2020). Sage: Task-environment platform for autonomy and generality evaluation. In *International Conference on Artificial General Intelligence*. Springer.

Goertzel, B. (2009). Cognitive synergy: A universal principle for feasible general intelligence. *Proceedings of the 2009 8th IEEE International Conference on Cognitive Informatics, ICCI 2009*, pages 464–468.

Goertzel, B., Iklé, M., and Wigmore, J. (2012). The architecture of human-like general intelligence. In *Theoretical Foundations of Artificial General Intelligence*, pages 123–144. Springer.

Goertzel, B., Pennachin, C., and Geisweiller, N. (2014). A preschool-based roadmap to advanced agi. In *Engineering General Intelligence, Part 1*, pages 355–364. Springer.

Grondman, I., Busoniu, L., Lopes, G. A., and Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307.

Hernández-Orallo, J. (2015). Stochastic tasks: Difficulty and levin search. In *International Conference on Artificial General Intelligence*, pages 90–100. Springer.

Hernández-Orallo, J., Baroni, M., Bieger, J., Chmait, N., Dowe, D. L., Hofmann, K., Martínez-Plumed, F., Strannegård, C., and Thórisson, K. R. (2017). A new ai evaluation cosmos: Ready to play the game? *AI Magazine*, 38(3):66–69.

Ishii, S., Yoshida, W., and Yoshimoto, J. (2002). Control of exploitation–exploration meta-parameter in reinforcement learning. *Neural networks*, 15(4-6):665–687.

Johnson, M., Hofmann, K., Hutton, T., and Bignell, D. (2016). The malmo platform for artificial intelligence experimentation. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 4246–4247.

Johnston, B. (2010). The toy box problem (and a preliminary solution). In *Conference on Artificial General Intelligence*. Atlantis Press.

Kaptelinin, V. and Nardi, B. A. (2006). *Acting with technology: Activity theory and interaction design*. MIT press.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. (2016). Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE.

Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., and Farhadi, A. (2017). Ai2-thor: An interactive 3d environment for visual ai. *arXiv preprint arXiv:1712.05474*.

Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014.

Kostrikov, I. (2018). Pytorch implementations of asynchronous advantage actor critic.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Legg, S., Hutter, M., et al. (2005). A universal measure of intelligence for artificial agents. In *International Joint Conference on Artificial Intelligence*, volume 19, page 1509. Lawrence Erlbaum Associates Ltd.

Legg, S., Hutter, M., et al. (2007). A collection of definitions of intelligence. *Frontiers in Artificial Intelligence and applications*, 157:17.

Levesque, H., Davis, E., and Morgenstern, L. (2012). The winograd schema challenge. In *Thirteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

Li, Y. (2017). Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Luger, G. F. (2005). *Artificial intelligence: structures and strategies for complex problem solving*. Pearson education.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. (2018). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562.

McCarthy, J. (1988). Mathematical logic in artificial intelligence. *Daedalus*, pages 297–311.

McCarthy, J., Minsky, M. L., Rochester, N., and Shannon, C. E. (2006). A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12.

Mikolov, T., Joulin, A., and Baroni, M. (2016). A roadmap towards machine intelligence. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 29–61. Springer.

Minsky, M. (1988). *Society of mind*. Simon and Schuster.

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.

Newell, A. and Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126.

Nivel, E., Thórisson, K. R., Steunebrink, B. R., Dindo, H., Pezzulo, G., Rodriguez, M., Hernandez, C., Ognibene, D., Schmidhuber, J., Sanz, R., et al. (2013). Bounded recursive self-improvement. *arXiv preprint arXiv:1312.6764*.

Oppy, G. and Dowe, D. (2019). The turing test. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2019 edition.

Pennachin, C. and Goertzel, B. (2007). Contemporary approaches to artificial general intelligence. In *Artificial general intelligence*, pages 1–30. Springer.

Pomerantz, J. R. (2006). Perception: overview. *Encyclopedia of cognitive science*.

Poole, D. L. and Mackworth, A. K. (2010). *Artificial Intelligence: foundations of computational agents*. Cambridge University Press.

Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan.

Riedl, M. O. (2014). The lovelace 2.0 test of artificial creativity and intelligence. *arXiv preprint arXiv:1410.6142*.

Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Sheikhlar, A., Thórisson, K. R., and Eberding, L. M. (submitted in 2020). Autonomous cumulative transfer learning. In *International Conference on Artificial General Intelligence*. Springer.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

Sloman, A. (2001). Varieties of affect and the cogaff architecture schema. In *Proceedings of the AISB01 symposium on emotions, cognition, and affective computing. The Society for the Study of Artificial Intelligence and the Simulation of Behaviour*, volume 58.

Stanford Artificial Intelligence Laboratory et al. (2018). Robotic operating system. available at https://www.ros.org.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Świechowski, M., Park, H., Mańdziuk, J., and Kim, K.-J. (2015). Recent advances in general game playing. *The Scientific World Journal*, 2015.

Synnaeve, G., Nardelli, N., Auvolat, A., Chintala, S., Lacroix, T., Lin, Z., Richoux, F., and Usunier, N. (2016). Torchcraft: a library for machine learning research on real-time strategy games. *arXiv preprint arXiv:1611.00625*.

Taylor, M. E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685.

Thorarensen, T. (2016). *FraMoTEC: A Framework for Modular Task-Environment Construction for Evaluating Adaptive Control Systems*. PhD thesis.

Thórisson, K. R. (2012). A new constructivist ai: from manual methods to self-constructive systems. In *Theoretical Foundations of Artificial General Intelligence*, pages 145–171. Springer.

Thórisson, K. R., Bieger, J., Li, X., and Wang, P. (2019). Cumulative learning. In *International Conference on Artificial General Intelligence*, pages 198–208. Springer.

Thórisson, K. R., Bieger, J., Schiffel, S., and Garrett, D. (2015). Towards flexible task environments for comprehensive evaluation of artificial intelligent systems and automatic learners. In *International Conference on Artificial General Intelligence*, pages 187–196. Springer.

Thórisson, K. R., Bieger, J., Thorarensen, T., Sigurardóttir, J. S., and Steunebrink, B. R. (2016). Why artificial intelligence needs a task theory. In *International Conference on Artificial General Intelligence*, pages 118–128. Springer.

Thrun, S. B. (1992). Efficient exploration in reinforcement learning. Technical report, Technical Report CMU-CS-92-102, School of Computer Science, Carnegie Mellon .

Turing, A. M. (2009). Computing machinery and intelligence. In *Parsing the Turing Test*, pages 23–65. Springer.

Van Hasselt, H., Guez, A., and Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.

Wang, P. (2006). *Rigid flexibility: the logic of intelligence*, volume 34. Springer Science & Business Media.

Wang, P. (2007). The logic of intelligence. In *Artificial general intelligence*, pages 31–62. Springer.

Wang, P. (2019). On defining artificial intelligence. *Journal of Artificial General Intelligence*, 10(2):1–37.

Wang, P., Liu, K., and Dougherty, Q. (2018). Conceptions of artificial intelligence and singularity. *Information*, 9(4):79.

Xie, M., Jean, N., Burke, M., Lobell, D., and Ermon, S. (2016). Transfer learning from deep features for remote sensing and poverty mapping. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Zamora, I., Lopez, N. G., Vilches, V. M., and Cordero, A. H. (2016). Extending the openai gym for robotics: a toolkit for reinforcement learning using ros and gazebo. *arXiv preprint arXiv:1608.05742*.