Interactive Segmentation using Object Singulation

Patrick Pfreundschuh
Autonomous Systems Lab
ETH Zuerich
patripfr@ethz.ch

Simon Schaefer
Autonomous Systems Lab
ETH Zuerich
sischaef@ethz.ch

Michel Breyer
Autonomous Systems Lab
ETH Zuerich

Fadri Furrer
Autonomous Systems Lab
ETH Zuerich

Tonci Novkovic
Autonomous Systems Lab
ETH Zuerich

Abstract: An accurate segmentation is crucial for performing manipulation tasks in real-world environments, but especially in cluttered environments is still poses a problem as adjacent objects are segmented together. This problem can be solved by enhancing visual with interactive segmentation, using push movements to singulate objects from each other. Current learning based approaches for this task use Deep Q-Learning (DQL) with a discretized action space. As using continuous actions instead would allow more complex and more efficient pushing movements, in this paper it was evaluated whether Proximal Policy Optimization (PPO) can also solve the task of object singulation. It is shown that with PPO only simple static scenes can be solved with the proposed setup, while in contrast also simple dynamic scenes could be solved using DQL.

Keywords: CORL, Robots, Learning

1 Introduction

Detection of objects is of high importance for applications where robots are supposed to interact with the environment. Often robots are used to sort or arrange objects like in several household tasks or in general in pick and place tasks. To do so successfully, in most cases visual segmentation is used, which works already well for structured environments, where the objects can be easily identified. However, handling unstructured environments that have to be manipulated is a major challenge in robotics, since often objects can only be viewed partially due to occlusion, which makes segmentation hard and thus complicates grasping of objects.
In order to simplify segmentation and picking under these conditions objects can be segmented interactively, e.g. the objects can be singulated, using push movements. However, efficient and precise object singulation still poses a problem, especially as a-priori the number of objects is unknown. Therefore, the robot has to explore, if a segmented area really consists of a single object, or if it might be multiple objects side by side. Furthermore, in applications where the robot should interact with objects that were unseen before, interactive segmentation can help to reason about the new and unknown object.
While previous approaches tackle the problem of object singulation by discretizing the state space, only allowing a limited number of actions, within this project we want to explore the feasibility of a continuous approach for object singulation.

The main contributions of this paper are:

- A new reinforcement learning task design for continuous object singulation
- Object singulation for a simple static task using a continuous policy
- Object singulation for simple dynamic tasks using Deep Q-Learning
- Illustration of pitfalls using unimodal policies for object singulation tasks

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

2 Related Work

The problem of singulating objects can be tackled in various different ways in terms of manipulation techniques and policies. While some approaches use a robotic hand that only pushes objects in a two-dimensional manner [1], others also use grasping for singulation [2]. Clearly, grasping can facilitate the task, especially when it comes to objects that were segmented such that the agent is not sure whether it is a single or multiple objects. In the case of multiple objects side by side, the robot will with high chance only grasp one of the objects, while the other one remains unchanged. With pure pushing of objects, the appearance of multiple objects side by side might remain unchanged, depending on the direction they are pushed to. Hence, both the starting position as well as the pushing sequence itself are crucial for the success of the overall action, while for grasping dominantly the choice of starting point is important. However, physically executing a pushing action is a lot more straightforward and extendable to a bunch of different actuators compared to grasping. Therefore, allowing pushing actions only is a broad research area.
Work can furthermore be split into approaches using hand-tuned deterministic policies or learning based approaches. Zeng et al. et al. [3] have already successfully shown, that deep neural networks are capable of learning the discussed task using grasping and pushing, using two large 121-DenseNets in parallel, one predicting the Q values for pushing and one for grasping state-action-pairs. Eitel et al. et al. [4] also showed that a network can learn to solve the problem using pushing only, by training a convolutional neural network which ranks certain pushing action by their priority. However, both approaches discretize the action space and use DQL, limiting the range of possible actions. To the best of our knowledge we are the first ones combining deep reinforcement learning with a continuous, pushing-only action space for the object singulation task.
There are several deep reinforcement learning algorithms with continuous action space, such as off-policy and Q-value based methods as Deep Deterministic Policy Gradient [5] or on-policy advantage-based methods such as TRPO [6], A3C [7] or PPO [8], whereby on-policy methods usually are more sample-efficient. While Henderson et al. et al. [9] state Trust-Region-Policy-Optimization (TRPO) and PPO to be the most stable model-free policy gradient methods (i.e. having the smallest confidence bounds in reproduction) Dhariwal et al. et al. [10] show that PPO outperforms TRPO in most common testing environments such as HalfCheetah, Walker2d and InvertedDoublePendulum and is easier to tune because of having less tuning parameters. PPO maximizes a surrogate advantage, estimated as $E[min(\rho_t(\theta)A_t, clip(\rho_t(\theta), 1-\epsilon, 1+\epsilon)A_t]$ which $A_t = R_t - V(s_t)$ being the advantage function and $\rho_t(\theta) = \pi(a_t|s_t, \theta)/\pi(a_t|s_t, \theta_{old}$ is the likelihood ratio of the action at timestep t between the updated and sampled policies, clipped using the hyperparameter ϵ.

3 Method

In this project, a manipulator should learn to singulate objects using push movements only. An RGD-D camera is used to perceive the environment. During this project, training was performed in a simulation which is further described in Section 4. The infinite set of possible states, continuous actions as well as hardly predictable state transitions exacerbate deterministic modeling. Therefore a reward-based learning approach is used. Since a continuous policy was desired, PPO [8] was chosen as reinforcement learning algorithm, as it has shown to be very successful for tasks, is easy to tune and a hence a favorable tradeoff between sample-complexity, simplicity, and wall-time as described in Section 2, combined with a deep fully connected network.

3.1 Image Preprocessing

Images are taken in a bird-eye view from above the object plane, since this minimizes occlusion. The depth channel of the RGB-D image is used to remove background from the objects, since only pixels with objects incorporate relevant information. A 64x64 px binary mask is then created, indicating where objects are located. From the binary mask, a distance map is calculated. This map shows the distance (in pixel) to the closest object for each pixel. Pixels inside objects are set to -1.

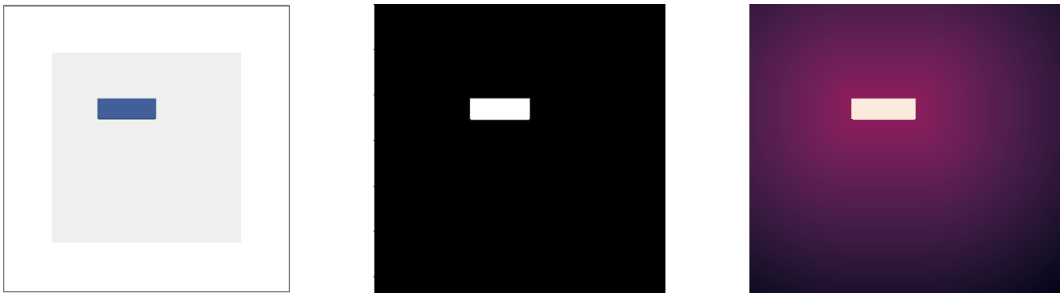

Figure 1: Different representations for a scenario where 3 quadratic blocks are placed next to each other. From left to right: RGB image observation, binary mask, distance map

3.2 Task

Since the experiment scenarios used in this project can be solved with one push, one episode consists of one step, where one step corresponds to one push movement. The push movement is realized with a simulated gripper that performs a straight line motion, which is explained in more detail in Section 4.

The action space is heavily biased on non-successful actions, not touching any object. Choosing a starting point close to an object leads to a successful pushing action with higher probability than far from the object. Therefore the learning task was implemented with the goal to lead the agent to choose actions close to objects

3.2.1 State

As input state, the 64x64 px distance map described in Fig. 3.1 is used. This way, possible good starting points are already indicated by low distance values in the input map and good pushing directions can be obtained following the distance gradient at every position of the map.

3.2.2 Action

The action consists of a point in the workspace and one of the four cardinal directions. The agent then performs a straight line push from this point in the chosen direction. A fixed length of one quarter of the workspace was chosen to reduce the complexity.
Action $a = (x, y, c)$ with $x, y \in [-l, l], c \in \{up, down, left, right\}$ where l corresponds to half of the length of the workspace.

3.2.3 Reward

Since we want to singulate the objects from each other, a metric to maximize is the inter-object distance d_{IO}. The inter-object distance is defined as

$$d_{IO} = \sum_{i,j \in C} d_{i,j} \tag{1}$$

where C is the set of all combinations of two objects, i.e $C = \binom{O_{seg}}{2}$ with O_{seg} being the set of all segmented objects and $d_{i,j}$ is the distance between object i and j. Since the objects only have to be separated from each other by a fixed distance threshold d_{goal}, we are using the truncated inter-object distance d_{IOT} defined as:

$$d_{IOT} = \sum_{i,j \in C} min(d_{i,j}, d_{goal}) \tag{2}$$

$$r_{Dist_t} = d_{IOT_t} - d_{IOT_{t-1}} \tag{3}$$

This way, the agent receives a reward if the truncated inter-object distance is increased, and is punished if it is decreased. If two objects that were segmented as one before become singulated, this

will result in new summands in d_{IOT_t} since O_{seg_t} will be bigger than $O_{seg_{t-1}}$.
To set an incentive to start close to objects, a second reward r_{Start} is added. Starting far away very often results in a non-rewarding pushing, as no object is hit. At the same time, starting very close to an object might be not feasible having a real-world robot gripper in mind. Therefore, r_{Start} is defined peaking at a goal distance ($d_{Goal} = 0.2m$), while decreasing when moving closer to the object as well as further away from the goal distance:

$$r_{Start}(d_{Object}) = \begin{cases} K_1 * sigmoid(d_{Object}) + K_2 & \text{if } d_{Object} \leq d_{Goal} \\ K_3/d_{Object} + K_4 & \text{otherwise} \end{cases} \tag{4}$$

Additionally, a penalty r_{Inside} is given, if a starting point inside an object is chosen (since this action can not be applied in a real world). Furthermore, if objects are separated successfully, a fixed success reward is added.

$$r_{Inside} = \begin{cases} 1 & (x, y) \text{ is in object} \\ 0 & \text{else} \end{cases} \tag{5}$$

$$r_{success} = \begin{cases} 1 & d_{IO} \geq d_{IO_{goal}} \\ 0 & d_{IO} < d_{IO_{goal}} \end{cases} \tag{6}$$

The total reward is then calculated as shown in Eq. 7. For our experiments we chose $k_{Start} = 300$, $k_{Dist} = 1000$, $k_{Inside} = -100$, $k_{Success} = 1000$. This way, a very high reward is given for successful singulation, a high reward is given for partial singulation, a small reward is given for choosing a close starting point and a small penalty is given for starting inside an object.

$$r_t = r_{Start} * k_{Start} + r_{Dist} * k_{Dist} + r_{Inside} * k_{Inside} + r_{Success} * k_{Success} \tag{7}$$

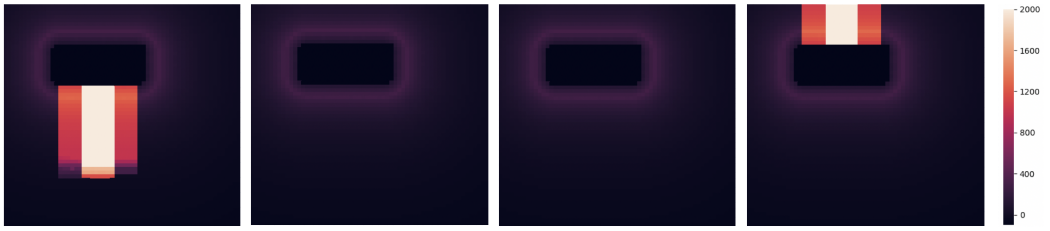

Figure 2: Reward heatmaps for all cardinal pushing directions for scenario shown in Fig. 1, mapped by starting point (from left to right: up, right, left, down)

3.3 Network Architecture

The network used for the results consists of 5 fully connected layers with units [128,64,64,32,16] and ReLU activation after each layer. During the design of the network, it has shown that smaller networks than the proposed could not model the task's complexity.

4 Simulation Setup

For simulation the pybullet simulation engine[1] was used, enabling basic rigid body physics, with the goal to model an interactive object singulation task close to a real-world application. The objects are therefore placed on a table with finite dimensions and observed by a fixed RGBD camera on top of it. The objects that are to be segmented are cubic, oriented in parallel to the table, varying in size and either always placed at the same position repeatedly (*fixed scenario*) or placed randomly (*random scenario*). The actuator is modelled as a small block being able to start anywhere in the scene, position controlled and having a comparably large mass in order to ensure that the determined action is executed as stated. Modeling the scene using simple cubes thereby reduces the problem to a fundamental level and speeds up the training procedure, since a comparably computationally efficient simulation configuration, i.e. a comparably high time step ($= 0.01s$) and low number of

[1] https://pybullet.org

solver iterations ($= 100$), can be used. Before and after executing an action, the camera takes an image (64x64x3), which serves as input for the learning agent as well as for determining the reward, after being pre-processed as described in Section 3. The depth channel is thereby used to determine which pixels correspond to objects and which to the background. In order to represent the real-world scenario as good as possible no further data from the simulation is used for the proposed algorithm.

5 Experimental Results

Since there are no established baselines for a continuous approach to solve object singulation, an approach that was used in the beginning of the project is used to show how the task defined in Section 3 improved learning efficiency. This earlier approach uses simpler representations for state, action and reward. In detail, the following is compared:
The state representation of our approach is compared to a simple binary map. This binary map indicates if a pixel belongs to an object or background. The reward of our approach is compared to another reward using only a difference of inter-object distance reward, i.e. $r_t = r_{Dist_t}$. Finally, we compare the reduced action definition proposed in our approach with a more basic action definition that simply predicts the start and end point of the pushing line, i.e $a = (x_{Start}, y_{Start}, x_{End}, x_{End})$ with $x_{Start}, y_{Start}, x_{End}, x_{End} \in [-l, l]$.

5.1 Experimental Setup

As a reinforcement learning framework we used the Baselines framework from OpenAI [10]. The following parameters were used for PPO training:

Configuration parameter	**Value**
Learning rate	$5e-4$
Mini-Batchsize	4
Batch size	2048
Number of Training epochs per update	3

Table 1: PPO configuration

5.2 Improvement of Learning Efficiency

In the following the impact of the proposed approach is compared to the baseline approach defined above, for a scenario where two connected blocks are always positioned at the same points. Fig. 3 displays a direct comparison between them in terms of their convergence of the reward between the state, action and reward definition (blue: proposed approach, orange: baseline approach).
As described in Section 3 the distance map introduces a bias for good starting point choices, compared to a binary map, which does not differ between pixels close and far away from an object. Consequently, as shown in Fig. 3 a model trained on the distance map converges much faster. The reward as specified in Section 3 is only large, if blocks are separated from each other, touching the blocks while not separating them does not return any (r_{Dist}) reward. Using a starting point and cardinal direction only instead of predicting both the start and end position of the action reduces the number of possible actions but increases the ratio between succeeding and failing actions, since, although there are less possible starting position from which an object can be hit, the cardinal pushing direction (applied on the cubes we use as objects in simulation) guarantees separating the objects as good as possible, whereas

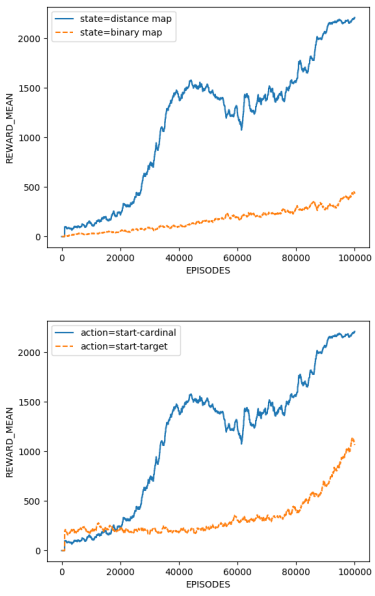

Figure 3: Learning sample-efficiency improvements of the proposed against the baseline task (1: state, 2: action

the start-target-action might lead to hitting but not separating objects.

Fig. 4 compares the evolution of the returned reward for several reward function definitions. In this context *pretrained* refers to a scheduled reward function, returning $r = r_{Start} * k_{Start} + r_{Inside} * k_{Inside}$ only in the first $40 * 10^3$ iterations. As explained in Section 5.3 merely rewarding a successful pushing movement is too sparse to guide to a good policy. Pretraining can improve the sparsity of the reward by introducing a bias to the starting position to be close to an object and can therefore slightly improve the result, however using r_{Start} for all training iterations is necessary to come up with a highly-valued policy, as shown in Fig. 4($r = R_{Dist} + R_{Start}(pretrained)$).

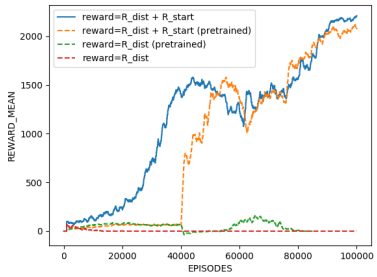

Figure 4: Learning sample-efficiency improvements of the proposed against the baseline task for reward definition

5.3 PPO Unimodality Issue

While a simple fixed scenario with two blocks could be solved using PPO, it failed when it came to scenarios, where blocks are positioned at random positions. We believe that the reason for this lies in the unimodality of the gaussian probability density funtion used in the PPO implementation. In general object singulation tasks, the reward usually is multimodal. This is due to the fact, that the problem has multiple symmetries in most cases. E.g. in the scenario shown in Fig. 6 the agent will receive the same reward if it starts on top of the object and pushes downwards or if it starts below the object and pushes upwards. At the same time, all starting points in between (i.e. points inside the object) will lead to a negative reward, since starting inside the object is not possible. However PPO uses unimodal gaussian policies to capture the reward [8][11].

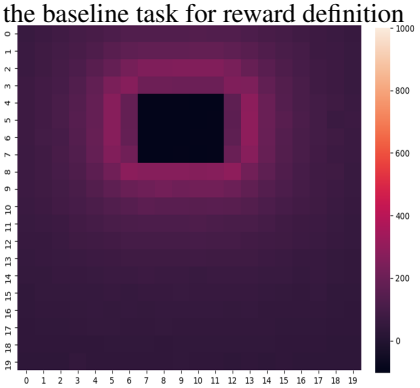

Figure 5: Q-Values for the task explained in Section 5.3 after 4000 steps.

To visualize this behaviour the agent was trained to only spawn close to the object ($r = r_{Start} * k_{Start} + r_{Inside} * k_{Inside}$). As seen in Fig. 6 PPO first tries to capture this multimodal distribution with a unimodal policy with high variance and mean inside the block, which complicates converging to one highly rewarded action.

In a scenario where blocks are placed randomly and also including push directions, this behaviour becomes even more problematic, such that for our object singulation task PPO was not able to solve random scenarios.

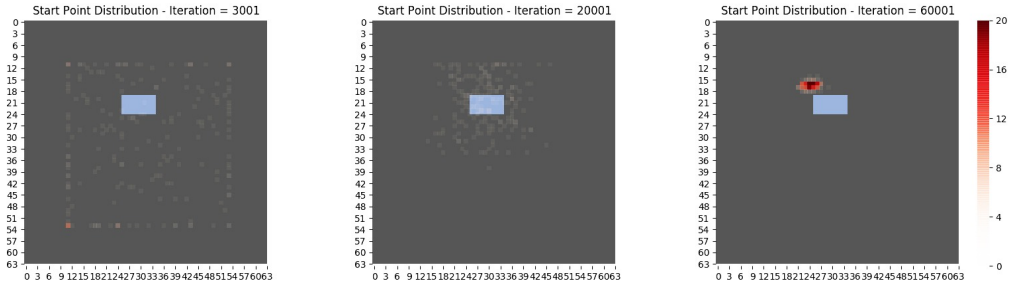

Figure 6: Action start position frequency evolution over 60000 iterations. It can be seen, that the Gaussian is centered at the object after 20000 iterations, while it takes additional 40000 iterations to converge to one specific spot.

5.4 Deep Q-Learning Results

To show, that the task is learnable with the proposed setup, we also applied Deep Q-Learning [12]. The same network, state, and rewards as described in Section 3 were used, while the action's start point had to be discretized in order to make it compatible with Deep Q-Learning. We discretize the workspace in 10 segments in both directions of the plane. This allowed us to solve simple scenarios where 2 or 3 blocks where placed connected at random positions. Videos of those results can be found at https://bit.ly/2EYsDJ4.
It could also be shown that Deep Q-Learning converged much faster for the task that was used in Fig. 6. As shown in the Q-Value visualization below, the Q-Values already converged to a correct representation after 4000 steps, while it took PPO 60000 steps to converge to one point. Next to better performance for the proposed task, DQL is also capable of finding optimal policies in cases in which PPO fails, e.g. the $r = r_{Dist}$ scene. Due to the random exploration, DQL is less likely to get stuck in a cyclic behaviour especially in the beginning of the training. Thus, DQL can find a good policy for the $r = r_{Dist}$ scenario.

6 Conclusion

Within this project the feasibility of applying more continuous, experience-based approaches, specifically PPO, for the object singulation task was examined. Although improvements in terms of sample-efficiency of PPO were made, it was shown that it is merely applicable to simple and fixed scenarios, as it is not able to model multimodal policies, while random scenarios could be solved using DQL.
We have shown that using PPO is not a feasible approach for solving the object singulation task as we defined it, our insights are however transferable to other continuous (advantage-based) reinforcement learning algorithms like TRPO [6], that also use unimodal policy distributions. In future work in order to use a continuous action space in an object singulation task either an approach with a multimodal policy space could be used or the transformation of the task from a multimodel to a unimodal definition could be learned, e.g. by using Fourier transformation parametrizing the multimodality. For learning efficiency of both a continuous and a discrete action the state-space could be downsized to a more dense representation using an autoencoder. While the project focused on simulation only, further work could also apply the proposed method to a real-world scenario, such as a pre-segmentation step.

Acknowledgments

We would like to thank a number of people who have encouraged and helped us in realizing this Semester Project. We appreciate the support we received from all sides.

We want to thank Prof. Dr. Roland Siegwart for offering this course, and the possibility to do a project in this format.

We also appreciate the feedback about the proposal and intermediate report from Dr. Cesar Cadena and Dr. Jen Jen Chung, who also made this project possible at all by allowing us to create a reinforcement learning project together with our supervisors.

Special thanks to our supervisors Fadri Furrer, Tonci Novkovic and Michel Breyer. Their broad knowledge and experiences in reinforcement learning had a wide influence on the results.

Zurich in June 2019,
Simon Schaefer & Patrick Pfreundschuh

References

[1] T. Hermans, J. M. Rehg, and A. F. Bobick. Guided pushing for object singulation. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012*, 2012. URL https://doi.org/10.1109/IROS.2012.6385903.

[2] L. Y. Chang, J. R. Smith, and D. Fox. Interactive singulation of objects from a pile. In *IEEE International Conference on Robotics and Automation, ICRA 2012*, 2012. URL https://doi.org/10.1109/ICRA.2012.6224575.

[3] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. A. Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. *CoRR*, abs/1803.09956, 2018. URL http://arxiv.org/abs/1803.09956.

[4] A. Eitel, N. Hauff, and W. Burgard. Learning to singulate objects using a push proposal network. *CoRR*, abs/1707.08101, 2017. URL http://arxiv.org/abs/1707.08101.

[5] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015. URL http://dblp.uni-trier.de/db/journals/corr/corr1509.html#LillicrapHPHETS15.

[6] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. *CoRR*, abs/1502.05477, 2015. URL http://arxiv.org/abs/1502.05477.

[7] 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. *CoRR*, abs/1602.01783, 2016. URL http://arxiv.org/abs/1602.01783.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[9] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. *CoRR*, abs/1709.06560, 2017. URL http://arxiv.org/abs/1709.06560.

[10] 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.

[11] Y. Tang and S. Agrawal. Implicit Policy for Reinforcement Learning. jun 2018. URL http://arxiv.org/abs/1806.06798.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013. URL http://arxiv.org/abs/1312.5602.