Honey, I Shrunk The Actor: A Case Study on Preserving Performance with Smaller Actors in Actor-Critic RL

Siddharth Mysore
Department of Computer Science
Boston University
Boston, U.S.A.
sidmys@bu.edu

Bassel El Mabsout
Department of Computer Science
Boston University
Boston, U.S.A.
bmabsout@bu.edu

Renato Mancuso
Department of Computer Science
Boston University
Boston, U.S.A.
rmancuso@bu.edu

Kate Saenko
Department of Computer Science, Boston University
Co-affiliated with MIT-IBM Watson AI Lab
Boston, U.S.A.
saenko@bu.edu

***Abstract*—Actors and critics in actor-critic reinforcement learning algorithms are functionally separate, yet they often use the same network architectures. This case study explores the performance impact of network sizes when considering actor and critic architectures independently. By relaxing the assumption of architectural symmetry, it is often possible for smaller actors to achieve comparable policy performance to their symmetric counterparts. Our experiments show up to 99% reduction in the number of network weights with an average reduction of 77% over multiple actor-critic algorithms on 9 independent tasks. Given that reducing actor complexity results in a direct reduction of run-time inference cost, we believe configurations of actors and critics are aspects of actor-critic design that deserve to be considered independently, particularly in resource-constrained applications or when deploying multiple actors simultaneously.**

***Index Terms*—Reinforcement Learning, Artificial Intelligence, Actor-Critic Asymmetry, Case Study**

I. Introduction

As a tool for developing machine learning (ML) based artificial intelligence (AI) for games, deep reinforcement learning (RL) has received increased interest in recent years. RL has been used to develop game-play AI ranging from human-like [1]–[3] to superhuman [4], [5], with a wide array of applications, including game testing [1]–[3], competition against human players [4], [5], and simply as a means to develop more diverse styles of in-game play [2]. While RL can enable learning models for interesting behavior, the models can be significantly more computationally expensive than classical controllers or heuristic-based algorithms, particularly if models need to be deployed at scale (e.g. when controlling multiple in-game characters). In exploring how changing model complexity affects learning performance, we discovered that, for a class of RL algorithms, actor-critic RL, the size and run-time cost of learned actors could often be significantly reduced, while preserving performance parity with larger networks.

RL algorithms typically attempt to train control policies that maximize expected rewards on a given task, where the action-to-reward relationship is treated as a black-box. Since a function that maps actions to rewards is not known a priori, a value function is learned to estimate it and is in turn used to optimize the policy to maximize rewards. Actor-critic methods are currently amongst the most common class of model-free deep reinforcement learning (RL) algorithms. They are characterized by their separation of the functions representing the actor — the RL policy — and the critic — the value function estimator. This allows actor-critic methods to theoretically take advantage of the improved sample efficiency of value-based RL approaches like Q-learning [6] while also capitalizing on the ability to use policy-based approaches such as policy gradient [7], enabling learning on continuous action domains and improved robustness to stochasticity.

The separation of actors and critics means that only the actor is required during inference, which saves the run-time cost of estimating policy value, as it is mainly useful only during training. Beyond excluding the critic, further reducing run-time compute requires reducing actor sizes, leading us to ask how small actors can get before losing learning efficacy. We noted that works considering the impact of network architectures on RL performance typically adjust both actor and critic sizes equivalently [8]–[12], i.e. their architectures are kept 'symmetric'. We also see this implicit assumption in game-development tools such as Unity's ML-agents API [13]. Nonetheless, nothing in the theoretical foundations precludes using different architectures for actors and critics. In this work, we study if symmetry is important to actor-critic methods and how relaxing the self-imposed symmetry impacts performance and model complexity. We find this actor-critic architectural 'asymmetry' allows for a significant reduction in actor network sizes without compromising the algorithms' performances.

We hypothesize that modeling the value function associated with an environment often requires a higher capacity for modeling complexity. This is because the critic needs to develop an understanding of both the dynamics of a black-box system and how they contribute to learning rewards. Actors try to maximize the value estimate, which is done by gradient ascent on the value function estimate. Evidence suggests this is often easier to optimize, requiring less modeling complexity, allowing for smaller actors. We test 4 popular actor-critic algorithms on 9 different environments of varying system dynamics and complexity. We show that it is not simply that network architectures often presented in contemporary literature have a lot of headroom with model complexity but that actors often require less modeling capacity than critics, allowing them to function with smaller architectures. The practical implications of this intuitive observation have received little attention, both in the game AI and broader ML communities, and no prior study has been conducted to systematically understand and evaluate the potential of this architectural asymmetry in actor-critic methods. Our results show that architectural asymmetry enables reductions in actor sizes in excess of 98% even over actors that have already been shrunk through symmetric architecture tuning, with an average reduction of 77% in model parameters.

II. Preliminaries

Actor-critic methods separate the actor and critic functions of an RL algorithm. The actor function represents the policy, i.e. the core of the decision making aspect of the RL algorithm, while the critic estimates the value of trajectories of state transitions under the policy. Actor-critic algorithms are typically trained through sampling-based value iteration and policy gradients, where critics are trained to minimize the loss on the measured returns against the estimated value and actors are trained to maximize the estimated value of the trajectories generated under the actor policy. Specifics of how the optimization criteria are estimated vary between algorithms. Prominent contemporary algorithms include A2C [14], A3C [15], PPO [16], TRPO [17], DDPG [18], TD3 [19], and SAC [20], to name a few.

Network architecture is one of many hyper-parameters that can impact actor-critic performance [9], [11], [21]–[24], but actors and critics are rarely considered separately. We believe, however, that this is an important consideration to make, but one that is easily missed given that tools most commonly available typically hard-code actor/critic symmetry. Actor-critic algorithms are widely represented in common baselines and benchmarks, but when surveying existing baseline codes including OpenAI Baselines [25], OpenAI Spinning Up [26], Stable Baselines (v2 and v3) [27], [28], Unity ML-agents [13], Tensorflow RL Agents [29], and rllab [12], we found that actor and critic architectures are generally entangled and symmetric — in most cases allowing users to define an architecture for both the actor and critic networks together but not independently. There are, of course, many aspects of network architecture to consider, including input handling, layer sizes, network depths, recurrence, information sharing between nodes, etc. For the purposes of this study, and in the interest of not diluting the focus on variables, we focused on layer sizes while keeping all other parameters fixed as per values found in benchmarks for 2-hidden-layer networks.

III. Litmus Test: A Toy Problem

We start with a toy-problem that we know could be solved, in principle, by a very small, linear (1-neuron) actor. We constructed a simple one-dimensional goal-tracking problem without complex dynamics or system noise (details in Appendix A). The agent's observed state o is the difference between the desired goal g and current internal state s, $o = g - s$, where $s, g \in [-1, 1]$. Actions a affect the internal state such that the next state $s' = s + a$, while the observed reward $r = -|g - s'|$. This problem is solved with the ideal policy $a^* = o$, which can be represented by a single neuron.

We considered every algorithm included in the OpenAI Spinning Up code-base [26], which includes TRPO, PPO, DDPG, SAC and TD3. The code-base, by default, allows users to jointly adjust the number of hidden layers and the number of hidden neurons per layer in the actor and critic networks. For all algorithms tested however, when trained with the implicit assumption of actor and critic network symmetry, it was not possible for agents to learn to solve even this simple problem with a 1-neuron policy. We will note however that it is possible for the single neuron weight to be randomly initialized close to 1, which effectively solves the problem, though does not count as a 'learned' solution. We tested network architectures including $|1|$, $|4, 4, 1|$, $|8, 8, 1|$ and $|16, 16, 1|$, where numbers within the vertical brackets provide a comma-delimited representation of the number of neurons per layer in the neural network, with the last layer being the 1-D output. When structured symmetrically, we found that agents did not consistently train to solve this problem for networks smaller than $|8, 8, 1|$, but were most stable with $|16, 16, 1|$. Fig. 1 shows training trends for each of the algorithms trained on this problem while Fig. 2 compares the quality of learned control.

Note that a linear 1-neuron actor policy is not capable of learning to solve this problem when paired with a 1-neuron critic, but learns viable policies when trained with a larger critic. Trends in rewards and losses during training indicated that an improvement in value estimation was followed shortly thereafter by an improvement in the observed rewards. Based on this observation, we hypothesized that it was the smaller critics that were lacking in their ability to model the underlying value functions. To test this, we restricted the actor size to be 1-neuron and set the critic to be $|16, 16, 1|$. By breaking from the implicit assumption of actor-critic symmetry and separately considering the actor and critic architecture, it was possible to consistently train 1-neuron actors where the smaller symmetrically sized 1-neuron critics had failed. Figs. 1 and 2 demonstrate that, in addition to allowing for successful training, the asymmetric models share similar training and loss optimization characteristics to their larger symmetric counterparts.

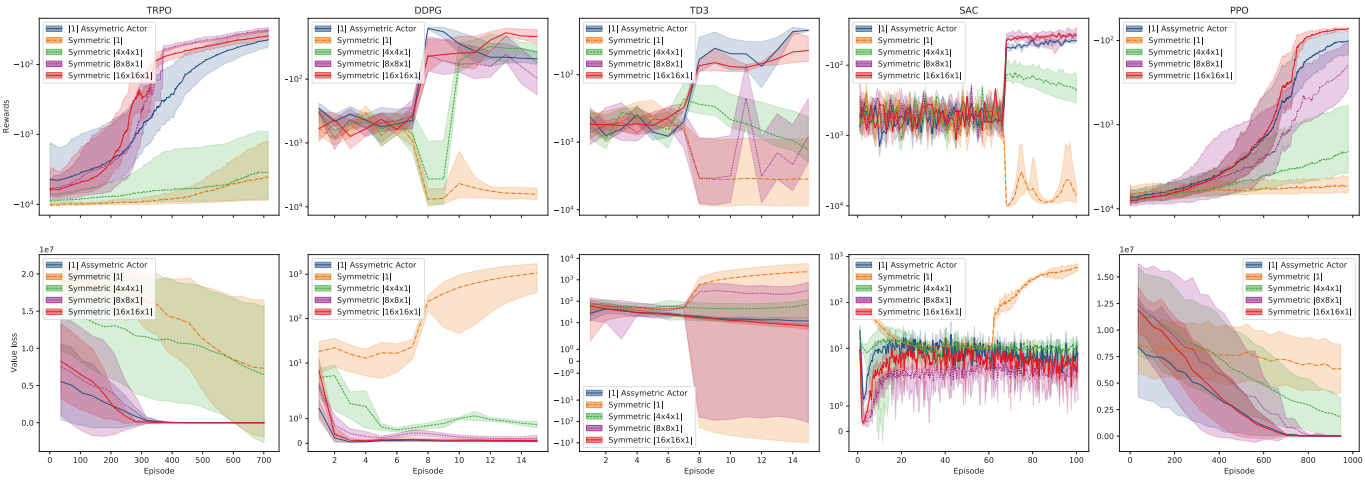

Fig. 1. Comparison of how rewards and value (critic) losses evolve during training on a simple toy-problem. Results suggest a strong correlation between critic size and policy viability. Symmetric policies share the same size as their critics while the asymmetric actor has a 1-neuron policy and a $|16, 16, 1|$ critic. Note that the asymmetric actors performs very similarly to the symmetric $|16, 16, 1|$ policies and share similar value loss profiles too.

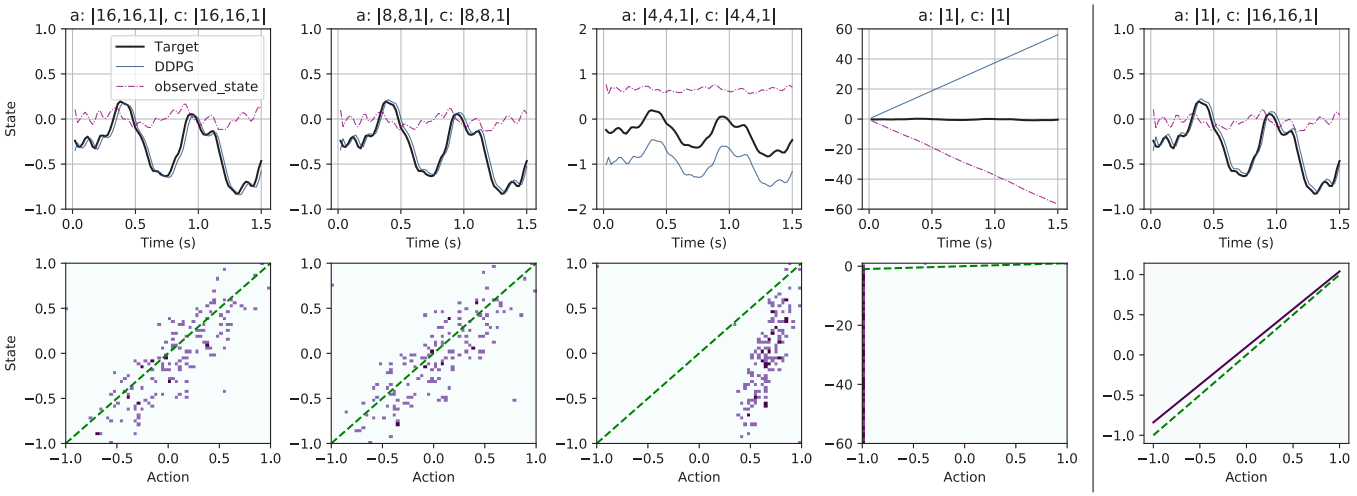

Fig. 2. Representative samples of DDPG behavior for different actor "a" and critic "c" sizes. This visualizes the ability of differently configured agents, or lack thereof, to solve the simple toy problem described in Section III. We compare both the state response over a 1.5s window, as well as a 2-D histogram of state-action response for the sampled agents over the valid state and action space. The green dotted line in the state-action plots represents the ideal action policy, i.e. $a = o$. Note that only the asymmetric linear actor (right) is able to learn a near-ideal policy, while its symmetric counterpart simply fails to learn and larger networks learn more (needlessly) complex policies. Crucially, this shows that, with sufficient modeling capacity in the critic, a single neuron indeed learns to solve this simple problem, while an assumption of actor-critic symmetry would not allow for this.

IV. Architectural-Asymmetry Benchmarked

It was evident that separately defining actor and critic architectures was at least helpful in the case of a simple toy problem. To better understand the practical implications of actor-critic asymmetry, we evaluated how different network sizes affect performance on a number of common benchmark tasks from OpenAI's Gym [30] benchmark environments, in addition to games from the Pygame Learning Environment (PLE) [31] and Unity's ML-agents [13] example training environments.

To better characterize how network size can affect performance, and specifically how smaller networks impact performance, we consider two aspects of the actor-critic architecture: (i) the smallest symmetric actor-critic architecture that can meaningfully solve the problem, and similarly (ii) the smallest asymmetric architecture with comparable performance. Treating $|400, 300|$ (the DDPG and TD3 default) as the upper bound on a set of possible 2-hidden-layer architectures, we consider the hidden-layer structures: $|1, 1|$, $|4, 4|$, $|8, 8|$, $|16, 16|$, $|32, 32|$, $|64, 64|$, $|128, 128|$, $|256, 256|$ and $|400, 300|$ (note that these do not include the output layer, which is defined with respect to the action space for each test environment). This list covers typical architectures found in actor-critic literature but also goes smaller to get a sense the lower bound on the network complexity needed for the considered tasks.

Experimental Method: We determine appropriate sizes for the smallest symmetric and asymmetric architectures sequentially. By running the algorithms on 6 random seeds with their originally reported architectures, we establish a baseline and a threshold target performance for smaller networks. We

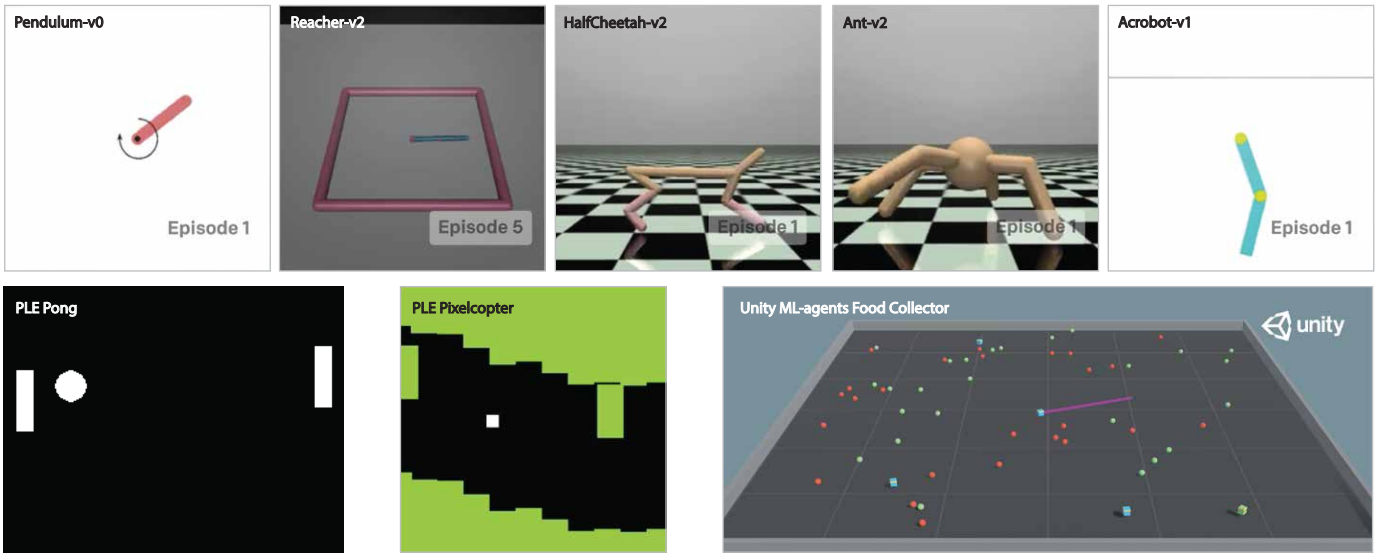

Fig. 3. Environments used to benchmark various actor-critic algorithms. Environments were chosen to cover a diverse set of dynamics and objectives.

then use this threshold, with a 10% tolerance on the lower-bound, to perform a binary search over the actor-critic network sizes to find the smallest one that can solve the problem while preserving actor-critic symmetry, similarly averaging performance over 6 seeds. Finally, by locking the critic architecture to the smallest symmetric size found, we repeat the binary search over the possible actor architectures to find the smallest asymmetric actor that would solve the problem. This allows us to determine more conclusively that, in cases where asymmetry allows for actor size reduction, it is not simply attributable to excess modeling capacity in both the actor and critic networks when using baseline network architectures.

Learning Tasks: In an effort to cover a wide range of possible system dynamics, we experimented with 8 different training environments (Fig. 3). From the Gym benchmarks, we tested controls tasks of varying levels of complexity: Pendulum-v0, Reacher-v2, Ant-v2, HalfCheetah-v2 and Acrobot-v1. Agents have control over various actuators and use information observed about joint dynamics to control joint actuation. From PLE, we tested an implementation of Pong, where the RL agent competes against a heuristic-based opponent AI, as well as Pixelcopter, an obstacle-avoidance side-scroller. For both the PLE games, the agents directly observe relevant game-state information such as player position, obstacle/ball positions, player speed, etc. instead of relying on more complex information structures such as the video frame buffer or RAM state as found in some game benchmarks — this was a deliberate choice to better reflect how an RL agent might more likely be practically implemented to play these games. Finally, we experimented with the Food Collector game from Unity ML-agents (release 14) set of example training environments, where the agent observes information about the game-state in a local region around its position and is tasked with collecting objective mobile markers using a designated capture action, while avoiding mobile obstacles. This game was chosen specifically as its construction bears relevance to a wide variety of objective-capture type games. While other games from the ML-agents examples were considered, we opted against using any that employed self-play in training to avoid behavioral artefacts that could arise and felt that the other environments did not offer significantly more representative value, given the controls tasks and games already tested.

Algorithms Tested: For the continuous control tasks, we tested 4 algorithms: DDPG [18], TD3 [19], SAC [20] and PPO [16]. With the exception of PPO, we used codes provided by the OpenAI Spinning Up [26] code-base, with training hyper-parameters from the RL-zoo [27]. As shown by Engstrom et al. [32] and Ilyas et al. [33] PPO can be highly sensitive to implementation details, so we instead used the OpenAI Baselines [25] implementation of PPO to maintain parity with the original work. For the discrete action-space environments (Acrobot, Pong and Pixelcopter), DDPG and TD3 were not tested as they are designed specifically for continuous control. For Unity's Food Collector, we utilized the ML-agents implementations of SAC and PPO as adapting the Spinning Up codes to work with the API proved non-trivial.

Results: Our results are shown in Table I, with a visual contextualization of the relative performances of symmetrically and asymmetrically constructed actors shown in Fig. 4. Asymmetry has demonstrable utility in reducing actor sizes in every environment tested, though not necessarily with every algorithm in each environment. We would also note that, though it is possible for the automated search to find actors larger than their critics, such a case did not arise in testing. Network size reductions vary by algorithm and environment, with between 70–97% actor size reductions, and with the Q-learning based algorithms (DDPG, TD3 and SAC) sharing similar trends. Network sizes are computed per task to account for the different state and action dimensions (see Appendix B for full weight-counts for each task).

TABLE I

EXPLORING THE IMPACT OF ACTOR-CRITIC ASYMMETRY ON ALGORITHM PERFORMANCE (IMPROVEMENTS IN SIZE ARE IN **bold**)

Algorithm	Threshold	Baseline		Symmetric Actor		Asymmetric Actor			
		Size	Reward	Size ↓	Reward ↑	Actor Size ↓	Reduction ↑	Critic Size	Reward ↑
Gym Pendulum-v0									
DDPG	-160	$\lvert 400, 300\rvert$	-145.56 ± 10.64	$\lvert 16, 16\rvert$	-150.28 ± 9.08	$\lvert \mathbf{4}, \mathbf{4}\rvert$	**88.38%**	$\lvert 16, 16\rvert$	-158.97 ± 5.33
TD3	-160	$\lvert 400, 300\rvert$	-152.71 ± 9.47	$\lvert 16, 16\rvert$	-152.42 ± 6.65	$\lvert \mathbf{4}, \mathbf{4}\rvert$	**88.38%**	$\lvert 16, 16\rvert$	-167.57 ± 14.53
SAC	-160	$\lvert 256, 256\rvert$	-139.86 ± 8.29	$\lvert 16, 16\rvert$	-155.32 ± 11.25	$\lvert \mathbf{4}, \mathbf{4}\rvert$	**88.38%**	$\lvert 16, 16\rvert$	-153.86 ± 10.97
PPO	-200	$\lvert 64, 64\rvert$	-668.60 ± 551.85	$\lvert 128, 128\rvert$	-193.55 ± 40.14	$\lvert 128, 128\rvert$	0%	$\lvert 128, 128\rvert$	-193.55 ± 40.14
Gym Reacher-v2									
DDPG	-5.0	$\lvert 400, 300\rvert$	-4.26 ± 0.25	$\lvert 64, 64\rvert$	-4.75 ± 0.19	$\lvert \mathbf{8}, \mathbf{8}\rvert$	**96.33%**	$\lvert 64, 64\rvert$	-4.80 ± 0.44
TD3	-7.0	$\lvert 400, 300\rvert$	-6.52 ± 1.12	$\lvert 64, 64\rvert$	-6.91 ± 0.74	$\lvert \mathbf{32}, \mathbf{32}\rvert$	**70.23%**	$\lvert 64, 64\rvert$	-6.68 ± 1.21
SAC	-6.5	$\lvert 256, 256\rvert$	-5.96 ± 0.47	$\lvert 128, 128\rvert$	-6.05 ± 0.91	$\lvert \mathbf{16}, \mathbf{16}\rvert$	**97.28%**	$\lvert 128, 128\rvert$	-6.02 ± 1.07
PPO	-5.5	$\lvert 64, 64\rvert$	-4.37 ± 1.74	$\lvert 64, 64\rvert$	-4.37 ± 1.74	$\lvert \mathbf{16}, \mathbf{16}\rvert$	**90.15%**	$\lvert 64, 64\rvert$	-5.49 ± 1.00
Gym HalfCheetah-v2									
DDPG	7000	$\lvert 400, 300\rvert$	7026.01 ± 202.78	$\lvert 64, 64\rvert$	7450.01 ± 950.15	$\lvert \mathbf{32}, \mathbf{32}\rvert$	**68.01%**	$\lvert 64, 64\rvert$	8273.76 ± 437.66
TD3	8000	$\lvert 400, 300\rvert$	8861.92 ± 870.02	$\lvert 64, 64\rvert$	8315.13 ± 262.78	$\lvert \mathbf{32}, \mathbf{32}\rvert$	**68.01%**	$\lvert 64, 64\rvert$	8145.84 ± 262.55
SAC	10000	$\lvert 256, 256\rvert$	11554.76 ± 779.91	$\lvert 64, 64\rvert$	10180 ± 759.10	$\lvert \mathbf{32}, \mathbf{32}\rvert$	**68.01%**	$\lvert 64, 64\rvert$	9619 ± 158.40
PPO	3000	$\lvert 64, 64\rvert$	3395 ± 1156.30	$\lvert 64, 64\rvert$	3395 ± 1156.30	$\lvert \mathbf{32}, \mathbf{32}\rvert$	**68.01%**	$\lvert 64, 64\rvert$	3089 ± 919.25
Gym Ant-v2									
DDPG[a]	–	$\lvert 400, 300\rvert$	225.23 ± 362.88	–	–	–	–	–	
TD3	3000	$\lvert 400, 300\rvert$	3087.86 ± 888.75	$\lvert 256, 256\rvert$	3944.78 ± 745.48	$\lvert \mathbf{32}, \mathbf{32}\rvert$	**94.92%**	$\lvert 256, 256\rvert$	3553.52 ± 396.30
SAC	3000	$\lvert 256, 256\rvert$	3366.07 ± 1522.45	$\lvert 64, 64\rvert$	3108.63 ± 519.59	$\lvert 64, 64\rvert$	0%	$\lvert 64, 64\rvert$	3108.63 ± 519.59
PPO	3000	$\lvert 64, 64\rvert$	3734.58 ± 988.29	$\lvert 8, 8\rvert$	3723.57 ± 760.94	$\lvert 8, 8\rvert$	0%	$\lvert 8, 8\rvert$	3723.57 ± 760.94
Gym Acrobot-v1									
SAC[b]	-100	$\lvert 256, 256\rvert$	-76.5 ± 4.30	$\lvert 32, 32\rvert$	-86.5 ± 7.54	$\lvert \mathbf{16}, \mathbf{16}\rvert$	**68.46%**	$\lvert 32, 32\rvert$	-90.5 ± 8.12
PPO[b]	-100	$\lvert 64, 64\rvert$	-74.7 ± 0.30	$\lvert 8, 8\rvert$	-70.75 ± 0.33	$\lvert \mathbf{1}, \mathbf{1}\rvert$	**90.33%**	$\lvert 8, 8\rvert$	-71.75 ± 0.50
PLE Pong									
SAC[b]	0.90	$\lvert 256, 256\rvert$	0.93 ± 0.05	$\lvert 64, 64\rvert$	0.90 ± 0.06	$\lvert \mathbf{16}, \mathbf{16}\rvert$	**90.73%**	$\lvert 64, 64\rvert$	0.89 ± 0.03
PPO[b]	0.80	$\lvert 64, 64\rvert$	0.85 ± 0.13	$\lvert 64, 64\rvert$	0.85 ± 0.13	$\lvert \mathbf{4}, \mathbf{4}\rvert$	**98.62%**	$\lvert 64, 64\rvert$	0.83 ± 0.09
PLE Pixelcopter									
SAC[b]	35	$\lvert 256, 256\rvert$	36.83 ± 9.65	$\lvert 64, 64\rvert$	33.87 ± 5.80	$\lvert \mathbf{8}, \mathbf{8}\rvert$	**96.79%**	$\lvert 64, 64\rvert$	35.98 ± 4.93
PPO[b]	20	$\lvert 64, 64\rvert$	22.24 ± 9.86	$\lvert 64, 64\rvert$	22.24 ± 9.86	$\lvert \mathbf{32}, \mathbf{32}\rvert$	**71.30%**	$\lvert 64, 64\rvert$	19.91 ± 10.03
Unity ML-agents Food Collector									
SAC[c,d]	60	$\lvert 128, 128\rvert$[e]	61.77 ± 6.16	$\lvert 32, 32\rvert$	65.22 ± 8.26	$\lvert \mathbf{8}, \mathbf{8}\rvert$	**81.48%**	$\lvert 32, 32\rvert$	60.31 ± 5.49
PPO[c]	50	$\lvert 128, 128\rvert$[e]	50.25 ± 2.01	$\lvert 128, 128\rvert$	50.25 ± 2.01	$\lvert \mathbf{32}, \mathbf{32}\rvert$	**87.82%**	$\lvert 128, 128\rvert$	48.05 ± 3.12

[a]Due to DDPG being incapable of reasonably solving the Ant task in our experiments, it is omitted for the Ant-v2 environment.
[b]OpenAI Spinning up implementation of algorithm is adjusted to support discrete action spaces.
[c]Unity ML-agents implementation of algorithm is used to preserve compatibility with recommended baseline configuration.
[d]Due to the instability of ML-agent's implementation of SAC's training on Food Collector, we report the average maximum reward achieved during training.
[e]We use as a baseline the sizes recommended by the ML-agents baseline configuration parameters.

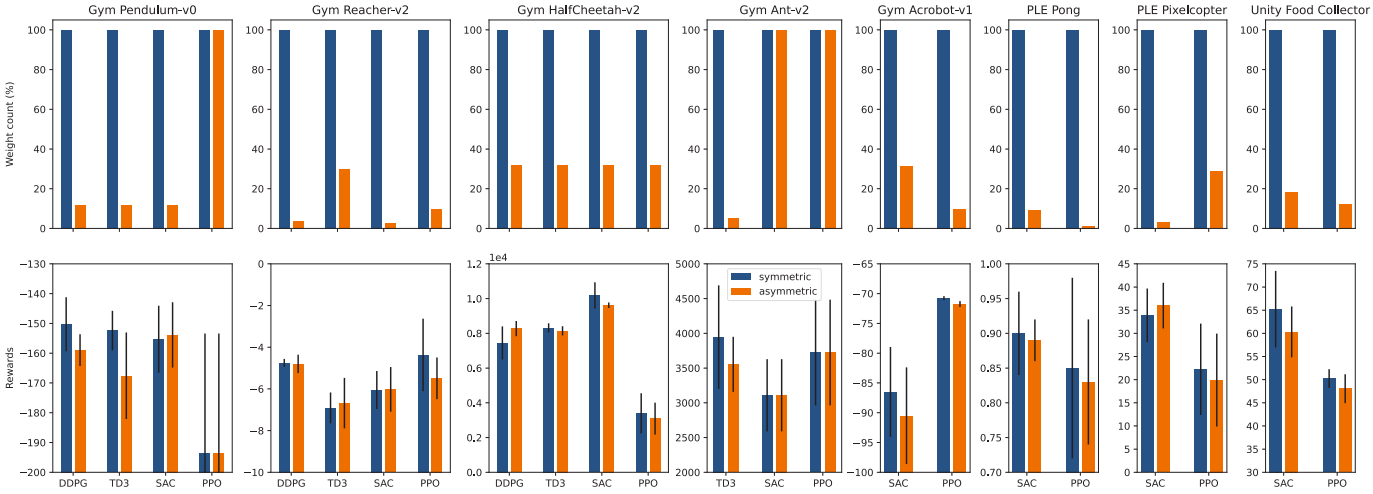

Fig. 4. Visual representation of the relative network size reductions and rewards presented in Table I. Network sizes are computed per task (see Appendix B). Results evidence that it is often possible to train significantly smaller actors with larger critics without compromising performance by breaking implicit assumptions of network architectural symmetry between actors and critics.

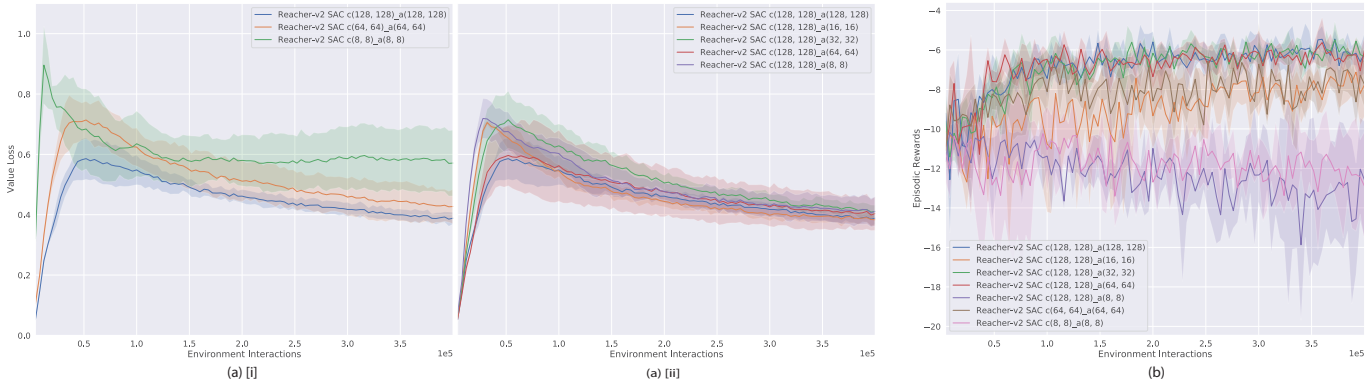

Fig. 5. Training progress for SAC on Reacher-v2: (a) Q-Vlaue Loss during training and (b) the episodic returns. (a)[i] Shows the value loss during training for different critic sizes while (a)[ii] shows the same for different actors trained with the same critic size. Changing the critic's size impacts value-loss more than changing the actor's, suggesting that the critic is more sensitive to changes in modeling capacity.

PPO appears to be more unpredictable in performance, particularly when considering the disparities in network size to performance over Pendulum-v0, Ant-v2 and Acrobot-v1. Despite poor performance on the simple inverted pendulum problem with smaller networks where other algorithms succeed, PPO solves the more complicated multi-pedal locomotion problem in Ant-v2 with significantly smaller actor and critic networks compared to the other algorithms tested and is similarly able to solve the Acrobot-v1 problem with smaller networks than SAC. It does however allow actor size reduction through asymmetry in most of the tested environments.

Experimental observations on these more complex environments also provide further empirical validation of our hypothesis that modeling capacity in the critic is often the limiting factor in achieving successful training. This is best illustrated by data on the training of SAC on the Reacher-v2 task and is shown in Fig. 5 where, much like in Fig. 1, there are relatively clear demarcations in where the Q-value losses and episodic rewards saturate for different network sizes. While changing the size of the critic during the symmetric binary search has a noticeable impact on the Q-value loss, changing the actor size during asymmetric search does not impact the Q-value loss significantly, as shown when comparing Figures 5(a)[i] and [ii]. A larger critic also allows for a much smaller actor to be successfully trained (in this case a critic with hidden layers of size $|128, 128|$ allows for an actor of size $|16, 16|$). We can therefore draw the conclusion that the dominant limiting factor in learning, for this algorithm, on this environment, is indeed the modeling capacity of the critic. Similar trends were observed for other algorithms in other training tasks too.

Our project website: http://ai.bu.edu/littleActor/ provides access to view full training data for the experiments summarized in Table I and Figs. 4 and 5, in addition to training code used.

V. Reflection and Recommendations

The main goal of this work was to draw attention the the practical benefits of separately considering actor and critic architectures in RL-based AI, and experiments over a wide array of tasks show that large reductions in actor-network sizes are often possible. Furthermore, these reductions come without incurring significant degradation in performance (capped here to 10%) — and crucially, the tasks are still solved. We do recognize that, on the scale of typical 'large' neural network models, the architectures considered in this paper could all be considered relatively small. However, with applications such as games, where running the game AI is just one of many compute tasks that need to be managed on machines with varying resources budgets and where it may be necessary to deploy multiple independent AIs simultaneously, we believe that network size can be an important optimization consideration. Reducing actor network sizes was also observed to have some benefit in reducing training time — presenting with up to 50% reduction in training time — though these were inconsistent due to the impact of other tasks running concurrently on the training machines.

While we made efforts to thoroughly explore one aspect of network architectures, i.e. the network size, as the network size is typically the largest contributor to compute cost, this should not be regarded as an exhaustive study of the impact of network architectures. There are still more aspects such as depth, output activation, network initialization, etc., and as shown by Henderson et al. [21] and Andrychowicz et al. [23], those can strongly contribute to policy performance. The key take-aways are that these aspects of network design bear serious consideration when developing AI for deployment, and while our analysis has been focused on network size, we suspect that it would be worth considering other architectural design choices separately for actors and critics as well. We would recommend that practitioners pay specific attention to how policy rewards and value losses evolve as a function of these architectural choices. If starting from symmetric architectures, we would recommend identifying the point of diminishing returns when increasing the critic parameters and building from there to reduce actor complexity.

VI. Related Work

Prior work by Islam et al. [11] and Henderson et al. [21] touch on the impact of network size on the performance of RL algorithms but their analyses are more limited to a smaller set of environments and architectures. Both works consider

networks with 3 different possible hidden layer configurations: $|400, 300|$, $|64, 64|$ and $|100, 50, 25|$, common sizes used in prior literature. There is anecdotal evidence in these works to suggest that larger actors are not necessarily more performant and that larger critics can usually learn better, but the data presented does not lend itself to clear conclusions.

Asymmetry in actors and critics appears to have first been explicitly addressed by Pinto et al. [34] in controlling robotic manipulation. The authors explore a form of input asymmetry between actors and critics, where actor networks are offered reduced state information from an RGBD sensor, while the critic is offered full-state information from a simulator during training. They demonstrate that actors are able to actually perform better under this reduced information paradigm, likely due to the reduced representational complexity. An interpretation of the implications of their work is that it also hints at the idea of the onerous of learning more robust and complex representations may perhaps lie more with the critic side of the network.

Henderson et al. [21] provide a cursory analysis on the impact of disentangling the actor and critic architectures and consider how different algorithms behave in different training environments as a result of changing network structures. They note the sensitivity of algorithms to these details but a limitation of their analysis is that it sacrifices depth for breadth as this is just one of many hyper-parameters they investigate. Their analysis appears to primarily seek to identify *if* there is a correlation between performance and actor and critic architectures when considered independently. However only two environments and a limited set of network configurations are tested, preventing clear conclusions being drawn on *how* actor and critic architectures relate to each other. By considering a wider range of network sizes and environments, we are able to formulate and test a clearer hypothesis on the relationship between network sizes, both for the actors and critics, and their impact on performance.

Recent concurrent work by Andrychowicz et al. [23] discusses a large scale study on the impact of different hyperparameters on RL performance on 5 OpenAI Gym [30] continuous controls benchmark tasks. Network architecture is among the many parameters considered and, like us, the authors also extend their study past the typically used architectures to consider a wider range of network sizes, the impact of varying actor and critic sizes separately and also of varying the depth of the networks. Like us, the authors make note of the fact that it is possible to use smaller actors and that RL agents more often benefit from having larger critics. They also identified that increasing network depth was not necessarily beneficial, an observation also noted by Sinha et al. [24]. The authors' focus, being on performance maximization, as opposed to network complexity reduction however, results in them ultimately prioritizing different aspects of the network architecture — such as policy initialization and output activation — in addition to a wider array of non-architecture-related parameters. While they do contribute useful and important observations and recommendations on tuning those parameters for more performant policies, the do not make stronger statements on how the sizes of actors and critics may interact with each other, but mainly that they ought to be paid attention to. Their results do however independently corroborate ours (and ours, theirs).

Reducing the parameter count of neural networks has also received interest in the broader machine learning community. Techniques like pruning [35], knowledge distillation [36], and weight sharing [37] have been shown to accelerate inference and reduce memory requirements, enabling deployment on resource-constrained mobile hardware. In RL, policy distillation [38] has been demonstrated as an effective tool for reducing network sizes using a teacher-student method where a larger pre-trained teacher network guides a smaller student network which learns to match the teacher's performance. Smaller students learning equivalently functional policies suggests an excess in modeling capacity of more typical architectures for policy networks.

VII. Conclusion

We hypothesized that this capacity for actor size reduction comes from the burden of modeling and understanding environment dynamics falling largely to the critic, rather than the actor. Provided that the critic has sufficient capacity to learn a decent estimation of the value function, we predicted that it would often be possible to train actors that are significantly smaller than the critic. The results presented in this case study substantiate this hypothesis. We demonstrated that, by relaxing the implicit assumption of symmetry in the architectures of actors and critics in actor-critic RL, it is possible to train significantly smaller networks for tasks without a significant degradation in policy performance. We demonstrate as much as a 97% reduction in the number of weights needed to represent viable actor policies, with an average actor size reduction of 77% in actor network sizes for the tested tasks. We believe that the implications of these results can be highly significant to practical applications of RL, particularly in resource-constrained systems, and that it is also worth being generally aware of as part of the broader effort to understand how network sizes can impact actor-critic RL performance.

References

[1] S. Ariyurek, A. Betin-Can, and E. Surer, "Automated video game testing using synthetic and humanlike agents," *IEEE Transactions on Games*, vol. 13, no. 1, pp. 50–67, 2021.

[2] Y. Zhao, I. Borovikov, F. de Mesentier Silva, A. Beirami, J. Rupert, C. Somers, J. Harder, J. Kolen, J. Pinto, R. Pourabolghasem, J. Pestrak, H. Chaput, M. Sardari, L. Lin, S. Narravula, N. Aghdaie, and K. Zaman, "Winning is not everything: Enhancing game development with intelligent agents," *IEEE Transactions on Games*, vol. 12, no. 2, pp. 199–212, 2020.

[3] Y. Zhao, I. Borovikov, J. Rupert, C. Somers, and A. Beirami, "On multi-agent learning in team sports games," *arXiv preprint arXiv:1906.10124*, 2019.

[4] C. Berner *et al.*, "Dota 2 with large scale deep reinforcement learning," *CoRR*, vol. abs/1912.06680, 2019.

[5] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[6] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, pp. 279–292, 1992.
[7] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.
[8] J. Pineau, P. Vincent-Lamarre, K. Sinha, V. Larivière, A. Beygelzimer, F. d'Alché Buc, E. Fox, and H. Larochelle, "Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program)," 2020.
[9] C. Zhang, O. Vinyals, R. Munos, and S. Bengio, "A study on overfitting in deep reinforcement learning," *CoRR*, vol. abs/1804.06893, 2018.
[10] A. X. Zhang, N. Ballas, and J. Pineau, "A dissection of overfitting and generalization in continuous reinforcement learning," *CoRR*, vol. abs/1806.07937, 2018.
[11] R. Islam, P. Henderson, M. Gomrokchi, and D. Precup, "Reproducibility of benchmarked deep reinforcement learning tasks for continuous control," *arXiv preprint arXiv:1708.04133*, 2017.
[12] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 1329–1338.
[13] A. Juliani, V.-P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar *et al.*, "Unity: A general platform for intelligent agents," *arXiv preprint arXiv:1809.02627*, 2018.
[14] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.
[15] H. Shen, K. Zhang, M. Hong, and T. Chen, "Asynchronous advantage actor critic: Non-asymptotic analysis and linear speedup," 2020.
[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.
[17] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, F. Bach and D. Blei, Eds., vol. 37. Lille, France: PMLR, 07–09 Jul 2015, pp. 1889–1897.
[18] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *International Conference on Learning Representations*, 2016.
[19] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning*, 2018, pp. 1587–1596.
[20] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *International Conference on Machine Learning (ICML)*, 2018.
[21] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.
[22] T. Zahavy, Z. Xu, V. Veeriah, M. Hessel, J. Oh, H. P. van Hasselt, D. Silver, and S. Singh, "A self-tuning actor-critic algorithm," *Advances in Neural Information Processing Systems*, vol. 33, 2020.
[23] M. Andrychowicz, A. Raichuk, P. Stańczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, S. Gelly, and O. Bachem, "What matters for on-policy deep actor-critic methods? a large-scale study," in *International Conference on Learning Representations*, 2021.
[24] S. Sinha, H. Bharadhwaj, A. Srinivas, and A. Garg, "D2rl: Deep dense architectures in reinforcement learning," *arXiv preprint arXiv:2010.09163*, 2020.
[25] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," https://github.com/openai/baselines, 2017.
[26] J. Achiam, "Spinning Up in Deep Reinforcement Learning," 2018.
[27] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," 2018.
[28] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," https://github.com/DLR-RM/stable-baselines3, 2019.
[29] S. Guadarrama, A. Korattikara, O. Ramirez, P. Castro, E. Holly, S. Fishman, K. Wang, E. Gonina, N. Wu, E. Kokiopoulou, L. Sbaiz, J. Smith, G. Bartók, J. Berent, C. Harris, V. Vanhoucke, and E. Brevdo, "TF-Agents: A library for reinforcement learning in tensorflow," https://github.com/tensorflow/agents, 2018, [Online; accessed 25-June-2019]. [Online]. Available: https://github.com/tensorflow/agents
[30] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016.
[31] N. Tasfi, "Pygame learning environment," https://github.com/ntasfi/PyGame-Learning-Environment, 2016.
[32] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "Implementation matters in deep rl: A case study on ppo and trpo," in *International Conference on Learning Representations*, 2020.
[33] A. Ilyas, L. Engstrom, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry, "A closer look at deep policy gradients," in *International Conference on Learning Representations*, 2020.
[34] L. Pinto, M. Andrychowicz, P. Welinder, W. Zaremba, and P. Abbeel, "Asymmetric actor critic for image-based robot learning," *arXiv preprint arXiv:1710.06542*, 2017.
[35] S. Han, H. Mao, and W. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *arXiv: Computer Vision and Pattern Recognition*, 2016.
[36] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *ArXiv*, vol. abs/1503.02531, 2015.
[37] K. Ullrich, E. Meeds, and M. Welling, "Soft weight-sharing for neural network compression," *ArXiv*, vol. abs/1702.04008, 2017.
[38] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 2015.
[39] "Python noise library version 1.2.3," accessed: 2021-02-04. [Online]. Available: https://github.com/caseman/noise/tree/bb32991ab97e90882d0e46e578060717c5b90dc5

APPENDIX

A. Toy Problem Setup Details

Additional setup details for the toy-problem in Section III:

- **Goal generation**: Goals g_t are procedurally generated by functions mapping an input 'time' signal to the range $[-1, 1]$ using a simplex [39] function.
- **System Dynamics**: The response to action, a_t is computed as $s_{t+1} = \text{clip}(s_t + a_t, -1, 1)$.

B. Per-task Policy Network Sizes

Weight-counts for each environment are computed separately to account for their different state and action dimensions.

TABLE A.2
NUMBER OF ACTOR-NETWORK WEIGHT PARAMETERS PER ENVIRONMENT FOR EACH OF THE ENVIRONMENTS CONSIDERED

Hidden Size	Pen	Rea	HCh	Ant	Acr	Png	Pxl	FdC
$\|400, 300\|$	122201	125702	129306	167508	124003	124403	124102	143104
$\|256, 256\|$	67073	69378	71942	96520	68355	68611	68354	80644
$\|128, 128\|$	17153	18306	19590	31880	17795	17923	17794	23940
$\|64, 64\|$	4481	5058	5702	11848	4803	4867	4802	7876
$\|32, 32\|$	1217	1506	1830	4904	1379	1411	1378	2916
$\|16, 16\|$	353	498	662	2200	435	451	434	1204
$\|8, 8\|$	113	186	270	1040	155	163	154	540
$\|4, 4\|$	41	78	122	508	63	67	62	256
$\|1, 1\|$	8	18	32	130	15	-	-	-

Pen: Gym Pendulum-v0
Rea: Gym Reacher-v2
HCh: Gym HalfCheetah-v2
Ant: Gym Ant-v2
Png: PLE Pong
Pxl: PLE Pixelcopter
FdC: Unity ML-agents Food Collector