A survey of the Actor-Critic-Family

Benjamin Möckl

ABSTRACT

Actor-critic algorithms are at the forefront of the fast evolving research field of reinforcement learning. Recent breakthroughs, like beating the world-champion in the game of Go [SHM+16] in 2015, as well as reaching *Grandmaster* level in StarCraftII [VBC+19], used variants of the actor-critic framework. This catalysed the development of many variants and improvements, cross-influenced by the progress made on different model-free reinforcement learning algorithms. This work gives an overview over the state of the art of actor-critic algorithms, with a focus on popular benchmarks like the Arcade Learning Environment [BNVB13] and MuJoCo [?]. First, an introduction to the actor-critic framework is given. Two popular benchmarks are described, followed by the history of different improvements and algorithms over the past 6 years. Finally, current state-of-the-art actor-critic methods are presented. The work concludes with a discussion about the difficulty of comparing different reinforcement learning algorithms.

KEYWORDS

Reinforcement Learning, Actor Critic Methods

1 RELATED WORK

"A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients" [GBLB12] from 2012 is very similar in nature to this work, however it is outdated due to many advancements in the past nine years, which makes a new survey appropriate.

2 INTRODUCTION

2.1 Markov Decision Process

Markov Decision Process *MDP* is a formal description of reinforcement learning (*RL*). It enables precise theoretical statements to be made, to mathematically analyse and formulate reinforcement learning algorithms. Most RL environments and applications can be formulated as an MDP.

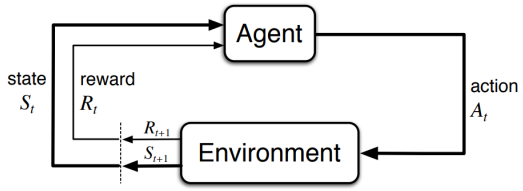

Figure 1: Agent-environment-interaction cycle in an MDP [SB18]

The learner/decision-maker is called the *agent*; the thing it interacts with is called the *environment*. The agent interacts continuously with the environment, by observing its *state*, then choosing an *action* and executing it. The environment changes according to the action, and reports a reward signal back to the agent, together with the new state. This cycle continues, until the environment reaches a terminal state, or a time-limit is reached. This forms a discrete sequence of Observations, Actions and Rewards, denoted with *time* $t \in \mathbb{N}_0$, which increments each cycle, froming a *trajectory* (1).

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, ... \tag{1}$$

$$p : S \times A \times S' \to [0, 1] \tag{2}$$

$$r : S \times A \times S' \to \mathbb{R} \tag{3}$$

A MDP is a tuple $\langle S, A, p, r \rangle$, where S denotes the state-space and A the action-space. p (2) is the state transition probability function, describing the dynamics if the MDP and r (3) is the reward function [SB18].

$$G_t = \sum_{k=0}^{H} \gamma^t R_{t+k+1};\ 0 \leq \gamma \leq 1 \tag{4}$$

$$G_t = R_{t+1} + \gamma G_{t+1} \tag{5}$$

An agent seeks to maximise the *expected discounted return* G (4), where the *discount rate* γ weights immediate returns higher than ones far in the future. This can also be written as a recursive function (5).

To describe the inner workings of an agent in a MDP, further notions are required. The *policy* π (6) maps a probability to each action available in a given state, and is used to decide, how an agent acts. The *value function* V (7) returns the expected sum of rewards from the given state s, under a given policy π. The *Q-function* (8) returns the expected sum of rewards from a given state s after choosing an action a, thus denotes the quality of taking the action in that state on top of the value of s. To isolate the quality of an action in a given state from its value, the notion of the *advantage* A (9) is useful [WSH+16].

$$\pi : s_t, a \to [0, 1] \tag{6}$$

$$V : v_\pi(s) \doteq \mathbb{E}_\pi \left[G_t \middle| S_t = s \right] \tag{7}$$

$$Q : q_\pi(s, a) \doteq \mathbb{E}_\pi \left[G_t \middle| S_t = s, A_t = a \right] \tag{8}$$

$$A : A_\pi(s, a) \doteq v_\pi(s) - q_\pi(s, a) \tag{9}$$

2.2 Actor-Critic Reinforcement Learning

Actor-critic methods try to learn the policy function π (the *actor*) directly, and at the same time learn the value-function (the *critic*) to use as baseline (or as target for the policy, see 4.2 DDPG).

To actually train the policy, the policy gradient theorem (10) provides an expression for the gradient of the performance $\mathcal{J}$ of a policy π_θ with respect to the policy parameters θ without requiring a derivative of the state distribution μ.

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a a_\pi(s, a) \nabla \pi_\theta(s, a) \tag{10}$$

The sum over the whole state distribution μ can be estimated through samples collected under the current policy, as well as the

One-step Actor–Critic (episodic), for estimating $\pi_\theta \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s,\mathbf{w})$
Parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0$, $\alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., to $\mathbf{0}$)
Loop forever (for each episode):
  Initialize S (first state of episode)
  $I \leftarrow 1$
  Loop while S is not terminal (for each time step):
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma\hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\nabla\hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I\delta\nabla \ln\pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Figure 2: One-step Actor-Critic algorithm [SB18].

sum over all actions a in an encountered state s. Rearranging the equation with these modifications yields equation (11), which can be turned into the weight-update rule (12) of the *REINFORCE* algorithm [SB18].

$$\nabla J(\theta) \propto \mathbb{E}_\pi \left[G_t \frac{\nabla\pi_\theta(A_t,S_t)}{\pi_\theta(A_t,S_t)} \right] \tag{11}$$

$$\theta_{t+1} \doteq \theta_t + \alpha G_t \frac{\nabla\pi_\theta(A_t,S_t)}{\pi_\theta(A_t,S_t)} = \theta_t + \alpha G_t ln(\pi_\theta(A_t,S_t)) \tag{12}$$

It can be shown, that an arbitrary baseline b can be subtracted from G, as long as b does not vary with a. This can greatly reduce variance. A natural choice for a good baseline is the value function v. The actor-critic algorithm goes one step further and uses the state-value estimate of the second state of the given transition $(S_t, A_t, R_{t+1}, S_{t+1})$, which can be used to assess the chosen action A_t due to it being one step in the future. When the state-value function is used to assess actions in this way it is called a critic, and the overall policy-gradient method is termed an actor–critic method [SB18]. The derived update rule can be seen in (13) and (14), a full overview over the basic algorithm can be found in figure 2.

$$\theta_{t+1} \doteq \theta_t + \alpha(G_{t:t+1} - v(S_t))ln(\pi_\theta(A_t,S_t)) \tag{13}$$

$$\theta_{t+1} \doteq \theta_t + \alpha(R_{t+1} + \gamma v(S_{t+1}) - v(S_t))ln(\pi_\theta(A_t,S_t)) \tag{14}$$

3 BENCHMARKS

3.1 ALE

The goal of reinforcement learning research is to develop generally competent agents, which comes with the problem on how to evaluate and compare them. The Arcade Learning Environment *ALE* [BNVB13], released in 2013, is one of the most popular benchmarks for reinforcement learning algorithms. It combines a rich palette of different games, which are challenging for reinforcement learning for different reasons: Some require long-term planning, some have very sparse rewards and require sophisticated exploration, and due to their nature of being popular games, there exist strong human baselines, that in some of the games are yet to be beaten. The inputs to the environment are 18 discrete actions defined by the joystick controller, the outputs are a 2D array of 160 by 210 7-bit pixels, as well as a reward signal. All of the software, including some baseline agents, is publicly available.

3.2 MuJoCo

Continuous control problems are at major interest in reinforcement learning, due to its close connection to robotics (and controlling robots with RL) and the real world. For decades, simple physics simulations like cart-pole have been used to evaluate agents or controllers. To fit reinforcement learning, interaction with such environments is abstracted to receiving a state and reward signal, and sending an action signal. The reward signal usually contains only sparse information in case of failure or success, to impose the credit assigning problem of how to avoid failure or how to achieve success to the agent. An early description of cart-pole can be found in [BSA83] from 1983.

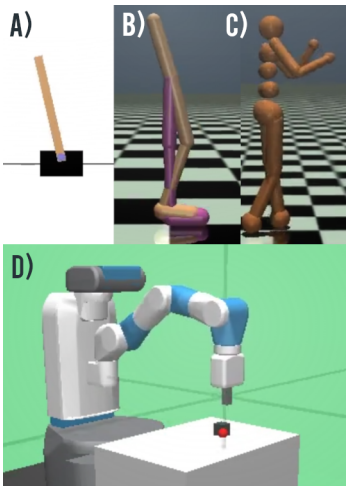

Figure 3: MuJoCo OpenAI Gym Environments.
A) *CartPole_v1*,
B) *Walker2d_v2*,
C) *Humanoid_v2*
D) *FetchPush_v1*

MuJoCo [TET12] is a publicly available physics engine, which runs very efficient. It is used in a variety of continuous control benchmarks, most famous for learning bipedal locomotion or teaching a whole human mannequin to walk, as well as robotics simulations and classic control problems like the one described above, examples can be seen in figure 3. Since the release of *OpenAI Gym* [BCP+16] in 2016, these environments are conveniently accessible, and already served with a python wrapper, which only exposes an observation, reward, and action interface.

4 HISTORY OF AC METHODS

4.1 Methods Predating Deep-Learning In RL

The idea of splitting learning components of an reinforcement learning algorithm into an actor, which directly represents the policy, and a critic, which evaluates the actions taken by the actor and provides a learning signal to it, was already described in 1983 [BSA83], the original sketch can be seen in figure 4. Concrete actor-critic algorithms for solving Markov decision processes, where the dynamics model of an environment is not available but can be sampled, were first described in 1999, together with the proposal of simultaneously improving the actor and the critic while preserving convergence properties [KB99]. In 2009, temporal difference learning was incorporated [BSGL09].

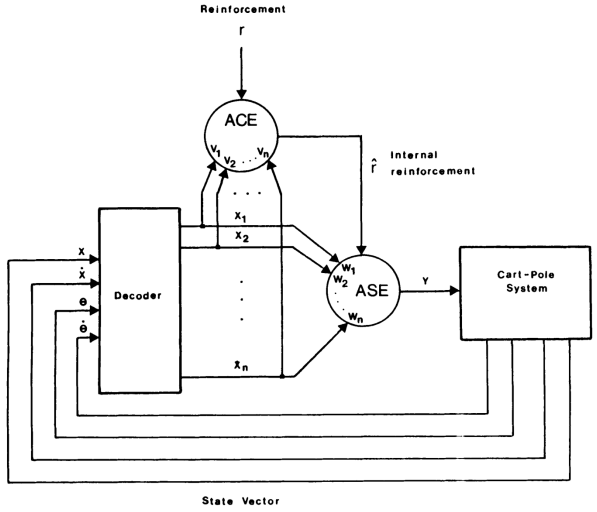

Figure 4: Actor-critic framework described by Sutton and Barto [BSA83] in 1983.
Original caption: "Fig. 3. ASE and ACE configured for pole-balancing task. ACE receives same nonreinforcing input as ASE and uses it to compute an improved or internal reinforcement signal to be used by ASE."

The first off-policy actor-critic algorithm *Off-PAC* [DWS12] was formulated in 2012. Finally, in 2014 the deterministic policy gradient algorithm *DPG* [SLH+14] introduced using the gradient of the action-value function to update the policy.

4.2 2013 - 2015

The successful use of deep neural networks as function approximators in reinforcement learning in 2013, which enabled it to learn to play computer games from raw pixels and match human performance ([MKS+13], [MKS+15]) kick-started the development of deep-learning-based algorithms. While this first work was using value-based Q-Learning, soon deep neural networks were also incorporated into policy-gradient and actor-critic methods.

Trust Region Policy Optimization *TRPO* is a policy-learning based method from 2015. It guarantees policy improvement with non-trivial step sizes, by taking the KL-divergence between the old policy (under which a batch of data was collected) and the current policy as a measure of a trust region, within which the collected samples remain valid for further policy updates, due to their similarity. This increases the stability of the vanilla Policy-Gradient algorithm and can optimize neural-network based policies. It matched the *DQN* performance in the tested *ALE* environments, as well as matching or surpassing the performance of prior works in continuous control tasks like 2d-walker [SLA+15]. It is not an actor-critic algorithm, but mentioned here for completeness of the timeline, due to it being the first algorithm, which robustly was able to learn 2d and later 3d bipedal locomotion in humanoid robots.

In 2015, *AlphaGo* [SHM+16] was released, which was able to defeat the human European Go champion by 5 games to 0. It uses policy-gradient to train a pre-trained policy-network, and a separate value-network, which is trained with supervised learning. Both networks were then combined in a monte-carlo tree-search algorithm. AlphaGo is not an actor-critic algorithm, but was a major breakthrough for deep reinforcement-learning.

DDPG [LHP+15] is the deep-network version of *DPG*. In DDPG, a batch of state-transitions is collected (sampled on- or off-policy, using some exploration noise), through which a Q-function is learned using Q-Learning (or a multi-step variant of Q-Learning). This results in good sample efficiency and off-policy learning capabilities. At the same time a policy is learned to maximise the prediction of the Q-function for the given states from the batch. It enabled continuous control (acting in a continuous action space) for solving various physics tasks, and additionally could be trained directly from raw pixel inputs.

4.3 2016

The main improvement in Asynchronous Advantage Actor-Critic *A3C* is the usage of multiple actors, acting simultaneously on separate instances of the environment. This searches most likely different parts of the environment, performing a stabilizing role similar to experience replay in DQN algorithms. Additionally, the authors added the entropy of the policy to the actor-loss due to it improving exploration by discouraging premature convergence to suboptimal deterministic policies [MBM+16]. Due to scaling well with an increase in compute, A3C still performs close to the current state-of-the-art algorithms.

The framework *UNREAL* [JMC+16] from 2017 extended A3C with three unsupervised auxilary tasks, leading to a 10x speedup in learning and new state-of-the-art performance on *ALE*.

4.4 2017

Proximal Policy Optimisation *PPO* from 2017 is an direct improvement to the previously mentioned *TRPO*. It moves the trust-region concept into the loss itself through a minimum and a clipping operator, limiting parameter updates once the similarity ratio of old and new policy gets too big. This results in a much simpler algorithm with better sample complexity, that scales great with compute, outperforming other online policy gradient methods [SWD+17]. This algorithm as-well is not an actor-critic algorithm, but is included due to it being the most popular policy-gradient method to date, and being used in the training of OpenAI Five, which did win against the world-champion team in the popular e-sports game DotA2 in 2019 [BBC+19].

Rainbow DQN [HMVH+18], also from 2017, is a Q-Learning approach, which combined several independent improvements to the DQN algorithm. It is listed here despite not being an actor-critic method, because of its performance on the *ALE* environment, beating *A3C*, and because some of these improvements are applicable to actor-critic methods.

4.5 2018

The twin delayed deep deterministic policy gradient algorithm *TD3* from 2018 is one example of a Q-Learning improvement applied to an actor-critic setting. It builds on Double Q-Learning [Has10], similar to *DDQN* [VHGS16], but with multiple changes to adapt

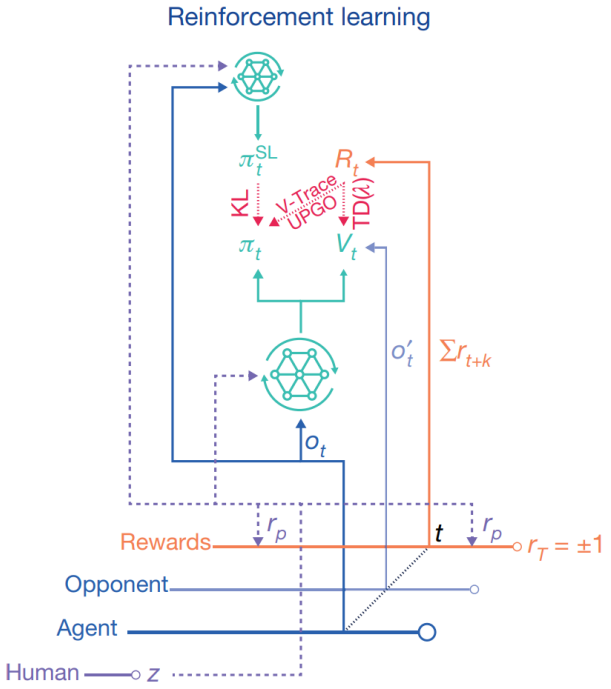

Figure 5: Reinforcement learning framework used to train *AlphaStar*. Original caption: "[...] In reinforcement learning [...], human data are used to sample the statistic z, and agent experience is collected to update the policy and value outputs via reinforcement learning (TD(λ), V-trace, UPGO) combined with a KL loss towards the supervised agent. [...]" [VBC+19]

it effectively for actor-critic learning. It uses two critics instead of one, and uses a minimum operator across both for the learning target, to remove overestimation bias of the Q-values of the critic. This resulted in a new state-of-the-art performance on the continuous control *MuJoCo* benchmark across all tested environments.

Soft Actor Critic *SAC* [HZAL18], [HZH+18] from 2018 can be seen as the maximum entropy version of *DDPG*. The target of the Q-function is adapted to incorporate a max-entropy term of the current policy added into the expected future reward, its influence is controlled by an automatically tuned temperature parameter. The actor therefor learns to maximise reward while acting as randomly as possible. SAC achieved state-of-the-art performance on a range of continuous control tasks, outperforming all previous works. It also is stable and reproduces very similar results across different random seeds, which was one of the major disadvantages of the prior *DDPG*. In 2019, SAC was extended to *SAC-Discrete* [Chr19] to also work with discrete action spaces. In a subset of 20 Atari games from the *ALE*, it was able to surpass the performance of *Rainbow-DQN* in about half of them, while falling significantly behind in the rest.

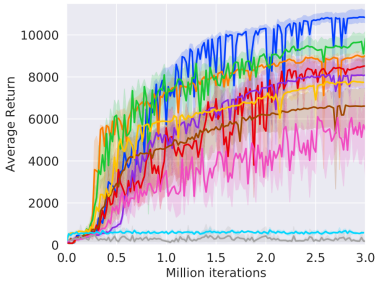

Figure 6: Environment humanoid_v2
blue : *DSAC*
orange : *SAC*
green : *Double – QSAC*
red : *Single – QSAC*
purple : *TD4*
brown : *TD3*
pink : *DDPG*
grey : *D4PG*
yellow : *PPO*
cyan : *TRPO*
[DGL+21]

4.6 2019

In 2019 *AlphaStar* [VBC+19] was able to reach Grandmaster level in the full game of StarCraft II. At its core is a sophisticated policy network containing Deep LSTM layers, self-attention mechanisms and more. Similar to the previous *AlphaGo* framework, it first undergoes supervised learning from human data, and is then improved with reinforcement learning on self-play. The used RL algorithm is similar to *A3C*; Multiple actors collect trajectories on multiple environments. Asynchronous updates are performed on the policys replaying these experiences for the value and policy updates. An overview of this framework can be seen in figure 5.

4.7 State Of The Art Methods Of 2020 and 2021

4.7.1 DSAC. Distributional Soft Actor Critic *DSAC* integrates learning the reward distribution instead of the expected future reward (from Distributional DQN [BDM17]) with *SAC*, which adds a max-entropy term to its objective. This is another example of a Q-Learning improvement applied to an actor-critic setting. It stands as the current state-of-the-art in continuous control settings, evaluated on *MuJoCo* environments, and beating TD3, SAC, PPO and others, as can be seen in figure (6). The goal of incorporating distribution learning was to reduce overestimation bias of learned Q-values, similar to what *DDQN* and *TD3* tried to achieve. The paper [DGL+21] shows, that this indeed works well, and results in more stable learning. A follow-up paper [MZX+20] also discusses using the learned distributions for risk-assessment.

4.7.2 STAC. Self Tuning Actor Critic *STAC* uses metagradients to self-tune all differentiable hyperparameters in an actor-critic loss function, online and within a single lifetime. The paper [ZXV+20] shows, that the self-tuning increased the performance of the original algorithm, while being computational efficient and robust to their hyperparameters.

4.7.3 WD3. Weighted Delayed Deep Deterministic Policy Gradient *WD3* builds on *TD3*. TD3 tried to overcome the overestimation bias of Q-learning (of its critic) by introducing a second critic, and a minimum operator across both of them. This however, argue the authors of WD3 [HH20], introduces an underestimation bias. To unite these opposites, they introduce a new parameter β, which weighs the influence of the two critics, thus regulating between overestimation and underestimation, reducing estimation error and therefor increasing performance. WD3 outperforms the

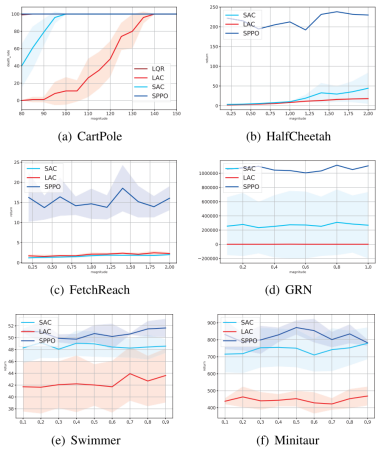

Figure 7: X-axis: the magnitude of the applied disturbance

Y-axis: the death rate in CartPole (a) and the variation of the cumulative cost in other examples (b) to (d).
All of the trained policies were evaluated for 100 trials in each setting. [HZWP20]

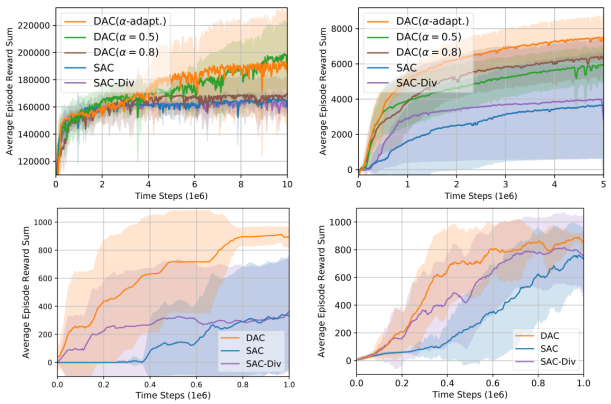

Figure 8: DAC performance [HS21]
top − left : *HumanoidStandup*
top − right : *DelayedHalfCheetah*
bottom − left : *SparseHalfCheetah*
bottom − right : *SparseHopper*

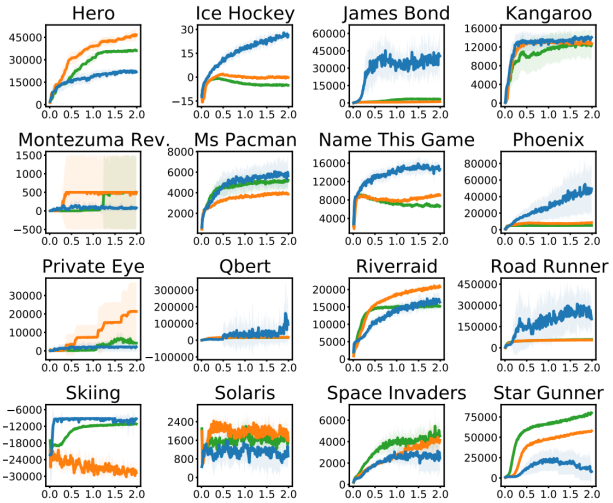

Figure 9: DreamerV2 performance on ALE [HLNB20]
blue : *DreamerV2*
green : *RainbowDQN*
orange : *IQN*

state-of-the-art algorithms in continuous control tasks from the MuJoCo environment on all tested experiments. Only one of these environments, *InvertedDoublePendulum*, did overlap with the ones tested in *DSAC* (4.7.1), where the performance-difference lies within the margin of error.

4.7.4 LAC. Lyapunov Actor-Critic *LAC* uses a so called Lyapunov critic function, to guarantee closed-loop stability. The authors of the paper [HZWP20] argue, that stability is the most important property of any control system because of its close relation to safety, robustness, and reliability in robotic systems. The algorithm was evaluated on continuous 3D control tasks from MuJoCo, and is significantly more robust while performance is on par with compared SAC and SPPO methods. This was measured by introducing random disturbances to the agent, and evaluating the impact this had on performance, tested with a range of different magnitudes of disturbances, see figure 7.

4.7.5 DAC. Diversity Actor Critic *DAC* [HS21] builds on top of *SAC*, by extending the entropy maximisation term of the objective from only the current policy to a weighted sum of the current policy action distribution and the sample action distribution from the replay-buffer. The intuition is not only acting as random as possible while maximising performance (SAC), but also trying to differ from previous strategies. The algorithm is evaluated on a set of sparse and delayed MuJoCo environments (sparse means only a reward of 1 is received, if a objective is met, and 0 otherwise; delayed means only in fixed time-intervals D, reward, which accumulated the in the mean-time, is received). DAC consistently and significantly outperforms SAC in these scenarios, see figure 8.

4.7.6 DreamerV2. DreamerV2 [HLNB20] is a model-based actor-critic algorithm, that learns behaviours purely from predictions in the compact latent space of its powerful world model. It is the first agent, that achieved human-level performance on ALE, training its policy only inside its world model, and it significantly outperforms the top single-GPU agents *Rainbow DQN* and *IQN*, in final agent performance, sample-efficiency and wall-clock time. It learns its world model from a replay buffer of past experiences, then learns an actor and a critic from imagined sequences of the world models compacted state representation, and execute the actor on the environment to collect new experiences. Learning curves on several Atari games from ALE can be seen in figure 9.

4.8 Advances In Related Fields 2020 and 2021

This section presents some actor-critic algorithms, whose improvements were not focused on performance, but on different RL problems.

4.8.1 FP16 SAC. This work [BCDS+21] successfully adapted low precision training (FP16) to *SAC* with a set of easy-to-implement

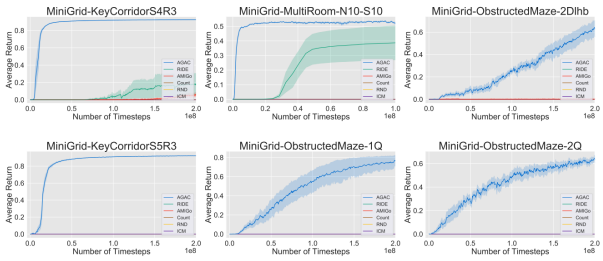

Figure 10: AGAC performance on VizDoom environments [FFP+21].

modifications. Their evaluation shows the low precision SAC variant *FP16 SAC* to match its full precision counterpart, with lower memory and compute costs.

4.8.2 LPG. Learned Policy Gradient *LPG* [OHC+20] is the result of an approach at meta-learning for discovering new reinforcement learning algorithms. The algorithm was shown to discover concepts like value functions and bootstrapping by interacting with a set of environments. Surprisingly to the authors, the agent generalized effectively to complex Atari games and achieved non-trivial performance, matching the performance of A2C in some of the games.

4.8.3 SEAC. Shared Experience Actor-Critic *SEAC* applies experience sharing between multiple agents to the actor-critic framework. It updates the actor and critic parameters of an agent by combining gradients computed on the agent's experience with weighted gradients computed on other agents' experiences. It achieves state-of-the-art performance in sparse multi-agent environments, with only a minimal increase of 3% in compute time.

4.8.4 AGAC. Adversarially Guided Actor-Critic *AGAC* introduces a third component to actor and critic: the *adversary*. It tries to mimic the actor by minimizing the KL-divergence between their action distributions, while the actor tries to differentiate itself from it. Similar to *DAC*, this induces search for novel strategies, creating innovation in extremely sparse environments. AGAC outperforms the previous state-of-the-art algorithms on various hard-exploration tasks, see figure 10.

5 CONCLUSION

Reinforcement learning progressed at a fast pace over the last 6 years. Every year the state-of-the-art was improved multiple times, by sometimes very different strategies. Improvements from Q-Learning (most notably the ones aggregated in Rainbow DQN [HMVH+18]) were integrated into the actor-critic framework and showed significant improvements. While researching for this work, also some deficits became obvious: Comparability between different methods is not trivial. While most algorithms use popular benchmarking platforms like ALE and MuJoCo, often other important performance metrics like compute efficiency, scale-ability with more compute, sample-efficiency and robustness of the trained agents remain to be explored. The scale-ability aspects of algorithms is expected to become more relevant in the future, due to Moores Law as proposed by Richard Sutton [Sut19], and has already shown its impact with the recent breakthroughs in Go [SHM+16] as well as Dota2 [BBC+19] in 2019, which used the basic PPO algorithm, but scaled up the training architecture to hundreds of GPUs.

REFERENCES

[BBC+19] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[BCDS+21] Johan Björck, Xiangyu Chen, Christopher De Sa, Carla P Gomes, and Kilian Weinberger. Low-precision reinforcement learning: Running soft actor-critic in half precision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 980–991. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/bjorck21a.html.

[BCP+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016, 1606.01540. URL http://arxiv.org/abs/1606.01540.

[BDM17] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/bellemare17a.html.

[BNVB13] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013.

[BSA83] Andrew G Barto, Richard S Sutton, and Charles W Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics*, (5):834–846, 1983.

[BSGL09] Shalabh Bhatnagar, Richard S Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor–critic algorithms. *Automatica*, 45(11):2471–2482, 2009. doi:10.1016/j.automatica.2009.07.008.

[Chr19] Petros Christodoulou. Soft actor-critic for discrete action settings. *CoRR*, abs/1910.07207, 2019, 1910.07207. URL http://arxiv.org/abs/1910.07207.

[DGL+21] Jingliang Duan, Yang Guan, Shengbo Eben Li, Yangang Ren, Qi Sun, and Bo Cheng. Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2021. doi:10.1109/TNNLS.2021.3082568.

[DWS12] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[FFP+21] Yannis Flet-Berliac, Johan Ferret, Olivier Pietquin, Philippe Preux, and Matthieu Geist. Adversarially guided actor-critic. *CoRR*, abs/2102.04376, 2021, 2102.04376. URL https://arxiv.org/abs/2102.04376.

[GBLB12] Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

[Has10] Hado Hasselt. Double q-learning. *Advances in neural information processing systems*, 23:2613–2621, 2010.

[HH20] Qiang He and Xinwen Hou. Reducing estimation bias via weighted delayed deep deterministic policy gradient. *arXiv preprint arXiv:2006.12622*, 2020.

[HLNB20] Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *CoRR*, abs/2010.02193, 2020, 2010.02193. URL https://arxiv.org/abs/2010.02193.

[HMVH+18] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[HS21] Seungyul Han and Youngchul Sung. Diversity actor-critic: Sample-aware entropy regularization for sample-efficient exploration. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4018–4029. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/han21a.html.

[HZAL18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/haarnoja18b.html.

[HZH+18] 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. *CoRR*, abs/1812.05905, 2018, 1812.05905. URL http://arxiv.org/abs/1812.05905.

[HZWP20] Minghao Han, Lixian Zhang, Jun Wang, and Wei Pan. Actor-critic reinforcement learning for control with stability guarantee. *IEEE Robotics and Automation Letters*, 5(4):6217–6224, 2020. doi:10.1109/LRA.2020.3011351.

[JMC+16] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *CoRR*, abs/1611.05397, 2016, 1611.05397. URL http://arxiv.org/abs/1611.05397.

[KB99] Vijaymohan R Konda and Vivek S Borkar. Actor-critic–type learning algorithms for markov decision processes. *SIAM Journal on control and Optimization*, 38(1):94–123, 1999.

[LHP+15] 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 preprint arXiv:1509.02971*, 2015.

[MBM+16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/mniha16.html.

[MKS+13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013, 1312.5602. URL http://arxiv.org/abs/1312.5602.

[MKS+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. doi:10.1038/nature14236.

[MZX+20] Xiaoteng Ma, Qiyuan Zhang, Li Xia, Zhengyuan Zhou, Jun Yang, and Qianchuan Zhao. Distributional soft actor critic for risk sensitive learning. *CoRR*, abs/2004.14547, 2020, 2004.14547. URL https://arxiv.org/abs/2004.14547.

[OHC+20] Junhyuk Oh, Matteo Hessel, Wojciech M. Czarnecki, Zhongwen Xu, Hado van Hasselt, Satinder Singh, and David Silver. Discovering reinforcement learning algorithms. *CoRR*, abs/2007.08794, 2020, 2007.08794. URL https://arxiv.org/abs/2007.08794.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. The MIT Press, 2 edition, 2018. URL https://lccn.loc.gov/2018023826.

[SHM+16] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016. doi:10.1038/nature16961.

[SLA+15] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1889–1897, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schulman15.html.

[SLH+14] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 387–395, Bejing, China, 22–24 Jun 2014. PMLR. URL https://proceedings.mlr.press/v32/silver14.html.

[Sut19] Richard Sutton. The bitter lesson. http://www.incompleteideas.net/IncIdeas/BitterLesson.html, 2019.

[SWD+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017, 1707.06347. URL http://arxiv.org/abs/1707.06347.

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012. doi:10.1109/IROS.2012.6386109.

[VBC+19] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. doi:10.1038/s41586-019-1724-z.

[VHGS16] 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. URL https://ojs.aaai.org/index.php/AAAI/article/view/10295.

[WSH+16] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1995–2003, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/wangf16.html.

[ZXV+20] Tom Zahavy, Zhongwen Xu, Vivek Veeriah, Matteo Hessel, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. A self-tuning actor-critic algorithm. *arXiv preprint arXiv:2002.12928*, 2020.