An Attention Mechanism for Deep Q-Networks with Applications in Robotic Pushing

Marco Ewerton, Sylvain Calinon, Jean-Marc Odobez

Abstract— **Recent work has used Deep Q-Networks (DQNs) to address manipulation tasks involving pushing. While successful in general, these DQNs and other deep models have been observed to sometimes get stuck in local optima when dealing with large state-action spaces, repeating for instance the same ineffective action. We address this issue by proposing a mechanism for DQNs to attend only to actions that matter. Simulation experiments show that this mechanism helps the DQN learn faster and achieve higher performance in a push task involving objects with unknown dynamics.**

I. Introduction and Related Work

Pushing is a key motor skill that can help to solve many manipulation tasks by isolating objects or reorienting them to improve their grasping, bringing them closer, or putting them into a container. Pushing can also aid perception by improving object segmentation. It is interesting to note that 25 out of the 50 manipulation tasks of the Meta-World benchmark involve pushing [1].

In this view, Zeng et al. [2] proposed an interesting model-free deep reinforcement learning to synergistically perform pushing and grasping. However, we and other researchers [3] have observed that deep models can get stuck in a loop, predicting actions that do not cause any change in the actual scene, and also spend a large amount of training time only to learn how to perform basic pushes that move objects. To address these two issues, our main contribution is to design a simple yet effective attention mechanism based on visual processing for Deep Q-Networks (DQNs) that improves their sampling efficiency by constraining the set of possible actions to actually lead to changes in the environment. Such an approach could be applied to any manipulation tasks which involve pushing. Note that here, by attention mechanism, we mean that our system *pays attention* only to actions that can at least change the environment.

Other approaches to manipulation have used, for example, Logic-Geometric Programming [4], Model Predictive Control [5], Recurrent Neural Networks [6] or Switch-Linear Models [3]. However, differently from our work, none of these works have dealt with a pile of different 3D objects with unknown dynamics.

The authors are with the Idiap Research Institute, CH-1920, Martigny, Switzerland {marco.ewerton, sylvain.calinon, jean-marc.odobez}@idiap.ch

The research leading to these results has received funding from the Swiss National Science Foundation through the HEAP project (Human-Guided Learning and Benchmarking of Robotic Heap Sorting, ERA-net CHIST-ERA).

II. Pushing Task and Learning Architecture

Fig. 1a depicts the task studied in this paper. The objective is to make the robot push all the objects on the tabletop into the transparent box. A camera positioned in front of the robot captures RGB-D images from the workspace. The orthogonal projections of these images are fed as inputs to the learning architecture (Fig. 1b). The robot can push at the positions corresponding to any of the 224×224 pixels on the image. The pushing actions can be performed in 16 different orientations and are 10 cm long.

Learning approach. We have adopted the DQN approach of [2], and have defined the reward for taking action a_t at state s_t and transitioning to state s_{t+1} as:

$$R_{a_t}(s_t, s_{t+1}) = \begin{cases} 0 \ \text{ if no change or object falls to ground,} \\ \max(0, \bar{d}_t - \bar{d}_{t+1}) \ \text{ if change,} \\ +10 \ \text{ if object falls into box,} \end{cases}$$

where a change is detected when there is a large enough difference between the images before and after the push. The term $\bar{d}_t = \frac{1}{N_t}\sum_{p_i \in O_t} d(p_i, p_{\text{tbox}})$ is the average distance between the $N_t = |O_t|$ pixels p_i belonging to objects at time t and p_{tbox} denoting the target position of the box we want to push objects into. The set O_t of object pixels is obtained from the depth image by identifying all pixels which are above the table plane. Hence, the reward due to change will be positive if $\bar{d}_t - \bar{d}_{t+1} > 0$, that is, if after the push the average distance between object pixels and the target (represented by $\bar{d}_{t+1}$) is shorter than before the push (represented by $\bar{d}_t$).

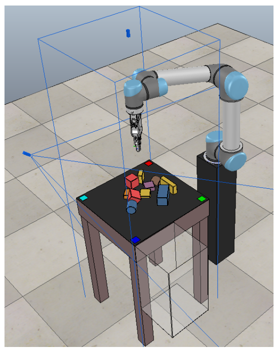
Learning system Workflow. It is depicted in Fig. 1b. The parameters of the DenseNet [7] are optimized via back-propagation to minimize the temporal difference error $\delta_t = |Q^*(s_t, a_t) - y_t|$ between the predicted Q-value $Q^*(s_t, a_t)$ and a target value

$$y_t = R_{a_t}(s_t, s_{t+1}) + \gamma Q^*\left(s_{t+1}, \arg\max_{a'}\left(Q^*(s_{t+1}, a')\right)\right),$$

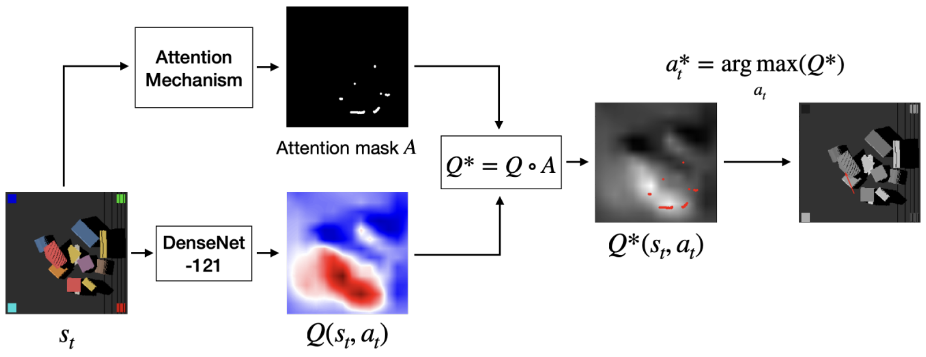
where γ is the discount factor and a' is an action in the set of all available actions. During training, the actions are sampled according to an ϵ-greedy approach with $\epsilon = 0.1$. During test, the selected actions are the ones that maximize Q^*.

III. Attention Mechanism

We propose an attention mechanism that improves sampling efficiency, reduces training time, and selects better actions. It works by generating a mask A which is applied to the output Q of the DenseNet such that the output effectively

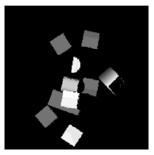


(a)

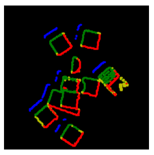


(b)

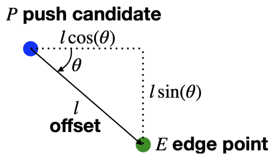
Fig. 1. (a) Pushing into the box task. (b) Workflow of our learning system. A DenseNet predicts the state-action values $Q(s_t, a_t)$ of pushing at each pixel with a certain angle while our attention mechanism based on visual processing identifies push start position candidates that lead to moving objects. The mask A provided by this attention mechanism is combined via the Hadamard product with Q to produce the state-action values Q^*. The optimal push a_t^* is the one that maximizes Q^*. The attention mask A prevents trying pushes that do not lead to any changes, speeding up the learning process and reducing the chances of choosing ineffective actions at test time.



(a)



(b)



(c)

Fig. 2. (a) Depth heightmap. (b) The green points correspond to edges that can be pushed from a given angle, 45 degrees in this case. The blue points correspond to suitable pushing start positions given that the push should start at a distance l from the edges. (c) Geometrical relation between an edge point E and a suitable start pushing position candidate P.

used to sample actions becomes $Q^* = Q \circ A$, where $\circ$ stands for the Hadamard product. The mask A has a value 1 for each pixel and orientation corresponding to a suitable push candidate and 0 otherwise. It is obtained by applying a Canny edge detection algorithm to the input image followed by geometrical considerations about the task (see Fig. 2). It aims at keeping only effective pushes and eliminating the large set of obvious actions which will not affect the scene.

IV. Experimental Results and Conclusion

Experiments. We performed experiments in simulation using a UR5 robot arm performing the task depicted in Fig. 1a. Our simulation involved eight different object shapes (including cylinders, which can roll on the table) and ten different object colors, as in [2]. Training and test scenarios were generated by sampling object shapes and colors at random and letting the objects fall around the center of the table. The training and test procedures moved from one stage to the next if no objects were remaining on the tabletop or if five pushes were performed without any change.

Results. Tests were conducted using 200 pregenerated scenarios: 100 with one and 100 with ten objects. Table I shows that the model using the attention mechanism consistently outperformed the model without it. It learns more accurate pushes, on average with more objects pushed into the box and far fewer objects remaining on the table (results are nearly perfect with only one object), and is more efficient, with far fewer actions needed per object in the box, especially when there were more objects in the environment. Note that both models have been trained for the same number of iterations, and have achieved convergence during training.

Total # objects	1		10	
Model	with mask	no mask	with mask	no mask
# objects in the box $\mu(\sigma)$	0.99 (0.10)	0.92 (0.27)	7.39 (1.50)	4.20 (2.79)
# objects on the ground $\mu(\sigma)$	0.01 (0.10)	0.07 (0.26)	2.47 (1.29)	1.63 (1.43)
# objects left on the table $\mu(\sigma)$	0.00 (0.00)	0.01 (0.10)	0.14 (1.01)	4.17 (3.38)
# actions $\mu(\sigma)$	3.19 (0.96)	3.67 (1.18)	19.00 (6.57)	21.32 (6.66)
$\frac{\text{avg. \# objects in the box}}{\text{avg. \# actions}}$	0.31	0.25	0.39	0.20

TABLE I
Results starting from 1 or 10 randomly fallen objects.

Conclusion. We proposed an attention mechanism based on visual processing for DQNs. In particular, we demonstrated that this mechanism helps a DQN to learn a push task faster and to achieve better performance. It improves the sampling efficiency of the DQN by constraining the set of possible actions to pushes that lead to changes in the environment.

References

[1] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Conf. on Robot Learning*, 2020.
[2] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *IROS*. IEEE, 2018, pp. 4238–4245.
[3] H. Suh and R. Tedrake, "The surprising effectiveness of linear models for visual foresight in object pile manipulation," *Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2020.
[4] M. Toussaint, K. Allen, K. Smith, and J. Tenenbaum, "Differentiable physics and stable modes for tool-use and manipulation planning." in *RSS*, 2018.
[5] F. R. Hogan and A. Rodriguez, "Feedback control of the pusher-slider system: A story of hybrid and underactuated contact dynamics," in *Algorithmic Foundations of Robotics XII*, 2020, pp. 800–815.
[6] A. Ajay, J. Wu, N. Fazeli, M. Bauza, L. P. Kaelbling, J. B. Tenenbaum, and A. Rodriguez, "Augmenting physical simulators with stochastic neural networks: Case study of planar pushing and bouncing," in *IROS*. IEEE, 2018, pp. 3066–3073.
[7] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017, pp. 4700–4708.