6th ICML Workshop on Automated Machine Learning (2019)

Evolving Rewards to Automate Reinforcement Learning

Aleksandra Faust FAUST@GOOGLE.COM
Anthony Francis CENTAUR@GOOGLE.COM
Dar Mehta DARM@GOOGLE.COM
Robotics at Google, Mountain View, CA, 94043, USA

Abstract

Many continuous control tasks have easily formulated objectives, yet using them directly as a reward in reinforcement learning (RL) leads to suboptimal policies. Therefore, many classical control tasks guide RL training using complex rewards, which require tedious hand-tuning. We automate the reward search with AutoRL, an evolutionary layer over standard RL that treats reward tuning as hyperparameter optimization and trains a population of RL agents to find a reward that maximizes the task objective. AutoRL, evaluated on four Mujoco continuous control tasks over two RL algorithms, shows improvements over baselines, with the the biggest uplift for more complex tasks. The video can be found at: `https://youtu.be/svdaOFfQyC8`.

1. Introduction

Despite solving a number of challenging problems (Kalashnikov et al., 2018; Chen et al., 2015; Levine et al., 2016), training RL agents remains difficult and tedious. One culprit is reward design, which currently requires many iterations of manual tuning. Reward is a scalar that communicates the task objective and desirable behaviors. Ideally, it should be an indicator of task completion (Sutton et al., 1992) or a simple metric over resulting trajectories. Consider a Humanoid task (Tassa et al., 2012), where objectives might be for the agent to travel as far or as fast as possible. Learning directly on these metrics is problematic, especially for high dimensional continuous control problems, for two reasons. First, RL requires the agent to explore until it stumbles onto the goal or improves the task metric, which can take a prohibitively long time (Andrychowicz et al., 2017). Second, there are many ways to accomplish a task, and some are less than ideal. For example, the Humanoid might run while flailing its arms, or roll on the ground. Practitioners circumvent those cases through *reward shaping* (Wiewiora, 2010). Beginning with simple rewards, such as distance travelled, they train a policy and observe the training outcome. Next, practitioners either add terms to the reward that provide informative feedback about progress, such as energy spent or torso verticality, or tune the weights between the reward terms. Then they retrain the policy, observe, and repeat until training is tractable and the agent is well-behaved. In the Humanoid task, the agent is expected to maximize the speed and time alive while minimizing the control and impact cost. The standard reward collapses this multi-objective into one scalar with carefully selected weights. This human-intensive process raises questions: a) Can we automate the tuning and learn a proxy reward that both promotes the learning and meets the task objective? and b) Given an already hand-tuned reward, is there a better parameterization that accomplishes the same multi-objective?

Our prior work (Chiang et al., 2019) introduces AutoRL over Deep Deterministic Policy Gradients (DDPG, Lillicrap et al. (2015)) in the context of robot navigation to learn two end-to-end (sensor to controls) tasks: point-to-point and path-following. Using large scale hyper-parameter optimization, Chiang et al. (2019) first find the reward parameters that maximize the goal reached sparse objective, and then find the neural network architecture that maximizes the learned proxy reward. In that setting, AutoRL improves both training stability and policy quality for sparse objectives for both tasks. But it remains an open question whether AutoRL, particularly the search for proxy rewards, is useful for other RL algorithms, tasks, and objectives.

This paper applies AutoRL's evolutionary reward search to four continuous control benchmarks from OpenAI Gym (Brockman et al., 2016), including Ant, Walker2D, HumanoidStandup, and Humanoid, over two RL algorithms: off-policy Soft Actor-Critic (SAC, Haarnoja et al. (2018b)) and on-policy Proximal Policy Optimization (PPO, Schulman et al. (2017)). We optimize parameterized versions of the standard environment rewards (proxy rewards) over two different objectives: *metric-based single-task objectives* including distance travelled and height reached, and the multi-objective *standard returns* typically used in these environments. The results yield three findings. First, evolving rewards trains better policies than hand-tuned baselines, and on complex problems outperforms hyperparameter-tuned baselines, showing a 489% gain over hyperparameter tuning on a single-task objective for SAC on the Humanoid task. Second, the optimization over simpler single-task objectives produces comparable results to the carefully hand-tuned standard returns, reducing the need for manual tuning of multi-objective tasks. Lastly, under the same training budget the reward tuning produces higher quality policies faster than tuning the learning hyperparameters.

2. Related Work

AutoML and RL: AutoML automates neural network architecture searches for supervised learning with RL (Zoph and Le, 2017; Zoph et al., 2018a; Cai et al., 2018; Zoph et al., 2018b) and Evolutionary Algorithms (EA) (Real et al., 2017, 2018b; Liu et al., 2017), with the latter showing an edge (Real et al., 2018a). While RL is part of the AutoML toolset, tuning RL itself has been very limited. For example, EA mutated the actor network weights (Khadka and Tumer, 2018).

Reward design: Aside from reward shaping (Zoph et al., 2018a), reward design methods include curriculum learning (Florensa et al., 2017; Ivanovic et al., 2018; Gur et al., 2019), bootstrapping (Silver et al., 2018), and Inverse Reinforcement Learning (IRL) (Ng and Russell, 2000). AutoRL keeps task difficulty constant, trains policies from scratch, and requires no demonstrations, but it could be used in addition to curriculum and bootstrapping.

Large-scale hyperparameter optimization has become a standard technique for improving deep RL performance by tuning learning hyperparameters (Shah et al., 2018; Gur et al., 2019). AutoRL uses the same toolset but for reward and network tuning. (Chiang et al., 2019) uses sparse objectives, such as reaching a goal for point-to-point navigation, or waypoints achieved for a path-following task. This is effective because agents can be given a variety of tasks, many of which are easy to achieve on a true sparse objective (such as nearby goals in uncluttered environments). Here, we focus on metric-based objectives.

3. AutoRL

Preliminaries: Consider a Partially-Observable Markov Decision Process (POMDP) with continuous actions, $\mathcal{M}(S, O, A, \boldsymbol{D}, R, \gamma)$. S are states, $O \subset \mathbb{R}^{d_o}$ are observations, $A \subset \mathbb{R}^{d_a}$ are actions, and the system's unknown dynamics are modeled as a transition distribution $\boldsymbol{D} : S \times A \times S \to [0, 1]$. The reward, $R : S \times A \to \mathbb{R}$ is an immediate feedback to the RL agent, while $0 < \gamma \leq 1$ is a discount factor. The goal of RL is to find a policy, $\hat{\boldsymbol{\pi}}_R : S \times A \to [0, 1]$ that maximizes the expected cumulative discounted reward R of trajectories $\mathcal{T}$ drawn from a set of initial conditions, $N \subset O$, guided by the policy π w.r.t. the system dynamics $\boldsymbol{D}$:

$$\hat{\boldsymbol{\pi}}_R = \arg\max_{\boldsymbol{\pi}} \mathbb{E}_{\mathcal{T} \sim (\boldsymbol{D}, N, \boldsymbol{\pi})} [\sum_{i=0}^{\|\mathcal{T}\|} \gamma^i R(\boldsymbol{s_i}, \boldsymbol{a_i})]. \tag{1}$$

POMDPs model the world, observations, actions, and reward; all affect learning and performance, and are usually chosen through trial and error. AutoRL addresses the gap by applying the AutoML toolset to RL's pain points of POMDP modelling.

Problem formulation: An agent's success at a task can often be evaluated over a trajectory $\mathcal{T}$ with a metric $G(\mathcal{T})$. For example, success for the Humanoid Standup task is standing as tall as possible, while for Ant or Humanoid it is traveling as fast as possible. We can evaluate $G(\mathcal{T})$ over trajectories $\mathcal{T}$ drawn from initial conditions N and controlled by a policy $\boldsymbol{\pi}$ to gauge the policy's quality $J(\boldsymbol{\pi})$ w.r.t. the fitness metric:

$$J(\boldsymbol{\pi}) = \mathbb{E}_{\mathcal{T} \sim (\boldsymbol{D}, N, \boldsymbol{\pi})} [G(\mathcal{T})]. \tag{2}$$

While the reward R measures immediate feedback to the agent, the objective metric G reflects human-interpretable trajectory quality and induces an ordering over policies. Humans want high-quality trajectories, which are sparse, but RL agents learn best from dense feedback, so we use G to help pick R as follows.

Consider an augmented POMDP, $\tilde{\mathcal{M}}(S, O, A, \boldsymbol{D}, R(\boldsymbol{\theta}), \gamma, G)$, where $\boldsymbol{\theta} \in \Theta$ is the parameter of a proxy reward function $R(\boldsymbol{\theta})$. The goal of AutoRL is to solve $\tilde{\mathcal{M}}$ by finding a policy $\tilde{\boldsymbol{\pi}}$ that maximizes the fitness metric J defined in (2) given a population of agents $\hat{\boldsymbol{\pi}}(R(\boldsymbol{\theta}))$ with parameterization $\boldsymbol{\theta}$ drawn from Θ:

$$\tilde{\boldsymbol{\pi}} = \text{argmax}_{\{\hat{\boldsymbol{\pi}}_{R(\boldsymbol{\theta})} | \theta \sim \Theta\}} J(\hat{\boldsymbol{\pi}}_{R(\boldsymbol{\theta})}), \text{ where } \hat{\boldsymbol{\pi}}_{R(\boldsymbol{\theta})} \text{ is given in (1).} \tag{3}$$

Proxy rewards, $R(\boldsymbol{\theta})$, may contain new features that guide learning, or may reweight the objective $G(\mathcal{T})$ to more closely match the gradient of the value function.

Algorithm: Denote a policy $\boldsymbol{\pi}(\boldsymbol{\theta})$ learned with an externally-provided RL algorithm that optimizes the cumulative return (1) for a fixed $\boldsymbol{\theta}$ as:

$$\boldsymbol{\pi}(\boldsymbol{\theta}) = \text{RL}(R(\boldsymbol{\theta})) \tag{4}$$

and let n_g be the maximum population size, with n_{mc} as the number of parallel trials.

We train a population of n_{mc} parallel RL agents according to (4), each initialized with a different reward parameterization $\boldsymbol{\theta_i}, i \in \{1, \cdots, n_g\}$. The parameterization for the first n_{mc} agents are selected randomly. When training of the i^{th} RL agent completes, Monte Carlo rollouts estimate the fitness metric (2) of the resulting policy $\boldsymbol{\pi}(\boldsymbol{\theta_i})$ to obtain, j_i.

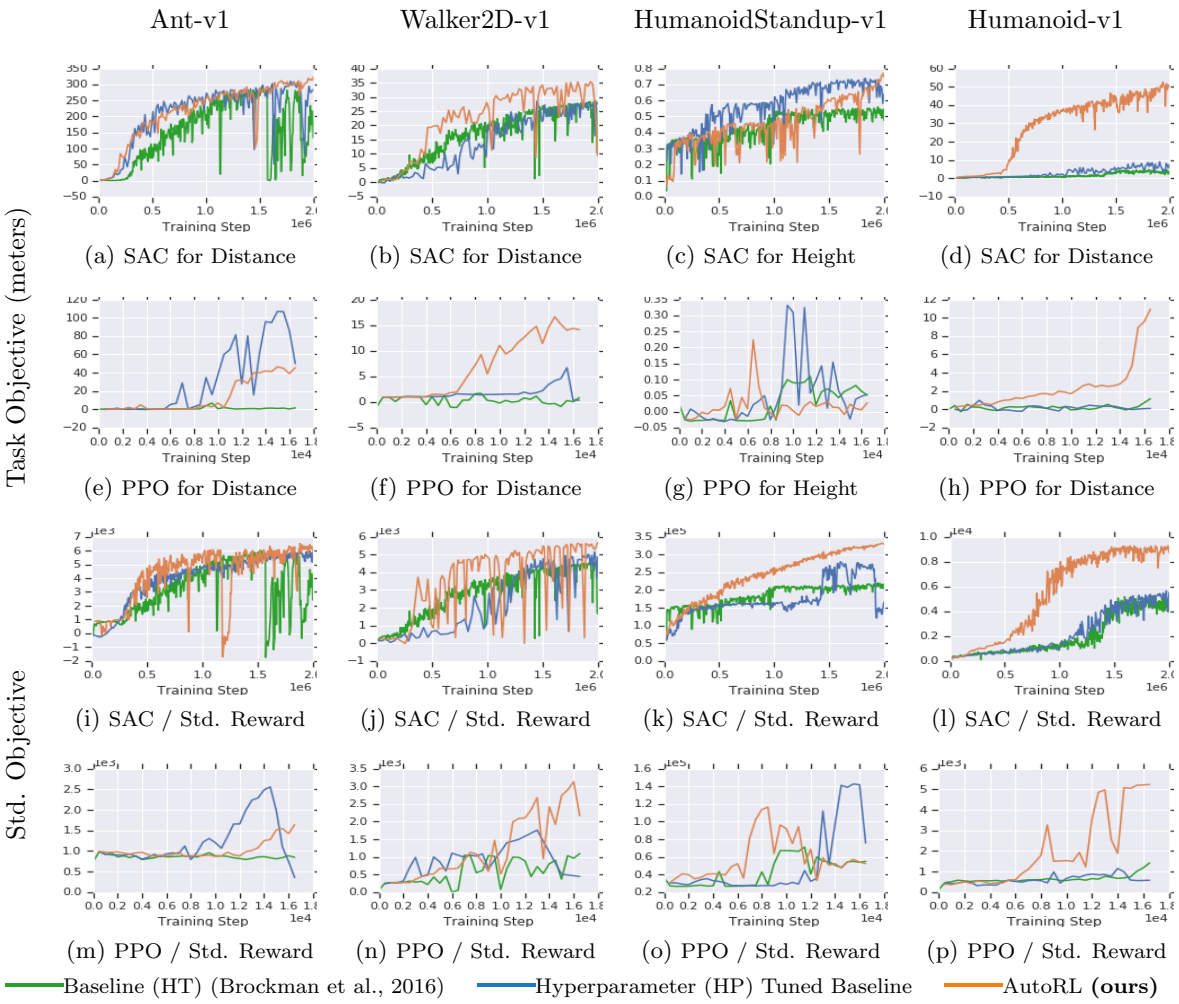


Figure 1: Task objective results a-d) using SAC and e-h) using PPO and standard return objective i-l) using SAC and m-p) using PPO for Ant, Walker2d, Humanoid Standup, and Humanoid.

This estimate and reward parameterization, $(\boldsymbol{\theta_i}, j_i)$, are added to the population experience set Θ, while the policy and estimate, $(\boldsymbol{\pi(\theta_i)}, j_i)$, are added to the policy evaluation set, Π. As the evaluation set Π is updated, AutoRL continually updates the current best policy $\tilde{\boldsymbol{\pi}}$ according to (2). Next, AutoRL selects the parameterization for the next trial from the population experience, Θ, according to Gaussian Process Bandits (Srinivas et al., 2012), and starts training a new RL agent. Finally, training stops after n_g trials.

AutoRL scales linearly with the population size n_g. Concurrent trials n_{mc} must be smaller than n_g, in order to have enough completed experience to select the next set of parameters. Overall, AutoRL requires $O(n_{mc})$ processors, and runs $\frac{n_g}{n_{mc}}$ times longer than vanilla RL.

4. Results

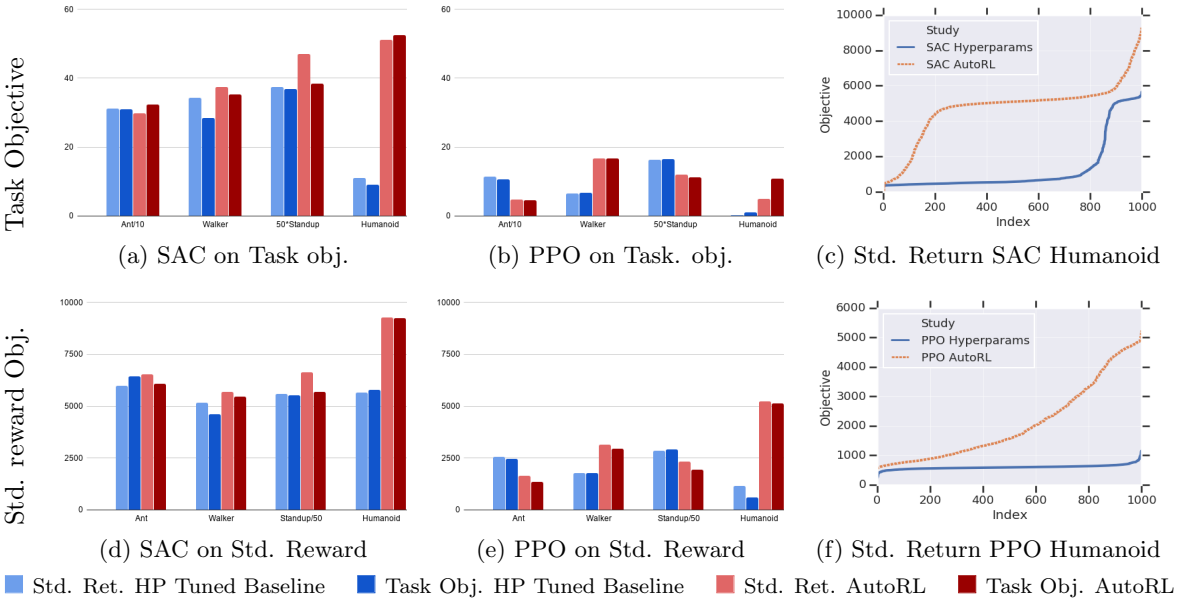
We implement AutoRL hyperparameter optimization with Vizier (Golovin et al., 2017) over the PPO and SAC and apply it to four widely-used MuJoCo (Todorov et al., 2012) continuous control environments in order of increasing complexity: Ant, Walker, Humanoid Standup, and Humanoid (Table 1; see Appendix A).

To assess AutoRL's ability to reduce reward engineering while maintaining quality on existing metrics, we contrast two objectives: task objectives and standard returns. *Task objectives* measure task achievement for continuous control: distance traveled for Ant, Walker, and Humanoid, and height achieved for Standup. *Standard returns* are the metrics by which tasks are normally evaluated. For both objectives, the parameterized reward $\boldsymbol{\theta}$ is a re-weighted standard reward (see Appendix B). We compare AutoRL with two baselines, hand-tuned and hyperparameter-tuned. Hand-tuned (HT) uses default learning parameters for each algorithm. Hyperparameter-tuned (HP) uses Vizier to optimize learning hyperparameters such as learning rate and discount factor. In all cases, AutoRL uses HT's default hyperparameters. We train up to $n_g = 1000$ agents parallelized across $n_{mc} = 100$ workers. SAC trains for 2 million steps, while PPO's training episodes depends on the environment (Table 1; see Appendix A). Policy quality (fitness metric (2)) w.r.t. the objective is evaluated over 50 trajectories.

Task Objective Evaluation: AutoRL outperforms the HP tuned baseline (blue) for all tasks trained with SAC (Figures 1a-d) and on Walker and Humanoid for PPO (Figures 1e-h). Both outperform the hand-tuned baseline, whose parameters AutoRL uses. Note that AutoRL uses non-tuned learning hyperparameters. AutoRL's benefit over HP tuning is consistent - though relatively small for simpler tasks - but is very noticeable on the most complex task, Humanoid, with 489% improvement over HP tuning. AutoRL shows 64% improvement on Humanoid. It is interesting to note that in some cases AutoRL converges more slowly, and does not reach peak performance until very late in the training process (Figures 1a, 1c, and 1h). We suspect this is because the tasks simply require more training iterations to converge, and the baselines end up getting stuck in a local minima, while AutoRL manages to escape it.

Std. Reward Evaluation: AutoRL outperforms HP tuning on all SAC tasks (Figure 1i-l) and on Walker and Humanoid for PPO (Figure 1m-p). Both beat the hand-tuned baseline.

Single-Task vs Multi-Objective: If our goal is finding objectives that are easier to provide than hand-tuning the multiple objectives of a standard reward, then we want to know how well AutoRL optimizes simple task objectives. On Humanoid, the task objective agents travel the farthest for both SAC and PPO (Figures 2a and 2b dark red vs. light red), while on other tasks optimizing over the task objective is comparable to optimizing over the standard reward. Task objectives and standard returns have similar performance, suggesting task objectives obtain good policies with reduced reward engineering effort (see Appendix B.5 for further discussion). Unsurprisingly, AutoRL over the standard reward produces the highest scores, when evaluated on the standard reward in 13 out of 16 conditions (Walker, Standup in Figure 2d, and Ant in Figure 2e. However, videos show the policies differ in style: Humanoid optimized for the standard reward produces a jumping and falling loop,



(a) SAC on Task obj. (b) PPO on Task. obj. (c) Std. Return SAC Humanoid

(d) SAC on Std. Reward (e) PPO on Std. Reward (f) Std. Return PPO Humanoid

Figure 2: Cross-evaluation w.r.t. standard returns (SAC (a), PPO (b)) and task objectives (SAC (d), PPO (e)). Standup and Ant are scaled. c) and f) Sorted Vizier trials show the benefits of reward tuning over hyperparameter tuning.

while the height-reached policy stably rises to just above kneeling.[1] We leave it for future work to apply AutoRL for non-scalarized multi-objective optimization problems.

Reward vs HP tuning: AutoRL shows promising benefits for hyperparameter optimization performance: evolving rewards produces more high-performing trials and has a higher peak performance than hyperparameter tuning on both SAC and PPO (Fig. 2 a,b,d,e). Given a limited computational training budget, reward tuning explores better policies than hyperparameter tuning (Fig. 2 c,f) and is more likely to produce good policies.

4.1 Conclusion

In this paper we learn proxy rewards for continuous control tasks with AutoRL, a method that automates RL reward design by using evolutionary optimization over a given objective. Benchmarking over two RL algorithms, four MujoCo tasks, and two different true objectives show that: a) AutoRL outperforms both hand-tuned and learning hyperparameter tuned RL; b) produces comparable and often superior policies with a simpler true objective, hence reducing human engineering time; and c) often produces better policies faster than hyperparameter tuning, suggesting that under a limited training budget tuning proxy rewards might be more beneficial than tuning hyperparameter and that a more in-depth analysis would be appropriate. All three conclusions hold even stronger for more complex

1. `https://youtu.be/svdaOFfQyC8`

environments such as Humanoid, making AutoRL a promising technique for training RL agents for complex tasks, with less hand engineering and better results.

Acknowledgments

We thank Oscar Ramirez, Rico Jonschkowski, Shane Gu, Sam Fishman, Eric Jang, Sergio Guadarrama, Sergey Levine, Brian Ichter, Hao-Tien Chiang, Jie Tan & Vincent Vanhoucke for their input.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *CoRR*, abs/1707.01495, 2017. URL `http://arxiv.org/abs/1707.01495`.

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

Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. Efficient architecture search by network transformation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2787–2794, 2018.

Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2722–2730, 2015.

Hao-Tien Lewis Chiang, Aleksandra Faust, Marek Fiser, and Anthony Francis. Learning navigation behaviors end-to-end with autorl. *IEEE Robotics and Automation Letters*, 4(2):2007–2014, April 2019. ISSN 2377-3766. doi: 10.1109/LRA.2019.2899918.

Tom Erez, Yuval Tassa, and Emanuel Todorov. Infinite horizon model predictive control for nonlinear periodic tasks. *Manuscript under review*, 4, 2011.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. Reverse curriculum generation for reinforcement learning. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 482–495. PMLR, 13–15 Nov 2017.

Daniel Golovin, Benjamin Solnik, Subhodeep Moitra, Greg Kochanski, John Karro, and D. Sculley. Google vizier: A service for black-box optimization. In *Proc. of ACM International Conference on Knowledge Discovery and Data Mining*, pages 1487–1495. ACM, 2017. doi: 10.1145/3097983.3098043.

Sergio Guadarrama, Anoop Korattikara, Oscar Ramirez, Pablo Castro, Ethan Holly, Sam Fishman, Ke Wang, Ekaterina Gonina, Chris Harris, Vincent Vanhoucke, and Eugene Brevdo. TF-Agents: A library for reinforcement learning in tensorflow. `https://github.com/tensorflow/agents`, 2018. URL `https://github.com/tensorflow/agents`.

Izzeddin Gur, Ulrich Rueckert, Aleksandra Faust, and Dilek Hakkani-Tur. Learning to navigate the web. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BJemQ209FQ`.

Tuomas Haarnoja, Aurick Zhou, Sehoon Ha, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *CoRR*, abs/1812.11103, 2018a. URL `http://arxiv.org/abs/1812.11103`.

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, 2018b. URL `http://arxiv.org/abs/1812.05905`.

Boris Ivanovic, James Harrison, Apoorva Sharma, Mo Chen, and Marco Pavone. Barc: Backward reachability curriculum for robotic reinforcement learning. *CoRR*, abs/1806.06161, 2018.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Scalable deep reinforcement learning for vision-based robotic manipulation. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 651–673. PMLR, 29–31 Oct 2018.

Shauharda Khadka and Kagan Tumer. Evolution-guided policy gradient in reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 1188–1200. Curran Associates, Inc., 2018.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

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. *CoRR*, abs/1509.02971, 2015. URL `http://arxiv.org/abs/1509.02971`.

Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan L. Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *CoRR*, abs/1712.00559, 2017. URL `http://arxiv.org/abs/1712.00559`.

Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2902–2911. JMLR.org, 2017.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Evolutionary algorithms and reinforcement learning: A comparative case study for architecture search. In *AutoML@ICMLRobotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, 2018a.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. Regularized evolution for image classifier architecture search. *CoRR*, abs/1802.01548, 2018b. URL `http://arxiv.org/abs/1802.01548`.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2015.

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

Pararth Shah, Marek Fiser, Aleksandra Faust, Chase Kew, and Dilek Hakkani-Tur. Follownet: Robot navigation by following natural language directions with deep reinforcement learning. In *Third Machine Learning in Planning and Control of Robot Motion Workshop at ICRA*, 2018.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Pack Kaelbling. Residual policy learning. *CoRR*, abs/1812.06298, 2018. URL http://arxiv.org/abs/1812.06298.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Information-theoretic regret bounds for gaussian process optimization in the bandit setting. *IEEE Transactions on Information Theory*, 58(5):3250–3265, May 2012. doi: 10.1109/TIT.2011.2182033.

Richard S Sutton, Andrew G Barto, and Ronald J Williams. Reinforcement learning is direct adaptive optimal control. *IEEE Control Systems*, 12(2):19–22, 1992.

Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913, Oct 2012. doi: 10.1109/IROS.2012.6386025.

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

Eric Wiewiora. *Reward Shaping*, pages 863–865. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_731. URL https://doi.org/10.1007/978-0-387-30164-8_731.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8697–8710, 2018a.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8697–8710, 2018b.

Appendix A. AutoRL Parameter Settings

We use the TF Agents (Guadarrama et al., 2018) implementation of SAC (Haarnoja et al., 2018b,a) and PPO (Schulman et al., 2017). In the PPO implementation, training steps shown in the charts are equal to the number of gradient update steps, and training is done 25 times every N episodes where N is environment-specific and defined in Table 1. For Ant and Walker the approximate number of environment steps is 19.8 million, and for Standup and Humanoid it is 158.4 million.

Table 1: Environments

Name	Description	Reference	PPO Episodes per Iteration	State Dim.	Action Dim.
Ant	Ant 3D Locomotion	Schulman et al. (2015)	30	111	8
Walker	Walker 2D Locomotion	Erez et al. (2011)	30	17	6
Standup	Humanoid Standup	N/A	240	376	17
Humanoid	Humanoid Locomotion	Tassa et al. (2012)	240	376	17

Table 2: Tuned Reward Parameters

Parameters	Ant	Walker	Standup	Humanoid
Achievement	Linear Velocity	Linear Velocity	Height / Time	Linear Velocity
Cost	Control Cost	Control Cost	Quadratic Cost	Control Cost
Impact	Contact Cost	N/A	Quadratic Impact	Impact Cost
Survival	Alive Bonus	Alive Bonus	Alive Bonus	Alive Bonus

Appendix B. True Objective Selection

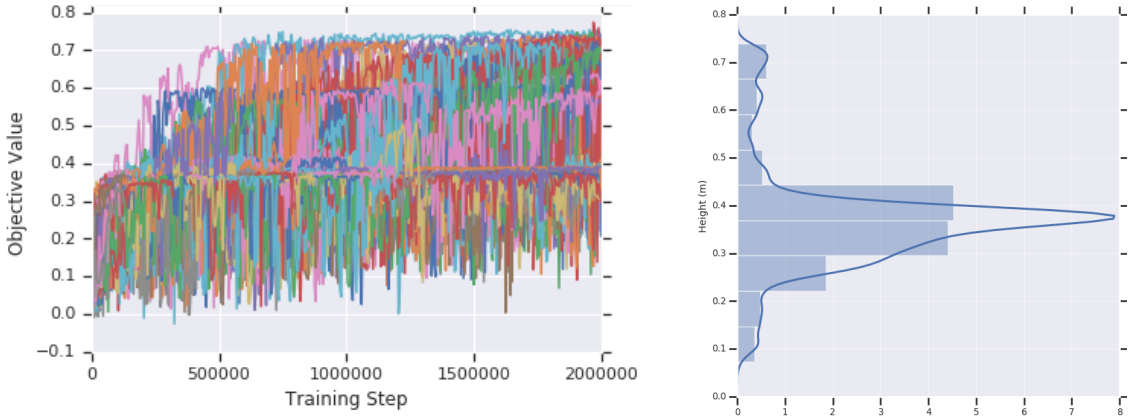
In our prior work for point-to-point navigation tasks (Chiang et al. (2019)), we used sparse true objectives such as reaching a goal. For continuous control tasks, however, this is problematic, because sparse true objectives are difficult to achieve, while intermediate stages in learning are valuable. Figure 3 illustrates this for Humanoid Standup. A sparse true objective for this task is for the agent to stand up to approximately 1.4 m, but few agents successfully achieve this height. Instead, plateaus of performance can be seen at intermediate heights of 0.4 m, 0.6 m and 0.75 m where agents have likely learned important intermediate behaviors, such as sitting up or rising to one knee. All of these behaviors look the same to a sparse true objective: they are failures.

B.1 Single-Task Objectives

Metric-based objectives provide a way out of the sparse optimization issue by splitting the difference between a true sparse objective and a hand-engineered reward. Without committing to reward

Table 3: Tuned Hyperparameters

Algorithm	Parameters	Hand Tuned	Algorithm	Parameters	Hand Tuned
				Normalize Observations	True
				Normalize Rewards	True
	Replay Buffer Size	256		Episodes per Iteration	See Table 1
	Target Update τ	0.005		Learning Rate	0.0001
	Target Update Period	1		Entropy Regularization	0.0
SAC	γ	0.99	**PPO**	Importance Ratio Clipping	0.0
	Critic Learning Rate	0.0003		KL Cutoff Factor	2.0
	Actor Learning Rate	0.0003		KL Cutoff Coefficient	100
	Alpha Learning Rate	0.0003		Initial Adaptive KL Beta	1.0
				Adaptive KL Target	0.01
				Discount Factor	0.995



(a) Training on Humanoid Standup

(b) Histogram of Best Values

Figure 3: Optimizing Humanoid Standup over a dense true objective. a) Training performance for approximately 100 agents. b) Histogram of final values achieved by 1000 studies.

component weights, one can say that policies that achieve more height are preferred for standup tasks, and policies that achieve more distance are preferred for locomotion. Metric-based single-task objectives (2) form a metric space over the space of policies, and induce a partial ordering of all policies w.r.t. the $<$ relation. They reach the true objective in the limit and are continuous, providing a clear signal to the evolutionary optimization process. We use two metric-based task objectives: distance traveled for Ant, Walker and Humanoid, and height achieved for Humanoid Standup.

B.2 Multi-Objective Standard Returns

Another kind of dense true objective is the standard return by which tasks are normally evaluated. The rewards that these returns are based on are generally multiple objectives combined with hand-engineered weights, and may not have the same gradient or maxima as the true value function given the environment dynamics. A parameterization which more closely matches the true value function will encourage agents to make good decisions earlier in training and to explore more fruitful parts of the search space. Therefore, optimizing a reparameterization of the standard reward against the standard return can yield improved performance.

B.3 Single-Task Objective Evaluation

Analysis of the videos[2] for optimization over task objectives reveals that SAC generally outperforms PPO, and AutoRL outperforms hand-tuning and hyperparameter tuning. Hand and hyperparameter tuning frequently fell on PPO, whereas AutoRL for SAC had the fastest travel. For Humanoid-Standup, SAC again performed better than PPO, while AutoRL performed better than hand tuning and hyperparameter tuning. No policy fully stood, but both hyperparameter tuning and AutoRL on SAC rose to a consistent crouch.

2. `https://youtu.be/svdaOFfQyC8`

Table 4: AutoRL Parameterized Reward Weights with SAC

Parameters	Ant		Walker		Standup		Humanoid	
	Std. Obj.	Task Obj.	Std. Obj.	Task Obj.	Std. Obj.	Task Obj.	Std. Obj.	Task Obj.
Achievement	0.10205	0.34949	0.19719	0.46260	0.33561	0.21251	0.86552	0.90872
Cost	0.34042	0.15624	0.77295	0.97506	0.65550	0.96195	0.54985	0.99147
Impact	0.20531	0.53880	N/A	N/A	0.34548	0.89288	0.31891	0.33972
Survival	0.05205	0.01969	0.01421	0.47166	0.86399	0.97217	0.02702	0.06027

Table 5: AutoRL Parameterized Reward Weights with PPO

Parameters	Ant		Walker		Standup		Humanoid	
	Std. Obj.	Task Obj.	Std. Obj.	Task Obj.	Std. Obj.	Task Obj.	Std. Obj.	Task Obj.
Achievement	0.73208	0.97706	0.08453	0.71359	0.53268	0.67899	0.40621	0.19427
Cost	0.84979	0.73720	0.21984	0.04812	0.73103	0.88079	0.32404	0.49128
Impact	0.35485	0.80661	N/A	N/A	0.62980	0.40959	0.35179	0.84411
Survival	0.46004	0.44921	0.00677	0.06200	0.50112	0.01883	0.08046	0.046723

B.4 Multi-Objective Standard Return Evaluation

Evaluation of optimization over the standard return was similar to evaluation over the task objective, except we collected returns over the complete standard rewards listed in Table 2 (see Section B.2). AutoRL was consistently superior for SAC and superior for Walker and Humanoid in PPO.

Analysis of the videos[3] for optimization over the standard returns similarly reveals that SAC generally outperforms PPO, that AutoRL outperforms hand-tuning and hyperparameter tuning, and that hand and hyperparameter tuning frequently fell in PPO, whereas AutoRL for SAC had the fastest travel. For HumanoidStandup, SAC again performed better than PPO, while AutoRL performed better than hand tuning and hyperparameter tuning. AutoRL on SAC was the only policy that fully stood, though it got into a falling and standing loop, whereas hyperparameter tuning rose to a consistent crouch.

B.5 Cross-Evaluation of Single-Task Objectives and Standard Returns

All the environments in Table 1 define standard rewards with predefined components listed in Table 2. AutoRL's reward optimization changes the parameterization of these components, but the return collected over the standard reward parameterization is the normal way that these environments are evaluated. However, task objective optimization is evaluating over a different objective - normally, just the achievement objective of Table 2. We would expect optimizing over a different reward to produce different performance on the standard return; conversely, we would expect the standard return to produce different performance on the task objectives.

To enable a fair comparison of both conditions, we conducted a cross-evaluation study in which we evaluated policies optimized for task objectives against the standard returns (Fig. 4 a-h), as well as policies optimized on the standard returns against the task objectives (Fig. 4 i-p). These results show task objectives and standard returns have similar performance, suggesting task objectives obtain good policies with reduced reward engineering effort.

As discussed in the text, detailed analysis reveals differences in these objectives. Videos show differences in style, and in Humanoid, the task objective agents travel the farthest for both PPO and SAC (Figures 2a and 2b dark red vs light red), while on other tasks optimizing over the task objective is comparable to optimizing over the standard reward. Unsurprisingly, AutoRL over the standard reward produces the highest scores when evaluated on the standard reward in most conditions.

3. `https://youtu.be/svdaOFfQyC8`

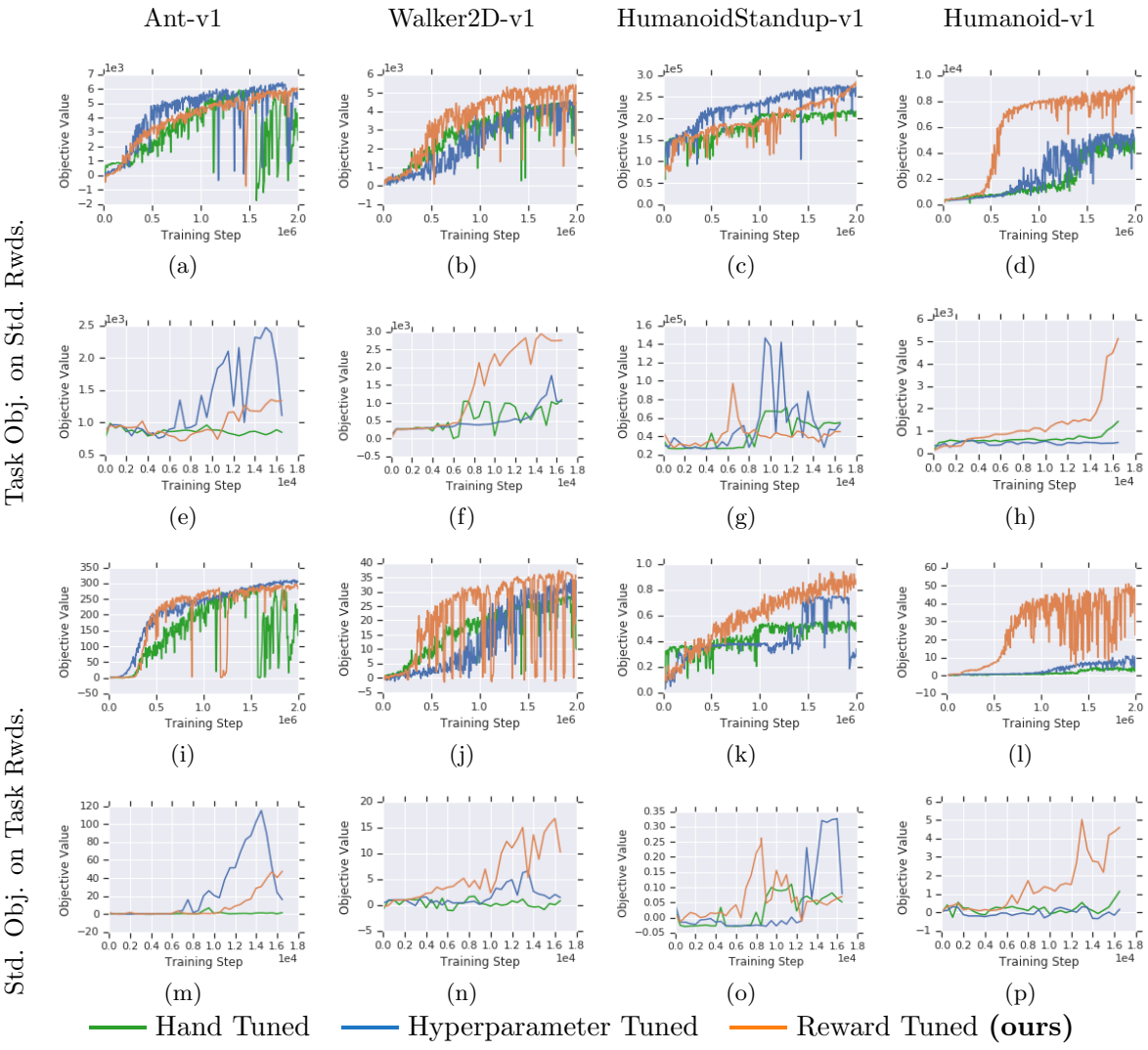


Figure 4: AutoRL policies optimized over task objectives evaluated on standard return a-d) using SAC and e-h) using PPO. AutoRL policies optimized over standard return evaluated on task objectives i-l) using SAC and m-p) using PPO.