JAISCR, 2024, Vol. 14, No. 1, pp. 45 – 61
10.2478/jaiscr-2024-0003

METRICS FOR ASSESSING GENERALIZATION OF DEEP REINFORCEMENT LEARNING IN PARAMETERIZED ENVIRONMENTS

Maciej Aleksandrowicz* and Joanna Jaworek-Korjakowska

*Department of Automatic Control and Robotics, AGH University of Krakow,
al. A. Mickiewicza 30, Building B-1, 30-059 Kraków*

**E-mail: macal@agh.edu.pl*

Submitted: 24th June 2023; Accepted: 19th October 2023

Abstract

In this work, a study focusing on proposing generalization metrics for Deep Reinforcement Learning (DRL) algorithms was performed. The experiments were conducted in DeepMind Control (DMC) benchmark suite with parameterized environments. The performance of three DRL algorithms in selected ten tasks from the DMC suite has been analysed with existing generalization gap formalism and the proposed ratio and decibel metrics. The results were presented with the proposed methods: average transfer metric and plot for environment normal distribution. These efforts allowed to highlight major changes in the model's performance and add more insights about making decisions regarding models' requirements.

Keywords: deep reinforcement learning, optimization, generalization, Sim2Sim transfer, adaptation

1 Introduction

The field of artificial intelligence (AI) research dedicated to arbitrary environments, known as reinforcement learning (RL), took off in the last decade thanks to advancements in hardware and software for training universal approximate models called deep neural networks. The promise to use such compute-expensive models is the ability to generalize from training task conditions to new, unseen cases, which is also known as transfer learning to Out Of Distribution (OOD) tasks [1]. Such ability was always a concern in control theory, where the regulator (control policy) needs a mathematical model of the dynamics of the control object described as:

$$\begin{aligned} \dot{x} &= f(x,t), \\ y &= g(x,t), \end{aligned} \tag{1}$$

where x is the system state vector, y is the output of the system, t is time, f describes underlying dynamics of the system and g defines which values can be observed. The parameters of f and g must also be identified in terms of changing conditions of a non-stationary non-linear system. A classical approach to such problems is to manually model and identify the system, then linearize the mathematical model around all considered working points.

In control theory, the assessment of the designed regulator can be performed from multiple criteria. Very well-established variables to construct metrics are: control error, robustness to disruptions, stability, overshoot, and constant offset. Such a variety of criteria could be proposed due to the numerical properties of the state space of the controlled object. Moreover, even if the variable under control can not be measured directly (e.x. due to noise) it is possible to create an adequate observer for that variable [2].

Estimation of "*how good*" an RL algorithm is (especially, a subgroup that combines deep neural networks with RL, called Deep RL (DRL)), can be performed in an environment that heavily depends on the sole definition of the reward function. The reward can be extrinsic, from the environment, or intrinsic, from the algorithm itself (for instance, as a bonus reward for exploration of unknown states). Generally speaking, the measure of *mean cumulative reward* is the main evaluation metric for a trained RL algorithm [3, 4]:

$$\mathcal{R}(\tau_\pi) = \frac{1}{N}\sum_{n=0}^{N}\sum_{t=0}^{T_n} r(s_t, a_t), \tag{2}$$

where $\mathcal{R}$ is mean cumulative reward function, which takes as an input τ set of N trajectories of states and actions, τ_π is the set of trajectories generated by trained policy π with corresponding T_n discrete time horizons, s is state, a is action, t is timestep, and r is an environment reward function, which takes state and action as input.

Other important parameters in the evaluation of RL algorithms are: training steps, the type of input data (i.e. visual observations are harder to process than already presented state vector), the distribution of permutations for the training environment, sample efficiency, the size of the model or the cost of computation.

Intuitively, it is possible to propose a qualitative metric through the comparison of a variety of environments and distinguish permutations that are objectively harder than others due to unfortunate circumstances (e.g. a greater gravity constant or more objects to avoid). However, defining a quantitative metric for decision-making assessment is not trivial, and can be seen as a part of defining the environment's reward function. Nevertheless, such an approach to developing a metric was explored in [5].

Another way to tackle the problem of formulating qualitative metrics for assessing generalization to unseen environments can be to focus on parameterized mean cumulative reward (2) by the environment parameters. Rephrasing that in a more concrete way, such a metric can be defined as a form of generalization gap in terms of comparing the performance of doing the same task in two different environments. This problem has been analysed in [1] and the aim of this work is to further investigate metrics for assessment of RL agent's generalization to unseen environments based on already defined generalization gap formalism.

The contributions of this work can be summarized as follows:

1. a proposal of metrics for assessing the generalization of DRL algorithms to the OOD tasks domains,
2. analysis of generalization of three DRL algorithms in 10 tasks of DeepMind Control Suite,
3. conductance of a proof-of-concept research showing the comparison of the metrics to the existing generalization gap formalism with proposed two methods,
4. findings, that the proposed metrics help to highlight the drop of the performance and add more insights about the generalization.

Motivation and importance

The ability to transfer a trained model from one dataset distribution domain to another is crucial for any statistical models, especially for machine learning (ML) ones. From recommendation systems, through computer vision to robotic tasks, each field depends on the reusability of the trained models in the real world problems. Such transfer ability, without significant change in performance and additional learning, is called *generalization*. If it is possible to further train the model after a transfer, such a technique is called *adaptation* [1] or fine-tuning. Adaptation is often used in supervised learning computer vision problems, where the additional training of the model to the new task domain should be done in as few training samples as possible (hence the name "few-shot transfer"). It is worth mentioning that in literature, "generalization" can be referred to as "zero-shot transfer". In both variants of the rules for model transferring, a method for measuring *generalization gap* (see definition in section 2) is essential for performing a quality assessment.

In the RL scenario, acting in the real world is often much more expensive and harder than running an agent inside a simulation. The amount of data needed to train such models to achieve enough robustness is excessively high. This is a well-known issue in DRL called *sample efficiency* [1] which motives the development of future algorithms towards maximization usage of already sampled experience. As an example, please consider the field of autonomous vehicles (AVs). Gathering experience from real world data (i.e. driving an autonomous vehicle around multiple sceneries, including city centres) requires, at least, the supervision of trained personnel or even permission from the local government. Efficient methods for training models for AVs in simulation will require some kind of assessment metrics to compare algorithm performance in the virtual and real worlds.

It is important to remark, that in terms of model transfer to other task instances, the tasks' domains can be Independent and Identically Distributed (IID) or OOD, with respect to the training domain [1] (see figure 1). An easy, but resource-consuming approach to deal with OOD generalization is to train from scratch (or adapt) an RL agent to the problematic domain, which relaxes the problem to IID generalization. A universal method to achieve OOD generalization in DRL is an ongoing research subject called *generalist agent*, which aims to obtain an *Artificial General Intelligence* (AGI) model. Assessing performance on multiple tasks (for instance, in a meta-learning scenario) is beyond the scope of this work. Nevertheless, in both distribution cases, it is important to develop techniques of quality comparison of agent's performance over multiple task instances.

In essence, to improve reliability, achieve better generalization in particular tasks, and cut costs of gathering data of DRL algorithms, it is important to think about possible metrics for measuring the performance of trained agents over multiple instances of task domains.

2 Preliminaries

Reinforcement learning is a category of machine learning algorithms dedicated to obtaining suboptimal solutions of Dynamic Programming (DP) problems [4, 3]. Such problems can be modelled as Markov Decision Problems (MDPs) $\mathcal{M} = (s_0 \in \mathcal{S}, p(\cdot|s,a), R(s), \mathcal{A}, \mathcal{S}, \gamma)$, where s_0 is the initial state, $\mathcal{A}$ is the set of possible actions, $\mathcal{S}$ is the set of possible states, γ is discount factor, $R(S)$ is reward function, $p(\cdot|s,a)$ is transition distribution function. In general, the underlying states of the MDP can only be partially observed, thus a Partially Observable Markov Decision Problem (POMDP) $\mathcal{M}_P = (s_0 \in \mathcal{S}, p(\cdot|s,a), R(s), \mathcal{A}, \mathcal{S}, \gamma, f : s \in \mathcal{S} \to o \in \mathcal{O})$ introduces a function f for observing state s as observation o [1]. Furthermore, to extend this formalism to include MDP parametrization, which shall be fixed only for episode duration, a Contextual MDP (CMDP) [1] is defined as follows: an MDP state s is decomposed into a tuple $s = (c, s') \in \mathcal{S}_C$, where $s' \in \mathcal{S}$ is the underlying MDP state and $c \in \mathcal{C}$ is the *context*. An arbitrary context c shall be thought of as an instance of a task from its domain distribution. Assuming that the context can not be observed by the agent, CMDP can be easily written as POMDP by the introduction of an observation function $f_c((c,s')) \to f(s') := o$ which discards the information of the context and maps the underlying state to an observation.

By having different c_{train} context for training and c_{target} context (see figure 1) for evaluation, the authors of [1] defined a *generalization gap* metric:

$$\text{GenGap}(\pi) := \mathbf{R}\left(\pi, \mathcal{M}_c|_{c_{\text{train}}}\right) - \mathbf{R}\left(\pi, \mathcal{M}_c|_{c_{\text{target}}}\right), \tag{3}$$

where π is trained DRL policy, $\mathcal{M}_c$ is any CMDP from context set C and $\mathbf{R}$ is the expected return of the policy. In this work, the generalization gap metric will also be referred to as a *difference* metric.

RL algorithms can be briefly divided into two groups: *model-based*, which uses some kind of information about the environment, and *model-free*, which requires more data to learn the dynamics of the environment. In this work, we will focus on model-free algorithms. These kinds of algorithms tackle the DP problem by many approaches, for instance (but not limited to): learning state-action value function (i.e. Q-function) to predict the highest cumulative reward in whole decision horizon (commonly known as *Q-Learning*), introducing latent space from partial state observations (*encoding)*, parameterized learned policies (of an *actor)* or learning dynamics models (*world models*). [6]. In this work three well-tested RL algorithms have been deployed: Proximal Policy Optimization (PPO), Soft Actor-Critic (SAC), and Deep Deterministic Policy Gradient (DDPG). They can be considered as benchmark baselines with the following specifications:

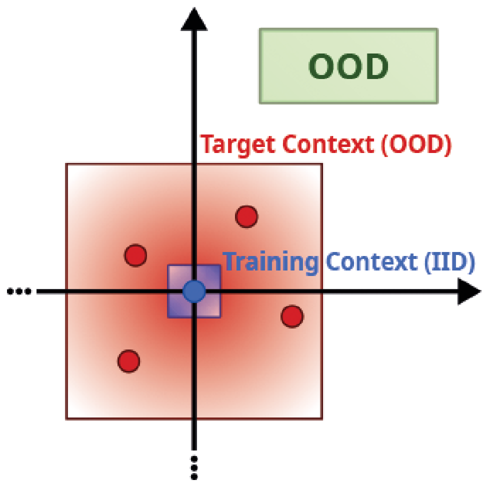

Figure 1. Relation of different task contexts (domains) in parameters space. The rectangular boxes define the contexts, where the points indicate a corresponding task instance. In this work, the agents are trained on only one instance inside the training context (blue point). They are tested against slightly modified task environments from the target context (red points). In a more general case, the target context will not have an intersection with training one (green area).

Proximal Policy Optimization (PPO) [7] algorithm was designed around two ideas: 1) updating the policy similarly to the Trust Region Policy Optimization (TRPO) algorithm, where the policy weights are changed in a way, that they do not deviate too far from the current policy [8], 2) clipping a surrogate objective function that constrains the policy update to a small region around the current policy [7]. For further tinkering with policy gradient, PPO uses an advantage estimate to scale the policy gradient updates. The advantage function $A(s,a) = Q(s,a) - V(s)$ allows calculating the difference between possible actions a_i in state s in terms of greater future reward. The value function $V(s)$ is trained separately from the policy network and is used to compute an advantage estimate $A(s,a)$.

Soft Actor-Critic (SAC) [9] is an off-policy (i.e. agent learns from experience generated by different policy), actor-critic algorithm which means that the agent consists of two parts: a policy *actor* which learns ”how to act” and a *critic* which learns some function, typically Q-value, to ”critique actual state” and is based on maximum entropy framework. This framework allows for the optimization of a trade-off between the expected return and the entropy of the policy. The definition of policy entropy can be interpreted as a measure of randomness (or exploration) in the agent’s decision-making process. This approach of modelling exploration with entropy allows an elegant tackling of exploration vs exploitation problem [9].

Deep Deterministic Policy Gradient (DDPG) [10] is a model-free algorithm that learns both the Q-function and the policy. The Q-function estimates the expected sum of rewards that can be obtained from taking a specific action in a given state, while the policy specifies the agent’s behaviour, mapping the states to actions. The Q-function and policy are learned in two separate steps, with the Q-function being learned first, followed by the policy. The authors introduced the requirement of a differentiable Q-function, which allowed it to be used during policy updates in gradient descent. This decision resulted in another property, that this algorithm supports only continuous action space.

3 Related works

A comprehensive survey of generalization in DRL presented in [1] is a good introduction to the problem. The authors made an exhaustive description of multiple approaches to tackle the DRL generalization, including (but not limited to): a brief foundation to Contextual MDP (CMDP) using original formulation [11] with [12] formalism, transfer learning evaluation protocols in terms of task distribution (IID and OOD), classification of popular environments and formulation of *generalization gap* formalism (equation 3) in DRL based on analogous problem in deep supervised learning.

During the last years in the area of deep RL research (DRL), there were multiple attempts to introduce normalized benchmarks for developing algorithms, such as collection of environments from OpenAI Gym [13], DeepMind Control Suite (DMC) [14] or procedurally generated Procgen Benchmark [15]. Due to the simplicity of the OpenAI Gym interface, it became *de facto* a standard for defining new environments. An example of that is a Powderworld [16] environment, designed to test the generalization of agents via rich task distributions. However, the field is dynamic. The current maintainers of the Gym decided to create a fork of it to create a new version called Gymnasium [17]. This variety of benchmark suites helps in the quality assessment of the agents from the perspective of diverse task scenarios.

As an example from other branches of ML, a simple generalization metric, based on robustness to augmentations, was proposed in [18] for classification problems in supervised learning. For every augmented sample for the model's input, a penalty is calculated based on the difference between the probabilities of the predicted classes. The difference can be interpreted as the disparity in class confidence. The value of the penalty is determined by the strength of the augmentation. Such an approach could be used in modelling a generalization metric in RL if the underlying parameters of an environment's dynamics could be measured.

A common technique to obtain success transfer of a DRL model is to introduce a variety of perturbations during the training [19, 20]. Such procedure is often pursued in terms of increasing agent robustness, and it is evaluated in terms of comparing performance after transfer similarly to the generalization gap metric (3).

Another approach to the RL generalization is combining DRL with meta-learning techniques. This field is developing fast and was summarized in Meta RL survey [21]. It is worth mentioning, that this approach can be seen as another step towards artificial general intelligence, according to the Alberta Plan for AI Research [6].

The team behind [22] researched two ideas of modifying DRL algorithms to tackle environment context: 1) EPOpt, which goal was to maximize the expected reward over a fraction of environments, with the worst expected reward. 2) RL^2, which tried to adapt RL agents to identify the dynamics of the environment at hand by modelling policy and value functions as recurrent neural networks. The hidden states of these networks were used as an environment embedding. The work shows that ”*vanilla*” DRL algorithms generalize better than the proposed modifications.

From the perspective of learning representation of the system state from observations, the authors of [23] introduced a *Reducing Approximation Gap* (RAP) metric to measure ”distance” between two state representations. The derivation of RAP was followed by careful analysis of problems with previous approaches to representation metrics and resulted in a state-of-the-art modification to the SAC algorithm. In opposition to their advancements of representation learning, this work focuses on extending the generalization gap (3) based on extrinsic rewards instead of intrinsic (i.e. inside the agent) approximations, to develop methods for assessing the algorithms' generalization.

Similarly to this work, the objective of [24] study was measurement and defining the generalization in DRL. The authors were using mean cumulative reward (2) to compare generalization performance. They used the last layers of deep Q-networks to calculate the Euclidean distance between them. This approach assumes that the policy is deterministic, which can not be held in general, due to the stochastic nature of many RL algorithms.

The generalization is of course not measurable in terms of qualitative metrics, due to the ambiguous definition of ”performing well enough” across whole task contexts. Before assessing it, some kind of quantization must be performed, similarly to [25], which mainly studied the problem of overfitting in DRL. The authors of that work proposed a new procedurally generated testbed environment called CoinRun, which was used to construct a simple framework to measure overfitting to the training domain.

Two metrics for evaluation of environment difficulty, Policy Information Capacity (PIC) and Policy-Optimal Information Capacity (POIC) were proposed in a broad study [5]. These metrics can be used in the procedure of assessing the goodness of reward-shaping proposals. During experiments, it was found that high values of both metrics effectively correspond to regions of the fastest learning. Despite all the advantages and similarly to this work, the obvious limitation of these metrics falls mainly in the effective search area.

The matter of assessing the generalization can also be found in adaptation tasks. For instance, the problem of domain adaptive human pose estimation with the absence of access to the source training data can be initially interpreted as a transfer from a training context to an external target context (see red and green contexts in Figure 1) [26, 27].

In addition, it is worth pointing out two more things: 1) there is a high connection between generalization and hyperparameters, which was investigated in [28] and 2) generalization could be evaluated in terms of robustness to perturbations in new environment, which was explored in [29].

4 Methodology

In this work, a generalization assessment problem has been considered, which can be formulated as follows: an agent is trained to solve an independent task in a parameterized environment. Then, one of the environment parameters (which is related to the underlying dynamics of the system) must be changed by some degree, relative to the training value. After that, the trained agent is run against a modified environment to evaluate its performance, mainly in terms of maximizing the reward. With collected data, it is possible to compute generalization gap metrics (e.g. from equation (3)) and compare them.

The mentioned procedure has been repeated against different DRL algorithms, which were trained multiple times with different initialization seeds. To maintain full control over the environment, this procedure has been done in simulation. These steps have been presented in Figure 2.

It is worth mentioning that in literature, transfers to other task domains can be characterized as *repetition* (transfer to the same task in the same environment), *interpolation* (transfer to IID context), and *extrapolation* (transfer to OOD context) [24, 22].

4.1 Proposed metrics

As mentioned in section 1 there are multiple ways to evaluate the performance of the transferred agent to the new environment. This work focuses only on metrics based on the collected reward by extension of the generalization gap metric (3).

The proposed metrics are based on the following ideas. The first one is based on observation of a group of DRL researchers, who were investigating results of transfer learning. During that procedure, their idle talk was about the subjective judgment of the agents in terms of their performance. Considering that they were just comparing one model to the others and commenting on their judgments in an exaggerated way, a generalization metric based on performance relation and decibel scale is proposed as follows:

$$G_{\text{ratio}}(\pi) := \frac{\mathbf{R}\left(\pi, \mathcal{M}_c|_{c_{\text{target}}}\right)}{\mathbf{R}\left(\pi, \mathcal{M}_c|_{C_{\text{train}}}\right)} \tag{4}$$

$$G_{\text{dB}}(\pi) := 10\log_{10} G_{\text{ratio}}(\pi), \tag{5}$$

where the rewards $\mathbf{R}$ are defined analogously as in equation (3). To maintain a valid domain of the logarithm and fraction, the rewards $\mathbf{R}$ must be normalized to a positive, non-zero range.

Furthermore, taking into consideration multiple transfer targets ($n \geq 2$) of task contexts, an average transfer metric can be defined:

$$G_{\mu}(\pi) := \frac{1}{||C_{\text{target}}||} \sum_{c \in C_{\text{target}}} G_{\text{any}}(\pi), \tag{6}$$

where $||C_{\text{target}}||$ is the size of the target context (i.e. number of all environment instances in the target domain) and G_{any} is any generalization metric, like difference (3), ratio (4) or decibel (5). The latter comes from an assumption, that in general, investigating the whole target context is too expensive, but an educated guess about performance could be told with an appropriate number of tests.

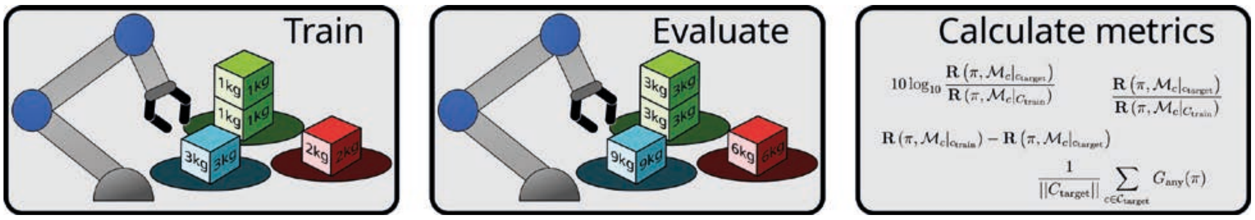
Train
1kg
1kg
1kg
1kg
3kg
3kg
2kg
2kg
Evaluate
3kg
3kg
3kg
3kg
9kg
9kg
6kg
6kg
Calculate metrics
$10\log_{10}\frac{\mathbf{R}(\pi,\mathcal{M}_c|_{c_{\text{target}}})}{\mathbf{R}(\pi,\mathcal{M}_c|_{C_{\text{train}}})}$
$\frac{\mathbf{R}(\pi,\mathcal{M}_c|_{c_{\text{target}}})}{\mathbf{R}(\pi,\mathcal{M}_c|_{C_{\text{train}}})}$
$\mathbf{R}(\pi,\mathcal{M}_c|_{c_{\text{train}}})-\mathbf{R}(\pi,\mathcal{M}_c|_{c_{\text{target}}})$
$\frac{1}{||C_{\text{target}}||}\sum_{c\in C_{\text{target}}} G_{\text{any}}(\pi)$

Figure 2. The methodology used in this work. First, the agents are trained in a fixed environment. Then an evaluation procedure is performed on the transferred agent to a slightly changed environment. Please note, that from the perspective of the agent the space and observation domain are the same — only the underlying dynamics of the environment are different. Finally, the obtained results (in terms of total reward) are used to calculate generalization metrics.

5 Experiments

5.1 Experimental setup

The algorithms were trained on the DMC suite using the implementation from Ray RLlib framework [30]. Ray is a software framework for scaling up machine learning research and development through tools for easily running multiple experiments in parallel, and then gathering all data and trained models in one place. RLlib is one of the submodules of Ray, which offers support for highly distributed RL setups, including support for multi-agent environments, training from offline datasets (recorded trajectory of actions and states), and integration with external environments. The authors of RLlib included implementations of many well-established RL algorithms both in PyTorch and Tensorflow deep learning frameworks, including PPO, DDPG, and SAC.

To set up a simple scenario for testing generalization over some perturbations to the environment parameters, a simple modification to the DMC *benchmark* suite was made. For each of its 14 environments (see figure 3) one parameter related to the system dynamics was selected. The selection of these variables was made on the difficulty basis of interference in the environment definition. In other words, the simplest variables to modify were selected. These parameters are presented in Table 1. Each parameter has 20 discrete values, defined within boundaries of 10% and 200% of the base value, with a step of 10%. The logic behind the selection of these unified ranges' boundaries was insurance to have significantly different dynamics of the underlying system, which still have physical meaning.

It should be noted that the DMC benchmark suite contains 28 tasks, defined unevenly in 14 environments. For reference, please consult the DMC suite paper [14] or check appendix A.

For obtaining qualitative generalization assessment, metrics (2), (3), (4), (5) and (6) were implemented. DMC suite avoids negative rewards, but there is still a chance that an algorithm could obtain zero rewards in the whole episode. In that case, the trained model should be discarded and trained again. In this work, such a problem did not happen.

Table 1. Selection of underlying parameters in DMC environments to parameterize. The boundaries of the parameter ranges are defined as 10% and 200% of the base value. Each parameter has 20 discrete values (with the step of 10% of the base value).

Environment	Parameter	Base	Range	Units
acrobot	joints mass	1.00	$[0.1, 2.0]$	kg
ball in cup	string length	0.30	$[0.03, 0.6]$	m
cartpole	pole mass	0.10	$[0.01, 0.2]$	kg
cheetah	torso half length	0.50	$[0.05, 1.0]$	m
finger	damping coefficient	2.30	$[0.23, 4.6]$	-
fish	fins servo gain	$3.00 \cdot 10^{-4}$	$[0.3, 6, 0] \cdot 10^{-4}$	-
hopper	upper torso height	0.20	$[0.02, 0.4]$	m
humanoid	hips gear ratio	120.00	$[12, 240]$	-
manipulator	motor gear ratio	12.00	$[1.2, 24]$	-
pendulum	pendulum mass	1.00	$[0.1, 2]$	kg
point mass	point mass	0.30	$[0.03, 0.6]$	kg
reacher	joints gears ratio	0.05	$[0.005, 0.1]$	-
swimmer	motors gears ratio	$5.00 \cdot 10^{-4}$	$[0.5, 10.0] \cdot 10^{-4}$	-
walker	hips gear ratio	100.00	$[10, 200]$	-

The whole project is available as an open source code on GitHub: https://github.com/macmacal/drl_generalization_metrics.

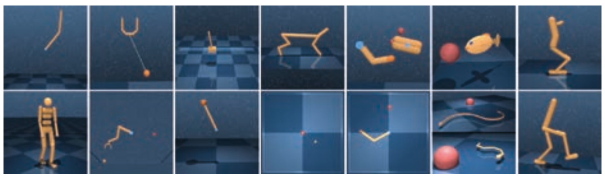

Figure 3. DeepMind Control Suite environments from a subset called *benchmark*. They are sorted alphabetically (same as in Table 1), in order from the upper left to the bottom right. Each environment ha_s defined at least one task, with a total of 28 tasks in a selected subset. Image taken from [14].

5.2 Agents training

The agents were trained using almost default configs from the Ray RLlib framework, differing only in the number of layers for underlying neural networks as 2, across all RL algorithms. The observations of the environment were the state vector of the system. For each of the 28 tasks in DMC benchmark suite, a separate model has been learned to use 5 different initialization random seeds. The training procedure resulted in 15 models for each of the 28 tasks. Please note, that not all models are expected to perform well. Due to the lack of hyperparameters and architecture optimization, some trials of the selected algorithms struggled during training, even in simple environments. See Appendix A for a summary of all learning trials and configurations.

Performing an analysis of the training performance of the models, it was possible to conclude which algorithms were able to find (near) suboptimal solutions. For further discussion 18 tasks were discarded due to **inability to find a solution** (acrobot, cartpole swigup sparse, hopper, manipulator, pendulum, humanoid run) or **very high uncertainty in episode reward** (finger turn easy, finger turn hard, fish, humanoid run & walk, reacher, swimmer). The training process of the remaining 10 tasks is presented in Figure 4.

5.3 Evaluation of agent generalization

According to the setup described in section 5.1, each of the trained models was evaluated in terms of mean cumulative reward (2) in 20 slightly different environment instances of the given task. Each model seed was tried 30 times, which resulted in a total of 150 data points per environment instance. The results for considered 10 tasks are presented in Figure 5. The tenth instances of the environments are identical to the training context, where the rest are considered as the target domain (in accordance to the figure 1).

Investigation of obtained results shows a tendency of decreasing performance across different instances of the environment. Of course, it highly depends on the selection of the environment parameters, which in specific cases could relax the problem to a simpler one. The variance of the model's performance can be interpreted as stability or repeatability, which ideally should be minimized. In the case of the PPO algorithm for all walker tasks, the performance is increasing in terms of higher *hips gear ratio* values. Such an effect is probably connected with poor training results for this DRL algorithm (figure 4, walker tasks). For the considered tasks of cartpole, point mass environment and walker-stand task, the SAC algorithm generalized well in both maintaining relative (in comparison to the training value) high rewards and a relatively small variance.

The results show that the SAC algorithm was the most stable. In comparison with PPO and SAC, DDPG struggled and finally failed in terms of stability, both in training and evaluation.

5.4 Metrics comparison

The measured mean cumulative rewards (2) obtained during the evaluation analysis in section above were further used to calculate relative metrics of: difference (3), ratio (4) and decibel (5). The aggregated results are presented in Figure 8 (in Appendix A). For the sake of clear presentation, the five tasks out of ten (finger spin, point mass easy, walker run, walker stand & walker walk tasks) are presented in Figure 6. These tasks have relatively more informative box plots than the rest five plots.

From the perspective of formulation of an optimization problem, the generalization gap difference metric is sufficient to show the trend of algorithm performance. Both ratio and decibel metrics can be considered as normalized charts of the generaliza-

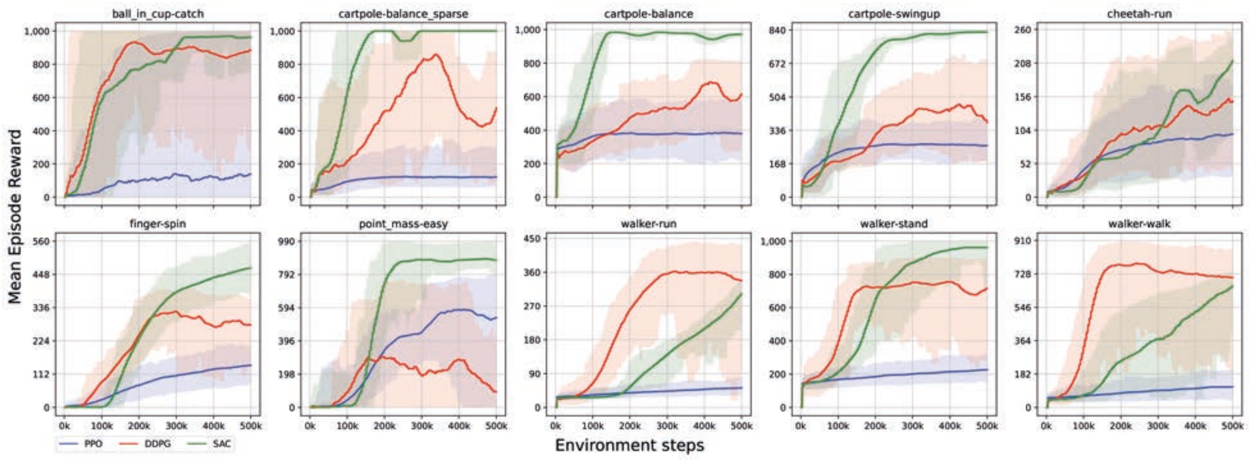

Figure 4. Selected results of training the DRL algorithms on DMC benchmark suite with 500 steps horizon. The bold line with corresponding area indices is the mean reward and its minimal and maximal value during model evaluation. Each algorithm was trained for 500k environment steps with 5 different network initialization seeds. The presented results show the mean of all model seeds. The SAC results are interpolated for alignment with PPO and DDPG.

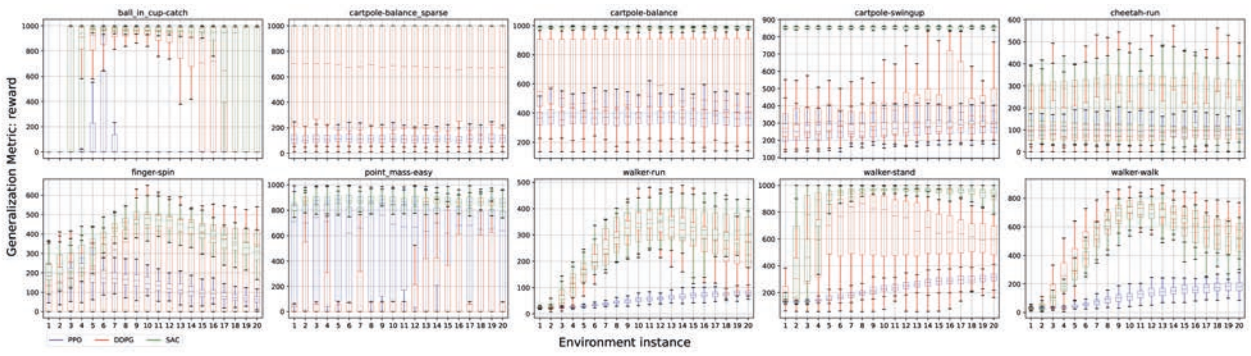

Figure 5. Performance of the DRL algorithms across selected tasks with slightly different underlying parameters of the environment. The performance is described with the value of the cumulative reward (2). The tendency to increase performance after the transfer of the PPO algorithm is probably connected with insufficient training time.

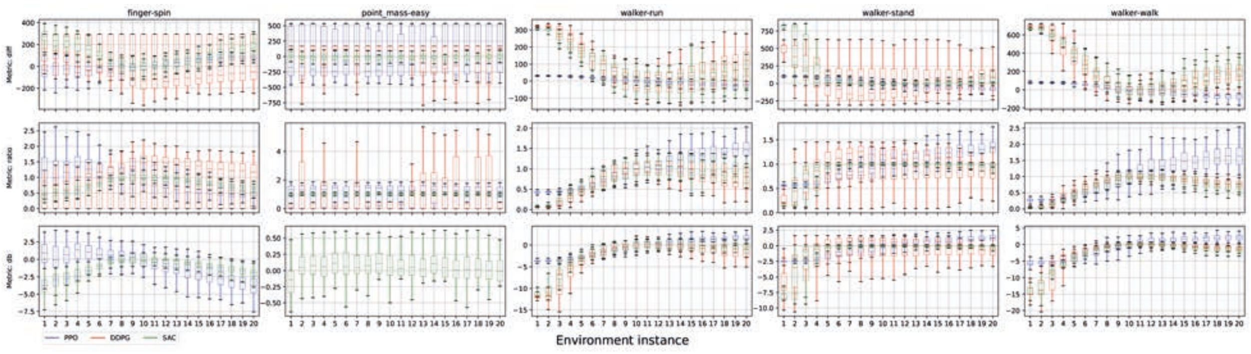

Figure 6. Three generalization gap metrics (difference (3), ratio (4) and decibel (5)), for sub-selection of the tasks from figure 5. The relative value (final training performance) is the mean of the tenth environment instance. The chart for decibel metric for point mass easy task lacks data for PPO and DDPG due to the problems with near-zero numerical errors.

tion gap. On the other hand, determining the criteria of ”good enough” performance of the transformed model can be easier done in these metrics.

The rewards’ ratio can be used to formulate straightforward requirements for transfers. For instance, for the finger spin task a value of $\geq 95\%$ performance after transfer disqualifies all DDPG models, where PPO and SAC models in a significant number of environment instances. However, using any kind of relative metric does not inform about the original performance of the algorithm. The results must be investigated, as it is easier to generalize failure than success over different environment instances.

Transforming the ratio values into decibels scale allows highlighting of a great decrease in the performance, like in cases of walker run and walker walk tasks.

All three metrics can shed more light on the problem, both in interpolation and extrapolation transfers. It must be noted, that calculating the logarithm-based metrics must take the design of the reward function into consideration – such an approach is not feasible for non-positive rewards. Problems with near-zero calculations occurred with decibel metric for the following tasks: finger spin, point mass easy & ball in cup catch.

5.5 Mean generalization metrics

From a practical perspective, evaluation of the whole target context is computationally expensive and time-consuming. To address that, a simple method for assessing the generalization performance across the whole target domain is proposed: an **average generalization metric** (6). It can be used with any other mentioned metric G_{any}. The main idea is to sample environment instances in the target domain, evaluate transfers, and calculate the average generalization metric. The results for applying this method for difference (3), ratio (4) and decibel (5) metrics are presented in Table 2.

The average difference metric does not inform much about the potential problem with order of magnitude, which could be highlighted with ratio or decibel metrics. Calculating the average and accompanying standard deviation can be used to provide information about the repeatability and robustness of the trained models. Especially high (over 43%) uncertainly is observed for: DDPG algorithm in ball in cup catch, cartpole balance sparse, cartpole balance & point mass easy tasks; PPO algorithm in the ball in cup catch and point mass easy tasks. On the other hand, the SAC algorithm performs more robustly on average across ten selected tasks.

Obviously, it must be stated that every average is prone to outliers and the input values should be inspected before proceeding with such an approach. The method of sampling environment instances from the target context remains open for further work.

5.6 Environment normal distribution

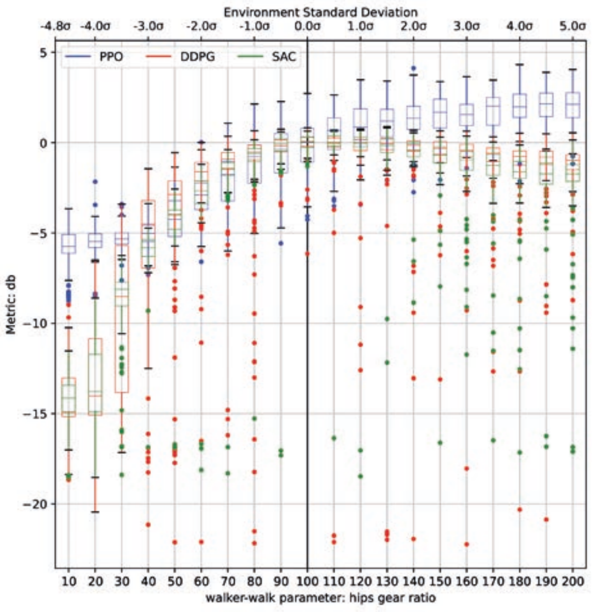

Figure 7. Decibels generalization metric (5) for all DRL algorithms across instances of the walker environment for the walk task. The dots represent outliers. With the assumption, that the boundaries of the underlying parameter (here: hips gear ratio) can be interpreted as a range from -5 to 5 standard deviation and the mean is the training value, it is possible to add auxiliary information for assessing the transfer performance. For the sake of clear presentation, the standard deviation range was calculated for $[0, 200]$ boundaries instead of $[10, 200]$.

Table 2. Averages of normal distribution parameters among all environment instances, used as the average generalization metric (6). These results were calculated for considered relative generalization gap metrics. The values of σ standard deviation, which can be interpreted as policy robustness, are even more crucial than the actual μ mean performance value.

Task	DDPG		PPO		SAC	
	μ	σ	μ	σ	μ	σ
Difference metric						
ball_in_cup-catch	43.15	311.26	553.99	328.74	−53.30	227.71
cartpole-balance_sparse	120.62	377.22	575.19	49.27	−305.54	0.00
cartpole-balance	117.55	297.67	321.84	73.57	−287.97	14.81
cartpole-swingup	320.95	156.72	429.21	56.47	−131.27	61.45
cheetah-run	551.94	137.84	595.43	32.67	468.47	130.36
finger-spin	461.26	195.36	557.43	41.81	319.45	48.48
point_mass-easy	509.67	312.61	164.92	357.91	−165.12	67.63
walker-run	461.15	50.34	642.38	8.37	434.40	54.58
walker-stand	104.05	236.15	466.62	35.72	−162.67	88.48
walker-walk	178.15	112.91	584.13	27.38	239.05	82.34
Ratio metric						
ball_in_cup-catch	0.94	0.45	0.20	0.47	1.08	0.33
cartpole-balance_sparse	0.83	0.54	0.17	0.07	1.44	0.00
cartpole-balance	0.83	0.43	0.54	0.11	1.41	0.02
cartpole-swingup	0.54	0.23	0.38	0.08	1.19	0.09
cheetah-run	0.21	0.20	0.14	0.05	0.33	0.19
finger-spin	0.34	0.28	0.20	0.06	0.54	0.07
point_mass-easy	0.27	0.45	0.76	0.52	1.24	0.10
walker-run	0.34	0.07	0.08	0.01	0.37	0.08
walker-stand	0.85	0.34	0.33	0.05	1.23	0.13
walker-walk	0.74	0.16	0.16	0.04	0.66	0.12
Decibel metric						
ball_in_cup-catch	0.73	1.11	0.07	0.88	1.07	0.58
cartpole-balance_sparse	−3.81	7.16	−7.98	1.66	1.58	0.00
cartpole-balance	−1.65	3.01	−2.79	0.87	1.51	0.07
cartpole-swingup	−3.10	1.75	−4.28	0.93	0.74	0.34
cheetah-run	−11.69	9.57	−8.72	1.58	−6.27	4.20
finger-spin	−3.92	5.97	−7.55	1.41	−2.85	0.64
point_mass-easy	−4.63	10.61	−0.69	4.76	0.91	0.36
walker-run	−5.92	1.22	−11.69	0.66	−5.20	1.03
walker-stand	−1.42	2.52	−5.06	0.67	0.48	0.74
walker-walk	−2.80	2.38	−8.91	1.13	−3.20	1.70

In special cases of the environment, where one of its dynamic parameters can be described in terms of normal distribution, it is possible to add an auxiliary axis to the chart of generalization metric, like in Figure 7.

This representation view allows highlighting the portion of the transfer target domain in which the model will perform within the acceptance range from the original. Moreover, it can also be used to show the comparison between training and target domains, which one can further use to find optimal boundaries for training.

It should be emphasized that adding this second axis for standard deviation introduces an implicit transition from discrete instances of the environment to a continuous scale.

6 Limitations and future work

The presented ideas of extending the generalization gap metric have their limitations. The most straightforward issue is the definition of the environment's reward function. In general, such a definition is hard and affects the training and evaluation, thus the whole generalization assessment relies solely on it. It also must be noted, that relative metrics with any logarithm term can suffer from numerical errors for very small values of the mean cumulative reward.

The increase in relative generalization metrics can be interpreted in at least two ways: the parameter change relaxed the task, or the trained policy was not performing well in the first place.

The considered metrics were used in well-defined, simulated environments, which guaranteed the reproduction of the same instances. In general, this assumption of stationarity of underlying parameters will not hold in real world scenarios. Therefore, the metrics should be used carefully in real world evaluation, which is the potential direction of future work. Especially considering long-horizon scenarios. Assessment of such long horizon model reliability was investigated in [31].

Moreover, the number of environment instances, which differed only in one parameter, allowed to treat them like a discrete value in the charts. In real world scenario, such a quantitative approach probably will be reduced to qualitative comparison, especially, when the exact difference of underlying parameters (e.g. mechanical tolerances) will be not known.

The list of evaluated DRL algorithms should also include more recent ones, like DrQ-v2 [32] or DreamerV3 [33]. This is especially relevant considering the experiment results: the DDPG algorithm performance was very noisy both in training and evaluation, while PPO could not find "better" solutions.

Each evaluation instance had its outliers (see figure 7), which could be further used to formulate a metric for stability and robustness assessment. However, both this proposition and the considered metrics are still based on performance in means of the mean cumulative reward. Future work should focus on developing metrics that include other important features, like: time of a training procedure, sample efficiency, biases towards early stages of training & execution time.

Estimation of average generalization metric values in Table 2 assumes that the number of sampled environment instances is computationally less demanding than assessing the whole target domain. As mentioned in section 5.5 it is also prone to exceptional poor or excellent performance in particular instances. Future work in this approach could investigate optimization methods to determine the number of environment instances to estimate the number of samples for the average.

The approach of environment normal distribution (section 5.6) assumes that there is a possibility to measure (and change) one of the dynamics parameters, which can be described as a normal distribution (e.x. the lighting conditions). The practicality of this chart, especially in real world evaluation, must be verified in future work.

As a final remark, the considered metrics can not be used straightforwardly in describing the transfer from a whole domain of tasks in one environment to another domain of tasks in the same environment. Such a problem is due to the usage of the task's reward definition, which in general will be not the same across different tasks.

7 Conclusions

In this work, an investigation of generalization metrics based on generalization gap formalism has been made on a simple transfer learning scenario in DeepMind Control Suite environments. The proposed relative metrics of ratio (4) and decibel (5) allowed to highlight major changes in the models' performance. Presenting obtained results in the form of average generalization metric (table 2) and as the chart with additional axis scaled in standard deviation (of the underlying parameter) (figure 7) allowed to add more insights into the results. In contrast with already established methods, these improvements allow for a more straightforward comparison with given requirements.

However, the subject of the metrics for generalization assessment is a very broad topic and future investigations shall be performed.

Furthermore, the metrics are based only on external reward scores from the environment. Without further information, it is impossible to tell anything in terms of *good enough* performance across the dynamics context of the trained DRL algorithms. Further requirements about the task and definition of the environment are needed to obtain even more qualitative, but more specific assessment methods.

Acknowledgments

We gratefully acknowledge the funding support by program “Excellence Initiative – Research University” for the AGH University of Krakow as well as the ARTIQ project ARTIQ/0004/2021.

A Models training and generalization metrics

The algorithms training was accomplished using Ray 2.1.0 framework inside a Docker container with Ubuntu 20.04, CUDA 11.6.2, cuDNN 8, and PyTorch 1.13.1. A corresponding Dockerfile with all dependencies is available within the project's repository. Hardware used for training consists of AMD Ryzen 9 7950X CPU, NVIDIA GeForce RTX 3090 (24 GB VRAM) with 64 GB of RAM.

The summary of the training configuration is presented in Table 3. In the early stages of the research, there were attempts to modify the maximum episode horizon with Ray configuration. Regardless of the modification of this parameter, Ray's RLlib wrapper for DeepMind Control Suite fixed the horizon to 500 steps. Each algorithm was trained for 500k environment steps with 5 different network initialization seeds.

The training of the selected three DRL algorithms on all 28 DMC benchmark suite tasks is presented in Figure 9. Refer to section 5.2 for information on which of these trails are discarded.

In Figure 8 the aggregated results of generalization metrics for 10 tasks are present. For further information, refer to the section 5.4.

Table 3. Training parameters shared along all DRL algorithms.

General	
evaluation_config/render_env	false
evaluation_interval	10
evaluation_num_workers	2
framework	torch
log_level	ERROR
num_gpus	1
num_workers	8
Model	
fcnet_activation	relu
fcnet_hiddens	[64, 64]

References

[1] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A Survey of Zero-shot Generalisation in Deep Reinforcement Learning. Journal of Artificial Intelligence Research, 76: 201–264, January 2023. ISSN 1076-9757. doi:10.1613/jair.1.14174.

[2] Katsuhiko Ogata. Modern Control Engineering. Prentice Hall, 2010. ISBN 978-0-13-615673-4.

[3] Richard S. Sutton and Andrew G. Barto. Sutton & Barto Book: Reinforcement Learning: An Introduction. 2018. ISBN 978-0-262-03924-6.

[4] Dimitri P. Bertsekas. Reinforcement Learning and Optimal Control. 2019. ISBN 978-1-886529-39-7.

[5] Hiroki Furuta and et al. Policy Information Capacity: Information-Theoretic Measure for Task Complexity in Deep Reinforcement Learning. In Proceedings of the 38th International Conference on

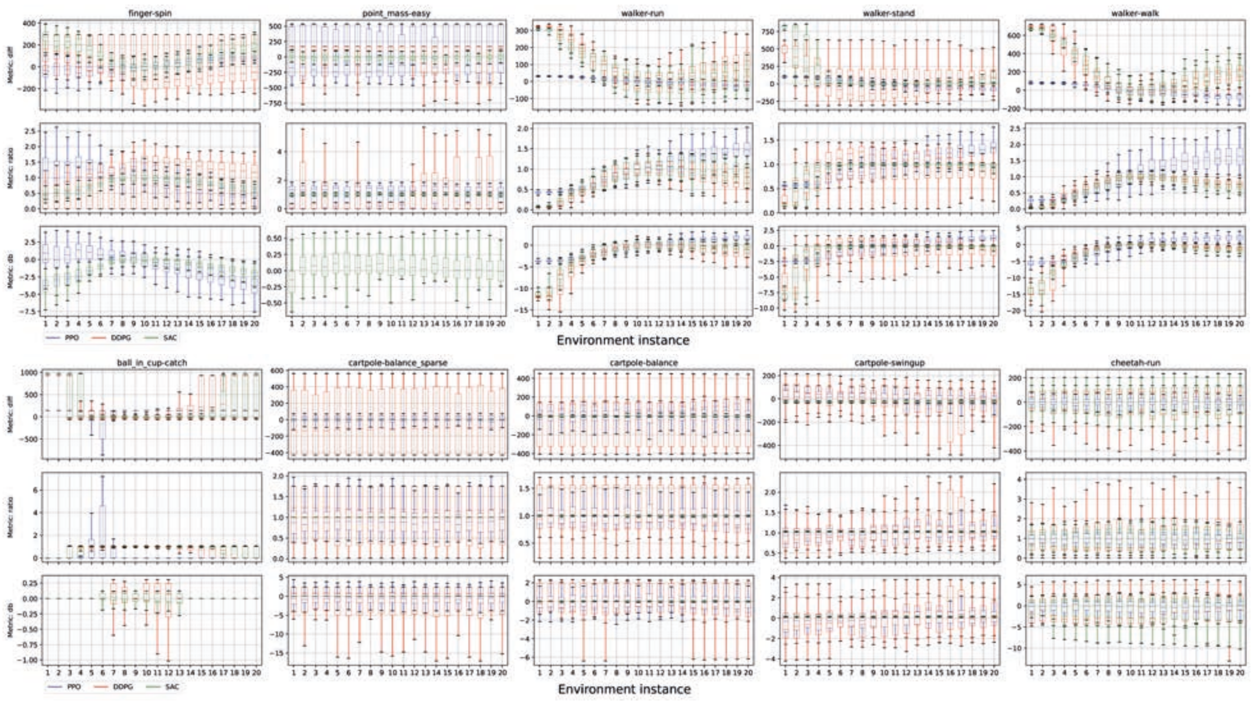

Figure 8. The aggregated results of generalization metrics for selected tasks (see section 5.4).

Machine Learning, pages 3541–3552. PMLR, July 2021.

[6] Richard S. Sutton, Michael H. Bowling, and Patrick M. Pilarski. The Alberta Plan for AI Research, August 2022.

[7] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. August 2017.

[8] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust Region Policy Optimization. April 2017.

[9] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. arXiv:1801.01290 [cs, stat], August 2018.

[10] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. arXiv:1509.02971 [cs, stat], July 2019.

[11] Assaf Hallak, Dotan Di Castro, and Shie Mannor. Contextual Markov Decision Processes, February 2015.

[12] Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P. Adams, and Sergey Levine. Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability, July 2021.

[13] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. arXiv:1606.01540 [cs], June 2016.

[14] Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. Dm_control: Software and tasks for continuous control. Software Impacts, 6: 100022, November 2020. ISSN 26659638. doi:10.1016/j.simpa.2020.100022.

[15] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging Procedural Generation to Benchmark Reinforcement Learning. In Proceedings of the 37th International Conference on Machine Learning, pages 2048–2056. PMLR, November 2020.

[16] Kevin Frans and Phillip Isola. Powderworld: A Platform for Understanding Generalization via Rich Task Distributions, November 2022.

[17] Farama Foundation. Gymnasium, 2023. URL https://gymnasium.farama.org/.

[18] Sumukh Aithal K, Dhruva Kashyap, and Natarajan Subramanyam. Robustness to Augmentations

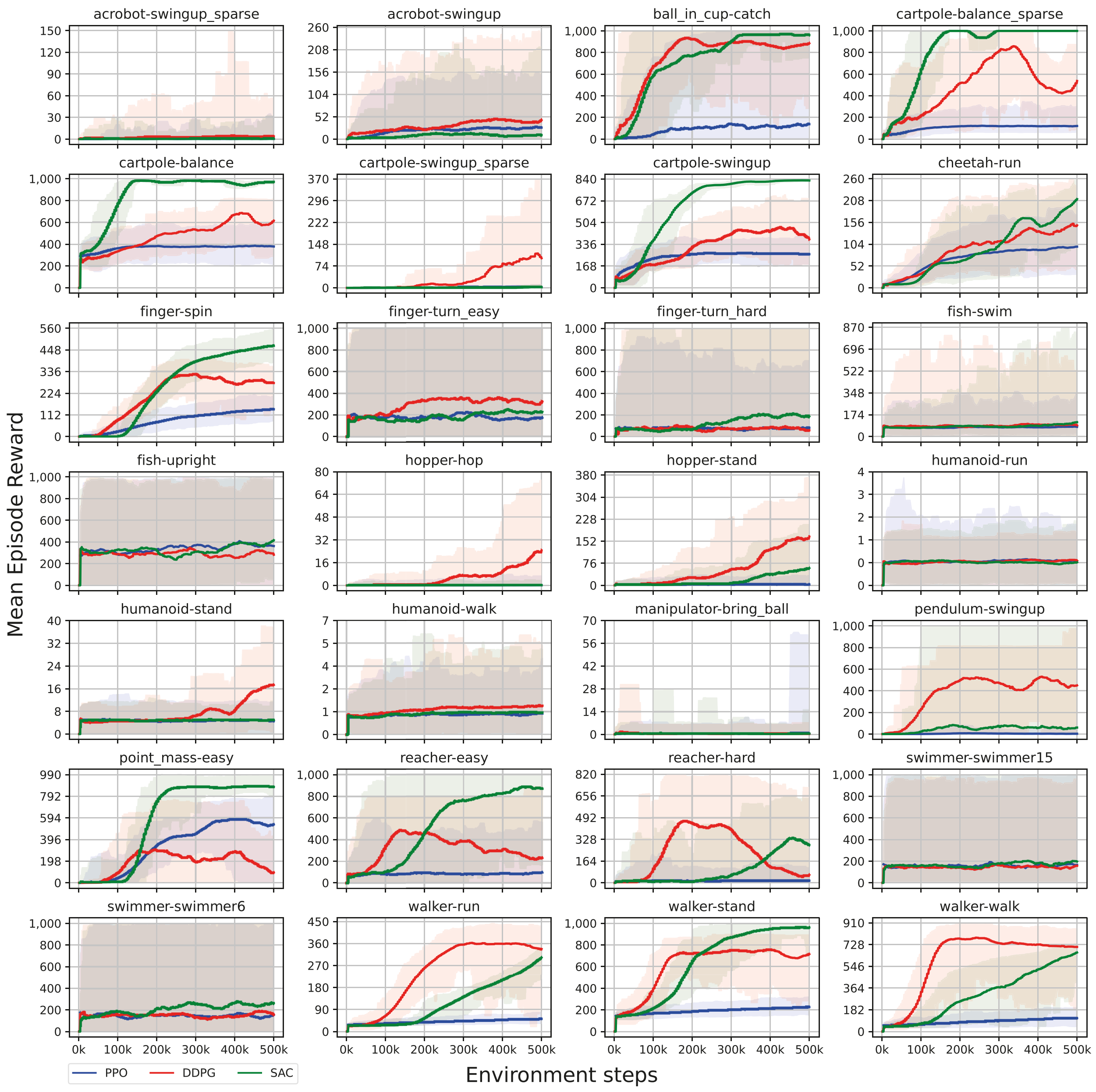

Figure 9. Aggregated results of training selected DRL algorithms on DMC benchmark suite with 500 steps horizon. The bold line with corresponding area indices is the mean reward and its minimal and maximal value during model evaluation. Each algorithm was trained for 500k environment steps with 5 different network initialization seeds. The presented results show the mean of all model seeds. The SAC results are interpolated for alignment with PPO and DDPG.

as a Generalization metric. arXiv:2101.06459 [cs], January 2021.

[19] OpenAI, Ilge Akkaya, and et al. Solving Rubik's Cube with a Robot Hand, October 2019.

[20] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications. arXiv:1812.05905 [cs, stat], January 2019.

[21] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A Survey of Meta-Reinforcement Learning, January 2023.

[22] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing Generalization in Deep Reinforcement Learning. March 2019.

[23] Jianda Chen and Sinno Pan. Learning representations via a robust behavioral metric for deep reinforcement learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, Advances in Neural Information Processing Systems, volume 35, pages 36654–36666. Curran Associates, Inc., 2022.

[24] Sam Witty, Jun K. Lee, Emma Tosch, Akanksha Atrey, Kaleigh Clary, Michael L. Littman, and David Jensen. Measuring and characterizing generalization in deep reinforcement learning. Applied AI Letters, 2(4), December 2021. ISSN 2689-5595, 2689-5595. doi:10.1002/ail2.45.

[25] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying Generalization in Reinforcement Learning. In Proceedings of the 36th International Conference on Machine Learning, pages 1282–1289. PMLR, May 2019.

[26] Qucheng Peng, Zhengming Ding, Lingjuan Lyu, Lichao Sun, and Chen Chen. RAIN: RegulArization on Input and Network for Black-Box Domain Adaptation. In Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, pages 4118–4126. International Joint Conferences on Artificial Intelligence Organization, . ISBN 978-1-956792-03-4. doi:10.24963/ijcai.2023/458. URL `https://www.ijcai.org/proceedings/2023/458` .

[27] Qucheng Peng, Ce Zheng, and Chen Chen. Source-free Domain Adaptive Human Pose Estimation. pages 4826–4836. URL `https://openaccess.thecvf.com/content/ICCV2023/html/Peng_Source-free_Domain_Adaptive_Human_Pose_Estimation_ICCV_2023_paper.html` .

[28] Xingyou Song, Yilun Du, and Jacob Jackson. An Empirical Study on Hyperparameters and their Interdependence for RL Generalization. June 2019.

[29] Aravind Rajeswaran, Kendall Lowrey, Emanuel V. Todorov, and Sham M Kakade. Towards Generalization and Simplicity in Continuous Control. In Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

[30] Philipp Moritz and et al. Ray: A distributed framework for emerging AI applications. In Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation, OSDI'18, pages 561–577, USA, October 2018. USENIX Association. ISBN 978-1-931971-47-8.

[31] Stephanie C. Y. Chan, Samuel Fishman, John Canny, Anoop Korattikara, and Sergio Guadarrama. Measuring the Reliability of Reinforcement Learning Algorithms, February 2020.

[32] Denis Yarats, Rob Fergus, Alessandro Lazaric, and Lerrel Pinto. Mastering Visual Continuous Control: Improved Data-Augmented Reinforcement Learning. In Deep RL Workshop NeurIPS 2021, 2021.

[33] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Domains through World Models, January 2023.

Maciej Aleksandrowicz received the M.Sc./Eng. Degree in automatic control and robotics (machine vision focused on AI methods) from the AGH University of Krakow in 2021. From 2019 to 2022, he was working as Robotics Software Engineer in private sector, focusing mainly on industrial manipulators. Currently, as Ph.D. student, his research focuses on deep reinforcement learning solutions dedicated for industrial robots and agents adaptation in model transfers, both from simulation and real environments.
https://orcid.org/0000-0003-3388-5653

Joanna Jaworek-Korjakowska received the Habilitation degree in technical sciences with an emphasis on artificial intelligence from the AGH University of Krakow, Kraków, Poland, in 2019.

She is currently a University Professor and the Director of the Centre of

Excellence in Artificial Intelligence, AGH University of Krakow, where she is also the Deputy Head of the Department of Automatic Control and Robotics. Her research interests include computer vision, data mining, and artificial intelligence, with an emphasis on deep learning methods, anomaly detection, and clustering.

Dr. Jaworek-Korjakowska is an Expert at the Confederation of Laboratories for Artificial Intelligence Research in Europe. She is a member of the Polish Artificial Intelligence Society and the International Dermoscopy Society. She is also an Alumnus of the TOP 500 Innovator Program at Stanford University, Stanford, CA, USA. She received the Honorable Mention Award during the Computer Vision and Pattern Recognition (CVPR'19) Conference (ISIC workshop) and the Bekker Fellowship'22 to conduct research at Stanford University.
https://orcid.org/0000-0003-0146-8652