REPRODUCTION OF THE PAPER "SOFT Q-LEARNING WITH MUTUAL-INFORMATION REGULARIZATION" ICLR 2019 REPRODUCIBILITY CHALLENGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Reproducibility is one of the pillars of scientific research. Reproducing a paper can sometimes be a challenging task, and so far no well-established reproducibility standard exists. To make a step towards more reproducibility in the field of Machine Learning, Joelle Pineau and several other scientists started the ICLR Reproducibility Challenge, where the papers submitted to the ICLR conference are free to be reproduced by any team of scientists or students. This report is a summary of our contribution to the second edition of this challenge, in which we detail the challenges that came with reproducing a paper with no attached codebase.

1 INTRODUCTION

For our participation to this challenge, we chose to reproduce the paper named *Soft Q-Learning with Mutual-Information Regularization* (Anonymous, 2019), because it seemed to bring information theory along with approximate reinforcement learning, which we found interesting, both for the subjects themselves and the opportunity to learn more about them. The goal of this work is to find how difficult it is to reproduce the findings of this paper. The report is split in four sections, first a short background on the reproduced paper, second the details of our reproduction methodology, and finally the reproduction results and conclusions. The repository containing the reproduction code is located at `https://github.com/lcalem/reproduction-soft-qlearning-mutual-information`.

2 REPRODUCED PAPER

The reproduced paper, "Soft Q-learning with Mutual Information Regularization" (Anonymous, 2019), introduces a novel form of regularization. Regularization is a technique that aims to control the solution's complexity. In our case, the regularization method is a penalty term added to the formulation of the value function, which constrains the form of the policy. Entropy regularization is a common form of regularization used in several state-of-the-art methods (Haarnoja et al., 2017; Teh et al., 2017) that puts an additional probability mass on each action indistinctly, which can be a drawback in cases where actions have widely varying consequences. This observation lead the authors to present their novel form of regularization: Mutual Information Regularization.

The paper proposes an algorithm, Mutual Information Reinforcement Learning (MIRL), as a practical implementation of an off-policy, soft-q learning algorithm that uses this Mutual Information Regularizer.

This section gives an overview of the notions used in the paper, to introduce the relevant information to understand the reproduction. This is by no means a full standalone summary of the paper, and to fully grasp the notions presented here we encourage the reader to refer to the original paper (Anonymous, 2019).

2.1 BACKGROUND

Reinforcement Learning In this paper, we consider a Markov Decision Process (MDP) as a tuple (S, A, P, γ), where S is the set of states, A the set of actions, P the transition function and γ

the discount factor. The common reinforcement learning notion of a policy π is also used, which represents the probability for each action in each state.

Entropy Regularization Maximum entropy regularization is a common form of regularization that promotes policies with a high entropy, which tend to spread probability mass across actions more equally. To do so, entropy regularization adds to the standard reward objective a regularization term:

$$\mathcal{V}^*(s) = \max_{\pi} \mathbb{E}\Big[\sum_{t=0}^{T} \big(r(s_t, a_t) - \frac{1}{\beta} \log \pi(a_t|s_t)\big)\Big] \tag{1}$$

In this equation, r represents the reward, which is the standard RL reward objective part, and the $\log$ part represents the entropy regularizing part, with the parameter β controlling the trade-off between reward and entropy. As β goes to infinity, we retrieve the standard RL objective.

The policy that solves this is a softmax policy of the form:

$$\pi^*(a|s) = \frac{e^{\beta Q^*(s,a)}}{\sum_{a \in A} e^{\beta Q^*(s,a)}} \tag{2}$$

Where Q^* is the optimal action-value function. Adding an entropy regularizer helps with exploration, and produces more robust policies (Grau-Moya et al., 2016; Haarnoja et al., 2017).

Soft Q-learning Soft Q-learning is a notion closely linked to entropy regularization. It has been introduced under many forms (Schulman et al., 2017; Leibfried et al., 2017) with the idea of replacing the hard max of the q-learning objective with a weighted sum over all actions, using a distribution ρ as a prior over all actions, used to weight their values:

$$(T^{\rho}_{\text{soft}} Q)(s, a, s') = r(s, a) + \gamma \frac{1}{\beta} \log \sum_{a'} \rho(a') \exp(\beta Q(s', a')) \tag{3}$$

This soft operator is used in the update of the Q-function, either in the tabular Q-value update of equation 4 or in the loss in the approximate setting (equation 5).

$$Q(s, a) \leftarrow Q(s, a) + \alpha_Q((T^{\rho}_{\text{soft}} Q)(s, a, s') - Q(s, a)) \tag{4}$$

$$L(\theta, \rho) = \mathbb{E}_{s,a,r,s' \sim M}\left[\Big((T^{\rho}_{\text{soft}} Q_{\bar{\theta}})(s, a, s') - Q_{\theta}(s, a)\Big)^2\right] \tag{5}$$

Where γ is the discount factor, $\mathcal{M}$ is the replay memory from which we draw samples. θ refers to the parameters of the network estimating Q and $\bar{\theta}$ to a set of parameters used in the target network, that are frozen and regularly copied from θ.

2.2 CONTRIBUTIONS

Relevant previous work (Haarnoja et al., 2017; Levine, 2018) to this article introduce the idea of entropy-regularized RL via an entropy term. In maximum entropy regularization, this term is uniform, but many methods exist, such as a regularizer taking a shape that depends on the Kullback-Leibler divergence between the prior over the actions and the current policy. In this paper, the authors introduce a new regularizer based on mutual information. This regularizer is learned during training and the learning procedure is also provided.

Mutual Information Mutual information is a notion coming from Information Theory that measures how two random variables are dependent, in the sense of knowing how much information we gain for variable A, having an observation of variable B. The proposed regularizer of this paper is based on the mutual information between an explicit adaptive prior over the action space and the current policy. This regularizer forces the policy to be a non-uniform prior that assigns higher probability masses to most used actions. It can be seen as pushing the policy to go in states that are more likely to actually be visited.

Learning the prior This adaptive prior ρ will be used in the optimization procedure for the Q function and to weight the policy. One of the main contributions of this paper is the algorithm for learning this prior ρ, which is derived from mutual information and a variational inference formulation of the RL problem, outlined in section 3 of the original paper.

Using the prior This learned prior is used during learning for the q-function update (see 4 and 5), and also for the behavioural policy. This policy works the same way as an epsilon-greedy policy. Given a random sample $u \sim$ Uniform[0, 1] and a given epsilon ϵ, the selected action a_i for timestep i is obtained by:

$$a_i = \begin{cases} \text{argmax}_a \pi_i(a|s_i) & \text{if } u \geq \epsilon \\ a \sim \rho_i(\cdot) & \text{if } u \leq \epsilon \end{cases} \tag{6}$$

Where the prior ρ_i is also used in the policy:

$$\pi_i(a|s) = \frac{1}{Z} \rho_i(a) \exp(\beta_i Q_i(s, a)) \tag{7}$$

Where Z plays the role of a partition function and is defined by:

$$Z = \sum_a \rho(a) \exp(\beta Q(s, a)) \tag{8}$$

This is detailed in section 3.2 of the original paper where the method for obtaining this optimal policy is detailed. We outlined the key equations here that are used in the reproduction work.

3 Reproduction details

3.1 Methodology

Reproducing a paper is a challenging task. In Computer Science related fields, where the findings are obtained by coding the proposed method, access to the source is the best possible setting. When source code is not available, the reproduction is based on the explanations provided in the paper. For our paper, no implementation was available so our first challenge was to decide how to build our codebase.

Re-implementing the method from scratch didn't seem a practical path at first so we tried to find a similar paper with open source code and build from that codebase.

For this reproduction, we found the paper (Haarnoja et al., 2017) which introduced soft q-learning, a method on which this paper is based. The paper has associated open source code [1] and we intended to build on this codebase.

Unfortunately, the continuing setting of (Haarnoja et al., 2017) is quite different from the experiments of the reproduced paper and it seemed easier and clearer to start from scratch with a simple Deep Q Network (DQN).

To ensure our code is working as intended, we first test it on Minigrid [2], an additional environment for OpenAI gym (Brockman et al., 2016), representing various simple gridworlds. Learning on a

[1] https://github.com/haarnoja/softqlearning

[2] https://github.com/maximecb/gym-minigrid

smaller environment allows to test whether our algorithms work in an environment that is faster to train. The initial plan was to make our implementation work on a simple gridworld, and then scale it to the Atari benchmark, which was the only experiment in the initial version of the paper.

When experiments in the tabular setting on gridworlds were introduced in the paper, we decided to focus on the reproduction of the tabular setting first.

3.2 IMPLEMENTATION

3.2.1 APPROXIMATE SETTING

Since the first version of the paper only included the Atari experiments, we started by implementing the algorithm from scratch in python using the Tensorflow library (Abadi et al., 2015). The reproduction code is built in an incremental way: we first code a simple DQN (Mnih et al., 2015), leveraging the simplicity of this first approach to architecture the code as simple as possible, to facilitate reading.

We implement the classical DQN stability improvements, such as using an experience replay and a target network with parameters $\bar{\theta}$ copied regularly from the parameters θ of the q estimator network. The number of timesteps between these copies is an hyperparameter, given in the original paper as 10000.

We then transform this Q Learning in a Soft Q Learning algorithm by replacing the q-learning argmax by the soft q-learning defined in 3. In this step we start including the prior ρ which is for now uniform.

The last step was to implement the β update and the prior ρ update using the equations provided in section 4 of the paper:

$$\rho_{i+1}(a) = (1 - \alpha_\rho)\rho_i(a) + \alpha_\rho \pi_i(a|s_i) \tag{9}$$

$$\beta_{i+1} = (1 - \alpha_\beta)\beta_i + \alpha_\beta(\frac{1}{L(\theta_i, \rho_{i+1})}) \tag{10}$$

Where α_ρ and α_β are the learning rates given in the paper (respectively $2 \cdot 10^{-6}$ and $3.3 \cdot 10^{-6}$), π_i is the policy defined in equation 7, and $L(\theta_i, \rho_{i+1})$ is the loss defined in equation 5.

This incremental way of coding allowed us to separate agents to better test the results for each agent separately. All the approximate experiments code is in the `approximate` subfolder of our implementation repository [3].

3.2.2 TABULAR SETTING

The tabular gridworld experiments were introduced later in the original paper and we put our reproduction efforts in the `tabular` subfolder of our repository.

We followed an incremental pattern similar to the one used for the approximate setting reproduction. It was directly useful since the experiments in the tabular case were comparing four algorithms: regular q-learning; then soft q-learning (SQL), in which we replace the max by the soft operator (equation 3) with a uniform ρ prior; then SQL with the marginal distribution (SQL_m) of actions instead of the uniform prior; and finally the MIRL implementation, with the ρ learning.

Since the gridworld experiments were using an unknown gridworld benchmark, we implemented a generic gridworld environment that could create any gridworld based on a configuration specifying the size, the location of the walls and the goal. This benchmark is located in `tabular/environment.py`.

[3] https://github.com/lcalem/reproduction-soft-qlearning-mutual-information

4 RESULTS

The first step of our reproduction work was to understand very well all the mathematical concepts involved. We spent time making sure we understood all the mathematical foundations of the paper. Some of the concepts mentioned in the paper gave rise to a blogpost [4].

4.1 ATARI

After this initial understanding step, for which we found the material presented in the paper was a good summary, we starting reproducing the approximate experiments following the incremental scheme described in section 3.2. We started with the approximate case because at the time of the reproduction, only the approximate experiments were described in the paper.

Unfortunately, we couldn't reproduce the method on the Atari experiments, because of numerical instabilities that kept making our code crash. Moreover we didn't apply to GPU credit at first because we had access to a GPU, but we found the access to be harder than originally thought due to restrictions on the docker images we could run. We managed to train on a simple gridworld but the results were unconclusive, the network seemingly forgetting the action every 50k timesteps.

The original submission was then updated with the tabular experiments, and we decided to try and reproduce these before the Atari experiments, which we ended up having no time left for.

4.2 TABULAR CASE

The initial task of creating the gridworld environment was completed rapidly, and raised our first questions: what is the reward schemed for the tabular experiments? Are they the same for the two griworlds? No explicit values are mentioned in the paper, but judging by the reward values in the figure 1 of the paper, we went with a reward of -1 for every step except if the agent reaches the goal, in which case we gave a +15 reward. After our first contact with the authors, we implemented the actual scheme they used in the experiments, which is -1/+9.

We tested our environment with a random agent and then started by implementing q learning, then using the soft operator (equation 3) and a uniform ρ prior to create the SQL agent, and then maintaining the marginal action distribution to implement the SQL_m agent.

For these implementations, we tested on $1 \cdot 10^4$ steps first to tackle small bugs and see if learning was happening. We had good results for these three baseline algorithms, consistent with what the paper is showing in figure 1.

We then introduced the ρ update step and produced the MIRL implementation. When tested on $1 \cdot 10^4$ steps we found that it was doing worse than the other implementations (the learning was slower than expected on the corridor gridworld, and on the second one a drop in learning started at around 2000 environment interactions). The close-up results for $1 \cdot 10^4$ steps can be found in figure 1. We checked our implementation of the update equations provided in the paper an still didn't find the errors, but we will continue to investigate with the help of the original authors to find the adequate implementation.

Since the experiment results shown in figure 1 of the original article were produced using around $1 \cdot 10^5$ environment interactions, we then ran our implementation for longer than the initial $1 \cdot 10^4$ testing steps. Q Learning went until $1 \cdot 10^5$ without complications, but at around $6 \cdot 10^4$ timesteps, SQL, SQL_m and MIRL were crashing due to numerical under an overflow. After investigation, we found that the value of β, which in the tabular setting is updated at each time step using $\beta_{i+1} = c \cdot i$, becomes big enough so that around $6 \cdot 10^4$ timesteps, when it is multiplied by the Q value in $\exp(\beta Q(s, a))$, the product becomes too big or too small (the Q value can be negative), resulting in over/underflow. This $\exp(\beta Q(s, a))$ pattern arises in the soft operator (equation 3), and the policy (equation 7). To counteract this effect and avoid crashes during learning, we bounded the $\beta Q(s, a)$ product by $[-700, +700]$. This bound is arbitrary and just enough to prevent the original errors, and no details about numerical instability handling were present in the paper. Our implementation might be incorrectly giving rise to these errors.

[4] https://lcalem.github.io/blog/2018/10/17/mutual-information

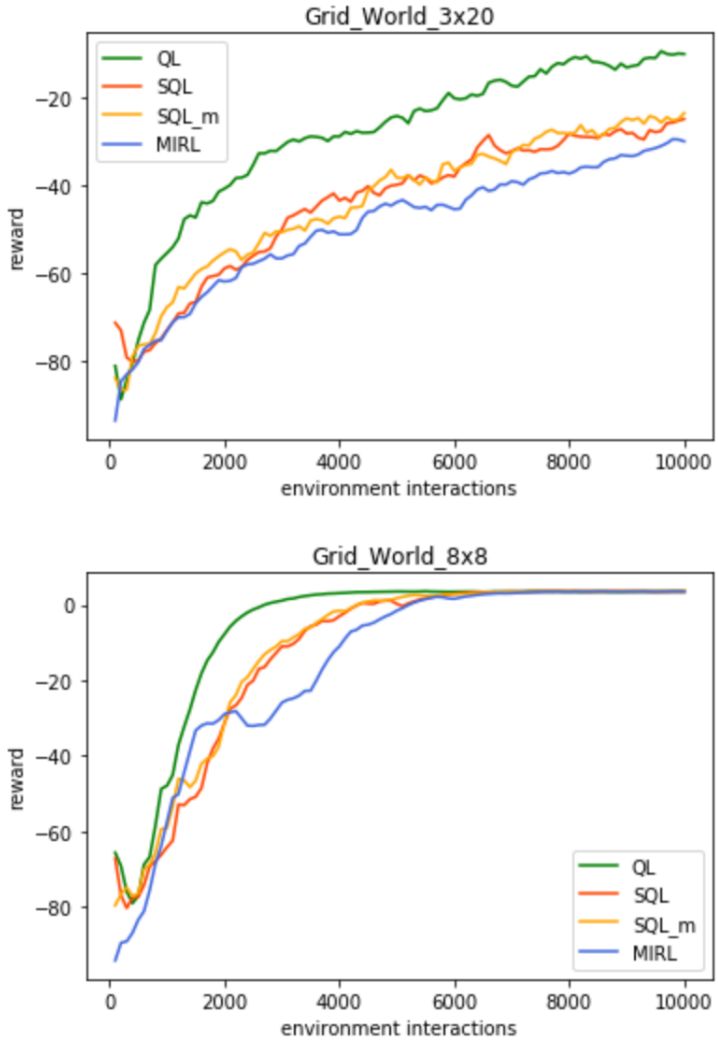

Figure 1: Tabular results for $1 \cdot 10^4$ environment interactions

Unfortunately, even without crashes, we found that performance is degraded at higher timesteps (figure 2). After investigation, we found that this degradation is due to our bound, because when the bound is tighter (we tested with $[-300; +300]$, $[-500; +500]$ and $[-700; +700]$), the results are degrading earlier. After further investigation, we didn't find any solution allowing the training in higher timesteps without crashing while not degrading performance, and we didn't want to add more differences between our implementation and the algorithm described in the paper.

It is interesting to note that Q Learning is not affected by these instabilities since it does not use the $\beta Q(s, a)$ product in any of its equations. The policy used for the Q Learning experiment is the baseline epsilon-greedy one.

The last column of figure 1 in the original paper is a plot describing the correct action taken by the agent over time. In our results (figure 2), the top plot is for the 3x20 corridor gridworld, where we find that the results match the original paper's results until about 40k timesteps, where numerical instability issues arise. We see that MIRL finds the correct action faster and the relative ordering of all the algorithms is respected. The bottom plot is for the 8x8 gridworld, for which the global shape of the curve is the same as the original paper. Variance can appear different since we didn't know which smoothing the authors used to create the original figure.

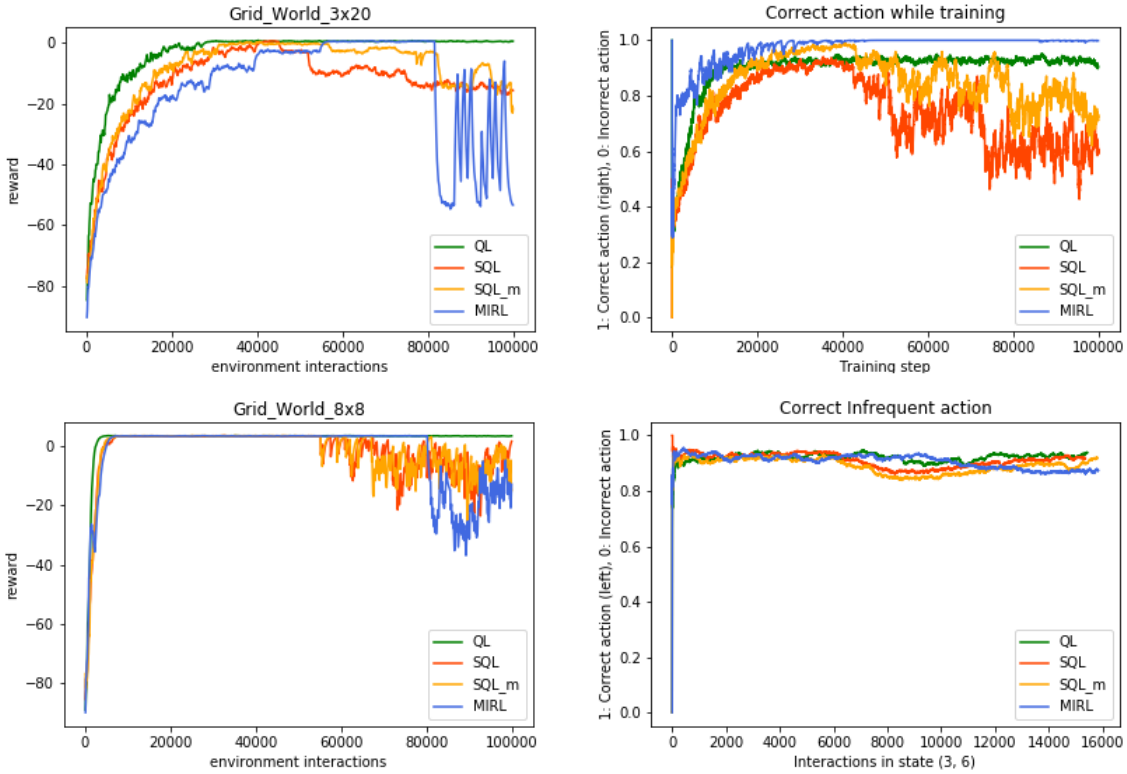

Figure 2: Results for tabular case over 4 algorithms. QL is Q Learning, SQL is Soft Q Learning, which uses a uniform prior, SQL_m is Soft Q Learning with the marginal action distribution as a prior, and MIRL is the Mutual Information Reinforcement Learning algorithm proposed in the original paper. The first line represents the experiments on the corridor 3x20 gridworld, and the second line the 8x8 gridworld with an important action (left) to be executed exactly once to reach the goal. The first column describes reward over time for each algorithm, and the last column an indication of the correct action taken by each algorithm over time. Due to implementation approximations, our reproduction couldn't match the performance of MIRL in the original paper, and numerical instabilities arise at higher timesteps.

4.3 Main difficulties

Besides the initial time spent to find a good baseline to build upon and finally deciding to start from scratch, our main difficulties to reproduce the paper can be regrouped in two parts: numerical instability and implementation details.

Numerical instabilities were the main difficulty, making our Atari experiments fail and giving no reproduction results. For the tabular case, training for higher timesteps is impaired by under and overflow in the exponential. Setting a bound to avoid this problem creates another one, impairing the algorithm's performance, as described in section 4.2.

Implementation details are what makes an implementation possible. The paper has an hyperparameter table, which is invaluable for reproduction, and has details in the body of the article, like a complete explanation of the evaluation procedure used to produce the results shown in the figures. But detailing everything can be very challenging, especially given the limited space of an article and the difficulty of wording details properly. We had numerous questions that fortunately the authors were able to answer when we asked them. These implementation details answers are summarized in the following section.

Reward scheme and initialization The reward scheme is the same for both gridworlds, with -1 given for each timestep except the goal step where a reward of +9 is given. The initial state of the environment involves a random location for the agent, as opposed to a fixed starting state.

Epsilon value The epsilon value is detailed in the paper for the evaluation (0.05), but the training value is missing. The authors informed us this value was set to 0.1 during training.

Plot smoothing The initial plots produced with training results had a very high variance compared to the smoothness exhibited by the plots in the middle column of figure 1 of the original paper. We asked the authors if some kind of smoothing were used and they answered that an exponential moving average with window 10 was used. We didn't ask about smoothing for the last column of figure 1 so we used the same smoothing scheme, which resulted in a completely non-understandable figure. We used a window value of 1000 to produce the plots for the correct action in figure 2.

Evaluation method The evaluation method is very well detailed in the paper, and we were missing the exact way of doing the average of the reward over the 10 seed times 30 episodes runs. The answer is an average of an average.

SQL_m experiment In order to avoid any confusion, we asked if the authors implemented the SQL_m experiment using a separate action table to create the marginal action distribution. The answer is yes.

Our implementation doesn't match the performance of the original paper, and we believe this is due to approximations we made because information wasn't available in the paper and we couldn't pinpoint the exact questions to ask the authors. Such an approximation we made is the clipping of the $\beta Q(s, a)$ product, which is not mentioned in the paper.

5 Conclusion

Reproducing a paper without an available codebase can be challenging, because numerous details are present in the code that cannot be described practically in the body of the article. The high level description of the algorithm in the original paper is well written for understanding the concepts and methods, but some implementation details are missing, which are unfortunately crucial to reproduce the findings of the paper. The authors were very quick and available to answer our questions, but we weren't able to find an implementation matching their results in time.

Even if reward performance was not matched, we found similar results for the correct action plots, in terms of order of performance of the various tested algorithms. It nevertheless has been an interesting experience, in which we learned a lot.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Anonymous. Soft q-learning with mutual-information regularization. In *Submitted to International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HyEtjoCqFX`. under review.

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

Jordi Grau-Moya, Felix Leibfried, Tim Genewein, and Daniel A. Braun. Planning with information-processing constraints and model uncertainty in markov decision processes. *CoRR*, abs/1604.02080, 2016. URL http://arxiv.org/abs/1604.02080.

Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. *CoRR*, abs/1702.08165, 2017. URL http://arxiv.org/abs/1702.08165.

Felix Leibfried, Jordi Grau-Moya, and Haitham Bou-Ammar. An information-theoretic optimality principle for deep reinforcement learning. *CoRR*, abs/1708.01867, 2017. URL http://arxiv.org/abs/1708.01867.

Sergey Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *CoRR*, abs/1805.00909, 2018. URL http://arxiv.org/abs/1805.00909.

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*, 02 2015. URL https://doi.org/10.1038/nature14236.

John Schulman, Pieter Abbeel, and Xi Chen. Equivalence between policy gradients and soft q-learning. 04 2017.

Yee Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 4496–4506. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7036-distral-robust-multitask-reinforcement-learning.pdf.