BSc in Computer Science

Performance of an AGI-Aspiring System & Narrow-AI Approaches: A Systematic Comparison

Author Sindri P. Andrason
Instructor Kristinn R. Thórisson
Evaluator Guðmundur M. Einarsson

June 2020

Department of Computer Science
School of Science and Engineering

Abstract. While research progresses in artificial general intelligence (AGI), as well as in traditional narrow-AI such as machine learning algorithms, direct comparisons between AI learners are few and far between. Evaluating and comparing the performance of different architectures is likely to prove valuable, and so is estimating the generality of an architecture by measuring traits such as transfer learning ability and knowledge retention. In this context, transfer learning ability describes how knowledge gathered learning one task speeds up learning of a similar task, while knowledge retention describes how well a task is performed after a second task has also been learned. We examine how OpenNARS for Applications (ONA), a version of the AGI-aspiring Non-Axiomatic Reasoning System, measures up against traditional narrow-AI implementations on these metrics. Few, if any, tools allow such a direct comparison of AGI-aspiring and more traditional narrow-AI systems, an exception being the Simulator for Autonomy & Generality Evaluation (SAGE) system. We tested ONA in SAGE, using tasks identical to those previously tested on a double deep Q network (DDQ) and an actor-critic (AC). ONA was found to vastly outperform the narrow-AI learners in proficiency and learning speed, but could not handle one variable being random or its actions being swapped. The results indicate that the DDQ is the most general of the compared agents, though questions remain about whether further tuning of the interpretation between SAGE and ONA would lead to different results.

Introduction

The pursuit of artificially intelligent (AI) systems has a long history in computer science, all the way to the first half of last century (Turing, 1948). With an original aim of creating what is now referred to as artificial general intelligence (AGI), the field of AI now largely focuses on applications of well defined but limited technologies that fall short of the field's original goal. These technologies, while tremendously valuable in many use cases, always have a limited scope and appear to be dead ends when it comes to AGI research (Pennachin et al., 2007, Thórisson, 2009).

Tasks solved by modern narrow-AI systems range from relatively simple, such as balancing a pole, to quite complex, like driving cars (in predictable and favorable conditions) using a combination of techniques (e.g. deep neural networks and hand-coded controllers). Video games and other simulated tasks can be situated anywhere on this range. While extremely useful, narrow-AI systems such as reinforcement learners have clear limitations, such as that an inversion of dimensions tends to lead to negative transfer learning: take for example a typical reinforcement learner that has learned a simple task that at every iteration required it to pick one of two actions that move something in the environment in opposite directions. If the task were then inverted, i.e. the effect of the actions was swapped, the expectation is that it would take the learner much longer than otherwise to learn the new inverted task because its internal model of the environment predicts the exact opposite of the new reality.

Symbolic AIs, i.e. systems that typically use some form of reasoning and often considerable amount of effort by developers to fit to specific tasks, have largely sat at the wayside in recent years. Function-fitting learners meanwhile have been very popular, including connectionist technologies such as ANNs (artificial neural networks). These are characterized by considerable flexibility, as one type of system may be effectively applied to many different problems. While in theory anything can be modeled by function-fitting, one problem with ANN learners is that they do not have the capacity to reason. Instead,

when something changes in the world, for instance some end-effector axis is swapped 180 degrees out of phase, as in the example above, and the previous action does not work, they search the solution space randomly for what to do instead. Clearly, if the number of variables is large in the task at hand, such an approach will not scale. In a rules-based world, such as the physical one, reasoning may be done to exclude all nonsensical causes for such a reversal, possibly finding it relatively quickly.

Many kinds of reasoning have been proposed (cf. Georgeff et al., 1989; Kolodner, 2014), but the non-axiomatic reasoning theory of Dr. Pei Wang (2006) in particular shows promise when implemented in learning systems. Such systems are typically referred to as NARS (Non-Axiomatic Reasoning System). NARS uses a specially developed language called Narsese, which is entirely based on units called *terms* and relationships between them. The system derives relationships and assigns each one a so-called *truth value*[1] based on its observations, creating an internal model of its environment. It is one of only a few systems designed to learn continuously and cumulatively, a key capability in the context of AGI (Thórisson et al., 2019). A curious limitation of NARS is that it has no concept of, and cannot work in any obvious way with, numbers.

The latest implementation of NARS, ONA (OpenNARS for Applications) was designed with a pragmatic approach in mind. Among the more notable consequences of this approach is that while previous versions would continually derive and reconsider relationships, taking up valuable processing power, ONA does so only when prompted. Written in C, it runs very efficiently and provides an ideal candidate to evaluate and compare with traditional machine learners.

A platform to perform an evaluation of task performance is provided in SAGE (Simulator for Autonomy & Generality Evaluation), which is a recent answer to more popular such platforms, for instance OpenAI Gym (Brockman et al., 2016; Hernández-Orallo et al., 2017). Unlike OpenAI Gym, SAGE addresses specifically the ability of comparing a variety of learners (Eberding, 2020a). The cart-pole task, where a top-heavy pole needs to be balanced by moving a cart on which it stands, has previously been implemented with three variations and two narrow-AI learners evaluated. The task is implemented as a turn-based physics simulation that represents the environment with continuous variables. The variations of the task are designed to evaluate the traits of learning speed, proficiency, transfer learning ability, knowledge retention, and noise sensitivity. The latter three of these traits reflect how a learner reacts to changes to, or differences in, the task environment.

The NARS learning architecture is expected to lend itself particularly well to these sorts of problems. To explore this in practice we evaluate ONA with SAGE and compare its results with previous evaluations of an AC and a DDQ on the same tasks.

[1] A truth value consists of two numbers that each ranges from 0 to 1, frequency and confidence. Frequency is the amount of positive evidence divided by the amount of total relevant evidence. Confidence meanwhile is the amount of total evidence divided by one plus the amount of total evidence, confidence therefore converges to 1 with more evidence.

Motivation

Modern narrow-AI systems perform simple tasks well, but struggle with more complex ones. Even an inversion of a well-known task, where the effects of actions have been swapped, may stump modern reinforcement and deep neural net learners. Different learning architectures, in particular AGI-aspiring ones, show great promise, but are largely unexplored and untested. By systematically testing different learning architectures on a number of tasks, each designed to evaluate specific capabilities, a more holistic view of the AI landscape can be achieved. This has three clear benefits of significant importance:

1. It makes picking and choosing the right learner for a specific type of task easier, which is good for practical applications as well as applied research.
2. It benefits researchers designing learning systems (such as ONA) by making clearer whether inherent limitations of a design mean that concepts should be scrapped or revealing spots where their system can be improved.
3. More and better measurements of advancements lead to an important understanding of the progress of the field of AI as a whole and where it is headed.

Contribution

This work contains some important firsts; it is the first time an AGI-aspiring system is evaluated in SAGE, and the first time a version of NARS has been directly compared to narrow-AI learners. To our knowledge, it is also the first time such a direct comparison between an AGI-aspiring system and well-established narrow-AI learners have been compared on identical tasks.

The work involved the following:

1. Design the experimental setup to fit ONA.
2. Tuning ONA to perform at its peak, e.g. modifying its range of actions.
3. Translating the environment to Narsese. This had to be done to get the system to learn and perform well in a reinforcement learning scenario.
4. Minor modifications to the environment, including heavy discretization of environmental variables.
5. Running experiments and collecting results.
6. Interpreting results.
7. Suggesting future work.

Related Work

As mentioned in the Contributions section, this work is possibly the first direct comparison between an established AGI-aspiring system and narrow-AI systems on an identical task, and specifically a comparison of transfer learning ability.

The notion that a generally intelligent agent needs to be capable of autonomously transferring its knowledge of one task to another has been thoroughly argued (Sheikhlar et al., 2020). For measuring this and related capabilities, the SAGE (Simulator for Autonomy & Generality Evaluation) platform shows considerable promise. While other platforms such as OpenAI Gym (Brockman et al., 2016; Hernández-Orallo et al., 2017) are good for evaluating narrow-AI learners, they struggle with the wider generality assessments. Designed for evaluating different AI architectures, SAGE has been used to test and compare the transfer learning abilities and knowledge retention of both an AC (actor-critic) and a DDQ (double deep Q network) (Eberding, 2020a; Eberding et al., 2020b). Until now, however, it has not been used to evaluate an AGI-aspiring system. For the present comparisons, we use data already produced by Eberding et al. (2020b) for the DDQ and AC learners, and attempt to set ONA up in a similar way so that a comparison may be made between them.

NARS is among the longest-running and most developed AGI-aspiring projects in existence. Its strength lies in its non-axiomatic reasoning which allows it to work with incomplete information (Wang, 2006). Due to its unique architecture, it has thus far proved difficult to compare with narrow-AI systems. A new version of the NARS architecture based on the successful OpenNARS implementation, ONA provides an ideal subject to compare to narrow-AI approaches (Hammer et al., 2016, 2020).

Method

Three learners; a double deep Q neural network (DDQ), an actor-critic (AC), and OpenNARS for Applications (ONA), are each tested on distinct tasks running in SAGE, each designed to evaluate specific capabilities. These capabilities are learning speed, proficiency, transfer learning ability, knowledge retention, and noise sensitivity. Minor tweaks to the environment may be necessary to facilitate the drastically different architectures of the learners.

Learners

DDQ. Double deep Q networks are an advancement of Q-learning developed at DeepMind (Van Hasselt et al., 2016). A DDQ is an off-policy learner that combines the strengths of double Q-learners, with those of deep Q networks. The DDQ used here works with continuous numbers. It can be modified by changing its learning rate, discount factor, *epsilon*, minimum *epsilon*, *epsilon* decay, and update factor.

AC. An actor-critic is an on-policy temporal difference learner (Konda et al., 2000) that is separated into two deep neural networks; a policy structure known as an actor, that makes decisions, and an estimated value function called a critic, whose only job is to critique the actor. The AC used here works with continuous numbers. It can be modified by changing its learning rate and discount factor.

ONA. OpenNARS for Applications, recently developed by Hammer et al. (2020), is designed to be a practical and performant system built on the principles of the Non-Axiomatic Reasoning System (Wang, 2006). To maximize speed and portability, it is written in C. It learns cumulatively (Thórisson et al., 2016) even in production, and its non-axiomatic logic allows it to make decisions based on reasoning

about its observations in a way that is explainable to a human. ONA uses ampliative reasoning, in that it uses a combination of deduction, abduction, induction, and analogy to perform its tasks. It can be communicated with by reading from, and writing to, its shell interface, where it accepts logical statements in the form of Narsese. ONA can reason about temporal input, but its perception of time is linked to the number of 'thought' cycles it goes through. To represent time passing in the environment, ONA can be instructed to think for any number of cycles by sending an integer to its input stream. It has a number of configurable parameters including how often a random action is picked (motor babbling), confidence threshold for making decisions, confidence threshold for making decisions instead of motor babbling, as well as a large number of memory parameters.

Experimental Apparatus: SAGE

Built with Robot Operating System 2 (ROS 2) as its foundation, SAGE is a flexible platform for evaluating AI systems. For this purpose, it employs easily configurable simulated task environments. Learners interfacing with SAGE do so through a Python interface. It is designed on the principles of the model-view-controller-agent model, with each part running as a subprocess and communicating with the others over the network. This provides clear separation of the agents and the task environments.

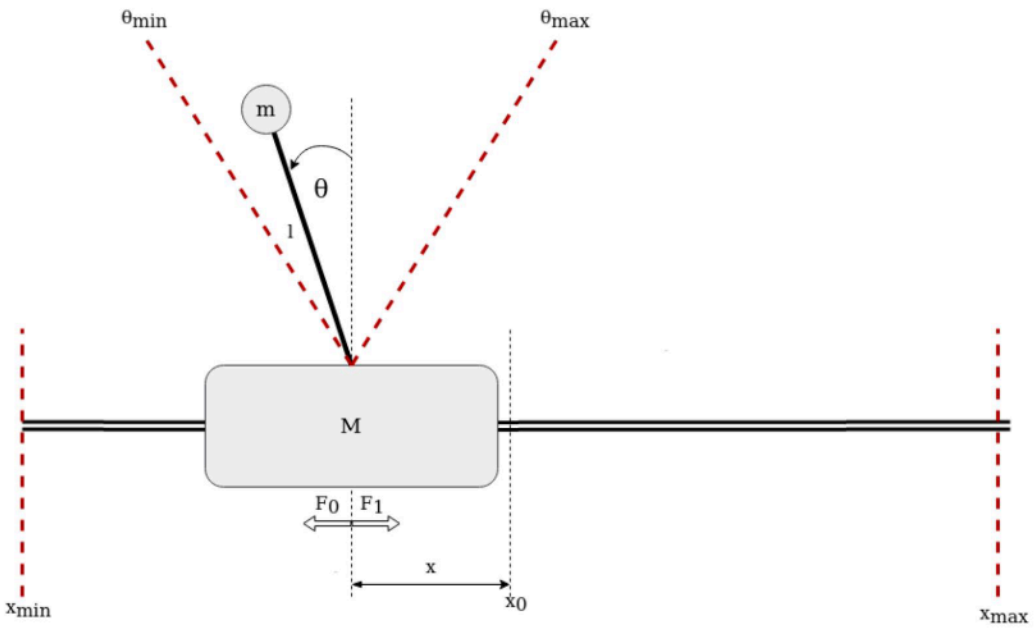

Figure 1: *A diagram of the cart-pole problem as implemented in SAGE. The variables visible to the agent are* x *(position of cart),* v *(velocity of cart),* θ *(angle of pole), and* ω *(angular velocity of pole). The actions available to the agent are* F_0 *and* F_1*, they each add a force to the cart, -10 N (left) and 10 N (right) respectively. The mass of the cart* M *is 1 kg and the mass of the weight on top of the pole 0.1 kg. The length of the pole* l *is 0.5 m. If either the angle of the pole or the cart reach past their max positions, the agent is respawned in the middle. All variables are represented in SAGE as continuous numbers. The task is turn-based, with 0.02 s being simulated in each step. (From Eberding, 2020a.)*

Tasks

The three implemented tasks are variations on the cart-pole problem (see Figure 1):

Task-1: ***Plain cart-pole***

Learn to balance a pole on a cart by moving the cart on a single dimension to the left and right.

Task-2: ***Inverted actions***

After learning **Task-1**, an agent learns to do it with the left and right actions inverted, that is left becomes 10N and right -10N. The actions are then inverted once more and the agent must perform **Task-1** again.

Task-3: ***Hidden vs. random variable***

This task involves 2 separate runs; one where *v*, the velocity of the cart, is made random, and another where *v* is hidden from the agent. With a general agent the outcomes of these two versions of this task are expected to be equivalent, as the agent should simply learn to ignore the random variable.

Independent variables

- **AI architecture and learning style**
  - *ONA*, an AGI-aspiring system.
  - *Double Deep Q Network* (DDQ), a narrow-AI.
  - *Actor-Critic* (AC), a narrow-AI.

  These are described in the section *Learners*, above.

- **Task**
  - *Cart-pole*, where an agent has the goal of balancing a top-heavy pole. Three different permutations of this task are tested, where elements of the task are changed, such as the stochasticity of measurements.

  Details of the task are described in the *Tasks* section above.

- **Exploration-exploitation** and other parameters for ONA learning control are described in the *Learners* section and further elaborated on in *Tuning ONA*, below.

- **Exploration-exploitation** settings and other parameters of the DDQ and AC testing are described in Eberding's work (2020a).

Dependent variables

- **Learning speed**, the average number of episodes until a learning agent consistently completes the task.
- **Proficiency**, the average number of episodes it takes an agent to complete a task once it has learned to complete the task consistently. In the case of holding tasks, the average number of episodes until the agent fails the task.
- **Transfer learning ability**, calculated as the learning speed divided by the average number of episodes it takes an agent to become proficient at a task once it has learned to complete the task consistently and an aspect of the task is changed.
- **Knowledge retention**, the average time it takes an agent, having learned two separate tasks in succession, to be able to complete the first task proficiently again.
- **Noise sensitivity**, the difference in learning speed and proficiency between two runs of a task: the first where a variable's value is random, the second where the same variable is hidden.

Hypotheses

Due to the lack of previous work by which to base them on, each hypothesis is based solely on intuitions derived from the architecture of the different learners. The variables and measurements are defined in the preceding sections.

1. *ONA is quicker to learn how to complete tasks than narrow-AI methods.* Since ONA can reason about what it observes in the environment, it should require fewer episodes to find a solution to the problem than DDQ and AC learners, which do not reason.

   H1: Learning speed is faster for ONA than narrow-AI learners, as measured in **Task-1**.

2. *ONA achieves a good proficiency in a larger variety of tasks than any individual narrow-AI method.* Due to the general design of its reasoning system, ONA is able to competently complete a large variety of tasks, whereas the narrow-AI methods are better suited to narrower sets of tasks. Therefore, it should achieve better performance.

   H2: ONA achieves higher **proficiency** in more tasks than any narrow-AI learner, as measured in **Task-1**.

3. *ONA takes fewer episodes to recover than narrow-AI methods when aspects of the task environment are changed.* Since ONA stores information in the form of logical statements, it is able to adapt to changes in the environment by reasoning. Narrow-AI methods meanwhile have to learn everything from scratch once an element of the environment is changed. Reversing an action will almost certainly be slow for narrow-AI learners, whereas ONA has the potential to use reasoning and do it quickly.

   H3: ONA has better **transfer learning ability** than narrow-AI learners, as measured in **Task-2**.

4. *The narrow-AI learners are likelier than ONA to unlearn (forget) a task they have previously learned.* The behavior of reinforcement learners can be unpredictable when they are learning, and they may unlearn what they had previously learned if they have been trained on a different task. ONA on the other hand should seldom, if ever, unlearn a task it has already learned, due to the way it reasons.

   H4: ONA has better **knowledge retention** than narrow-AI learners, as measured in **Task-2**.

5. *ONA will perform equally well on a task that has one random variable as it does on a task where the same variable is hidden, while narrow-AI learners perform worse with a random variable.* Because ONA can reason about the individual variables, and is built to learn from experience, it should be able to ignore a variable which does not matter. Narrow-AI learners, on the other hand, are typically forced to consider the whole set of variables as a single unified state.

   H5: ONA has less **noise sensitivity** than narrow-AI learners, as its learning speed and proficiency will be the same for both subtasks of **Task-3**.

Execution

Tasks were set up in SAGE, which allowed for fine-grained control over the properties of its task environments. The tasks were all variations on **Task-1** (cart-pole task), where an agent must balance a top-heavy pole by moving a cart the pole stands on. At every instance the controlling agent must choose whether to push the cart to the left or to the right. Each AI system runs each task 40 times, and the results for every AI-task pair are averaged and compared. From this data, the relevant dependent variables are extracted. Due to both ONA and SAGE being deterministic, repeating each task 40 times was in those cases pointless.

The tasks for the DDQ and AC were performed by Eberding (2020a) in the spring of 2020; the results presented here are from that work exclusively. The tasks for ONA were performed as part of this thesis and were performed on the same machine as those for the other learners. The machine had the following specifications:

OS: Ubuntu 18.04
CPU: AMD Ryzen 9 3900X (12 cores)
GPU: NVIDIA GeForce RTX 2060
RAM: 64GB
SSD: 250GB

To get ONA to perform at its peak, significant tuning was required. This is described in the next section.

Tuning ONA

The methodology for tuning ONA was simple: optimize for proficiency and learning speed, in that order. The only modification done to ONA itself was a removal of non-applicable built-in actions[2], leaving only *left* and *right* as output actions. While not strictly necessary, this prevented ONA from trying actions that were not available in the environment, thus speeding up learning.

Three modifications were made to the task environment. First, the option of not picking an action was added; in addition to making more sense for ONA, this made the task slightly more complex and analogous to the real world. This change was made because ONA does not pick an option every time it is prompted; as part of its learning process, it observes the environment. This made the environment unpredictable to it when an action was forced regardless. Second, the reward structure was changed such that positive feedback was only received when the pole was within 2 degrees of the upright position and negative feedback was never received. Finally the initial position of the pole was moved from the upright position to 2.9 degrees to the right, so that ONA would not receive a reward immediately after respawning. Without this modification, ONA was observed failing on purpose in order to reach the goal state. This also fixed an issue that arose when adding the no-action, as in the original spawn location the pole could balance perpetually if no movement was applied to the cart.

[2] ONA can only perform actions that it is compiled with, and does not learn new actions.

The bulk of the work, however, revolved around tuning the way the environment was represented to ONA. The translation of the environmental variables as shown by SAGE needed much iteration. For the best performance and learning speed, the four environmental variables were presented as two separate terms, one with 9 different variations, the other 12; due to its strict memory constraints, each of the environmental variables had to be heavily discretized. Three of the environmental variables had 3 possible states: negative, near-zero, and positive. The fourth variable, θ (the angle), had 4 positive states: negative, negative near-zero, positive near-zero, and positive. The angle (θ) and angular velocity (ω) were then combined into one input, as were cart position (x) and cart velocity (v).

To represent time between iterations, ONA was told to "think" for 10 cycles. This prevented it from basing its decision on something it saw in previous frames. To separate episodes from one another, ONA was told to think for 110 cycles.

Results

ONA learned **Task-1** to perfect proficiency with a very high learning speed, reaching the maximum score of 1000 iterations after only 10 episodes[3] with only minor hiccups afterwards (Figure 2). The AC and DDQ both had lower learning speed and proficiency, with the AC requiring about 200 episodes to reach its proficiency of roughly 180 iterations per episode, and the DDQ requiring over 1000 episodes to reach its proficiency of around 400 (leftmost parts of Figure 3).

ONA demonstrated its learning speed and performance again in **Task-2**, but showed no sign of any transfer learning ability. While ONA performed perfectly again after actions were inverted back to their original state, knowledge retention could not be measured because it did not learn the inverted task (Figure 4). The AC took almost 5 times as long to learn the inverted task as it did the original, demonstrating negative transfer learning. It did not retain its knowledge of the original task. The DDQ had a very fast transfer learning ability, quickly reaching the same proficiency level as before. It also demonstrated good knowledge retention, reaching proficiency levels quicker the second time actions were inverted (Figure 3).

ONA had excellent learning speed and proficiency when the cart velocity was hidden from it in **Task-3**. When the same variable was random, the learner had a slightly slower learning speed and a very low peak proficiency, only slightly improving over its initial proficiency (Figure 6). ONA was therefore very sensitive to noise. The DDQ performed similarly at the beginning in the two parts of the task, but after less than 300 episodes the plots diverged as its proficiency increased at a faster rate with the variable hidden than it did when the variable was random (Figure 5). While having less noise sensitivity than ONA, it was not immune.

[3] Referred to as epochs in figures 2-6.

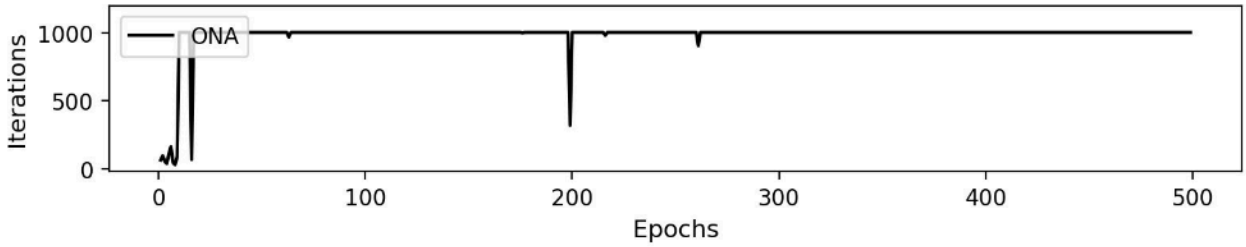

Figure 2: ONA learning speed and proficiency on Task-1 (Plain cart-pole). *ONA learns the task incredibly quickly, and, after a few hiccups, performs flawlessly. This proficiency and learning speed is considerably better than in the case of the DDQ and AC, as can be seen in the leftmost section of Figure 3.*

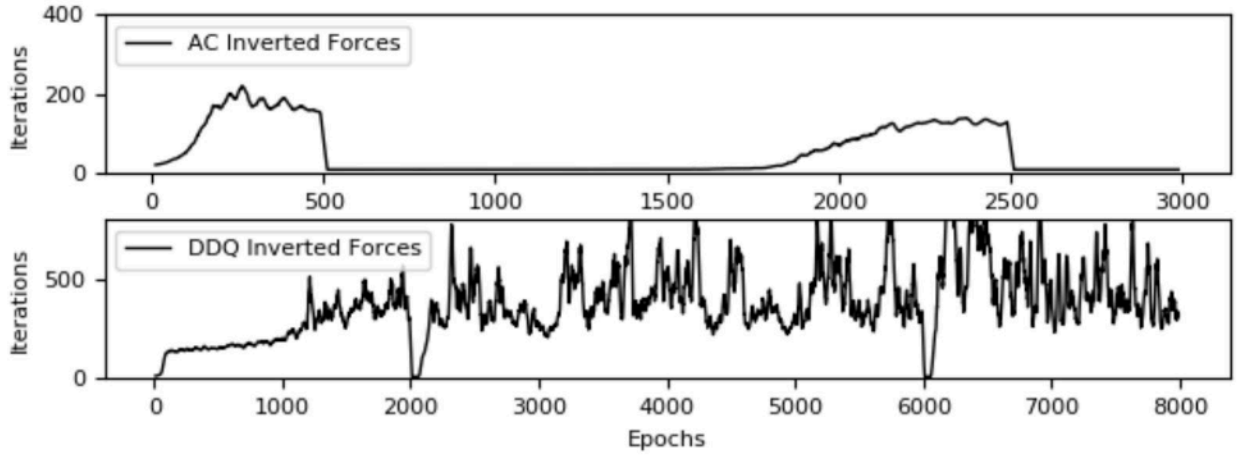

Figure 3: AC and DDQ learning speed, performance, transfer learning ability and knowledge retention on Task-2 (Inverted actions). *Sourced from Eberding et al. (2020b). The AC learned the task quickly and reached a reasonable level of proficiency, surviving for over 150 iterations per episode. After 500 episodes the actions were inverted, and the agent required more than 2000 further episodes to achieve a slightly worse proficiency. When the actions were inverted again, it became clear that the AC did not retain its ability to perform the original task. The DDQ needed over 1000 episodes to reach its proficiency, and when the actions were inverted after 2000 episodes, the learner quickly reached proficiency again. When the actions were once again inverted after 4000 further episodes, it reached proficiency fast.*

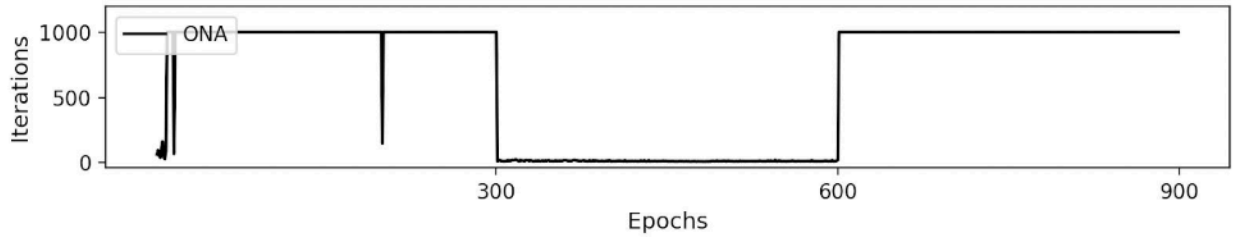

Figure 4: ONA learning speed, performance, transfer learning ability and knowledge retention on Task-2 (Inverted Actions). *As in the plain run, ONA learned the task quickly and eventually performed flawlessly, surviving for the maximum of 1000 iterations per episode. After 300 episodes, when ONA had demonstrated excellent extended performance, its actions were inverted. Left became right and right became left. ONA failed to grasp this modification of the task, seemingly learning nothing. After further 300 episodes, the actions were inverted back to their original state, and ONA instantly performed perfectly.*

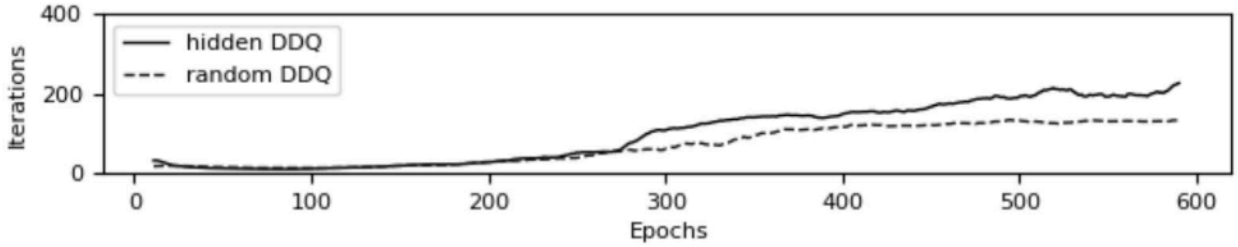

Figure 5: DDQ learning speed, performance, and noise sensitivity on Task-3 (Hidden vs. random variable). *Sourced from Eberding et al. (2020b). The DDQ performed slightly better with cart velocity hidden than it did with cart velocity random.*

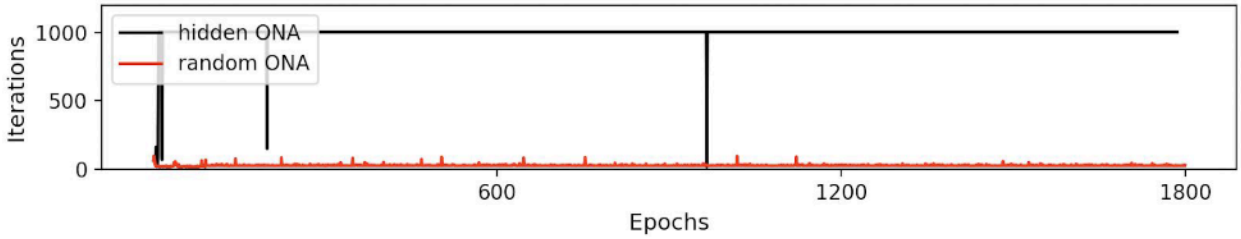

Figure 6: ONA learning speed, performance, and noise sensitivity on Task-3 (Hidden vs. random variable). *When a single variable (cart velocity) was randomized, ONA only improved slightly in the beginning, but the overall performance was abysmal. When the same variable was hidden, ONA learned quickly and performed the task near perfectly.*

Conclusion

SAGE proved to be an excellent tool for systematic comparisons between these drastically different AI architectures, but the results of the experiments were unexpected. Reviewing the hypotheses, *H1* and *H2* turned out to be true, as ONA learned **Task-1** (plain cart-pole) much faster and reached a performance far exceeding both the DDQ and AC. *H3* and *H4* meanwhile did not turn out as expected, as ONA demonstrated a clear lack of ability when actions were flipped in **Task-2** (inverted actions). Its failure to ignore the random variable in **Task-3** (hidden vs random variable), thus failing *H5*, was also unexpected, especially given that in testing ONA was sometimes observed to make decisions based only on a single observed input. Ultimately, the AGI-aspiring system performed worse than the narrow-AI systems in the tasks specifically designed to test for generality.

Investigating why exactly *H3*, *H4*, and *H5* were false and whether further tuning and tweaking of ONA and its interpreter will show improvements is a clear immediate next step. It may be that optimizing ONA only for learning speed and proficiency undercut its full capabilities. Exploring ways to let ONA understand numbers other than heavy-handed discretization is likely to considerably increase its utility.

References

1. Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *arXiv preprint arXiv:1606.01540*.

2. Eberding, L.M. (2020a). Task-environments for evaluating learning and autonomy in AI architectures. MSc thesis, *Institute for Photogrammetry and Geoinformation*, Leibniz University, Hannover.

3. Eberding, L.M., Sheikhlar, A., Thórisson, K.R., & Andrason, S.P. (2020b, June). SAGE: Task-Environment Platform for Autonomy and Generality Evaluation. In *International Conference on Artificial General Intelligence*. Springer.

4. Georgeff, M., & Ingrand, F. (1989). Decision-Making in an Embedded Reasoning System. In *International Joint Conference on Artificial Intelligence (IJCAI)*. pp. 972-978.

5. Hammer, P., & Lofthouse, T. (2020, June). 'OpenNars for Applications': Architecture and Control. In *International Conference on Artificial General Intelligence*. Springer.

6. Hammer, P., Lofthouse, T., & Wang, P. (2016, July). The OpenNARS Implementation of the Non-Axiomatic Reasoning System. In *International Conference on Artificial General Intelligence* (pp. 160-170). Springer.

7. Hernández-Orallo, J., Baroni, M., Bieger, J., Chmait, N., Dowe, D.L., Hofmann, K., Martínez-Plumed, F., Strannegård, C., & Thórisson, K.R. (2017). A New AI Evaluation Cosmos: Ready to Play the Game?. *AI Magazine*, *38*(3), 66-69.

8. Kolodner, J. (2014). *Case-based Reasoning*. Morgan Kaufmann.

9. Pennachin, C., & Goertzel, B. (2007). Contemporary Approaches to Artificial General Intelligence. In *Artificial General Intelligence* (pp. 1-30). Springer, Berlin, Heidelberg.

10. Sheikhlar, A., Thórisson, K.R., & Eberding, L. (2020, June). Autonomous Cumulative Transfer Learning. In *International Conference on Artificial General Intelligence*. Springer.

11. Thórisson, K.R., Bieger, J., Li, X., & Wang, P. (2019). Cumulative Learning. In *International Conference on Artificial General Intelligence*. pp. 198-209. Springer.

12. Thórisson, K. R. (2009, October). From Constructionist to Constructivist AI. In *2009 AAAI Fall Symposium Series*.

13. Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-Critic Algorithms. In *Advances in Neural Information Processing Systems* (pp. 1008-1014).

14. Turing, A. (1948). Intelligent machinery. *The Essential Turing*, 395-432.

15. Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep Reinforcement Learning with Double Q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*.

16. Wang, P. (2006). *Rigid Flexibility: The Logic of Intelligence*. Springer.