The Successful Ingredients of Policy Gradient Algorithms

Sven Gronauer* , **Martin Gottwald** , **Klaus Diepold**
Technical University of Munich, Germany
{sven.gronauer, martin.gottwald, kldi}@tum.de

Abstract

Despite the sublime success in recent years, the underlying mechanisms powering the advances of reinforcement learning are yet poorly understood. In this paper, we identify these mechanisms - which we call ingredients - in on-policy policy gradient methods and empirically determine their impact on the learning. To allow an equitable assessment, we conduct our experiments based on a unified and modular implementation. Our results underline the significance of recent algorithmic advances and demonstrate that reaching state-of-the-art performance may not need sophisticated algorithms but can also be accomplished by the combination of a few simple ingredients.

1 Introduction

Reinforcement learning (RL) is a data-driven paradigm that can be leveraged to learn complex strategies for controlling dynamical systems. RL algorithms excel at problems that can be simulated or where exact models are known, but conventional planning is not feasible, e.g. in the game of Go [Silver *et al.*, 2016]. Although the RL domain has witnessed significant advances in recent years [Arulkumaran *et al.*, 2017; Mnih *et al.*, 2015], the progress is aggravated by various impediments. First, experiments are difficult to reproduce because numerical results are sensitive to the selection of hyper-parameters and can vary over different random seeds [Henderson *et al.*, 2018; Islam *et al.*, 2017]. More drastically, different code bases produce inconsistent results, although describing the same algorithm. Second, a major share of the claimed performance increments in recently proposed methods is less achieved by innovative algorithmic properties but more through clever implementation [Engstrom *et al.*, 2020; Tucker *et al.*, 2018]. Third, state-of-the-art algorithms based on neural networks can be disputed by simpler learning models. Despite their expressiveness, works demonstrated that representations like radial basis functions or linear function mappings could achieve similar results on contemporary benchmarks [Mania *et al.*, 2018; Rajeswaran *et al.*, 2017].

Until yet, it is poorly understood which underlying mechanisms drive the learning of RL agents. The intricate interplay between different algorithm components and the abundance of adjustable parameters render it difficult to study the roots of the recent progress. Our understanding remains opaque until we assess the importance of new algorithmic innovations through careful analysis. First works investigated the underlying mechanisms by conducting ablation studies in large-scale experiments [Andrychowicz *et al.*, 2020] or on a selected subset of parameters [Engstrom *et al.*, 2020]. To maintain sustainable progress, the RL community must build a profound understanding of the ingredients and their respective proportion to the learning success, individually and as a whole. Empirical as well as theoretical analysis about why an algorithm surpasses the other is therefore crucial as argued by Sigaud and Stulp [2019].

In this paper, we shed light on the components - which we denote as *ingredients* - powering on-policy policy gradient algorithms in continuous control problems.[1] Our contribution is two-fold: (1) we empirically study the significance of specific ingredients and show the roots of algorithmic progress based on a modular and unified code base and (2) identify a minimal setup of ingredients that challenges state-of-the-art approaches while exhibiting a manageable size of algorithm complexity. According to the principle of Occam's Razor,[2] a simple baseline is preferable because fewer hyper-parameters must be tuned and, thus, is easier to reproduce.

2 Related Work

A plethora of literature has emerged in recent years that criticizes the reproducibility of RL. First of all, [Islam *et al.*, 2017] investigated the influence of hyper-parameter choices and revealed the inherent performance variance of RL algorithms. The authors objected to the under-reporting of hyper-parameters, which are crucial for a fair comparison between different algorithms, and provided recommendations for good research practice. In a similar vein, [Henderson *et al.*, 2018]

*Contact Author

[1] For the supplemental materials and the implementation see: https://github.com/SvenGronauer/successful-ingredients-paper

[2] We associate the minimum description length as one form of interpretation with the principle of Occam's Razor, which states the trade-off between the quality of data fitting and the complexity of the used algorithm model [Rissanen, 1978].

emphasized the intricate interplay between hyper-parameter selections and underlined the importance of independent random seeds used to evaluate experiment trials. Further, Henderson *et al.* [2018] pointed out that different code bases, although describing the same underlying algorithm, can yield inconsistent results, and they gave evidence that these performance deviations can be attributed to implementation details which are often not neatly reported. Related to performance disparities caused by different code bases, Engstrom *et al.* [2020] showed that the implementation details indeed matter. The authors investigated selected code-level ingredients, which are found only in the implementation or described in detail in the appendices, and showed that these could be accounted for the performance gains observed between the TRPO and PPO algorithm. Alike, Tucker *et al.* [2018] observed that recent advances in gradient estimation, which were originally related to algorithmic improvements, can be attributed to subtle implementation details.

Similar to our empirical research, Reda *et al.* [2020] investigated the impact of the environment design on the learned policy performance in locomotion tasks, showing that the problem design is at least as important as the selected algorithm. Most related to our work, Andrychowicz *et al.* [2020] discussed the effect of ingredients on on-policy RL algorithms through the conduction of large-scale experiments. They provided practical recommendations for high and low-level choices and corresponding hyper-parameters values but, in contrast to our work, conducted only ablation studies and no incremental approach.

3 Background

We consider model-free policy gradient algorithms in continuous control problems where a policy is searched by interacting with an environment with unknown dynamics. We make use of the standard formulation for discounted infinite-horizon Markov decision processes (MDP) which are formalized by the tuple $(\mathcal{X}, \mathcal{U}, \mathcal{P}, r, \gamma)$, where $\mathcal{X}$ and $\mathcal{U}$ denote state and action space, respectively. The system transition probability is described by $\mathcal{P}$ while r is the reward function and γ denotes the discount factor. The agent's goal is to learn a policy $\pi : \mathcal{X} \to \mathcal{P}(\mathcal{U})$ that maximizes the expected return

$$J(\pi) = \int_{\mathcal{X}} \rho_\pi(x) \int_{\mathcal{U}} \pi(u|x)\, r(x,u)\, du\, dx. \tag{1}$$

Under the assumption of an ergodic MDP and an infinite horizon, the problem is stationary and the expected policy performance can be calculated over the un-normalized steady state distribution $\rho_\pi(x) = \gamma^0 P(x_0 = x) + \gamma^1 P(x_1 = x) + \dots$ under policy π. The policy π_θ is represented by a neural network that is parametrized by the vector θ and assumed to be at least once differentiable. We denote the state-value function as $V_{\pi_\theta}(x) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, u_t) \mid x_0 = x\right]$, and similarly $Q_{\pi_\theta}(x,u) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, u_t) \mid x_0 = x, u_0 = u\right]$ as the action-value function. Policy gradient methods optimize the learning objective by building the gradient of the expected return with respect to the policy parameters $\nabla_\theta J(\pi_\theta)$ and update the policy parameters by taking a step along the gradient direction. Likelihood ratio methods require a re-sampling of data for every gradient step. To reuse generated trajectory data over multiple iterations, importance sampling can be applied to perform updates based on a local approximation

$$\hat{J}(\pi_\theta) = \mathbb{E}_\mu\left[\frac{\pi_\theta(u|x)}{\mu(u|x)}\Psi(x,u)\right] \tag{2}$$

that matches J to first order. Off-policy samples can be incorporated under the assumption that $\mu(\cdot|x) = 0 \Rightarrow \pi_\theta(\cdot|x) = 0$ for all x. However, the estimation of the policy gradient can exhibit high variance. Thus, a reward estimator Ψ is typically used which can be expressed by several terms as proposed by Schulman *et al.* [2016] but typically takes the form of the advantage function $A_{\pi_\theta}(x,u) = Q_{\pi_\theta}(x,u) - V_{\pi_\theta}(x)$.

4 Methodology

In this paper, we consider on-policy gradient methods[3] in continuous control domains. We take the importance-weighted policy gradient (IWPG) objective from Eq. (2) as the basis and systematically add ingredients to observe their impact on the learned policy performance. We set up our experiments in an incremental and modular approach such that the influence of individual ingredients becomes transparent. In particular, we study ingredients chronologically in three stages:

1. *Code-level ingredients* are enhancements to the algorithm, which are provided as supplementary details or can only be found in the implementation. Most of these ingredients are considered as good practices in the RL community and are not regarded in the hyper-parameter search, leaving their impact on the learning unexplored.
2. *Algorithmic ingredients* lie at the heart of new algorithm proposals and depict the core innovation. These ingredients are precisely described in theoretical terms, but the realization may only become transparent from the provided implementation code.
3. *Structural ingredients* refer to choices and hyper-parameters that describe the neural network architecture and the optimization. We consider those ingredients that can vary between different implementation frameworks and are often neglected in the experimental description.

Our experiments are conducted in the procedure from above to cope with two challenges. First, ingredients cross-correlate, making it difficult to determine their causal direction of influence. We alleviate cross-correlations by randomly adding ingredients and assess their impact based on the performance difference, whether an ingredient is applied or not. Second, the combination of configurations grows exponentially with the number of tunable hyper-parameters. We try to contain this through the proposed experimental structure in three stages.

[3] As on-policy, we also denote policy iteration algorithms that collect data based on the current policy and update a policy several times by using the same data through importance weighting before generating new data with the iterated policy. Transition samples become off-policy after the first policy update but are still close to the generating policy.

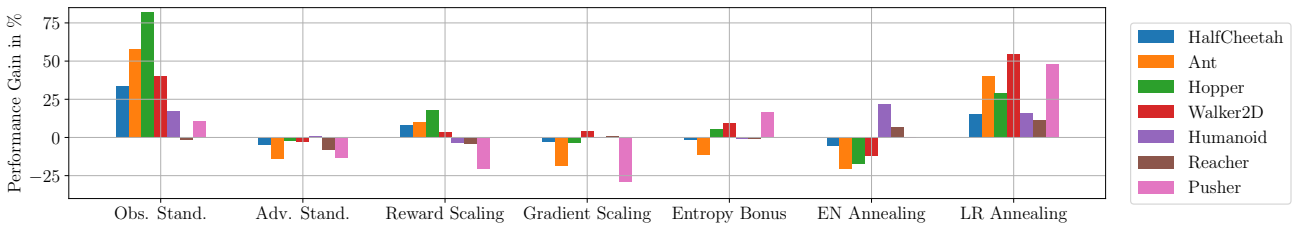

Figure 1: The impact of code-level ingredients on the policy performance. The normalized scores show the relative change of the policy performance when a particular ingredient is used. Note that these numbers are generated based on the IWPG objective without algorithmic ingredients such as trust-regions, which are added thereafter in Experiments 5.2. The Kuka task is excluded since we were not able to improve upon a random policy without advanced variance reduction methods.

To benchmark the performance in continuous control problems, we use the five locomotion environments HalfCheetah, Hopper, Ant, Walker2D, and Humanoid as well as the three robotic manipulations tasks Reacher, Pusher, and Kuka. We measure the expected return after 10^7 environment interactions when exploration noise is disabled. All eight tasks are evaluated in the PyBullet physics engine [Coumans and Bai, 2016].

5 Experiments

In this section, we explain the details of our experiments and present the results. We aligned the hyper-parameters to the ones suggested in Henderson *et al.* [2018]. We applied a single learner setup, used as discount factor $\gamma = 0.99$, collected batches of size 32000 for each policy iteration, and ran each seed over a total of 10^7 environment interactions. For the neural networks, we used the same structure for both policy and value networks, i.e. multi-layer perceptrons with two hidden layers consisting of 64 neurons each followed by tanh non-linearities. The default optimizer was Adam [Kingma and Ba, 2015] for the policy and value network, respectively. Our studied ingredients are only applied to the policy network but not to the value network. For all experiments, we performed hyper-parameter grid searches to average the effect of ingredients over a wide range of hyper-parameters (Experiments 5.1) or to determine the best hyper-parameter configuration (Experiments 5.2-5.4). An overview of the used hyper-parameters and configurations is provided in the supplemental.

5.1 Impact of Code-level Ingredients

Being augmentations to the algorithmic core, code-level ingredients are provided as supplementary details or can only be found in implementations. Although not grounded on theoretic insights, many code-level ingredients are used as a heuristic on current benchmarks. We agree on these ingredients as the first stage since they can be applied to all on-policy gradient algorithms and are leveraged in the succeeding experiments. We investigated the seven following ingredients:

1. *Observation Standardization.* Each batch of observations is made mean-free and re-scaled to unit variance by first subtracting the mean value and then dividing through the standard deviation.
2. *Advantage Standardization.* Analogous to observation standardization, each batch is transformed into a mean-free and unit variance distribution.
3. *Reward Scaling.* The received rewards are divided by the standard deviation of a running discounted sum of rewards.
4. *Gradient Scaling.* To maintain the magnitude, parameter gradients (as they are concatenated into a single vector) are re-scaled if they exceed a certain threshold.
5. *Entropy Bonus Term.* To promote exploration, the entropy bonus term can be added to the optimization objective, which encourages uniformly distributed actions.
6. *Exploration Noise Annealing.* Outputs of the policy networks typically represent the mean of a multivariate Gaussian distribution. Through exploration noise annealing, the entries of the covariance matrix are linearly decreased over the training.
7. *Learning Rate Annealing.* The learning rate of the policy optimizer is linearly decayed to zero over the training.

Approach. To study how code-level ingredients affect learning, we randomly added each ingredient to the IWPG objective from (2) by a chance of 50% in each run. We used the random enabling to remedy cross-correlations between the investigated ingredients. We conducted experiments over a 4×4 grid of learning rate and number of policy iterations combinations. Each combination was evaluated over 16 independent runs, resulting in a total of 256 random seeds for each environment. The impact of each code-level ingredients was determined by $(J_\iota^+ - J_\chi)/(J_\iota^- - J_\chi)$ where J_ι^+ denotes the average policy performance when ingredient ι is applied, and J_ι^- when ingredient ι is not used, and J_χ is the performance of a uniform random policy χ. We used plain advantage estimation $\Psi = A$ as the reward estimator.

Results. As shown in Figure 1, observation standardization and learning rate annealing significantly affect the learned policy performance. Both ingredients yield in at least six out of seven tasks performance gains, where observation standardization yielded an average performance increase of 34.3% and learning rate annealing an improvement of 30.7%. Reward Scaling showed in four out of five locomotion tasks performance gains, while for Humanoid a negative score.

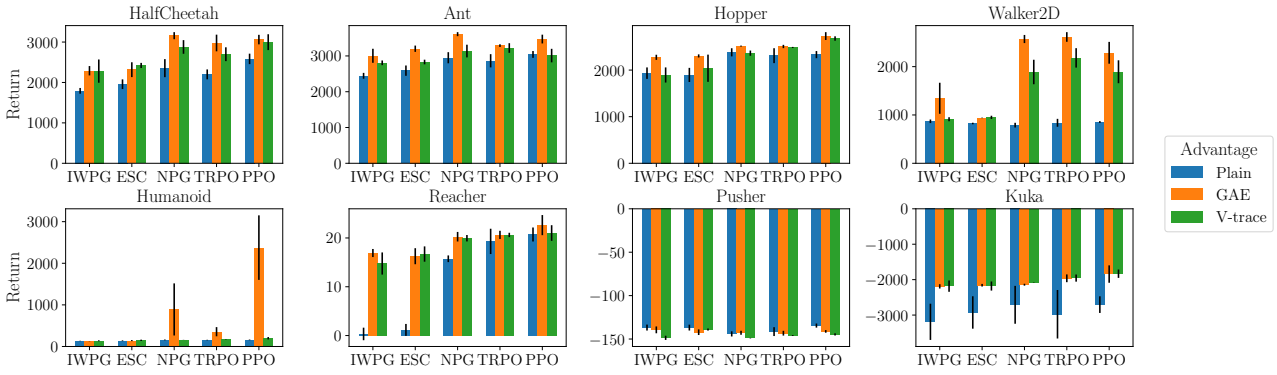

Figure 2: Impact of the algorithmic ingredients on the policy performance. Different trust-region enforcement methods are compared conditioned on the variance reduction method of the critic estimates. Only the best results obtained through grid search are depicted.

The manipulation tasks did neither profit from reward scaling. The entropy bonus term had a mixed impact, showing in Pusher and Walker gains of at least 9.7%, whereas Ant denotes a performance loss of 11.6%. The annealing of the exploration noise showed in four out of five tasks a negative impact, but performance gains in Humanoid, Reacher, and Pusher. In contrast to the former ingredients, both the scaling of policy gradients and the standardization of advantages showed on average performance losses. Note that we could not achieve an improvement over a random policy at the Kuka task without advanced variance reduction methods and, thus, we excluded the Kuka environment from Figure 1. The plots visualizing the learning curves and the numerical results of all eight tasks can be taken from the supplemental.

5.2 Effect of Algorithmic Ingredients

In this section, we elaborate on the former experiment's code-level findings and endow the learning objective with *algorithmic ingredients*, which are claimed to be responsible for the state-of-the-art achievements.

Jumps in policy parameter updates can deteriorate the learned performance and may eventually cause policy collapse [Duan *et al.*, 2016]. However, well-chosen policy updates can alleviate this issue by constraining the distance between consecutive policy iterates through *trust-region enforcement*. We investigated the following four methods:

1. *IWPG with an Early Stopping Criterion* (ESC) measures the KL divergence between consecutive policy iterates and terminates the updates when the distance criterion is met.
2. *Natural Policy Gradient* (NPG) [Kakade, 2002] regards the curvature of the policy parameter space by approximating the Fisher information matrix. The update direction is given by the matrix-vector multiplication between the Fisher information matrix and the policy gradient.
3. *Trust-region Policy Optimization* (TRPO) [Schulman *et al.*, 2015] optimizes over a surrogate function, which matches the original objective to first order. In contrast to NPG, TRPO explicitly regards the KL divergence between policy iterates by performing a backtracking line search until the distance criterion is fulfilled.
4. *Proximal Policy Optimization* (PPO) [Schulman *et al.*, 2017] utilizes clipped probability ratios to maintain the step size in the parameter space. The resulting surrogate objective is then optimized to first order.

Low variance critic estimates are considered as crucial component for good policy performance [Peters and Schaal, 2008]. In our experiments we studied three techniques for the *Variance Reduction of Critic Estimations*:

1. *Plain Advantage Estimation* is used as baseline where the advantages are determined by $A(x_t, u_t) = r(x_t, u_t) + \gamma V_\pi(x_{t+1}) - V_\pi(x_t)$.
2. *Generalized Advantage Estimation* (GAE) [Schulman *et al.*, 2016] aims to reduce variance in critic estimates at the cost of introducing bias. Advantages are calculated by the weighted sum $A(x_t, u_t) = \sum_{k=0}^{n-1} (\lambda\gamma)^k \, \delta_{t+k}$ of temporal differences $\delta_{t+k} = r(x_{t+k}, u_{t+k}) + \gamma V_\pi(x_{t+k+1}) - V_\pi(x_{t+k})$. The scalar λ governs the trade-off between variance and bias.
3. *V-trace* [Espeholt *et al.*, 2018] was introduced to account for off-policy critic updates in distributed architectures to compensate policy lags. Updates follow $A(x_t, u_t) = r(x_t, u_t) + \gamma v(x_{t+1}) - V(x_t)$ while the values become $v(x_t) = V(x_t) + \sum_{s=t}^{t+n-1} \gamma^{s-t} \left(\prod_{i=t}^{s-1} c_i \right) \delta_s V$.

Approach. We continued to apply observation standardization and linear learning rate decay as default code-level ingredients. Reward scaling was added only to the locomotion tasks but not to the manipulation tasks. The experiments were run over all possible combinations of trust-region enforcement and variance reduction methods. We conducted a grid search over learning rates and numbers of policy iterations and determined the best configuration according to the highest $\text{mean}(J) - \text{std}(J)$ averaged over four independent seeds (see detailed hyper-parameters in the supplemental). As a reference for comparison, we used the plain IWPG objective from Experiments 5.1 which makes no use of a trust-region enforcement and uses plain advantage estimation.

Results. Figure 2 indicates that low variance critic estimates significantly boost the policy performance. In particular, for complex domains such as Humanoid and Kuka,

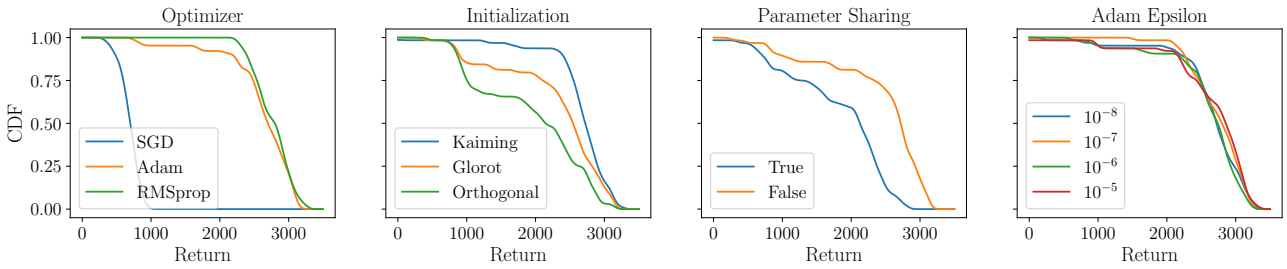

Figure 3: The CDF line plot of the policy performance smoothed over 64 independent seeds for each configuration in the Ant environment.

all tested algorithms could not accomplish high returns with plain advantage estimates. The combination of PPO and GAE showed the best results, scoring in three out of eight tasks the highest performance. Even when plain advantage estimation is used, PPO showed the most robust results across all tasks. Surprisingly and despite its simplicity, the IWPG algorithm performed without trust-region enforcement only slightly worse than its counterparts but necessitated the use of GAE or V-trace. ESC adds an early stopping criterion to IWPG which resulted in a more stable learning and overall small performance gains. This underlines the importance of constraining step sizes in the parameter space. The learning curves of all eight environments can be found in the supplemental material.

5.3 Study of Structural Ingredients

In this part, we analyze the effect of structural ingredients on the learned performance that can vary between different software frameworks. Some might be neglected in the experimental description and can only be found in the implementation. We investigated the following four ingredients:

1. *Optimizer.* Besides Adam, we investigated Stochastic Gradient Descent (SGD) and RMSprop as optimizers for the policy network.
2. *Initialization scheme.* We compared Kaiming Uniform, Glorot, and Orthogonal which determine the initial values of the neural network weights. The bias parameters were initialized as zero vectors.
3. *Parameter Sharing.* Accelerating the learning of representation features, parameters can be shared and updated by both policy and value network.
4. *Adam Epsilon.* A term added to the denominator of the update step to improve the numerical stability of Adam.

Approach. Continuing the stage-wise analysis, we built on the findings from previous experiments and used IWPG with observation standardization and learning rate annealing as well as reward scaling (only for locomotion tasks) as code-level ingredients and applied GAE for variance reduction. We performed a grid search over 4×4 combinations of policy learning rate and number of training iterations and accumulated the scores as cumulative distribution functions (CDF) over four independent seeds, resulting in 64 independent runs for each environment (see Figure 3).

Results. The scores for the optimizer vary from environment to environment. RMSprop showed across all environments the best average CDF score and was the most robust choice over the grid search, whereas Adam yielded the highest performing configuration. The weight initialization scheme showed in Ant, Humanoid, and Kuka large impacts, whereas in other tasks negligible effects. Overall, the Kaiming Uniform scheme yielded the most robust scores. The parameter sharing of two hidden layers between the policy and value network resulted in all tasks to worse performance scores than separated networks, except for the Pusher environment. Surprisingly, the ϵ term added to the denominator of Adam's policy updates showed an influence on the learning, where $\epsilon \in \{10^{-8}, 10^{-5}\}$ yielded the best average performance scores across all tasks. The plots for all eight environments are provided in the supplemental materials.

5.4 Identifying the Minimal Setup

The second objective of our paper is to find a minimal configuration that can challenge state-of-the-art RL algorithms. We inferred from our previous findings and used the IWPG objective with standardized observations, learning rate annealing and GAE as the minimal setup. Reward scaling was applied in the locomotion tasks but not in the manipulation tasks. We tested both RMSprop and Adam as policy optimizers and applied the Kaiming initialization scheme. To validate the reliability of our results, we benchmarked the identified setup against the popular OpenAI Baselines repository [Dhariwal *et al.*, 2017]. For each algorithm, we conducted a grid search over different learning rates, the number of policy iterations and trust-region sizes, while fixing the other hyperparameters such as the batch size of generated trajectories, networks architectures for policy and value network, etc. We depict the best score averaged over four independent seeds for each code base in Figure 4. The results show that our identified setup of ingredients can keep up with the state-of-the-art algorithms TRPO and PPO.

6 Discussion

Our experiments showed that observation standardization is crucial for good policy performance, which was also reported in Andrychowicz *et al.* [2020] and Engstrom *et al.* [2020]. The only task that did not profit was Reacher. We suppose that this is due to the well-designed observation space with small magnitudes and that random exploration already discovers the state space sufficiently well. The second essen-

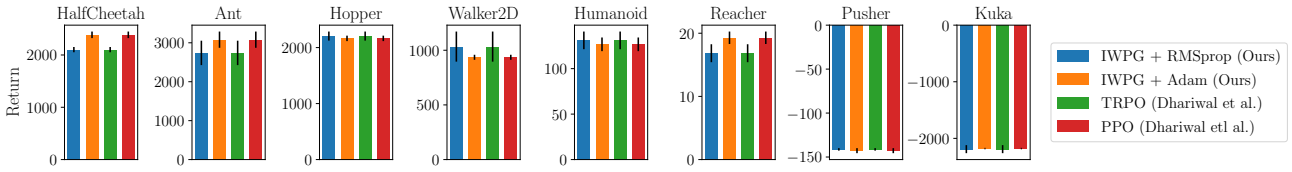

Figure 4: Comparison between our identified minimal setup of ingredients and the two algorithms TRPO and PPO from the OpenAI's Baselines repository. All algorithms use the same default hyper-parameters and the best scores are determined through grid search.

tial ingredient is learning rate annealing. We utilized a linear decay scheme which showed in all tasks performance gains. Our conclusion is similar to the one of Andrychowicz *et al.* [2020], who noticed on average small gains while using GAE. Our extended analysis showed performance increases independent from the applied advantage estimation technique. The decay of the learning rate reduces the step size in the policy parameter space towards the end of learning, eventually promoting the convergence to better local solutions. Reward scaling was found to be beneficial by Engstrom *et al.* [2020] but showed in our experiments only improvements in the locomotion tasks. We claim that reward scaling can be omitted when environments exhibit dense rewards with well-scaled magnitudes. An additional entropy bonus usage showed mixed outcomes, being preferable in environments with more unstable dynamics such as Hopper, Walker, and Pusher over tasks with more stable dynamics such as HalfCheetah and Ant. In contrast, the standardization of advantages yielded performance losses in all except one environment, which was not investigated but used by default in Engstrom *et al.* [2020]. Our extended analysis showed that advantage standardization yields profits when no reward scaling is applied but reduces performance when both are applied. We recommend to exclusively use the one or the other, where reward scaling is preferable. We noticed that gradient scaling hindered learning on average. Since Adam is in-variant to gradient magnitudes [Kingma and Ba, 2015], we assume that the non-linear operation of gradient scaling disturbs Adam's internal updates.

Our investigation of algorithmic ingredients showed that those are necessary to reach state-of-the-art results. In fact, variance reduction techniques for critic estimates deliver significant performance increases. A similar assertion can be made for trust-region enforcement methods, which helped to improve the policy performance and the robustness towards the hyper-parameter selection in all tasks, but had less effect on the learning performance than variance reduction techniques.

Regarding the structural ingredients, we did not find parameter-sharing useful similar to the results of Andrychowicz *et al.* [2020]. However, the choice of the optimizer and the weight initialization scheme had an impact, while other works reported no performance difference [Andrychowicz *et al.*, 2020]. Our experiments showed that RMSprop is more robust over a variety of hyper-parameters than Adam.

Overall, the preceding discussion points to a broader problem of RL algorithms: numerical sensitivity. Many ingredients aim to improve learning stability through standardized inputs and regression targets. Many of the discussed ingredients may be obsolete for algorithms if the environment is already well-specified in terms of input-output range and tailored to the peculiarities of neural networks. This suggests that environment specifics also matter and demands the practitioner's carefulness already during the problem design.

7 Conclusion

In this work, we took a step towards a transparent approach that investigates the inner workings of on-policy policy gradient algorithms. We studied algorithm ingredients in a modular and incremental approach and empirically assessed their contribution to the learning. We found that only a subset of such ingredients is necessary to achieve state-of-the-art results on contemporary benchmarks. Further, we confirmed that recent algorithmic advances such as the variance reduction methods of critic estimates are essential to obtain good policy performance. To this end, we identified a minimum setup of algorithm ingredients that can confront state-of-the-art RL algorithms while promising better reproducibility due to less algorithm complexity.

Our paper shows that simple algorithms can also perform well and may not be limited to the currently benchmark-driven development of new algorithms. For a steady progress, we suggest that new RL proposals should be assessed on a unified implementation with a modular structure. Otherwise, side-effects such as code-level ingredients may be accountable for the claimed performance gains. Further works on this topic may investigate off-policy algorithms and test a broader range of tasks in both continuous and discrete state spaces.

Acknowledgments

We thank Matthias Kissel and appreciate the support of the Federal Ministry of Education and Research who sponsored this project under the funding code 01IS17049 and the Deutsche Forschungsgemeinschaft (DFG) through TUM International Graduate School of Science and Engineering (IGSSE), GSC 81.

References

[Andrychowicz *et al.*, 2020] Marcin Andrychowicz, Anton Raichuk, Piotr Stanczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, L'eonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? a large-scale empirical study. *ArXiv*, abs/2006.05990, 2020.

[Arulkumaran *et al.*, 2017] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, Nov 2017.

[Coumans and Bai, 2016] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016. Accessed: 2021-05-23.

[Dhariwal *et al.*, 2017] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017. Accessed: 2021-05-30.

[Duan *et al.*, 2016] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, pages 1329–1338, 2016.

[Engstrom *et al.*, 2020] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020.

[Espeholt *et al.*, 2018] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1406–1415. PMLR, 2018.

[Henderson *et al.*, 2018] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 3207–3214. AAAI Press, 2018.

[Islam *et al.*, 2017] Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *CoRR*, abs/1708.04133, 2017.

[Kakade, 2002] Sham M Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems 14*, pages 1531–1538. MIT Press, 2002.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

[Mania *et al.*, 2018] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems 31*, pages 1800–1809. Curran Associates, Inc., 2018.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529 EP –, 02 2015.

[Peters and Schaal, 2008] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682 – 697, 2008. Robotics and Neuroscience.

[Rajeswaran *et al.*, 2017] Aravind Rajeswaran, Kendall Lowrey, Emanuel V. Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems 30*, pages 6550–6561. Curran Associates, Inc., 2017.

[Reda *et al.*, 2020] Daniele Reda, Tianxin Tao, and Michiel van de Panne. Learning to locomote: Understanding how environment design matters for deep reinforcement learning. In *Motion, Interaction and Games*, MIG '20, New York, NY, USA, 2020. Association for Computing Machinery.

[Rissanen, 1978] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465 – 471, 1978.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1889–1897. PMLR, 2015.

[Schulman *et al.*, 2016] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations*, 2016.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[Sigaud and Stulp, 2019] Olivier Sigaud and Freek Stulp. Policy search in continuous action domains: An overview. *Neural Networks*, 113:28 – 40, 2019.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[Tucker *et al.*, 2018] George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 5015–5024. PMLR, 2018.