JMLR: Workshop and Conference Proceedings 159:49–63, 2021 International Workshop on Self-Supervised Learning

# Comparison of Machine Learners on an ABA Experiment Format of the Cart-Pole Task

**Leonard M. Eberding**[1] LEONARD20@RU.IS
**Kristinn R. Thórisson**[1,2] THORISSON@RU.IS
**Aadhav Prabu**[2] AADHAVPRABU03@GMAIL.COM
**Shubh Jaroria**[2] SHUBH.JARORIA@GMAIL.COM
**Arash Sheikhlar**[1] ARASH19@RU.IS

[1] *Center for Analysis and Design of Intelligent Agents, Reykjavik Univ., Iceland*
[2] *Icelandic Institute for Intelligent Machines, Reykjavik, Iceland*

**Editors:** K. R. Thórisson and Paul Robertson

## Abstract

Current approaches to online learning focus primarily on reinforcement learning (RL) – algorithms that learn through feedback from experience. While most current RL algorithms have shown good results in learning to perform tasks for which they were specifically designed, most of them lack a level of generalization needed to use existing knowledge to handle novel situations – a property referred to as autonomous transfer learning. Situations encountered by such systems which were not present during the training phase can lead to critical failure. In the present research we analyzed the autonomous transfer learning capabilities of five different machine learning approaches – i.e. an Actor-Critic, a Q-Learner, a Policy Gradient Learner, a Double-Deep Q-Learner, and OpenNARS for Applications. Following a classic ABA experimental format, the learners were all trained on the well-known cart-pole task in phase A-1, before strategic changes to the task were introduced in phase B, consisting of inverting the direction of control of the cart (move-left command moved the cart to the right and vice versa), as well as the introduction of noise. All analyzed learners show an extreme performance drop when the action command is inverted in phase B, resulting in long (re-)training periods trying to reach A1 performance. Most learners do not reach initial A1 performance levels in phase B, some falling very far from them. Furthermore, previously learned knowledge is not retained during the re-training, resulting in an even larger performance drop when the task is changed back to the original settings in phase A2. Only one learner (NARS) reached comparable performance in A1 and A2, demonstrating retention of, and return to, priorly-acquired knowledge.

**Keywords:** Artificial Intelligence, Transfer Learning, Online Learning, Task Analysis, Generalization

## 1. Introduction

One of the biggest challenges in artificial intelligence (AI) is cumulative learning Thórisson et al. (2019). While proposals to this challenge exist, most focus on reinforcement learning (RL). The learning strategy in RL is letting the machine learning (ML) algorithm inter-

act with the environment, getting repeatedly a belated 'reward' that indicates whether a particular action path chosen by the learner was fruitful for the task at hand or not. RL approaches, however, lack the ability to adapt to changes in the environment which were not experienced by the agent during the training phase (Eberding et al., 2020), and all variables must be defined beforehand. The ability to adapt to changes in the environment and/or task, however, including dealing with new data, is of utmost importance for any self-supervised learner, especially those targeting general machine intelligence (GMI).

While research on transfer learning has increased in recent years (cf. Taylor and Stone (2009); Da Silva and Costa (2019)), it is not clear to what extent the numerous learning approaches (well-known and lesser-known) handle changes to tasks already learned. Until recently, researchers aiming for more flexible and general machine learning had seen limited success compared to machine learning methods with a more targeted learning focus. However, some recent advances in general machine intelligence have demonstrated progress on tasks requiring cumulative (incremental, life-long) learning and reasoning (cf. Wang (2006); Nivel et al. (2013)).

No accepted theory of transfer learning exists as of yet (cf. Sheikhlar et al. (2020)). An understanding of how the various machine learning methods developed to date handle transfer learning currently could provide valuable insight into how they might be improved, and could also serve as an important baseline for new developing learning approaches. A comprehensive comparison across learners, tasks, and task modifications would be a logical step, but to the best of our knowledge, no such database exists.

Most evaluations of ML systems involve a single unique task hand-tuned for the system at hand; comparing results from such work, where both learners and tasks are created under a wide variety of assumptions, is not feasible. A proper comparison should hold key task parameters steady between tasks for each learner, but few papers present systematic comparisons of multiple learners on comparable or identical tasks. In this paper we describe the evaluation of five learners on the well-known 'cart-pole task' (an upright pole balanced on a moving cart). The learners we compared were (1) a Policy-Gradient learner (PG), (2) a Q-Learner, (3) a Double-Deep Q-Network learner (DDQN), (4) an Actor-Critic learner (AC), and (5) the GMI-aspiring system OpenNARS for Applications (ONA), a learner based on the principles of the Non-Axiomatic Reasoning System (NARS).

We tested three different test conditions of the cart-pole task. We introduced a change to the environment by swapping the left and right movement of the cart (i.e. inverting the direction of the right and left commands). The agents were first trained on a 'normal' setup (A1). When they had learned the task we inverted the forces applied by the agent's actions to the cart by 180° (B), after which we later reinstated the original setup (A2). We present the results of this ABA version of the cart-pole task for the learners and draw conclusions about their abilities to cope with sudden changes in a learned task.

The paper is structured as follows: In the next section we discuss related work. Then we give an overview of the methodology used to evaluate the learners on the cart-pole task including (1) the exact set-up of the cart-pole dynamics, (2) the different test conditions applied during evaluation, and (3) the different learners and adjustments done to the task to suit the agent's interfaces. In the following section we show the evaluation results for different learners on the three test conditions before we discuss the results in the next section. Lastly we will conclude by giving an overview about what we believe would enable

self-supervised systems to cope with changing environments where novelty can be introduced at any moment.

## 2. Related Work

Evaluation is important in any engineering field that has a clear idea of its targets. The clarity of research targets in AI has fluctuated somewhat throughout the decades, and as a result, the perceived gold standards have shifted quite a bit since the field was founded. Throughout its history there has been a focus on frameworks for system comparison, starting as early as the late 70s with chess becoming a primary measuring stick for many. This was significantly reduced when Deep Blue won against Kasparov in 1997, as many felt that that threshold had been crossed and could no longer serve as a target. In the early naughts the General Game Playing Competition (Genesereth et al., 2005) picked up where chess tournaments left of. The RoboCup (Kitano, 1998) was another framework that served as a boost to real-world AI development, starting in 1997 AI algorithms were used to develop soccer strategies for soccer playing robots.

When reinforcement learning became a major field of AI research new evaluation methods were introduced for these algorithms. Today a two-phase assessment is done, where in the first phase a learner is trained on a task, and in the second phase it is tested on a similar, but in some ways different, task (Whiteson et al., 2010). For example Bellemare et al. (2013) designed a platform called ALE for this purpose. Agents are trained on a set of Atari games before they are evaluated on a different set of games. In a similar vein, Johnson et al. (2016) introduced the Malmo platform that is focused on testing of AI systems on different scenarios of the Minecraft game. Juliani et al. (2018) examine the Unity game engine, which provides a physical, realistic environment, for testing ML-agents. Crosby et al. (2020) propose Animal-AI testbeds and outline an environment and a paradigm for AI system evaluation inspired by the the cognition of animals.

While the previous described evaluation platforms aim to evaluate the generality of the algorithms (i.e. the ability to cope with novelty), the competitions had been limited to a set of tests which focus on the learners' differences rather than on physical properties of the task-environments and how a change in the environment's structure influences the learning process. Additionally all evaluation platforms listed above focus exclusively on reinforcement learners, neglecting the existence of other online learners completely.

While the described evaluation platforms were noteworthy efforts, here we are concerned with deeper issues of comparison that rather requires a more fine-grained analysis than can be provided by the above frameworks. Essentially, we call for a detailed analysis of ML performance across a wide range of identical tasks. Unfortunately, our perceived need for such data has not been widely shared by the AI research community at large: The big question of *how the various machine learning approaches compare on identical tasks* and therefore of *how changes to tasks and learning procedures affect learning and performance* has not been systematically addressed to any satisfactory extent to date, to the best of our knowledge. While the present work represents only a tiny step in that direction, we are hopeful that we will see a new trend in the coming years where standardized tasks and tests become the norm. For this purpose we designed previous to this study an evaluation

platform called SAGE (simulator for autonomy and generality evaluation) which has been used in this work (Eberding et al., 2020).

## 3. Methodology

In this section we present the task-environment used for evaluation, as well as the learners evaluated - especially changes to the task-environment necessary to suit the learner.

### 3.1. The Cart-Pole Task

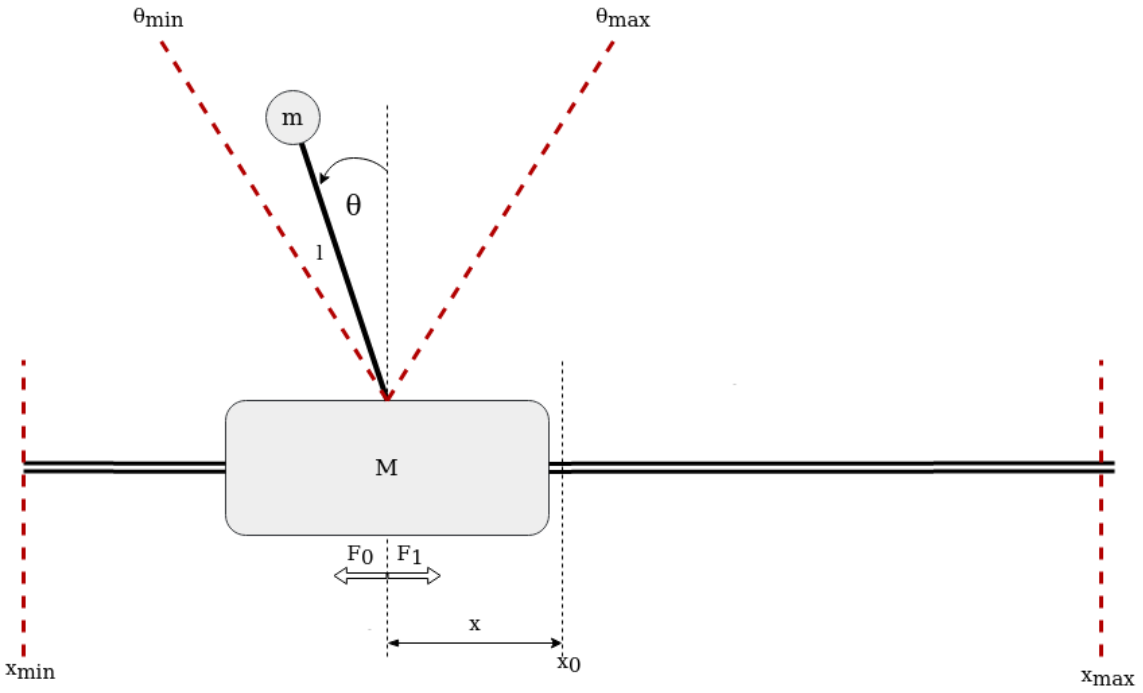


Figure 1: Illustration of the cart-pole task. The goal is to keep the pole upright for as long as possible. In our setup the cart is positioned on a frictionless slide and the pole is connected to the cart by an unactuated, frictionless joint. **m**: Pole weight, $\theta_{min}$: Maximum possible angle of pole to the left, $\theta_{min}$: Maximum possible angle of pole to the right, $\Theta$: Angle of pole from the vertical, M: Cart, $X_{min}$: The leftmost position possible for the cart, $X_{max}$: The rightmost position of the cart possible, $F_0$: Force pushing the cart left, $F_1$: Force pushing the cart to the right, $X_0$: Center of the cart slide.

The cart-pole task is a well-known task for evaluating different machine learners on. Thresholds, observations, conditions, variable values, and initial values have been adopted here as they presented in the OpenAI Gym implementation (Brockman et al., 2016). Figure 1 shows an illustration of the cart-pole setup. For our evaluation the following setup and conditions were applied:

- **Static variables.** Mass of the cart: $M = 1.0$ kg; Mass of the pole: $m = 0.1$ kg; Length of the pole: $l = 0.5$ m; Gravity: $g = 9.81\ \frac{m}{s^2}$
- **Manipulatable variables/ actions.** The agent can apply one of two forces: $F_0$ = -10 N; $F_1 = 10$ N. In the case of ONA, a third action was added due to special constraints of the learner: $F_2 = 0$ N (see Discussion section).
- **Observable variables.** The angle of the pole $\theta$, the distance traveled by the cart $x$, the angular velocity of the pole $\omega$, as well as the linear velocity of the cart $v$ are observable to the agent.
- **Iteration.** An iteration is a single time-step as simulated by the evaluation platform. We used time-steps of 0.02 s.
- **Epoch.** A single trial of the cart-pole before failure occurs and the environment is reset to its initial conditions.
- **Failure condition.** Whenever either $abs(\theta) > 12°$ or $abs(x) > 2.4\ m$ the task counts as failed, the epoch is ended and the task-environment is reset to its initial conditions.
- **Success condition.** The task is regarded as completed successfully, if the learner achieves an average of 195 iterations per epoch over 100 consecutive epochs.
- **Reward.** For the RL algorithms a reward was provided by the environment of either +1, if the pendulum was within the specified failure constraints, or -1, if the task was failed and an epoch ended.

During the task's execution the cart-pole dynamics are applied in a discretized manner for each iteration. For each state transition for a single iteration the following transition function was used:

**Input:** $x$, $v$, $\theta$, $\omega$, $F$ $M$, $m$, $l$, $g$
**Output:** $x_{new}$, $v_{new}$, $\theta_{new}$, $\omega_{new}$
$\partial t = 0$
**while** $\partial t < 0.02$ **do**
  $\dot{\omega} = \frac{(M+m)\cdot g\cdot sin(\theta) - cos(\theta)\cdot\left(F+m\cdot l\cdot\omega^2\cdot sin(\theta)\right)}{\frac{4}{3}\cdot l\cdot(M+m) - m\cdot l\cdot cos^2(\theta)}$;
  $a = \frac{F+m\cdot l\cdot\left(\omega^2\cdot sin(\theta) - \dot{\omega}\cdot cos(\theta)\right)}{M+m}$;
  $v_{new} = v + a\cdot 0.001$;
  $x_{new} = x + v\cdot 0.001$;
  $\omega_{new} = \omega + \dot{\omega}\cdot 0.001$;
  $\theta_{new} = \theta + \omega\cdot 0.001$;
  $\partial t += 0.001$
**end**

**Algorithm 1:** Transition functions to transition the state of the environment $(x, v, \theta, \omega)$ into a new state $(x_{new}, v_{new}, \theta_{new}, \omega_{new})$ given an action $F$ and static variables of the cart-pole setup. For a more precise calculation of the new state a single iteration was divided into 20 sub-iterations used to better approximate the true differential equation.

### 3.2. ABA Setup of the Cart-Pole Task

The environment's dynamics are changed during the evaluation to introduce novelty, different from original training. For this we decided to use force inversion to create an ABA version of the cart-pole task with the following three phases:

A1: The agent trains to perform the cart-pole task with actions $F_0 = +10$ N, and $F_1 =$ -10 N.

B: The actions are inverted to $F_0 =$ -10 N, and $F_1 = +10$ N.

A2: The original action setup reinstated ($F_0 = +10$ N, and $F_1 =$ -10 N).

The change between each phase has been done under three different conditions. These conditions are as follows:

**'Simple':** The forces are (re-)inverted after the agent reaches the success condition (195 iterations per epoch in average over 100 epochs).

**'InvSIG':** An additional observations is introduced: Signal indicating whether current phase is A or B (i.e. whether inverted or not).

**'FixedEPOCH':** Inversion (or re-inversion) is done after a fixed number of episodes.

### 3.3. Learners

Five different learners were evaluated including four reinforcement learners (RL) and one general machine intelligence (GMI) aspiring system. The learners are

- Q-learner, a model-free, off-policy RL that learns its actions' values in every state (Watkins and Dayan, 1992),
- Policy-Gradient learner (PG), an on-policy RL based on policy optimization (Silver et al., 2014),
- Double-Deep-Q-Network learner (DDQN), a Q-learner that uses max operation decomposition to solve the overestimation issue caused by bootstrapping (Van Hasselt et al., 2016),
- Actor-Critic (AC), an on-policy RL that learns both value and policy functions simultaneously (Konda and Tsitsiklis, 2000), and
- Open-NARS-for-Applications (ONA), an AGI-aspiring system (Hammer and Lofthouse, 2020) based on the principles of the Non-Axiomatic-Reasoning-System (Wang, 1995).

Table 1: Hyperparameters of the different learner. Dash ("—") indicates that such a hyperparameter does not exist (ONA has none of the hyperparameters shown); LR: learning rate; ER: exploration rate; DF: discount factor.

| **Learner** | **LR** | **LR-decay** | **Min LR** | **ER** | **ER-Decay** | **Min ER** | **DF** |
|---|---|---|---|---|---|---|---|
| PG | 0.05 | — | — | — | — | — | 0.99 |
| Q | 1.0 | 0.999 | 0.1 | 1.0 | 0.999 | 0.1 | 1.0 |
| DDQN | 0.001 | — | — | 1.0 | 0.99 | 0.01 | 0.99 |
| AC | 0.00075 | — | — | — | — | — | 0.99 |
| ONA | — | — | — | — | — | — | — |

**Discretization.** Neither the Q-learner nor ONA are able to cope with a continuous state-space, making discretization of observable variables necessary. The other three learners were evaluated in the continuous state-space. For discretization we chose the following setup:

- **Q-learner:** The state space variables are bucketed in different sized buckets.

  $\theta$: 12 evenly sized buckets ranging from $-0.21\ rad$ to $+0.21\ rad$ ($-12°$ to $+12°$),

  $x$: 6 evenly sized buckets ranging from $-2.4\ m$ to $+2.4\ m$,

  $v$: 6 evenly sized buckets ranging from $-20\ \frac{m}{s}$ to $20\ \frac{m}{s}$,

  $\omega$: 6 evenly sized buckets from $-6\frac{rad}{s}$ to $+6\frac{rad}{s}$.

- **ONA:** The following discretization was used:

  $\theta$: 4 Buckets: $[-0.21\ rad,\ -0.02\ rad[$, $[-0.02\ rad,\ 0.0\ rad[$, $[0.0\ rad,\ 0.02\ rad]$, $]0.02\ rad,\ 0.21\ rad]$

  $x$: 4 Buckets: $[-2.4\ m,\ -0.2\ m[$, $[-0.2\ m,\ 0.0\ m[$, $[0.0\ m,\ 0.2\ m]$, $]0.2\ m,\ 2.4\ m]$

  $v$: 4 Buckets: $[-\infty\ \frac{m}{s},\ -0.3\ \frac{m}{s}[$, $[-0.3\ \frac{m}{s},\ 0.0\ \frac{m}{s}[$, $[0.0\ \frac{m}{s},\ 0.3\ m]$, $]0.3\ \frac{m}{s},\ \infty\ \frac{m}{s}]$

  $\omega$: 4 Buckets: $[\infty\ \frac{rad}{s},\ -0.01\ \frac{rad}{s}[$, $[-0.01\ \frac{rad}{s},\ 0.0\ \frac{rad}{s}[$, $[0.0\ \frac{rad}{s},\ 0.01\ \frac{rad}{s}]$, $]0.01\ \frac{rad}{s},\ \infty\ \frac{rad}{s}]$

**Hyperparameters.** For each learner the hyperparameters (if any are available) were tuned such that the task could be reliably performed. However, no excessive hyperparameter optimization took place. This was not done due to the idea that these tests are not done to evaluate the general performance of the learners on the cart-pole task, but rather to test their ability to cope with changes. Also, the hyperparameters of the different learners differ, due to the different approaches and algorithms.

Especially important for the Q-Learner and the DDQN-Learner is that neither learning-rate nor exploration-rate was reset when (re-)inversion took place. This is done to evaluate the learners "as is" instead of "perfectly tuned for this particular task variation".

## 4. Results

In the following we show the evaluation results of the different learners on the different test conditions. Where not differently stated each evaluation shown is the median of the performance over 30 independent trials, together with the upper and lower quartile presented. The plots all show the results averaged using a rolling window mean with a window width of 5 epochs.

### 4.1. Inversion After Success ('Simple')

All five learners were evaluated on the **Simple** test condition. ONA was only done 5 times, however the deterministic behavior of ONA supports such a low sample size. No quartiles are shown for ONA. Figure 2 and 3 show the plots for this test condition.

### 4.2. Additional Observable to Show Task Phase ('InvSIG')

Three of the learners were evaluated on the **InvSIG** test condition, the DDQN, the AC, and the ONA system. ONA again was only done 5 times. For the InvSIG testing an additional observable was introduced which shows the phase of the task. Again the inversion took place when the success condition was met. Figure 4 and 5 show the plots for the InvSIG test condition.

### 4.3. Inversion After Fixed Number of Epochs ('FixedEPOCH')

The tests for the learners' re-learning performance in ABA after fixed number of epochs was only done for the AC reinforcement learning algorithm. In this case, A1 and A2 phases have 250 epochs length. However, the B phase has different length in four different tests. As can be seen, the longer the leaner is trained in phase B, the worse the performance becomes in phase A2. Figure 6 shows the plots for the FixedEPOCH test condition.

## 5. Discussion

The Q-learner and DDQN are the algorithms that get better after every inversion, reaching the success condition faster than in the previous phase. The reason is that these two are off-policy learners that have continuous exploration, which speeds up the training and re-training process. On the other hand, on-policy learners such as AC and PG only try to optimize the initial policy that is given to them upfront and thus, does not change their behavior when there is a phase shift. Another reason for the Q-learner showing superior adaptivity can be the results of the heavy discretization done on the state-space. By doing such a discretization we, the designers, bring knowledge into the agent that other learners do not have (e.g. the resolution and form of the discretization). The ONA learner, however, starts its learning using a semi-random exploration, similar to epsilon-greedy method in Q-learning, and then increases the exploitation based on the current solution until it reaches a deterministic satisficing solution, rather similar to on-policy RL algorithms' policy learning style.

Both the PG and the AC algorithms show long re-training times in comparison to the original training. Especially when looking at the **FixedEPOCH** test condition we can see

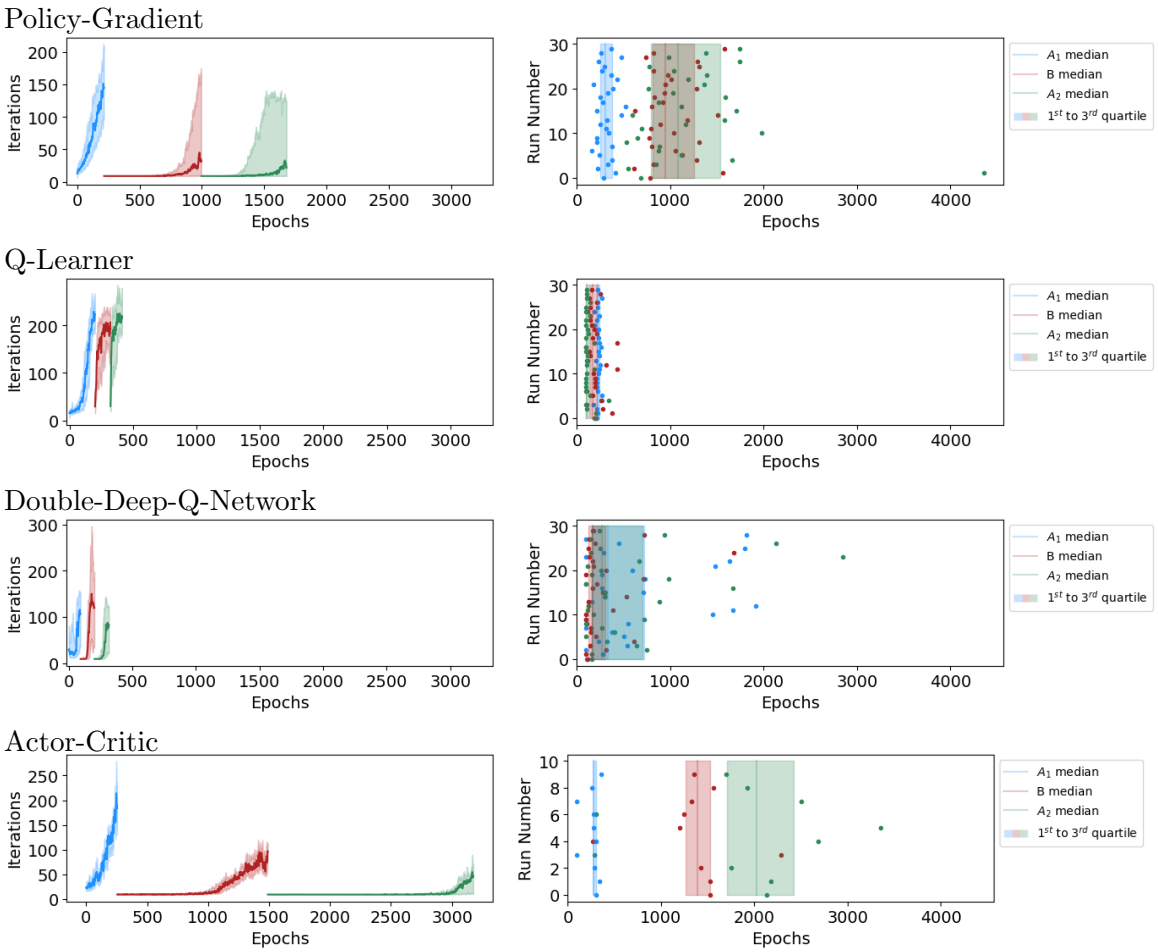


Figure 2: The four reinforcement learning algorithm evaluated on the ABA task under the **Simple** test condition. Left: Median and area between upper and lower quartile for the set of runs done for each learner; Right: Epochs needed until the success condition was met for each of the different runs per learner. For the actor-critic only 10 runs were done. Line shows median of runs, shaded area represents 50% of the data points inside the area. From top to bottom: PG learner, Q-learner, DDQN learner, and AC learner. The x-axis for all learners have been adjusted to have the same scale. Blue: A1 phase; Red: B phase after first inversion; Green: A2 phase after reinstating the original setup.

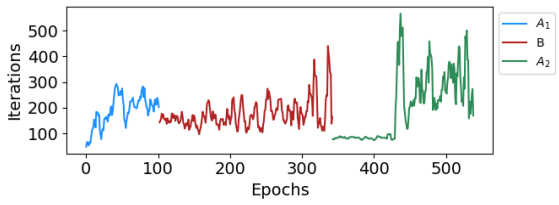


Figure 3: ONA evaluated under the **Simple** test condition. Five runs are shown. No upper quartile, lower quartile, or epochs per run are shown due to the deterministic behavior of ONA.

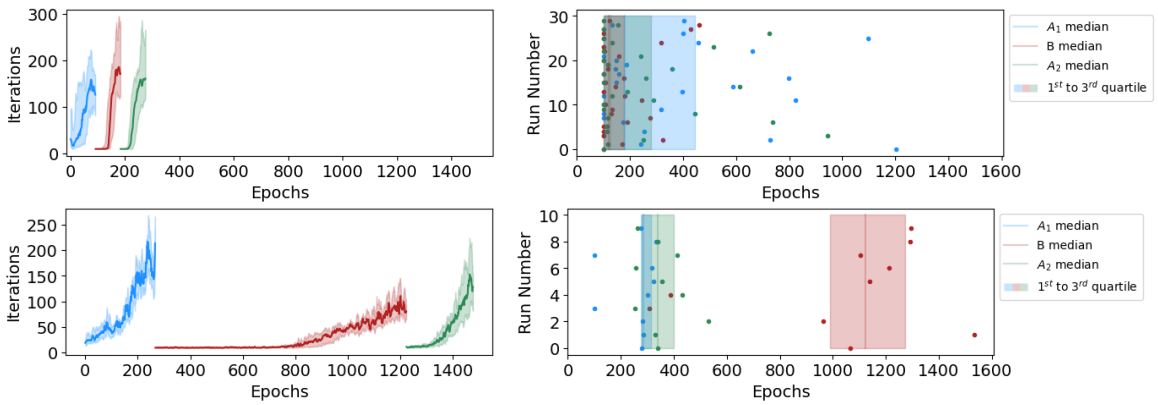


Figure 4: The DDQN (top) and AC (bottom) reinforcement learning algorithms evaluated on the ABA task under the **InvSIG** test condition. Inversion took place when the agent achieved the success condition (i.e. achieve an average of ate least 195 iterations per epoch over 100 consecutive iterations). See figure 2 for further description of the plots.

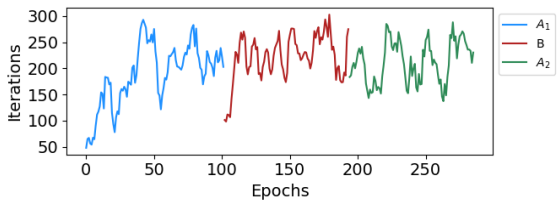


Figure 5: ONA evaluated under the **InvSIG** test condition. Five runs are shown. No upper quartile, lower quartile, or epochs per run are shown due to the deterministic behavior of ONA.

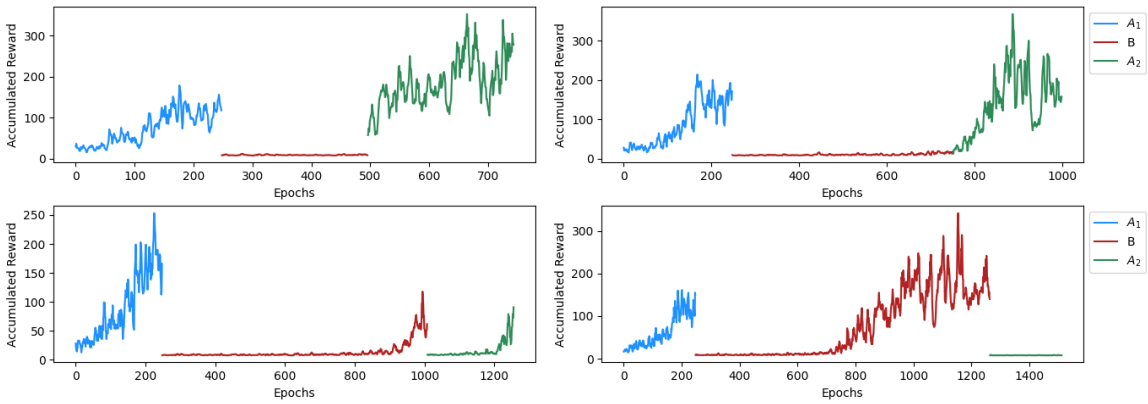


Figure 6: In FixedEPOCH evaluation of AC reinforcement learning algorithm, the A1 and A2 phases take 250 epochs for all four tests, but the B phase takes 250, 500, 750, and 1000 epochs for tests 1, 2, 3 and 4, respectively.

that the amount of time spent on training under one phase heavily influences the speed of learning in the following phase. This can come from the on-policy approaches in both of the algorithms. After inversion the previously acquired policy must first be "unlearned" before new, correct knowledge can be accumulated. When the inversion phase is short more of the original policy remains unchanged making the following $A_2$ phase easier to perform for the agent.

Comparing the results from the **Simple** with the **InvSIG** test condition shows that providing information about the inversion improves performance for the DDQN, the AC, and ONA (see figures 7,8,9). In case of the Q-Learner and PG-Learner, both failed in reaching the success condition in the InvSIG setup. This may be due to inappropriate settings of hyperparameters (on all learners, these were not changed between different test conditions).

The original goal for this study was to better understand what mechanisms are necessary for a machine learning system to handle environments with changing conditions, where novelty can occur at any moment. We conclude that the following four points need to be taken into account when designing such a system:

*Identification of differences between expected and actual outcome.* To enable a system to cope with novel situations, a necessary mechanism must be the ability to identify novelty. This means that an agent must identify when previously acquired knowledge is not suitable for a situation. This can be done by identifying differences in the expected and the actual outcome. For this to work, however, the agent must first be able to

*predict future states*, not just expected rewards. A system must be able to learn the environment's dynamics and predict a future state after interacting with the environment by itself: It is the contrast with prediction that provides a learner information about novelty. Only then there exist an expected outcome of the intervention which can be compared to the actual outcome. If such a mechanism exists, we argue that the agent must be able to re-learn the B phase of our used ABA version of the cart-pole task at least as fast as the A1 phase. This claim can easily supported. The worst case after identification of novelty is that the learner needs to learn from scratch (i.e. it should learn as fast as before since the task is fundamentally the same). If the learner can use previously acquired knowledge it can become even faster in the B and A2 phases. This brings us to the next point we want to make:

*Similarity detection helps in coping with novelty.* A system which can not only identify a change in the environment's dynamics, but can also identify similarities between two phases has a huge advantage in performing the ABA version of the cart pole task. Since the underlying dynamics are the same a system could easily identify that the only difference is the action taken (state transitions etc. are the same just mirrored) and therefore immediately use the full knowledge base to perform in the inverted task. We described such a similarity description in the past (see (Sheikhlar et al., 2020)) and argue that with such a mechanism learning of novel situations could be sped up drastically.

Lastly, we want to focus on two of the hyperparameters, learning-rate and exploration-rate. We argue that

*a system designed to learn cumulatively may not only rely solely on simple values to define the learning- and exploration rate.* Using passage of time as a measure of importance

for any observation (as is the case when learning rate is decayed over time) plays a huge role if change is only encountered late in a learner's lifetime. Additionally, exploration by pseudo-random action selection can be unfruitful if the environment changes repeatedly. When a novel situation occurs in an agent's lifetime exploration might be of higher priority, than exploitation (since only little about the novel situation is known). Exploration rate should therefore be a similarity dependent variable which changes depending on the situation the agent finds itself in. Another related point is that in narrow learners, that are unequipped to handle multiple tasks, a global setting for exploration may suffice. However, general learners must have mechanisms for targeted exploration, based on missing knowledge or detection of incorrect knowledge (for instance, through incorrect prediction of the outcome of actions).

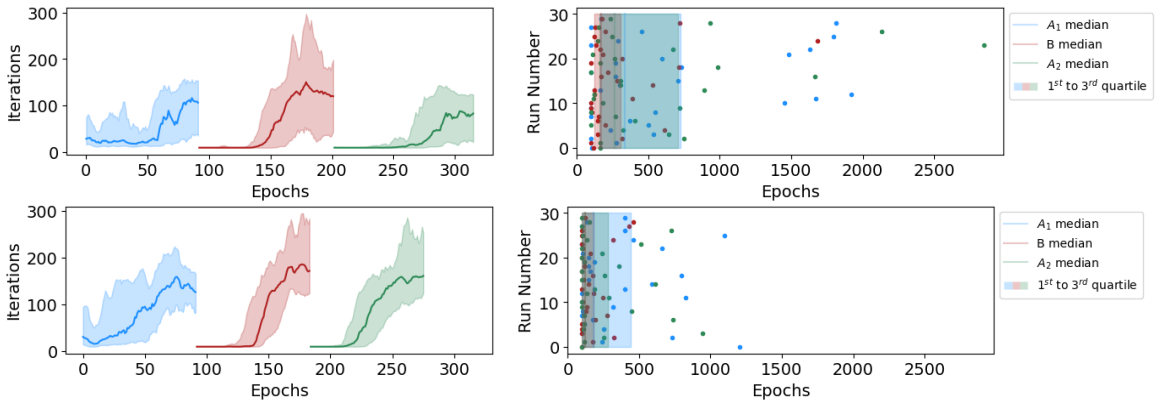


Figure 7: Comparison between the performance of the DDQN learner in Simple (top) and InvSIG (bottom) conditions. See caption of figure 2 for further information.

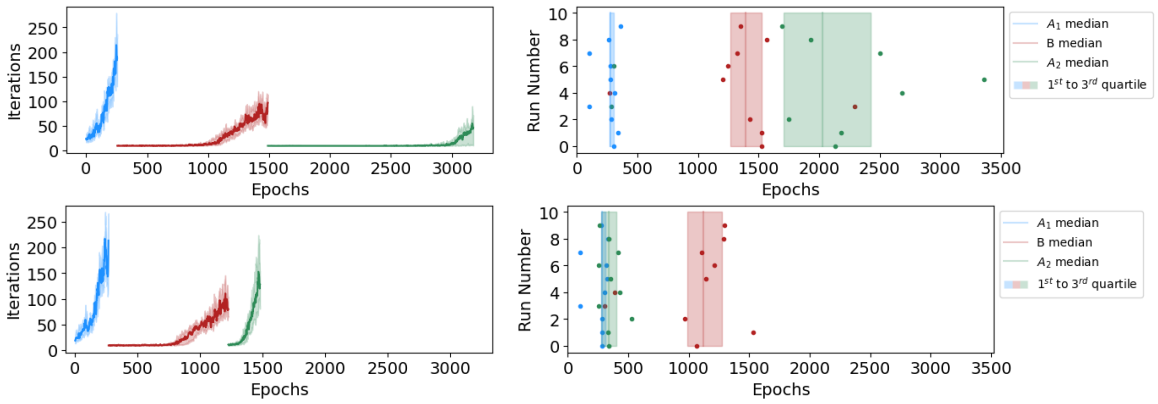


Figure 8: Comparison between the performance of AC reinforcement learning in Simple (top) and InvSIG (bottom) conditions.

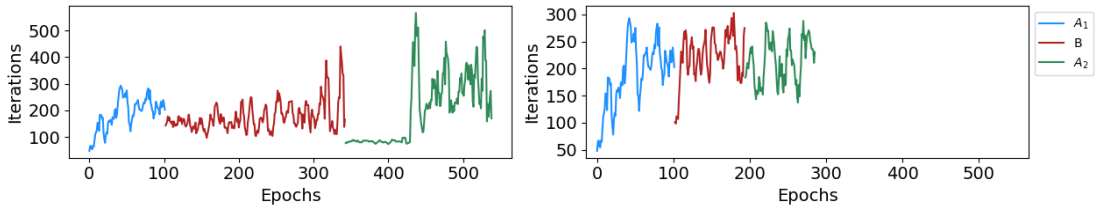


Figure 9: Comparison between the performance of ONA learner in Simple (left) and InvSIG (right) test conditions. x-Axis are on the same scale.

*How an AI system/algorithm can quickly switch to a learning mode which builds new policy conditioned on a context change rather than unlearning the existing policy?*
This requires systems that contextualize knowledge in relation to the changes in the environment. The AERA system is an example of such systems that loads relevant models into its working memory when it discovers a context change (Nivel et al., 2013). AERA creates preconditions for the usage of causal models it learns. When its models receive more pieces of positive evidence, their success rate increases. If a certain threshold of positive evidence is reached the models are marked as *strong*. Strong models are those where if they fail a context change is expected rather than an error of the model. If such models exist and a sudden change in the environment occurs, leading to misbehavior of these models, the system does not remove them. Instead, it keeps the models in a different group in its memory and reloads them when the context is switched back. In other words, even if the change is not signaled to the agent, AERA finds out that the context has changed, since it assumes its strong models must not fail all of a sudden drastically. In this way, the system does not have to relearn every model from scratch when a change takes place. The implementation of AERA on a dynamic task, with an eye to our ABA evaluation strategy, is the next phase of this research.

## 6. Conclusion

In this work we present an in-depth analysis of different learners doing a simple task like the cart-pole. By testing the different learners on the ABA-version of the task we were able to get a deeper insight into the (dis-)advantages of the different learners. Using an inversion of the action space showed how reinforcement learning falls short when tackling previously untrained scenarios. From this we drew conclusions on necessary properties a self-supervising system should have to manage novelty and, in the long run, be able to be deployed in highly complex environments like the real world.

Obvious next steps include creating a larger set of tasks that cover a wider range of task types, which cover continuity, causal relations, similarities of different levels, and higher levels of complexity in the observation and action space, as well as the dynamics of the environment.

**Acknowledgments**
This work was supported in part by a grant from Cisco Systems Inc., the Department

of Computer Science at Reykjavik University, and the Icelandic Institute for Intelligent Machines.

## References

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Matthew Crosby, Benjamin Beyret, Murray Shanahan, José Hernández-Orallo, Lucy Cheke, and Marta Halina. The animal-ai testbed and competition. In *NeurIPS 2019 Competition and Demonstration Track*, pages 164–176. PMLR, 2020.

Felipe Leno Da Silva and Anna Helena Reali Costa. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64:645–703, 2019.

Leonard M Eberding, Kristinn R Thórisson, Arash Sheikhlar, and Sindri P Andrason. Sage: task-environment platform for evaluating a broad range of ai learners. In *International Conference on Artificial General Intelligence*, pages 72–82. Springer, 2020.

Michael Genesereth, Nathaniel Love, and Barney Pell. General game playing: Overview of the aaai competition. *AI magazine*, 26(2):62–62, 2005.

Patrick Hammer and Tony Lofthouse. opennars for applications: Architecture and control. In *International Conference on Artificial General Intelligence*, pages 193–204. Springer, 2020.

Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *IJCAI*, pages 4246–4247. Citeseer, 2016.

Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, et al. Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*, 2018.

Hiroaki Kitano. *RoboCup-97: robot soccer world cup I*, volume 1395. Springer Science & Business Media, 1998.

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

Eric Nivel, Kristinn R Thórisson, Bas Steunebrink, Harris Dindo, Giovanni Pezzulo, Manuel Rodriguez, Carlos Corbato-Hernandez, Dimitri Ognibene, Jörgen Schmidhüber, Ricardo Sanz, Helgi P. Helgason, and Antonio Chella. Autocatalytic endogenous reflective architecture. *Tech report RUTR-SCS13002, Reykjavik University School of Computer Science*, 2013.

Arash Sheikhlar, Kristinn R Thórisson, and Leonard M Eberding. Autonomous cumulative transfer learning. In *International Conference on Artificial General Intelligence*, pages 306–316. Springer, 2020.

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

Kristinn R Thórisson, Jordi Bieger, Xiang Li, and Pei Wang. Cumulative learning. In *International Conference on Artificial General Intelligence*, pages 198–208. Springer, 2019.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

Pei Wang. *Non-axiomatic reasoning system: Exploring the essence of intelligence.* PhD thesis, Indiana University, 1995.

Pei Wang. *Rigid flexibility: The logic of intelligence.* Springer Science, vol. 34, 2006.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Shimon Whiteson, Brian Tanner, and Adam White. Report on the 2008 reinforcement learning competition. *AI Magazine*, 31(2):81–81, 2010.