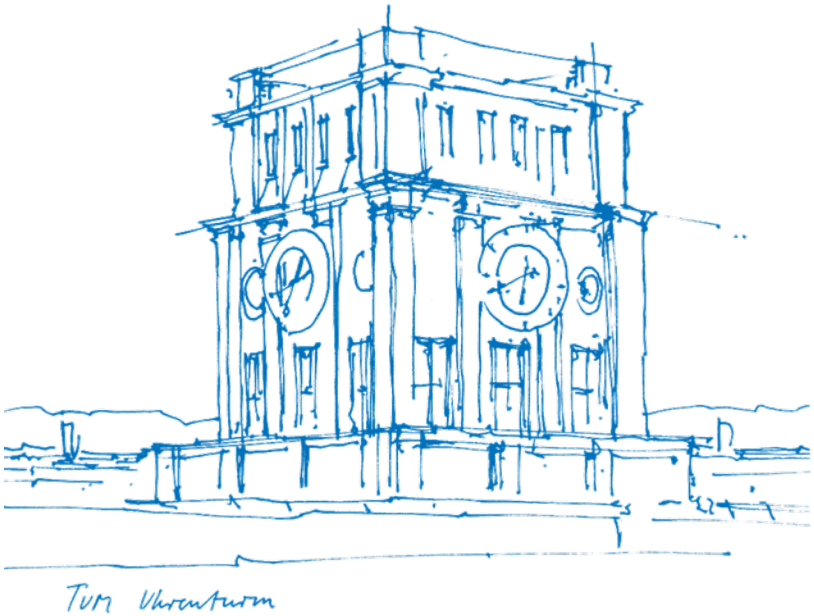
Proximal Policy Optimization Algorithms

Maximilian Stadler

Recent Trends in Automated Machine-Learning

Thursday 16th May, 2019

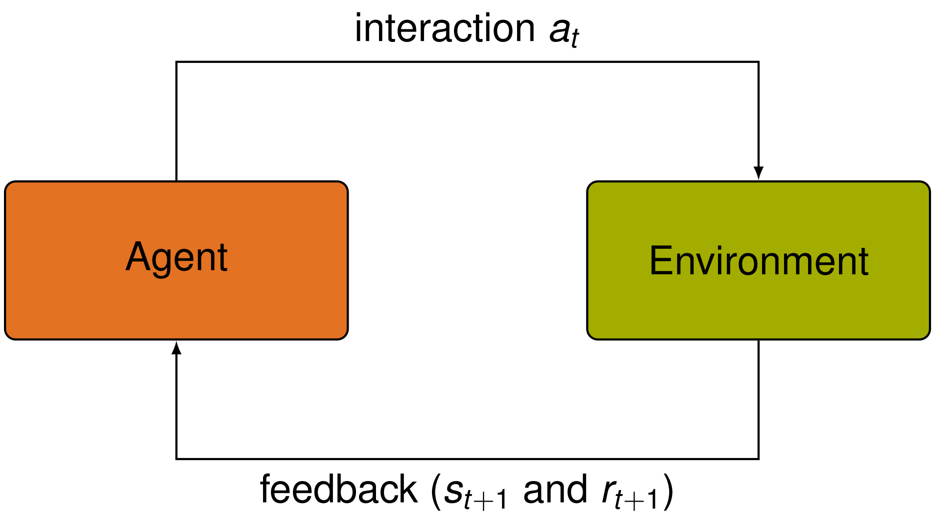


Reinforcement Learning

Definitions and Basic Principles

"Reinforcement learning is learning what to do — how to map situations to actions — so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them"

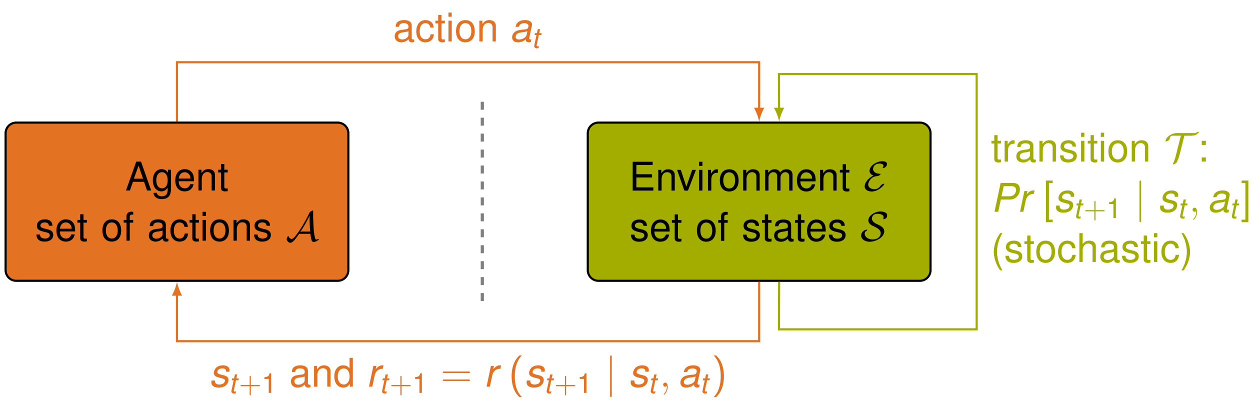
— Sutton and Barto, *Reinforcement Learning - An Introduction*, [SB18, p. 1]



Definitions and Basic Principles

Markov Decision Process (MDP)

- formalization for episodic tasks with final state at time T, i.e. $t < T$
- defined by tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r)$



- Markov Property: future depends only on present state, not on past
- trajectory τ: sequence of states and actions $(s_0, a_0, \ldots s_{T-1}, a_{T-1}, s_T)$
- extension to partially observable environments (POMDPs) and continuous settings often straightforward

Definitions and Basic Principles

Agent

- stochastic policy $\pi(a_t \mid s_t)$, i.e. given state s_t, agent takes action a_t with some probability
- Reinforcement: choose policy $\pi(a_t \mid s_t)$ such that expected reward $\mathbb{E}_{\pi,\tau}[R_t]$ is maximized
- Learning: assume parametric policy $\pi_\theta(a_t \mid s_t)$ and learn parameters θ

Rewards

- simplest measure: $R_\tau = \sum_{k=0}^{T-1} r_{k+1}$ (total accumulated reward)
- for long episodes, discount influence of future rewards

$$R_{\tau,\gamma} = \sum_{k=0}^{T-1} \gamma^k r_{k+1} \quad \text{with} \quad \gamma \in [0,1] \tag{1}$$

- better for learning: cumulative reward from action a_t onwards

$$R_{t,\gamma} = \sum_{k=t}^{T-1} \gamma^{k-t} r_{k+1} \quad \text{with} \quad \gamma \in [0,1] \tag{2}$$

- general performance measure G_t (e.g. $R_{t,\gamma}$)

Policy Gradient Methods

- define objective $\mathcal{L}_\theta = \mathbb{E}_{\pi_\theta,\tau}[G_t]$ and choose $\theta^* = \arg\max_\theta \mathcal{L}_\theta$
- Gradient Ascent: $\theta \leftarrow \theta + \alpha \nabla_\theta \mathcal{L}_\theta$

$$
\begin{aligned}
\nabla_\theta \mathcal{L}_\theta &= \nabla_\theta \mathbb{E}_{\pi_\theta(\tau),\tau}[G_t] = \nabla_\theta \int \pi_\theta(\tau) G_\tau d\tau = \\
&= \int \nabla_\theta \pi_\theta(\tau) G_\tau d\tau = \int \pi_\theta(\tau) \frac{1}{\pi_\theta(\tau)} \nabla_\theta \pi_\theta(\tau) G_\tau d\tau = \\
&= \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) G_\tau d\tau = \mathbb{E}_{\pi_\theta,\tau}[\nabla_\theta \log \pi_\theta(\tau) G_\tau] = \\
&= \mathbb{E}_{\pi_\theta,\tau}\left[\left(\sum_t \nabla_\theta \log \pi_\theta(a_t \mid s_t) G_t\right)\right]
\end{aligned}
\tag{3}
$$

- intuition: high performance $\Rightarrow$ increase (log)likelihood of action a_t
- intuition: low performance $\Rightarrow$ make action a_t less likely

REINFORCE [Wil92]

$$\underbrace{\Delta\theta}_{\substack{\text{REward} \\ \text{Increment}}} \Leftarrow \underbrace{\alpha}_{\substack{\text{Nonnegative} \\ \text{Factor}}} \times \underbrace{G_t}_{\substack{\text{Offset} \\ \text{Reinforcement}}} \times \underbrace{\nabla_\theta \log \pi_\theta (a_t \mid s_t)}_{\substack{\text{Characteristic} \\ \text{Eligibility}}} \tag{4}$$

Algorithm 1: REINFORCE, modified from [SB18]

for *iteration*=$1, 2, \ldots$ **do**
  run policy π_θ in environment for T timesteps
  to obtain trajectory $\{s_0, a_0, s_1, a_1, \ldots s_{T-1}, a_{T-1}, s_T\}$
  with rewards $\{r_1, \ldots r_T\}$
  for $t = 0, \ldots T-1$ **do**
    compute cumulative reward $G_{t,\gamma} = \sum_{k=t}^{T-1} \gamma^{k-t} r_{k+1}$
    $\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a_t \mid s_t) G_{t,\gamma}$
  end
end

REINFORCE

Advantages

- simple
- widely used in combination with stabilizing extensions (e.g. Neural Data Filter[Fan+17], Model-Agnostic ML [FAL17] or Neural Architecture Search [ZL16] and A3C [Mni+16])

Supervised Learning

- $\mathbb{E}_{x \sim p_{data}}[\mathcal{L}(x)] = \int_{x \sim p_{data}} p_{data}(x)\mathcal{L}_\theta(x)dx$
- with p_{data} fixed and $\mathcal{L}_\theta$ known

Policy Gradient Methods

- $\mathbb{E}_{\tau \sim \pi_\theta}[G(\tau)] = \int_{\tau \sim \pi_\theta} \pi_\theta(\tau)G_\tau d\tau$
- with π_θ being updated and dynamics of G_τ unknown

REINFORCE

Problems

- gradient updates might change π_θ (i.e. data distribution) such that the agent ends up in “useless” regions
- $|\theta| << |\mathcal{S}|$, i.e. changing one weight changes actions for many states
- sampled trajectory and rewards only valid on current policy (not on updated one)
- usually high variance by estimating gradient (instead of loss)

$\Rightarrow$ multiple updates on same trajectory might lead to destructively large updates

$\Rightarrow$ learning rate has to be chosen carefully

$\Rightarrow$ sample inefficiency

Proximal Policy Optimization Algorithms

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov

OpenAI

{joschu, filip, prafulla, alec, oleg}@openai.com

Proximal Policy Optimization

- builds up on work of Trust-Region Policy Optimization (TRPO) [Sch+15]
- mathematical derivation how to prohibit large deviations of policy π_θ from $\pi_{\theta_{old}}$
- penalize a large value of the KL-divergence $KL(\pi_{\theta_{old}}(\cdot \mid s_t) \parallel \pi_\theta(\cdot \mid s_t))$
- optimization should take place in “trust-region”

- PPO simplifies of TRPO-objective function heuristically
- idea: clip objective function whenever policy differs too much from old policy
- additional hyperparameter ε (clipping range)

$$\mathcal{L}^{CLIP}(\theta) = \mathbb{E}_t\left[\min\{\sigma_t G_t, clip\left(\sigma_t, 1-\varepsilon, 1+\varepsilon\right) G_t\}\right] \quad \text{with} \quad \sigma_t = \frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{old}}\left(a_t \mid s_t\right)} \tag{5}$$

Proximal Policy Optimization - Objective

$$\mathcal{L}^{CLIP}(\theta) = \mathbb{E}_t\left[\min\{\sigma_t G_t, clip\left(\sigma_t, 1-\varepsilon, 1+\varepsilon\right) G_t\}\right] \quad \text{with} \quad \sigma_t = \frac{\pi_\theta\left(a_t \mid s_t\right)}{\pi_{\theta_{old}}\left(a_t \mid s_t\right)} \tag{6}$$

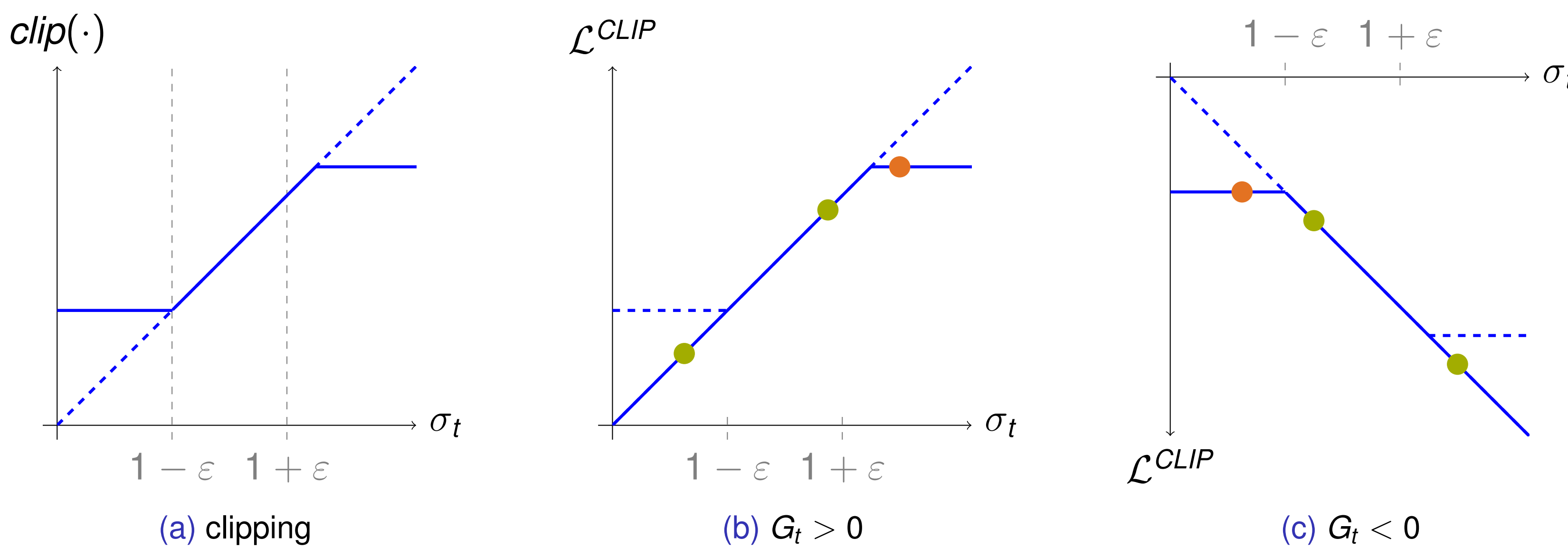


Figure: Visualization of the PPO-objective function, adopted examples from [Wil18] and images from [Sch+17b]

Proximal Policy Optimization - Algorithm

Algorithm 2: PPO, modified from [Sch+17b]

for *iteration*=$1, 2, \ldots$ **do**

  run policy $\pi_{\theta_{old}}$ in environment for T timesteps
  to obtain trajectory $\{s_0, a_0, \ldots s_{T-1}, a_{T-1}, s_T\}$
  with rewards $\{r_1, \ldots r_T\}$

  for t=$1, \ldots T$ **do**

    compute performance measure G_t

  end

  compute objective function $\mathcal{L}^{CLIP}$ by summing trajectories and averaging time-steps

  for *epoch in* $1, \ldots K$ **do**

    optimize surrogate $\mathcal{L}^{CLIP}(\theta)$ w.r.t. θ using mini-batches
    obtain θ by Gradient Ascent

  end

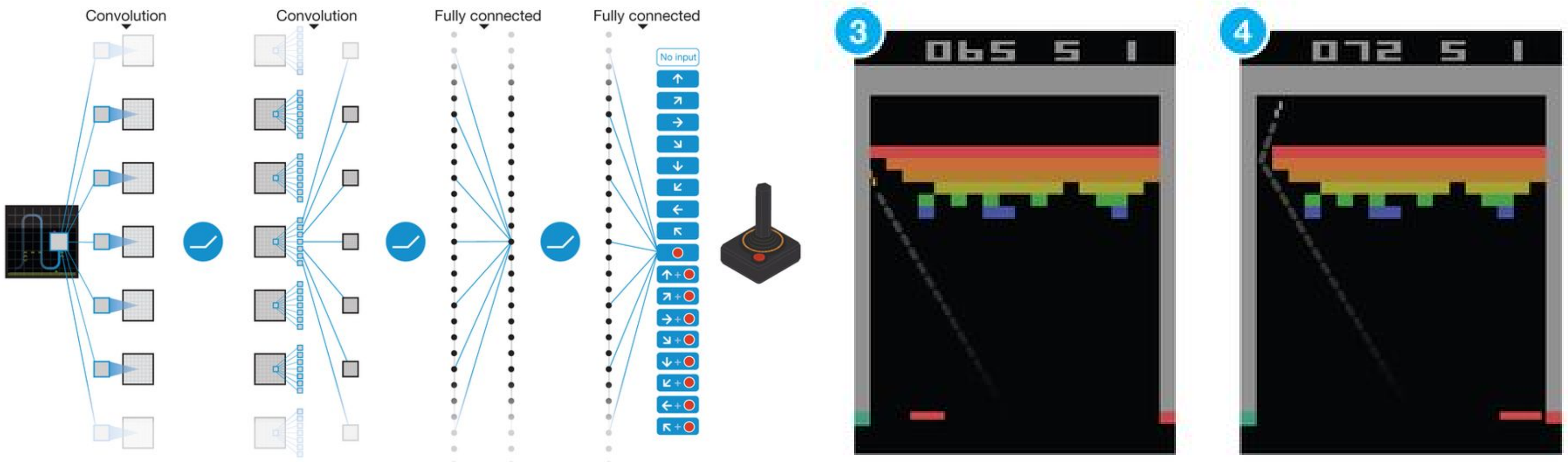
  $\theta_{old} \leftarrow \theta$

end

Proximal Policy Optimization - Experiments

Atari Domain (discrete control, episodic environment)

- experiments on 49 Atari games from the Arcade Learning Environment [Bel+13]
- comparison with other approaches (A2C [Mni+16] and ACER [Wan+16])



(a) Exemplary Architecture

(b) Atari Screenshots

Figure: Images from [Mni+15] (actually in context of Deep Q-Learning)

Proximal Policy Optimization - Experiments

Robotic Simulations (continuous domain and environment)

- locomotion experiments OpenAI Gym [Bro+16] within MuJoCo [TET12] environments
- comparison with other approaches (Vanilla PG, TRPO, A2C, evolutionary algorithms)

Figure: Example Videos from OpenAI-Blog [Sch+17a]

Proximal Policy Optimization - Experiments

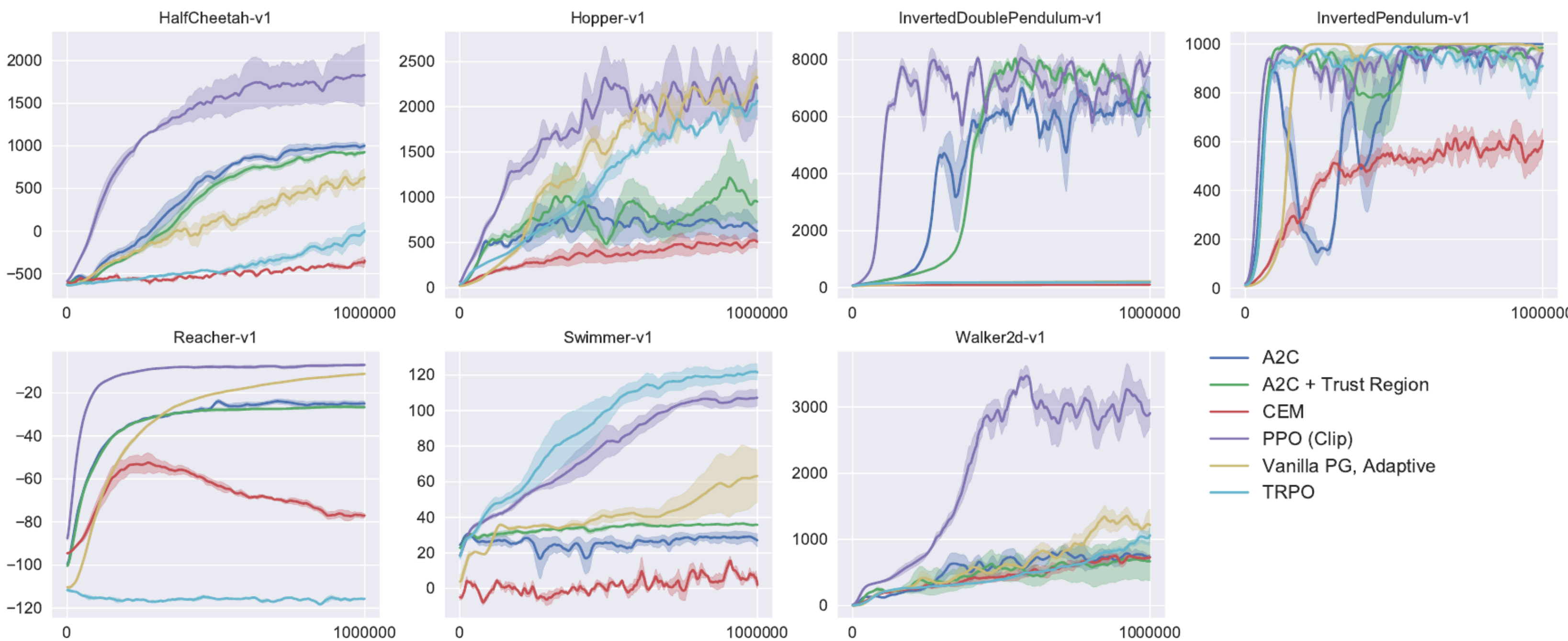


Figure: Comparison of approaches on seven MuJoCo environments, plots from [Sch+17b]

Proximal Policy Optimization - Advantages

- heuristic clipping ensures that updates take place in “trust-region”
- less variance than vanilla gradient methods
- policy π_θ does not change too drastically per update-step

$\Rightarrow$ allows for optimizing the objective $\mathcal{L}$ in multiple epochs on the same episodes/trajectories

- simple and straightforward implementation (e.g. with PyTorch or TensorFlow or)
- can be easily extended with other RL-techniques

- sample efficient (important for RL-based Meta-Learning)
- widely used (e.g. with AutoAugment [Cub+18], Searching for Activation Functions [RZL17], Learning Transferable Architectures [Zop+17])

How to further reduce variance?

- PPO provides a powerful improvement of policy gradients
- PPO might still suffer from a high gradient variance

- BUT: so far only $R_{t,\gamma}$ used for performance estimate
- What about parts of an environment where the rewards are very high or extremely negative?

How to further reduce variance

- What about parts of an environment where the rewards are very high or extremely negative?

- bring reward in some context by subtracting a baseline, i.e $G_t = R_{t,\gamma} - b_t$
- simple baseline: average reward
- advanced: find an approximation of how valuable certain states are

$\Rightarrow$ tell the agent (actor, i.e. policy) how good or bad certain states/actions are (critic)
$\Rightarrow$ (Advantage) Actor-Critic-Methods

Thank you for your attention!

References I

[Bel+13] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. “The arcade learning environment: An evaluation platform for general agents”. In: *Journal of Artificial Intelligence Research* 47 (2013), pp. 253–279.

[Bro+16] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. “OpenAI Gym”. In: (2016), pp. 1–4. arXiv: 1606.01540. URL: http://arxiv.org/abs/1606.01540.

[Cub+18] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. “AutoAugment: Learning Augmentation Policies from Data”. In: (2018). arXiv: 1805.09501.

[Dha+17] 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.

[FAL17] C. Finn, P. Abbeel, and S. Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks”. In: (2017). arXiv: 1703.03400.

[Fan+17] Y. Fan, F. Tian, T. Qin, J. Bian, and T.-Y. Liu. “Learning What Data to Learn”. In: (2017). arXiv: 1702.08635.

References II

[Lev18] S. Levine. *Policy Search (DLRL Summer School 2018 - Toronto)*. 2018. URL: http://videolectures.net/DLRLsummerschool2018%7B%5C_%7Dlevine%7B%5C_%7Dpolicy%7B%5C_%7Dsearch/ (visited on 05/09/2019).

[Mni+15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. "Human-level control through deep reinforcement learning.". In: *Nature* 518.7540 (2015), pp. 529–33. DOI: 10.1038/nature14236.

[Mni+16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. "Asynchronous Methods for Deep Reinforcement Learning". In: 48 (2016). arXiv: 1602.01783.

[RZL17] P. Ramachandran, B. Zoph, and Q. V. Le. "Searching for Activation Functions". In: (2017), pp. 1–13. arXiv: 1710.05941.

[SB18] R. S. Sutton and A. G. Barto. *Reinforcement Learning - An Introduction*. second edi. Cambridge, Massachusets: The MIT Press, 2018.

[Sch+15] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. "Trust Region Policy Optimization". In: (Feb. 2015). arXiv: 1502.05477.

References III

[Sch+17a] J. Schulman, O. Klimov, F. Wolski, P. Dhariwal, and A. Radfold. *Proximal Policy Optimization*. 2017. URL: https://openai.com/blog/openai-baselines-ppo/ (visited on 05/09/2019).

[Sch+17b] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. "Proximal Policy Optimization Algorithms". In: (2017), pp. 1–12. arXiv: 1707.06347.

[Sch16] J. Schulman. "Optimizing Expectations: From Deep Reinforcement Learning to Stochastic Computation Graphs". PhD thesis. University of California, Berkely, 2016.

[TET12] E. Todorov, T. Erez, and Y. Tassa. "Mujoco: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2012, pp. 5026–5033.

[Wan+16] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas. "Sample Efficient Actor-Critic with Experience Replay". In: 2016 (2016). arXiv: 1611.01224. URL: http://arxiv.org/abs/1611.01224.

[Wil18] C. M. Wild. *The Pursuit of (Robotic) Happiness: How TRPO and PPO Stabilize Policy Gradient Methods*. 2018. URL: https://towardsdatascience.com/the-pursuit-of-robotic-happiness-how-trpo-and-ppo-stabilize-policy-gradient-methods-545784094e3b (visited on 05/09/2019).

References IV

[Wil92] R. J. Williams. “Simple statistical gradient-following algorithms for connectionist reinforcement learning”. In: *Machine Learning* 8.3-4 (1992), pp. 229–256.

[ZL16] B. Zoph and Q. V. Le. “Neural Architecture Search with Reinforcement Learning”. In: (2016), pp. 1–16. arXiv: 1611.01578.

[Zop+17] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. “Learning Transferable Architectures for Scalable Image Recognition”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (July 2017), pp. 8697–8710. arXiv: 1707.07012.